aws

Guia do Desenvolvedor

AWS Step Functions

AWS Step Functions: Guia do Desenvolvedor

Table of Contents

O que AWS Step Functionsé

AWS Step Functions é um serviço de fluxo de trabalho visual que ajuda você a criar aplicativos distribuídos, automatizar processos, orquestrar microsserviços e criar pipelines de dados e aprendizado de máquina (ML).

No console gráfico do Step Functions, você pode ver o fluxo de trabalho do seu aplicativo como uma série de etapas orientadas por eventos.

O Step Functions é baseado em máquinas e tarefas de estado. Em Step Functions, as máquinas de estado são chamadas de fluxos de trabalho, que são uma série de etapas orientadas por eventos. Cada etapa em um fluxo de trabalho é chamada de estado. Por exemplo, um estado de tarefa representa uma unidade de trabalho que outro AWS serviço executa, como chamar outro AWS service (Serviço da AWS) ou uma API.

Com os controles integrados do Step Functions, você pode examinar o estado de cada etapa em seu fluxo de trabalho para garantir que seu aplicativo seja executado em ordem e conforme o esperado. Dependendo do seu caso de uso, você pode fazer com que os AWS serviços de chamadas do Step Functions, como o Lambda, executem tarefas. Você pode criar fluxos de trabalho que processam e publicam modelos de machine learning. Você pode ter AWS serviços de controle do Step Functions AWS Glue, como criar fluxos de trabalho de extração, transformação e carregamento (ETL). Além disso, você tem a capacidade de criar fluxos de trabalho automatizados e de longa duração para aplicativos que exigem interação humana.

> **Tip**
>
> Para aprender a usar o Step Functions, siga os módulos interativos no AWS Step Functions Workshop ou leia a seção Getting Started neste guia para criar um fluxo de trabalho de solicitação de cartão de crédito.

Tópicos

- Regiões com suporte
- É a primeira vez que você usa o Step Functions?

AWS SDK e integrações otimizadas

Para ligar para outros AWS serviços, você pode usar as integrações AWS SDK do Step Functions ou usar uma das integrações otimizadas do Step Functions.

- As integrações do AWS SDK permitem que você chame qualquer um dos mais de duzentos AWS serviços diretamente de sua máquina de estado, dando acesso a mais de nove mil ações de API.
- As integrações otimizadas do Step Functions foram personalizadas para simplificar a utilização nas máquinas de estado.

Fluxos de trabalho Padrão e Expressos

O Step Functions tem dois tipos de fluxo de trabalho. Os fluxos de trabalho padrão têm execução de fluxo de trabalho única e podem durar até um ano. Isso significa que cada etapa do fluxo de trabalho padrão será executada somente uma vez. Os fluxos de trabalho expressos, no entanto, têm execução at-least-once de fluxo de trabalho e podem ser executados por até cinco minutos. Isso significa que uma ou mais etapas em um fluxo de trabalho expresso podem ser executadas mais de uma vez, enquanto cada etapa do fluxo de trabalho é executada pelo menos uma vez.

As execuções são instâncias em que o fluxo de trabalho é executado para a realização de tarefas. Os fluxos de trabalho padrão são ideais para fluxos de trabalho auditáveis e de longa duração, pois mostram o histórico de execução e a depuração visual. Os fluxos de trabalho expressos são ideais para high-event-rate cargas de trabalho, como processamento de dados de streaming e ingestão de dados de IoT.

Especificações de fluxos de trabalho padrão

- Taxa de execução de 2 mil por segundo
- Taxa de transição de estado de 4 mil por segundo
- Cobrança realizada por transição de estado
- Exibição do histórico de execução e depuração visual
- Compatibilidade com todas as integrações e padrões de serviço

Especificações de fluxos de trabalho expressos

- Taxa de execução de cem mil por segundo
- Taxa de transição de estado quase ilimitada
- Cobrança realizada conforme o número e duração das execuções
- Enviar histórico de execução para a Amazon CloudWatch
- Exibição do histórico de execução e depuração visual com base no nível de log ativo
- Compatibilidade com a todas as integrações e a maioria dos padrões de serviço

Para obter mais informações sobre os fluxos de trabalho padrão e expresso, incluindo preços do Step Functions, consulte:

- Comparação entre os fluxos de trabalho padrão e expresso
- Definição de preços do AWS Step Functions

Casos de uso

O Step Functions gerencia os componentes e a lógica do aplicativo para que você possa escrever menos código e se concentrar na criação e atualização rápida de aplicativos. Esta seção descreve casos de uso comuns para trabalhar com o Step Functions.

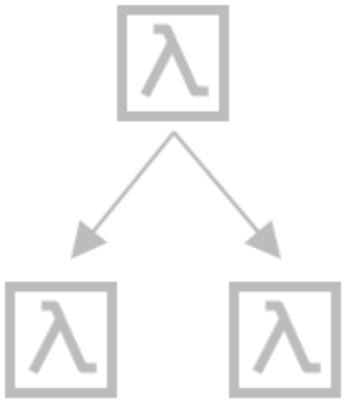
Caso de uso 1: Orquestração de funções

Crie um fluxo de trabalho que executa um grupo de funções (etapas) do Lambda em uma ordem específica. A saída de uma função do Lambda é transferida para a próxima entrada da função do Lambda. A última etapa do fluxo de trabalho apresenta o resultado. Com o Step Functions, é possível ver como cada etapa do fluxo de trabalho interage entre si, assim você pode garantir que cada etapa execute a função pretendida.

Para obter um tutorial que mostra como criar uma máquina de estado com um grupo de funções, consulte:

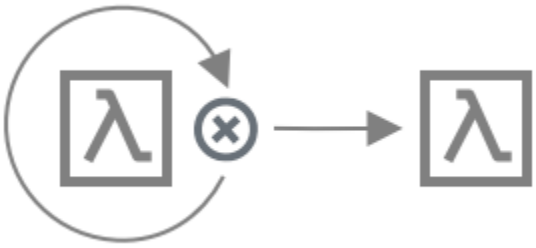
- Começando com AWS Step Functions

Caso de uso 2: Ramificação



O cliente solicita um aumento do limite de crédito. Usando o estado `Choice`, faça o Step Functions tomar decisões com base na entrada do estado `Choice`. Se a solicitação for maior do que o limite de crédito pré-aprovado para o cliente, faça o Step Functions enviar a solicitação do cliente para a aprovação do gerente. Se a solicitação for menor do que o limite de crédito pré-aprovado para o cliente, faça o Step Functions aprovar automaticamente a solicitação.

Caso de uso 3: Tratamento de erros



`Retry`

Nesse caso de uso, o cliente solicita um nome de usuário. Devido a um erro, a solicitação do cliente não foi concluída na primeira tentativa. Usando a instrução `Retry`, faça o Step Functions repetir a solicitação do cliente. Na segunda tentativa, a solicitação do cliente é concluída com êxito.

`Catch`

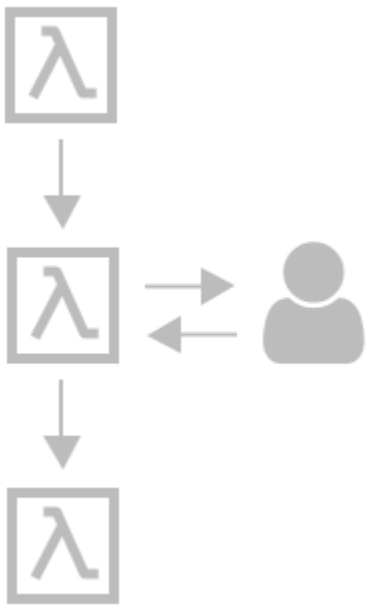
Em um caso de uso semelhante, o cliente solicita um nome de usuário indisponível. Usando a instrução `Catch`, faça o Step Functions sugerir um nome de usuário disponível. Se o cliente usar o

nome de usuário disponível, faça o Step Functions seguir para a próxima etapa do fluxo de trabalho, que é enviar um e-mail de confirmação. Se o cliente não usar o nome de usuário disponível, faça o Step Functions seguir para uma etapa diferente do fluxo de trabalho, que é reiniciar o processo de cadastro.

Para obter exemplos mais detalhados sobre as instruções `Retry` e `Catch`, consulte:

- Tratamento de erros no Step Functions

Caso de uso 4: Notificar pessoas

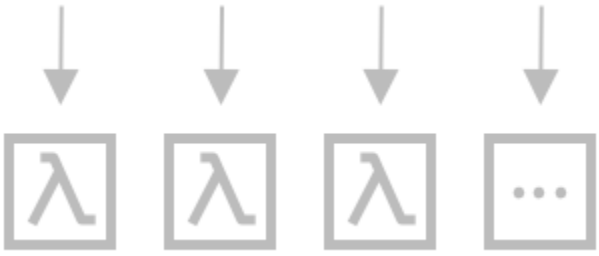


Usando um aplicativo bancário, um dos clientes envia dinheiro para um amigo. O cliente espera por um e-mail de confirmação. Com um retorno de chamada e um token de tarefa, faça o Step Functions dizer ao Lambda para enviar o dinheiro do cliente e informar quando o amigo do cliente o receber. Depois que o Lambda informar que o amigo do cliente recebeu o dinheiro, faça o Step Functions seguir para a próxima etapa do fluxo de trabalho, que é enviar ao cliente um e-mail de confirmação.

Para ver um exemplo de projeto que mostra o retorno de chamada com um token de tarefa, consulte o seguinte:

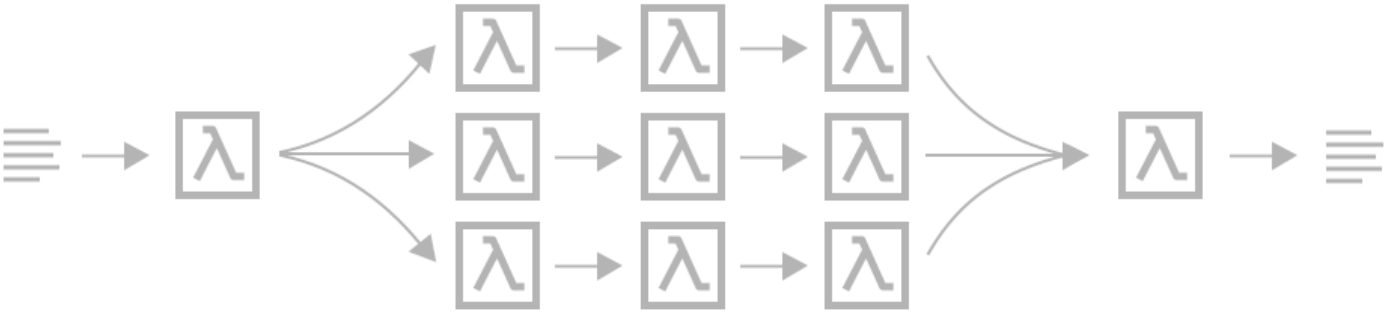
- Exemplo de Padrão de Retorno de Chamada (Amazon SQS, Amazon SNS, Lambda)

Caso de uso 5: Processamento paralelo



O cliente converte um arquivo de vídeo em cinco resoluções de exibição diferentes, para que os espectadores possam assistir ao vídeo em vários dispositivos. Usando o estado `Parallel`, o Step Functions insere o arquivo de vídeo para que o Lambda possa processá-lo nas cinco resoluções de exibição ao mesmo tempo.

Caso de uso 6: Paralelismo dinâmico



O cliente pede três itens e você precisa preparar cada um deles para a entrega. Você verifica a disponibilidade de cada item, os separa e, em seguida, embala cada item para a entrega. Usando o estado `Map`, o Step Functions faz o Lambda processar cada um dos itens do cliente paralelamente. Depois que todos os itens do cliente forem embalados para entrega, o Step Functions segue para a próxima etapa do fluxo de trabalho, que é enviar ao cliente um e-mail de confirmação com as informações de rastreamento.

Para ver um exemplo de projeto que mostra o paralelismo dinâmico usando o estado `Map`, consulte:

- Processar dados dinamicamente com um estado Mapa

Integrações de serviços

O Step Functions se integra a vários AWS serviços. Para combinar o Step Functions com esses serviços, use os padrões de integração de serviços a seguir.

Solicitar resposta (padrão)

- Chame um serviço e faça o Step Functions avançar para o próximo estado assim que obter uma resposta HTTP.

Executar tarefa (.sync)

- Chame um serviço e faça o Step Functions aguardar a conclusão de uma tarefa.

Aguarde um retorno de chamada com um token de tarefa (. waitForTaskSímbolo)

- Chame um serviço com um token de tarefa e faça o Step Functions esperar até que o token de tarefa responda com um retorno de chamada.

A tabela abaixo mostra as integrações de serviços e os padrões de integração de serviços disponíveis para o Step Functions.

Os fluxos de trabalho padrão e os fluxos de trabalho expressos oferecem suporte às mesmas integrações, mas não aos mesmos padrões de integração.

- O suporte ao padrão de integrações otimizadas é diferente para cada integração.
- Os fluxos de trabalho expressos não oferecem suporte a Run a Job (.sync) nem a Wait for Callback (). waitForTaskSímbolo).
- Para ter mais informações, consulte Comparação entre os fluxos de trabalho padrão e expresso.

Standard Workflows

Integrações de serviços compatíveis

	Serviço	Resposta de solicitação	Executar um trabalho (`.sync`)	Aguardar o retorno de chamada (`.waitForTaskToken`)
Integrações otimizadas	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	✓		✓
	AWS Elemental MediaConvert	✓	✓	

	Serviço	Resposta de solicitação	Executar um trabalho (`.sync`)	Aguardar o retorno de chamada (`.waitForTaskToken`)
	Amazon SageMaker	✓	✓	
	Amazon SNS	✓		✓
	Amazon SQS	✓		✓
	AWS Step Functions	✓	✓	✓
AWS Integrações de SDK	Mais de duzentas	✓		✓

Express Workflows

Integrações de serviços compatíveis

	Serviço	Resposta de solicitação	Executar um trabalho (`.sync`)	Aguardar o retorno de chamada (`.waitForTaskToken`)
Integrações otimizadas	Amazon API Gateway	✓		
	Amazon Athena	✓		
	AWS Batch	✓		
	Amazon Bedrock	✓		

	Serviço	Resposta de solicitação	Executar um trabalho (`.sync`)	Aguardar o retorno de chamada (`.waitForTaskToken`)
	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	✓		
	AWS Step Functions	✓		

	Serviço	Resposta de solicitação	Executar um trabalho (`.sync`)	Aguardar o retorno de chamada (`.waitForTaskToken`)
AWS Integrações de SDK	Mais de duzentas	✓		

Regiões com suporte

A maioria das AWS regiões oferece suporte a Step Functions. Para obter uma lista completa das AWS regiões em que o Step Functions está disponível, consulte a Tabela de AWS regiões.

É a primeira vez que você usa o Step Functions?

Se esta é sua primeira vez usando o Step Functions, os tópicos a seguir ajudarão você a entender diferentes partes do trabalho com o Step Functions, incluindo como o Step Functions se combina com outros AWS serviços:

- Tutoriais do Step Functions
- Projetos de amostra para Step Functions
- AWS Step Functions SDK de ciência de dados para Python

Começando com AWS Step Functions

O Step Functions é um serviço de orquestração de tecnologia sem servidor que permite definir um fluxo de trabalho da aplicação como uma série de etapas orientadas a eventos. Cada etapa no fluxo de trabalho é chamada de estado. Você geralmente usa estados comuns, como Estado da tarefa, Choice, Paralelo e Mapa, para definir fluxos de trabalho. Dentro `Task` dos estados, você pode usar as integrações de AWS SDK suportadas pelo Step Functions e orquestrar várias Serviços da AWS em seus fluxos de trabalho.

Tópicos

Principais conceitos

Antes de começar os tutoriais, revise os seguintes termos-chave do Step Functions para contextualizar.

Prazo	Descrição
Fluxo de trabalho	Uma sequência de etapas que geralmente refletem um processo de negócios.

Prazo	Descrição
Estados	Etapas individuais em sua máquina de estado que podem tomar decisões com base em suas entradas, executar ações a partir dessas entradas e passar a saída para outros estados. Para ter mais informações, consulte States.
Workflow Studio	Um designer de fluxo de trabalho visual que ajuda você a criar protótipos e criar fluxos de trabalho com mais rapidez. Para ter mais informações, consulte AWS Step Functions Studio de fluxo de trabalho.
Máquina de estado	Um fluxo de trabalho definido usando texto JSON que representa os estados ou etapas individuais no fluxo de trabalho junto com campos `StartAt`, `TimeoutSeconds` e `Version`. Para ter mais informações, consulte Estrutura da máquina de estado.
Amazon States Language	Uma linguagem estruturada baseada em JSON usada para definir suas máquinas de estado. Com o ASL, você define uma coleção de estados que podem funcionar (`Task`estado), determina quais estados fazer a transição para o próximo (`Choice`estado) e interrompe uma execução com um erro (`Fail`estado). Para ter mais informações, consulte Amazon States Language.
Configuração de entrada e saída	Os estados em um fluxo de trabalho recebem dados JSON como entrada e geralmente passam dados JSON como saída para o próximo estado. Step Functions fornece filtros para controlar o fluxo de dados entre os estados. Para ter mais informações, consulte Processamento de entrada e saída no Step Functions.
Integração de serviços	Você pode chamar ações AWS de API de serviço a partir do seu fluxo de trabalho. Para ter mais informações, consulte Usando AWS Step Functions com outros serviços.

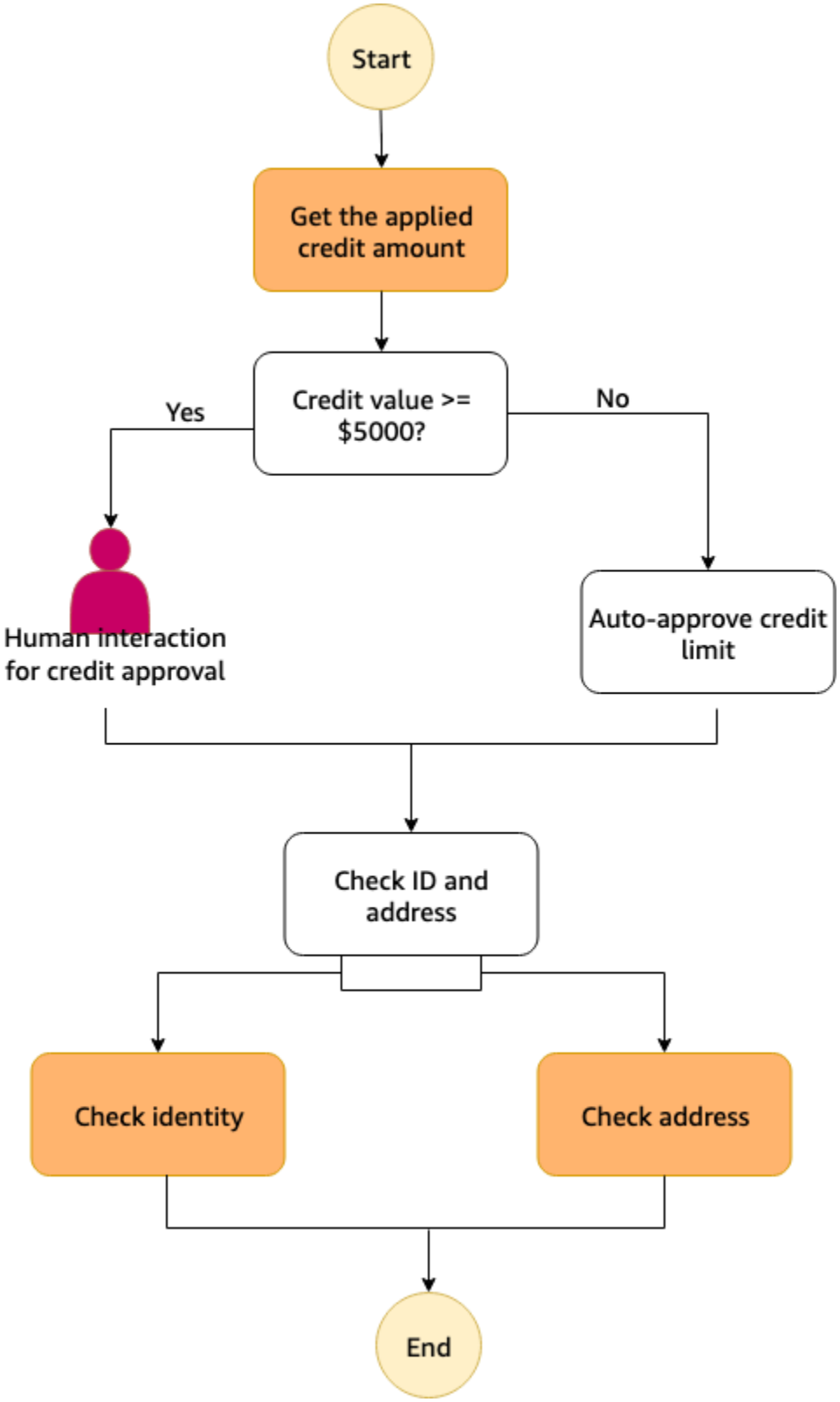
Prazo	Descrição
Tipo de integração de serviços	• AWS Integrações de SDK — forma padrão de chamar qualquer uma das mais de duzentas Serviços da AWS e mais de nove mil ações de API diretamente de sua máquina de estado. • Integrações otimizadas — integrações personalizadas que simplificam as chamadas e a troca de dados com determinados serviços. Por exemplo, o Lambda Invoke converterá automaticamente o `Payload` campo da resposta de uma string JSON de escape em um objeto JSON.
Padrão de integração de serviço	Ao chamar um AWS service (Serviço da AWS), você usa um dos seguintes padrões de integração de serviços: • Solicitar uma resposta (padrão) — Chame um serviço e passe para o próximo estado imediatamente após receber uma resposta HTTP. • Executar um trabalho (.sync) — Chame um serviço e faça com que o Step Functions aguarde a conclusão de um trabalho. • Aguarde um retorno de chamada com um token de tarefa (.wait ForTask Token) — Chame um serviço com um token de tarefa e faça com que o Step Functions espere até que o token de tarefa retorne com um retorno de chamada.
Execução	As execuções de máquinas de estado são instâncias em que o fluxo de trabalho é executado para a realização de tarefas. Para ter mais informações, consulte Execuções no Step Functions.

Tutoriais desta série

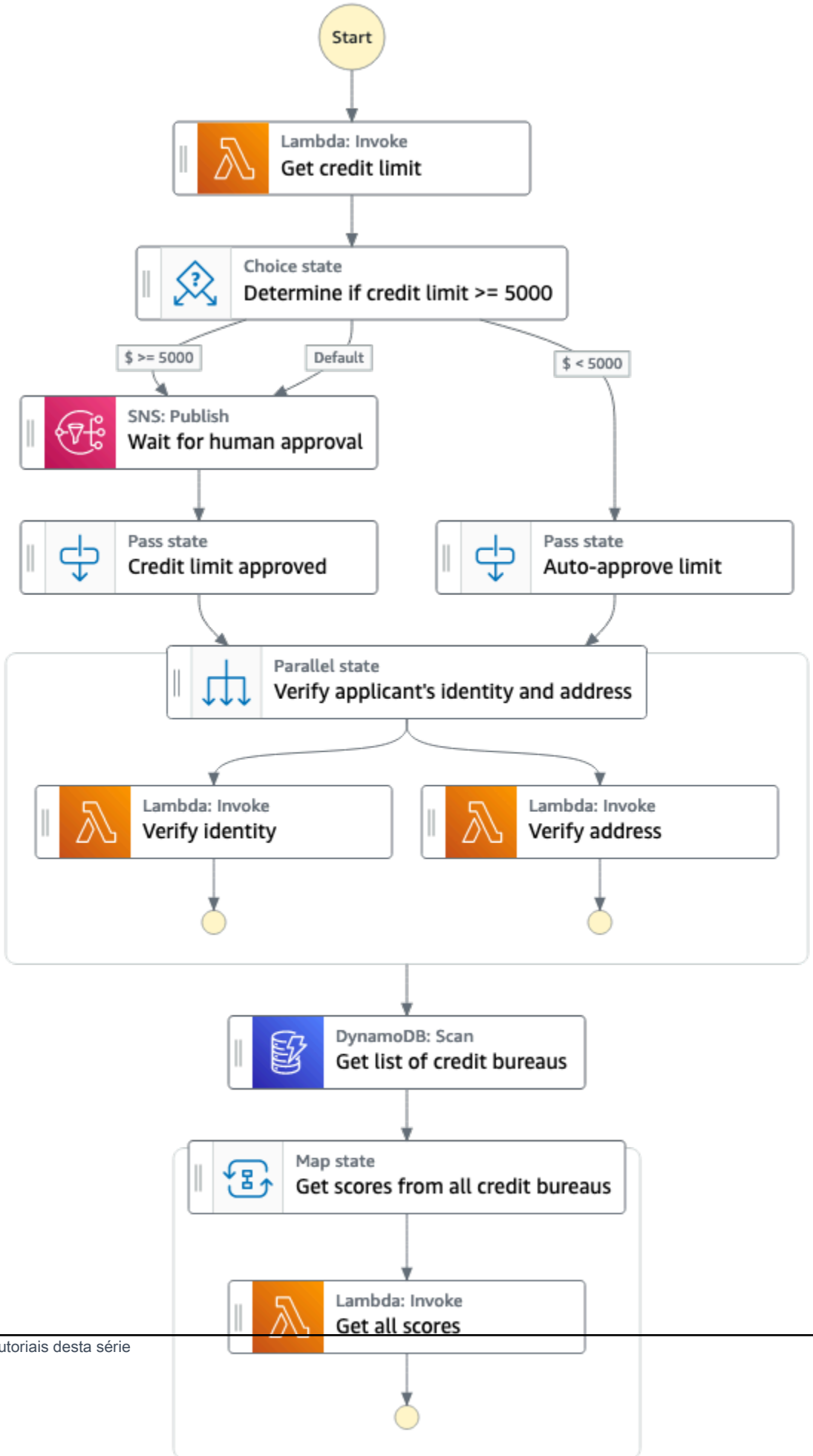
Depois de concluir esses tutoriais, você terá um fluxo de trabalho que simula o processamento de um pedido de cartão de crédito. Você aprenderá a usar estados comuns e integrar seu fluxo de trabalho com outros Serviços da AWS

O Step Functions pode ser usado para criar vários tipos de fluxos de trabalho, como processamento de dados, automação de TI, aprendizado de máquina e codificação de mídia.

O fluxograma a seguir mostra as etapas para uma empresa processar uma solicitação de cartão de crédito. Se o valor do crédito solicitado for inferior a $5000, o limite de crédito será aprovado automaticamente. Se a solicitação exceder o limite, o fluxo de trabalho adicionará um humano ao loop para verificar a identidade do solicitante e revisar as pontuações de crédito.

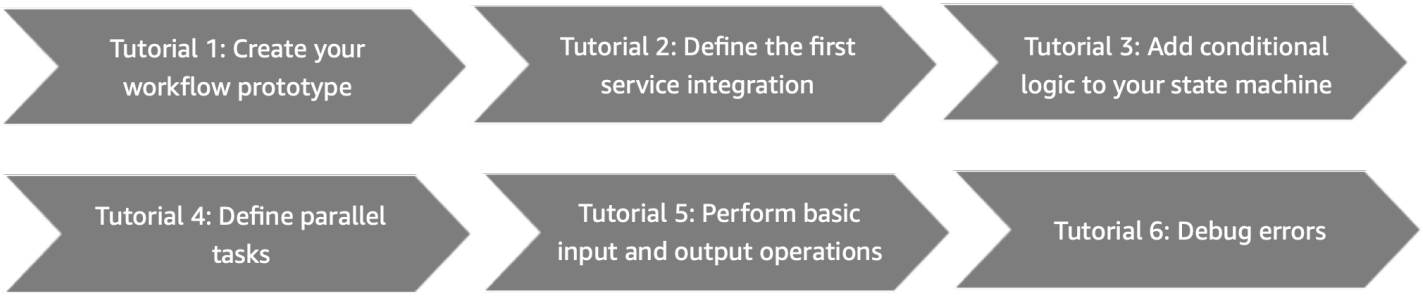


O diagrama a seguir mostra como as etapas do processo comercial do pedido de crédito são representadas por estados em um fluxo de trabalho do Step Functions.


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

Na série de tutoriais a seguir, você criará o fluxo de trabalho de processamento de cartões de crédito.

Recomendamos concluir esses tutoriais para aprender os principais recursos do Step Functions.



Antes de começar, certifique-se de completar os pré-requisitos.

Pré-requisitos para começar a usar AWS Step Functions

Antes de usar AWS Step Functions pela primeira vez, conclua as tarefas a seguir.

Inscreva-se para um Conta da AWS

Se você não tiver um Conta da AWS, conclua as etapas a seguir para criar um.

Para se inscrever em um Conta da AWS

1. Abra https://portal.aws.amazon.com/billing/signup.
2. Siga as instruções online.

   Parte do procedimento de inscrição envolve receber uma chamada telefônica e inserir um código de verificação no teclado do telefone.

   Quando você se inscreve em um Conta da AWS, um Usuário raiz da conta da AWSé criado. O usuário raiz tem acesso a todos os Serviços da AWS e atributos na conta. Como prática recomendada de segurança, atribua o acesso administrativo a um usuário e use somente o usuário-raiz para executar tarefas que exigem acesso de usuário-raiz.

AWS envia um e-mail de confirmação após a conclusão do processo de inscrição. A qualquer momento, é possível visualizar as atividades da conta atual e gerenciar sua conta acessando https://aws.amazon.com/ e selecionando Minha conta.

Criar um usuário com acesso administrativo

Depois de se inscrever em um Conta da AWS, proteja seu Usuário raiz da conta da AWS AWS IAM Identity Center, habilite e crie um usuário administrativo para que você não use o usuário root nas tarefas diárias.

Proteja seu Usuário raiz da conta da AWS

1. Faça login AWS Management Consolecomo proprietário da conta escolhendo Usuário raiz e inserindo seu endereço de Conta da AWS e-mail. Na próxima página, insira sua senha.

   Para obter ajuda ao fazer login usando o usuário raiz, consulte Fazer login como usuário raiz no Guia do usuário do Início de Sessão da AWS .
2. Habilite a autenticação multifator (MFA) para o usuário raiz.

   Para obter instruções, consulte Habilitar um dispositivo de MFA virtual para seu usuário Conta da AWS raiz (console) no Guia do usuário do IAM.

Criar um usuário com acesso administrativo

1. Habilitar o IAM Identity Center.

   Para obter instruções, consulte Habilitar AWS IAM Identity Center no Guia do usuário do AWS IAM Identity Center .
2. No Centro de Identidade do IAM, conceda o acesso administrativo para um usuário.

   Para ver um tutorial sobre como usar o Diretório do Centro de Identidade do IAM como fonte de identidade, consulte Configurar o acesso do usuário com o padrão Diretório do Centro de Identidade do IAM no Guia AWS IAM Identity Center do usuário.

Iniciar sessão como o usuário com acesso administrativo

- Para fazer login com seu usuário do Centro de Identidade do IAM, use o URL de login que foi enviado ao seu endereço de e-mail quando você criou o usuário do Centro do Usuário do IAM.

  Para obter ajuda para fazer login usando um usuário do IAM Identity Center, consulte Como fazer login no portal de AWS acesso no Guia Início de Sessão da AWS do usuário.

Atribuir acesso a usuários adicionais

1. No Centro de Identidade do IAM, crie um conjunto de permissões que siga as práticas recomendadas de aplicação de permissões com privilégio mínimo.

   Para obter instruções, consulte Create a permission set no Guia do usuário do AWS IAM Identity Center .

2. Atribua usuários a um grupo e, em seguida, atribua o acesso de autenticação única ao grupo.

   Para obter instruções, consulte Add groups no Guia do usuário do AWS IAM Identity Center .

Tutorial 1: Criar o protótipo para sua máquina de estado

Neste tutorial, você cria o protótipo para seu fluxo de trabalho de processamento de cartão de crédito usando o Workflow Studio do Step Functions. Você vai escolher as ações e os estados de API necessários na guia Ações eFluxo e usar o atributo de arrastar e soltar do Workflow Studio para criar o protótipo do fluxo de trabalho. Nos tutoriais subsequentes, você vai aprender a configurar os Serviços da AWS e os estados do Step Functions que vai usar neste fluxo de trabalho.

Para criar um protótipo da máquina de estado

1. Abra o console do Step Functions e clique em Criar máquina de estado.
2. Na caixa de diálogo Escolher um modelo, selecione Em branco.
3. Escolha Selecionar. Isso abre o Workflow Studio no Modo de design.
4. No Workflow Studio, na guia Ações, arraste uma ação da API Invocação AWS Lambda e solte-a no estado vazio chamado Drag first state here (Arraste o primeiro estado para cá). Configure desta forma:

   - Na guia Configuração, em Nome do estado, insira **`Get credit limit`**.

5. Arraste a guia Fluxo, arraste e solte o estado Escolha abaixo do estado Get credit limit (Obter limite de crédito). Altere o nome do estado de Escolha para **`Credit applied >= 5000?`**.
6. Arraste e solte os seguintes estados como ramificações do estado Credit applied >= 5000? (Crédito aplicado >= 5.000?).

   a. Amazon SNS Publish: na guia Ações, arraste e solte a ação da API Amazon SNS Publish. Altere esse estado para **`Wait for human approval`**.

   b. Estado Aprovado: na guia Fluxo, arraste e solte o estado Aprovado. Altere o nome dessa ramificação para **`Auto-approve limit`**.

7. Arraste e solte um estado Aprovado abaixo do estado Aguardar aprovação humana. Altere o nome do estado Aprovado para **`Credit limit approved`**.
8. Arraste e solte um estado Em paralelo após o estado Escolha da seguinte forma:

   a. Largue o estado Em paralelo após o estado Credit limit approved (Limite de crédito aprovado).

   b. Altere o nome do estado Em paralelo para **`Verify applicant's identity and address`**.

   c. Em ambas as ramificações do estado Em paralelo, arraste e solte duas ações da API Invocação AWS Lambda.

   d. Altere o nome desses estados como **`Verify identity`** e **`Verify address`**, respectivamente.

   e. Escolha o estado Auto-approve limit (Limite de aprovação automática) e, em Próximo estado, selecione Verify applicant's identity and address (Verificar a identidade e o endereço do candidato).

9. Arraste um estado DynamoDB Scan (Verificação do DynamoDB) e solte-o abaixo do estado Verify applicant’s identity and address (Verificar identidade e endereço do candidato). Altere o nome do estado DynamoDB Scan (Verificação do DynamoDB) para **`Get list of credit bureaus`**.
10. Arraste e solte um estado Mapa após o estado Get lst of credit bureaus (Obter lista de agências de crédito). Configure o estado Mapa da seguinte forma:

    a. Altere o nome para **`Get scores from all credit bureaus`**.

    b. Para Modo de processamento, retenha a seleção-padrão de Inline.

    c. Arraste e solte uma ação da API Invocação AWS Lambda no estado vazio chamado Drop state here.

    d. Altere o nome do estado Invocar AWS Lambda para **`Get all scores`**.

11. Mantenha essa janela aberta e prossiga para o próximo tutorial para realizar outras ações.

Próximas etapas

No próximo tutorial, você aprenderá a integrar a função do Lambda usada pelo estado Get credit limit (Obter limite de crédito).

Tutorial 2: Definir a primeira integração de serviços usando uma função do Lambda

Neste tutorial, você vai aprender a definir a primeira integração de serviços para seu fluxo de trabalho. Você usa o estado `Task` chamado Get credit limit (Obter limite de crédito) para invocar uma função do Lambda. Dentro `Task` dos estados, você pode usar as integrações de AWS SDK suportadas pelo Step Functions.

Para definir a primeira integração de serviços para seu fluxo de trabalho, primeiro crie uma função do Lambda. Então atualize seu fluxo de trabalho para especificar a integração do serviço com a função do Lambda. A função do Lambda usada neste tutorial retorna um número inteiro gerado aleatoriamente representando o limite de crédito ao qual um candidato se inscreveu.

Tópicos

Etapa 1: Criar e testar a função do Lambda

Você pode escrever código para a função no editor AWS Management Console ou em seu editor favorito. Nas etapas a seguir, você vai criar uma função do Lambda Node.js chamada `RandomNumberforCredit`.

> **Important**
>
> Certifique-se de que o protótipo de fluxo de trabalho que você criou no Tutorial 1 esteja sob a Região da AWS mesma função Lambda que você criará neste tutorial.

1. Em uma nova guia ou janela, abra o console do Lambda e crie uma função do Lambda do Node.js 16.x intitulada **`RandomNumberforCredit`**. Para obter mais informações sobre como

usar a função do Lambda usando o console, consulte Criar uma função do Lambda no console no Guia do desenvolvedor do AWS Lambda .

2. Na RandomNumberforCreditpágina, escolha index.mjs e substitua o código existente na área Fonte do código pelo código a seguir.

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

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

};
```

3. Na seção Function overview (Visão geral da função), copie o nome de recurso da Amazon da função do Lambda e salve-o em um arquivo de texto. Você vai precisar da função ARN ao especificar a integração do serviço para o estado Get credit limit (Obter limite de crédito). A seguir está um exemplo de ARN.

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

4. Escolha Implantar e escolha Testar para implantar as alterações e ver a saída da função do Lambda.

Etapa 2: Atualizar o fluxo de trabalho, configurar o estado Get credit limit (Obter limite de crédito)

No console do Step Functions, você vai atualizar seu fluxo de trabalho para especificar a integração do serviço com a função do Lambda `RandomNumberforCredit` criada na Etapa 1.

1. Abra a janela do console do Step Functions contendo o protótipo do fluxo de trabalho que você criou no Tutorial 1.
2. Escolha o estado Get credit limit (Obter limite de crédito) e, na guia Configuração, faça o seguinte:

   a. Para o Tipo de integração, retenha a seleção-padrão de Otimizado.

      Usando Step Functions, você pode se integrar com outros Serviços da AWS e orquestrá-los em seus fluxos de trabalho. Para obter mais informações sobre integrações de serviços e seus tipos, consulte Usando AWS Step Functions com outros serviços.

b. Em Nome da função, escolha a função RandomNumberforCreditLambda na lista suspensa.

c. Mantenha as seleções padrão para os demais itens.

3. Mantenha essa janela aberta e prossiga para o próximo tutorial para realizar outras ações.

> **Note**
>
> Neste tutorial, você aprendeu a fazer a integração com uma função do Lambda em um estado `Task` em seus fluxos de trabalho. Você também pode usar outras integrações de AWS SDK compatíveis no `Task` estado especificando o nome do serviço e a chamada de API, conforme mostrado na sintaxe a seguir:
>
> ```
> arn:aws:states:::aws-sdk:serviceName:apiAction
> ```
>
> Para ter mais informações, consulte Usando AWS Step Functions com outros serviços.

Próximas etapas

No próximo tutorial, você vai implementar a lógica condicional em seu fluxo de trabalho. A lógica condicional nas máquinas de estado do Step Functions se comporta de modo semelhante a uma instrução if-else na maioria das linguagens de programação comuns. Você vai usar a lógica condicional em seu fluxo de trabalho para determinar o caminho de execução com base nas informações condicionais.

Tutorial 3: Implemente uma condição if-else em seu fluxo de trabalho

Você pode implementar condições if-else em seus fluxos de trabalho usando o estado `Choice`. Ele determina o caminho de execução do fluxo de trabalho com base no fato de uma condição especificada ser avaliada como verdadeira ou falsa.

Neste tutorial, você vai adicionará lógica condicional para determinar se o valor do crédito aplicado retornado pela função do Lambda `RandomNumberforCredit` usada no Tutorial 2 excede um limite específico. Se o valor exceder o limite, o aplicativo exigirá uma interação humana para aprovação. Caso contrário, o aplicativo é aprovado automaticamente e segue para a próxima etapa.

Você vai simular a etapa de interação humana pausando a execução do fluxo de trabalho até que um token de tarefa seja retornado. Para fazer isso, você passará um token de tarefa para a integração do SDK AWS que você vai usar neste tutorial, que é o Amazon Simple Notification Service. A execução do fluxo de trabalho será pausada até que ele receba o token da tarefa de volta com uma chamada de API `SendTaskSuccess`. Para mais informações sobre o uso de tokens da tarefa, consulte Aguardar um retorno de chamada com um token de tarefa.

Como você já definiu as etapas para aprovação humana e aprovação automática em seu protótipo de fluxo de trabalho, neste tutorial, primeiro você cria um tópico do Amazon SNS que recebe o token de retorno de chamada. Em seguida, você cria uma função do Lambda para implementar a funcionalidade de retorno de chamada. Por fim, você atualiza seu protótipo de fluxo de trabalho adicionando os detalhes destas integrações de AWS service (Serviço da AWS).

Tópicos

Etapa 1: Criar um tópico do Amazon SNS que recebe o token de retorno de chamada

Para implementar a etapa de interação humana, você publicará em um tópico do Amazon Simple Notification Service e passará o token da tarefa de retorno de chamada para esse tópico. A tarefa de retorno de chamada pausará a execução do fluxo de trabalho até que o token da tarefa seja retornado com uma carga útil.

1. Abra o console do Amazon SNS e crie um tipo de tópico Padrão. Para obter informações sobre a criação de um tópico, consulte Criar um tópico do Amazon SNS no Guia do desenvolvedor do Amazon Simple Notification Service.
2. Especifique o nome do tópico como **`TaskTokenTopic`**.
3. Copie o ARN do tópico e salve-o em um arquivo de texto. Você precisará do ARN do tópico ao especificar a integração do serviço para o estado Aguardar aprovação humana. Veja um exemplo de ARN do tópico a seguir:

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

4. Crie uma assinatura baseada em e-mail para o tópico e confirme sua inscrição. Para obter informações sobre assinar um tópico, consulte Criar uma assinatura do tópico no Guia do desenvolvedor do Amazon Simple Notification Service.

Etapa 2: Criar uma função do Lambda para lidar com o retorno de chamada

Para lidar com a funcionalidade de retorno de chamada, você vai definir uma função do Lambda e adicionar o tópico do Amazon SNS criado na Etapa 1 como um gatilho para essa função. Quando você publica no tópico do Amazon SNS com um token de tarefa, a função do Lambda é invocada com a carga útil da mensagem publicada.

- Etapa 2.1: Criar a função do Lambda para lidar com o retorno de chamada
- Etapa 2.2: Adicionar o tópico do Amazon SNS como um gatilho para a função do Lambda
- Etapa 2.3: Fornecer permissões necessárias para a função do Lambda: perfil do IAM

Etapa 2.1: Criar a função do Lambda para lidar com o retorno de chamada

Nessa função, você vai processar a solicitação de aprovação do limite de crédito e retornar o resultado da solicitação como bem-sucedido com a chamada de API `SendTaskSuccess`. Essa função do Lambda também vai retornar o token de tarefa que recebeu do tópico do Amazon SNS.

Para simplificar, a função do Lambda usada para a etapa de interação humana aprova automaticamente qualquer tarefa e retorna o token da tarefa com uma chamada de API `SendTaskSuccess`. Você pode nomear a função do Lambda como **`callback-human-approval`**.

1. Em uma nova guia ou janela, abra o console do Lambda e crie uma função do Lambda do Node.js 16.x intitulada **`callback-human-approval`**. Para obter mais informações sobre como usar a função do Lambda usando o console, consulte Criar uma função do Lambda no console no Guia do desenvolvedor do AWS Lambda.
2. Na página callback-human-approval, substitua o código na área Origem do código pelo código a seguir.

```
// 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. Mantenha essa janela aberta e execute as etapas na próxima seção para ações adicionais.

Etapa 2.2: Adicionar o tópico do Amazon SNS como um gatilho para a função do Lambda

Quando você adiciona o tópico do Amazon SNS criado na Etapa 1 deste tutorial como um gatilho para a função do Lambda que você criou na Etapa 2.1 deste tutorial, a função do Lambda é acionada sempre que você publica no tópico do Amazon SNS. Quando você publica no tópico do Amazon SNS com um token de tarefa, a função do Lambda é invocada com a carga útil da mensagem publicada. Para obter mais informações sobre a configuração de acionadores para funções do Lambda, consulte Como configurar gatilhos no Guia do desenvolvedor do AWS Lambda.

1. Na seção Visão geral da função da função do Lambda`callback-human-approval`, escolha Adicionar gatilho.
2. Na lista suspensa de gatilhos, escolha um SNS como o gatilho.

3. Para o tópico do SNS, comece digitando o nome do tópico do Amazon SNS que você criou na Etapa 1 deste tutorial e escolha-o na lista suspensa que aparece.
4. Escolha Add (Adicionar).
5. Mantenha essa janela aberta e execute as etapas na próxima seção para ações adicionais.

Etapa 2.3: Fornecer permissões necessárias para a função do Lambda: perfil do IAM

Você deve fornecer à função do Lambda `callback-human-approval` as permissões para acessar Step Functions para retornar o token da tarefa junto com a chamada de API `SendTaskSucess`.

1. Na página callback-human-approval, escolha a guia Configuração e escolha Permissões.
2. Em Função de execução, escolha o Nome da função para navegar até a página Funções do console do AWS Identity and Access Management.
3. Para adicionar a permissão necessária, escolha Adicionar permissão e Anexar políticas.
4. Na caixa Pesquisar, digite **`AWSStepFunctions`** e pressione Enter.
5. Escolha AWSStepFunctionsFullAccess e role para baixo para escolher Anexar políticas. Isso adiciona a política que contém a permissão necessária para a função do Lambda `callback-human-approval`.

Etapa 3: Atualizar o fluxo de trabalho, adicionar a lógica de condição if-else no estado Escolha

No console do Step Functions, defina a lógica condicional para seu fluxo de trabalho usando o estado `Choice`. Se a saída retornada pela função do Lambda `RandomNumberforCredit` for menor que 5.000, o crédito solicitado será aprovado automaticamente. Se a saída retornada for maior ou igual a 5.000, a execução do fluxo de trabalho prosseguirá para a etapa de interação humana para aprovação do limite de crédito.

No estado `Choice`, você usa um operador de comparação para comparar uma variável de entrada com um valor específico. Você pode especificar a variável de entrada como entrada de execução ao iniciar a execução de uma máquina de estado ou usar a saída de uma etapa anterior como entrada para a etapa atual. Por padrão, a saída de uma etapa é armazenada em uma variável chamada `Payload`. Para usar o valor da variável `Payload` para comparação no estado `Choice`, use a sintaxe $ conforme mostrado no procedimento a seguir.

Para obter informações sobre como as informações fluem de um estado para outro e especificar entradas e saídas em seus fluxos de trabalho, consulte Tutorial 7: Configurar entrada e saída e Processamento de entrada e saída no Step Functions.

> **Note**
>
> Se o estado `Choice` usar uma variável de entrada especificada na entrada de execução da máquina de estado para comparação, use a sintaxe `$.variable_name` para realizar a comparação. Por exemplo, para comparar uma variável, como `myAge`, use a sintaxe `$.myAge`.

Como nessa etapa, o estado `Choice` receberá informações do estado Get credit limit (Obter limite de crédito), você usará a sintaxe $ para a configuração do estado `Choice`. Para explorar como o resultado da execução da máquina de estado difere quando você usa a sintaxe `$.variable_name` na configuração de estado `Choice` para se referir à saída de uma etapa anterior, consulte a seção Como depurar o caminho inválido Erro de estado de escolha no Tutorial 8.

Para adicionar a lógica de condição if-else usando o estado **`Choice`**

1. Abra a janela do console do Step Functions contendo o protótipo do fluxo de trabalho que você criou no Tutorial 1: Criar o protótipo para sua máquina de estado.
2. Escolha o estado Credit applied >= 5000? (Crédito aplicado >= 5000) e, na guia Configuração, especifique a lógica condicional da seguinte forma:
   a. Em Regras de escolha, escolha o ícone Editar no quadro Regra 1 para definir a regra de primeira escolha.
   b. Clique em Adicionar condições.
   c. Na caixa de diálogo Condições da regra 1, para Variável, insira **$**.
   d. Para Operador, escolha é menor que.
   e. Em Valor, escolha Número constante e insira **5000** no campo ao lado da lista suspensa Valor.
   f. Clique em Salvar condições.
   g. Para a lista suspensa Then next state is: (Então o próximo estado é:), escolha Auto-approve limit (Limite de aprovação automática).

h. Escolha Adicionar nova regra de escolha e defina a regra de segunda opção quando o valor do crédito for maior ou igual a 5.000, repetindo as subetapas 2.b a 2.f. Em Operador, escolha é maior ou igual a.

i. Para a lista suspensa Então o próximo estado é:, escolha Aguardar aprovação humana.

j. No campo Regra padrão, clique no ícone de Editar para definir a regra de escolha padrão e selecione Aguardar aprovação humana na lista suspensa Estado padrão. Você define a regra padrão para especificar o próximo estado para o qual fazer a transição se nenhuma das condições do estado de Escolha for avaliada como verdadeira ou falsa.

3. Configure o estado Aguardar aprovação humana da seguinte forma:

a. Na guia Configuração, em Tópico, comece digitando o nome do tópico do Amazon SNS, TaskTokenTopic, e escolha o nome conforme ele aparece na lista suspensa.

b. Em Mensagem, escolha Inserir mensagem na lista suspensa. No campo Mensagem, você especifica a mensagem que deseja publicar no tópico do Amazon SNS. Neste tutorial, você publica um token de tarefa como a mensagem.

Um token de tarefa permite pausar um fluxo de trabalho do Step Functions do tipo padrão até que um processo externo seja concluído e o token de tarefa seja retornado. Quando você especifica um estado de Tarefa como uma tarefa de retorno de chamada especificando o padrão de integração do serviço `.waitForTaskToken`, um token de tarefa é gerado e colocado no objeto de contexto quando a tarefa é iniciada. O objeto de contexto é uma estrutura JSON interna que está disponível durante uma execução e contém informações sobre sua máquina de estado e execução. Para obter mais informações sobre objetos de contexto, consulte Objeto de contexto.

c. Na caixa que aparece, insira o seguinte como mensagem:

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

d. Escolha a caixa de seleção Aguardar retorno de chamada.

e. Escolha Concluído na caixa de diálogo que aparece.

4. Mantenha essa janela aberta e prossiga para o próximo tutorial para realizar outras ações.

Próximas etapas

No próximo tutorial, você aprenderá a realizar várias tarefas em paralelo.

Tutorial 4: Definir várias tarefas a serem executadas em paralelo

Até agora, você aprendeu a executar fluxos de trabalho de modo sequencial. Porém, você pode executar duas ou mais etapas em paralelo usando o estado `Parallel`. Um estado `Parallel` faz com que o interpretador execute cada ramificação ao mesmo tempo.

Ambas as ramificações em um estado `Parallel` recebem a mesma entrada, mas cada ramificação processa as partes da entrada específicas para ela. O Step Functions espera até que cada ramificação conclua a execução para ir para a próxima etapa.

Neste tutorial, você usa o estado Em paralelo para verificar ao mesmo tempo a identidade e o endereço do candidato.

Tópicos

Etapa 1: Criar as funções do Lambda para realizar as verificações necessárias

Esse fluxo de trabalho do aplicativo de cartão de crédito invoca duas funções Lambda dentro do estado Em paralelo para verificar a identidade e o endereço do solicitante. Essas verificações são realizadas ao mesmo tempo usando o estado Em paralelo. A máquina de estado conclui a execução só depois do fim da execução de ambas as ramificações paralelas.

Para criar as funções do Lambda de verificação de identidade e verificação de endereço

1. Em uma nova guia ou janela, abra o console do Lambda e crie duas funções do Lambda do Node.js 16.x intituladas `check-identity` e `check-address`. Para obter mais informações sobre como usar a função do Lambda usando o console, consulte Criar uma função do Lambda no console no Guia do desenvolvedor do AWS Lambda.
2. Abra a página da função check-identity e substitua o código existente na área Fonte do código pelo seguinte 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 a página da função check-address e substitua o código existente na área Fonte do código pelo seguinte 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 as funções do Lambda, na seção Function overview (Visão geral da função), copie seus respectivos nomes de recursos da Amazon (ARN) e salve-os em um arquivo de texto. Você precisará dos ARNs da função ao especificar a integração do serviço para o estado Verify applicant's identity and address (Verificar identidade e endereço do candidato). A seguir está um exemplo de ARN:

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

Etapa 2: Atualizar o fluxo de trabalho, Adicionar tarefas paralelas a serem executadas

No console do Step Functions, você vai atualizar seu fluxo de trabalho para especificar a integração do serviço com as funções do Lambda check-identity e check-address criadas na Etapa 1.

Para adicionar tarefas paralelas ao fluxo de trabalho

1. Abra a janela do console do Step Functions contendo o protótipo do fluxo de trabalho que você criou no Tutorial 1: Criar o protótipo para sua máquina de estado.
2. Escolha o estado Verify identity (Verificar identidade) e, na guia Configuração, faça o seguinte:

   a. Para o Tipo de integração, retenha a seleção-padrão de Otimizado.

   > **Note**
   >
   > Usando o Step Functions, você pode se integrar com outros Serviços da AWS e orquestrá-los em seus fluxos de trabalho. Para obter mais informações sobre integrações de serviços e seus tipos, consulte Usando AWS Step Functions com outros serviços

   b. Em Function name (Nome da função), escolha a função do Lambda check-identity na lista suspensa.

   c. Em Carga útil, escolha Inserir carga útil e substitua a carga útil de exemplo pela seguinte como carga útil:

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

3. Escolha o estado Verify address (Verificar endereço) e, na guia Configuração, faça o seguinte:

   a. Para o Tipo de integração, retenha a seleção-padrão de Otimizado.

   b. Em Function name (Nome da função), escolha a função do Lambda check-address na lista suspensa.

   c. Em Carga útil, escolha Inserir carga útil e substitua a carga útil de exemplo pela seguinte como carga útil:

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

4. Clique em Próximo.

Tutorial 5: Iterar de modo simultâneo em uma coleção de itens

No tutorial anterior, você aprendeu a executar ramificações separadas de etapas em paralelo usando o estado `Parallel`. Usando o estado `Map`, é possível executar um conjunto de etapas do fluxo de trabalho para cada item em um conjunto de dados. As iterações do estado `Map` são executadas em paralelo, o que possibilita o processamento rápido de um conjunto de dados.

Ao incluir o estado `Map` em seus fluxos de trabalho, você pode realizar tarefas, como processamento de dados, usando uma destas opções Modos de processamento do estado do mapa: modo em linha e modo distribuído. Para configurar um estado `Map`, você define um `ItemProcessor`, que contém objetos JSON que especificam o modo de processamento do estado `Map` e sua definição. Neste tutorial, você executa o estado `Map` no modo inline padrão, que aceita até 40 iterações simultâneas. Quando você executa o estado `Map` no modo Distribuído, ele aceita até 10 mil execuções paralelas de fluxo de trabalho secundário.

Quando a execução do fluxo de trabalho entrar no estado `Map`, ela vai iterar em uma matriz JSON especificada na entrada do estado. Para cada item da matriz, sua iteração correspondente é executada no contexto do fluxo de trabalho contendo o estado `Map`. Quando todas as iterações forem concluídas, o estado `Map` retornará uma matriz contendo a saída de cada item processado pelo `ItemProcessor`.

Neste tutorial, você aprende a usar o estado `Map` no modo inline para obter a pontuação de crédito de um candidato por meio da iteração em um conjunto de agências de crédito. Para fazer isso, primeiro você busca os nomes de todas as agências de crédito armazenadas em uma tabela do Amazon DynamoDB e usa o estado `Map` para percorrer a lista de agências de crédito para obter a pontuação de crédito do solicitante que cada uma dessas agências relata.

Tópicos

Etapa 1: Criar uma tabela do DynamoDB para armazenar o nome de todas as agências de crédito

Nesta etapa, você cria uma tabela chamada **`GetCreditBureau`** usando o console do DynamoDB. A tabela usa o atributo de string Name como chave de Partição. Nessa tabela, você armazena o nome de todas as agências de crédito das quais deseja obter a pontuação de crédito do candidato.

1. Faça login no AWS Management Console e abra o console do DynamoDB em https://console.aws.amazon.com/dynamodb/.
2. No painel de navegação no console, selecione Tabelas e Criar tabela.
3. Insira os detalhes da tabela assim:
   a. Para o Table name (Nome da tabela), insira **`GetCreditBureau`**.
   b. Em Partition key (Chave de partição), insira **`Name`**.
   c. Mantenha as seleções padrão e escolha Criar tabela.
4. Depois da criação da tabela, na lista Tabelas, escolha a tabela GetCreditBureau.
5. Escolha Ações e Criar item.
6. Em Valor, insira o nome de uma agência de crédito. Por exemplo, **`CredTrack`**.
7. Selecione Create Item (Criar item).
8. Repita esse processo e crie itens para nomes de outras agências de crédito. Por exemplo, **`KapFinn`** e **`CapTrust`**.

Etapa 2: Atualizar a máquina de estado, Buscar resultados da tabela do DynamoDB

No console do Step Functions, você vai adicionar um estado `Task` e usar a integração do SDK AWS para buscar os nomes das agências de crédito da tabela do DynamoDB que você criou na Etapa 1. Você vai usar a saída dessa etapa como entrada para o estado `Map` que adicionará posteriormente em seu fluxo de trabalho neste tutorial.

1. Abra a máquina de estado CreditCardWorkflow para atualizá-la.
2. Escolha o estado Get list of credit bureaus (Obter lista de agências de crédito).
3. Para API Parameters (Parâmetros da API), especifique o valor do Nome da tabela como **`GetCreditBureau`**.

Etapa 3: Criar uma função do Lambda que retornará as pontuações de crédito de todas as agências de crédito

Nesta etapa, você cria uma função do Lambda que recebe os nomes de todas as agências de crédito como entrada e retorna a pontuação de crédito do solicitante para cada uma dessas agências de crédito. Essa função do Lambda será invocada do estado `Map` que você adicionará ao seu fluxo de trabalho na Etapa 4 deste tutorial.

1. Crie uma função do Lambda Node.js 16.x e dê a ela o nome de **`get-credit-score`**.
2. Na página intitulada get-credit-score, cole o código a seguir na área Fonte do código.

```
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. Implante a função do Lambda.

Etapa 4: Atualizar a máquina de estado, adicionar um estado do mapa para buscar iterativamente as pontuações de crédito

No console do Step Functions, você adiciona um estado `Map` que invoca a função do Lambda get-credit-score para verificar a pontuação de crédito do candidato em todas as agências de crédito retornadas pelo estado Get list of credit bureaus (Obter lista de agências de crédito).

1. Abra a máquina de estado CreditCardWorkflow para atualizá-la.
2. Escolha Get scores from all credit bureaus (Obter pontuações de todas as agências de crédito).

3. Na guia Configuração, escolha Provide a path to items array (Fornecer um caminho para a matriz de itens) e insira **`$.Items`**.
4. Escolha a etapa Get all scores (Obter todas as pontuações) dentro do estado `Map`.
5. Na guia Configuração, certifique-se de que, em Tipo de integração, Otimizado esteja selecionado.
6. Para o nome da Função, comece digitando o nome da função do Lambda get-credit-score e escolha-o na lista suspensa exibida.
7. Em Carga útil, escolha No payload (Sem carga útil).

Tutorial 6: Salvar o fluxo de trabalho e executar a máquina de estado

Agora que você configurou os recursos de tudo o que Serviços da AWS você está usando no protótipo do fluxo de trabalho, você pode salvá-lo como uma máquina de estado do Step Functions e começar a executá-lo.

Tópicos

- Etapa 1: Revisar a definição da máquina de estado gerada automaticamente e salvar a máquina de estado
- Etapa 2: Adicionar as políticas do IAM restantes
- Etapa 3: Executar a máquina de estado

Etapa 1: Revisar a definição da máquina de estado gerada automaticamente e salvar a máquina de estado

Conforme você arrasta e solta os estados da guia Fluxo para a tela no Workflow Studio para criar o protótipo de fluxo de trabalho, o Step Functions cria automaticamente a definição de Amazon States Language (ASL) do seu fluxo de trabalho em tempo real. Você pode editar essa definição conforme necessário no Editor de código.

Para revisar a definição de ASL e salvar a máquina de estado

1. (Opcional) Escolha Definição no Inspector para ver a definição da máquina de estado Amazon States Language (ASL), que é gerada automaticamente com base nas suas seleções nas guias Ações e Fluxo e no painel Inspetor.

> **Tip**
>
> Para editar a definição, você pode abrir o editor de código escolhendo Código na parte superior da página. Para este tutorial, continue com a definição gerada automaticamente.

2. Especifique um nome para a máquina de estado. Para fazer isso, escolha o ícone de edição ao lado do nome padrão da máquina de estado de MyStateMachine. Em seguida, em Configuração da máquina de estado, insira um nome na caixa Nome da máquina de estado.

   Para este tutorial, insira o nome **`CreditCardWorkflow`**.

3. (Opcional) Em Configuração da máquina de estado, especifique outras configurações do fluxo de trabalho, como o tipo de máquina de estado e a função de execução.

   Para este tutorial, mantenha todas as seleções padrão nas Configurações da máquina de estado.

   > **Note**
   >
   > (Opcional) O Step Functions cria automaticamente uma função de execução para a máquina de estado com os privilégios mínimos necessários para invocar a função do Lambda `RandomNumberforCredit` e publicar no tópico do Amazon SNS.

   Se você já criou um perfil do IAM com as permissões corretas para a máquina de estado e deseja usá-lo, em Permissões, clique em Escolher um perfil existente e selecione uma função na lista. Ou selecione Inserir um ARN de função e forneça o ARN para esse perfil do IAM.

4. Na caixa de diálogo Confirmar criação do perfil, selecione Confirmar para continuar.

   Você também pode escolher Exibir configurações do perfil para voltar às Configurações da máquina de estado.

   > **Note**
   >
   > Se você excluir o perfil do IAM criado pelo Step Functions, não será possível recriá-lo posteriormente. Da mesma forma, se você modificar a função (por exemplo, removendo

o Step Functions das entidades principais na política do IAM), o Step Functions não poderá restaurar as configurações originais dela posteriormente.

Etapa 2: Adicionar as políticas do IAM restantes

Como o Step Functions não gera automaticamente as permissões para invocar as funções Lambda usadas no estado `Parallel`, você precisa adicionar a política necessária.

Para adicionar a política restante

1. Na CreditCardWorkflowpágina, escolha a função do IAM para sua máquina de estado para navegar até o console do IAM. Você vai adicionar as permissões necessárias para as funções restantes do Lambda nesta página.
2. Escolha Adicionar permissões e depois Anexar políticas.
3. Na caixa Pesquisar, digite **`AWSLambdaRole`** e pressione Enter.
4. Escolha AWSLambdaRolee, em seguida, escolha Anexar políticas. Essa política agora é adicionada à função de execução da sua máquina de estado. Essa política permite que você invoque qualquer função do Lambda em sua máquina de estado.

Etapa 3: Executar a máquina de estado

As execuções de máquinas de estado são instâncias em que o fluxo de trabalho é executado para a realização de tarefas.

Para executar a máquina de estado

1. Na CreditCardWorkflowpágina, escolha Iniciar execução.

   A caixa de diálogo Iniciar execução é exibida.
2. Na caixa de diálogo Iniciar execução, faça o seguinte:

   a. (Opcional) Para identificar a execução, insira um nome para ela no campo Nome. Por padrão, o Step Functions gera automaticamente um nome de execução exclusivo.

> **Note**
>
> O Step Functions permite criar nomes de máquina de estado, execuções, atividades e labels que contenham caracteres não ASCII. Esses nomes não ASCII não funcionam com a Amazon. CloudWatch Para garantir que você possa acompanhar CloudWatch as métricas, escolha um nome que use somente caracteres ASCII.

> **Note**
>
> Você não precisa fornecer nenhuma entrada para executar essa máquina de estado. Mas você pode especificar uma entrada de execução, se necessário, na área Entrada da caixa de diálogo Iniciar execução para outras máquinas de estado. Para ver um exemplo de como fornecer entrada de execução para uma máquina de estado, consulte Etapa 4: Iniciar uma nova execução do tutorial Aprenda a usar o AWS Step Functions Workflow Studio.

b. Selecione Iniciar execução.

3. O console do Step Functions direciona você para uma página em que o título é o ID da execução. Essa página é conhecida como página de Detalhes da execução. Nesta página, você pode revisar os resultados da execução à medida que a execução avança ou após a conclusão.

   Para revisar os resultados da execução, escolha estados individuais na Exibição em gráfico e, em seguida, escolha as guias individuais no painel Detalhes da etapa para visualizar os detalhes de cada estado, incluindo entrada, saída e definição, respectivamente. Para obter detalhes sobre as informações de execução que você pode visualizar na página Detalhes da execução, consulte Página de Detalhes da execução — Visão geral da interface.

Tutorial 7: Configurar entrada e saída

Uma execução do Step Functions recebe um texto JSON como entrada e passa essa entrada para o primeiro estado no fluxo de trabalho. Os estados individuais em um fluxo de trabalho recebem dados JSON como entrada e geralmente passam dados JSON como saída para o próximo estado. Por padrão, os dados passam de um estado para o próximo estado no fluxo de trabalho, a menos que você tenha configurado a entrada e/ou a saída para um ou mais estados no fluxo de trabalho. Ter noções básicas sobre como as informações fluem de um estado para outro e aprender a filtrar

e manipular esses dados é a chave para efetivamente projetar e implementar fluxos de trabalho no Step Functions.

O Step Functions fornece vários filtros para controlar o fluxo de dados de entrada e saída entre os estados. Os filtros a seguir estão disponíveis para uso em seus fluxos de trabalho:

> **Note**
>
> Dependendo do seu caso de uso, talvez você não precise aplicar todos esses filtros em seus fluxos de trabalho.

`InputPath`

Seleciona QUAL parte de toda a carga de entrada a ser usada como entrada de uma tarefa. Se você especificar esse campo, o Step Functions o aplicará primeiro.

`Parâmetros`

Especifica COMO deve ser a entrada antes de invocar a tarefa. Com o campo `Parameters`, você pode criar uma coleção de pares de chave/valor que são passados como entrada para a integração de serviços da AWS service (Serviço da AWS), como uma função do AWS Lambda. Esses valores podem ser estáticos ou selecionados dinamicamente da entrada de estado ou do objeto de contexto do fluxo de trabalho.

`ResultSelector`

Determina O QUE escolher na saída de uma tarefa. Com o campo `ResultSelector`, você pode criar uma coleção de pares de chave/valor que substituem o resultado de um estado e enviam essa coleção para `ResultPath`.

`ResultPath`

Determina ONDE colocar a saída de uma tarefa. Use o `ResultPath` para determinar se a saída de um estado é uma cópia de sua entrada, o resultado que produz ou uma combinação das duas opções.

`OutputPath`

Determina O QUE enviar para o próximo estado. Com o `OutputPath`, é possível filtrar informações indesejáveis e transmitir somente a parte do JSON que é importante para você.

> **Tip**
>
> Os filtros `Parameters` e `ResultSelector` funcionam construindo JSON, enquanto os filtros `InputPath` e `OutputPath` funcionam filtrando nós específicos em um objeto de dados JSON, e o filtro `ResultPath` funciona criando um campo no qual a saída pode ser adicionada.

Neste tutorial, você aprende a realizar as seguintes tarefas:

- Selecione partes específicas da entrada bruta usando o InputPath filtro
- Manipule a entrada selecionada usando o filtro Parâmetros
- Configure a saída usando os filtros ResultSelector, ResultPath e OutputPath

Para obter mais informações sobre como configurar a entrada e a saída em seus fluxos de trabalho, consulte Processamento de entrada e saída no Step Functions.

Selecione partes específicas da entrada bruta usando o InputPath filtro

Use o filtro `InputPath` para selecionar uma parte específica da carga de entrada.

Se você não especificar `InputPath`, o valor padrão será $, o que fará a tarefa do estado se referir à entrada bruta inteira, em vez de a uma parte específica.

Para aprender a usar o filtro InputPath, execute as seguintes etapas:

- Etapa 1: Criar uma máquina de estado
- Etapa 2: Executar a máquina de estado
- Etapa 3: Usar o filtro InputPath para selecionar partes específicas de uma entrada de execução

Etapa 1: Criar uma máquina de estado

> **Important**
>
> Certifique-se de que sua máquina de estado esteja na mesma AWS conta e região da função Lambda que você criou anteriormente.

1. Use o exemplo de estado `Parallel` que você aprendeu no Tutorial 4 para criar uma máquina de estado. Verifique se o protótipo do fluxo de trabalho é semelhante ao protótipo a seguir.
2. Configure as integrações para as funções do Lambda `check-identity` e `check-address`. Para obter informações sobre como criar as funções do Lambda e usá-las em sua máquina de estado, consulte Etapa 1: Criar as funções do Lambda para realizar as verificações necessárias e Etapa 2: Atualizar o fluxo de trabalho, Adicionar tarefas paralelas a serem executadas.
3. Para Carga útil, retenha a seleção-padrão de Usar entrada de estado como carga útil.
4. Escolha Avançar e execute as etapas de 1 a 3 em Etapa 1: Salvar a máquina de estado do Tutorial 5 para criar uma máquina de estado. Para este tutorial, dê à sua máquina de estado o nome de **`WorkflowInputOutput`**.

Etapa 2: Executar a máquina de estado

1. Na WorkflowInputOutputpágina, escolha Iniciar execução.
2. (Opcional) Para identificar a execução, insira um nome para ela no campo Nome. Por padrão, o Step Functions gera automaticamente um nome de execução exclusivo.

> **Note**
>
> O Step Functions permite criar nomes de máquina de estado, execuções, atividades e labels que contenham caracteres não ASCII. Esses nomes não ASCII não funcionam com a Amazon. CloudWatch Para garantir que você possa acompanhar CloudWatch as métricas, escolha um nome que use somente caracteres ASCII.

3. Na área Entrada, adicione os seguintes dados JSON como entrada de execução.

```
{
  "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. Selecione Iniciar execução.
5. A execução da máquina de estado resulta em um erro porque você não especificou quais partes da entrada de execução as funções do Lambda `check-identity` e `check-address` devem usar para realizar a verificação de identidade e endereço necessária.
6. Continue com a Etapa 3 deste tutorial para corrigir o erro.

Etapa 3: Usar o filtro **`InputPath`** para selecionar partes específicas de uma entrada de execução

1. Na página Detalhes da execução, escolha Edit state machine (Editar máquina de estado).
2. Para verificar a identidade do candidato conforme mencionado na entrada de execução fornecida em Etapa 2: Executar a máquina de estado, edite a definição da tarefa Verificar identidade da seguinte forma:

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

Assim, os seguintes dados JSON ficam disponíveis como entrada para a função `check-identity`.

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

3. Para verificar o endereço do candidato conforme mencionado na entrada de execução, edite a definição da tarefa `Verify address` da seguinte forma:

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

Assim, os seguintes dados JSON ficam disponíveis como entrada para a função `check-address`.

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

4. Selecione Iniciar execução. A execução da máquina de estado agora é concluída com êxito.

Manipule a entrada selecionada usando o filtro Parâmetros

Embora o filtro `InputPath` ajude a limitar a entrada JSON bruta fornecida, usando o filtro `Parameters`, você pode passar uma coleção de pares de valores-chave como entrada. Esses pares de chave-valor podem ser valores estáticos que você define na definição de máquina de estado, ou valores selecionados na entrada bruta usando `InputPath`.

Em seus fluxos de trabalho, `Parameters` são aplicados depois de `InputPath`. O `Parameters` ajuda você a especificar como a tarefa subjacente aceita sua carga de entrada. Por exemplo, se a função do Lambda `check-address` aceitar um parâmetro de string como entrada em vez dos dados JSON, você poderá usar o filtro `Parameters` para transformar a entrada.

No exemplo a seguir, o filtro `Parameters` recebe a entrada que você selecionou usando `InputPath` em Etapa 3: Usar o filtro `InputPath` para selecionar partes específicas de uma entrada de execução e aplica a função intrínseca `States.Format` nos itens de entrada para criar uma string chamada `addressString`. As funções intrínsecas ajudam você a realizar operações básicas de processamento de dados em uma determinada entrada. Para obter mais informações, consulte Funções intrínsecas.

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

Assim, a sequência de caracteres a seguir é criada e fornecida à função do Lambda `check-address` como entrada.

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

Configure a saída usando os filtros ResultSelector, ResultPath e OutputPath

Quando a função do Lambda `check-address` é invocada na máquina de estado WorkflowInputOutput, a função retorna uma carga de saída após realizar a verificação do endereço. Na página Detalhes da execução, escolha a etapa Verificar endereço e visualize a carga de saída dentro do Resultado da tarefa no painel Detalhes da etapa.

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

Como usar ResultSelector

Agora, se precisar fornecer o resultado das verificações de identidade e endereço para os seguintes estados em seu fluxo de trabalho, você poderá selecionar o nó Payload.body no JSON de saída e usar a função intrínseca `StringToJson` no filtro `ResultSelector` para formatar os dados conforme necessário.

`ResultSelector` seleciona o que é necessário na saída da tarefa. No exemplo a seguir, `ResultSelector` pega a string em $.Payload.body e aplica a função `States.StringToJson` intrínseca para converter a string em JSON e coloca o JSON resultante dentro do nó de identidade.

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

Assim, os seguintes dados JSON são criados.

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

Ao trabalhar com esses filtros de entrada e saída, você também pode encontrar erros de runtime decorrentes da especificação de expressões de caminho JSON inválidas. Para obter informações, consulte.

Use o ResultPath

Você pode especificar um local na carga inicial de entrada para salvar o resultado do processamento da tarefa de um estado usando o campo `ResultPath`. Se você não especificar `ResultPath`, o valor padrão será $, o que fará a carga inicial de entrada ser substituída pelo resultado bruto da tarefa. Se você especificar `ResultPath` como `null`, o resultado bruto será descartado e a carga inicial de entrada passará a ser a saída efetiva.

Se você aplicar o campo `ResultPath` nos dados JSON criados usando o campo `ResultSelector`, o resultado da tarefa será adicionado dentro do nó de resultados na carga de entrada, conforme mostrado no exemplo a seguir:

```
{
  "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
    }
  }
}
```

Use o OutputPath

Você pode selecionar uma parte da saída do estado após a aplicação de `ResultPath` para passar ao estado seguinte. Assim, é possível filtrar informações indesejadas e transmitir apenas a parte do JSON importante para você.

No exemplo a seguir, o campo `OutputPath` salva a saída do estado dentro do nó de resultados: `"OutputPath": "$.results"`. Dessa forma, a saída final do estado, que você pode passar para o próximo estado, é a seguinte:

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

Como usar os recursos do console para visualizar os fluxos de dados de entrada e saída

Você pode visualizar o fluxo de dados de entrada e saída entre os estados em seus fluxos de trabalho usando o Simulador de fluxo de dados do console Step Functions ou a opção de visualização avançada na página Detalhes da execução.

Tutorial 8: Erros de depuração no console

Ao trabalhar com o Step Functions, você pode encontrar erros de runtime decorrentes de motivos como:

- Um caminho JSON inválido para o campo Variable (Variável) no estado `Choice`.
- Problemas de definição da máquina de estado, como nenhuma regra de correspondência em um estado `Choice`.
- Expressões de caminho JSON inválidas ao aplicar filtros para manipular entrada e saída.
- Falhas na tarefa devido a uma exceção da função do Lambda.

- Erros de permissão do IAM.

Neste tutorial, você vai aprender a depurar alguns desses erros usando o console do Step Functions. Para obter mais informações, consulte Tratamento de erros no Step Functions.

Tópicos

- Como depurar o caminho inválido Erro de estado de escolha
- Como depurar erros de expressão de caminho JSON ao aplicar filtros de entrada e saída

Como depurar o caminho inválido Erro de estado de escolha

Quando você especifica um caminho JSON incorreto ou não solucionável no campo Variável do estado `Choice` ou não define uma regra de correspondência no estado `Choice`, recebe um erro ao executar seu fluxo de trabalho.

Para ilustrar o erro de caminho inválido, este tutorial apresenta um erro de estado `Choice` em seu fluxo de trabalho. Você vai usar a máquina de estado CreditCardWorkflow e editar sua definição para introduzir o erro.

1. Abra o console do Step Functions e escolha a máquina de estado CreditCardWorkflow.
2. Escolha Editar para editar a definição da máquina de estado. Faça a alteração destacada no código a seguir na definição da 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. Escolha Salvar e Salvar mesmo assim.
4. Execute a máquina de estado.
5. Na página Detalhes da execução da máquina de estado faça uma destas ações:
   a. Escolha Causa na mensagem de erro para ver o motivo da falha na execução.
   b. Escolha Mostrar detalhes da etapa na mensagem de erro para ver a etapa que causou o erro.
6. Na guia Input & Output (Entrada e saída) da seção Detalhes da etapa, escolha o botão de alternância Advanced View (Visualização avançada) para ver o caminho de transferência de dados de entrada e saída para um estado selecionado.
7. Em Exibição em gráfico, selecione Credit applied >= 5000? (Crédito aplicado >= 5.000?) e faça o seguinte:
   a. Visualize o valor de entrada do estado na caixa Entrada.
   b. Escolha a guia Definição e observe o caminho JSON especificado para o campo Variável.

      O valor de entrada para Credit applied >= 5000? (Crédito aplicado >=5.000?) é um valor numérico, embora você tenha especificado o caminho JSON para o valor de entrada como `$.Payload`. Durante a execução da máquina de estado, o estado `Choice` não pode resolver esse caminho JSON porque ele não existe.
8. Edite a máquina de estado para especificar o valor do campo Variable (Variável) como **$**.

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

Como depurar erros de expressão de caminho JSON ao aplicar filtros de entrada e saída

Ao trabalhar com os filtros de entrada e saída, você também pode encontrar erros de runtime decorrentes da especificação de expressões de caminho JSON inválidas.

O exemplo a seguir usa a máquina de estado WorkflowInputOutput criada no Tutorial 5 e demonstra um cenário em que você usa o filtro `ResultSelector` para selecionar partes da saída da tarefa.

1. Aplique o filtro `ResultSelector` para escolher uma parte da saída da tarefa para a etapa Verify identity (Verificar identidade). Para fazer isso, edite sua definição de máquina de estado da seguinte forma:

```
{
  "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. Execute a máquina de estado.
3. Na página Detalhes da execução da máquina de estado, faça o seguinte:
   a. Escolha Causa na mensagem de erro para ver o motivo da falha na execução.
   b. Escolha Mostrar detalhes da etapa na mensagem de erro para ver a etapa que causou o erro.
4. Na mensagem de erro, observe que o conteúdo do nó $.Payload.body é uma string JSON com escape. O erro ocorreu porque você não conseguiu se referir a uma string usando a notação de caminho JSON.
5. Para se referir ao nó $.Payload.body.message, faça o seguinte:
   a. Use a função intrínseca States.StringToJSON para primeiro converter a string em um formato JSON.
   b. Especifique o caminho JSON para o nó $.Payload.body.Message dentro da função intrínseca.

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

6. Execute a máquina de estado novamente.

Casos de uso

O AWS Step Functions permite criar fluxos de trabalho visuais que ajudam a transformar com rapidez os requisitos de negócios em aplicativos. O Step Functions gerencia estados, pontos de verificação e reinicializações para você e fornece recursos integrados para lidar automaticamente com erros e exceções. Para entender melhor os recursos que o Step Functions pode oferecer, leia os seguintes casos de uso:

Tópicos

Processamento de dados

Conforme o volume de dados cresce, vindo de fontes cada vez mais diversas, as organizações descobrem que precisam agir com celeridade para processar esses dados a fim de garantir que tomem decisões de negócios mais rápidas e bem informadas. Para processar dados em grande escala, as organizações precisam provisionar recursos de forma elástica para gerenciar as informações que recebem de dispositivos móveis, aplicativos, satélites, marketing e vendas, armazenamentos de dados operacionais, infraestrutura e muito mais.

O Step Functions oferece a escalabilidade, a confiabilidade e a disponibilidade necessárias para gerenciar com sucesso os fluxos de trabalho de processamento de dados. Você pode gerenciar milhões de execuções simultâneas com o Step Functions, pois ele escala horizontalmente e oferece fluxos de trabalho tolerantes a falhas. Processe dados com mais rapidez usando execuções paralelas, como o tipo de estado `Paralelo` do Step Functions ou paralelismo dinâmico usando o tipo de estado `Mapa`. Como parte do fluxo de trabalho, você pode usar o estado `Mapa` para iterar sobre objetos em um armazenamento de dados estático, como um bucket do Amazon S3. O Step Functions também permite que você repita com facilidade execuções com falha ou escolha uma maneira específica de lidar com erros sem a necessidade de gerenciar um processo complexo.

Dependendo das necessidades de processamento de dados, o Step Functions se integra diretamente a outros serviços de processamento de dados oferecidos pela AWS como o AWS Batch,

para processamento em lote, o Amazon EMR, para processamento de big data, o AWS Glue, para preparação de dados, o Athena, para análise de dados e o AWS Lambda, para computação.

Exemplos dos tipos de fluxos de trabalho de processamento de dados que os clientes usam o Step Functions para realizar incluem:

Processamento de arquivos, vídeos e imagens

- Pegue uma coleção de arquivos de vídeo faça a conversão para outros tamanhos ou resoluções ideais para o dispositivo em que serão exibidos, como telefones celulares, notebooks ou televisão.
- Pegue uma grande coleção de fotos enviadas pelos usuários e faça a conversão para miniaturas ou imagens de várias resoluções que podem ser exibidas nos sites dos usuários.
- Pegue dados semiestruturados, por exemplo, um arquivo CSV e faça uma combinação com dados não estruturados, como uma fatura, para produzir um relatório de negócios que é enviado mensalmente às partes interessadas da empresa.
- Pegue os dados de observação da Terra coletados de satélites, faça a conversão para formatos que se alinhem e, em seguida, adicione outras fontes de dados coletadas na Terra para obter informações adicionais.
- Pegue os registros de transporte de vários modos de transporte de produtos e procure otimizações usando simulações de Monte Carlo e, em seguida, envie relatórios para as organizações e pessoas que confiam em você para enviar os produtos.

Coordene os trabalhos de extração, transformação e carregamento (ETL):

- Combine registros de oportunidades de vendas com conjuntos de dados de métricas de marketing por meio de uma série de etapas de preparação de dados usando o AWS Glue e produza relatórios de inteligência de negócios que podem ser usados em toda a organização.
- Crie, inicie e encerre um cluster do Amazon EMR para processamento de big data.

Processamento em lote e workload computação de alta performance (HPC):

- Crie um pipeline de análise secundária e genômica que processe sequências brutas do genoma inteiro em chamadas de variantes. Alinhe arquivos brutos a uma sequência de referência e chame variantes em uma lista especificada de cromossomos usando paralelismo dinâmico.
- Encontre eficiências na produção do próximo dispositivo móvel ou outros eletrônicos, simulando vários layouts e usando diferentes compostos elétricos e químicos. Execute o processamento em grandes lotes das cargas de trabalho por meio de várias simulações para obter o design ideal.

Machine learning

O machine learning permite que as organizações analisem com rapidez os dados coletados para identificar padrões e tomar decisões com o mínimo de intervenção humana. O machine learning começa com um conjunto inicial de dados, conhecido como dados de treinamento. Esses dados de treinamento ajudam a aumentar a precisão da previsão de um modelo de machine learning e servem como base para o modelo aprender. Quando o modelo é considerado preciso o suficiente para atender às necessidades da empresa, ele é implantado na produção. O Kit de desenvolvimento de software (SDK) de ciência de dados do AWS Step Functions é uma biblioteca de código aberto que permite criar com facilidade fluxos de trabalho que pré-processam dados, treinam e publicam os modelos usando o Amazon SageMaker e o Step Functions.

O pré-processamento de conjuntos de dados é a forma como uma organização geralmente cria dados de treinamento. Esse método adiciona informações, como etiquetar objetos em uma imagem, fazer anotações em texto ou processar áudio. Para pré-processar dados, você pode usar o AWS Glue ou criar uma instância do notebook SageMaker que execute o aplicativo Caderno Jupyter. Quando os dados estiverem prontos, eles poderão ser enviados para o Amazon S3 para facilitar o acesso. Conforme os modelos de machine learning forem treinados, você pode fazer ajustes nos parâmetros de cada modelo para melhorar a precisão até que esteja pronto para implantação.

O Step Functions permite que você orquestre fluxos de trabalho de machine learning de ponta a ponta no SageMaker. Esses fluxos de trabalho podem incluir pré-processamento de dados, pós-processamento, engenharia de atributos, validação de dados e avaliação de modelos. Depois que o modelo for implantado na produção, você poderá refinar e testar novas abordagens para melhorar continuamente os resultados dos negócios. Você pode criar fluxos de trabalho prontos para produção diretamente no Python ou pode usar o SDK de ciência de dados do Step Functions para copiar esse fluxo de trabalho, experimentar novas opções e colocar o fluxo de trabalho refinado em produção.

Veja abaixo alguns tipos de fluxos de trabalho de machine learning para os quais os clientes usam o Step Functions:

Detecção de fraudes

- Identifique e evite que transações fraudulentas, como fraude de crédito, ocorram.
- Detecte e evite invasões de contas usando modelos treinados de machine learning.
- Identifique abusos promocionais, incluindo a criação de contas falsas, para que você possa agir com rapidez.

Personalização e recomendações

- Recomende produtos para clientes-alvo com base nos interesses deles.
- Preveja se um cliente atualizará a conta de um nível gratuito para uma assinatura paga.

Enriquecimento de dados

- Use o enriquecimento de dados como parte do pré-processamento para oferecer dados de treinamento melhores para modelos de machine learning mais precisos.
- Anote trechos de texto e áudio para adicionar informações sintáticas, como sarcasmo e gírias.
- Identifique objetos adicionais em imagens para oferecer informações essenciais para o modelo aprender, como se um objeto é uma maçã, uma bola de basquete, uma pedra ou um animal.

Orquestração de microsserviços

A arquitetura de microsserviços divide os aplicativos em serviços com acoplamento fraco. Os benefícios incluem maior escalabilidade, maior resiliência e menor tempo para entrar no mercado. Cada microsserviço é independente, facilitando aumentar a escala verticalmente de um único serviço ou função sem a necessidade de escalar todo o aplicativo. Os serviços individuais têm acoplamento fraco, permitindo que equipes independentes se concentrem em um único processo de negócios, sem a necessidade de entender todo o aplicativo. Os microsserviços também permitem que você escolha quais componentes individuais atendem às necessidades dos negócios, oferecendo a flexibilidade de alterar a seleção sem reescrever todo o fluxo de trabalho. Equipes diferentes podem usar as linguagens e estruturas de programação da própria escolha para trabalhar com o microsserviço e esse microsserviço ainda pode se comunicar com qualquer outro no aplicativo por meio de interfaces de programação de aplicativos (APIs).

O Step Functions oferece várias maneiras de gerenciar os fluxos de trabalho de microsserviços. Para fluxos de trabalho de longa duração, você pode usar fluxos de trabalho padrão com a integração do AWS Fargate para orquestrar aplicativos executados em contêineres. Para fluxos de trabalho de curta duração e alto volume que exigem uma resposta imediata, os fluxos de trabalho expresso síncronos são ideais. Eles podem ser usados para aplicativos de dispositivos móveis ou baseados na web, que, em geral, têm fluxos de trabalho de curta duração e exigem a conclusão de uma série de etapas antes de retornarem uma resposta. Você pode acionar diretamente um fluxo de trabalho expresso síncrono do Amazon API Gateway e a conexão é mantida aberta até que o fluxo

de trabalho seja concluído ou atinja o tempo limite. Para fluxos de trabalho de curta duração que não exigem uma resposta imediata, o Step Functions fornece fluxos de trabalho expresso assíncronos.

Exemplos de algumas orquestrações de API que usam o Step Functions:

Fluxos de trabalho síncronos ou em tempo real

- Altere um valor em um registro, como atualizar o sobrenome de um funcionário e deixe a alteração imediatamente visível na tela.
- Atualize um pedido durante a finalização da compra, como adicionar, remover ou alterar a quantidade de um item e depois faça imediatamente a atualização para o cliente.
- Execute um trabalho de processamento rápido e devolva imediatamente o resultado ao solicitante.

Orquestração de contêiner

- Execute trabalhos no Kubernetes com o Amazon Elastic Kubernetes Service ou no Amazon Elastic Container Service (ECS) com o Fargate e faça a integração com outros serviços da AWS, como o envio de notificações com o Amazon SNS, como parte do mesmo fluxo de trabalho.

Automação de TI e segurança

A automação de TI pode ajudar a gerenciar operações cada vez mais complexas e demoradas, como atualizar e corrigir software, implantar atualizações de segurança para solucionar vulnerabilidades, selecionar infraestrutura, sincronizar dados, rotear tíquetes de suporte e muito mais. A automação de tarefas repetitivas e demoradas pode permitir que a organização conclua as operações de rotina de maneira rápida e consistente em grande escala. Isso permite que você se concentre no trabalho estratégico, como desenvolvimento de atributos, solicitações complexas de suporte e inovação, ao mesmo tempo em que atende a essas demandas crescentes.

O Step Functions permite que você crie fluxos de trabalho que escalam automaticamente para atender às necessidades da empresa sem exigir intervenção manual. Nos casos em que ocorre um erro no fluxo de trabalho, ele geralmente não requer intervenção manual. O Step Functions permite que você, automaticamente, repita as tarefas que falharam e faça um recuo exponencial que pode gerenciar erros no fluxo de trabalho.

Pode haver situações em que a intervenção humana seja necessária antes que o fluxo de trabalho possa progredir. Por exemplo, aprovar um aumento substancial de crédito pode exigir aprovação humana. Para gerenciar isso, você pode definir a lógica de ramificação no Step Functions, de modo

que somente solicitações acima de um valor definido exijam aprovação humana, enquanto todas as outras solicitações sejam concluídas automaticamente. Nos casos em que a aprovação humana é necessária, o Step Functions permite pausar o fluxo de trabalho em uma etapa específica, esperar por uma resposta e continuar o fluxo de trabalho depois que a resposta for recebida.

Veja alguns exemplos dos tipos de fluxos de trabalho de automação para os quais os clientes usam o Step Functions:

Automação de TI

- Corrija automaticamente incidentes como a abertura de uma porta SSH, pouco espaço em disco ou quando um acesso público é concedido a um bucket do Amazon S3.
- Automatize a implantação de StackSets do AWS CloudFormation.

Automação de segurança

- Automatize a resposta a um cenário em que um usuário e a chave de acesso do usuário tenham sido expostos.
- Corrija automaticamente as respostas a incidentes de segurança conforme as ações de políticas definidas, como restringir ações a ARNs específicos ou aplicar outras ações.
- Avise os funcionários sobre e-mails de phishing segundos após o recebimento.

Aprovação humana

- Automatize o treinamento do modelo de machine learning e exija a aprovação manual do modelo por um cientista de dados antes de implantar ou rejeitar automaticamente o modelo com base na resposta recebida.
- Automatize o encaminhamento dos comentários dos clientes recebidos com base na análise de sentimentos para que aqueles com um sentimento negativo sejam imediatamente encaminhados para análise manual.

Como funciona o Step Functions

Esta seção descreve conceitos importantes para ajudar você a se familiarizar com o AWS Step Functions e entender como ele funciona.

Tópicos

Comparação entre os fluxos de trabalho padrão e expresso

Ao criar uma máquina de estado, é necessário selecionar um Tipo, que pode ser Padrão ou Expresso. O Tipo padrão para máquinas de estado é Padrão. Uma máquina de estado cujo Tipo é padrão recebe a designação de fluxo de trabalho padrão, enquanto aquela cujo Tipo é expresso é chamada de fluxo de trabalho expresso.

Para fluxos de trabalho padrão e expresso, você define a máquina de estado usando a Amazon States Language. Suas execuções de máquina de estado se comportarão de forma diferente, dependendo de qual Tipo você selecionar.

> **Important**
>
> O Tipo escolhido não pode ser alterado após a criação da máquina de estado.

> **Note**
>
> Se você definir as máquinas de estado fora do console do Step Functions, como em um editor de sua escolha, será necessário salvar as definições de máquina de estado com a extensão .asl.json.

Os fluxos de trabalho padrão são ideais para fluxos de trabalho de execução demorada (de até um ano), duráveis e auditáveis. Você pode recuperar o histórico de execução completo usando a API do Step Functions, até 90 dias após a conclusão da execução. Os fluxos de trabalho padrão empregam um modelo do tipo exatamente uma vez, onde suas tarefas e estados nunca são executados mais de uma vez, a menos que você tenha especificado o comportamento de `Retry` em ASL. Isso torna os fluxos de trabalho padrão adequados para orquestrar ações não idempotentes, como iniciar um cluster do Amazon EMR ou processar pagamentos. As execuções dos fluxos de trabalho padrão são faturadas de acordo com o número de transições de estado processadas.

Os fluxos de trabalho expresso são ideais para cargas de trabalho de processamento de eventos de alto volume, como ingestão de dados de IoT, processamento e transformação de dados de streaming e back-ends de aplicativos móveis. Eles podem ser executados por até cinco minutos. Os fluxos de trabalho expresso empregam um modelo do tipo pelo menos uma vez, onde a execução pode ser feita mais de uma vez. Isso torna os fluxos de trabalho expresso ideais para orquestrar ações idempotentes, como transformar dados de entrada e armazenar via PUT no Amazon DynamoDB. As execuções de fluxos de trabalho expresso são cobradas de acordo com o número de execuções, sua duração e a memória consumida.

Os fluxos de trabalho padrão e expresso podem ser iniciados automaticamente em resposta a eventos como solicitações HTTP do Amazon API Gateway (APIs totalmente gerenciadas em escala), regras de IoT e mais de 140 fontes de eventos no Amazon EventBridge.

> **Tip**
>
> Para implantar um exemplo de fluxo de trabalho expresso em sua Conta da AWS, consulte o Módulo 7 - API Gateway, estado Paralelo, fluxos de trabalho expresso do AWS Step Functions Workshop.

Para obter informações sobre a experiência de console para execuções de fluxo de trabalho padrão e expresso, consulte Execuções de Fluxo de trabalho Padrão e Expresso no console.

Fluxos de trabalho padrão em comparação aos expressos

	Fluxos de trabalho padrão	Fluxos de trabalho expresso: síncronos e assíncronos
Duração máxima	Um ano	Cinco minutos
Taxa inicial de execução suportada	Para obter informações sobre cotas relacionadas à taxa inicial de execução compatível, consulte Cotas relacionadas ao controle de utilização das ações de API.	Para obter informações sobre cotas relacionadas à taxa inicial de execução compatível, consulte Cotas relacionadas ao controle de utilização das ações de API.
Taxa de transição de estado suportada	Para obter informações sobre cotas relacionadas à taxa de transição de estado compatível, consulte Cotas relacionadas aos controles de utilização de estado.	Sem limite
Definição de preço	Preço por número de transições de estado. Uma transição de estado é contada cada vez que uma etapa em sua execução é concluída.	Cobrado pelo número de execuções, pela duração de cada uma delas e pelo consumo de memória.
Histórico de execução	As execuções podem ser listadas e descritas com	Histórico de execução ilimitado, ou seja, não há limite

	Fluxos de trabalho padrão	Fluxos de trabalho expresso: síncronos e assíncronos
	as APIs do Step Functions . As execuções podem ser depuradas visualmente por meio do console. É possível inspecioná-las no CloudWatch Logs ativando o registro em log na sua máquina de estado. Para obter mais informações sobre como depurar as execuções do fluxo de trabalho padrão no console, consulte Execuções de Fluxo de trabalho Padrão e Expresso no console e Visualizar e depurar execuções.	para manutenção de entradas do histórico de execução geradas em um período de 5 minutos. As execuções podem ser inspecionadas no CloudWatch Logs ou no console do Step Functions ativando o registro em log na sua máquina de estado. Para obter mais informações sobre como depurar as execuções do fluxo de trabalho expresso no console, consulte Execuções de Fluxo de trabalho Padrão e Expresso no console e Visualizar e depurar execuções.
Semântica de execução	Execução de fluxo de trabalho exatamente uma vez.	Fluxos de trabalho expresso assíncronos: execução de fluxo de trabalho de pelo menos uma vez. Fluxos de trabalho expresso síncronos: execução de fluxo de trabalho de no máximo uma vez.

	Fluxos de trabalho padrão	Fluxos de trabalho expresso: síncronos e assíncronos
Integrações de serviços	Oferece suporte a todas as integrações e padrões de serviço.	Oferece suporte a todas as integrações de serviço. **Note** Os fluxos de trabalho expresso não oferecem suporte aos padrões de integração de serviços Job-run (`.sync`) ou Callback (`.waitForTaskToken`).
Atividades do Step Functions	Suporte a atividades do Step Functions.	Não oferece suporte às atividades do Step Functions.

Fluxos de trabalho expresso síncronos e assíncronos

Há dois tipos de fluxos de trabalho expresso que você pode escolher: fluxos de trabalho expresso assíncronos e os síncronos.

- Os fluxos de trabalho expresso assíncronos retornam a confirmação de que o fluxo de trabalho foi iniciado, mas não esperam a conclusão dele. Para obter o resultado, você deve pesquisar o CloudWatch Logs do serviço. Você pode usar os fluxos de trabalho expresso assíncronos quando não precisar de uma saída de resposta imediata, por exemplo, nos serviços de mensagens ou processamento de dados dos quais outros serviços não dependem. Você pode iniciar fluxos de trabalho expresso assíncronos em resposta a um evento, por meio de um fluxo de trabalho aninhado no Step Functions ou usando a chamada de API `StartExecution`.
- Os fluxos de trabalho expresso síncronos iniciam um fluxo de trabalho, aguardam a conclusão dele e, em seguida, retornam o resultado. Os fluxos de trabalho expresso síncronos podem ser usados para orquestrar microsserviços. Com os fluxos de trabalho expresso síncronos, você pode desenvolver aplicativos sem a necessidade de desenvolver um código adicional para lidar

com erros, novas tentativas ou executar tarefas paralelas. Você pode executar fluxos de trabalho expresso síncronos invocados pelo Amazon API Gateway, AWS Lambda, ou usando a chamada de API `StartSyncExecution`.

> **Note**
>
> Se você executar os fluxos de trabalho expresso do Step Functions de forma síncrona a partir do console, a solicitação `StartSyncExecution` expirará após 60 segundos. Para executar os fluxos de trabalho expresso de forma síncrona por até cinco minutos, faça a solicitação `StartSyncExecution` usando o AWS SDK ou AWS Command Line Interface (AWS CLI) em vez do console do Step Functions.

As chamadas de API da execução do expresso síncrono não contribuem para os limites existentes de capacidade da conta. O Step Functions fornece a capacidade sob demanda e escala automaticamente com workload sustentada. Os picos na workload podem ser reduzidos até que a capacidade esteja disponível.

Garantias de execução

Fluxos de trabalho padrão	Fluxos de trabalho expresso assíncronos	Fluxos de trabalho expresso síncronos	
Execução de fluxo de trabalho exatamente uma vez.	Execução de fluxo de trabalho pelo menos uma vez	Execução de fluxo de trabalho no máximo uma vez	
O estado de execução persiste internamente entre as transições de estado.	O estado de execução não persiste internamente entre as transições de estado.	O estado de execução não persiste internamente entre as transições de estado.	
Retorna automatic amente uma resposta idempotente ao iniciar uma execução com	A idempotência não é gerenciada automatic amente. Iniciar vários fluxos de trabalho	A idempotência não é gerenciada automaticamente. O Step Functions	

Fluxos de trabalho padrão	Fluxos de trabalho expresso assíncronos	Fluxos de trabalho expresso síncronos	
o mesmo nome de um fluxo de trabalho em execução no momento. O novo fluxo de trabalho não é iniciado e uma exceção é lançada quando o fluxo de trabalho em execução é concluído.	com o mesmo nome resulta em execuções simultâneas. Poderá resultar na perda do estado do fluxo de trabalho interno se a lógica da máquina de estado não for idempotente.	espera quando uma execução é iniciada e retorna o resultado da máquina de estado após a conclusão. Os fluxos de trabalho não são reiniciados se ocorrer uma exceção.	

Fluxos de trabalho padrão	Fluxos de trabalho expresso assíncronos	Fluxos de trabalho expresso síncronos	
Dados do histórico de execução removidos após 90 dias. Os nomes dos fluxos de trabalho podem ser reutilizados após a remoção dos dados de execução desatualizados. Para atender aos requisitos de conformidade, organizacionais ou regulamentares, você pode reduzir o período de retenção do histórico de execução para 30 dias, enviando uma solicitação de cota. Para fazer isso, use o AWS Support Center Console e crie um novo caso.	O histórico de execução não é capturado pelo Step Functions. O log deve ser ativado por meio do Amazon CloudWatch Logs.	O histórico de execução não é capturado pelo Step Functions. O log deve ser ativado por meio do Amazon CloudWatch Logs.	

Otimização de custos usando o fluxos de trabalho expressos

O Step Functions determina os preços dos fluxos de trabalho Padrão e Expressos com base no tipo de fluxo de trabalho que você usa para criar máquinas de estado. Para otimizar o custo dos fluxos de trabalho de tecnologia sem servidor, você pode seguir uma ou ambas as seguintes recomendações:

Tópicos

- Dica nº 1: agrupamento de fluxos de trabalho expressoo dentro dos fluxos de trabalho Padrão
- Dica nº 2: agrupamento de fluxos de trabalho Padrão dentro dos fluxos de trabalho expressoos

Para obter informações sobre como a escolha de um tipo de fluxo de trabalho Padrão ou Expresso afeta o faturamento, consulte Preços do AWS Step Functions.

Dica nº 1: agrupamento de fluxos de trabalho expressoo dentro dos fluxos de trabalho Padrão

O Step Functions executa fluxos de trabalho que têm uma duração e um número de etapas finitos. Alguns fluxos de trabalho podem concluir a execução em um curto período. Outros podem exigir uma combinação de fluxos de trabalho de longa duração e com alta taxa de eventos. Com o Step Functions, você pode criar fluxos de trabalho grandes e complexos a partir de vários fluxos de trabalho menores e mais simples.

Por exemplo, para criar um fluxo de trabalho de processamento de pedidos, você pode incluir todas as ações não idempotentes em um fluxo de trabalho Padrão. Isso pode incluir ações, como aprovação de pedidos por meio de interação humana e processamento de pagamentos. Em seguida, você pode combinar uma série de ações idempotentes, como enviar notificações de pagamento e atualizar o estoque de produtos, em um fluxo de trabalho expressoo. Você pode aninhar esse fluxo de trabalho expressoo no fluxo de trabalho Padrão. Neste exemplo, o fluxo de trabalho Padrão é conhecido como máquina de estado pai. O fluxo de trabalho expressoo aninhado é conhecido como máquina de estado filha.

Dica nº 2: agrupamento de fluxos de trabalho Padrão dentro dos fluxos de trabalho expressoos

Você pode converter fluxos de trabalho Padrão existentes em fluxos de trabalho expressoos se eles atenderem aos seguintes requisitos:

- O fluxo de trabalho deve concluir a execução em cinco minutos.
- O fluxo de trabalho está em conformidade com um modelo de execução do tipo pelo menos uma vez. Isso significa que cada etapa no fluxo de trabalho pode ser executada mais de uma vez.
- O fluxo de trabalho não usa os padrões de integração de serviços `.waitForTaskToken` ou `.sync`.

> **Important**
>
> Os fluxos de trabalho expressoos usam o Amazon CloudWatch Logs para registrar históricos de execução. Serão incorridos custos adicionais ao usar o CloudWatch Logs.

Para converter um fluxo de trabalho Padrão em um fluxo de trabalho expressoo usando o console

1. Abra o console do Step Functions.
2. Na página Máquinas de estado, escolha uma máquina de estado do tipo Padrão para abri-la.

   > **Tip**
   >
   > Na lista suspensa Qualquer tipo, escolha Padrão para filtrar a lista de máquinas de estado e visualizar somente fluxos de trabalho Padrão.

3. Escolha Copiar para novo.

   O Workflow Studio é aberto no Modo de design que exibe o fluxo de trabalho da máquina de estado que você selecionou.
4. (Opcional) Atualize o design do fluxo de trabalho.
5. Especifique um nome para a máquina de estado. Para fazer isso, clique no ícone de edição ao lado do nome da máquina de estado padrão MyStateMachine. Em seguida, em Configuração da máquina de estado, insira um nome na caixa Nome da máquina de estado.
6. (Opcional) Em Configuração da máquina de estado, especifique outras configurações do fluxo de trabalho, como o tipo de máquina de estado e a função de execução.

   Certifique-se de que, para Tipo, você escolha Expresso. Mantenha todas as outras seleções padrão nas Configurações da máquina de estado.

   > **Note**
   >
   > Se você estiver convertendo um fluxo de trabalho Padrão definido anteriormente em AWS CDK ou AWS SAM, você deverá alterar o valor de `Type` e o nome de `Resource`.

7. Na caixa de diálogo Confirmar criação do perfil, selecione Confirmar para continuar.

Você também pode escolher Exibir configurações do perfil para voltar às Configurações da máquina de estado.

> **Note**
>
> Se você excluir o perfil do IAM criado pelo Step Functions, não será possível recriá-lo posteriormente. Da mesma forma, se você modificar a função (por exemplo, removendo o Step Functions das entidades principais na política do IAM), o Step Functions não poderá restaurar as configurações originais dela posteriormente.

Para obter mais informações sobre as melhores práticas e diretrizes ao gerenciar a otimização de custos para os fluxos de trabalho, consulte Como criar fluxos de trabalho do AWS Step Functions econômicos.

States

Os estados individuais podem tomar decisões e executar ações com base nos dados de entrada e transmitir os dados de saída para outros estados. No AWS Step Functions, você define os fluxos de trabalho no Amazon States Language (ASL). O console do Step Functions fornece uma representação gráfica dessa máquina de estado para ajudar a visualizar a lógica do aplicativo.

> **Note**
>
> Se você definir as máquinas de estado fora do console do Step Functions, como em um editor de sua escolha, será necessário salvar as definições de máquina de estado com a extensão .asl.json.

Os estados são elementos na máquina de estado. Um estado é chamado por seu nome, que, embora possa ser qualquer string, deve ser exclusivo no escopo da máquina de estado como um todo.

Os estados podem realizar uma variedade de funções na máquina de estado:

- Realizar alguns trabalhos na máquina de estado (um estado Task)
- Escolher entre ramificações da execução (um estado Choice).

- Interromper uma execução com falha ou que tenha tido êxito (um estado Fail ou Succeed).
- Passar a entrada para a saída ou injetar alguns dados fixos no fluxo de trabalho (um estado Passagem).
- Introduzir um atraso de determinada duração ou até uma data e hora especificada (um estado Aguardar).
- Iniciar ramificações paralelas da execução (um estado Parallel).
- Iterar dinamicamente as etapas (estado de mapa)

Veja a seguir um exemplo de estado denominado `HelloWorld` que executa uma função AWS Lambda.

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

Os estados compartilham muitos recursos comuns:

- Um campo `Type` para indicar que tipo de estado ele é.
- Um campo `Comment` opcional para armazenar um comentário ou uma descrição humanamente legível do estado.
- Todo estado (exceto um estado `Succeed` ou `Fail`) exige um campo `Next` ou, de outro modo, pode se tornar um estado final por meio da especificação de um campo `End`.

> **Note**
>
> Um estado `Choice` pode ter mais de um `Next`, mas apenas um em cada Choice Rule. Um estado `Choice` não pode usar `End`.

Determinados tipos de estado exigem campos adicionais ou podem redefinir o uso de campos comuns.

Depois de criar e executar fluxos de trabalho padrão, será possível acessar informações sobre cada estado, a entrada e saída, quando e por quanto tempo ficou ativo, exibindo a página Detalhes da

execução no console do Step Functions. Para obter mais informações, consulte Visualizar e depurar execuções no console do Step Functions.

Depois de criar e executar os fluxos de trabalho expressos e se o registro em log estiver habilitado para o fluxo de trabalho expresso, é possível acessar informações sobre a execução no Amazon CloudWatch Logs ou no console do Step Functions. Para obter mais informações, consulte Visualizar e depurar execuções no console do Step Functions.

Tópicos

Amazon States Language

A Amazon States Language é uma linguagem estruturada baseada em JSON que é usada para definir a máquina de estado, um conjunto de estados, que podem realizar trabalhos (estados `Task`), determinar para quais estados fazer transição (estados `Choice`), interromper uma execução com erro (estados `Fail`) e assim por diante.

Para obter mais informações, consulte Especificação da Amazon States Language e Statelint, uma ferramenta que valida o código da Amazon States Language.

Para criar uma máquina de estado no console do Step Functions usando a Amazon States Language, consulte Conceitos básicos.

> **Note**
>
> Se você definir as máquinas de estado fora do console do Step Functions, como em um editor de sua escolha, será necessário salvar as definições de máquina de estado com a extensão .asl.json.

Exemplo de especificação da 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
    }
  }
}
```

Tópicos

Estrutura da máquina de estado

As máquinas de estado são definidas por meio de um texto JSON que representa uma estrutura que contém os campos a seguir.

`Comment` (opcional)

Uma descrição humanamente legível da máquina de estado.

`StartAt` (obrigatório)

Uma string que deve corresponder exatamente (faz distinção de maiúsculas e minúsculas) ao nome de um dos objetos de estado.

`TimeoutSeconds` (Opcional)

O número máximo de segundos que uma execução da máquina de estado pode durar. Se a execução durar mais do que o tempo especificado, ela falhará com um `States.Timeout` Nome de erro.

`Version` (opcional)

A versão da Amazon States Language usada na máquina de estado (o padrão é "1.0").

`States` (obrigatório)

Um objeto que contém um conjunto de estados separados por vírgula.

O campo `States` contém estados.

```
{
    "State1" : {
    },

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

A máquina de estado é definida pelos estados que ela contém e pelos relacionamentos entre eles.

Veja um exemplo a seguir.

```
{
  "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
    }
  }
}
```

Quando uma execução dessa máquina de estado é iniciada, o sistema começa com o estado mencionado no campo `StartAt` (`"HelloWorld"`). Se esse estado tiver um campo `"End": true`,

a execução será interrompida e retornará um resultado. Do contrário, o sistema procurará um campo `"Next":` e passará para o estado seguinte. Esse processo é repetido até que o sistema alcance um estado final (um estado com `"Type": "Succeed"`, `"Type": "Fail"` ou `"End": true`) ou até que ocorra um erro de tempo de execução.

As regras a seguir aplicam-se a estados dentro de uma máquina de estado:

- Os estados podem ocorrer em qualquer ordem dentro do bloco delimitador, mas a ordem na qual eles são listados não afeta a ordem na qual eles são executados. O conteúdo dos estados determina essa ordem.
- Dentro de uma máquina de estado, pode haver somente um estado designado como estado `start`, que é indicado pelo valor do campo `StartAt` na estrutura de nível superior. Esse é o primeiro estado executado quando a execução se inicia.
- Qualquer estado para o qual o campo `End` seja `true` é considerado um estado `end` (ou `terminal`). Dependendo da lógica da máquina de estado (por exemplo, se sua máquina de estado tiver várias ramificações de execução), você pode ter mais de um estado `end`.
- Se sua máquina de estado tiver somente um estado, esse estado poderá ser `start` ou `end`.

Funções intrínsecas

A Amazon States Language fornece várias funções intrínsecas que ajudam você a realizar operações básicas de processamento de dados sem usar um estado `Task`. Funções intrínsecas são construções que se parecem com funções em linguagens de programação. Eles podem ser usados para ajudar os criadores de carga a processar os dados que entram e saem do campo `Resource` de um estado `Task`.

Na Amazon States Language, as funções intrínsecas são agrupadas nas seguintes categorias, com base no tipo de tarefa de processamento de dados que você deseja realizar:

- Funções intrínsecas para matrizes
- Funções intrínsecas para codificação e decodificação de dados
- função intrínseca para cálculo de hash
- função intrínseca para manipulação de dados JSON
- Funções intrínsecas para operações matemáticas
- Função intrínseca para operação de strings
- Função intrínseca para geração de identificadores exclusivos

- Função intrínseca para operação genérica

> **Note**
>
> - Para usar funções intrínsecas, é necessário especificar `.$` no valor da chave nas definições da máquina de estado, conforme mostrado no exemplo a seguir:
>
> ```
> "KeyId.$": "States.Array($.Id)"
> ```
>
> - Você pode agrupar até 10 funções intrínsecas em um campo em seus fluxos de trabalho. O exemplo a seguir mostra um campo chamado *myArn* que inclui nove funções intrínsecas aninhadas:
>
> ```
> "myArn.$": "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**
>
> Se você usar o Step Functions em um ambiente de desenvolvimento local, verifique se está usando a versão 1.12.0 ou superior para poder incluir todas as funções intrínsecas em seus fluxos de trabalho.

Campos compatíveis com funções intrínsecas

As tabelas a seguir mostram quais campos são compatíveis com funções intrínsecas para cada estado.

Campos compatíveis com funções intrínsecas

State	Passagem	Tarefa	Escolha	Wait	Êxito	Falha	Paralelo	Mapa
InputPath								
Parâmetros	✓	✓					✓	✓
ResultSelector		✓					✓	✓
ResultPath								
OutputPath								
Variável								
<ComparisonOperator>Path								
TimeoutSecondsPath								
HeartbeatSecondsPath								
Credenciais		✓						

Funções intrínsecas para matrizes

Use as seguintes funções intrínsecas para realizar manipulações de matriz.

States.Array

A função intrínseca `States.Array` aceita zero ou mais argumentos. O interpretador retorna uma matriz JSON que contém os valores dos argumentos na ordem fornecida. Por exemplo, dada a seguinte entrada:

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

Você poderia usar

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

Que retornaria o seguinte resultado:

```
“BuildId”: [123456]
```

States.ArrayPartition

Use a função intrínseca `States.ArrayPartition` para particionar uma matriz grande. Você também pode usar essa função intrínseca para dividir os dados e, em seguida, enviar a carga em blocos menores.

Essa função intrínseca usa dois argumentos. O primeiro argumento é uma matriz, enquanto o segundo argumento define o tamanho do bloco. O interpretador divide a matriz de entrada em várias matrizes do tamanho especificado pelo tamanho do bloco. O comprimento do último bloco da matriz pode ser menor que o comprimento dos blocos anteriores se o número de itens restantes na matriz for menor que o tamanho do bloco.

Validação de entrada

- Você deve especificar uma matriz como o valor de entrada para o primeiro argumento da função.
- Você deve especificar um número inteiro positivo diferente de zero para o segundo argumento que representa o valor do tamanho do bloco.

  Se você especificar um valor não inteiro para o segundo argumento, o Step Functions o arredondará para o número inteiro mais próximo.

- A matriz de entrada não pode exceder o limite de tamanho de carga do Step Functions de 256 KB.

Por exemplo, dada a seguinte matriz de entrada:

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

Você pode usar a função `States.ArrayPartition` para dividir a matriz em blocos de quatro valores:

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

O que retornaria os seguintes blocos de matriz:

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

No exemplo anterior, a função `States.ArrayPartition` gera três matrizes. Cada uma das duas primeiras matrizes contém quatro valores, conforme definido pelo tamanho do bloco. Uma terceira matriz contém o valor restante e é menor que o tamanho do bloco definido.

`States.ArrayContains`

Use a função intrínseca `States.ArrayContains` para determinar se um valor específico está presente em uma matriz. Por exemplo, você pode usar essa função para detectar se houve um erro em uma iteração do estado `Map`.

Essa função intrínseca usa dois argumentos. O primeiro argumento é uma matriz, enquanto o segundo argumento é o valor a ser pesquisado dentro da matriz.

Validação de entrada

- Você deve especificar uma matriz como o valor de entrada para o primeiro argumento da função.
- Você deve especificar um objeto JSON válido como segundo argumento.
- A matriz de entrada não pode exceder o limite de tamanho de carga do Step Functions de 256 KB.

Por exemplo, dada a seguinte matriz de entrada:

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

```
   "lookingFor": 5
}
```

Você pode usar a função `States.ArrayContains` para encontrar o valor de `lookingFor` na `inputArray`:

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

Uma vez que o valor armazenado em `lookingFor` está incluído na `inputArray`, `States.ArrayContains` retorna o seguinte resultado:

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

`States.ArrayRange`

Use a função intrínseca `States.ArrayRange` para criar uma nova matriz contendo um intervalo específico de elementos. A nova matriz pode conter até 1000 elementos.

Essa função usa três argumentos. O primeiro argumento é o primeiro elemento da nova matriz, o segundo argumento é o elemento final da nova matriz e o terceiro argumento é o valor do incremento entre os elementos na nova matriz.

Validação de entrada

- É necessário especificar valores inteiros para todos os argumentos.

  Se você especificar um valor não inteiro para qualquer um dos argumentos, o Step Functions o arredondará para o número inteiro mais próximo.
- É necessário especificar um valor diferente de zero para o terceiro argumento.
- A matriz recém-gerada não pode conter mais de 1000 itens.

Por exemplo, o uso da função `States.ArrayRange` a seguir criará uma matriz com o primeiro valor de 1, um valor final de 9 e os valores entre o primeiro e o valor final aumentarão em dois para cada item:

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

O que retornaria a seguinte matriz:

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

`States.ArrayGetItem`

Essa função intrínseca retorna o valor de um índice especificado. Essa função usa dois argumentos. O primeiro argumento é uma matriz de valores e o segundo argumento é o índice da matriz do valor a ser retornado.

Por exemplo, use os seguintes valores de `inputArray` e `index`:

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

A partir desses valores, você pode usar a função `States.ArrayGetItem` para retornar o valor na posição 5 do `index` dentro da matriz:

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

Neste exemplo, `States.ArrayGetItem` deverá retornar o seguinte resultado:

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

`States.ArrayLength`

A função intrínseca `States.ArrayLength` retorna o comprimento de uma matriz. Ele tem um argumento: retornar o comprimento da matriz.

Por exemplo, dada a seguinte matriz de entrada:

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

Você pode usar `States.ArrayLength` para retornar o comprimento da `inputArray`:

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

Neste exemplo, `States.ArrayLength` deverá retornar o seguinte objeto JSON que representa o comprimento da matriz:

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

States.ArrayUnique

A função intrínseca `States.ArrayUnique` remove valores duplicados de uma matriz e retorna uma matriz contendo somente elementos exclusivos. Essa função usa uma matriz, que pode ser desclassificada, como seu único argumento.

Por exemplo, a seguinte `inputArray` contém uma série de valores duplicados:

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

Você pode usar a função `States.ArrayUnique` e especificar a matriz da qual deseja remover valores duplicados:

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

A função `States.ArrayUnique` retornaria a seguinte matriz contendo somente elementos exclusivos, removendo todos os valores duplicados:

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

Funções intrínsecas para codificação e decodificação de dados

Use as seguintes funções intrínsecas para codificar ou decodificar dados com base no esquema de codificação Base64.

States.Base64Encode

Use a função intrínseca `States.Base64Encode` para codificar dados com base no esquema de codificação MIME Base64. Você pode usar essa função para transmitir dados para outros serviços da AWS sem usar uma função do AWS Lambda.

Essa função usa uma sequência de dados de até 10.000 caracteres para codificar como seu único argumento.

Por exemplo, considere a seguinte string de `input`:

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

Você pode usar a função `States.Base64Encode` para codificar a string de `input` como uma string MIME Base64:

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

A função `States.Base64Encode` retorna os seguintes dados codificados como resposta:

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

`States.Base64Decode`

Use a função intrínseca `States.Base64Decode` para decodificar dados com base no esquema de decodificação MIME Base64. Você pode usar essa função para transmitir dados para outros serviços da AWS sem usar uma função do Lambda.

Essa função usa uma sequência de dados codificados com Base64 de até 10.000 caracteres para decodificar como seu único argumento.

Por exemplo, dada a seguinte entrada:

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

Você pode usar a função `States.Base64Decode` para decodificar a string base64 em uma string legível:

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

A `States.Base64Decode function` retornaria os seguintes dados decodificados em resposta:

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

função intrínseca para cálculo de hash

`States.Hash`

Use a função intrínseca `States.Hash` para calcular o valor de hash de uma determinada entrada. Você pode usar essa função para transmitir dados para outros serviços da AWS sem usar uma função do Lambda.

Essa função usa dois argumentos. O primeiro argumento são os dados dos quais você deseja calcular o valor de hash. O segundo argumento é o algoritmo de hash a ser usado para realizar o cálculo de hash. Os dados fornecidos devem ser uma sequência de objetos contendo 10.000 caracteres ou menos.

O algoritmo de hash especificado pode ser qualquer um dos seguintes algoritmos:

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

Por exemplo, você pode usar essa função para calcular o valor de hash da string `Data` usando o `Algorithm` especificado:

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

Você pode usar a função `States.Hash` para calcular o valor do hash:

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

A função do `States.Hash` retorna o seguinte valor de hash em resposta:

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

função intrínseca para manipulação de dados JSON

Use essas funções para realizar operações básicas de processamento de dados em objetos JSON.

`States.JsonMerge`

Use a função intrínseca `States.JsonMerge` para mesclar dois objetos JSON em um único objeto. Essa função usa três argumentos. Os dois primeiros argumentos são os objetos JSON que você deseja mesclar. O terceiro argumento é um valor booleano de `false`. Esse valor booleano determina se o modo de mesclagem profunda está ativado.

Atualmente, o Step Functions é compatível apenas com o modo de mesclagem superficial; portanto, você deve especificar o valor booleano como `false`. No modo superficial, se a mesma chave existir nos dois objetos JSON, a chave do último objeto substituirá a mesma chave no primeiro objeto. Além disso, objetos aninhados em um objeto JSON não são mesclados ao usar a mesclagem superficial.

Por exemplo, você pode usar a função `States.JsonMerge` para mesclar os seguintes objetos JSON que compartilham a chave `a`.

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

Você pode especificar os objetos json1 e json2 como entradas na função `States.JsonMerge` para mesclá-los:

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

Como resultado, `States.JsonMerge` retorna o seguinte objeto JSON mesclado. Na `output` do objeto JSON mesclado, a chave `a` do objeto `json2` substitui a chave `a` do objeto `json1`. Além disso, o objeto aninhado na chave `a` do objeto `json1` é descartado porque o modo superficial não é compatível com mesclagem de objetos aninhados.

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

`States.StringToJson`

A função `States.StringToJson` usa um caminho de referência para uma string JSON com escape como seu único argumento.

O interpretador aplica um analisador JSON e retorna o formulário JSON analisado da entrada. Por exemplo, você pode usar essa função para efetuar o escape da seguinte string de entrada:

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

Use a função `States.StringToJson` e especifique a `escapedJsonString` como argumento de entrada:

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

A função `States.StringToJson` retorna o seguinte resultado:

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

`States.JsonToString`

A função `States.JsonToString` usa apenas um argumento, que é o caminho que contém os dados JSON a serem retornados como uma string sem escape. O interpretador retorna uma string que contém texto JSON representando os dados especificados pelo caminho. Por exemplo, você pode fornecer o seguinte caminho JSON contendo um valor de escape:

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

Forneça à função `States.JsonToString` os dados contidos em `unescapedJson`:

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

A função `States.JsonToString` retorna a seguinte resposta:

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

Funções intrínsecas para operações matemáticas

Use essas funções para realizar operações matemáticas.

`States.MathRandom`

Use a função intrínseca `States.MathRandom` para retornar um número aleatório entre o número inicial especificado (inclusivo) e o número final (exclusivo).

Você pode usar essa função para distribuir uma tarefa específica entre dois ou mais recursos.

Essa função usa três argumentos. O primeiro argumento é o número inicial, o segundo argumento é o número final e o último argumento controla o valor de seed. O argumento do valor de seed é opcional. Se você usar essa função com o mesmo valor de seed, ela retornará um número idêntico.

> **Important**
>
> Uma vez que a função `States.MathRandom` não retorna números aleatórios com criptografia segura, recomendamos que você não a use para aplicativos com informações confidenciais.

Validação de entrada

- É necessário especificar valores inteiros para os argumentos do número inicial e do número final.

  Se você especificar um valor não inteiro para o argumento do número inicial ou final, o Step Functions o arredondará para o número inteiro mais próximo.

Por exemplo, para gerar um número aleatório entre um e 999, você pode usar os seguintes valores de entrada:

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

Para gerar o número aleatório, forneça os valores `start` e `end` para a função `States.MathRandom`:

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

A função `States.MathRandom` retorna o seguinte número aleatório como resposta:

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

States.MathAdd

Use a função intrínseca `States.MathAdd` para retornar a soma de dois números. Por exemplo, você pode usar essa função para incrementar valores dentro de um loop sem invocar uma função do Lambda.

Validação de entrada

- É necessário especificar valores inteiros para todos os argumentos.

  Se você especificar um valor não inteiro para um ou ambos os argumentos, o Step Functions o arredondará para o número inteiro mais próximo.
- É necessário especificar valores inteiros no intervalo de -2147483648 e 2147483647.

Por exemplo, você pode usar os seguintes valores para subtrair um de 111:

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

Em seguida, usar a função `States.MathAdd` definindo `value1` como o valor inicial e `step` como o valor de aumento do `value1`:

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

A função `States.MathAdd` retornaria o seguinte número como resposta:

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

Função intrínseca para operação de strings

States.StringSplit

Use a função intrínseca `States.StringSplit` para dividir uma string em uma matriz de valores. Essa função usa dois argumentos. O primeiro argumento é uma string e o segundo argumento é o caractere delimitador que a função usará para dividir a string.

Example - Divida uma string de entrada usando um único caractere delimitador

Neste exemplo, use `States.StringSplit` para dividir a seguinte `inputString`, que contém uma série de valores separados por vírgula:

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

Use a função `States.StringSplit` e defina `inputString` como o primeiro argumento e o caractere delimitador `splitter` como o segundo argumento:

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

A função `States.StringSplit` retorna a seguinte matriz de string como resultado:

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

Example - Divida uma string de entrada usando vários caracteres delimitadores

Neste exemplo, use `States.StringSplit` para dividir a seguinte `inputString`, que contém vários caracteres delimitadores:

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

Use a função `States.StringSplit` da seguinte forma:

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

A função `States.StringSplit` retorna a seguinte matriz de string como resultado:

```
{"myStringArray": [
```

```
  "This",
  "is",
  "a",
  "test",
  "string"
]}
```

Função intrínseca para geração de identificadores exclusivos

`States.UUID`

Use a função intrínseca `States.UUID` para retornar um identificador exclusivo universal da versão 4 (UUID v4) gerado usando números aleatórios. Por exemplo, você pode usar essa função para chamar outros serviços ou recursos da AWS que precisam de um parâmetro UUID ou inserir itens em uma tabela do DynamoDB.

A função `States.UUID` é chamada sem nenhum argumento especificado:

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

A função retorna um UUID gerado aleatoriamente, como no exemplo a seguir:

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

Função intrínseca para operação genérica

`States.Format`

Use a função intrínseca `States.Format` para construir uma string a partir de valores literais e interpolados. Essa função usa um ou mais argumentos. O valor do primeiro argumento deve ser uma string e pode incluir zero ou mais instâncias da sequência de caracteres `{}`. É necessário que haja o mesmo número de argumentos restantes na invocação da função intrínseca que o número de ocorrências de `{}`. O interpretador retorna a string definida no primeiro argumento, com cada `{}` substituída pelo valor do argumento correspondente posicionalmente na invocação da função intrínseca.

Por exemplo, você pode usar as seguintes entradas do `name` de uma pessoa e uma frase `template` para inserir o nome dela:

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

Use a função `States.Format` e especifique a string `template` e a string a ser inserida no lugar dos caracteres `{}`:

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

or

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

Com qualquer uma das entradas anteriores, a função `States.Format` retorna a string completa como resposta:

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

Caracteres reservados em funções intrínsecas

Os caracteres a seguir são reservados para funções intrínsecas e seu escape deve ser efetuado com uma barra invertida ('\') se você quiser que eles apareçam no Valor: '{} e \.

Se o caractere \ precisar aparecer como parte do valor sem servir como um caractere de escape, você deverá efetuar o escape dele com uma barra invertida. As seguintes sequências de caracteres com escape são usadas com funções intrínsecas:

- A string literal `\'` representa `'`.
- A string literal `\{` representa `{`.
- A string literal `\}` representa `}`.
- A string literal `\\` representa `\`.

Em JSON, o escape de barras invertidas contidas em um valor literal de string deve ser efetuado com outra barra invertida. A lista equivalente para JSON é:

- A string de escape `\\\'` representa `\'`.

- A string de escape `\\\{` representa `\{`.
- A string de escape `\\\}` representa `\}`.
- A string de escape `\\\\` representa `\\`.

> **Note**
>
> Se uma barra invertida de escape aberta `\` for encontrada na string de invocação da função intrínseca, o interpretador retornará um erro de runtime.

Campos de estado comuns

`Type` (obrigatório)

O tipo de estado.

`Next`

O nome do próximo estado que será executado quando o estado atual for concluído. Alguns tipos de estado, como `Choice`, permitem vários estados de transição.

Se o estado atual for o último estado no fluxo de trabalho ou um estado terminal, como Succeed ou Fail, não será necessário especificar o campo `Next`.

`End`

Designa o estado como um estado terminal (encerra a execução) caso seja definido como `true`. Pode haver qualquer quantidade de estados finais por máquina de estado. Apenas um `Next` ou `End` pode ser usado em um estado. Alguns tipos de estado, como `Choice`, ou estados terminais, como Succeed eFail, não são compatíveis nem usam o campo `End`.

`Comment` (opcional)

Apresenta uma descrição humanamente legível da máquina de estado.

`InputPath` (opcional)

Um caminho que seleciona uma parte da entrada do estado a ser passada para a tarefa do estado para processamento. Se omitido, terá o valor $, que designa a entrada completa. Para obter mais informações, consulte Processamento de entrada e saída.

`OutputPath` (opcional)

Um caminho que seleciona uma parte da saída do estado a ser transmitida para o próximo estado. Se omitido, terá o valor $, que designa a saída completa. Para obter mais informações, consulte Processamento de entrada e saída.

Pass

Um estado `Pass` (`"Type": "Pass"`) passa sua entrada para sua saída, sem executar o trabalho. Os estados `Pass` são úteis na construção e na depuração de máquinas de estado.

Você também pode usar um estado `Pass` para transformar a entrada de estado JSON usando filtros e, em seguida, transmitir os dados transformados para o próximo estado nos fluxos de trabalho. Para ver mais informações sobre a transformação de entrada, consulte InputPath, Parâmetros e ResultSelector.

Além dos campos de estado comuns, os estados `Pass` permitem os campos a seguir.

`Result` (opcional)

Refere-se à saída de uma tarefa virtual que é transmitida para o próximo estado. Se você incluir o campo `ResultPath` na definição da máquina de estado, o `Result` será colocado conforme especificado pelo `ResultPath` e enviado para o próximo estado.

`ResultPath` (opcional)

Especifica onde colocar a saída (em relação à entrada) da tarefa virtual especificada no `Result`. A entrada é filtrada adicionalmente, conforme especificado pelo campo `OutputPath` (se houver) antes de ser usada como a saída do estado. Para obter mais informações, consulte Processamento de entrada e saída.

`Parameters` (opcional)

Cria um conjunto de pares de chave-valor que será transmitido como entrada. Você pode especificar `Parameters` como um valor estático ou selecionar a partir da entrada usando um caminho. Para obter mais informações, consulte InputPath, Parâmetros e ResultSelector.

Exemplo de estado Pass

Veja a seguir um exemplo de um estado `Pass` que injeta alguns dados fixos na máquina de estado, provavelmente para finalidade de teste.

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

Vamos supor que a entrada para esse estado seja a seguinte.

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

A saída seria esta.

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

Estado da tarefa

Um estado `Task` (`"Type": "Task"`) representa uma unidade de trabalho específica executada por uma máquina de estado. Uma tarefa executa o trabalho usando uma atividade ou AWS Lambda função, integrando-se a outras suportadas Serviços da AWS ou invocando uma API de terceiros, como o Stripe.

A Amazon States Language representa tarefas definindo o tipo de um estado como `Task` e fornecendo à tarefa o nome do recurso da Amazon (ARN) da atividade, da função do Lambda ou do endpoint de API de terceiros. A seguinte definição de estado Tarefa invoca uma função do Lambda chamada *`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
}
```

Neste tópico

Tipos de tarefa

O Step Functions é compatível com os seguintes tipos de tarefa que você pode especificar em uma definição de estado Tarefa.

- Atividades
- Funções do Lambda
- Um suportado AWS service (Serviço da AWS)
- Uma tarefa HTTP

Você especifica um tipo de tarefa fornecendo seu ARN no campo `Resource` da definição do estado Tarefa. O exemplo a seguir mostra a sintaxe do campo `Resource`. Todos os tipos de tarefa, exceto aquele que invoca uma API de terceiros, usam a sintaxe a seguir. Para obter informações sobre a sintaxe da tarefa HTTP, consulte Chamar APIs de terceiros.

Na definição do estado da tarefa, substitua o texto em itálico na sintaxe a seguir pelas informações específicas do AWS recurso.

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

A lista a seguir explica os componentes individuais dessa sintaxe.

- `partition`é a AWS Step Functions partição a ser usada, mais comumente`aws`.
- `service`indica o AWS service (Serviço da AWS) usado para executar a tarefa e pode ser um dos seguintes valores:
  - `states` para uma atividade.
  - `lambda` para uma função do Lambda. Se você se integrar com outros Serviços da AWS, por exemplo, Amazon SNS ou Amazon DynamoDB, use ou. `sns` `dynamodb`
- `region`é o código da AWS região em que a atividade Step Functions ou o tipo de máquina de estado, a função Lambda ou qualquer outro AWS recurso foi criado.
- `account`é o Conta da AWS ID no qual você definiu o recurso.
- `task_type` é o tipo de tarefa a ser executada. Pode ter um dos valores a seguir:
  - `activity` – Uma atividade.
  - `function` – Uma função do Lambda.
  - *`servicename`* – O nome de um serviço conectado compatível (consulte Integrações otimizadas para o Step Functions).
- `name` é o nome de recurso registrado (nome da atividade, nome da função do Lambda ou ação de API de serviço).

> **Note**
>
> O Step Functions não é compatível com a referência de ARNs entre partições ou regiões. Por exemplo, `aws-cn` não consegue invocar tarefas na partição da `aws` e vice-versa.

As seções a seguir oferecem mais detalhes sobre cada tipo de tarefa.

Atividade

As atividades representam operadores (processos ou threads), implementados e hospedados por você, que executam uma tarefa específica. Eles oferecem suporte apenas de fluxos de trabalho padrão, não de expressos.

Os ARNS de `Resource` de atividade usam a sintaxe a seguir.

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

> **Note**
>
> Você deve criar atividades com Step Functions (usando uma CreateActivityação de API ou o console Step Functions) antes de usá-las pela primeira vez.

Para obter mais informações sobre a criação de uma atividade e a implementação de operadores, consulte Atividades.

Funções do Lambda

As tarefas Lambda executam uma função usando o. AWS Lambda Para especificar uma função do Lambda, use o ARN da função do Lambda no campo `Resource`.

Dependendo do tipo de integração (integração otimizada ou integração do SDK da AWS) que você usa para especificar uma função do Lambda, a sintaxe do campo `Resource` da função do Lambda varia.

A seguinte sintaxe do campo `Resource` é um exemplo de uma integração otimizada com uma função do Lambda.

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

A sintaxe de `Resource` campo a seguir é um exemplo de integração do AWS SDK com uma função Lambda.

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

A definição de estado `Task` a seguir mostra um exemplo de uma integração otimizada com uma função do Lambda chamada *`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"
```

```
}
```

Depois que a função Lambda especificada no `Resource` campo for concluída, sua saída será enviada para o estado identificado no `Next` campo (" "). NextState

Um suportado AWS service (Serviço da AWS)

Ao fazer referência a um recurso conectado, o Step Functions chama diretamente as ações de API de um serviço compatível. Especifique o serviço e a ação no campo `Resource`.

Os ARNS de `Resource` do serviço conectado usam a sintaxe a seguir.

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

> **Note**
>
> Para criar uma conexão síncrona com um recurso conectado, acrescente `.sync` à entrada *`APIname`* no ARN. Para ter mais informações, consulte Como trabalhar com outros serviços.

Por exemplo: .

```
{
 "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 do estado Tarefa

Além dos campos de estado comuns, os estados `Task` têm os campos a seguir.

`Resource` (obrigatório)

Um URI, especialmente um ARN que identifica exclusivamente a tarefa específica a ser executada.

`Parameters` (Opcional)

Usado para passar informações para as ações de API de recursos conectados. Os parâmetros podem usar uma combinação de JSON estático e. JsonPath Para ter mais informações, consulte Transmitir parâmetros para uma API de serviço.

`Credentials` (Opcional)

Especifica um perfil de destino que o perfil de execução da máquina de estado deve assumir antes de invocar o `Resource` especificado. Como alternativa, você também pode especificar um valor JSONPath ou uma função intrínseca que se resolve como um ARN de perfil do IAM no runtime com base na entrada de execução. Se você especificar um valor JSONPath, será necessário prefixá-lo com a notação `$.`.

Para obter exemplos de uso desse campo no estado `Task`, consulteExemplos do campo Credenciais do estado Tarefa. Para ver um exemplo de como usar esse campo para acessar um AWS recurso entre contas a partir de sua máquina de estado, consulteTutorial: Acessando recursos entre contas AWS.

> **Note**
>
> Esse campo é suportado por aqueles Tipos de tarefa que usam funções Lambda e um serviço compatível AWS.

`ResultPath` (Opcional)

Especifica onde (na entrada) inserir os resultados da execução da tarefa que é especificada em `Resource`. A entrada é então filtrada conforme especificado pelo campo `OutputPath` (se houver) antes de ser usada como a saída do estado. Para obter mais informações, consulte Processamento de entrada e saída.

`ResultSelector` (Opcional)

Transmitir um conjunto de pares de valores-chave, em que os valores são estáticos ou selecionados a partir do resultado. Para ter mais informações, consulte ResultSelector.

`Retry` (Opcional)

Uma matriz de objetos, chamados Retriers, que definem uma política de novas tentativas caso o estado encontre erros de tempo de execução. Para ter mais informações, consulte Exemplos de máquina de estado que usam Nova tentativa e Detecção.

`Catch` (Opcional)

Uma matriz de objetos, chamados Catchers, que definem um estado de fallback. Esse estado é executado caso o estado encontre erros de tempo de execução e sua política de novas tentativas esteja esgotada ou não tenha sido definida. Para obter mais informações, consulte Estados de fallback.

`TimeoutSeconds` (Opcional)

Especifica o tempo máximo que uma atividade ou tarefa pode ser executada antes que ela atinja o tempo limite com o erro States.Timeout e falhe. O valor do tempo limite deve ser um número inteiro positivo e diferente de zero. O valor padrão é 99999999.

A contagem do tempo limite começa após uma tarefa ser iniciada, por exemplo, quando eventos `ActivityStarted` ou `LambdaFunctionStarted` são registrados no histórico de eventos de execução. Para as atividades, a contagem começa quando `GetActivityTask` recebe um token e `ActivityStarted` é registrado no histórico de eventos de execução.

Quando uma tarefa é iniciada, o Step Functions espera por uma resposta de conclusão ou falha do operário da tarefa ou atividade dentro da duração de `TimeoutSeconds` especificada. Se o operário da tarefa ou da atividade não responder dentro desse período, o Step Functions marcará a execução do fluxo de trabalho como com falha.

`TimeoutSecondsPath` (Opcional)

Para fornecer dinamicamente um valor de tempo limite a partir da entrada de estado usando um caminho de referência, use `TimeoutSecondsPath`. Quando resolvido, o caminho de referência deve selecionar campos cujos valores sejam números inteiros positivos.

> **Note**
>
> Um estado `Task` não pode incluir `TimeoutSeconds` e `TimeoutSecondsPath`.

`HeartbeatSeconds` (Opcional)

Determina a frequência dos sinais de pulsação que um operário da atividade envia durante a execução de uma tarefa. As pulsações indicam que uma tarefa ainda está em execução e precisa de mais tempo para ser concluída. As pulsações evitam que uma atividade ou tarefa atinja o tempo limite durante o tempo de `TimeoutSeconds`.

`HeartbeatSeconds` deve ser um valor inteiro positivo, diferente de zero, menor que o valor do campo `TimeoutSeconds`. O valor padrão é 99999999. Se decorrer um tempo maior do que os segundos especificados entre as pulsações da tarefa, o estado Tarefa apresentará uma falha com o erro States.Timeout.

Para as atividades, a contagem começa quando `GetActivityTask` recebe um token e `ActivityStarted` é registrado no histórico de eventos de execução.

`HeartbeatSecondsPath` (Opcional)

Para fornecer dinamicamente um valor de pulsação a partir da entrada de estado usando um caminho de referência, use `HeartbeatSecondsPath`. Quando resolvido, o caminho de referência deve selecionar campos cujos valores sejam números inteiros positivos.

> **Note**
>
> Um estado `Task` não pode incluir `HeartbeatSeconds` e `HeartbeatSecondsPath`.

Um estado `Task` deverá definir o campo `End` como `true` se o estado encerrar a execução, ou deverá fornecer um estado no campo `Next` que será executado quando o estado `Task` for concluído.

Exemplos de definição de estado Tarefa

Os exemplos a seguir mostram como especificar a definição do estado Tarefa com base na sua necessidade.

- Especificar tempos limite e intervalos de pulsação do estado Tarefa
  - Exemplo de tempo limite estático e notificação de pulsação
  - Exemplo de tempo limite de tarefa dinâmica e notificação de pulsação
- Usar o campo Credenciais

- Especificar o ARN do perfil do IAM com codificação rígida
- Especificar o JSONPath como ARN do perfil do IAM
- Especificar uma função intrínseca como ARN do perfil do IAM

Tempos limite e intervalos de pulsação do estado Tarefa

É recomendável definir um valor de tempo limite e um intervalo de pulsação para atividades de longa duração. Isso pode ser feito especificando os valores de tempo limite e pulsação ou definindo-os dinamicamente.

Exemplo de tempo limite estático e notificação de pulsação

Quando `HelloWorld` for concluída, o próximo estado (chamado aqui de `NextState`) será executado.

Se essa tarefa não for concluída em 300 segundos ou não enviar notificações de pulsação em intervalos de 60 segundos, ela será marcada como `failed`.

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

Exemplo de tempo limite de tarefa dinâmica e notificação de pulsação

Neste exemplo, quando o AWS Glue trabalho for concluído, o próximo estado será executado.

Se essa tarefa não for concluída dentro do intervalo definido dinamicamente pelo trabalho AWS Glue , a tarefa será marcada como `failed`.

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

```
  "TimeoutSecondsPath": "$.params.maxTime",

  "Next": "NextState"
}
```

Exemplos do campo Credenciais do estado Tarefa

Especificar o ARN do perfil do IAM com codificação rígida

O exemplo a seguir especifica um perfil do IAM de destino que o perfil do IAM de execução de uma máquina de estado deve assumir para acessar uma função do Lambda entre contas chamada `Echo`. Neste exemplo, o ARN do perfil de destino é especificado como um valor com codificação 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 o JSONPath como ARN do perfil do IAM

O exemplo a seguir especifica um valor JSONPath, que será resolvido em um ARN do perfil do IAM em runtime.

```
{
  "StartAt": "Lambda",
  "States": {
    "Lambda": {
      "Type": "Task",
      "Resource": "arn:aws:states:::lambda:invoke",
      "Credentials": {
```

```
        "RoleArn.$": "$.roleArn"
      },
      ...
    }
  }
}
```

Especificar uma função intrínseca como ARN do perfil do IAM

O exemplo a seguir usa a função intrínseca `States.Format`, que será resolvida em um ARN do perfil do IAM em runtime.

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

Atividades

As atividades são um atributo do AWS Step Functions que permite que você tenha uma tarefa na máquina de estado em que o trabalho é executado por um operador que pode ser hospedado no Amazon Elastic Compute Cloud (Amazon EC2), no Amazon Elastic Container Service (Amazon ECS) e em dispositivos móveis, praticamente em qualquer lugar.

Visão geral

No AWS Step Functions, as atividades são uma maneira de associar um código em execução em algum lugar (conhecido como um operador de atividade) a uma tarefa específica em uma máquina de estado. Você pode criar uma atividade usando o console do Step Functions ou chamando `CreateActivity`. Isso fornece um nome do recurso da Amazon (ARN) para o estado Tarefa. Use esse ARN para fazer uma sondagem do estado da tarefa para trabalhos no operador de atividade.

> **Note**
>
> As atividades não são versionadas e devem sempre ter compatibilidade com versões anteriores. Se for necessário fazer uma alteração incompatível com versões anteriores em uma atividade, crie uma nova atividade no Step Functions usando um nome exclusivo.

Um operador de atividade pode ser um aplicativo em execução em uma instância do Amazon EC2, uma função do AWS Lambda, um dispositivo móvel: qualquer aplicativo que possa estabelecer uma conexão HTTP, hospedado em qualquer lugar. Quando o Step Functions atinge um estado de tarefa de atividade, o fluxo de trabalho aguarda um operador de atividade para fazer a sondagem de uma tarefa. Um operador de atividade faz uma sondagem do Step Functions usando `GetActivityTask` e enviando o ARN da atividade relacionada. `GetActivityTask` retorna uma resposta incluindo a `input` (uma string de entrada JSON para a tarefa) e um `taskToken` (um identificador exclusivo para a tarefa). Após o operador de atividade concluir seu trabalho, ele pode fornecer um relatório de êxito ou falha usando `SendTaskSuccess` ou `SendTaskFailure`. Essas duas chamadas usam o `taskToken` fornecido por `GetActivityTask` para associar o resultado a essa tarefa.

APIs relacionadas a tarefas de atividade

O Step Functions proporciona APIs para criar e listar atividades, solicitar uma tarefa e gerenciar o fluxo da máquina de estado com base nos resultados do operador.

Veja a seguir as APIs do Step Functions que estão relacionadas a atividades:

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

> **Note**
>
> A sondagem de tarefas de atividade com `GetActivityTask` pode causar latência em algumas implementações. Consulte Evitar latência ao fazer uma sondagem de tarefas de atividade.

Aguardar a conclusão de uma tarefa de atividade

Configure por quanto tempo um estado deve aguardar definindo `TimeoutSeconds` na definição da tarefa. Para manter a tarefa ativa e em espera, envie uma pulsação periodicamente do operador de atividade usando `SendTaskHeartbeat` dentro do tempo configurado em `TimeoutSeconds`. Ao configurar uma longa duração do tempo limite e enviar uma pulsação de maneira ativa, uma atividade no Step Functions pode aguardar até um ano para uma execução ser concluída.

Por exemplo, se você precisa de um fluxo de trabalho que aguarda a saída de um longo processo, faça o seguinte:

1. Crie uma atividade usando o console ou `CreateActivity`. Anote o ARN da atividade.
2. Faça referência a esse ARN em um estado de tarefa de atividade na definição da máquina de estado e defina `TimeoutSeconds`
3. Implemente um operador de atividade que faça uma sondagem de trabalhos usando `GetActivityTask`, fazendo referência a esse ARN de atividade.
4. Use `SendTaskHeartbeat` periodicamente dentro do período que você definiu em `HeartbeatSeconds` na definição da tarefa da máquina de estado para evitar que a tarefa expire.
5. Inicie uma execução da máquina de estado.
6. Inicie um processo do operador de atividade.

A execução coloca o estado de tarefa da atividade em pausa e aguarda que o operador de atividade faça sondagem de uma tarefa. Assim que um `taskToken` é fornecido para o operador de atividade, seu fluxo de trabalho aguardará `SendTaskSuccess` ou `SendTaskFailure` para fornecer um status. Se a execução não receber nenhum deles ou receber uma chamada `SendTaskHeartbeat` antes do tempo configurado em `TimeoutSeconds`, ocorrerá uma falha na execução, e o histórico de execução conterá um evento `ExecutionTimedOut`.

Próximas etapas

Para obter uma análise mais detalhada da criação de máquinas de estado que usam operadores de atividade, consulte:

- Como criar uma máquina de estado de Atividade usando o Step Functions
- Exemplo de operador de atividade em Ruby

Exemplo de operador de atividade em Ruby

Veja a seguir um exemplo de operador de atividade que usa o AWS SDK for Ruby para mostrar como usar as práticas recomendadas e implementar seu próprio operador de atividade.

O código implementa um padrão de produtor-consumidor com um número configurável de threads para agentes de sondagem e operadores de atividade. Os threads dos agentes de sondagem detectam longamente, de forma constante, a tarefa de atividade. Assim que uma tarefa de atividade é recuperada, ela é transmitida por meio de uma fila de bloqueio delimitada para ser capturada pelo thread de atividade.

- Para ver mais informações sobre o AWS SDK for Ruby, consulte a Referência de API AWS SDK for Ruby.
- Para fazer download desse código e de recursos relacionados, consulte o repositório step-functions-ruby-activity-worker no GitHub.

O código Ruby a seguir é o ponto de entrada principal para esse operador de atividade do Ruby de exemplo.

```
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
```

O código inclui valores padrão que você pode alterar e usar como referência para a atividade e adaptar para sua implementação específica. Esse código usa a lógica de implementação real como entrada, permite que você faça referência a suas atividades e credenciais específicas e permite que você configure o número de threads e o atraso de pulsação. Para ver mais informações e fazer download do código, consulte Ruby Activity Worker do Step Functions.

Item	Descrição
`require_relative`	Caminho relativo para o seguinte código do operador de atividade de exemplo.
`region`	Região da AWS da atividade.
`workers_count`	O número de threads para o operador de atividade. Para a maioria das implementações, entre 10 e 20 threads deveria ser suficient e. Quanto mais tempo a atividade demorar para ser processada, maior será o número de threads que ela pode precisar. Como estimativ a, multiplique o número de atividades de processos por segundo pela latência do 99º percentil do processamento da atividade em segundos.
`pollers_count`	O número de threads dos seus agentes de sondagem. Entre 10 e 20 threads deveriam ser suficientes para a maioria das implementações.
`heartbeat_delay`	O atraso em segundos entre as pulsações.
`input`	Lógica de implementação da sua atividade.

O seguinte é o operador de atividade do Ruby, referenciado com `../lib/step_functions/activity` no 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

Um estado `Choice` (`"Type": "Choice"`) adiciona a lógica condicional a uma máquina de estado.

Além da maioria dos campos de estado comuns, os estados `Choice` contêm os campos adicionais a seguir.

`Choices` (obrigatório)

Um conjunto Choice Rules que determina para qual estado seguinte a máquina de estado deve mudar. Você usa um operador de comparação em uma Regra de escolha para comparar uma variável de entrada com um valor específico. Por exemplo, usando Regras de escolha, é possível comparar se uma variável de entrada é maior ou menor que 100.

Quando um estado `Choice` é executado, ele avalia cada Regra de escolha como verdadeiro ou falso. Com base no resultado dessa avaliação, o Step Functions muda para o próximo estado no fluxo de trabalho.

É necessário definir pelo menos uma regra no estado `Choice`.

`Default` (Opcional, Recomendado)

Nome do estado para o qual deve mudar se nenhuma das transições em `Choices` for feita.

> **Important**
>
> Os estados `Choice` não oferecem suporte ao campo `End`. Além disso, eles usam `Next` somente dentro do campo `Choices`.

> **Tip**
>
> Para implantar um exemplo de fluxo de trabalho que usa um estado `Choice` em sua Conta da AWS, consulte Módulo 5 - Estado Escolha e estado Mapa do workshop do AWS Step Functions.

Choice Rules

Um estado `Choice` deve ter um campo `Choices` cujo valor seja uma matriz não vazia. Cada elemento dessa matriz é um objeto chamado Regra de escolha, que contém o seguinte:

- Uma comparação: dois campos que especificam uma variável de entrada para comparação, o tipo de comparação e o valor a ser comparado com a variável. As Regras de escolha permitem a comparação entre duas variáveis. Dentro de uma Regra de escolha, o valor da variável pode ser comparado com outro valor da entrada de estado anexando `Path` ao nome dos operadores de comparação compatíveis. Os valores dos campos `Variable` e Caminho em uma comparação devem ser caminhos de referência válidos.
- Um campo **`Next`**: o valor desse campo deve corresponder a um nome de estado na máquina de estado.

O exemplo a seguir verifica se o valor numérico é igual a `1`.

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

O exemplo a seguir verifica se a string é igual a `MyString`.

```
{
```

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

O exemplo a seguir verifica se a string é maior que `MyStringABC`.

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

O exemplo a seguir verifica se a string é nula.

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

O exemplo a seguir mostra como a regra StringEquals só é avaliada quando `$.keyThatMightNotExist` existe devido à Regra de escolha `IsPresent` anterior.

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

O exemplo a seguir verifica se um padrão com um caractere curinga corresponde.

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

O exemplo a seguir verifica se o timestamp é igual a `2001-01-01T12:00:00Z`.

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

O exemplo a seguir compara uma variável com outro valor da entrada de estado.

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

O Step Functions examina cada uma das Regra de escolha na ordem listada no campo `Choices`. Depois disso, ele faz a transição para o estado especificado no campo `Next` do primeiro Choice Rule em que a variável corresponde ao valor, de acordo com o operador de comparação.

Os operadores de comparação a seguir são comportados:

- `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`

Para cada um desses operadores, o valor correspondente deve ser do tipo apropriado: string, número, booliano ou time stamp. O Step Functions não tenta equiparar um campo numérico com um valor de string. No entanto, como os campos de timestamp são logicamente strings, é possível que um campo considerado um timestamp corresponda a um comparador `StringEquals`.

> **Note**
>
> Por motivo de interoperabilidade, não presuma que as comparações numéricas funcionarão com valores fora da magnitude ou precisão que o tipo de dado `binary64` de IEEE 754-2008 representa. Mais especificamente, é provável que os inteiros fora do intervalo $[-2^{53}+1, 2^{53}-1]$ não sejam comparados da forma esperada.
> Os time stamps (por exemplo, `2016-08-18T17:33:00Z`) devem estar de acordo com o perfil RFC3339 da ISO 8601 e apresentar outras restrições:
>
> - Um `T` maiúsculo deve separar as partes de data e hora.
> - Um `Z` maiúsculo deve indicar que não existe uma compensação de fuso horário numérica.
>
> Para entender o comportamento das comparações de strings, consulte a documentação do `JavacompareTo`.
> Os valores dos operadores `And` e `Or` devem ser matrizes não vazias de Choice Rules que em si não devem conter campos `Next`. Da mesma forma, o valor de um operador `Not` deve ser um Choice Rule único que não deve conter campos `Next`.

Você pode criar Choice Rules complexos e aninhados usando `And`, `Not` e `Or`. No entanto, o campo `Next` pode ser exibido somente em um Choice Rule de nível superior.
A comparação de strings com padrões com um ou mais curingas (“*”) pode ser realizada com o operador de comparação StringMatches. O escape do caractere curinga é efetuado usando o padrão `\\ (Ex: “\\*”)`. Nenhum caractere além de “*” tem qualquer significado especial durante a correspondência.

Exemplo de estado Choice

Veja a seguir um exemplo de estado `Choice` e de outros estados para os quais ele muda.

Note

Você deve especificar o campo `$.type`. Se o estado de entrada não contiver o campo `$.type`, a execução falhará e um erro será exibido no histórico de execução. Só é possível especificar uma string no campo `StringEquals` que corresponda a um valor literal. Por exemplo, `"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!"
}
```

Neste exemplo, a máquina de estado começa com o valor de entrada a seguir.

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

Transições do Step Functions para o estado `ValueInTwenties` com base no campo `value`.

Se não houver correspondências para `Choices` do estado `Choice`, o estado fornecido no campo `Default` será executado. Se o estado `Default` não for especificado, a execução apresentará uma falha com erro.

Aguardar

O estado `Wait` (`"Type": "Wait"`) impede que a máquina de estado continue durante um período especificado. Você pode escolher um tempo relativo, especificado em segundos a partir do momento em que o estado inicia, ou um tempo final absoluto, especificado como um timestamp.

Além dos campos de estado comuns, os estados `Wait` têm um dos campos a seguir.

`Seconds`

Um tempo de espera em segundos antes de iniciar o estado especificado no campo `Next`. Você deve especificar o tempo como um valor inteiro positivo de 0 a 99999999.

`Timestamp`

Um tempo de espera absoluto até o início do estado especificado no campo `Next`.

Os time stamps devem estar de acordo com o perfil RFC3339 da ISO 8601. Existem também as restrições de se ter um `T` maiúsculo para as partes de data e hora e um `Z` maiúsculo para indicar que não existe uma compensação de fuso horário numérica; por exemplo, `2024-08-18T17:33:00Z`.

> **Note**
>
> Atualmente, se você especificar o tempo de espera como um timestamp, o Step Functions considerará o valor de tempo até segundos e truncará os milissegundos.

`SecondsPath`

Um tempo, em segundos, a ser aguardado antes de iniciar o estado especificado no campo `Next`, definido por meio de um caminho dos dados de entrada do estado.

É necessário especificar um valor inteiro para esse campo.

`TimestampPath`

Um tempo de espera absoluto até o início do estado especificado no campo `Next`, definido por meio de um caminho dos dados de entrada do estado.

> **Note**
>
> Você deve especificar exatamente uma destas opções: `Seconds`, `Timestamp`, `SecondsPath` ou `TimestampPath`. Além disso, o tempo máximo de espera que você pode especificar para fluxos de trabalho padrão e expressos é de um ano e cinco minutos, respectivamente.

Exemplos de estado Wait

O estado `Wait` a seguir apresenta um atraso de 10 segundos em uma máquina de estado.

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

No exemplo a seguir, o estado `Wait` aguarda um tempo absoluto: 14 de março de 2024, à 1h59, UTC.

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

Você não precisa codificar a duração da espera. Por exemplo, tendo em vista os dados de entrada a seguir:

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

Você pode selecionar o valor de "expirydate" da entrada usando um caminho de referência para selecioná-lo nos dados de entrada.

```
"wait_until" : {
```

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

Succeed

Um estado `Succeed` (`"Type": "Succeed"`) interrompe uma execução com êxito. O estado `Succeed` é útil para ramificações do estado `Choice` que não têm outra função senão interromper a execução.

Como os estados `Succeed` são terminais, eles não têm campos `Next` e não precisam de um campo `End`, conforme mostrado no exemplo a seguir.

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

Fail

Um estado `Fail` (`"Type": "Fail"`) interrompe a execução da máquina de estado e identifica isso como falha, a menos que seja capturada por um bloqueio de `Catch`.

O estado `Fail` só permite o uso dos campos `Type` e `Comment` do conjunto de campos de estado comuns. Além disso, o estado `Fail` permite os campos a seguir.

`Cause` (opcional)

Uma sequência de caracteres personalizada que descreve a causa do erro. Você pode especificar esse campo para fins operacionais ou de diagnóstico.

`CausePath` (opcional)

Para fornecer dinamicamente uma descrição detalhada sobre a causa do erro a partir da entrada de estado usando um caminho de referência, use `CausePath`. Quando resolvido, o caminho de referência deve selecionar um campo que contenha um valor de string.

Você também pode especificar `CausePath` usando uma função intrínseca que retorna uma string. Essas funções intrínsecas são: States.Format, States.JsonToString, States.ArrayGetItem, States.Base64Encode, States.Base64Decode, States.Hash e States.UUID.

> **Important**
>
> - Você pode especificar `Cause` ou `CausePath`, mas não ambos, na definição do estado Falha.
> - Como prática recomendada de segurança da informação, recomendamos remover informações confidenciais ou detalhes do sistema interno na descrição da causa.

`Error` (opcional)

Um nome de erro que você pode fornecer para realizar o tratamento de erros usando os campos Tentar novamente ou Capturar. Você também pode fornecer um nome de erro para fins operacionais ou de diagnóstico.

`ErrorPath` (opcional)

Para fornecer dinamicamente um nome para o erro a partir da entrada de estado usando um caminho de referência, use `ErrorPath`. Quando resolvido, o caminho de referência deve selecionar um campo que contenha um valor de string.

Você também pode especificar `ErrorPath` usando uma função intrínseca que retorna uma string. Essas funções intrínsecas são: States.Format, States.JsonToString, States.ArrayGetItem, States.Base64Encode, States.Base64Decode, States.Hash e States.UUID.

> **Important**
>
> - Você pode especificar `Error` ou `ErrorPath`, mas não ambos, na definição do estado Falha.
> - Como prática recomendada de segurança da informação, recomendamos remover informações confidenciais ou detalhes do sistema interno no nome do erro.

Como os estados `Fail` sempre saem da máquina de estado, eles não têm nenhum campo `Next` nem requerem um campo `End`.

Exemplos de definição de estado Falha

O exemplo de definição de estado Falha a seguir especifica valores estáticos dos campos `Error` e `Cause`.

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

O exemplo de definição do estado Falha a seguir usa dinamicamente caminhos de referência para resolver os valores dos campos `Error` e `Cause`.

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

O exemplo de definição do estado Falha a seguir usa a função intrínseca States.Format para especificar dinamicamente os valores dos campos `Error` e `Cause`.

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

Paralelo

O estado `Parallel` (`"Type": "Parallel"`) pode ser usado para adicionar ramificações separadas de execução em sua máquina de estado.

Além dos campos de estado comuns, os estados `Parallel` incluem os campos adicionais a seguir.

`Branches` (obrigatório)

Um conjunto de objetos que especificam máquinas de estado para execução em paralelo. Cada um desses objetos de máquinas de estado deve ter campos denominados `States` e `StartAt`, cujos significados são exatamente iguais aos daqueles no nível superior de uma máquina de estado.

`ResultPath` (Opcional)

Especifica onde colocar (na entrada) a saída das ramificações. A entrada é então filtrada conforme especificado pelo campo `OutputPath` (se houver) antes de ser usada como a saída do estado. Para obter mais informações, consulte Processamento de entrada e saída.

`ResultSelector` (Opcional)

Transmitir um conjunto de pares de valores-chave, em que os valores são estáticos ou selecionados a partir do resultado. Para ter mais informações, consulte ResultSelector.

`Retry` (Opcional)

Uma matriz de objetos, chamada retriers, que define uma política de novas tentativas caso o estado encontre erros de tempo de execução. Para ter mais informações, consulte Exemplos de máquina de estado que usam Nova tentativa e Detecção.

`Catch` (Opcional)

Uma matriz de objetos, chamados Catchers, que definem um estado de fallback que é executado caso o estado encontre erros de tempo de execução e sua política de novas tentativas esteja esgotada ou não tenha sido definida. Para obter mais informações, consulte Estados de fallback.

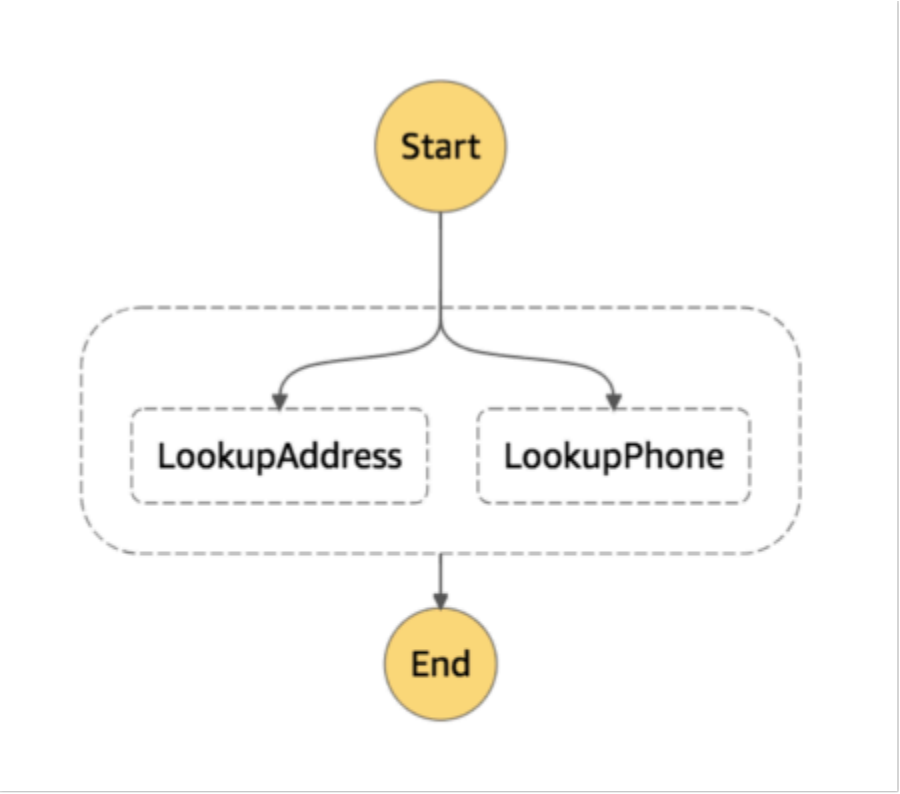
Um `Parallel` estado faz AWS Step Functions com que cada ramificação seja executada, começando com o estado nomeado no `StartAt` campo dessa ramificação, da forma mais simultânea possível, e espere até que todas as ramificações terminem (atinjam um estado terminal) antes de processar o campo do `Parallel` `Next` estado.

Exemplo de 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
          }
        }
      }
    ]
  }
}
```

Neste exemplo, as ramificações `LookupAddress` e `LookupPhone` são executadas em paralelo. Veja como fica o fluxo de trabalho visual no console do Step Functions.



Toda ramificação deve ser independente. Um estado em uma ramificação de estado `Parallel` não deve ter um campo `Next` direcionado a um campo fora da ramificação e qualquer outro estado fora da ramificação não pode mudar para essa ramificação.

Processamento de entrada e saída do estado Parallel

O estado `Parallel` fornece a cada ramificação uma cópia de seus próprios dados de entrada (que podem ser modificados pelo campo `InputPath`). Ele gera uma saída que é uma matriz com um elemento para cada ramificação, contendo a saída dessa ramificação. Não há necessidade de que todos os elementos sejam do mesmo tipo. A matriz de saída pode ser inserida nos dados de entrada (e o todo enviado como a saída do estado `Parallel`) usando um campo `ResultPath` normalmente (consulte Processamento de entrada e saída).

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

Se for atribuído o estado `FunWithMath` à matriz `[3, 2]` como entrada, os estados `Add` e `Subtract` receberão essa matriz como entrada. A saída das tarefas `Add` e `Subtract` seria a soma e a diferença entre os elementos 3 e 2 da matriz, que é 5 e 1, enquanto a saída do estado `Parallel` seria uma matriz.

```
[ 5, 1 ]
```

> **Tip**
>
> Se o estado Paralelo ou Mapa usado nas máquinas de estado retornar uma matriz de matrizes, você poderá transformá-las em uma matriz nivelada com o campo ResultSelector. Para ter mais informações, consulte Nivelamento de uma matriz de matrizes.

Como tratar erros

Quando alguma ramificação falha, devido a um erro não tratado ou com a transição para um estado `Fail`, considera-se que todo o estado `Parallel` falhou e todas as suas ramificações estão interrompidas. Se o erro não for tratado pelo estado `Parallel` em si, o Step Functions interromperá a execução com um erro.

> **Note**
>
> Quando um estado paralelo falha, as funções do Lambda invocadas continuam a ser executadas e os operadores de atividades que processam um token da tarefa não são interrompidos.
>
> - Para interromper atividades de longa execução, use pulsações para detectar se a respectiva ramificação foi interrompida pelo Step Functions e interrompa os operadores que estiverem processando tarefas. Chamar `SendTaskHeartbeat`, `SendTaskSuccess` ou `SendTaskFailure` lançará um erro se o estado falhou. Consulte Erros de pulsação.
> - Funções do Lambda em execução não podem ser interrompidas. Se você tiver implementado um fallback, use um estado `Wait` para que o trabalho de limpeza aconteça depois que a função do Lambda for concluída.

Mapa

Use o estado `Map` para executar um conjunto de etapas do fluxo de trabalho para cada item em um conjunto de dados. As iterações do estado `Map` são executadas em paralelo, o que possibilita o processamento rápido de um conjunto de dados. Os estados `Map` podem usar uma variedade de tipos de entrada, incluindo uma matriz JSON, uma lista de objetos do Amazon S3 ou um arquivo CSV.

O Step Functions fornece dois tipos de modos de processamento para usar o estado `Map` em seus fluxos de trabalho: modo Inline e modo Distribuído.

Para obter informações sobre esses modos e como usar o estado `Map` em qualquer um dos modos, consulte os seguintes tópicos:

- Modos de processamento do estado do mapa
  - Usar o estado Mapa no modo inline.
  - Usar o estado do mapa no modo distribuído

> **Tip**
>
> Para implantar um exemplo de fluxo de trabalho que usa um estado `Map` em sua Conta da AWS, consulte Módulo 5 - Estado Escolha e estado Mapa do workshop do AWS Step Functions.

Modos de processamento do estado do mapa

O Step Functions fornece os seguintes modos de processamento para o estado `Map`, dependendo de como você deseja processar os itens em um conjunto de dados.

- Em linha — modo de concorrência limitada. Nesse modo, cada iteração do estado `Map` é executada no contexto do fluxo de trabalho que contém o estado `Map`. O Step Functions adiciona o histórico de execução dessas iterações ao histórico de execuções do fluxo de trabalho principal. Por padrão, os estados `Map` são executados no modo Em linha.

  Nesse modo, o estado `Map` aceita somente uma matriz JSON como entrada. Além disso, esse modo oferece suporte para até 40 iterações simultâneas.

  Para obter mais informações, consulte Usar o estado Mapa no modo inline..
- Distribuído — modo de alta simultaneidade. Nesse modo, o estado `Map` executa cada iteração como uma execução de fluxo de trabalho secundário, o que permite um processamento simultâneo de até 10.000 execuções paralelas de fluxo de trabalho secundário. Cada execução de fluxo de trabalho secundário tem seu próprio histórico de execução separado do fluxo de trabalho principal.

Nesse modo, o estado `Map` pode aceitar uma matriz JSON ou uma fonte de dados do Amazon S3, como um arquivo CSV, como entrada.

Para obter mais informações, consulte Usar o estado do mapa no modo distribuído.

O modo a ser utilizado depende de como você deseja processar os itens em um conjunto de dados. Use o estado `Map` no modo Em linha se o histórico de execução do fluxo de trabalho não exceder 25.000 entradas ou se você não precisar de mais de 40 iterações simultâneas.

Use o estado `Map` no modo distribuído quando precisar orquestrar workloads paralelas em grande escala que atendam a qualquer combinação das seguintes condições:

- O tamanho do conjunto de dados excede 256 KB.
- O histórico de eventos de execução do fluxo de trabalho excede 25.000 entradas.
- Você precisa processar simultaneamente mais de 40 iterações paralelas.

Tópicos

Diferenças entre o modo Em linha e Distribuído

A tabela a seguir destaca as diferenças entre os modos Em linha e Distribuído.

Modo inline	Modo distribuído
Supported data sources	
Aceita uma matriz JSON transmitida de uma etapa anterior no fluxo de trabalho como entrada.	Aceita as seguintes fontes de dados como entrada: • Matriz JSON transmitida de uma etapa anterior no fluxo de trabalho • Arquivo JSON em um bucket do Amazon S3 contendo uma matriz

Modo inline	Modo distribuído
	• Arquivo CSV em um bucket do Amazon S3 • Lista de objetos do Amazon S3 • Inventário do Amazon S3
Map iterations	
Nesse modo, cada iteração do estado `Map` é executada no contexto do fluxo de trabalho que contém o estado `Map`. O Step Functions adiciona o histórico de execução dessas iterações ao histórico de execuções do fluxo de trabalho principal.	Nesse modo, o estado `Map` executa cada iteração como uma execução de fluxo de trabalho secundário, o que permite um processamento simultâneo de até 10.000 execuções paralelas de fluxo de trabalho secundário. Cada execução de fluxo de trabalho secundário tem seu próprio histórico de execução separado do fluxo de trabalho principal.
Maximum concurrency for parallel iterations	
Permite que você execute até 40 iterações da maneira mais simultânea possível.	Permite executar até 10.000 execuções paralelas de fluxo de trabalho secundário para processar milhões de itens de dados ao mesmo tempo.
Input payload and event history sizes	

Modo inline	Modo distribuído
Impõe um limite de 256 KB no tamanho do payload da entrada e 25.000 entradas no histórico de eventos de execução.	Permite superar a limitação do tamanho de payload porque o estado `Map` pode ler a entrada diretamente das fontes de dados do Amazon S3. Nesse modo, também é possível superar as limitações do histórico de execução, porque as execuções do fluxo de trabalho secundári o iniciadas pelo estado `Map` mantêm seus próprios históricos de execução separados do histórico de execução do fluxo de trabalho principal.
Monitoring and observability	
Você pode revisar o histórico de execução de fluxo de trabalho no console ou invocando a ação da API `GetExecutionHistory`. Você também pode visualizar o histórico de execução por meio do CloudWatch e do X-Ray.	Ao executar um estado `Map` no modo distribuí do, o Step Functions cria um recurso de Execução de mapa. Uma Execução de mapa se refere a um conjunto de execuções secundárias de fluxo de trabalho que um estado Mapa Distribuído inicia. Você pode ver uma Execução de mapa no console do Step Functions. Você também pode invocar a ação da API `DescribeMapRun`. Uma Execução de mapa também emite métricas para o CloudWatch. Para obter mais informações, consulte Examinando a execução do mapa de uma execução de estado de mapa distribuído.

Usar o estado Mapa no modo inline.

Por padrão, os estados `Map` são executados no modo inline. No modo inline, o estado Mapa aceita somente uma matriz JSON como entrada. Ele recebe essa matriz de uma etapa anterior no fluxo de

trabalho. Nesse modo, cada iteração do estado `Map` é executada no contexto do fluxo de trabalho que contém o estado `Map`. O Step Functions adiciona o histórico de execução dessas iterações ao histórico de execuções do fluxo de trabalho principal.

Nesse modo, o estado `Map` oferece suporte para até 40 iterações simultâneas.

Um estado `Map` definido como Inline é conhecido como estado Mapa inline. Use o estado `Map` no modo Em linha se o histórico de execução do fluxo de trabalho não exceder 25.000 entradas ou se você não precisar de mais de 40 iterações simultâneas.

Para obter uma introdução ao uso do estado Mapa inline, consulte o tutorial Repetir uma ação usando o estado Mapa em linha.

Índice

Principais conceitos neste tópico

Modo inline

Um modo de processamento simultâneo limitado do estado `Map`. Nesse modo, cada iteração do estado `Map` é executada no contexto do fluxo de trabalho que contém o estado `Map`. O Step Functions adiciona o histórico de execução dessas iterações ao histórico de execuções do fluxo de trabalho principal. Os estados `Map` são executados no modo inline como padrão.

Esse modo aceita somente uma matriz JSON como entrada e é compatível com até 40 iterações simultâneas.

Estado Mapa inline

Um estado `Map` definido como modo inline.

Fluxo de trabalho do mapa

O conjunto de etapas que o estado `Map` executa para cada iteração.

Iteração do estado Mapa

Uma repetição do fluxo de trabalho definida dentro do estado `Map`.

Campos do estado Mapa inline

Para usar o estado Mapa inline em fluxos de trabalho, especifique um ou mais dos campos a seguir. Você especifica esses campos além dos campos de estado comuns.

`Type` (obrigatório)

Define o tipo de estado, como `Map`.

`ItemProcessor` (obrigatório)

Contém os seguintes objetos JSON que especificam a definição e o modo de processamento estado `Map`.

A definição contém o conjunto de etapas a serem repetidas para processar cada item da matriz.

- `ProcessorConfig`— Um objeto JSON opcional que especifica o modo de processamento do estado `Map`. Esse objeto contém o subcampo `Mode`. Esse campo é padronizado como `INLINE`, que usa o estado `Map` no modo inline.

  Nesse modo, a falha de qualquer iteração causa uma falha no estado `Map`. Todas as iterações param quando uma falha ocorre no estado `Map`.

- `StartAt`— Especifica uma string que indica o primeiro estado em um fluxo de trabalho. Essa string diferencia maiúsculas de minúsculas e deve corresponder ao nome de um dos objetos de estado. Esse estado é executado primeiro para cada item no conjunto de dados. Qualquer entrada de execução fornecida ao estado `Map` é transmitida primeiro para o estado `StartAt`.
- `States` – Um objeto JSON que contém um conjunto de estados delimitado por vírgulas. Nesse objeto, você define o Map workflow.

  > **Note**
  >
  > - Os estados no campo `ItemProcessor` só podem fazer a transição entre si. Nenhum estado fora do campo `ItemProcessor` pode fazer a transição para um estado dentro dele.

- O campo `ItemProcessor` substitui o campo `Iterator`, agora obsoleto. Embora você possa continuar a incluir estados de `Map` que usam o campo `Iterator`, é altamente recomendável substituir esse campo por `ItemProcessor`.

  No momento, o Step Functions Local não é compatível com o campo `ItemProcessor`. Recomendamos o uso do campo `Iterator` com o Step Functions Local.

`ItemsPath` (opcional)

Especifica um caminho de referência usando a sintaxe JsonPath. Esse caminho seleciona o nó JSON que contém a matriz de itens dentro da entrada de estado. Para obter mais informações, consulte ItemsPath.

`ItemSelector` (opcional)

Substitui os valores da matriz de entrada antes de serem transmitidos para cada iteração do estado `Map`.

Nesse campo, você especifica um JSON válido que contém um conjunto de pares de valores-chave. Esses pares podem conter espaços ou um destes itens:

- Valores estáticos selecionados na definição da máquina de estado.
- Valores selecionados da entrada de estado usando um caminho.
- Valores acessados a partir do objeto de contexto.

Para obter mais informações, consulte ItemSelector.

O campo `ItemSelector` substitui o campo `Parameters`, agora obsoleto. Embora você possa continuar a incluir estados de `Map` que usam o campo `Parameters`, é altamente recomendável substituir esse campo por `ItemSelector`.

`MaxConcurrency` (opcional)

Especifica um valor inteiro que fornece o limite superior do número de iterações do estado `Map` que podem ser executadas em paralelo. Por exemplo, um valor `MaxConcurrency` de 10 limitará o estado `Map` a 10 iterações simultâneas em execução ao mesmo tempo.

> **Note**
>
> As iterações simultâneas podem ser limitadas. Quando isso ocorre, algumas iterações não começarão até que as iterações anteriores sejam concluídas. A probabilidade de isso ocorrer aumenta quando sua matriz de entrada tem mais de 40 itens.
> Para aumentar a capacidade de processamento simultâneo, considere Usar o estado do mapa no modo distribuído.

O valor padrão é `0`, o que não limita a capacidade de processamento simultâneo. O Step Functions invoca iterações da forma mais simultânea possível.

Um valor `MaxConcurrency` de `1` invoca o `ItemProcessor` uma vez para cada elemento da matriz. Os itens na matriz são processados na ordem em que aparecem na entrada. O Step Functions não iniciará uma nova iteração até concluir a iteração anterior.

`MaxConcurrencyPath` (opcional)

Para fornecer dinamicamente um valor máximo de processamento simultâneo a partir da entrada de estado usando um caminho de referência, use `MaxConcurrencyPath`. Quando resolvido, o caminho de referência deve selecionar um campo cujo valor seja um número inteiro não negativo.

> **Note**
>
> Um estado `Map` não pode incluir `MaxConcurrency` e `MaxConcurrencyPath`.

`ResultPath` (opcional)

Especifica onde na entrada armazenar a saída das iterações do estado `Map`. Em seguida, o estado Mapa filtra a entrada conforme especificado pelo campo `OutputPath`, se especificado. Em seguida, ele usa a entrada filtrada como saída do estado. Para obter mais informações, consulte Processamento de entrada e saída.

`ResultSelector` (opcional)

Transmitir um conjunto de pares de valores-chave, em que os valores são estáticos ou selecionados a partir do resultado. Para obter mais informações, consulte ResultSelector.

> **Tip**
>
> Se o estado Paralelo ou Mapa usado nas máquinas de estado retornar uma matriz de matrizes, você poderá transformá-las em uma matriz nivelada com o campo ResultSelector. Para obter mais informações, consulte Nivelamento de uma matriz de matrizes.

`Retry` (opcional)

Uma matriz de objetos, chamados Retriers, que definem uma política de novas tentativas. Os estados usam uma política de nova tentativa quando encontram erros de runtime. Para obter mais informações, consulte Exemplos de máquina de estado que usam Nova tentativa e Detecção.

> **Note**
>
> Se você definir Retriers para o estado Mapa inline, a política de nova tentativa se aplicará a todas as iterações do estado `Map`, em vez de somente às iterações com falha. Por exemplo, o estado `Map` contém duas iterações concluídas com êxito e uma com falha. Se você tiver definido o campo `Retry` para o estado `Map`, a política de nova tentativa se aplicará às três iterações do estado `Map`, em vez de apenas à iteração com falha.

`Catch` (opcional)

Uma matriz de objetos, chamados Catchers, que definem um estado de fallback. Os estados executarão um catcher se encontrarem erros de runtime e não tiverem uma política de nova tentativa ou se a política de nova tentativa estiver esgotada. Para obter mais informações, consulte Estados de fallback.

Campos descontinuados

> **Note**
>
> Embora você possa continuar a incluir estados de `Map` que usam o campo `Iterator`, é altamente recomendável substituir por `ItemProcessor` e `Parameters` por `ItemSelector`.

`Iterator`

Especifica um objeto JSON que define um conjunto de etapas que processam cada elemento da matriz.

`Parameters`

Especifica um conjunto de pares de valores-chave, no qual os valores podem conter espaços ou um destes itens:

- Valores estáticos selecionados na definição da máquina de estado.
- Valores selecionados da entrada usando um caminho.

Exemplo de estado Mapa inline

Considere os seguintes dados de entrada para um estado `Map` no modo inline.

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

Considerando a entrada anterior, o estado `Map` no exemplo a seguir invocará uma função do AWS Lambda chamada `ship-val` uma vez para cada item da matriz no 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 iteração do estado `Map` enviará um item na matriz, selecionada com o campo `ItemsPath` como entrada para a função do Lambda `ship-val`. Os valores a seguir são um exemplo de entrada que o estado `Map` envia para uma invocação da função do Lambda:

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

Quando concluída, a saída do estado `Map` é uma matriz JSON, onde cada item é a saída de uma iteração. Nesse caso, essa matriz contém a saída da função do Lambda `ship-val`.

Exemplo de estado Mapa inline com **`ItemSelector`**.

Vamos supor que a função do Lambda `ship-val` no exemplo anterior também precise de informações sobre a transportadora da remessa. Essas informações são adicionadas aos itens na matriz para cada iteração. Você pode incluir informações da entrada, juntamente com informações específicas à iteração atual do estado `Map`. Observe o campo `ItemSelector` no exemplo a seguir:

```
"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
}
```

O bloco `ItemSelector` substitui a entrada das iterações por um nó JSON. Esse nó contém os dados do item atual do objeto de contexto e as informações da transportadora do campo `delivery-partner` da entrada do estado `Map`. O exemplo a seguir é um entrada para uma única iteração. O estado `Map` transmite essa entrada para uma invocação da função do Lambda `ship-val`.

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

```
}
```

No exemplo anterior do estado Mapa inline, o campo `ResultPath` produz uma saída no mesmo formato da entrada. No entanto, ele substitui o campo `detail.shipped` por uma matriz na qual cada elemento é a saída da invocação do Lambda `ship-val` de cada iteração.

Para ver mais informações sobre como usar o estado Mapa inline e seus campos, consulte o seguinte.

- Repetir uma ação usando o estado Mapa em linha
- Processamento de entrada e saída no Step Functions
- ItemsPath
- Dados de objeto de contexto para estados Map

Processamento de entrada e saída do estado **`Map`** inline.

Para um determinado estado `Map`, o `InputPath` seleciona um subconjunto da entrada do estado.

A entrada de um estado `Map` deve incluir uma matriz JSON. O estado `Map` executa a seção `ItemProcessor` uma vez para cada item na matriz. Se você especificar o campo `ItemsPath`, o estado `Map` selecionará onde na entrada encontrar a matriz sobre a qual iterar. Se não for especificado, o valor de `ItemsPath` será `$` e a seção `ItemProcessor` esperará que a matriz seja a única entrada. Se você especificar o campo `ItemsPath`, seu valor deverá ser um caminho de referência. O estado `Map` aplica esse caminho à entrada efetiva depois de aplicar o `InputPath`. O `ItemsPath` deve identificar um campo cujo valor seja uma matriz JSON.

A entrada para cada iteração, por padrão, é um único elemento do campo de matriz identificado pelo valor `ItemsPath`. Você pode substituir esse valor pelo campo `ItemSelector`.

Quando concluída, a saída do estado `Map` é uma matriz JSON, onde cada item é a saída de uma iteração.

Para ver mais informações sobre entradas e saídas do estado Mapa inline, consulte:

- Repetir uma ação usando o estado Mapa em linha
- Exemplo de estado Mapa inline com `ItemSelector`.
- Processamento de entrada e saída no Step Functions

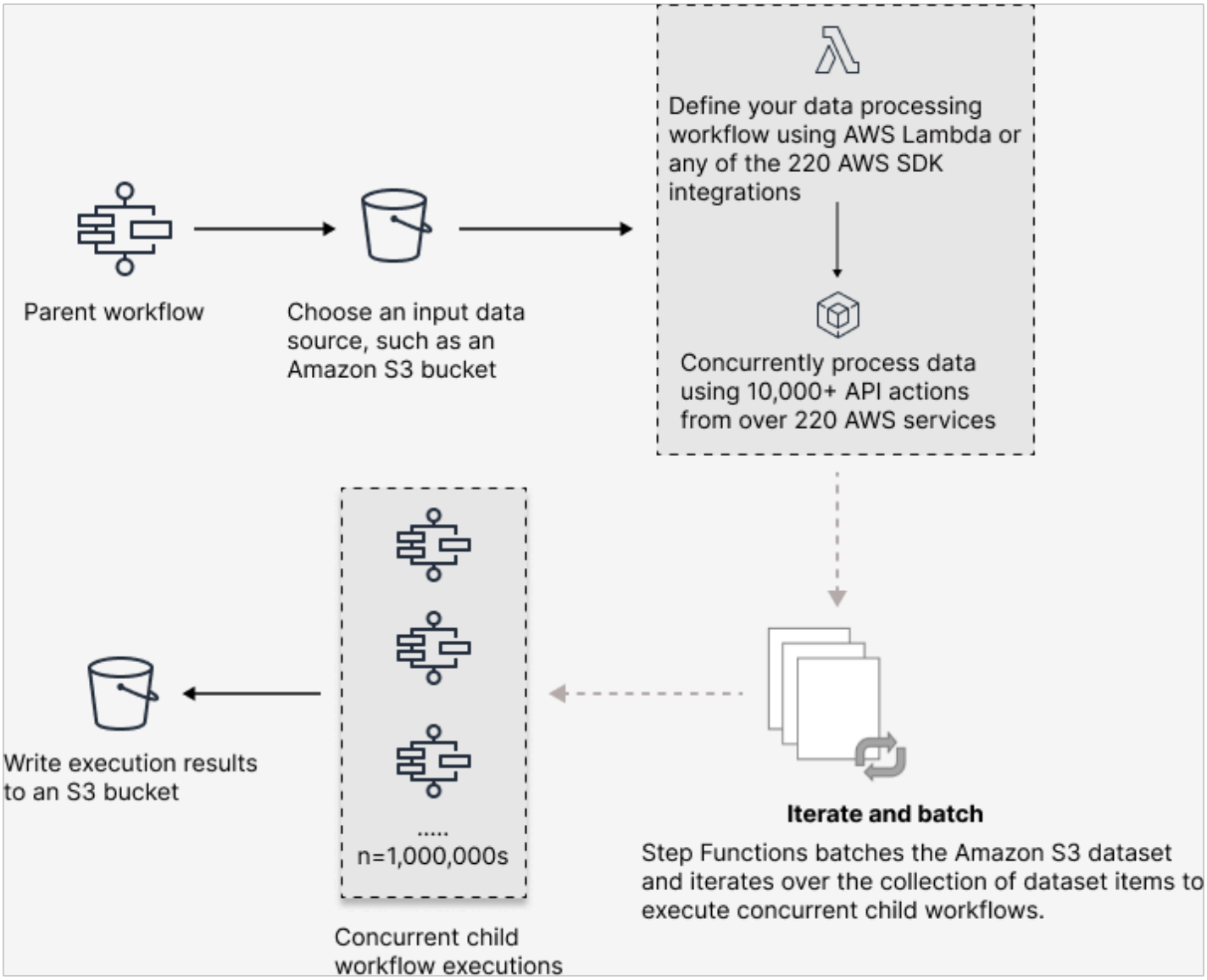
- Dados de objeto de contexto para estados Map
- Processar dados dinamicamente com um estado Mapa

Usar o estado Mapa no modo distribuído para orquestrar workloads paralelas em grande escala

Com o Step Functions, você pode orquestrar workloads paralelas em grande escala para realizar tarefas, como processamento sob demanda de dados semiestruturados. Essas workloads paralelas permitem que você processe simultaneamente fontes de dados em grande escala armazenadas no Amazon S3. Por exemplo, você pode processar um único arquivo JSON ou CSV que contém grandes quantidades de dados. Ou você pode processar um grande conjunto de objetos do Amazon S3.

Para configurar uma workload paralela em grande escala nos fluxos de trabalho, inclua um estado `Map` no modo distribuído. O estado Mapa processa itens em um conjunto de dados simultaneamente. Um estado `Map` definido como Distribuído é conhecido como estado Mapa distribuído. No modo distribuído, o estado `Map` permite o processamento simultâneo em grande escala. No modo distribuído, o estado `Map` processa os itens no conjunto de dados em iterações chamadas execuções de fluxo de trabalho secundário. É possível especificar o número de execuções de fluxo de trabalho secundário que podem ser executadas em paralelo. Cada execução de fluxo de trabalho secundário tem seu próprio histórico de execução separado do fluxo de trabalho principal. Se você não especificar, o Step Functions executará 10 mil execuções paralelas de fluxo de trabalho secundário.

A ilustração a seguir explica como você pode configurar workloads paralelas em grande escala em seus fluxos de trabalho.



Neste tópico

Principais termos

Modo distribuído

Um modo de processamento do estado Mapa. Nesse modo, cada iteração do estado `Map` é executada como uma execução de fluxo de trabalho secundário que permite processamento simultâneo em grande escala. Cada execução de fluxo de trabalho secundário tem seu próprio histórico de execução, que é separado do histórico de execução do fluxo de trabalho principal. Esse modo é compatível com a leitura de entradas de fontes de dados do Amazon S3 em grande escala.

Estado Mapa distribuído

Um estado Mapa definido para o modo de processamento Distribuído.

Fluxo de trabalho do mapa

Um conjunto de etapas que um estado `Map` executa.

Fluxo de trabalho principal

Um fluxo de trabalho que contém um ou mais estados Mapa distribuído.

Execuções de fluxo de trabalho secundário

Uma iteração do estado Mapa Distribuído. Uma execução de fluxo de trabalho secundário tem seu próprio histórico de execução, que é separado do histórico de execução do fluxo de trabalho principal.

Execução de mapa

Ao executar um estado `Map` no modo distribuído, o Step Functions cria um recurso de Execução de mapa. Uma Execução de mapa se refere a um conjunto de execuções de fluxo de trabalho secundário que um estado Mapa distribuído inicia e às configurações de runtime que controlam essas execuções. O Step Functions atribui um nome do recurso da Amazon (ARN) à Execução de mapa. Você pode examinar uma Execução de mapa no console do Step Functions. Você também pode invocar a ação da API `DescribeMapRun`. A Map Run também emite métricas para CloudWatch.

Para ter mais informações, consulte Examinando o Map Run.

Exemplo de definição do estado Mapa Distribuído

Use o estado `Map` no modo distribuído quando precisar orquestrar workloads paralelas em grande escala que atendam a qualquer combinação das seguintes condições:

- O tamanho do conjunto de dados excede 256 KB.
- O histórico de eventos de execução do fluxo de trabalho excede 25 mil entradas.
- Você precisa processar simultaneamente mais de 40 iterações paralelas.

O exemplo de definição de estado Mapa Distribuído a seguir especifica o conjunto de dados como um arquivo CSV armazenado em um bucket do Amazon S3. Ele também especifica uma função do Lambda que processa os dados em cada linha do arquivo CSV. Uma vez que esse exemplo usa um arquivo CSV, ele também especifica a localização dos cabeçalhos das colunas do CSV. Para ver a definição completa da máquina de estado desse exemplo, consulte o tutorial Copiar dados de CSV em grande escala usando o Mapa distribuído.

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

Permissões para executar o Mapa distribuído

Ao incluir um estado Mapa Distribuído nos fluxos de trabalho, o Step Functions precisa de permissões apropriadas para permitir que o perfil de máquina de estado invoque a ação da API `StartExecution` para o estado Mapa Distribuído.

O exemplo de política do IAM a seguir concede os privilégios mínimos necessários ao perfil da máquina de estado para executar o estado Mapa Distribuído.

> **Note**
>
> Substitua *stateMachineName* pelo nome da máquina de estado na qual você está usando o estado Mapa Distribuído. Por exemplo, `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:*"
    }
  ]
}
```

Além disso, você precisa garantir que tenha os privilégios mínimos necessários para acessar os AWS recursos usados no estado do Mapa Distribuído, como os buckets do Amazon S3. Para mais informações, consulte Políticas do IAM para usar o estado Mapa Distribuído.

Campos do estado Mapa Distribuído

Para usar o estado Mapa distribuído em fluxos de trabalho, especifique um ou mais dos campos a seguir. Você especifica esses campos além dos campos de estado comuns.

`Type` (obrigatório)

Define o tipo de estado, como `Map`.

`ItemProcessor` (obrigatório)

Contém os seguintes objetos JSON que especificam a definição e o modo de processamento estado `Map`.

- `ProcessorConfig`— Um objeto JSON que especifica a configuração do estado `Map`. O objeto contém os subcampos a seguir.
  - `Mode`— Definido como **`DISTRIBUTED`** para usar o estado `Map` no modo distribuído.

> **Note**
>
> Atualmente, se você usa o estado `Map` nos fluxos de trabalho expressos, não é possível definir o `Mode` como `DISTRIBUTED`. No entanto, se você usa o estado `Map` nos fluxos de trabalho padrão, é possível definir o `Mode` como `DISTRIBUTED`.

  - `ExecutionType`— Especifica o tipo de execução do fluxo de trabalho do mapa como PADRÃO ou EXPRESSO. Você deve fornecer esse campo se tiver especificado `DISTRIBUTED` para o subcampo `Mode`. Para ver mais informações sobre tipos de fluxos de trabalho, consulte Comparação entre os fluxos de trabalho padrão e expresso.
- `StartAt`— Especifica uma string que indica o primeiro estado em um fluxo de trabalho. Essa string diferencia maiúsculas de minúsculas e deve corresponder ao nome de um dos objetos de estado. Esse estado é executado primeiro para cada item no conjunto de dados. Qualquer entrada de execução fornecida ao estado `Map` é transmitida primeiro para o estado `StartAt`.
- `States` – Um objeto JSON que contém um conjunto de estados delimitado por vírgulas. Nesse objeto, você define o Map workflow.

`ItemReader`

Especifica um conjunto de dados e sua localização. O estado `Map` recebe seus dados de entrada do conjunto de dados especificado.

No modo distribuído, você pode usar uma carga JSON transferida de um estado anterior ou uma fonte de dados do Amazon S3 em grande escala como conjunto de dados. Para ter mais informações, consulte ItemReader.

`ItemsPath` (Opcional)

Especifica um caminho de referência usando a JsonPathsintaxe para selecionar o nó JSON que contém uma matriz de itens dentro da entrada de estado.

No modo distribuído, você especifica esse campo somente ao usar uma matriz JSON de uma etapa anterior como entrada de estado. Para ter mais informações, consulte ItemsPath.

`ItemSelector` (Opcional)

Substitui os valores de itens individuais do conjunto de dados antes de serem transmitidos para cada iteração do estado `Map`.

Nesse campo, você especifica uma entrada JSON válida que contém um conjunto de pares de valores-chave. Esses pares podem ser valores estáticos configurados na definição da máquina

de estado, valores selecionados da entrada de estado usando um caminho ou valores acessados a partir do objeto de contexto. Para ter mais informações, consulte ItemSelector.

`ItemBatcher` (Opcional)

Especifica o processamento dos itens do conjunto de dados em lotes. Em seguida, cada execução de fluxo de trabalho secundário recebe um lote desses itens como entrada. Para ter mais informações, consulte ItemBatcher.

`MaxConcurrency` (Opcional)

Especifica o número de execuções de fluxo de trabalho secundário que podem ser executadas em paralelo. O intérprete só permite até o número especificado de execuções paralelas de fluxo de trabalho secundário. Se você não especificar um valor de processamento simultâneo ou defini-lo como zero, o Step Functions não limitará o processamento simultâneo e executará 10 mil execuções paralelas de fluxo de trabalho secundário.

> **Note**
>
> Embora você possa especificar um limite maior de simultaneidade para execuções paralelas de fluxos de trabalho secundários, recomendamos que você não exceda a capacidade de um AWS serviço downstream, como. AWS Lambda

`MaxConcurrencyPath` (Opcional)

Para fornecer dinamicamente um valor máximo de processamento simultâneo a partir da entrada de estado usando um caminho de referência, use `MaxConcurrencyPath`. Quando resolvido, o caminho de referência deve selecionar um campo cujo valor seja um número inteiro não negativo.

> **Note**
>
> Um estado `Map` não pode incluir `MaxConcurrency` e `MaxConcurrencyPath`.

`ToleratedFailurePercentage` (Opcional)

Define a porcentagem de itens com falha a serem tolerados em uma Execução de mapa. A Execução de mapa falhará automaticamente se exceder essa porcentagem. O Step Functions calcula a porcentagem de itens com falha como resultado do número total de itens com falha ou com tempo limite esgotado dividido pelo número total de itens. Você deve especificar um valor

entre zero e cem. Para ter mais informações, consulte Limite de falha tolerado para o estado Mapa Distribuído.

`ToleratedFailurePercentagePath` (Opcional)

Para fornecer dinamicamente um valor de porcentagem de falha tolerada com base na entrada de estado utilizando um caminho de referência, use `ToleratedFailurePercentagePath`. Quando resolvido, o caminho de referência deve selecionar um campo cujo valor seja um número entre zero e cem.

`ToleratedFailureCount` (Opcional)

Define o número de itens com falha a serem tolerados em uma Execução de mapa. A Execução de mapa falhará automaticamente se exceder esse número. Para ter mais informações, consulte Limite de falha tolerado para o estado Mapa Distribuído.

`ToleratedFailureCountPath` (Opcional)

Para fornecer dinamicamente um valor de contagem de falhas toleradas com base na entrada de estado utilizando um caminho de referência, use `ToleratedFailureCountPath`. Quando resolvido, o caminho de referência deve selecionar um campo cujo valor seja um número inteiro não negativo.

`Label` (Opcional)

Uma string que identifica exclusivamente um estado `Map`. Para cada Execução de mapa, o Step Functions adiciona o rótulo ao ARN da Execução de mapa. Veja a seguir um exemplo de um ARN de Execução de mapa com um rótulo personalizado chamado `demoLabel`:

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

Se você não especificar um rótulo, o Step Functions gerará automaticamente um rótulo exclusivo.

> **Note**
>
> Os rótulos não podem ter mais de 40 caracteres, devem ser exclusivos em uma definição de máquina de estado e não podem conter nenhum dos caracteres a seguir.
>
> - Caracteres de espaço em branco
> - Caracteres curinga (`? *`)
> - Caracteres de colchete (`< > { } [ ]`)
> - Caracteres especiais (`: ; , \ | ^ ~ $ # % & ` "`)

- caracteres de controle (`\\u0000` - `\\u001f` ou `\\u007f` - `\\u009f`).

O Step Functions permite criar nomes de máquina de estado, execuções, atividades e labels que contenham caracteres não ASCII. Esses nomes não ASCII não funcionam com a Amazon. CloudWatch Para garantir que você possa acompanhar CloudWatch as métricas, escolha um nome que use somente caracteres ASCII.

`ResultWriter` (Opcional)

Especifica o local do Amazon S3 em que o Step Functions grava todos os resultados da execução do fluxo de trabalho secundário.

O Step Functions consolida todos os dados da execução de um fluxo de trabalho secundário, como a entrada e saída de execução, ARN e status da execução. Em seguida, ele exporta as execuções com o mesmo status para seus respectivos arquivos na localização especificada do Amazon S3. Para ter mais informações, consulte ResultWriter.

Se você não exportar os resultados do estado `Map`, ele retornará uma matriz de todos os resultados da execução do fluxo de trabalho secundário. Por exemplo: .

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

`ResultPath` (Opcional)

Especifica onde colocar a saída das iterações na entrada. A entrada é então filtrada conforme especificado pelo campo `OutputPath` (se presente), antes de ser transmitida como a saída do estado. Para obter mais informações, consulte Processamento de entrada e saída.

`ResultSelector` (Opcional)

Transmitir um conjunto de pares de valores-chave, em que os valores são estáticos ou selecionados a partir do resultado. Para ter mais informações, consulte ResultSelector.

> **Tip**
>
> Se o estado Paralelo ou Mapa usado nas máquinas de estado retornar uma matriz de matrizes, você poderá transformá-las em uma matriz nivelada com o campo ResultSelector. Para ter mais informações, consulte Nivelamento de uma matriz de matrizes.

`Retry` (Opcional)

Uma matriz de objetos, chamados Retriers, que definem uma política de novas tentativas. Uma execução usará a política de novas tentativas caso o estado encontre erros de runtime. Para ter mais informações, consulte Exemplos de máquina de estado que usam Nova tentativa e Detecção.

> **Note**
>
> Se você definir Retriers para o estado Mapa Distribuído, a política de novas tentativas se aplicará a todas as execuções do fluxo de trabalho secundário iniciadas pelo estado `Map`. Por exemplo, imagine que o estado `Map` iniciou três execuções de fluxo de trabalho secundário, das quais uma falha. Quando a falha ocorre, a execução usa o campo `Retry`, se definido, para o estado `Map`. A política de repetição se aplica a todas as execuções do fluxo de trabalho secundário e não apenas à execução com falha. Se uma ou mais execuções de fluxo de trabalho secundário falharem, o mesmo ocorrerá com a Execução de mapa.
> Ao repetir um estado `Map`, uma nova Execução de mapa é criada.

`Catch` (Opcional)

Uma matriz de objetos, chamados Catchers, que definem um estado de fallback. O Step Functions usará os Catchers definidos em `Catch` se o estado encontrar erros de runtime. Quando ocorre um erro, a execução usa primeiro todos os retriers definidos em `Retry`. Se a política de novas tentativas não estiver definida ou estiver esgotada, a execução usará seus Catchers, se definidos. Para obter mais informações, consulte Estados de fallback.

Próximas etapas

Para continuar aprendendo mais sobre o estado Mapa Distribuído, consulte os seguintes recursos:

- Processamento de entrada e saída

  Para configurar a entrada que um estado Mapa Distribuído recebe e a saída que ele gera, o Step Functions fornece os seguintes campos:

  - ItemReader
  - ItemsPath

- ItemSelector
- ItemBatcher
- ResultWriter
- Como analisar um arquivo CSV de entrada

Além desses campos, o Step Functions também fornece a capacidade de definir um limite de falha tolerado para Mapa distribuído. Esse valor permite especificar o número máximo ou a porcentagem de itens com falha como um limite de falha para uma Execução de mapa. Para ver mais informações sobre como configurar o limite de falhas toleradas, consulte Limite de falha tolerado para o estado Mapa Distribuído

- Usar o estado Mapa Distribuído

  Consulte os seguintes tutoriais e exemplos de projetos para começar a usar o estado Mapa Distribuído.

  - Conceitos básicos do estado de Mapa Distribuído
  - Como processar um lote inteiro de dados em uma função do Lambda
  - Como processar de itens de dados individuais com uma função do Lambda
  - Exemplo de projeto: processar um arquivo CSV com Mapa distribuído
  - Exemplo de projeto: processar dados em um bucket do Amazon S3 com Mapa distribuído
- Examine a execução do estado Mapa Distribuído

  O console do Step Functions tem uma página Detalhes da Execução de mapa que exibe todas as informações relacionadas à execução de um estado Mapa distribuído. Para ver informações sobre como examinar as informações exibidas nesta página, consulte Examinando o Map Run.

Limite de falha tolerado para o estado Mapa Distribuído

Ao orquestrar cargas de trabalho paralelas em grande escala, você também pode definir um limite de falha tolerado. Esse valor permite especificar o número máximo ou a porcentagem de itens com falha como um limite de falha para uma Execução de mapa. Dependendo do valor que você especificar, a Execução de mapa falhará automaticamente se exceder o limite. Se você especificar os dois valores, o fluxo de trabalho falhará quando exceder qualquer um dos valores.

Especificar um limite ajuda você a falhar em um número específico de itens antes que toda a execução do mapa falhe. O Step Functions retorna um erro de

`States.ExceedToleratedFailureThreshold` quando a Execução de mapa falha porque o limite especificado foi excedido.

> **Note**
>
> O Step Functions pode continuar executando fluxos de trabalho secundários em uma Execução de mapa mesmo depois que o limite de falha tolerado for excedido, mas antes que a Execução de mapa falhe.

Para especificar o valor do limite no Workflow Studio, selecione Definir um limite de falha tolerado em Configuração adicional, no campo Configurações de tempo de execução.

Porcentagem de falha tolerada

Define a porcentagem de itens com falha a serem tolerados em uma Execução de mapa. A Execução de mapa falhará se esse valor for excedido. O Step Functions calcula a porcentagem de itens com falha como resultado do número total de itens com falha ou com tempo limite esgotado dividido pelo número total de itens. Você deve especificar um valor entre zero e 100. O valor percentual padrão é zero, o que significa que o fluxo de trabalho falhará se qualquer uma das execuções de fluxo de trabalho secundárias falhar ou atingir o tempo limite. Se você especificar a porcentagem como 100, o fluxo de trabalho não falhará, mesmo que todas as execuções de fluxo de trabalho secundárias falhem.

Como alternativa, você pode especificar a porcentagem como um caminho de referência para um par de chave-valor existente na entrada do estado Mapa Distribuído. Esse caminho deve ser um número inteiro positivo entre 0 e 100 no runtime. O caminho de referência é especificado no subcampo `ToleratedFailurePercentagePath`.

Por exemplo, dada a seguinte entrada:

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

Você pode especificar a porcentagem usando um caminho de referência para essa entrada da seguinte forma:

```
{
```

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

> **Important**
>
> Você pode especificar `ToleratedFailurePercentage` ou `ToleratedFailurePercentagePath`, mas não ambos, na definição do estado Mapa Distribuído.

Contagem de falhas toleradas

Define o número de itens com falha a serem tolerados. A Execução de mapa falhará se esse valor for excedido.

Como alternativa, você pode especificar a contagem como um caminho de referência para um par de chave-valor existente na entrada do estado Mapa Distribuído. Esse caminho deve ser um inteiro positivo no runtime. O caminho de referência é especificado no subcampo `ToleratedFailureCountPath`.

Por exemplo, dada a seguinte entrada:

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

Você pode especificar o número usando um caminho de referência para essa entrada da seguinte forma:

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

```
    ...
  }
}
```

> **Important**
>
> Você pode especificar `ToleratedFailureCount` ou `ToleratedFailureCountPath`, mas não ambos, na definição do estado Mapa Distribuído.

Transições

Ao iniciar uma nova execução da máquina de estado, o sistema começa com o estado mencionado no campo de nível superior `StartAt`. Esse campo, uma string. deve corresponder exatamente, inclusive em maiúsculas e minúsculas, ao nome de um estado no fluxo de trabalho.

Depois que um estado é executado, o AWS Step Functions usa o valor do campo `Next` para determinar para qual estado seguinte avançar.

Os campos `Next` também especificam nomes de estados como strings. Essa string diferencia maiúsculas de minúsculas e deve corresponder exatamente ao nome do estado especificado na descrição da máquina de estado.

Por exemplo, o estado a seguir inclui uma transição para `NextState`.

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

A maioria dos estados permite somente uma regra de transição com o campo `Next`. No entanto, determinados estados de controle de fluxo, por exemplo, um estado `Choice`, permitem que você especifique várias regras de transição, cada uma com o próprio campo `Next`. A Amazon States Language fornece detalhes sobre cada um dos tipos de estado que você pode especificar, bem como informações sobre como especificar transições.

Os estados podem ter várias transições de entrada de outros estados.

Esse processo se repete até alcançar um estado terminal (um estado com `"Type": Succeed`, `"Type": Fail` ou `"End": true`) ou até que ocorra um erro de runtime.

Quando você redrive uma execução, ela é considerada uma transição de estado. Além disso, todos os estados que são executados novamente em um redrive também são considerados transições de estado.

As regras a seguir aplicam-se a estados dentro de uma máquina de estado:

- Os estados podem ocorrer em qualquer ordem no bloco delimitador. No entanto, a ordem em que eles estão listados não afeta a ordem em que estão sendo executados. Essa ordem é determinada pelo conteúdo dos estados.
- Em uma máquina de estado, só pode haver um estado designado como o estado `start`. O estado `start` é definido pelo valor do campo `StartAt` na estrutura de nível superior.
- Dependendo da lógica da máquina de estado, por exemplo, se a máquina de estado tiver várias ramificações lógicas, você pode ter mais de um estado `end`.
- Se a máquina de estado tiver somente um estado, esse estado poderá ser início ou fim.

Transições no estado Mapa Distribuído

Ao usar o estado `Map` no modo distribuído, será cobrada uma transição de estado para cada execução de fluxo de trabalho secundária iniciada pelo estado Mapa distribuído. Quando você usa o estado `Map` no modo em linha, não é cobrada uma transição de estado para cada iteração do estado Mapa inline.

Você pode otimizar o custo usando o estado `Map` no modo distribuído e incluir um fluxo de trabalho aninhado na definição do estado `Map`. O estado Mapa Distribuído também agrega mais valor ao iniciar execuções de fluxo de trabalho secundárias do tipo Express. O Step Functions armazena a resposta e o status das execuções de fluxo de trabalho secundárias do Express, o que reduz a necessidade de armazenar dados de execução no CloudWatch Logs. Você também pode obter acesso aos controles de fluxo disponíveis com um estado Mapa Distribuído, como definir limites de erro ou agrupar um grupo de itens em lotes. Para obter informações sobre a definição de preço do Step Functions, consulte Definição de preço para o AWS Step Functions.

Dados da máquina de estado

Os dados da máquina de estado assumem as seguintes formas:

- A entrada inicial em uma máquina de estado
- Os dados passados entre estados

- A saída de uma máquina de estado

Esta seção descreve como os dados de uma máquina de estado são formatados e usados no AWS Step Functions.

Tópicos

Formatos de dados

Os dados da máquina de estado são representados por texto JSON. Você pode fornecer valores para uma máquina de estado usando qualquer tipo de dados compatível com JSON.

> **Note**
>
> - Números em formato de texto JSON conforme a semântica JavaScript. Esses números geralmente correspondem a valores IEEE-854 de dupla precisão.
> - Veja um texto JSON válido a seguir:
>   - Cadeias de caracteres autônomas, delimitadas por aspas
>   - Objetos
>   - Matrizes
>   - Números
>   - Valores boolianos
>   - `null`
> - A saída de um estado se torna a entrada do estado seguinte. No entanto, você pode restringir os estados para que trabalhem em um subconjunto de dados de entrada usando o Processamento de entrada e saída.

Entrada/saída de máquina de estado

Você pode fornecer os dados de entrada iniciais para uma máquina de estado do AWS Step Functions em uma de duas maneiras. Você pode passar os dados para uma ação `StartExecution` ao iniciar uma execução. Você também pode passar os dados para a máquina de estado do console do Step Functions. Os dados iniciais são passados para o estado `StartAt` da máquina de estado. Se não for fornecida nenhuma entrada, o padrão será um objeto vazio (`{}`).

O resultado da execução é retornado pelo último estado (`terminal`). Essa saída aparece como texto JSON no resultado da execução.

Para fluxos de trabalho padrão, você pode recuperar os resultados de execução do histórico usando chamadores externos, por exemplo, na ação `DescribeExecution`. Você pode visualizar os resultados da execução no console do Step Functions.

Para fluxos de trabalho expressos, se você habilitou o registro em log, poderá recuperar os resultados do CloudWatch Logs ou visualizar e depurar as execuções no console do Step Functions. Para obter mais informações, consulte Como registrar usando o CloudWatch Logs e Visualizar e depurar execuções no console do Step Functions.

Você também deve considerar cotas relacionadas à sua máquina de estado. Para obter mais informações, consulte Cotas.

Entrada/saída de estado

A entrada de cada estado compreende o texto JSON do estado precedente ou, para o estado `StartAt`, a entrada da execução. Em alguns estados de controle de fluxo, a entrada ecoa a saída.

No exemplo a seguir, a máquina de estado adiciona dois números ao mesmo tempo.

1. Defina a função 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 a máquina de estado .

```
{
  "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 uma execução com o texto JSON a seguir.

```
{ "numbers": [3, 4] }
```

O estado Add recebe o texto JSON e o passa para a função do Lambda.

A função do Lambda retorna o resultado do cálculo para o estado.

O estado retorna o valor a seguir em sua saída.

```
{ "result": 7 }
```

Como Add é também o estado final na máquina de estado, esse valor é retornado como a saída da máquina de estado.

Se o estado final não retornar nenhuma saída, a máquina de estado retornará um objeto vazio ({}).

Para obter mais informações, consulte Processamento de entrada e saída no Step Functions.

Processamento de entrada e saída no Step Functions

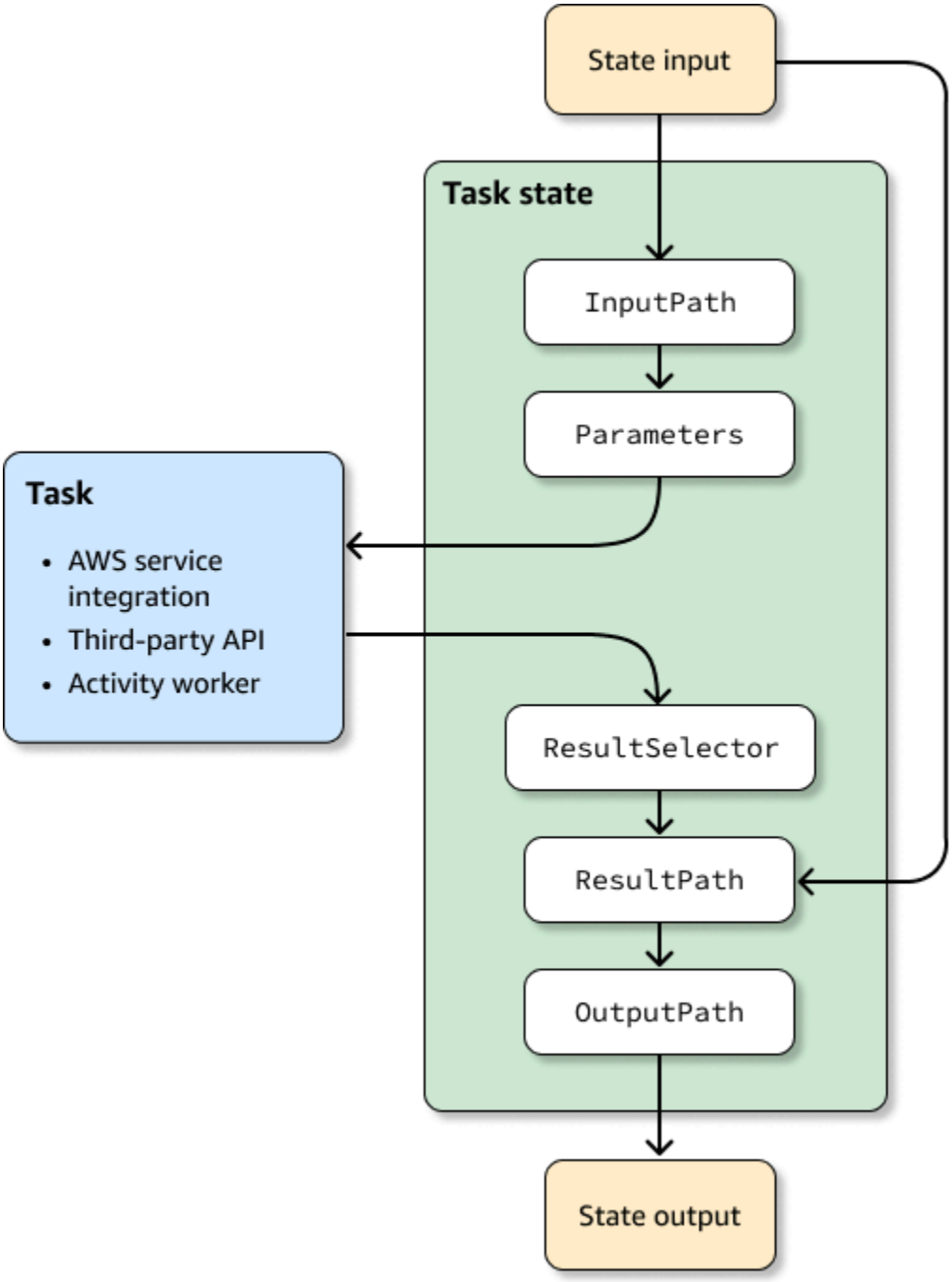
Uma execução do Step Functions recebe um texto JSON como entrada e passa essa entrada para o primeiro estado no fluxo de trabalho. Os estados individuais recebem JSON como entrada e

geralmente passam JSON como saída para o próximo estado. Noções básicas sobre como essas informações fluem de estado para estado, e aprender a filtrar e manipular esses dados, é a chave para efetivamente projetar e implementar fluxos de trabalho no AWS Step Functions.

Na Amazon States Language, esses campos filtram e controlam o fluxo de JSON de estado para estado:

- `InputPath`
- `Parameters`
- `ResultSelector`
- `ResultPath`
- `OutputPath`

O diagrama a seguir mostra como as informações JSON se movem em um estado de tarefa. `InputPath`seleciona quais partes da entrada JSON devem ser passadas para a tarefa do `Task` estado (por exemplo, uma AWS Lambda função). `ResultPath`em seguida, seleciona a combinação da entrada do estado e do resultado da tarefa a ser passada para a saída. `OutputPath`pode filtrar a saída JSON para limitar ainda mais as informações passadas para a saída.



`InputPath`, `Parameters`, `ResultSelector`, `ResultPath` e `OutputPath` manipulam o JSON conforme ele se move por cada estado no fluxo de trabalho.

Cada um pode usar caminhos para selecionar partes do JSON da entrada ou do resultado. Um caminho é uma string, começando com $, que identifica nós com texto JSON. Os caminhos do Step Functions usam JsonPathsintaxe.

> **Tip**
>
> Use o simulador de fluxo de dados no console do Step Functions para testar a sintaxe do caminho JSON, entender melhor como os dados são manipulados em um estado e ver como os dados são transmitidos entre os estados.

> **Tip**
>
> Para implantar um exemplo de fluxo de trabalho que inclui processamento de entrada e saída em seu Conta da AWS, consulte o Módulo 6 - Processamento de entrada e saída do AWS Step Functions workshop.

Tópicos

Caminhos

Na Amazon States Language um caminho é uma string que começa com $, que você pode usar para identificar componentes dentro do texto JSON. Os caminhos seguem a JsonPathsintaxe. É possível especificar um caminho para acessar os subconjuntos de entrada definindo valores para `InputPath`, `ResultPath` e `OutputPath`. Para obter mais informações, consulte Processamento de entrada e saída no Step Functions.

> **Note**
>
> Também é possível especificar um nó JSON da entrada ou objeto de contexto usando caminhos dentro do campo `Parameters` de uma definição de estado. Consulte Transmitir parâmetros para uma API de serviço.

Você deve usar a notação de colchetes se o nome do campo contiver algum caractere que não esteja incluído na `member-name-shorthand` definição da regra JsonPath ABNF. Portanto, para codificar caracteres especiais, como sinais de pontuação (excluindo _), você deve usar a notação de colchetes. Por exemplo, `$.abc['def ghi']`.

Caminhos de referência

Um caminho de referência é um caminho cuja sintaxe é limitada para que possa identificar somente um nó em uma estrutura JSON:

- Você pode acessar os campos de objeto usando somente ponto (`.`) e colchete (`[ ]`).
- Funções como `length()` não são compatíveis.
- Operadores lexicais, que não são simbólicos, como `subsetof` não são compatíveis.
- A filtragem por expressão regular ou por referência a outro valor na estrutura JSON não é compatível.
- Os operadores`@,,,:`, e não `?` são suportados

Por exemplo, se os dados de entrada de estado contivessem os seguintes valores:

```
{
  "foo": 123,
  "bar": ["a", "b", "c"],
  "car": {
      "cdr": true
  }
}
```

Os caminhos de referência a seguir retornariam o seguinte:

```
$.foo => 123
```

```
$.bar => ["a", "b", "c"]
$.car.cdr => true
```

Alguns estados usam caminhos e caminhos de referência para controlar o fluxo de uma máquina de estado ou configurar as definições ou opções de um estado. Para obter mais informações, consulte Modelagem do processamento do caminho de entrada e saída do fluxo de trabalho com simulador de fluxo de dados e Uso eficaz do JSONPath no. AWS Step Functions

Nivelamento de uma matriz de matrizes

Se o estado Paralelo ou Mapa em suas máquinas de estado retornar uma matriz de matrizes, você poderá transformá-las em uma matriz nivelada com o campo ResultSelector. Você pode incluir esse campo na definição do estado Paralelo ou Mapa para manipular o resultado desses estados.

Para nivelar matrizes, use a sintaxe JMESPath `[*]` no campo `ResultSelector`, conforme mostrado no exemplo a seguir.

```
"ResultSelector": {
    "flattenArray.$": "$[*][*]"
  }
```

Para exemplos mostrando como nivelar uma matriz, consulte a Etapa 3 nos seguintes tutoriais:

- Como processar um lote inteiro de dados em uma função do Lambda
- Como processar de itens de dados individuais com uma função do Lambda

InputPath, Parâmetros e ResultSelector

Os campos `InputPath`, `Parameters` e `ResultSelector` fornecem uma maneira de manipular o JSON à medida que ele se move por seu fluxo de trabalho. `InputPath` pode limitar a entrada que será transmitida filtrando a notação JSON por meio de um caminho (consulte Caminhos). O campo `Parameters` permite transmitir um conjunto de pares de chave-valor em que os valores podem ser estáticos, incluídos na definição da máquina de estado, ou selecionados na entrada por meio de um caminho. O `ResultSelector` campo fornece uma forma de manipular o resultado do estado antes de ser `ResultPath` aplicado.

AWS Step Functions aplica o `InputPath` campo primeiro e depois o `Parameters` campo. É possível primeiro filtrar a entrada bruta para uma seleção desejada usando `InputPath` e, então,

aplicar `Parameters` para manipular ainda mais essa entrada ou adicionar novos valores. Em seguida, você pode usar o campo `ResultSelector` para manipular a saída do estado antes da aplicação de `ResultPath`.

> **Tip**
>
> Use o simulador de fluxo de dados no console do Step Functions para testar a sintaxe do caminho JSON, entender melhor como os dados são manipulados em um estado e ver como os dados são transmitidos entre os estados.

InputPath

Use `InputPath` para selecionar uma parte da entrada de estado.

Por exemplo, vamos supor que a entrada para seu estado inclua o seguinte:

```
{
  "comment": "Example for InputPath.",
  "dataset1": {
    "val1": 1,
    "val2": 2,
    "val3": 3
  },
  "dataset2": {
    "val1": "a",
    "val2": "b",
    "val3": "c"
  }
}
```

Você poderá aplicar `InputPath`.

```
"InputPath": "$.dataset2",
```

Com o `InputPath` anterior, o JSON transmitido como a entrada é exibido a seguir.

```
{
  "val1": "a",
```

```
  "val2": "b",
  "val3": "c"
}
```

> **Note**
>
> Um caminho pode gerar uma seleção de valores. Considere o seguinte exemplo.
>
> ```
> { "a": [1, 2, 3, 4] }
> ```
>
> Se você aplicar o caminho `$.a[0:2]`, o resultado será o que é mostrado a seguir.
>
> ```
> [ 1, 2 ]
> ```

Parâmetros

Esta seção descreve as diferentes formas de usar o campo Parâmetros.

Pares de chave/valor

Use o campo `Parameters` para criar um conjunto de pares de chave-valor que serão transmitidos como entrada. Os valores de cada um podem ser valores estáticos, incluídos na definição da máquina de estado, ou selecionados na entrada ou no objeto de contexto por meio de um caminho. Para os pares de chave-valor, em que o valor é selecionado usando um caminho, o nome da chave deve terminar em `.$`.

Por exemplo, vamos supor que você forneça a entrada a seguir.

```
{
  "comment": "Example for Parameters.",
  "product": {
    "details": {
       "color": "blue",
       "size": "small",
       "material": "cotton"
    },
    "availability": "in stock",
    "sku": "2317",
    "cost": "$23"
```

```
  }
}
```

Para selecionar algumas das informações, você poderá especificar esses parâmetros na definição da máquina de estado.

```
"Parameters": {
        "comment": "Selecting what I care about.",
        "MyDetails": {
          "size.$": "$.product.details.size",
          "exists.$": "$.product.availability",
          "StaticValue": "foo"
        }
      },
```

Dada a entrada anterior e o campo `Parameters`, este é o JSON que será transmitido.

```
{
  "comment": "Selecting what I care about.",
  "MyDetails": {
      "size": "small",
      "exists": "in stock",
      "StaticValue": "foo"
  }
},
```

Além da entrada, é possível acessar um objeto JSON especial, conhecido como o objeto de contexto. O objeto de contexto inclui informações sobre a execução da máquina de estado. Consulte Objeto de contexto.

Recursos conectados

O campo `Parameters` também pode transmitir informações para recursos conectados. Por exemplo, se o estado da sua tarefa for orquestrar um AWS Batch trabalho, você poderá passar os parâmetros relevantes da API diretamente para as ações da API desse serviço. Para obter mais informações, consulte:

- Transmitir parâmetros para uma API de serviço
- Como trabalhar com outros serviços

Amazon S3

Se a quantidade de dados da função do Lambda que você está transmitindo entre estados puder ultrapassar 262.144 bytes, recomendamos o uso do Amazon S3 para armazenar os dados e implementar um dos seguintes métodos:

- Use o estado Mapa Distribuído em seu fluxo de trabalho para que o estado `Map` possa ler a entrada diretamente das fontes de dados do Amazon S3. Para ter mais informações, consulte Usar o estado do mapa no modo distribuído.
- Analise o nome do recurso da Amazon (ARN) do bucket no parâmetro `Payload` para obter o nome do bucket e o valor da chave. Para ter mais informações, consulte Use ARNs do Amazon S3 em vez de transmitir grandes cargas.

Como alternativa, é possível ajustar a implementação para passar cargas menores em suas execuções.

ResultSelector

Use o campo `ResultSelector` para manipular um resultado do estado antes da aplicação de `ResultPath`. O campo `ResultSelector` permite criar uma coleção de pares de chave-valor, em que os valores são estáticos ou selecionados a partir do resultado do estado. Usando o campo `ResultSelector`, você pode escolher quais partes do resultado de um estado deseja passar para o campo `ResultPath`.

> **Note**
>
> Com o campo `ResultPath`, você pode adicionar a saída do campo `ResultSelector` à entrada original.

`ResultSelector` é um campo opcional nos seguintes estados:

- Mapa
- Estado da tarefa
- Paralelo

Por exemplo, as integrações de serviço do Step Functions retornam metadados além da carga útil no resultado. `ResultSelector` pode selecionar partes do resultado e mesclá-las com a entrada

de estado com `ResultPath`. Neste exemplo, queremos selecionar apenas `resourceType` e `ClusterId` mesclá-los com a entrada de estado de um createCluster.sync do Amazon EMR. Considerando o seguinte:

```
{
  "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"
  }
}
```

Em seguida, você pode selecionar `resourceType` e `ClusterId` usando `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"
}
```

Com a entrada fornecida, usar `ResultSelector` gera:

```
{
  "OtherDataFromInput": {},
  "EMROutput": {
    "ResourceType": "elasticmapreduce",
    "ClusterId": "AKIAIOSFODNN7EXAMPLE"
  }
}
```

Nivelamento de uma matriz de matrizes

Se o estado Paralelo ou Mapa em suas máquinas de estado retornar uma matriz de matrizes, você poderá transformá-las em uma matriz nivelada com o campo ResultSelector. Você pode incluir esse campo na definição do estado Paralelo ou Mapa para manipular o resultado desses estados.

Para nivelar matrizes, use a sintaxe JMESPath `[*]` no campo `ResultSelector`, conforme mostrado no exemplo a seguir.

```
"ResultSelector": {
    "flattenArray.$": "$[*][*]"
  }
```

Para exemplos mostrando como nivelar uma matriz, consulte a Etapa 3 nos seguintes tutoriais:

- Como processar um lote inteiro de dados em uma função do Lambda
- Como processar de itens de dados individuais com uma função do Lambda

ResultPath

A saída de um estado pode ser uma cópia de sua entrada, o resultado que ele produz (por exemplo, a saída da função do Lambda de um estado `Task`) ou uma combinação da entrada e do resultado. Use `ResultPath` para controlar qual combinação desses itens são passadas para o estado de saída.

Os seguintes tipos de estado podem gerar um resultado e podem incluir `ResultPath`:

- Pass
- Estado da tarefa
- Paralelo

- Mapa

Use `ResultPath` para combinar um resultado de tarefa com entrada de tarefa, ou para selecionar um desses. O caminho que você fornece para `ResultPath` controla quais informações passam para a saída.

> **Note**
>
> `ResultPath` é limitado a usar caminhos de referência, que limitam o escopo para que possa identificar somente um nó em JSON. Consulte Caminhos de referência no Linguagem de estados da Amazon.

Esses exemplos se baseiam na máquina de estado e função do Lambda descritas no tutorial Como criar uma máquina de estado Step Functions que usa Lambda. Trabalhe nesse tutorial e teste saídas diferentes tentando vários caminhos em um campo `ResultPath`.

Use o ResultPath para:

- Use ResultPath para substituir a entrada pelo resultado
- Descartar o resultado e manter a entrada original
- Use ResultPath para incluir o resultado com a entrada
- Use ResultPath para atualizar um nó na entrada com o resultado
- Use ResultPath para incluir erro e entrada em um Catch

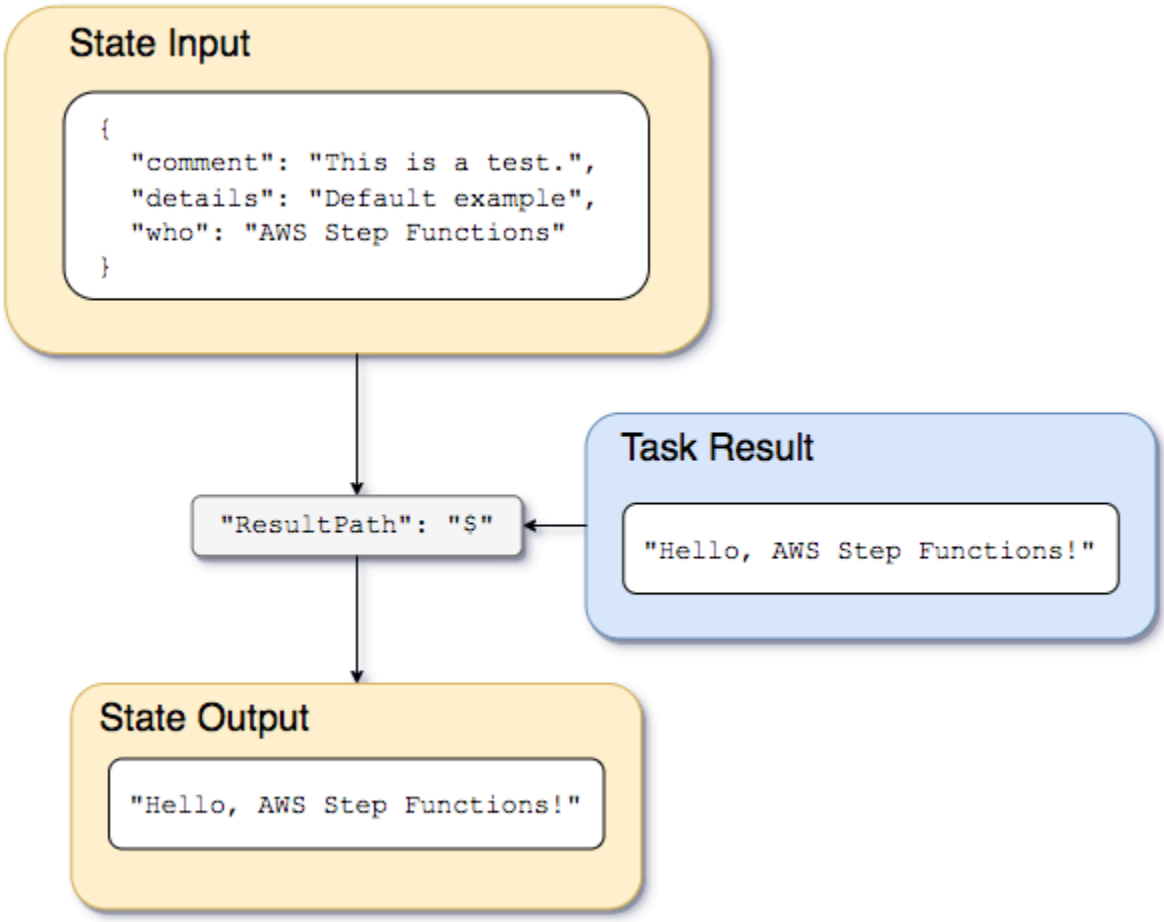
> **Tip**
>
> Use o simulador de fluxo de dados no console do Step Functions para testar a sintaxe do caminho JSON, entender melhor como os dados são manipulados em um estado e ver como os dados são transmitidos entre os estados.

Use ResultPath para substituir a entrada pelo resultado

Se você não especificar um `ResultPath`, o comportamento padrão será como se você tivesse especificado `"ResultPath": "$"`. Como isso informa ao estado para substituir a entrada completa

pelo resultado, o estado de entrada é totalmente substituído pelo resultado proveniente do resultado da tarefa.

O diagrama a seguir mostra como `ResultPath` pode substituir completamente a entrada pelo resultado da tarefa.



Use a máquina de estado e a função do Lambda descritas em Como criar uma máquina de estado Step Functions que usa Lambda e altere o tipo de integração do serviço para Integração SDK AWS para a função do Lambda. Para fazer isso, especifique o nome do recurso da Amazon (ARN) da função do Lambda no campo `Resource` do estado `Task`, conforme mostrado no exemplo a seguir. O uso da integração do AWS SDK garante que o resultado `Task` do estado contenha apenas a saída da função Lambda sem metadados.

```
{
   "StartAt":"CallFunction",
   "States":{
```

```
      "CallFunction": {
         "Type":"Task",
         "Resource":"arn:aws:lambda:us-east-2:123456789012:function:HelloFunction",
         "End": true
      }
   }
}
```

Em seguida, transmita a seguinte entrada:

```
{
  "comment": "This is a test of the input and output of a Task state.",
  "details": "Default example",
  "who": "AWS Step Functions"
}
```

A função do Lambda fornece o seguinte resultado.

```
"Hello, AWS Step Functions!"
```

> **Tip**
>
> Você pode visualizar esse resultados no console do Step Functions. Para fazer isso, na página de Detalhes da execução do console, escolha a função Lambda na Exibição em gráfico. Em seguida, escolha a guia Saída no painel Detalhes da etapa para ver esse resultado.

Se `ResultPath` não for especificado no estado, ou se `"ResultPath": "$"` for definido, a entrada do estado será substituída pelo resultado da função do Lambda e a saída do estado será a seguinte.

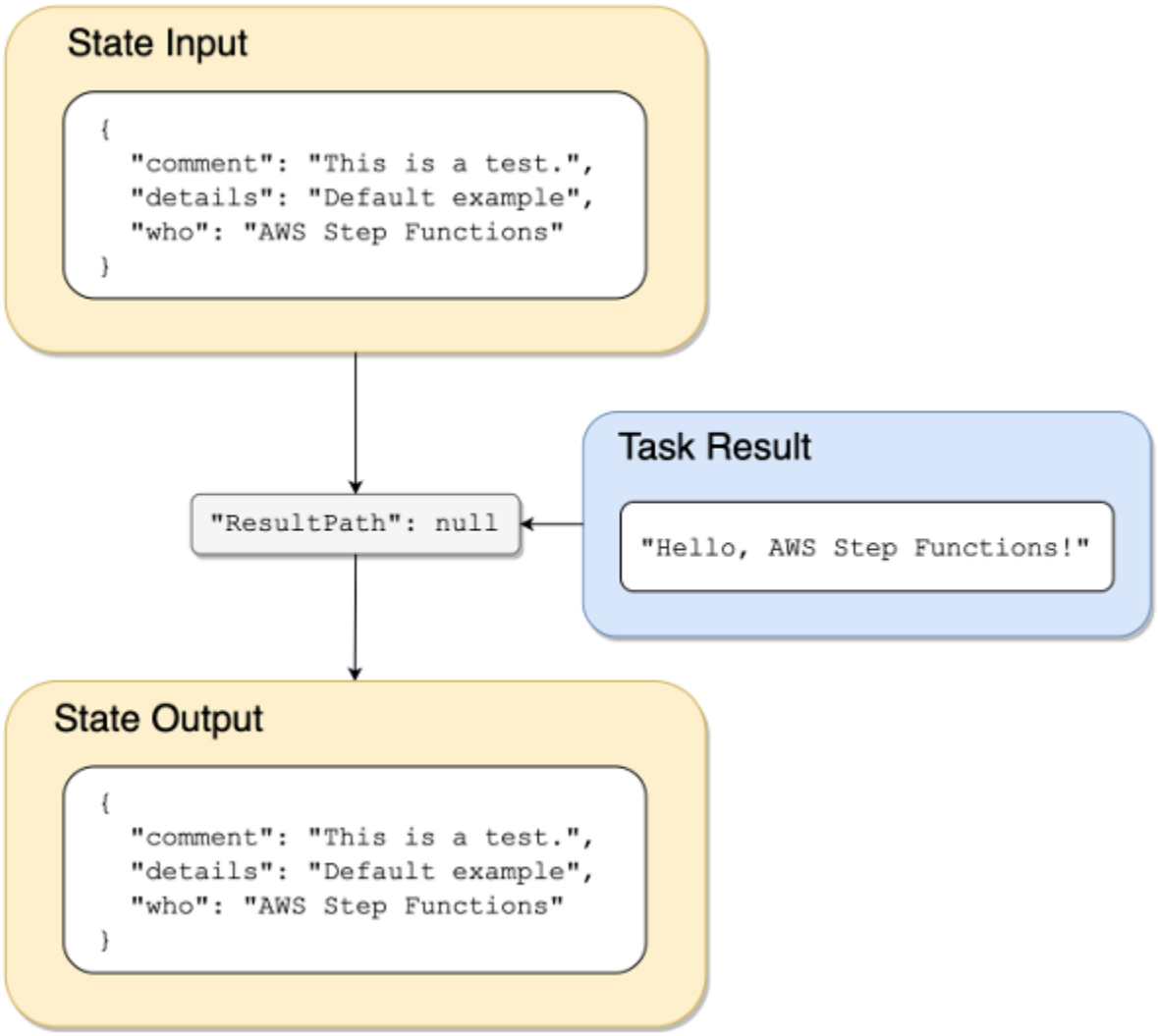
```
"Hello, AWS Step Functions!"
```

> **Note**
>
> `ResultPath` é usado para incluir conteúdo do resultado com a entrada, antes de transmiti-lo para a saída. No entanto, se `ResultPath` não for especificado, o padrão será substituir a entrada completa.

Descartar o resultado e manter a entrada original

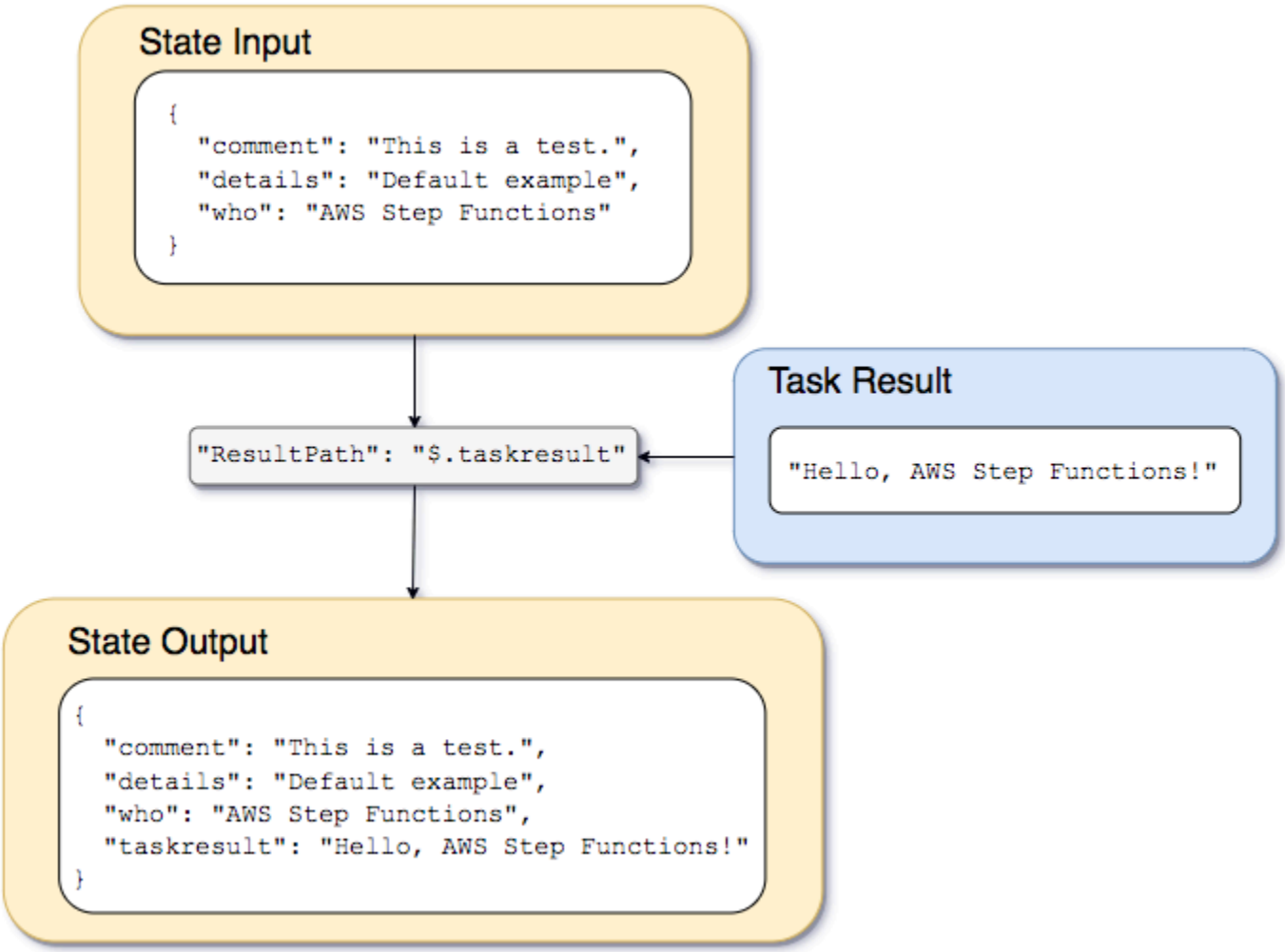
Se você definir o `ResultPath` como `null`, ele passará a entrada original para a saída. Ao usar `"ResultPath": null`, a carga útil de entrada do estado será copiada diretamente para a saída, sem considerar o resultado.

O diagrama a seguir mostra como um `ResultPath` nulo copiará a entrada diretamente para a saída.



Use ResultPath para incluir o resultado com a entrada

O diagrama a seguir mostra como `ResultPath` pode incluir o resultado com a entrada.



Usando a máquina de estado e a função do Lambda descritas no tutorial Como criar uma máquina de estado Step Functions que usa Lambda, podemos transmitir a entrada a seguir.

```
{
  "comment": "This is a test of the input and output of a Task state.",
  "details": "Default example",
  "who": "AWS Step Functions"
}
```

O resultado da função do Lambda é exibido a seguir.

```
"Hello, AWS Step Functions!"
```

Para preservar a entrada, insira o resultado da função do Lambda e passe o JSON combinado para o próximo estado. É possível definir `ResultPath` conforme descrito a seguir.

```
"ResultPath": "$.taskresult"
```

Isso inclui o resultado da função do Lambda com a 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!"
}
```

A saída da função do Lambda é anexada à entrada original como um valor para `taskresult`. A entrada, incluindo o novo valor, é passada para o próximo estado.

Você também pode inserir o resultado em um nó filho da entrada. Defina o `ResultPath` da seguinte forma.

```
"ResultPath": "$.strings.lambdaresult"
```

Inicie uma execução usando a entrada a seguir.

```
{
  "comment": "An input comment.",
  "strings": {
    "string1": "foo",
    "string2": "bar",
    "string3": "baz"
  },
  "who": "AWS Step Functions"
}
```

O resultado da função do Lambda é inserido como um filho do nó `strings` na entrada.

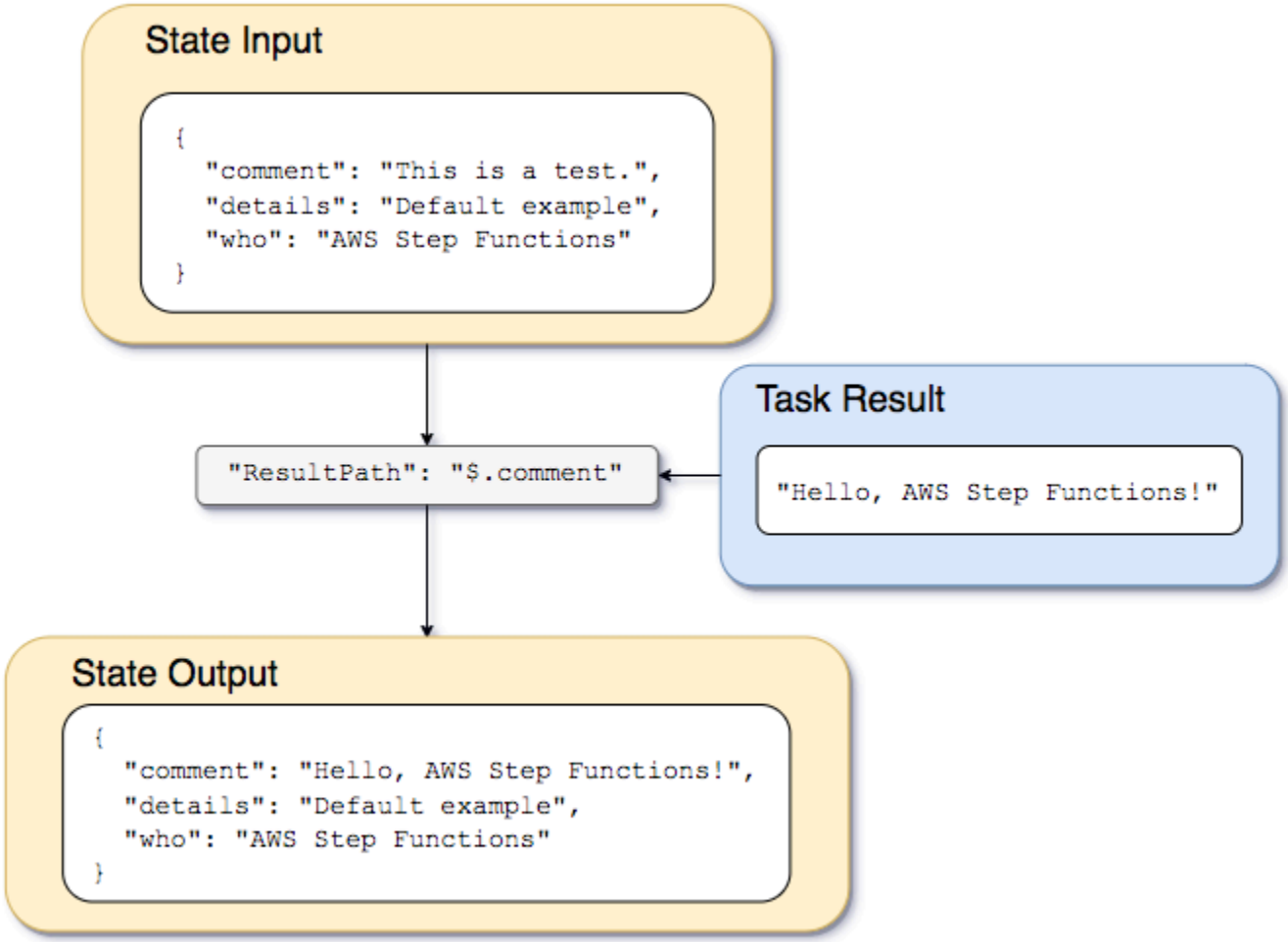
```
{
  "comment": "An input comment.",
  "strings": {
    "string1": "foo",
```

```
    "string2": "bar",
    "string3": "baz",
    "lambdaresult": "Hello, AWS Step Functions!"
  },
  "who": "AWS Step Functions"
}
```

A saída de estado JSON agora inclui a entrada original com o resultado como um nó filho.

Use ResultPath para atualizar um nó na entrada com o resultado

O diagrama a seguir mostra como `ResultPath` pode atualizar o valor de nós na entrada JSON existente com valores do resultado da tarefa.



Usando o exemplo da máquina de estado e da função do Lambda descrito no tutorial Como criar uma máquina de estado Step Functions que usa Lambda, podemos passar a entrada a seguir.

```
{
  "comment": "This is a test of the input and output of a Task state.",
  "details": "Default example",
  "who": "AWS Step Functions"
}
```

O resultado da função do Lambda é exibido a seguir.

```
Hello, AWS Step Functions!
```

Em vez de preservar a entrada e inserir o resultado como um novo nó no JSON, é possível substituir um nó existente.

Por exemplo, assim como a omissão ou a definição de `"ResultPath": "$"` substitui o nó inteiro, você pode especificar um nó individual a ser substituído pelo resultado.

```
"ResultPath": "$.comment"
```

Como o nó `comment` já existe na entrada de estado, definir `ResultPath` como `"$.comment"` substituirá esse nó na entrada pelo resultado da função do Lambda. Sem filtragem adicional por `OutputPath`, é passado para a saída o que está a seguir.

```
{
  "comment": "Hello, AWS Step Functions!",
  "details": "Default behavior example",
  "who": "AWS Step Functions",
}
```

O valor do nó `comment`, `"This is a test of the input and output of a Task state."`, é substituído pelo resultado da função do Lambda: `"Hello, AWS Step Functions!"` na saída de estado.

Use ResultPath para incluir erro e entrada em um **`Catch`**

O tutorial Tratar condições de erro usando uma máquina de estado Step Functions mostra como usar uma máquina de estado para capturar um erro. Em alguns casos, talvez você queira preservar a entrada original com o erro. Use `ResultPath` em um `Catch` para incluir o erro com a entrada original, em vez de substituí-lo.

```
"Catch": [{
```

```
  "ErrorEquals": ["States.ALL"],
  "Next": "NextTask",
  "ResultPath": "$.error"
}]
```

Se a declaração `Catch` anterior detectar um erro, ela incluirá o resultado em um nó `error` dentro da mesma entrada de estado. Por exemplo, com a seguinte entrada:

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

A saída do estado ao detectar um erro é exibida a seguir.

```
{
  "foo": "bar",
  "error": {
    "Error": "Error here"
  }
}
```

Para obter mais informações sobre como tratar erros, consulte o seguinte:

- Tratamento de erros no Step Functions
- Tratar condições de erro usando uma máquina de estado Step Functions

OutputPath

`OutputPath` permite selecionar parte da saída do estado para seguir para o próximo estado. Assim, é possível filtrar informações indesejadas e transmitir somente a parte do JSON que é importante para você.

Se você não especificar um `OutputPath`, o valor padrão será `$`. Isso transmitirá todo o nó JSON (determinado pela entrada de estado, o resultado da tarefa e `ResultPath`) para o próximo estado.

> **Tip**
>
> Use o simulador de fluxo de dados no console do Step Functions para testar a sintaxe do caminho JSON, entender melhor como os dados são manipulados em um estado e ver como os dados são transmitidos entre os estados.

Para mais informações, consulte:

- Caminhos na Amazon States Language
- Exemplos de InputPath, ResultPath e OutputPath
- Transmitir JSON estático como parâmetros
- Processamento de entrada e saída no Step Functions

Exemplos de InputPath, ResultPath e OutputPath

Qualquer estado que não seja Fail ou Succeed pode incluir os campos de processamento de entrada e saída, como `InputPath`, `ResultPath` ou `OutputPath`. Além disso, os estados Aguardar e Choice não são compatíveis com o campo `ResultPath`. Com esses campos, você pode usar um JsonPath para filtrar os dados JSON à medida que eles se movem pelo seu fluxo de trabalho.

Você também pode usar o campo `Parameters` para manipular os dados JSON à medida que eles se movem pelo seu fluxo de trabalho. Para obter informações sobre como usar o `Parameters`, consulte InputPath, Parâmetros e ResultSelector.

Por exemplo, comece com a função do AWS Lambda e a máquina de estado descrita no tutorial Como criar uma máquina de estado Step Functions que usa Lambda. Modifique a máquina de estado para que ela inclua o `InputPath`, o `ResultPath` e o `OutputPath` a seguir.

```
{
  "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 uma execução usando a entrada a seguir.

```
{
  "comment": "An input comment.",
  "data": {
    "val1": 23,
    "val2": 17
  },
  "extra": "foo",
  "lambda": {
    "who": "AWS Step Functions"
  }
}
```

Suponha que os nós `comment` e `extra` possam ser descartados, mas que desejamos incluir a saída da função do Lambda e preservar as informações no nó `data`.

Na máquina de estado atualizada, o estado `Task` é alterado para processar a entrada para a tarefa.

```
"InputPath": "$.lambda",
```

Essa linha na definição de máquina de estado limita a tarefa de entrada para apenas o nó `lambda` da entrada de estado. A função do Lambda recebe somente o objeto JSON `{"who": "AWS Step Functions"}` como entrada.

```
"ResultPath": "$.data.lambdaresult",
```

Esse `ResultPath` diz à máquina de estado para inserir o resultado da função do Lambda em um nó chamado `lambdaresult`, como um filho do nó `data` na entrada da máquina de estado original. Uma vez que não estamos realizando nenhuma outra manipulação na entrada original e no resultado usando `OutputPath`, a saída do estado agora inclui o resultado da função do Lambda com a entrada original.

```
{
  "comment": "An input comment.",
  "data": {
    "val1": 23,
    "val2": 17,
    "lambdaresult": "Hello, AWS Step Functions!"
  },
  "extra": "foo",
```

```
  "lambda": {
    "who": "AWS Step Functions"
  }
}
```

Porém, nosso objetivo era preservar apenas o nó `data` e incluir o resultado da função do Lambda. O `OutputPath` filtra esse JSON combinado antes de transmiti-lo para a saída do estado.

```
"OutputPath": "$.data",
```

Isso seleciona apenas o nó `data` da entrada original (incluindo o filho `lambdaresult` inserido por `ResultPath`) a ser passada para a saída. A saída do estado é filtrada como está a seguir.

```
{
  "val1": 23,
  "val2": 17,
  "lambdaresult": "Hello, AWS Step Functions!"
}
```

Nesse estado `Task`:

1. O `InputPath` envia somente o nó `lambda` da entrada para a função do Lambda.
2. `ResultPath` insere o resultado como um filho do nó `data` na entrada original.
3. O `OutputPath` filtra a entrada de estado (que agora inclui o resultado da função do Lambda) para que ele transmita somente o nó `data` para a saída de estado.

Example Para manipular a entrada, o resultado e a saída final da máquina de estado original usando JsonPath

Considere a seguinte máquina de estado que verifica a identidade e o endereço de um solicitante de seguro.

> **Note**
>
> Para ver o exemplo completo, consulte Como usar o JSON Path no 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
            }
          }
        }
      ]
    }
  }
}
```

Se você executar essa máquina de estado usando a entrada a seguir, a execução falhará porque as funções do Lambda que realizam a verificação esperam apenas os dados que precisam ser

verificados como entrada. Portanto, você deve especificar os nós que contêm as informações a serem verificadas usando um JsonPath apropriado.

```
{
  "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 o nó que a função do Lambda *`check-identity`* deve usar, use o campo `InputPath` da seguinte forma:

```
"InputPath": "$.data.identity"
```

E para especificar o nó que a função do Lambda *check-address* deve usar, use o campo `InputPath` da seguinte forma:

```
"InputPath": "$.data.address"
```

Agora, se você quiser armazenar o resultado da verificação na entrada original da máquina de estado, use o campo `ResultPath` da seguinte forma:

```
"ResultPath": "$.results"
```

No entanto, se você precisar apenas dos resultados de identidade e verificação e descartar a entrada original, use o campo `OutputPath` da seguinte forma:

```
"OutputPath": "$.results"
```

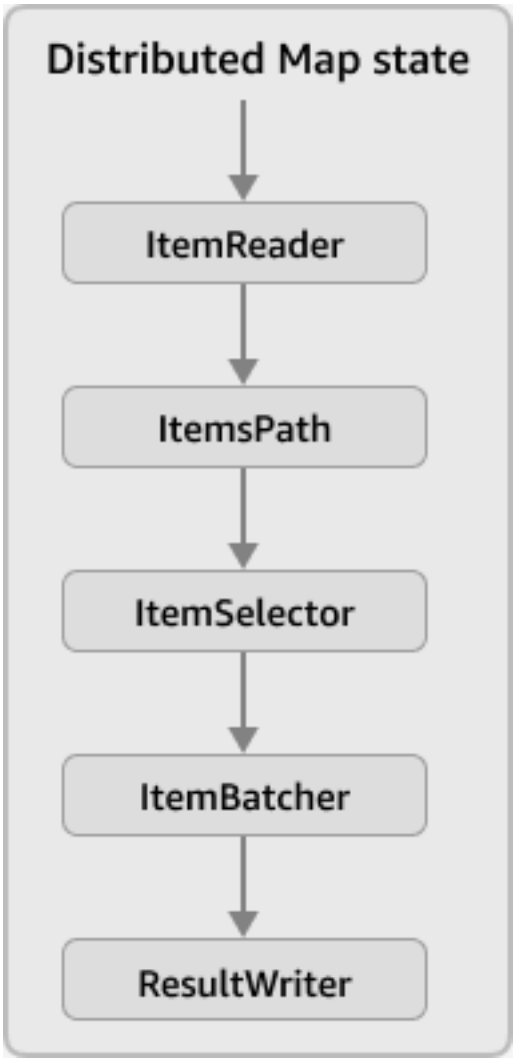
Para obter mais informações, consulte Processamento de entrada e saída no Step Functions.

Mapear campos de entrada e saída do estado

Os estados `Map` iteram simultaneamente em uma coleção de itens em um conjunto de dados, como uma matriz JSON, uma lista de objetos do Amazon S3 ou as linhas de um arquivo CSV em um bucket do Amazon S3. Um conjunto de etapas é repetido para cada item da coleção. Você pode configurar a entrada que o estado `Map` recebe e a saída que ele gera usando esses campos. O Step Functions aplica cada campo em seu estado Mapa Distribuído na ordem mostrada na lista e ilustração a seguir:

> **Note**
>
> Com base no seu caso de uso, pode não ser necessário aplicar todos esses campos.

1. ItemReader
2. ItemsPath
3. ItemSelector
4. ItemBatcher
5. ResultWriter



> **Note**
>
> Esses campos de entrada e saída do estado `Map` estão atualmente indisponíveis no simulador de fluxo de dados no console do Step Functions.

ItemReader

O campo `ItemReader` é um objeto JSON, que especifica um conjunto de dados e sua localização. Um estado Mapa Distribuído usa esse conjunto de dados como entrada. O exemplo a seguir mostra a sintaxe do campo `ItemReader` no caso de um conjunto de dados ser um arquivo CSV armazenado em um bucket do Amazon S3.

```
"ItemReader": {
```

```
    "ReaderConfig": {
        "InputType": "CSV",
        "CSVHeaderLocation": "FIRST_ROW"
    },
    "Resource": "arn:aws:states:::s3:getObject",
    "Parameters": {
        "Bucket": "myBucket",
        "Key": "csvDataset/ratings.csv"
    }
}
```

> **Tip**
>
> No Workflow Studio, você especifica o conjunto de dados e sua localização no campo Origem do item.

Índice

Conteúdo do campo ItemReader

Dependendo do seu conjunto de dados, o conteúdo do campo `ItemReader` varia. Por exemplo, se seu conjunto de dados for uma matriz JSON transmitida de uma etapa anterior do fluxo de trabalho, o campo `ItemReader` será omitido. Se seu conjunto de dados for uma fonte de dados do Amazon S3, esse campo conterá os seguintes subcampos.

`ReaderConfig`

Um objeto JSON que especifica os seguintes detalhes:

- `InputType`

  Especifica o tipo de fonte de dados do Amazon S3, como arquivo CSV, objeto, arquivo JSON ou uma lista do inventário Amazon S3. No Workflow Studio, você pode selecionar um tipo de entrada na lista suspensa de origem do item do Amazon S3, sob o campo Origem do item.

- `CSVHeaderLocation`

> **Note**
>
> Esse campo deverá ser especificado somente se um arquivo CSV for usado como conjunto de dados.

Aceita um dos seguintes valores para especificar a localização do cabeçalho da coluna:

> **Important**
>
> Atualmente, o Step Functions é compatível com cabeçalhos CSV de até 10 KB.

- `FIRST_ROW` — use essa opção se a primeira linha do arquivo for o cabeçalho.
- `GIVEN` — use essa opção para especificar o cabeçalho na definição da máquina de estado. Por exemplo, se seu arquivo CSV contém os seguintes dados.

```
1,307,3.5,1256677221
1,481,3.5,1256677456
1,1091,1.5,1256677471
...
```

Forneça a seguinte matriz JSON como cabeçalho CSV.

```
"ItemReader": {
    "ReaderConfig": {
        "InputType": "CSV",
        "CSVHeaderLocation": "GIVEN",
        "CSVHeaders": [
            "userId",
            "movieId",
            "rating",
            "timestamp"
        ]
    }
}
```

> **Tip**
>
> No Workflow Studio, você pode encontrar essa opção em Configuração adicional, no campo Origem do item.

- `MaxItems`

  Limita o número de itens de dados transmitidos para o estado `Map`. Por exemplo, suponha que você vá fornecer um arquivo CSV contendo 1.000 linhas e especificar um limite de 100. Em seguida, o intérprete transmite apenas 100 linhas para o estado `Map`. O estado `Map` processa os itens em ordem sequencial, começando depois da linha do cabeçalho.

  Por padrão, o estado `Map` itera todos os itens no conjunto de dados especificado.

  > **Note**
  >
  > Atualmente, você pode especificar um limite de até 100.000.000. O estado Mapa Distribuído interrompe a leitura de itens além desse limite.

  > **Tip**
  >
  > No Workflow Studio, você pode encontrar essa opção em Configuração adicional, no campo Origem do item.

  Como alternativa, você pode especificar um caminho de referência para um par de chave-valor existente na entrada do estado Mapa Distribuído. Esse caminho deve ser resolvido como um inteiro positivo. O caminho de referência é especificado no subcampo `MaxItemsPath`.

  > **Important**
  >
  > Você pode especificar o `MaxItems` ou o subcampo `MaxItemsPath`, mas não ambos.

`Resource`

A ação da API do Amazon S3 que o Step Functions deve invocar dependendo do conjunto de dados especificado.

`Parameters`

Um objeto JSON que especifica o nome do bucket do Amazon S3 e a chave do objeto em que o conjunto de dados está armazenado.

> **Important**
>
> Certifique-se de que seus buckets do Amazon S3 estejam na mesma Conta da AWS e Região da AWS que a máquina de estado.

Exemplos de conjuntos de dados

Você pode especificar uma das seguintes opções como conjunto de dados:

- Matriz JSON de uma etapa anterior
- Uma lista de objetos do Amazon S3
- Arquivo JSON em um bucket do Amazon S3
- Arquivo CSV em um bucket do Amazon S3
- Lista do Inventário Amazon S3

> **Important**
>
> O Step Functions precisa das devidas permissões para acessar conjuntos de dados do Amazon S3 que você usa. Para obter informações sobre políticas do IAM para o conjunto de dados, consulte Políticas do IAM para conjuntos de dados.

Matriz JSON de uma etapa anterior

Um estado Mapa Distribuído pode aceitar uma entrada JSON transmitida de uma etapa anterior no fluxo de trabalho. Essa entrada deve ser ou conter uma matriz dentro de um nó específico. Para selecionar um nó que contenha a matriz, você pode usar o campo `ItemsPath`.

Para processar itens individuais na matriz, o estado Mapa Distribuído inicia a execução de um fluxo de trabalho secundário para cada item da matriz. As guias a seguir mostram exemplos da entrada

transmitidas para o estado `Map` e a entrada correspondente para a execução de um fluxo de trabalho secundário.

> **Note**
>
> O Step Functions omite o campo `ItemReader` quando o conjunto de dados é uma matriz JSON de uma etapa anterior.

Input passed to the Map state

Considere a seguinte matriz JSON de três itens.

```
"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

O estado Mapa Distribuído inicia três execuções do fluxo de trabalho secundário. Cada execução recebe um item de matriz como entrada. O exemplo a seguir mostra a entrada recebida pela execução de um fluxo de trabalho secundário.

```
{
  "verdict": "true",
  "statement_date": "6/11/2008",
  "statement_source": "speech"
```

```
}
```

Exemplo de objetos do Amazon S3

Um estado Mapa Distribuído pode iterar os objetos que são armazenados em um bucket do Amazon S3. Quando a execução do fluxo de trabalho atinge o estado `Map`, o Step Functions invoca a ação da API ListObjectsV2, que retorna uma matriz dos metadados do objeto do Amazon S3. Nessa matriz, cada item contém dados, como ETag e Key, para os dados armazenados no bucket.

Para processar itens individuais na matriz, o estado Mapa Distribuído inicia a execução de um fluxo de trabalho secundário. Por exemplo, suponha que seu bucket do Amazon S3 contenha 100 imagens. Então, a matriz retornada após invocar a ação da API `ListObjectsV2` contém 100 itens. O estado Mapa Distribuído inicia 100 execuções de fluxo de trabalho secundário para processar cada item da matriz.

> **Note**
>
> - Atualmente, o Step Functions também inclui um item para cada pasta criada em um bucket do Amazon S3 específico usando o console do Amazon S3. Isso resulta em uma execução adicional de fluxo de trabalho secundário, iniciada pelo estado Mapa Distribuído. Para evitar a criação de uma execução adicional de fluxo de trabalho secundário para a pasta, recomendamos que você use a AWS CLI para criar pastas. Para obter informações, consulte Comandos de alto nível do Amazon S3 no Guia do usuário da AWS Command Line Interface.
> - O Step Functions precisa das devidas permissões para acessar conjuntos de dados do Amazon S3 que você usa. Para obter informações sobre políticas do IAM para o conjunto de dados, consulte Políticas do IAM para conjuntos de dados.

As guias a seguir mostram exemplos da sintaxe do campo `ItemReader` e da entrada transmitida para a execução de um fluxo de trabalho secundário para esse conjunto de dados.

ItemReader syntax

Nesse exemplo, foi mostrado como organizar seus dados, que incluem imagens, arquivos JSON e objetos, dentro de um prefixo nomeado `processData` em um bucket do Amazon S3 designado `myBucket`.

```
"ItemReader": {
    "Resource": "arn:aws:states:::s3:listObjectsV2",
    "Parameters": {
        "Bucket": "myBucket",
        "Prefix": "processData"
    }
}
```

Input passed to a child workflow execution

O estado Mapa Distribuído iniciará quantas execuções de fluxo de trabalho secundário forem necessárias para o número de itens no bucket do Amazon S3. O exemplo a seguir mostra a entrada recebida pela execução de um fluxo de trabalho secundário.

```
{
  "Etag": "\"05704fbdccb224cb01c59005bebbad28\"",
  "Key": "processData/images/n02085620_1073.jpg",
  "LastModified": 1668699881,
  "Size": 34910,
  "StorageClass": "STANDARD"
}
```

Arquivo JSON em um bucket do Amazon S3

Um estado Mapa Distribuído pode aceitar um arquivo JSON armazenado em um bucket do Amazon S3 como um conjunto de dados. O arquivo JSON deve conter uma matriz.

Quando a execução do fluxo de trabalho atinge o estado `Map`, o Step Functions invoca a ação da API GetObject para buscar o arquivo JSON especificado. O estado `Map` então itera cada item na matriz e inicia a execução de um fluxo de trabalho secundário para cada item. Por exemplo, se seu arquivo JSON contiver 1.000 itens de matriz, o estado `Map` iniciará 1.000 execuções de fluxo de trabalho secundário.

> **Note**
>
> - A entrada usada para iniciar a execução de um fluxo de trabalho secundário não pode exceder 256 KB. No entanto, o Step Functions oferece suporte à leitura de um item de até 8 MB de um arquivo CSV ou JSON quando você aplica o campo `ItemSelector` opcional para reduzir o tamanho de um item.

- Atualmente, o Step Functions oferece suporte de no máximo 10 GB a arquivos individuais em um relatório do Inventário Amazon S3. No entanto, o Step Functions é capaz de processar mais de 10 GB se o tamanho de cada arquivo individual é inferior a esse valor.
- O Step Functions precisa das devidas permissões para acessar conjuntos de dados do Amazon S3 que você usa. Para obter informações sobre políticas do IAM para o conjunto de dados, consulte Políticas do IAM para conjuntos de dados.

As guias a seguir mostram exemplos da sintaxe do campo `ItemReader` e da entrada transmitida para a execução de um fluxo de trabalho secundário para esse conjunto de dados.

Por exemplo, imagine que você tenha um arquivo JSON chamado *`factcheck.json`*. Você armazenou esse arquivo em um prefixo chamado *`jsonDataset`*, em um bucket do Amazon S3. A seguir, veja um exemplo do conjunto de dados 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

O estado Mapa Distribuído iniciará quantas execuções de fluxo de trabalho secundário forem necessárias para o número de itens de matriz presentes no arquivo JSON. O exemplo a seguir mostra a entrada recebida pela execução de um fluxo de trabalho secundário.

```
{
  "verdict": "true",
  "statement_date": "6/11/2008",
  "statement_source": "speech"
}
```

Arquivo CSV em um bucket do Amazon S3

Um estado Mapa Distribuído pode aceitar um arquivo CSV armazenado em um bucket do Amazon S3 como um conjunto de dados. Se você usar um arquivo CSV como conjunto de dados, precisará especificar um cabeçalho de coluna CSV. Para obter informações sobre como especificar um cabeçalho CSV, consulte Conteúdo do campo ItemReader.

Como não há um formato padronizado para criar e manter dados em arquivos CSV, o Step Functions analisa os arquivos CSV com base nas seguintes regras:

- As vírgulas (,) são um delimitador que separa campos individuais.
- As novas linhas são um delimitador que separa registros individuais.
- Os campos são tratados como strings. Para conversões de tipo de dados, use a função intrínseca `States.StringToJson` em ItemSelector.
- Não são necessárias aspas duplas (” “) para encerrar strings. No entanto, strings delimitadas por aspas duplas podem conter vírgulas e novas linhas que não funcionam como delimitadores.
- Repita as aspas duplas para evitá-las.
- Se o número de campos em uma linha for menor que o número de campos no cabeçalho, o Step Functions fornecerá strings vazias para os valores que estão faltando.

- Se o número de campos em uma linha for maior que aquele no cabeçalho, o Step Functions ignorará os campos adicionais.

Para obter mais informações sobre como Step Functions analisa um arquivo CSV, consulte Example of parsing an input CSV file.

Quando a execução do fluxo de trabalho atinge o estado `Map`, o Step Functions invoca a ação da API GetObject para buscar o arquivo CSV especificado. O estado `Map` então itera cada linha no arquivo CSV e inicia a execução de um fluxo de trabalho secundário para processar os itens em cada linha. Por exemplo, suponha que você vá fornecer um arquivo CSV contendo 100 linhas como entrada. Então, o intérprete transmitirá cada linha para o estado `Map`. O estado `Map` processa os itens em ordem serial, começando depois da linha do cabeçalho.

> **Note**
>
> - A entrada usada para iniciar a execução de um fluxo de trabalho secundário não pode exceder 256 KB. No entanto, o Step Functions oferece suporte à leitura de um item de até 8 MB de um arquivo CSV ou JSON quando você aplica o campo `ItemSelector` opcional para reduzir o tamanho de um item.
> - Atualmente, o Step Functions oferece suporte de no máximo 10 GB a arquivos individuais em um relatório do Inventário Amazon S3. No entanto, o Step Functions é capaz de processar mais de 10 GB se o tamanho de cada arquivo individual é inferior a esse valor.
> - O Step Functions precisa das devidas permissões para acessar conjuntos de dados do Amazon S3 que você usa. Para obter informações sobre políticas do IAM para o conjunto de dados, consulte Políticas do IAM para conjuntos de dados.

As guias a seguir mostram exemplos da sintaxe do campo `ItemReader` e da entrada transmitida para a execução de um fluxo de trabalho secundário para esse conjunto de dados.

ItemReader syntax

Por exemplo, digamos que você tenha um arquivo CSV chamado *`ratings.csv`*. Você armazenou esse arquivo em um prefixo designado *`csvDataset`*, em um bucket do 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

O estado Mapa Distribuído iniciará quantas execuções de fluxo de trabalho secundário forem necessárias para o número de linhas presentes no arquivo CSV, excluindo a linha de cabeçalho, se presente no arquivo. O exemplo a seguir mostra a entrada recebida pela execução de um fluxo de trabalho secundário.

```
{
  "rating": "3.5",
  "movieId": "307",
  "userId": "1",
  "timestamp": "1256677221"
}
```

Exemplo de inventário do S3

Um estado Mapa Distribuído pode aceitar um manifesto do Inventário Amazon S3 armazenado em um bucket do Amazon S3 como um conjunto de dados.

Quando a execução do fluxo de trabalho atinge o estado `Map`, o Step Functions invoca a ação da API GetObject para buscar o arquivo do manifesto do Inventário Amazon S3. O estado `Map` então itera os objetos no inventário para retornar uma matriz de metadados de objetos do Inventário Amazon S3.

> **Note**
>
> - Atualmente, o Step Functions oferece suporte de no máximo 10 GB a arquivos individuais em um relatório do Inventário Amazon S3. No entanto, o Step Functions é capaz de processar mais de 10 GB se o tamanho de cada arquivo individual é inferior a esse valor.

- O Step Functions precisa das devidas permissões para acessar conjuntos de dados do Amazon S3 que você usa. Para obter informações sobre políticas do IAM para o conjunto de dados, consulte Políticas do IAM para conjuntos de dados.

Veja a seguir o exemplo de um arquivo de inventário no formato CSV. Esse arquivo inclui os objetos chamados `csvDataset` e `imageDataset`, que são armazenados em um bucket do Amazon S3 com o nome `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

Atualmente, o Step Functions não oferece suporte ao uso do relatório do Inventário Amazon S3 definido pelo usuário como um conjunto de dados. Você também deve se certificar de que o formato de saída do seu relatório do Inventário Amazon S3 seja CSV. Para obter mais informações sobre os Inventários Amazon S3 e como configurá-los, consulte Inventário Amazon S3 no Guia do usuário do Amazon S3.

O exemplo a seguir de um arquivo de manifesto de inventário mostra os cabeçalhos CSV dos metadados do objeto de inventário.

```
{
  "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"
```

```
  } ]
}
```

As guias a seguir mostram exemplos da sintaxe do campo `ItemReader` e da entrada transmitida para a execução de um fluxo de trabalho secundário para esse conjunto de dados.

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"
}
```

Dependendo dos campos selecionados ao configurar o relatório do Inventário Amazon S3, o conteúdo do arquivo `manifest.json` pode variar do exemplo mostrado.

Políticas do IAM para conjuntos de dados

Ao criar fluxos de trabalho com o console do Step Functions, o Step Functions pode gerar automaticamente políticas do IAM com base nos recursos na definição de fluxo de trabalho. Essas políticas incluem os privilégios mínimos necessários para permitir que o perfil da máquina de estado invoque a ação da API `StartExecution` para o estado Mapa Distribuído. Essas políticas também

incluem os privilégios mínimos necessários para que o Step Functions acesse recursos da AWS, como buckets e objetos do Amazon S3 e funções do Lambda. É altamente recomendável que você inclua apenas as permissões que forem necessárias em suas políticas do IAM. Por exemplo, se o fluxo de trabalho incluir um estado `Map` no modo distribuído, defina o escopo de suas políticas até o bucket e a pasta específicos do Amazon S3 que contêm o conjunto de dados.

> **Important**
>
> Se você especificar um bucket e um objeto do Amazon S3, ou prefixo, com um caminho de referência para um par de valores-chave existente na entrada do estado Mapa Distribuído, certifique-se de atualizar as políticas de IAM do fluxo de trabalho. Defina o escopo das políticas até o bucket e os nomes de objetos para os quais o caminho é resolvido em runtime.

Os exemplos de políticas do IAM a seguir concedem os privilégios mínimos necessários para acessar os conjuntos de dados do Amazon S3 usando as ações de API ListObjectsV2 e GetObject.

Example Política do IAM para objeto do Amazon S3 como conjunto de dados

O exemplo a seguir mostra uma política do IAM que concede os privilégios mínimos para acessar os objetos organizados em *`processImages`* em um bucket do Amazon S3 chamado *`myBucket`*.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::myBucket"
            ],
            "Condition": {
                "StringLike": {
                    "s3:prefix": [
                        "processImages"
                    ]
                }
            }
```

```
        }
    ]
}
```

Example Política do IAM para um arquivo CSV como conjunto de dados

O exemplo a seguir mostra uma política do IAM que concede os privilégios mínimos para acessar um arquivo CSV chamado *ratings.csv*.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::myBucket/csvDataset/ratings.csv"
            ]
        }
    ]
}
```

Example Política do IAM para um inventário Amazon S3 como conjunto de dados

O exemplo a seguir mostra uma política do IAM que concede os privilégios mínimos para acessar um relatório de inventário 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

Use o campo `ItemsPath` para selecionar uma matriz dentro de uma entrada JSON fornecida para um estado `Map`. O estado `Map` repete um conjunto de etapas para cada item na matriz. Por padrão, um estado `Map` define `ItemsPath` como `$`, o que seleciona toda a entrada. Se a entrada para o estado `Map` for uma matriz JSON, ela executará uma iteração para cada item na matriz, transmitindo esse item para a iteração como entrada.

> **Note**
>
> Você só poderá usar `ItemsPath` no estado Mapa Distribuído se usar uma entrada JSON transmitida de um estado anterior no fluxo de trabalho.

Você pode usar o campo `ItemsPath` para especificar um local na entrada que aponta para a matriz JSON usada para iterações. O valor de `ItemsPath` deve ser um caminho de referência e esse caminho deve apontar para a matriz JSON. Por exemplo, considere a entrada para um estado `Map` que inclua duas matrizes, como o exemplo a seguir.

```
{
  "ThingsPiratesSay": [
    {
      "say": "Avast!"
    },
    {
      "say": "Yar!"
    },
    {
      "say": "Walk the Plank!"
    }
  ],
  "ThingsGiantsSay": [
    {
      "say": "Fee!"
    },
    {
      "say": "Fi!"
```

```
    },
    {
      "say": "Fo!"
    },
    {
      "say": "Fum!"
    }
  ]
}
```

Nesse caso, você pode especificar qual matriz usar para iterações do estado `Map` selecionando-a com `ItemsPath`. A definição de máquina de estado a seguir especifica a matriz `ThingsPiratesSay` na entrada usando `ItemsPath`. Em seguida, ela executa uma iteração do estado de passagem `SayWord` para cada item na matriz `ThingsPiratesSay`.

```
{
  "StartAt": "PiratesSay",
  "States": {
    "PiratesSay": {
      "Type": "Map",
      "ItemsPath": "$.ThingsPiratesSay",
      "ItemProcessor": {
         "StartAt": "SayWord",
         "States": {
           "SayWord": {
             "Type": "Pass",
             "End": true
           }
         }
      },
      "End": true
    }
  }
}
```

Ao processar a entrada, o estado `Map` aplica `ItemsPath` depois do `InputPath`. Ele é operado na entrada efetiva para o estado depois que `InputPath` filtra a entrada.

Para obter mais informações sobre estados `Map`, consulte os tópicos a seguir:

- Estado do mapa
- Modos de processamento do estado do mapa

- Repetir uma ação usando o estado Mapa em linha
- Processamento de entrada e saída do estado Map inline.

ItemSelector

Por padrão, a entrada efetiva para o estado `Map` é o conjunto de itens de dados individuais presentes na entrada de estado bruto. O campo `ItemSelector` permite que você substitua os valores dos itens de dados antes que eles sejam transmitidos para o estado `Map`. Para substituir os valores, especifique uma entrada JSON válida que contenha um coleção de pares de chave-valor. Esses pares podem ser valores estáticos fornecidos na definição da máquina de estado, valores selecionados da entrada de estado usando um caminho ou valores acessados a partir do objeto de contexto.

Se você especificar pares de chave-valor usando um caminho ou objeto de contexto, o nome da chave deverá terminar em `.$`.

> **Note**
>
> O campo `ItemSelector` substitui o campo `Parameters` dentro do estado `Map`. Se você usa o campo `Parameters` em suas definições do estado `Map` para criar entradas personalizadas, é altamente recomendável substituí-las por `ItemSelector`.

Você pode especificar o campo `ItemSelector` tanto em um estado Mapa inline quanto Mapa distribuído.

Por exemplo, considere a seguinte entrada JSON contendo uma matriz de três itens dentro do nó `imageData`. Para cada iteração do estado *`Map`*, um item de matriz é passado para a iteração como entrada.

```
[
  {
    "resize": "true",
    "format": "jpg"
  },
  {
    "resize": "false",
    "format": "png"
  },
```

```
  {
    "resize": "true",
    "format": "jpg"
  }
]
```

Usando o campo `ItemSelector`, você pode definir uma entrada JSON personalizada para substituir a entrada original, conforme mostrado no exemplo a seguir. O Step Functions então transmite essa entrada personalizada para cada iteração do estado *`Map`*. A entrada personalizada contém um valor estático para `size` e o valor dos dados de um objeto de contexto para o estado `Map`. O objeto de contexto `$$.Map.Item.Value` contém o valor de cada item de dados individual.

```
{
  "ItemSelector": {
    "size": 10,
    "value.$": "$$.Map.Item.Value"
  }
}
```

O exemplo a seguir mostra a entrada recebida por uma iteração do estado Mapa inline:

```
{
  "size": 10,
  "value": {
    "resize": "true",
    "format": "jpg"
  }
}
```

> **Tip**
>
> Para obter um exemplo completo de um estado Mapa Distribuído usando o campo `ItemSelector`, consulte Conceitos básicos do estado de Mapa Distribuído.

ItemBatcher

O campo `ItemBatcher` é um objeto JSON, que especifica o processamento de um grupo de itens na execução de um único fluxo de trabalho secundário. Use lotes ao processar arquivos CSV ou matrizes JSON grandes, ou conjuntos de objetos grandes do Amazon S3.

O exemplo a seguir mostra a sintaxe do campo `ItemBatcher`. Na sintaxe a seguir, o número máximo de itens que cada execução de fluxo de trabalho secundário deve processar é definido como 100.

```
{
  "ItemBatcher": {
    "MaxItemsPerBatch": 100
  }
}
```

Por padrão, cada item em um conjunto de dados é transmitido como entrada para execuções individuais de fluxos de trabalho secundários. Por exemplo, suponha que você especifique um arquivo JSON como entrada que contém a seguinte 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 a entrada fornecida, cada execução do fluxo de trabalho secundário recebe um item de matriz como entrada. O exemplo a seguir mostra a entrada da execução de um fluxo de trabalho secundário:

```
{
  "verdict": "true",
  "statement_date": "6/11/2008",
  "statement_source": "speech"
```

```
}
```

Para ajudar a otimizar o desempenho e o custo do seu trabalho de processamento, selecione um tamanho de lote que equilibre o número de itens em relação ao tempo de processamento deles. Se você usar lotes, o Step Functions adiciona os itens a uma matriz de Itens. Em seguida, ele transmite a matriz como entrada para a execução de cada fluxo de trabalho secundário. O exemplo a seguir mostra um lote de dois itens transmitido como entrada para a execução de um fluxo de trabalho secundário:

```
{
  "Items": [
    {
      "verdict": "true",
      "statement_date": "6/11/2008",
      "statement_source": "speech"
    },
    {
      "verdict": "false",
      "statement_date": "6/7/2022",
      "statement_source": "television"
    }
  ]
}
```

> **Tip**
>
> Para saber mais sobre como usar o campo `ItemBatcher` em seus fluxos de trabalho, experimente os seguintes tutoriais e workshops:
>
> - Processar um lote inteiro de dados em uma função do Lambda
> - Iterar itens em um lote dentro das execuções do fluxo de trabalho secundário
> - Paralelização em grande escala com Mapa distribuído no Módulo 14 - Processamento de dados do AWS Step Functions Workshop

Índice

Campos para especificar o agrupamento de itens em lotes

Para agrupar itens em lotes, especifique o número máximo de itens por lote, o tamanho máximo do lote ou ambos. É necessário especificar um desses valores para agrupar os itens.

Máximo de itens por lote

Especifica o número máximo de itens que a execução de cada fluxo de trabalho secundário processa. O intérprete limita o número de itens agrupados em lote na matriz `Items` a esse valor. Se você especificar o número e o tamanho do lote, o intérprete reduzirá o número de itens em um lote para evitar que o limite de tamanho do lote especificado seja excedido.

Se você não especificar esse valor, mas fornecer um valor para o tamanho máximo do lote, o Step Functions processará o máximo de itens possível na execução de cada fluxo de trabalho secundário sem exceder o tamanho máximo do lote, em bytes.

Por exemplo, imagine que você esteja executando uma execução com um arquivo JSON de entrada que contenha 1130 nós. Se você especificar um valor máximo de 100 itens a cada lote, o Step Functions criará 12 lotes. Destes, 11 lotes conterão 100 itens cada, enquanto o décimo segundo lote conterá os 30 itens restantes.

Como alternativa, você pode especificar o número máximo de itens para cada lote como um caminho de referência para um par de chave-valor existente na entrada do estado Mapa Distribuído. Esse caminho deve ser resolvido como um inteiro positivo.

Por exemplo, dada a seguinte entrada:

```
{
  "maxBatchItems": 500
}
```

Você pode especificar o número máximo de itens por lote usando um caminho de referência, da seguinte forma:

```
{
  ...
  "Map": {
    "Type": "Map",
    "MaxConcurrency": 2000,
    "ItemBatcher": {
      "MaxItemsPerBatchPath": "$.maxBatchItems"
```

```
    }
    ...
    ...
  }
}
```

> **Important**
>
> Você pode especificar o `MaxItemsPerBatch` ou o subcampo `MaxItemsPerBatchPath`, mas não ambos.

Máximo de KBs por lote

Especifica o tamanho máximo de um lote, em bytes, que é 256 KBs. Se você especificar tanto número quanto o tamanho máximos de um lote, o Step Functions reduzirá o número de itens em um lote para evitar que o limite de tamanho do lote especificado seja excedido.

Como alternativa, você pode especificar o tamanho máximo do lote como um caminho de referência para um par de chave-valor existente na entrada do estado Mapa Distribuído. Esse caminho deve ser resolvido como um inteiro positivo.

> **Note**
>
> Se você usar o agrupamento em lotes e não especificar um tamanho máximo para o lote, o intérprete processará o máximo possível de itens até 256 KB em cada execução de um fluxo de trabalho secundário.

Por exemplo, dada a seguinte entrada:

```
{
  "batchSize": 131072
}
```

Você pode especificar o tamanho máximo do lote usando um caminho de referência, da seguinte forma:

```
{
```

```
  ...
  "Map": {
    "Type": "Map",
    "MaxConcurrency": 2000,
    "ItemBatcher": {
      "MaxInputBytesPerBatchPath": "$.batchSize"
    }
    ...
    ...
  }
}
```

> **Important**
>
> Você pode especificar o `MaxInputBytesPerBatch` ou o subcampo `MaxInputBytesPerBatchPath`, mas não ambos.

Entrada em lote

Opcionalmente, você também pode especificar a inclusão de uma entrada JSON fixa em cada lote transmitido para a execução de cada fluxo de trabalho secundário. O Step Functions mescla essa entrada com aquela para cada execução individual do fluxo de trabalho secundário. Por exemplo, dada a seguinte entrada fixa de uma data de verificação de fatos em uma matriz de itens:

```
"ItemBatcher": {
    "BatchInput": {
        "factCheck": "December 2022"
    }
}
```

Cada execução do fluxo de trabalho secundário recebe o seguinte 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

O campo `ResultWriter` é um objeto JSON que especifica a localização do Amazon S3 onde o Step Functions grava os resultados das execuções de um fluxo de trabalho secundário iniciadas por um estado Mapa Distribuído. Por padrão, o Step Functions não exporta esses resultados.

> **Important**
>
> Certifique-se de que o bucket do Amazon S3 utilizado para exportar os resultados de uma Execução de mapa esteja sob a mesma Conta da AWS e Região da AWS que a sua máquina de estado. Caso contrário, a execução da sua máquina de estado falhará com o erro `States.ResultWriterFailed`.

A exportação dos resultados para um bucket do Amazon S3 será uma operação útil se o tamanho do seu payload de saída exceder 256 KB. O Step Functions consolida todos os dados da execução de um fluxo de trabalho secundário, como a entrada e saída de execução, ARN e status da execução. Em seguida, ele exporta as execuções com o mesmo status para seus respectivos arquivos na localização especificada do Amazon S3. O exemplo a seguir mostra a sintaxe do campo `ResultWriter` de quando você exporta os resultados da execução de um fluxo de trabalho secundário. Nesse exemplo, os resultados são armazenados em um bucket chamado `myOutputBucket`, dentro de um prefixo chamado `csvProcessJobs`.

```
{
  "ResultWriter": {
    "Resource": "arn:aws:states:::s3:putObject",
    "Parameters": {
```

```
      "Bucket": "myOutputBucket",
      "Prefix": "csvProcessJobs"
    }
  }
}
```

> **Tip**
>
> No Workflow Studio, você pode exportar os resultados da execução do fluxo de trabalho secundário selecionando Exportar resultados do estado Mapa para o Amazon S3. Em seguida, forneça o nome do bucket do Amazon S3 e o prefixo para o qual você deseja exportar os resultados.

O Step Functions precisa das permissões apropriadas para acessar o bucket e a pasta para onde você deseja exportar os resultados. Para obter informações sobre as políticas do IAM necessárias, consulte Políticas do IAM para ResultWriter.

Se você exportar os resultados da execução do fluxo de trabalho secundário, a execução do estado Mapa Distribuído retornará o ARN da Execução de mapa e os dados sobre o local de exportação do Amazon S3, no seguinte 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"
  }
}
```

O Step Functions exporta execuções com o mesmo status para seus respectivos arquivos. Por exemplo, se as execuções de fluxo de trabalho secundário resultaram em 500 resultados de êxito e 200 resultados de falha, o Step Functions criará dois arquivos no local especificado do Amazon S3 para os resultados de êxito e falha. Nesse exemplo, o arquivo de resultados de êxito contém os 500 resultados de êxito, enquanto o arquivo de resultados de falha contém os 200 resultados de falha.

Para uma determinada tentativa de execução, o Step Functions cria os seguintes arquivos no local especificado do Amazon S3, dependendo da saída da execução:

- `manifest.json` — contém metadados da Execução de mapa, como local de exportação, ARN e informações sobre os arquivos de resultados.

  Se você tiver redriven uma Execução de mapa, o arquivo `manifest.json` conterá as referências a todas as execuções bem-sucedidas do fluxo de trabalho secundário, em todas as tentativas de uma Execução de mapa. No entanto, esse arquivo contém referências às execuções com falhas e pendentes de um redrive específico.
- `SUCCEEDED_n.json` — contém os dados consolidados de todas as execuções bem-sucedidas do fluxo de trabalho secundário. n representa o número do índice do arquivo. O número do índice começa em 0. Por exemplo, `SUCCEEDED_1.json`.
- `FAILED_n.json` — contém os dados consolidados de todas as execuções do fluxo de trabalho secundário com falha, que atingiram o tempo limite ou foram abortadas. Use esse arquivo para se recuperar a partir de execuções com falha. n representa o índice do arquivo. O número do índice começa em 0. Por exemplo, `FAILED_1.json`.
- `PENDING_n.json` — contém os dados consolidados de todas as execuções do fluxo de trabalho secundário que não foram iniciadas porque a Execução de mapa falhou ou foi abortada. n representa o índice do arquivo. O número do índice começa em 0. Por exemplo, `PENDING_1.json`.

O Step Functions oferece suporte a arquivos de resultados individuais de até 5 GB. Se o tamanho do arquivo exceder 5 GB, o Step Functions criará outro arquivo para gravar os resultados restantes da execução e anexará um número de índice ao nome do arquivo. Por exemplo, se o tamanho do arquivo `Succeeded_0.json` exceder 5 GB, o Step Functions criará um arquivo `Succeeded_1.json` para registrar os resultados restantes.

Se você não especificou a exportação dos resultados da execução do fluxo de trabalho secundário, a execução da máquina de estado retornará uma matriz dos resultados da execução do fluxo de trabalho secundário, conforme mostrado no exemplo a seguir:

> **Note**
>
> Se o tamanho da saída retornada exceder 256 KB, a execução da máquina de estado falhará e retornará um erro `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 do IAM para ResultWriter

Ao criar fluxos de trabalho com o console do Step Functions, o Step Functions pode gerar automaticamente políticas do IAM com base nos recursos na definição de fluxo de trabalho. Essas políticas incluem os privilégios mínimos necessários para permitir que o perfil da máquina de estado invoque a ação da API `StartExecution` para o estado Mapa Distribuído. Essas políticas também incluem os privilégios mínimos necessários para que o Step Functions acesse recursos da AWS, como buckets e objetos do Amazon S3 e funções do Lambda. É altamente recomendável que você inclua apenas as permissões que forem necessárias em suas políticas do IAM. Por exemplo, se o fluxo de trabalho incluir um estado `Map` no modo distribuído, defina o escopo de suas políticas até o bucket e a pasta específicos do Amazon S3 que contêm o conjunto de dados.

> **Important**
>
> Se você especificar um bucket e um objeto do Amazon S3, ou prefixo, com um caminho de referência para um par de valores-chave existente na entrada do estado Mapa Distribuído, certifique-se de atualizar as políticas de IAM do fluxo de trabalho. Defina o escopo das políticas até o bucket e os nomes de objetos para os quais o caminho é resolvido em runtime.

O exemplo de política do IAM a seguir concede os privilégios mínimos necessários para gravar os resultados da execução do fluxo de trabalho secundário em uma pasta chamada *csvJobs* em um bucket do Amazon S3 usando a ação de API PutObject.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListMultipartUploadParts",
                "s3:AbortMultipartUpload"
            ],
            "Resource": [
                "arn:aws:s3:::resultBucket/csvJobs/*"
            ]
        }
    ]
}
```

Se o bucket do Amazon S3 no qual você está gravando o resultado da execução do fluxo de trabalho secundário for criptografado usando uma chave AWS Key Management Service (AWS KMS), você deverá incluir as permissões AWS KMS necessárias em sua política do IAM. Para obter mais informações, consulte Permissões do IAM para AWS KMS key bucket criptografado do Amazon S3.

Como analisar um arquivo CSV de entrada

Como não há um formato padronizado para criar e manter dados em arquivos CSV, o Step Functions analisa os arquivos CSV com base nas seguintes regras:

- As vírgulas (,) são um delimitador que separa campos individuais.
- As novas linhas são um delimitador que separa registros individuais.
- Os campos são tratados como strings. Para conversões de tipo de dados, use a função intrínseca States.StringToJson em ItemSelector.
- Não são necessárias aspas duplas (” “) para encerrar strings. No entanto, strings delimitadas por aspas duplas podem conter vírgulas e novas linhas que não funcionam como delimitadores.
- Repita as aspas duplas para evitá-las.

- Se o número de campos em uma linha for menor que o número de campos no cabeçalho, o Step Functions fornecerá strings vazias para os valores que estão faltando.
- Se o número de campos em uma linha for maior que aquele no cabeçalho, o Step Functions ignorará os campos adicionais.

Example de como analisar um arquivo CSV de entrada

Digamos que você tenha fornecido um arquivo CSV chamado *myCSVInput.csv* que contém uma linha como entrada. Em seguida, você armazenou esse arquivo em um bucket do Amazon S3 que é chamado de *my-bucket*. O arquivo CSV é como a seguir.

```
abc,123,"This string contains commas, a double quotation marks (""), and a newline (
)",{""MyKey"":""MyValue""},"[1,2,3]"
```

A seguinte máquina de estado lê esse arquivo CSV e usa ItemSelector para converter os tipos de dados de alguns dos 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)"
      }
    }
  }
}
```

Quando você executa essa máquina de estado, ela produz a saída a seguir.

```
[
  {
    "MyNumbers": 123,
    "MyObject": {
      "MyKey": "MyValue"
    },
    "MyString": "This string contains commas, a double quote (\"), and a newline (\n)",
    "MyLetters": "abc",
    "MyArray": [
      1,
      2,
      3
    ]
  }
]
```

Objeto de contexto

O objeto de contexto é uma estrutura JSON interna que está disponível durante uma execução e contém informações sobre sua máquina de estado e execução. Isso permite que os fluxos de trabalho acessem as informações sobre as execuções específicas deles. Você pode acessar o objeto de contexto a partir dos seguintes campos:

- `InputPath`
- `OutputPath`
- `ItemsPath` (em estados Mapa)
- `Variable` (em estados Escolha)
- `ResultSelector`
- `Parameters`
- Operadores de comparação de variável para variável

Formato do objeto de contexto

O objeto de contexto inclui informações sobre a máquina de estado, estado, execução e tarefa. Esse objeto JSON inclui nós para cada tipo de dados, e tem o formato abaixo.

```
{
    "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 uma execução, o objeto de contexto é preenchido com dados relevantes para o campo `Parameters` de onde ele é acessado. O valor de um campo `Task` será nulo se o campo `Parameters` estiver fora de um estado de tarefa.

O valor do objeto de contexto `RedriveCount` é 0, se você ainda não tiver redriven uma execução. Além disso, o objeto de contexto `RedriveTime` só está disponível se você tiver redriven uma execução. Se você tiver redriven a Map Run, o objeto de contexto `RedriveTime` só estará disponível para fluxos de trabalho secundários do tipo Padrão. Para uma Execução de mapa redriven com fluxos de trabalho secundários do tipo Express, `RedriveTime` não está disponível.

O conteúdo de uma execução que está em andamento inclui informações específicas no formato abaixo.

```
{
    "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 obter dados de objeto de contexto relacionados a estados `Map`, consulte Dados de objeto de contexto para estados Map.

Acessar o objeto de contexto

Para acessar o objeto de contexto, primeiro especifique o nome do parâmetro anexando `.$` no final, assim como você faz ao selecionar a entrada de um estado com um caminho. Depois, para acessar os dados do objeto de contexto em vez da entrada, prefixe o caminho com `$$.`. Isso diz AWS Step Functions para usar o caminho para selecionar um nó no objeto de contexto.

Os exemplos a seguir mostram como você pode acessar objetos de contexto, como ID de execução, nome, hora de início e contagem de redrive.

- Recuperar e transmitir o ARN de execução para um serviço downstream
- Acessar a hora de início e o nome da execução em um estado Passagem
- Acessar a contagem de redrive de uma execução

Recuperar e transmitir o ARN de execução para um serviço downstream

Este exemplo de um estado Tarefa usa um caminho para recuperar e transmitir o nome do recurso da Amazon (ARN) da execução para uma mensagem do 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 obter mais informações sobre como usar o token de tarefa ao chamar um serviço integrado, consulte Aguardar um retorno de chamada com um token de tarefa.

Acessar a hora de início e o nome da execução em um estado Passagem

```
{
  "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
    }
  }
}
```

Acessar a contagem de redrive de uma execução

O exemplo a seguir de uma definição do estado Tarefa mostra como você pode acessar a contagem de redrive de uma execução.

```
{
  "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
}
```

Dados de objeto de contexto para estados Map

Há dois itens adicionais disponíveis no objeto de contexto ao processar um estado Map: `Index` e `Value`. Para cada iteração do estado `Map`, `Index` contém o número do índice do item da matriz que está sendo processado atualmente, enquanto `Value` contém o item da matriz que está sendo processado. Em um estado `Map`, o objeto de contexto inclui os seguintes dados.

```
"Map": {
   "Item": {
      "Index": Number,
      "Value": "String"
   }
}
```

Eles estão disponíveis somente em um estado `Map` e podem ser especificados no campo `ItemSelector`.

> **Note**
>
> Você deve definir parâmetros do objeto de contexto no bloco `ItemSelector` do estado `Map` principal, não nos estados incluídos na seção `ItemProcessor`.

Dada uma máquina de estado com um estado `Map` simples, podemos injetar informações do objeto de contexto como a seguir.

```
{
  "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
    }
  }
}
```

Se você executar a máquina de estado anterior com a entrada a seguir, `Index` e `Value` serão inseridos na saída.

```
[
  {
    "who": "bob"
  },
  {
    "who": "meg"
  },
  {
    "who": "joe"
  }
]
```

A saída da execução retorna os valores dos itens `Index` e `Value` de cada uma das três iterações, da seguinte forma:

```
[
  {
    "ContextIndex": 0,
    "ContextValue": {
      "who": "bob"
    }
  },
  {
    "ContextIndex": 1,
    "ContextValue": {
      "who": "meg"
    }
  },
```

```
  {

    "ContextIndex": 2,
    "ContextValue": {
      "who": "joe"
    }
  }
]
```

Simulador de fluxo de dados

Você pode projetar, implementar e depurar fluxos de trabalho no console do Step Functions. Você também pode controlar o fluxo de dados nos fluxos de trabalho com o processamento de dados de entrada e saída do JsonPath. Com o simulador de fluxo de dados, você pode simular a ordem em que os estados Estado da tarefa no fluxo de trabalho processam os dados em runtime. Usando o simulador, você pode entender como filtrar e manipular dados à medida em que passam de um estado para outro. Ele simula cada um dos campos a seguir que o Step Functions usa para processar e controlar o fluxo de dados JSON:

`InputPath`

Seleciona QUAL parte de toda a carga de entrada a ser usada como entrada de uma tarefa. Se você especificar esse campo, o Step Functions o aplicará primeiro.

`Parâmetros`

Especifica COMO deve ser a entrada antes de invocar a tarefa. Com o campo `Parameters`, você pode criar uma coleção de pares de chave/valor que são passados como entrada para a integração de serviços da AWS service (Serviço da AWS), como uma função do AWS Lambda. Esses valores podem ser estáticos ou selecionados dinamicamente da entrada de estado ou do objeto de contexto do fluxo de trabalho.

`ResultSelector`

Determina O QUE escolher na saída de uma tarefa. Com o campo `ResultSelector`, você pode criar uma coleção de pares de chave/valor que substituem o resultado de um estado e enviam essa coleção para `ResultPath`.

`ResultPath`

Determina ONDE colocar a saída de uma tarefa. Use o `ResultPath` para determinar se a saída de um estado é uma cópia de sua entrada, o resultado que produz ou uma combinação das duas opções.

`OutputPath`

Determina O QUE enviar para o próximo estado. Com o `OutputPath`, é possível filtrar informações indesejáveis e transmitir somente a parte do JSON que é importante para você.

Neste tópico

Como usar o simulador de fluxo de dados

O simulador fornece uma comparação lado a lado em tempo real dos dados antes e depois de você aplicar um campo de processamento de dados de entrada e saída. Para usar o simulador, especifique uma entrada JSON. Em seguida, avalie-a por meio de cada um dos campos de processamento de entrada e saída. O simulador valida automaticamente a entrada JSON e destaca quaisquer erros de sintaxe.

Para usar o simulador de fluxo de dados

Nas etapas a seguir, você fornecerá a entrada JSON e aplica os campos `InputPath` e `Parâmetros`. Você também pode aplicar os outros campos disponíveis e visualizar os respectivos resultados.

1. Abra o console do Step Functions.
2. No painel de navegação, clique em Simulador de fluxo de dados.
3. Na área de Entrada do estado, substitua os dados JSON de exemplo preenchidos automaticamente pelos dados JSON a seguir. Em seguida, clique em Próximo.

```
{
  "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. Em InputPath, insira **`$.data.address`** para selecionar o nó de endereço dos dados JSON de entrada.

   O campo Entrada de estado após InputPath exibe os seguintes resultados.

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

5. Clique em Próximo.
6. Aplique o campo `Parameters` para converter os dados JSON resultantes em uma string. Para converter os dados, faça o seguinte:

   - No campo Parâmetros, insira o código a seguir para criar uma string chamada `addressString`.

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

7. Visualize o resultado da aplicação do campo `Parameters` no campo Entrada filtrada após Parâmetros.

Considerações sobre o uso do simulador de fluxo de dados

Tenha em mente as restrições do simulador de fluxo de dados antes de usá-lo, pois incluem, mas não se limitam a:

- Expressões de filtro incompatíveis

  As expressões de filtro no simulador se comportam de maneira diferente do serviço do Step Functions. Isso inclui expressões que usam a seguinte sintaxe: `[?(expression)]`. Veja a seguir uma lista de expressões incompatíveis. Se usadas, essas expressões podem não retornar o resultado esperado após a avaliação.

  - `$..book[?(@.isInStock==true)]`
  - `$..book[?(@.price > 10.0)].title`
- Avaliação JsonPath incorreta para matrizes de item único

  Se você filtrar dados com uma expressão JsonPath que retorna um único item de matriz, o simulador retornará o item sem a matriz. Por exemplo, considere a matriz de dados a seguir, chamada `items`:

```
{
  "items": [
    {
      "name": "shoe",
      "color": "blue",
      "comment": "nice shoe"
    },
    {
      "name": "hat",
      "color": "red"
    },
    {
      "name": "shirt",
      "color": "yellow"
    }
  ]
}
```

  Na matriz `items`, se você inserir **`$..comment`** no campo InputPath, a seguinte saída é esperada:

```
[
  "nice shoe"
]
```

No entanto, o simulador de fluxo de dados retorna a saída a seguir:

```
"nice shoe"
```

Para a avaliação JsonPath de uma matriz que contém vários itens, o simulador retorna a saída esperada.

Gerencie implantações contínuas de com versões e aliases

Você pode usar o Step Functions para gerenciar implantações contínuas dos fluxos de trabalho por meio de versões e aliases de máquina de estado. Uma versão é um snapshot numerado e imutável de uma máquina de estado que você pode executar. Um alias é um ponteiro para até duas versões de uma máquina de estado.

Você pode manter várias versões das máquinas de estado e gerenciar sua implantação no fluxo de trabalho de produção. Com aliases, você pode rotear o tráfego entre diferentes versões dos fluxos de trabalho e implantá-los gradualmente no ambiente de produção.

Além disso, você pode iniciar execuções de máquinas de estado usando uma versão ou um alias. Se você não usa uma versão ou alias ao iniciar a execução de uma máquina de estado, o Step Functions usa a revisão mais recente da definição da máquina de estado.

> **Revisão das máquinas de estado**
>
> Uma máquina de estado pode ter uma ou mais revisões. Ao atualizar uma máquina de estado usando a ação de API UpdateStateMachine, ela cria uma nova revisão da máquina de estado. Uma revisão é um snapshot imutável, somente para leitura, da definição e configuração de uma máquina de estado. Você não pode iniciar a execução de uma máquina de estado a partir de uma revisão, e as revisões não têm um ARN. As revisões têm um `revisionId`, que é um identificador universalmente exclusivo (UUID).

Índice

- Versões de máquina de estado
- Aliases de máquina de estado
- Autorização para versões e aliases
- Como associar execuções de máquinas de estado a uma versão ou alias
- Exemplo de implantação de aliases e versões
- Realizar a implantação gradual de versões da máquina de estado

Versões de máquina de estado

Uma versão é um snapshot numerado e imutável de uma máquina de estado. Você publica versões da revisão mais recente feita nessa máquina de estado. Cada versão da função tem um nome do recurso da Amazon (ARN) exclusivo. Esse ARN é uma combinação do ARN da máquina de estado e do número da versão separados por dois pontos (:). O exemplo a seguir mostra o formato de um ARN da versão da máquina de estado.

```
arn:partition:states:region:account-id:stateMachine:myStateMachine:1
```

Para começar a usar versões de máquina de estado, você deve publicar a primeira versão. Depois de publicar uma versão, você pode invocar a ação da StartExecutionAPI com o ARN da versão. Você não pode editar uma versão, mas pode atualizar uma máquina de estado e publicar uma nova versão. Você também pode publicar várias versões da máquina de estado.

Ao publicar uma nova versão da máquina de estado, o Step Functions atribui a ela um número de versão. Os números das versões começam em 1 e aumentam monotonicamente para cada nova versão. Os números de versão não são reutilizados para uma máquina de estado. Se você excluir a versão 10 da máquina de estado e depois publicar uma nova versão, o Step Functions a publicará como versão 11.

As propriedades a seguir são as mesmas para todas as versões de uma máquina de estado:

- Todas as versões de uma máquina de estado compartilham o mesmo tipo (padrão ou expresso).
- Não é possível alterar o nome ou a data de criação de uma máquina de estado entre as versões.
- As tags se aplicam globalmente às máquinas de estado. Você pode gerenciar tags para máquinas de estado usando as ações TagResourcee UntagResourceda API.

As máquinas de estado também contêm propriedades que fazem parte de cada versão e revision, mas essas propriedades podem diferir entre duas versões ou revisões fornecidas. Essas propriedades incluem Definição de máquina de estado, Perfil do IAM, Configuração de rastreamento e Configuração de registro.

Conteúdo

Como publicar uma versão de máquina de estado (Console)

Você pode publicar até 1000 versões de uma máquina de estado. Para solicitar um aumento desse limite flexível, use a página Support Center no AWS Management Console. Você pode excluir manualmente as versões não utilizadas do console ou invocando a ação da DeleteStateMachineVersionAPI.

Para publicar uma versão de máquina de estado:

1. Abra o console do Step Functions e escolha uma máquina de estado.
2. Na página Detalhes da máquina de estado, escolha Editar.
3. Edite a definição da máquina de estado conforme necessário e escolha Salvar.
4. Escolha Publish version (Publicar versão).
5. (Opcional) No campo Descrição da caixa de diálogo exibida, digite uma breve descrição sobre a versão da máquina de estado.
6. Selecione Publish.

> **Note**
>
> Ao publicar uma nova versão da máquina de estado, o Step Functions atribui a ela um número de versão. Os números das versões começam em 1 e aumentam monotonicamente para cada nova versão. Os números de versão não são reutilizados para uma máquina de estado. Se você excluir a versão 10 da máquina de estado e depois publicar uma nova versão, o Step Functions a publicará como versão 11.

Como gerenciar versões com operações da API do Step Functions

O Step Functions fornece as seguintes operações de API para publicar e gerenciar versões de máquinas de estado:

- PublishStateMachineVersion— Publica uma versão da corrente revision de uma máquina de estado.
- UpdateStateMachine— Publica uma nova versão da máquina de estado se você atualizar uma máquina de estado e definir o `publish` parâmetro como `true` na mesma solicitação.
- CreateStateMachine— Publica a primeira revisão da máquina de estado se você definir o `publish` parâmetro como. `true`
- ListStateMachineVersions— Lista as versões do ARN da máquina de estado especificada.
- DescribeStateMachine— Retorna os detalhes da versão da máquina de estado para uma versão ARN especificada em. `stateMachineArn`
- DeleteStateMachineVersion— Exclui uma versão da máquina de estado.

Para publicar uma nova versão da revisão atual de uma máquina de estado chamada *`myStateMachine`* usando o AWS Command Line Interface, use o `publish-state-machine-version` comando:

```
aws stepfunctions publish-state-machine-version --state-machine-arn arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine
```

A resposta retorna o `stateMachineVersionArn`. Por exemplo, o comando anterior devolve uma resposta de `arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine:1`.

> **Note**
>
> Ao publicar uma nova versão da máquina de estado, o Step Functions atribui a ela um número de versão. Os números das versões começam em 1 e aumentam monotonicamente para cada nova versão. Os números de versão não são reutilizados para uma máquina de estado. Se você excluir a versão 10 da máquina de estado e depois publicar uma nova versão, o Step Functions a publicará como versão 11.

Como executar uma versão de máquina de estado do console

Para começar a usar versões de máquina de estado, você deve publicar primeiro uma versão da revision da máquina de estado atual. Para publicar uma versão, use o console Step Functions ou invoque a ação da PublishStateMachineVersionAPI. Você também pode invocar a ação da UpdateStateMachineAliasAPI com um parâmetro opcional chamado `publish` para atualizar uma máquina de estado e publicar sua versão.

Você pode iniciar as execuções de uma versão usando o console ou invocando a ação da StartExecutionAPI e fornecendo o ARN da versão. Também é possível usar um alias para iniciar a execução de uma versão. Com base na configuração de roteamento, um alias roteia o tráfego para uma versão específica.

Se você iniciar a execução de uma máquina de estado sem usar uma versão, o Step Functions usará a revisão mais recente da máquina de estado para a execução. Para obter informações sobre como o Step Functions associa uma execução a uma versão, consulte Como associar execuções de máquinas a uma versão ou alias.

Para iniciar a execução usando uma versão da máquina de estado:

1. Abra o console do Step Functions e escolha uma máquina de estado para a qual você publicou uma ou mais versões. Para saber como publicar uma versão, consulte Como publicar uma versão de máquina de estado (Console).
2. Na página Detalhes da máquina de estado, escolha a guia Versões.
3. Na seção Versões, faça o seguinte:
   a. Selecione a versão com a qual você quer iniciar a execução.
   b. Selecione Iniciar execução.
4. (Opcional) Na caixa de diálogo Iniciar execução, digite um nome para a execução.

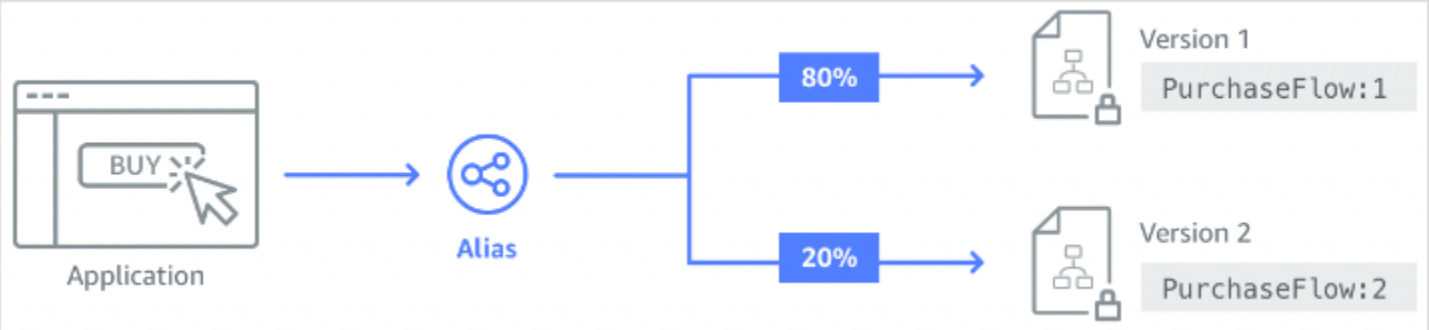
5. (Opcional) Digite os dados de entrada de execução e escolha Iniciar execução.

Aliases de máquina de estado

Um alias é um ponteiro para até duas versões da mesma máquina de estado. Você pode criar vários aliases para as máquinas de estado. Cada alias tem um nome do recurso da Amazon (ARN) exclusivo. O ARN do alias é uma combinação do ARN da máquina de estado e do nome do alias, separados por dois pontos (:). O exemplo a seguir mostra o formato de um ARN do alias de uma máquina de estado.

```
arn:partition:states:region:account-id:stateMachine:myStateMachine:aliasName
```

Você pode usar um alias para rotear o tráfego entre uma das duas versões da máquina de estado. Você também pode criar um alias que direcione para uma única versão. Os aliases só podem direcionar para versões de máquinas de estado. Você não pode usar um alias para direcionar para outro alias. O alias pode ser atualizado para direcionar para uma versão diferente da máquina de estado.



Conteúdo

- Como criar um alias de máquina de estado (Console)
- Como gerenciar aliases com operações da API do Step Functions
- Configuração de roteamento de alias
- Como executar uma máquina de estado usando um alias (Console)

Como criar um alias de máquina de estado (Console)

Você pode criar até 100 aliases para cada máquina de estado usando o console Step Functions ou invocando a ação da CreateStateMachineAliasAPI. Para solicitar um aumento desse limite flexível,

use a página Support Center no AWS Management Console. Exclua aliases não utilizados do console ou invocando a ação da DeleteStateMachineAliasAPI.

Para criar um alias de máquina de estado:

1. Abra o console do Step Functions e escolha uma máquina de estado.
2. Na página Detalhes da máquina de estado, escolha a guia Aliases.
3. Escolha Criar novo alias.
4. Na página Create alias (Criar alias), faça o seguinte:
   a. Insira um Nome do alias.
   b. (Opcional) Insira uma Description (Descrição) do alias.
5. Para configurar o roteamento no alias, consulte Configuração de roteamento de alias.
6. Escolha Criar alias.

Como gerenciar aliases com operações da API do Step Functions

O Step Functions fornece as seguintes operações de API que você pode usar para criar e gerenciar aliases de máquinas de estado ou obter informações sobre os aliases:

- CreateStateMachineAlias— Cria um alias para uma máquina de estado.
- DescribeStateMachineAlias— Retorna detalhes sobre um alias de máquina de estado.
- ListStateMachineAliases— Lista aliases para o ARN da máquina de estado especificada.
- UpdateStateMachineAlias— Atualiza a configuração de um alias de máquina de estado existente modificando seu `description` ou. `routingConfiguration`
- DeleteStateMachineAlias— Exclui uma versão da máquina de estado.

Para criar um alias chamado *`PROD`* que aponte para a versão 1 de uma máquina de estado chamada *`myStateMachine`* usando o AWS Command Line Interface, use o `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}]"
```

Configuração de roteamento de alias

Você pode usar um alias para rotear o tráfego entre duas versões de uma máquina de estado. Por exemplo, digamos que você queira lançar uma nova versão da sua máquina de estado. Você pode reduzir os riscos envolvidos na implantação da nova versão configurando o roteamento em um alias. Ao configurar o roteamento, você pode enviar a maior parte do tráfego para uma versão anterior e testada da sua máquina de estado. A nova versão pode então receber uma porcentagem menor, até que você possa confirmar que é seguro avançar com a nova versão.

Para definir a configuração de roteamento, verifique se publicou as duas versões da máquina de estado para as quais seu alias direciona. Ao iniciar uma execução a partir de um alias, o Step Functions escolhe aleatoriamente a versão da máquina de estado a ser executada a partir das versões especificadas na configuração de roteamento. Ele baseia essa escolha na porcentagem de tráfego que você atribui a cada versão na configuração de roteamento de alias.

Para definir a configuração de roteamento em um alias:

- Na página Criar alias, em Configuração de roteamento, faça o seguinte:

  a. Em Versão, escolha a primeira versão da máquina de estado para a qual o alias direciona.

  b. Marque a caixa de seleção Dividir tráfego entre duas versões.

  > **Tip**
  >
  > Para direcionar para uma única versão, desmarque a caixa de seleção Dividir tráfego entre duas versões.

  c. Em Versão, escolha a segunda versão para a qual o alias deve direcionar.

  d. Nos campos Porcentagem de tráfego, especifique a porcentagem de tráfego a ser roteada para cada versão. Por exemplo, digite **`60`** e **`40`** para rotear 60% do tráfego de execução para a primeira versão e 40% do tráfego para a segunda versão.

     As porcentagens de tráfego combinadas devem ser iguais a 100%.

Como executar uma máquina de estado usando um alias (Console)

Você pode iniciar as execuções da máquina de estado com um alias do console ou invocando a ação da StartExecutionAPI com o ARN do alias. O Step Functions então executa a versão

especificada pelo alias. Por padrão, se você não especificar uma versão ou alias ao iniciar a execução de uma máquina de estado, o Step Functions usará a revisão mais recente.

Para iniciar a execução de uma máquina de estado usando um alias:

1. Abra o console Step Functions e, em seguida, escolha uma máquina de estado existente para a qual você criou um alias. Para obter informações sobre como criar um alias, consulte Como criar um alias de máquina de estado (Console).
2. Na página Detalhes da máquina de estado, escolha a guia Aliases.
3. Na seção Aliases, faça o seguinte:
   a. Selecione o alias com o qual você deseja iniciar a execução.
   b. Selecione Iniciar execução.
4. (Opcional) Na caixa de diálogo Iniciar execução, digite um nome para a execução.
5. Se necessário, digite a entrada de execução e escolha Iniciar execução.

Autorização para versões e aliases

Para invocar ações da API do Step Functions com uma versão ou um alias, você precisa das permissões apropriadas. Para autorizar uma versão ou um alias a invocar uma ação de API, o Step Functions usa o ARN da máquina de estado em vez de usar o ARN da versão ou do alias. Você também pode definir o escopo das permissões de uma versão ou alias específico. Para obter mais informações, consulte Definir o escopo das permissões.

Você pode usar o seguinte exemplo de política do IAM de uma máquina de estado chamada *myStateMachine* para invocar a ação da API CreateStateMachineAlias para criar um 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"
    }
  ]
}
```

Ao definir permissões para permitir ou negar acesso às ações da API usando versões ou aliases de máquina de estado, considere o seguinte:

- Se você usar o parâmetro `publish` das ações da API CreateStateMachine e UpdateStateMachine para publicar uma nova versão da máquina de estado, também precisará da permissão `ALLOW` na ação da API PublishStateMachineVersion.
- A ação da API DeleteStateMachine exclui todas as versões e aliases associados a uma máquina de estado.

Neste tópico

Definir o escopo de permissões para uma versão ou alias

Você pode usar um qualificador para detalhar ainda mais a permissão de autorização necessária para uma versão ou um alias. Um qualificador se refere a um número de versão ou nome de alias. Você usa o qualificador para qualificar uma máquina de estado. O exemplo a seguir é um ARN de máquina de estado que usa um alias chamado `PROD` como o qualificador.

```
arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine:PROD
```

Para ver mais informações sobre ARNs qualificados e não qualificados, consulte Como associar execuções de máquinas a uma versão ou alias.

Você define o escopo das permissões usando a chave de contexto opcional nomeada `states:StateMachineQualifier` em uma declaração `Condition` da política do IAM. Por exemplo, a política do IAM a seguir para uma máquina de estado chamada `myStateMachine` nega acesso para invocar a ação da API DescribeStateMachine com um alias chamado `PROD` ou a versão `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"
          ]
        }
      }
    }
  ]
}
```

A lista a seguir especifica as ações da API nas quais você pode definir o escopo das permissões com a chave de contexto `StateMachineQualifier`.

- CreateStateMachineAlias
- DeleteStateMachineAlias
- DeleteStateMachineVersion
- DescribeStateMachine
- DescribeStateMachineAlias
- ListExecutions
- ListStateMachineAliases
- StartExecution
- StartSyncExecution
- UpdateStateMachineAlias

Como associar execuções de máquinas de estado a uma versão ou alias

Step Functions associa uma execução a uma versão ou alias com base no Amazon Resource Name (ARN) que você usa para invocar a ação da API. StartExecution O Step Functions executa essa ação no horário de início da execução.

É possível iniciar a execução de uma máquina de estado usando um ARN qualificado ou não qualificado.

- ARN qualificado — Refere-se ao ARN de uma máquina de estado com o sufixo de um número de versão ou nome de alias.

O seguinte exemplo de ARN qualificado se refere à versão 3 de uma máquina de estado chamada `myStateMachine`.

```
arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine:3
```

O seguinte exemplo de ARN qualificado se refere a um alias chamado `PROD` de uma máquina de estado chamada `myStateMachine`.

```
arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine:PROD
```

- ARN não qualificado — Refere-se ao ARN de uma máquina de estado sem o sufixo de um número de versão ou nome de alias.

```
arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine
```

Por exemplo, se o ARN qualificado se referir à versão 3, o Step Functions associará a execução a essa versão. Ele não associa a execução a nenhum alias que aponte para a versão 3.

Se o ARN qualificado se referir a um alias, o Step Functions associará a execução a esse alias e à versão para a qual o alias aponta. Uma execução só pode ser associada a um alias.

> **Note**
>
> Se você iniciar uma execução com um ARN não qualificado, o Step Functions não associará essa execução a uma versão, mesmo que a versão use a mesma máquina de estado revision. Por exemplo, se a versão 3 usar a revisão mais recente, mas você iniciar uma execução com um ARN não qualificado, o Step Functions não associará essa execução à versão 3.

Neste tópico

Como visualizar execuções iniciadas com uma versão ou um alias

O Step Functions fornece as seguintes maneiras pelas quais você pode visualizar as execuções iniciadas com uma versão ou um alias:

Como usar ações de API

Você pode visualizar todas as execuções associadas a uma versão ou a um alias invocando as ações DescribeExecutione ListExecutionsda API. Essas ações de API retornam o ARN da versão ou alias usado para iniciar a execução. Essas ações também retornam outros detalhes, incluindo status e ARN da execução.

Você também pode fornecer um ARN ou ARN de versão do alias da máquina de estado para listar as execuções associadas a um alias ou versão específica.

O exemplo de resposta da ação da ListExecutionsAPI a seguir mostra o ARN do alias usado para iniciar uma execução de máquina de estado chamada. *myFirstExecution*

O texto em *itálico* no trecho de código a seguir representa informações específicas do 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"
        }
    ]
}
```

Como usar o console do Step Functions

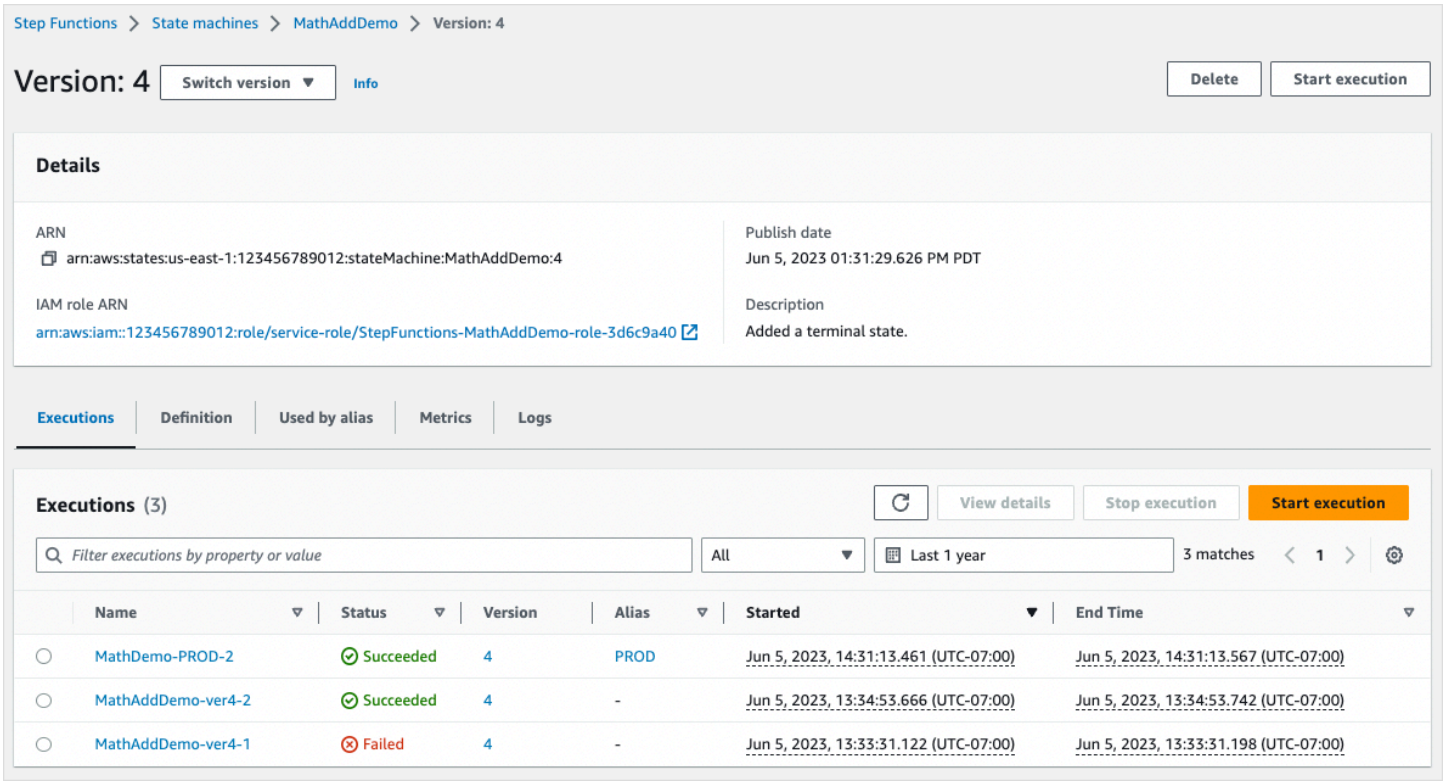
Você também pode ver as execuções iniciadas por uma versão ou um alias no console do Step Functions. O seguinte procedimento mostra como você pode visualizar as execuções iniciadas com uma versão específica:

1. Abra o console do Step Functions e, em seguida, escolha uma máquina de estado existente para a qual você publicou uma versão ou criou um alias. Este exemplo mostra como visualizar as execuções iniciadas com uma versão específica da máquina de estado.
2. Escolha a guia Versões e, em seguida, escolha uma versão na lista Versões.

> **Tip**
>
> Filtre por propriedade ou caixa de valor para pesquisar uma versão específica.

3. Na página de detalhes da versão, você pode ver uma lista de todas as execuções de máquina de estado em andamento e anteriores iniciadas com a versão selecionada.

A seguinte imagem mostra a página do console Detalhes da versão. Essa página lista as execuções iniciadas pela versão 4 de uma máquina de estado chamada *`MathAddDemo`*. Essa lista também exibe uma execução que foi iniciada por um alias chamado *`PROD`*. Esse alias roteou o tráfego de execução para a versão 4.



Uso de métricas do CloudWatch

Para cada execução de máquina de estado que você inicia com um Qualified ARN, o Step Functions emite métricas adicionais com o mesmo nome e valor das métricas emitidas atualmente. Essas

métricas adicionais contêm dimensões para cada identificador de versão e nome de alias com os quais você inicia uma execução. Com essas métricas, você pode monitorar as execuções da máquina de estado no nível da versão e determinar quando um cenário de reversão pode ser necessário. Você também pode criar CloudWatch alarmes da Amazon com base nessas métricas.

O Step Functions emite as seguintes métricas para execuções que você inicia com um alias ou uma versão:

- `ExecutionTime`
- `ExecutionsAborted`
- `ExecutionsFailed`
- `ExecutionsStarted`
- `ExecutionsSucceeded`
- `ExecutionsTimedOut`

Se você iniciou a execução com um ARN de versão, o Step Functions publica a métrica com o `StateMachineArn` e uma segunda métrica com dimensões `StateMachineArn` e `Version`.

Se você iniciou a execução com um ARN de alias, o Step Functions emite as seguintes métricas:

- Duas métricas para o ARN não qualificado e a versão.
- Uma métrica com as dimensões `StateMachineArn` e `Alias`.

Exemplo de implantação de aliases e versões

O seguinte exemplo da técnica de implantação canário mostra como você pode implantar uma nova versão da máquina de estado com a AWS Command Line Interface. Neste exemplo, o alias que você cria direciona 20% do tráfego de execução para a nova versão. Em seguida, ele direciona os 80% restantes para a versão anterior. Para implantar uma nova versão da máquina de estado e mudar o tráfego de execução com um alias, conclua as seguintes etapas:

1. Publique uma versão a partir da revisão da máquina de estado atual.

   Use o comando publish-state-machine-version na AWS CLI para publicar uma versão da revisão atual de uma máquina de estado chamada *`myStateMachine`*:

```
aws stepfunctions publish-state-machine-version --state-machine-arn
 arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine
```

A resposta retorna o `stateMachineVersionArn` da versão que você publicou. Por exemplo, `arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine:1`.

2. Crie um alias que aponte para a versão da máquina de estado.

   Use o chamado create-state-machine-alias para criar um alias chamado *PROD* que aponta para a versão 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. Verifique se as execuções iniciadas pelo alias usam a versão publicada correta.

   Inicie uma nova execução de *myStateMachine* fornecendo o ARN do alias **PROD** no comando start-execution:

   ```
   aws stepfunctions start-execution
     --state-machine-arn arn:aws:states:us-
   east-1:123456789012:stateMachineAlias:myStateMachine:PROD
     --input "{}"
   ```

   Se você fornecer o ARN da máquina de estado na solicitação StartExecution, ela usará a revision mais recente da máquina de estado em vez da versão especificada no alias para iniciar a execução.

4. Atualize a definição da máquina de estado e publique uma nova versão.

   Atualize *myStateMachine* e publique a nova versão. Para fazer isso, use o parâmetro opcional `publish` do 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
   ```

A resposta retorna o `stateMachineVersionArn` para a nova versão. Por exemplo, `arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine:2`.

5. Atualize o alias para apontar para as duas versões e defina a configuração de roteamento do alias.

   Use o comando update-state-machine-alias para atualizar a configuração de roteamento do alias PROD. Configure o alias para que 80% do tráfego de execução vá para a versão 1 e os 20% restantes para a versão 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. Substitua a versão 1 pela versão 2.

   Depois de verificar se a nova versão da máquina de estado funciona corretamente, você pode implantar a nova versão da máquina de estado. Para fazer isso, atualize o alias novamente para atribuir 100% do tráfego de execução à nova versão.

   Use o comando update-state-machine-alias para definir a configuração de roteamento do alias PROD para 100% para a versão 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 reverter a implantação da versão 2, edite a configuração de roteamento do alias para transferir 100% do tráfego para a versão recém-implantada.
>
> ```
> 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}]"
```

Você pode usar versões e aliases para realizar outros tipos de implantações. Por exemplo, você pode realizar uma implantação contínua de uma nova versão da máquina de estado. Para fazer isso, aumente gradualmente a porcentagem ponderada na configuração de roteamento do alias que aponta para a nova versão.

Você também pode usar versões e aliases para realizar uma implantação azul/verde. Para fazer isso, crie um alias chamado `green` que execute a versão 1 atual da máquina de estado. Em seguida, crie outro alias chamado `blue` que execute a nova versão, por exemplo, *2*. Para testar a nova versão, envie o tráfego de execução para o alias `blue`. Quando tiver certeza de que a nova versão funciona corretamente, atualize o alias `green` para apontar para a nova versão.

Realizar a implantação gradual de versões da máquina de estado

A implantação contínua é uma estratégia de implantação que substitui lentamente as versões anteriores de um aplicativo por novas versões. Para realizar a implantação contínua de uma versão da máquina de estado, envie gradualmente uma quantidade crescente de tráfego de execução para a nova versão. A quantidade de tráfego e a taxa de aumento são parâmetros que você configura.

Você pode realizar a implantação contínua de uma versão usando uma das opções a seguir:

- Console do Step Functions: crie um alias que aponte para duas versões da mesma máquina de estado. Para esse alias, você define a configuração de roteamento para transferir o tráfego entre as duas versões. Para obter mais informações sobre como usar o console para implementar versões, consulte Versões e Aliases.
- Scripts para a AWS CLI e o SDK: crie um script de shell usando a AWS CLI ou o AWS SDK. Para obter mais informações, consulte as seções a seguir sobre como usar a AWS CLI e o AWS SDK.
- Modelos do AWS CloudFormation: use os recursos `AWS::StepFunctions::StateMachineVersion` e `AWS::StepFunctions::StateMachineAlias` para publicar várias versões de máquinas de estado e criar um alias para apontar para uma ou duas dessas versões.

Usar a AWS CLI para implantar uma nova versão da máquina de estado

O script de exemplo nesta seção mostra como você pode usar a AWS CLI para transferir gradualmente o tráfego de uma versão anterior da máquina de estado para uma nova versão. Você pode usar esse script como exemplo ou atualizá-lo de conforme necessário.

O script mostra a implantação canário para implantar uma nova versão da máquina de estado usando um alias. As etapas a seguir descrevem as tarefas que o script executa:

1. Se o parâmetro `publish_revision` estiver definido como verdadeiro, publique a mais recente de revision como a próxima versão da máquina de estado. Esta versão se tornará a nova versão ativa se a implantação for bem-sucedida.

   Se você definir o parâmetro `publish_revision` como falso, o script implantará a última versão publicada da máquina de estado.
2. Crie um alias se ele ainda não existir. Se o alias não existir, direcione 100% do tráfego desse alias para a nova versão e saia do script.
3. Atualize a configuração de roteamento do alias para transferir uma pequena porcentagem do tráfego da versão anterior para a nova versão. Você define a porcentagem de canário no parâmetro `canary_percentage`.
4. Por padrão, monitore os alarmes configuráveis do CloudWatch a cada 60 segundos. Se algum desses alarmes for acionado, reverta a implantação imediatamente, apontando 100% do tráfego para a versão anterior.

   Após cada intervalo de tempo, em segundos, definido em `alarm_polling_interval`, continue monitorando os alarmes. Continue monitorando até que o intervalo de tempo definido em `canary_interval_seconds` tenha passado.
5. Se nenhum alarme for acionado durante `canary_interval_seconds`, transfira 100% do tráfego para a nova versão.
6. Se a nova versão for implementada com êxito, exclua todas as versões anteriores ao número especificado no parâmetro `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!
```

Usar o AWS SDK para implantar uma nova versão da máquina de estado

O script de exemplo em aws-stepfunctions-examples mostra como usar o AWS SDK para Python para transferir gradualmente o tráfego de uma versão anterior para uma nova versão da máquina de estado. Você pode usar esse script como exemplo ou atualizá-lo de conforme necessário.

O script mostra as seguintes estratégias de implantação:

- Canário: o tráfego é transferido em dois incrementos.

  No primeiro incremento, uma pequena porcentagem do tráfego, por exemplo, 10% é transferida para a nova versão. No segundo incremento, antes do fim do intervalo de tempo especificado em segundos, o tráfego restante é transferido para a nova versão. A transferência do tráfego restante para a nova versão ocorre somente se nenhum alarme do CloudWatch for acionado durante o intervalo de tempo especificado.
- Linear ou contínuo: transfere o tráfego para a nova versão em incrementos iguais com um número igual de segundos entre cada incremento.

  Por exemplo, se você especificar a porcentagem de incremento como **`20`** com um `--interval` de **`600`** segundos, a implantação aumentará o tráfego em 20% a cada 600 segundos até que a nova versão receba 100% do tráfego.

  Essa implantação reverte imediatamente a nova versão se algum alarme do CloudWatch for acionado.
- Tudo de uma vez ou azul/verde: transfere imediatamente 100% do tráfego para a nova versão. Essa implantação monitora a nova versão e a reverte automaticamente para a versão anterior se algum alarme do CloudWatch for acionado.

Usar o AWS CloudFormation para implantar uma nova versão da máquina de estado

O exemplo de modelo do CloudFormation a seguir publica duas versões da máquina de estado chamada *`MyStateMachine`*. Ele cria um alias chamado *`PROD`*, que aponta para essas duas versões e, em seguida, implanta a versão 2.

Neste exemplo, 10% do tráfego é transferido para a versão a 2 cada cinco minutos até que essa versão receba 100% do tráfego. O exemplo também mostra como você pode configurar os alarmes do CloudWatch. Se algum dos alarmes configurados entrar no estado `ALARM`, a implantação falhará e será revertida imediatamente.

```
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
```

Execuções no Step Functions

Uma execução de máquina de estado ocorre quando uma máquina de estado do AWS Step Functions é executada e realiza suas tarefas. Cada máquina de estado do Step Functions pode ter várias execuções simultâneas, que é possível iniciar pelo console do Step Functions ou usando os

AWS SDKs, as ações de API do Step Functions ou o AWS Command Line Interface (AWS CLI). Uma execução recebe entrada JSON e produz saída JSON. É possível iniciar uma execução do Step Functions das seguintes maneiras:

- Chame a ação da API StartExecution.
- Inicie uma nova execução no console do Step Functions.
- Use o Amazon EventBridge para iniciar uma execução em resposta a um evento.
- Use o Amazon EventBridge Scheduler para iniciar a execução de uma máquina de estado em um cronograma.
- Inicie uma execução com o Amazon API Gateway.
- Inicie uma execução de fluxo de trabalho aninhado usando estado da tarefa.

Para obter mais informações sobre as diferentes maneiras de trabalhar com o Step Functions, consulte Opções de desenvolvimento.

Iniciar execuções de fluxo de trabalho usando um estado Tarefa

O AWS Step Functions pode iniciar execuções de fluxo de trabalho diretamente de um estado `Task` de uma máquina de estado. Isso permite dividir seus fluxos de trabalho em máquinas de estado menores e iniciar execuções dessas outras máquinas de estado. Ao iniciar essas novas execuções de fluxo de trabalho, você poderá:

- Separar o fluxo de trabalho de nível superior dos fluxos de trabalho de nível inferior específicos à tarefa.
- Evitar elementos repetitivos chamando uma máquina de estado separada várias vezes.
- Criar uma biblioteca de fluxos de trabalho modulares reutilizáveis para um desenvolvimento mais rápido.
- Reduzir a complexidade e facilitar a edição e a solução de problemas em máquinas de estado.

O Step Functions pode iniciar essas execuções de fluxo de trabalho chamando sua própria API como um serviço integrado. Basta chamar a ação da API `StartExecution` do estado `Task` e transmitir os parâmetros necessários. É possível chamar a API do Step Functions usando qualquer um dos padrões de integração de serviço.

> **Tip**
>
> Para implementar um exemplo de fluxo de trabalho aninhado em seu Conta da AWS, consulte Módulo 13 - Fluxos de trabalho expressos aninhados.

Para iniciar uma nova execução de uma máquina de estado, use um estado `Task` semelhante ao seguinte.

```
{
   "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
}
```

Esse estado `Task` iniciará uma nova execução da máquina de estado `HelloWorld` e transmitirá o comentário JSON como entrada.

> **Note**
>
> As cotas de ação da API `StartExecution` podem limitar o número de execuções que podem ser iniciadas. Use o `Retry` no `StepFunctions.ExecutionLimitExceeded` para garantir que a execução seja iniciada. Veja o seguinte:
>
> - Cotas relacionadas ao controle de utilização das ações de API
> - Tratamento de erros no Step Functions

Associar execuções de fluxo de trabalho

Para associar uma execução de fluxo de trabalho iniciada à execução que a iniciou, transmita o ID de execução do objeto de contexto para a entrada da execução. É possível acessar o ID do objeto de contexto do estado `Task` em uma execução em andamento. Transmita o ID de execução anexando `.$` ao nome do parâmetro e fazendo referência ao ID no objeto de contexto com `$$.Execution.Id`.

```
"AWS_STEP_FUNCTIONS_STARTED_BY_EXECUTION_ID.$": "$$.Execution.Id"
```

É possível usar um parâmetro especial chamado `AWS_STEP_FUNCTIONS_STARTED_BY_EXECUTION_ID` ao iniciar uma execução. Se for incluída, essa associação fornecerá links na seção Detalhes da etapa do console do Step Functions. Quando eles forem fornecidos, será possível rastrear facilmente as execuções dos fluxos de trabalho, desde o início de execuções até as execuções de fluxo de trabalho iniciadas. Usando o exemplo anterior, associe o ID de execução à execução iniciada da máquina de estado `HelloWorld` da maneira indicada a seguir.

```
{
   "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 obter mais informações, consulte as informações a seguir.

- Como trabalhar com outros serviços
- Transmitir parâmetros para uma API de serviço
- Acessar o objeto de contexto
- AWS Step Functions

Como usar o Agendador do Amazon EventBridge com o Amazon AWS Step Functions

O Agendador do Amazon EventBridge utiliza a tecnologia sem servidor que permite criar, executar e gerenciar tarefas a partir de um serviço central gerenciado. Com o Agendador do EventBridge, você pode criar programações usando expressões cron e rate para padrões recorrentes ou configurar invocações únicas. Você pode configurar janelas de tempo flexíveis para entregas, definir limites de novas tentativas e o tempo máximo de retenção de invocações de API com falha.

Por exemplo, com o EventBridge Scheduler, você pode iniciar a execução de uma máquina de estado de acordo com uma programação quando ocorrer um evento relacionado à segurança ou para automatizar uma tarefa de processamento de dados.

Esta página explica como usar o Agendador do EventBridge para iniciar a execução de uma máquina de estado do Step Functions em uma programação.

Tópicos

Configurar o perfil de execução

Quando você cria uma programação, o Agendador do EventBridge deve ter permissão para invocar a operação da API de destino em seu nome. Você concede essas permissões ao Agendador do EventBridge usando um perfil de execução. A política de permissão que você anexa ao perfil de execução da programação define as permissões necessárias. As permissões dependem da API de destino que você deseja que o Agendador do EventBridge invoque.

Quando você usa o console do Agendador do EventBridge para criar programações, como no procedimento a seguir, o Agendador do EventBridge configura automaticamente um perfil de execução com base no destino selecionado. Se você quiser criar uma programação usando um dos SDKs do Agendador do EventBridge, a AWS CLI ou o AWS CloudFormation, você deve ter um perfil de execução que conceda as permissões que o Agendador do EventBridge precisa para invocar o destino. Para obter mais informações sobre como configurar manualmente um perfil de execução para programações, consulte Como configurar um perfil de execução no Guia do usuário do Agendador do EventBridge.

Criar uma programação

Para criar uma programação usando o console

1. Abra o console do Agendador do Amazon EventBridge em https://console.aws.amazon.com/scheduler/home.
2. Na página Programações, clique em Criar programação.
3. Na página Especificar detalhes da programação, na seção Nome e descrição da programação, faça o seguinte:

   a. Em Nome da programação, insira um nome para a programação. Por exemplo, **`MyTestSchedule`**.

   b. (Opcional) Em Descrição, insira a descrição da programação. Por exemplo, **`My first schedule`**.

   c. Em Grupo de programação, escolha um grupo de programação na lista suspensa. Se você não tiver um grupo, escolha padrão. Para criar um grupo de programação, escolha criar sua própria programação.

   Para adicionar tags a grupos de programação, você usa os grupos de programação.

4. • Escolha as opções de programação.

Ocorrência	Faça isso...	
Programação única A programação única invoca o destino somente uma vez na data e hora que você especificar.	Em Data e hora, faça o seguinte: • Insira uma data válida no formato `YYYY/MM/DD` . • Insira um carimbo de data/hora no formato de 24 horas `hh:mm`. • Em Fuso horário, escolha o fuso horário.	
Programação recorrente	a. Em Tipo de programação, siga um dos procedimentos a seguir:	

Ocorrência	Faça isso...	
A programação recorrent e invoca o destino em uma taxa especificada por você usando uma expressão cron ou rate.	• Para usar uma expressão cron para definir a programação, escolha Programaç ão baseada em cron e insira a expressão cron. • Para usar uma expressão rate para definir a programação, escolha Programaç ão baseada em rate e insira a expressão rate. Para obter mais informações sobre as expressões cron e rate, consulte Tipos de programaç ão no Agendador do EventBridge no Guia do usuário do Agendador do Amazon EventBridge. b. Em Janela de tempo flexível, escolha Desativar para desativar a opção ou escolha uma das janelas de tempo predefinidas. Por exemplo, se você escolher 15 minutos e definir uma programação	

Ocorrência	Faça isso...	
	recorrente para invocar o destino uma vez a cada hora, a programação será executada em até 15 minutos após o início de cada hora.	

5. (Opcional) Se você escolher Programação recorrente na etapa anterior, na seção Período, faça o seguinte:

   a. Em Fuso horário, escolha um fuso horário.

   b. Em Data e hora de início, insira uma data válida no formato `YYYY/MM/DD` e, em seguida, especifique um carimbo de data/hora no formato de 24 horas `hh:mm`.

   c. Para Data e hora de término, insira uma data válida no formato `YYYY/MM/DD` e, em seguida, especifique um carimbo de data/hora no formato 24 horas `hh:mm`.

6. Clique em Próximo.

7. Na página Selecionar destino, escolha a operação de API da AWS que o Agendador do EventBridge invoca:

   a. Clique em AWS Step FunctionsStartExecution.

   b. Na seção StartExecution, selecione uma máquina de estado ou clique em Criar máquina de estado.

      No momento, não é possível executar fluxos de trabalho expressos síncronos em uma programação.

   c. Insira uma carga útil JSON para a execução. Mesmo que a máquina de estado não exija nenhuma carga útil JSON, você ainda deve incluir a entrada no formato JSON, conforme mostrado no exemplo a seguir.

```
{
    "Comment": "sampleJSONData"
}
```

8. Clique em Próximo.

9. Na página Configurações, faça o seguinte:

a. Para ativar a programação, em Estado da programação, mude para Ativar programação.

b. Para configurar uma política de novas tentativas para a programação, em Política de novas tentativas e fila de mensagens não entregues (DLQ), faça o seguinte:

   - Mude para Tentar novamente.
   - Em Duração máxima do evento, insira o período máximo de horas e minutos que o Agendador do EventBridge deve manter um evento não processado.
   - O período máximo é de 24 horas.
   - Em Máximo de tentativas, insira o número máximo de vezes que o Agendador do EventBridge tentará executar a programação se o destino retornar um erro.

     O valor máximo é 185 tentativas.

   Com as políticas de novas tentativas, se a programação não conseguir invocar o destino, o Agendador do EventBridge tentará executar novamente a programação. Se configurado, você deve definir o tempo máximo de retenção e as novas tentativas da programação.

c. Escolha onde o Agendador do EventBridge armazena os eventos não entregues.

Opção Fila de mensagens não entregues (DLQ)	Faça isso...
Não armazene	Selecione Nenhum.
Armazenar o evento na mesma Conta da AWS em que você está criando a programação	a. Escolha Selecionar uma fila do Amazon SQS na minha Conta da AWS como uma DLQ. b. Escolha o nome do recurso da Amazon (ARN) da fila do Amazon SQS.

Opção Fila de mensagens não entregues (DLQ)	Faça isso...
Armazenar o evento em uma Conta da AWS diferente de onde você está criando a programação	a. Escolha Especificar uma fila do Amazon SQS em outras Contas da AWS como uma DLQ. b. Insira o nome do recurso da Amazon (ARN) da fila do Amazon SQS.

d. Para usar uma chave gerenciada pelo cliente para criptografar a entrada de destino, em Criptografia, escolha Personalizar as configurações de criptografia (avançado).

   Se você escolher essa opção, insira o ARN da chave do KMS existente ou escolha Criar AWS KMS key para navegar até o console do AWS KMS. Para obter mais informações sobre como o Agendador do EventBridge criptografa os dados em repouso, consulte Criptografia em repouso no Guia do usuário do Agendador do Amazon EventBridge.

e. Para que o Agendador do EventBridge crie um novo perfil de execução para você, escolha Criar novo perfil para esta programação. Depois, insira um nome em Nome do perfil. Se você escolher essa opção, o Agendador do EventBridge anexará as permissões necessárias para o destino de exemplo ao perfil.

10. Clique em Próximo.
11. Na página Revisar e criar programação, revise os detalhes da programação. Em cada seção, escolha Editar para voltar a essa etapa e editar seus detalhes.
12. Clique em Criar programação.

    Você pode ver a lista com as programações novas e existentes na página Programações. Na coluna Status, verifique se a nova programação está Ativada.

Para confirmar que o Agendador do EventBridge invocou a função, verifique os Amazon CloudWatch Logs da máquina de estado.

Recursos relacionados

Para obter mais informações sobre o Agendador do EventBridge, consulte:

- Guia do usuário do Agendador do EventBridge
- Referência da API do Agendador do EventBridge
- Preços do Agendador do EventBridge

Execuções de Fluxo de trabalho Padrão e Expresso no console

Ao criar uma máquina de estado, é necessário selecionar um Tipo, que pode ser Padrão ou Expresso. O Tipo padrão para máquinas de estado é Padrão. Uma máquina de estado cujo Tipo é padrão recebe a designação de fluxo de trabalho padrão, enquanto aquela cujo Tipo é expresso é chamada de fluxo de trabalho expresso.

Para fluxos de trabalho padrão e expresso, você define a máquina de estado usando a Amazon States Language. Suas execuções de máquina de estado se comportarão de forma diferente, dependendo de qual Tipo você selecionar.

> **Important**
>
> O Tipo escolhido não pode ser alterado após a criação da máquina de estado.

Para obter mais informações sobre as diferenças entre fluxos de trabalho Padrão e Expresso, consulte Comparação entre os fluxos de trabalho padrão e expresso.

O histórico das execuções do fluxo de trabalho Padrão é registrado no Step Functions, enquanto o histórico das execuções do fluxo de trabalho expressoo não é registrado no Step Functions. Para registrar o histórico da execução de um fluxo de trabalho do Express, você deve configurá-lo para enviar registros para a Amazon CloudWatch. Para ter mais informações, consulte Como registrar usando o CloudWatch Logs.

Depois que o registro em log estiver configurado em um fluxo de trabalho expressoo, você poderá visualizar execuções no console do Step Functions. A experiência do console para visualizar as execuções do fluxo de trabalho expressoo e as execuções do fluxo de trabalho Padrão é semelhante, exceto pelas diferenças e limitações a seguir.

> **Note**
>
> Como os dados de execução dos fluxos de trabalho do Express são exibidos usando o CloudWatch Logs Insights, a digitalização dos registros incorrerá em cobranças. Por padrão,

o grupo de logs lista apenas as execuções concluídas nas últimas três horas. Se você especificar um intervalo de tempo maior que inclua mais eventos de execução, os custos aumentarão. Para obter mais informações, consulte Vended Logs na guia Logs na página CloudWatch de preços e. Como registrar usando o CloudWatch Logs

Conteúdo

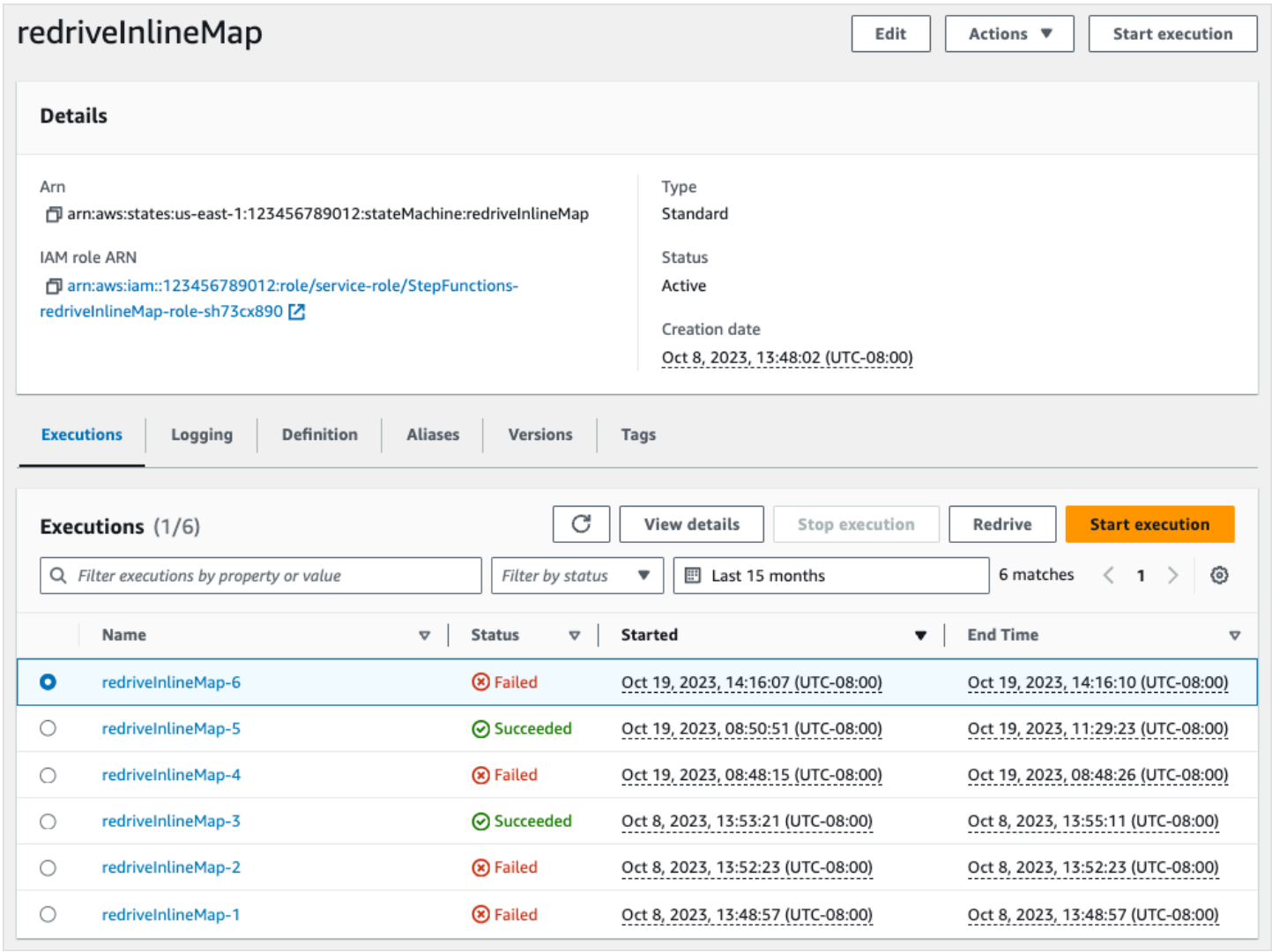
Diferenças na experiência do console

Para todos os fluxos de trabalho Padrão e Expresso, você pode visualizar detalhes, como a máquina de estado e o ARN do perfil do IAM, na página Detalhes da máquina de estado no console do Step Functions.

Na página Detalhes da máquina de estado, você também pode ver uma lista dos históricos de execução da máquina de estado na guia Execuções. Use a caixa Filtrar execuções por propriedade ou valor para pesquisar uma execução, versão ou alias específico da máquina de estado escolhida. Use o menu suspenso Todos para filtrar os históricos de execução por status. Você também pode escolher um histórico de execução e selecionar o botão Exibir detalhes para abrir a página Detalhes da execução.

Fluxos de trabalho Padrão

Os históricos de execução dos fluxos de trabalho padrão estão sempre disponíveis para execuções concluídas nos últimos 90 dias.



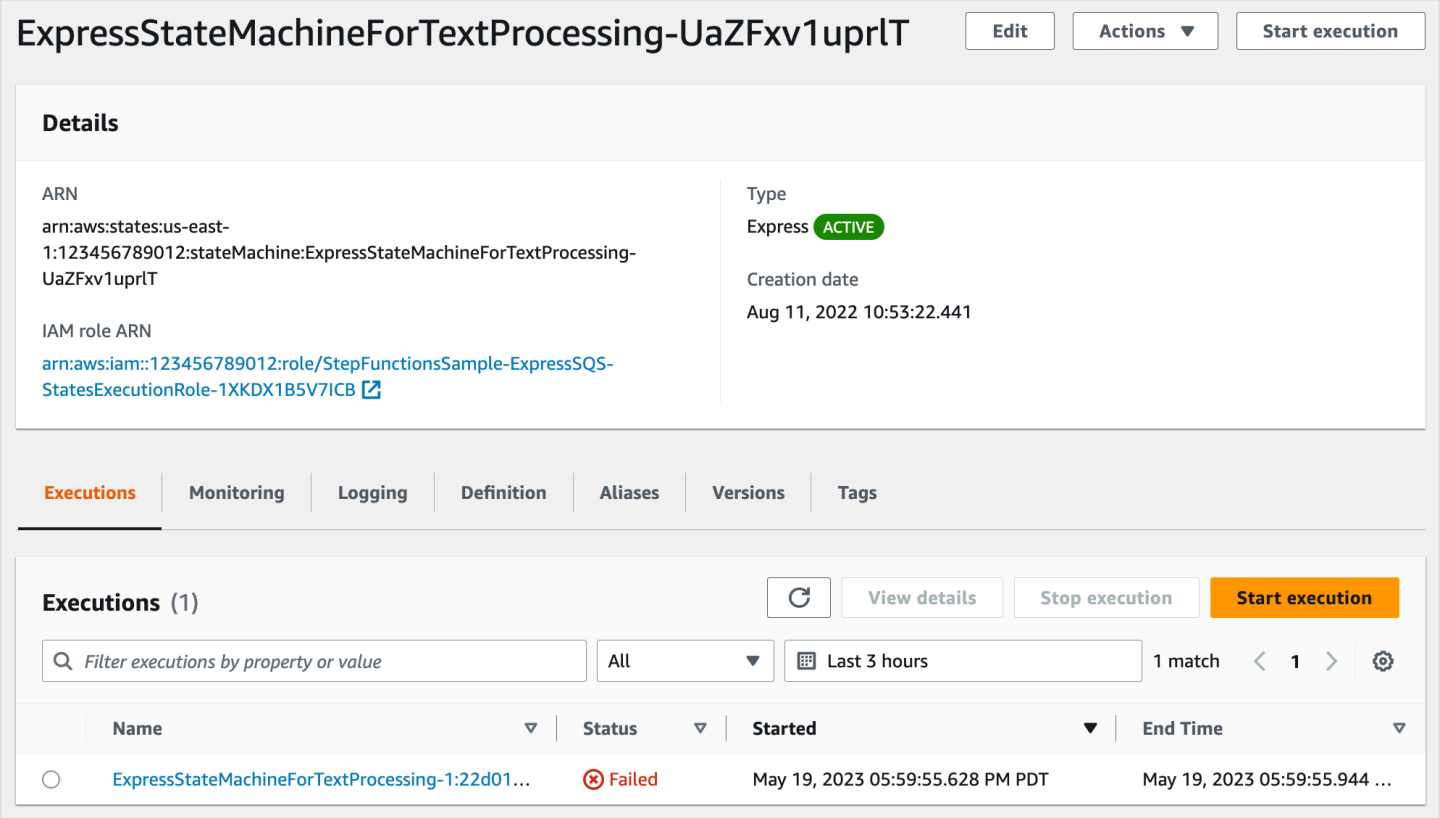
Fluxos de trabalho expressos

Para exibir o histórico de execução dos fluxos de trabalho do Express, o console Step Functions recupera os dados de registro coletados por meio de um grupo de registros de CloudWatch registros.

Você também deve ativar a nova experiência do console para visualizar as execuções do fluxo de trabalho expressoo. Para fazer isso, escolha o botão Ativar exibido dentro do banner na guia Execuções. Depois de escolher esse botão, ele não aparecerá novamente.

> **Tip**
>
> Para alternar entre ativar ou desativar a experiência do console, use o botão de alternância Ativar histórico de execução expressa.

Os históricos de execuções concluídas nas últimas três horas estão disponíveis por padrão. Você pode ajustar esse intervalo de tempo ou especificar um intervalo personalizado. Se você especificar um intervalo de tempo maior que inclua mais eventos de execução, o custo para verificar os logs aumentará. Para obter mais informações, consulte Vended Logs na guia Logs na página CloudWatch de preços e. Como registrar usando o CloudWatch Logs



Considerações e limitações da visualização de execuções do fluxo de trabalho expressoo

Ao visualizar execuções do fluxo de trabalho expressoo no console do Step Functions, tenha em mente as considerações e limitações a seguir.

- A disponibilidade dos detalhes da execução do fluxo de trabalho Express depende do Amazon CloudWatch Logs
- Os detalhes da execução parcial do fluxo de trabalho expressoo estão disponíveis se o nível de registro em log for ERROR ou FATAL
- A definição da máquina de estado de uma execução mais antiga não pode ser visualizada depois de atualizada

A disponibilidade dos detalhes da execução do fluxo de trabalho Express depende do Amazon CloudWatch Logs

> **Note**
>
> Se você não ativar a nova experiência do console para visualizar as execuções do fluxo de trabalho expressoo, os históricos de execução e os detalhes de execução correspondentes não estarão disponíveis no console do Step Functions. Para ativar a nova experiência do console, escolha o botão Ativar exibido dentro do banner na guia Execuções.

Para fluxos de trabalho do Express, seu histórico de execução e informações detalhadas de execução são coletados por meio do CloudWatch Logs Insights. Essas informações são mantidas no grupo de CloudWatch registros de registros que você especifica ao criar a máquina de estado. O histórico de execução da máquina de estado é mostrado na guia Execuções no console do Step Functions. Informações detalhadas sobre cada execução da máquina de estado são exibidas na página Detalhes da execução da execução escolhida.

> **Warning**
>
> Se você excluir os CloudWatch registros de um fluxo de trabalho do Express, eles não serão listados na guia Execuções.

Recomendamos que você use o nível de log padrão ALL para registrar em log todos os tipos de eventos de execução. Você pode atualizar o nível de log conforme necessário para as máquinas de estado existentes ao editá-las. Para obter mais informações, consulte Níveis de log e Como registrar usando o CloudWatch Logs.

Os detalhes da execução parcial do fluxo de trabalho expressoo estão disponíveis se o nível de registro em log for ERROR ou FATAL

Por padrão, o nível de registro em log para execuções de fluxo de trabalho expressoo é definido como ALL. Se você alterar o nível do log, os históricos e os detalhes de execução das execuções concluídas não serão afetados. No entanto, todas as novas execuções emitirão logs com base no nível de log atualizado. Para obter mais informações, consulte Níveis de log e Como registrar usando o CloudWatch Logs.

Por exemplo, se você alterar o nível do log de ALL para ERROR ou FATAL, a guia Executions no console do Step Functions listará somente as execuções com falha. Na guia Visualização de eventos, o console mostra somente os detalhes do evento para as etapas da máquina de estado que falharam.

Recomendamos que você use o nível de log padrão ALL para registrar em log todos os tipos de eventos de execução. Você pode atualizar o nível de log conforme necessário para as máquinas de estado existentes ao editar a máquina de estado.

A definição da máquina de estado de uma execução mais antiga não pode ser visualizada depois de atualizada

As definições da máquina de estado para execuções passadas não são armazenadas para fluxos de trabalho expressoo. Se você alterar a definição da máquina de estado, só poderá visualizar a definição da máquina de estado para execuções usando a definição mais atual.

Por exemplo, se você remover uma ou mais etapas da definição da máquina de estado, o Step Functions detectará uma incompatibilidade entre a definição e os eventos de execução anteriores. Como as definições anteriores não são armazenadas para fluxos de trabalho expressoo, o Step Functions não pode exibir a definição da máquina de estado para execuções realizadas em uma versão anterior da definição da máquina de estado. Como resultado, as guias de Entrada e saída da execução, Definição, Exibição em gráfico e Visualização em tabela não estão disponíveis para execuções realizadas em versões anteriores de uma definição de máquina de estado.

Visualizar e depurar execuções no console do Step Functions

A página Detalhes da execução no console do Step Functions apresenta informações sobre execuções anteriores e em andamento de máquinas de estado para fluxos de trabalho padrão e expresso. Essas informações são mostradas em um formato de painel. Por exemplo, você pode encontrar a definição da Amazon States Language da máquina de estado, o status de execução, ARN e número total de transições de estado. Você também pode visualizar os detalhes da execução de qualquer estado individual na máquina de estado.

Conteúdo

- Detalhes da etapa
- Eventos
- Tutorial: como examinar execuções de máquinas de estado usando o console do Step Functions
  - Etapa 1: criar e testar as funções do Lambda necessárias
  - Etapa 2: criar e executar a máquina de estado
  - Etapa 3: visualizar os detalhes da execução da máquina de estado
  - Etapa 4: explorar os diferentes Modos de visualização

Página de Detalhes da execução — Visão geral da interface

Você pode encontrar os detalhes de todas as execuções de máquina de estado, em andamento e anteriores, tanto para fluxos de trabalho padrão quanto expresso, na página de Detalhes da execução. Se você especificou um ID de execução ao iniciar uma execução, essa página será intitulada com esse ID. Caso contrário, ela será intitulada com um ID de execução exclusivo que o Step Functions gera automaticamente para você.

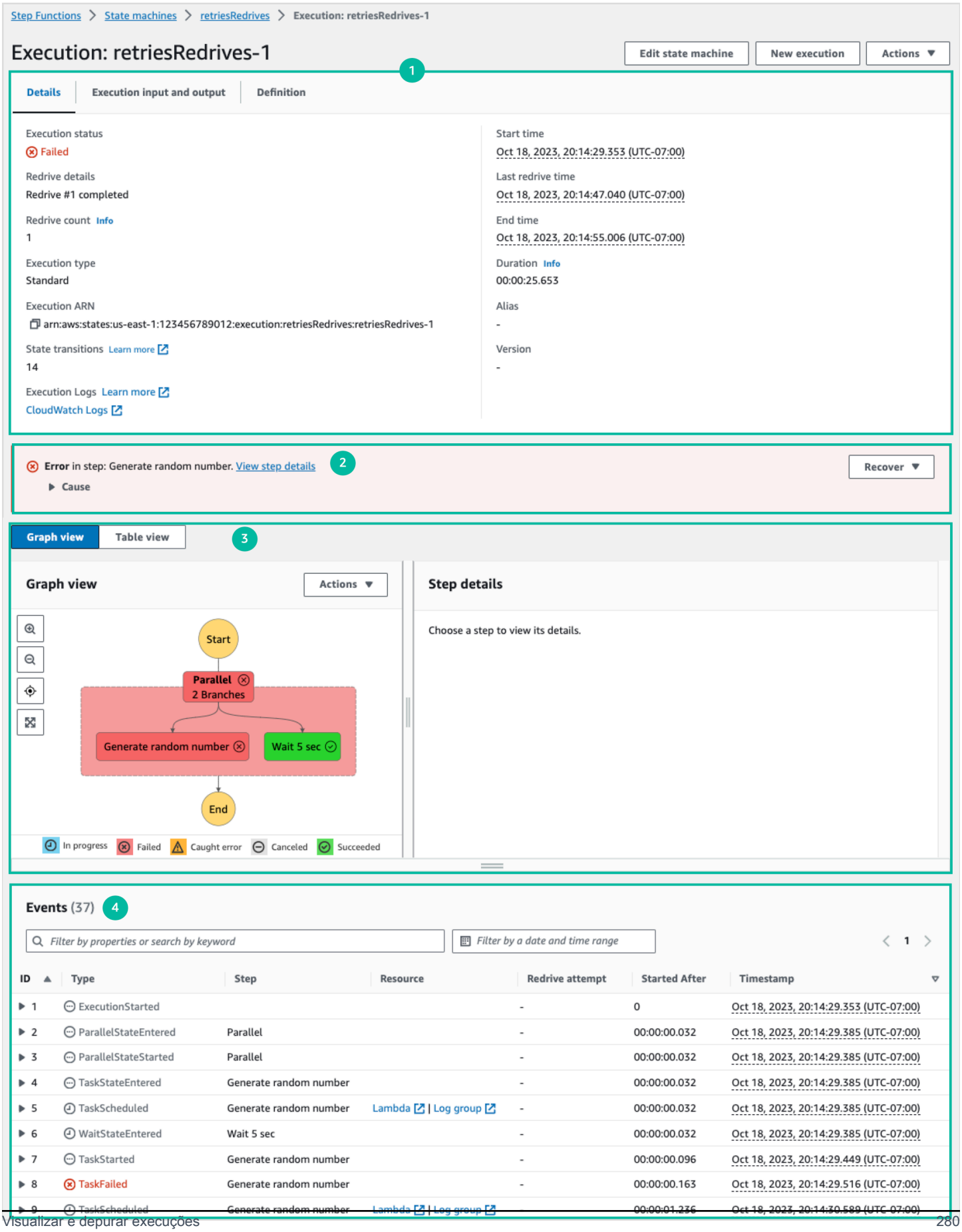
Além das métricas de execução, a página de Detalhes da execução fornece as seguintes opções para gerenciar a máquina de estado e sua execução:

Button	Selecione esse botão para:
Editar uma máquina de estado	Editar a definição da Amazon States Language da máquina de estado.
Nova execução	Iniciar uma nova execução da máquina de estado.
Ações	Fornece as seguintes opções de escolha: • Interromper execução — pare uma execução em andamento. Essa opção não está disponível para execuções concluídas. • Redrive — Redrive as execuções de Fluxos de trabalho padrão que não tenham sido concluídas com êxito nos últimos 14 dias. Isso inclui execuções com falha, abortadas

Button	Selecione esse botão para:
	ou expiradas. Para ter mais informações, consulte Redriving execuções. • Exportar — exporte os detalhes da execução no formato JSON para compartilhá-los com outras pessoas ou realizar análises off-line. • Enviar feedback — compartilhe comentários sobre a interface.

> **Visualizar execuções iniciadas com uma versão ou alias**
>
> Você também pode ver as execuções iniciadas com uma versão ou um alias no console do Step Functions. Para obter mais informações, consulte Listar execuções para versões e aliases.

A página do console de Detalhes da execução contém as seguintes seções:


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. Resumo da execução
2. Mensagem de erro
3. Modo de visualização
4. Detalhes da etapa
5. Eventos

Resumo da execução

A seção de Resumo da execução aparece na parte superior da página de Detalhes da execução. Essa seção fornece a você uma visão geral dos detalhes da execução do fluxo de trabalho. Essas informações são divididas nas três guias a seguir:

Detalhes

Mostra informações, como o status da execução, ARN e carimbos de data e hora do início e término da execução. Você também pode visualizar a contagem total de Transições de estado que ocorreram durante a execução da máquina de estado. Você também pode visualizar os links para o mapa de rastreamento X-Ray e os Amazon CloudWatch Execution Logs se tiver ativado o rastreamento ou os registros para sua máquina de estado.

Se a execução da sua máquina de estado tiver sido iniciada por outra máquina de estado, você conseguirá ver o link da máquina de estado principal nessa guia.

Se a execução da sua máquina de estado era redriven, essa guia exibirá informações relacionadas ao redrive, por exemplo, contagem de Redrive.

Entrada e saída de execução

Mostra a entrada e a saída da execução da máquina de estado side-by-side.

Definição

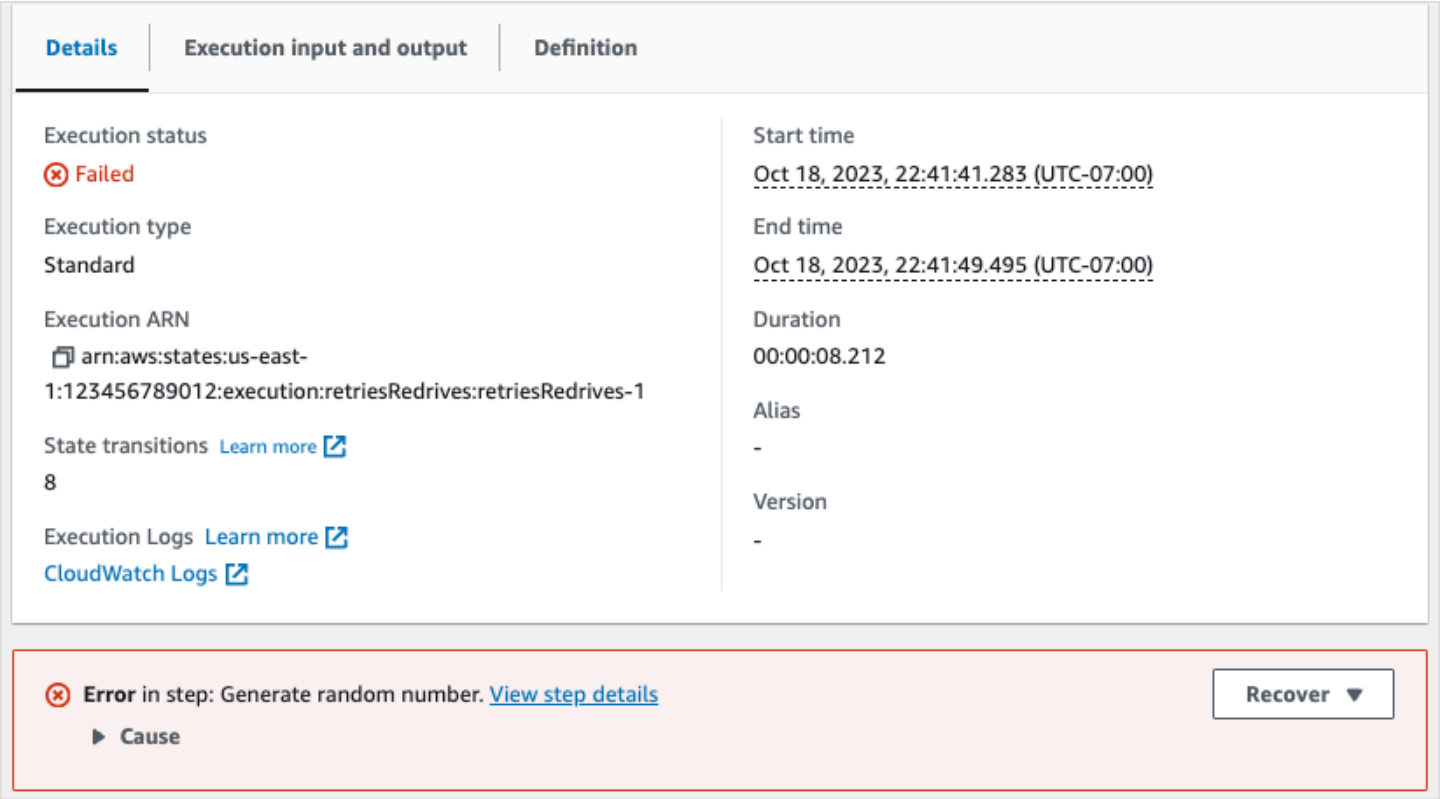
Mostra a definição da Amazon States Language da máquina de estado.

Mensagem de erro

Se a execução da máquina de estado falhar, a página Detalhes da execução exibirá uma mensagem de erro. Escolha Causa ou Exibir detalhes da etapa na mensagem de erro para ver o motivo da falha na execução ou a etapa que causou o erro.

Se você escolher Exibir detalhes da etapa, o Step Functions destacará a etapa que causou o erro nas guias Detalhes da etapa, Exibição em gráfico e Exibição em tabela. Se a etapa for um estado Tarefa, Mapa ou Paralelo para o qual você definiu novas tentativas, o painel de Detalhes da etapa exibirá a guia Repetir para a etapa. Além disso, se você tiver redriven a execução, será possível ver as novas tentativas e os detalhes do redrive da execução na guia Tentativas e redrives do painel de Detalhes da etapa.

No botão suspenso Recuperar dessa mensagem de erro, você pode redrive as execuções malsucedidas ou iniciar uma nova. Para ter mais informações, consulte Redriving execuções.



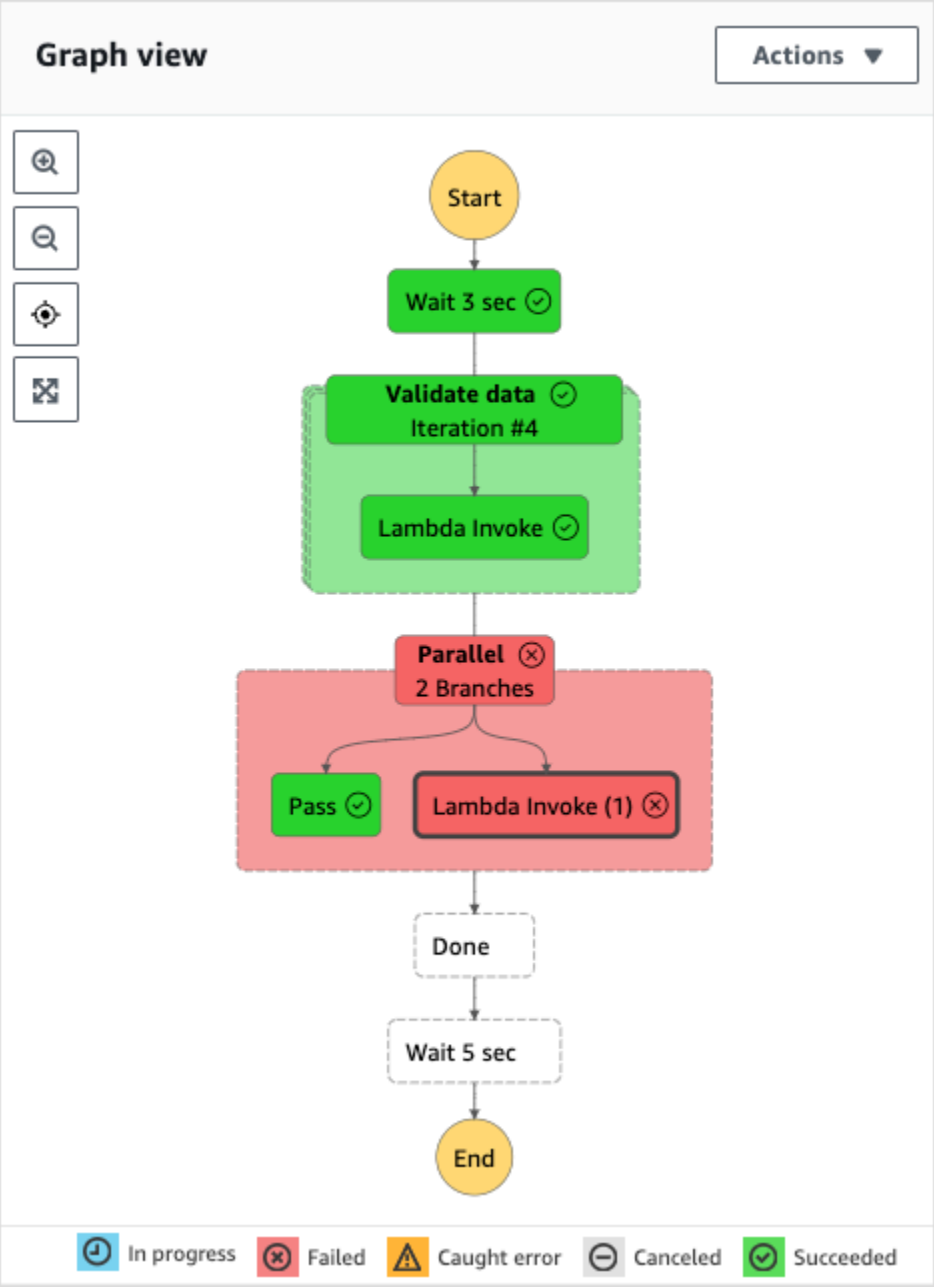
Modo de visualização

A seção de Modo de visualização contém duas visualizações diferentes para a máquina de estado. Você pode optar por visualizar uma representação gráfica do fluxo de trabalho, uma tabela descrevendo os estados em um fluxo de trabalho ou uma lista dos eventos associados à execução da máquina de estados:

> **Note**
>
> Escolha uma guia para ver o conteúdo.

Graph view

O modo de Modo de visualização exibe uma representação gráfica do fluxo de trabalho. Há uma legenda na parte inferior que indica o status de execução da máquina de estado. Ela também contém botões que permitem ampliar, reduzir o zoom, centralizar o fluxo de trabalho completo ou visualizar o fluxo de trabalho no modo de tela cheia.



Nessa exibição, você pode escolher qualquer etapa em um fluxo de trabalho para ver detalhes sobre sua execução no componente Detalhes da etapa. Quando você escolhe uma etapa na Exibição em gráfico, a Exibição em tabela também mostra essa etapa. Esse comportamento se

repete na ordem inversa. Se você escolher uma etapa na Exibição em tabela, a Exibição em gráfico mostrará a mesma etapa.

Se a máquina de estado contiver um estado `Map`, `Parallel` ou ambos, você poderá visualizar seus nomes no fluxo de trabalho da Exibição em gráfico. Além disso, para o estado `Map`, a Exibição em gráfico permite que você se mova entre diferentes iterações dos dados de execução do estado Mapa. Por exemplo, se o estado Mapa tiver cinco iterações e você quiser visualizar os dados de execução da terceira e quarta iterações, faça o seguinte:

1. Escolha o estado Mapa cujos dados de iteração você deseja visualizar.
2. Em Visualizador de iteração do mapa, escolha 2 na lista suspensa para a terceira iteração. Isso ocorre porque as iterações são contadas a partir de zero. Da mesma forma, escolha 3 na lista suspensa se quiser ver a quarta iteração do estado Mapa.

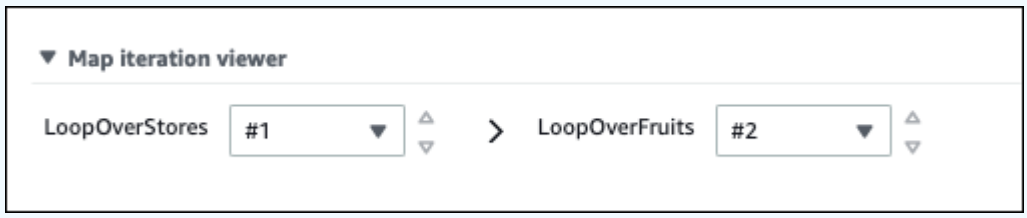
   Como alternativa, use os controles

   ▵

   e

   ▿

   para se mover entre diferentes iterações do estado Mapa.

   > **Note**
   >
   > Se sua máquina de estado contiver estados `Map` aninhados, as listas suspensas das iterações do estado `Map` principal e secundário serão exibidas como mostrado no exemplo a seguir:
   >
   > 
   >

3. (Opcional) Se uma ou mais das iterações de estado Mapa falharem quando executadas ou a execução for interrompida, você poderá visualizar seus dados escolhendo esses números de iteração em Com falha ou Abortada, na lista suspensa.

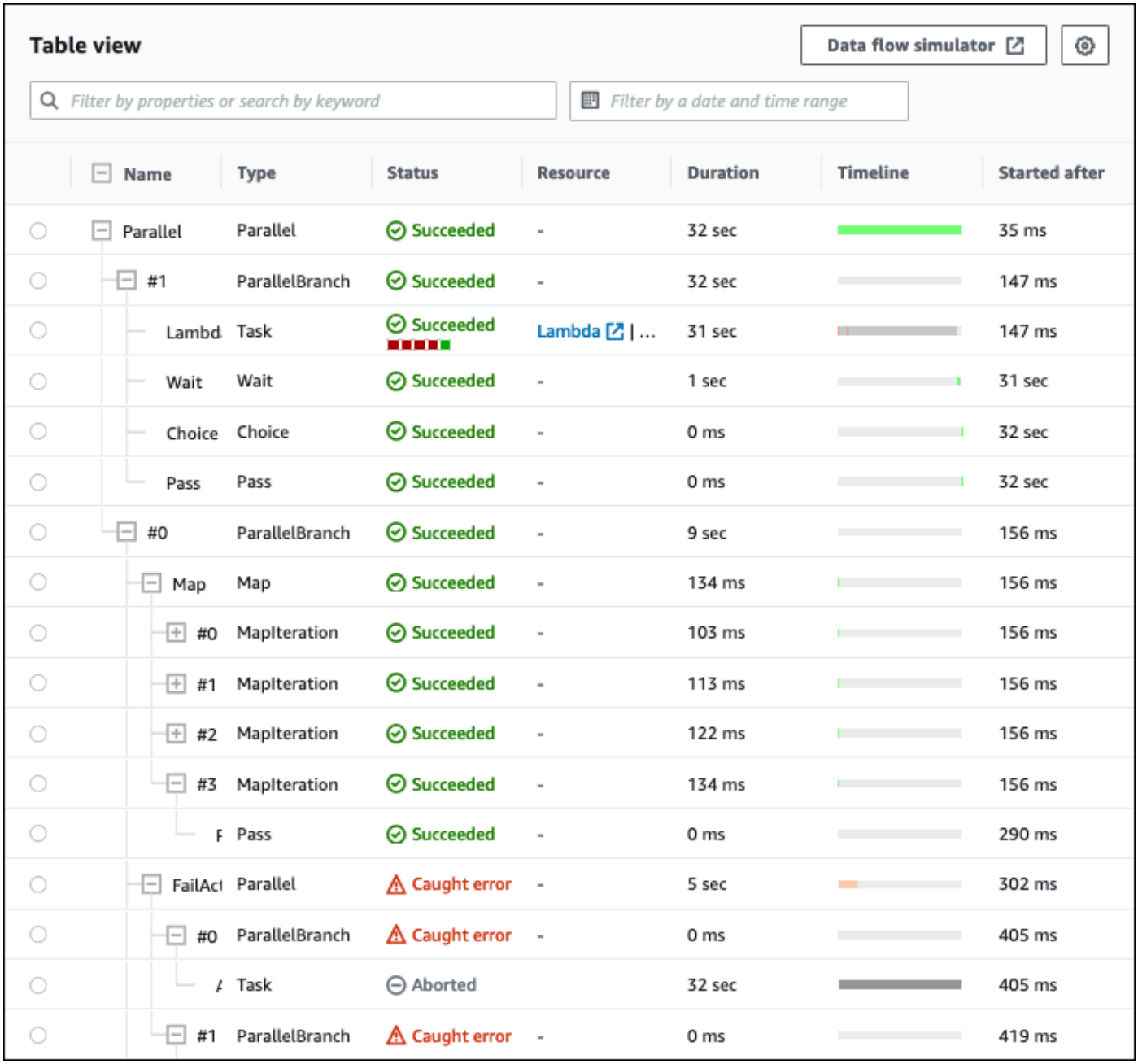
Por fim, você pode usar os botões Exportar e Layout para exportar o gráfico do fluxo de trabalho como uma imagem SVG ou PNG. Você também pode alternar entre as visualizações horizontal e vertical do seu fluxo de trabalho.

Table view

O modo de Exibição em tabela exibe uma representação tabular dos estados de um fluxo de trabalho. Nesse modo de exibição, você pode ver os detalhes de cada estado que foi executado em seu fluxo de trabalho, incluindo seu nome, o nome de qualquer recurso usado (como uma AWS Lambda função) e se o estado foi executado com êxito.

Nessa exibição, você pode escolher qualquer estado em um fluxo de trabalho para ver detalhes sobre sua execução no componente Detalhes da etapa. Quando você escolhe uma etapa na Exibição em tabela, a Exibição em gráfico também mostra essa etapa. Esse comportamento se repete na ordem inversa. Se você escolher uma etapa na Exibição em gráfico, a Exibição em tabela mostrará a mesma etapa.

Você também pode limitar a quantidade de dados exibidos no modo de Exibição em tabela aplicando filtros à exibição. Você pode criar um filtro para uma propriedade específica, como Status ou Tentativa de Redrive. Para ter mais informações, consulte Tutorial: como examinar execuções de máquinas de estado usando o console do Step Functions.

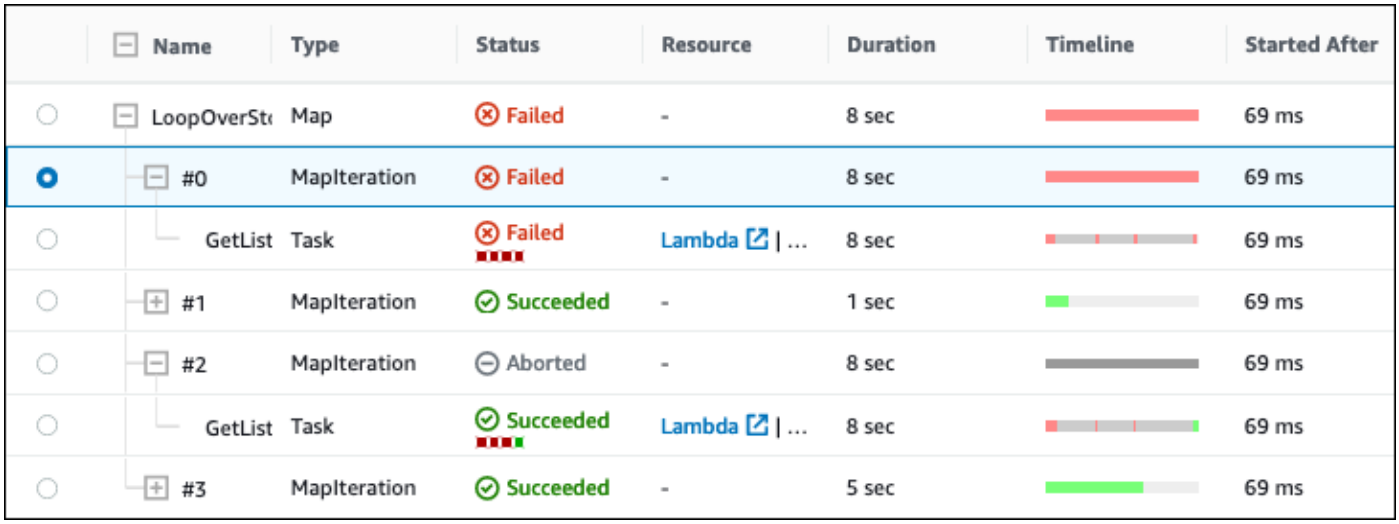


Por padrão, esse modo exibe as colunas Nome, Tipo, Status, Recurso e Iniciado depois. Você pode configurar as colunas que deseja visualizar usando a caixa de diálogo Preferências. As seleções feitas nessa caixa de diálogo persistem nas futuras execuções de máquinas de estado, até serem novamente alteradas.

Se você adicionar a coluna Cronograma, a duração da execução de cada estado será mostrada em relação ao tempo da execução inteira. Será exibido como um cronograma linear e codificado por cor. Isso pode ajudar você a identificar quaisquer problemas relacionados ao desempenho da execução de um estado específico. Os segmentos codificados por cores para cada estado no

cronograma ajudam a identificar o status da execução do estado, como em andamento, com falha ou abortada.

Por exemplo, se você definiu novas tentativas de execução para um estado na máquina de estado, elas serão mostradas no cronograma. Os segmentos vermelhos representam as tentativas `Retry` com falha, enquanto os segmentos cinza-claro representam o `BackoffRate` entre cada tentativa `Retry`.



Se a máquina de estado contiver um estado `Map`, `Parallel` ou ambos, você poderá visualizar seus nomes no fluxo de trabalho da Exibição em tabela. Para estados `Map` e `Parallel`, o modo de Exibição em tabela exibe os dados de execução de suas iterações e ramificações paralelas como nós dentro de uma visualização em árvore. Você pode escolher cada nó nesses estados para ver os detalhes individuais, na seção de Detalhes da etapa. Por exemplo, você pode revisar os dados de uma iteração específica do estado Mapa que causou a falha do estado. Basta expandir o nó do estado Mapa e, em seguida, visualizar o status de cada iteração na coluna Status.

Detalhes da etapa

A seção Detalhes da etapa é aberta à direita quando você escolhe um estado na Exibição em gráfico ou Exibição em tabela. Essa seção contém as seguintes guias, que fornecem informações detalhadas sobre o estado selecionado:

Entrada

Mostra os detalhes de entrada do estado selecionado. Se houver um erro na entrada, ele será indicado com um

no cabeçalho da guia. Além disso, também é possível ver o motivo do erro nessa guia.

Você também pode escolher o botão de alternância Advanced view para ver o caminho de transferência dos dados de entrada à medida que os dados passam pelo estado selecionado. Isso permite identificar como sua entrada foi processada quando um ou mais campos, como `InputPath`, `Parameters`, `ResultSelector`, `OutputPath` e `ResultPath`, foram aplicados aos dados.

Saída

Mostra a saída do estado selecionado. Se houver um erro na saída, ele será indicado com um

no cabeçalho da guia. Além disso, também é possível ver o motivo do erro nessa guia.

Você também pode escolher o botão de alternância Advanced view para ver o caminho de transferência dos dados de saída à medida que os dados passam pelo estado selecionado. Isso permite identificar como sua entrada foi processada quando um ou mais campos, como `InputPath`, `Parameters`, `ResultSelector`, `OutputPath` e `ResultPath`, foram aplicados aos dados.

Detalhes

Mostra informações, como o tipo de estado, status e duração da execução.

Para `Task` estados que usam um recurso, como AWS Lambda, essa guia fornece links para a página de definição do recurso e a página de CloudWatch registros da Amazon para a invocação do recurso. Também mostra valores, se especificados, para os campos `TimeoutSeconds` e `HeartbeatSeconds` do estado `Task`.

Para estados `Map`, essa guia mostra informações sobre a contagem total das iterações de um estado `Map`. As iterações são categorizadas como falhadas, abortadas, bem-sucedidas ou. InProgress

Definição

Mostra a definição da Amazon States Language correspondente ao estado selecionado.

Tentar novamente

> **Note**
>
> Essa guia aparece somente se você tiver definido um campo `Retry` no estado `Task` ou `Parallel` da máquina de estado.

Mostra as tentativas de repetição iniciais e subsequentes de um estado selecionado em uma tentativa de execução original. Para a tentativa inicial e todas as tentativas subsequentes com falha, escolha a opção

ao lado de Tipo para ver o Motivo da falha que aparece em uma caixa suspensa. Se a nova tentativa for bem-sucedida, você poderá ver a Saída que aparece em uma caixa suspensa.

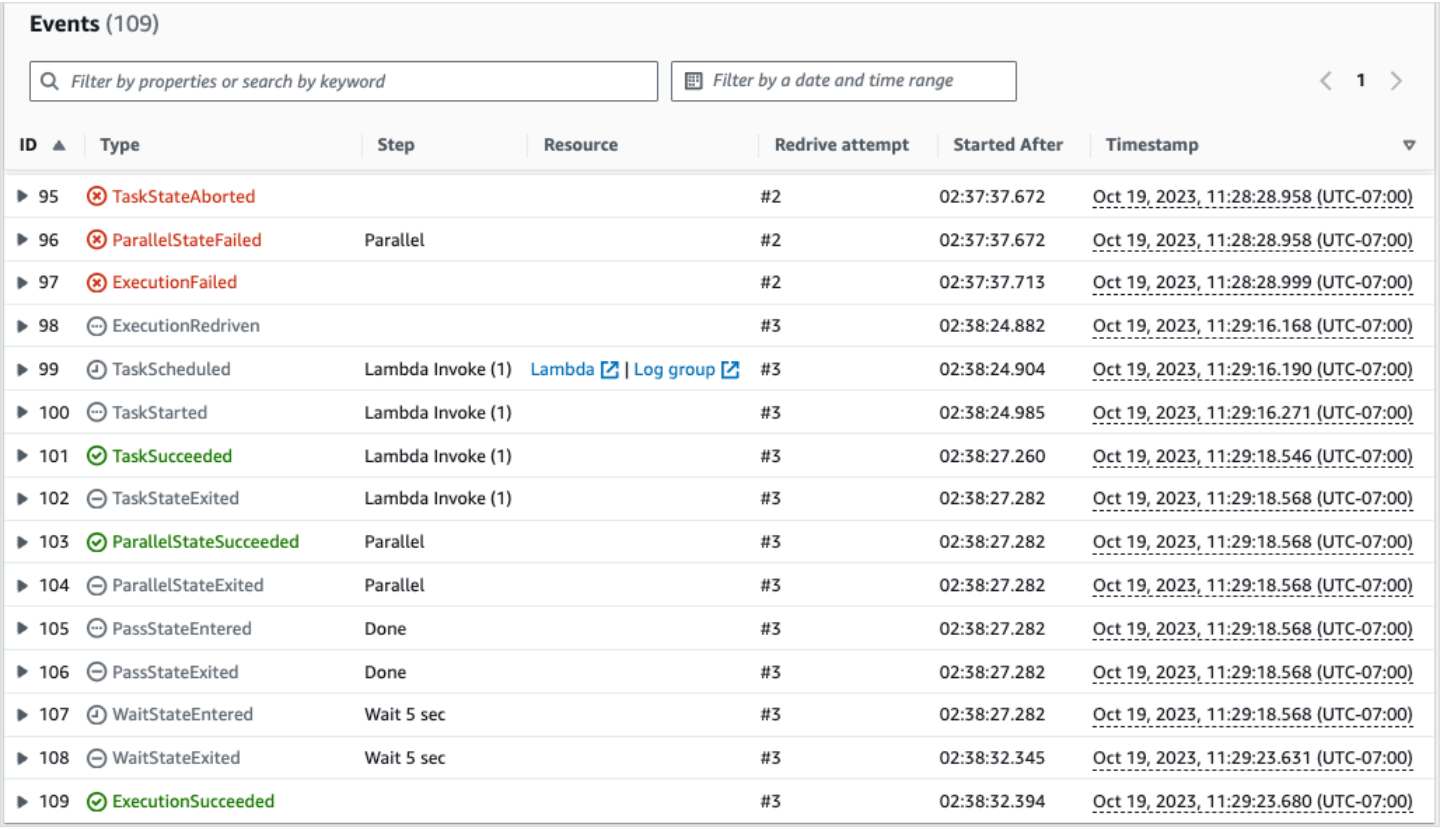
Se você tiver redriven a execução, o cabeçalho dessa guia exibirá o nome Tentativas e redrives e os detalhes da tentativa de repetição de cada redrive.

Eventos

Mostra uma lista filtrada dos eventos associados ao estado selecionado em uma execução. As informações que você vê nessa guia são um subconjunto do histórico completo de eventos de execução como visto na tabela Eventos.

Eventos

A tabela Eventos exibe o histórico completo da execução selecionada como uma lista de eventos abrangendo várias páginas. Cada página contém até 25 eventos. Essa seção também exibe a contagem total de eventos, o que pode ajudar você a determinar se a contagem máxima de 25.000 eventos do histórico foi excedida.



Events (109)

Filter by properties or search by keyword | Filter by a date and time range | 1

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)

Por padrão, os resultados na tabela Eventos são exibidos em ordem crescente, com base no Timestamp dos eventos. Você pode alterar a classificação do histórico de eventos de execução para ordem decrescente clicando no cabeçalho da coluna Timestamp.

Na tabela Eventos, cada evento é codificado por cores para indicar o status de execução. Por exemplo, eventos que falharam aparecem em vermelho. Para ver detalhes adicionais sobre um evento, escolha

ao lado do ID do evento. Depois de aberto, os detalhes do evento mostram a entrada, saída e invocação do recurso para o evento.

Além disso, na tabela Eventos, você pode aplicar filtros para limitar os resultados do histórico de eventos de execução exibidos. Você pode escolher propriedades como ID ou Tentativa de Redrive. Para ter mais informações, consulte Tutorial: como examinar execuções de máquinas de estado usando o console do Step Functions.

Tutorial: como examinar execuções de máquinas de estado usando o console do Step Functions

Neste tutorial, você aprenderá como inspecionar as informações de execução exibidas na página Detalhes da execução e ver o motivo de uma falha na execução. Em seguida, você vai aprender como acessar diferentes iterações de uma execução de estado `Map`. Por fim, você aprenderá como configurar as colunas na Visualização em tabela e aplicar filtros adequados para visualizar somente as informações de seu interesse.

Neste tutorial, você cria uma máquina de estado do tipo Padrão, que obtém o preço de um conjunto de frutas. Para fazer isso, a máquina estadual usa três AWS Lambda funções que retornam uma lista aleatória de quatro frutas, o preço de cada fruta e o custo médio das frutas. As funções do Lambda foram projetadas para gerar um erro se o preço das frutas for menor ou igual a um valor limite.

> **Note**
>
> Embora o procedimento a seguir contenha instruções sobre como examinar os detalhes da execução de um fluxo de trabalho Padrão, você também pode examinar os detalhes das execuções do Fluxo de trabalho expressoo. Para obter informações sobre as diferenças entre os detalhes de execução dos tipos de fluxo de trabalho Padrão e Expresso, consulte Execuções de Fluxo de trabalho Padrão e Expresso no console.

Conteúdo

Etapa 1: criar e testar as funções do Lambda necessárias

1. Abra o console do Lambda e execute as etapas de 1 a 4 na seção Etapa 1: criar uma função do Lambda. Certifique-se de nomear a função do Lambda **`GetListOfFruits`**.
2. Depois de criar a função do Lambda, copie o nome do recurso da Amazon (ARN) da função exibido no canto superior direito da página. Para copiar o ARN, clique no  .

Veja a seguir um exemplo de ARN, em que *`function-name`* é o nome da função do Lambda (nesse caso, `GetListOfFruits`):

```
arn:aws:lambda:us-east-1:123456789012:function:function-name
```

3. Copie o código a seguir para a função Lambda na área de código-fonte da 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. Escolha Implantar e, em seguida, escolha Testar, para implantar as alterações e ver a saída da função do Lambda.

5. Crie duas funções do Lambda adicionais, nomeadas **`GetFruitPrice`** e **`CalculateAverage`** respectivamente, com as seguintes etapas:

   a. Copie o código a seguir na área de origem do código da função 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 o código a seguir na área de origem do código da função 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. Certifique-se de copiar os ARNs dessas duas funções do Lambda e, em seguida, Implante-os e Teste-os.

Etapa 2: criar e executar a máquina de estado

Use o console do Step Functions para criar uma máquina de estado que invoca as funções do Lambda que você criou na Etapa 1. Nessa máquina de estado, três estados `Map` são definidos. Cada um desses estados `Map` contém um estado `Task` que invoca uma das funções do Lambda. Além disso, um campo `Retry` é definido em cada estado `Task` com várias tentativas definidas para cada estado. Se um estado `Task` encontrar um erro de runtime, ele será executado novamente até o número de novas tentativas definido para esse `Task`.

1. Abra o console do Step Functions e escolha Gravar o fluxo de trabalho no código.

> **Important**
>
> Certifique-se de que sua máquina de estado esteja na mesma AWS conta e região da função Lambda que você criou anteriormente.

2. Para Tipo, mantenha a seleção padrão de Padrão.
3. Copie a seguinte definição de Amazon States Language e cole-a em Definição. Substitua os ARNs mostrados pelos das funções do Lambda que você criou 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. Insira um nome para a máquina de estado. Mantenha as seleções padrão para outras opções desta página e escolha Criar máquina de estado.
5. Abra a página intitulada com o nome da máquina de estado. Execute as etapas de 1 a 4 na seção Etapa 4: Executar a máquina de estado, mas use os seguintes dados como entrada de execução:

```
{
    "stores": [
      "Store A",
      "Store B",
      "Store C",
      "Store D"
    ]
}
```

Etapa 3: visualizar os detalhes da execução da máquina de estado

Na página intitulada com o ID de execução, você pode revisar os resultados da execução e depurar quaisquer erros.

1. (Opcional) Escolha entre as guias exibidas na página Detalhes da execução para ver as informações presentes em cada uma delas. Por exemplo, para visualizar a entrada da máquina de estado e a saída de execução, escolha Entrada e saída da execução na seção Resumo de execução.

2. Se a execução da máquina de estado falhar, escolha Causa ou Mostrar detalhes da etapa na mensagem de erro. Detalhes sobre o erro são exibidos na seção Detalhes da etapa. Observe que a etapa que causou o erro, que é um `Task` estado chamado GetListofFruits, está destacada na visualização em gráfico e na exibição em tabela.

> **Note**
>
> Como a GetListofFruitsetapa está definida dentro de um `Map` estado e a etapa não foi executada com êxito, a etapa Status do `Map` estado é exibida como Falha.

Etapa 4: explorar os diferentes Modos de visualização

Você pode escolher um modo preferido para visualizar o fluxo de trabalho da máquina de estado ou o histórico de eventos de execução. Algumas das tarefas que você pode executar nesses Modos de visualização são as seguintes:

Exibição em gráfico — Alterne entre diferentes iterações de estado **`Map`**

Se o estado Mapa tiver cinco iterações, e você quiser visualizar os dados de execução da terceira e quarta iterações, faça o seguinte:

1. Escolha o estado `Map` cujos dados de iteração você deseja visualizar.
2. No Visualizador de iteração do mapa, escolha a iteração que você deseja visualizar. As iterações são contadas a partir de zero. Para escolher a terceira iteração entre cinco, escolha nº 2 na lista suspensa ao lado do nome do estado Mapa.

> **Note**
>
> Se a máquina de estado contiver estados `Map` aninhados, o Step Functions exibirá as iterações de estado `Map` pai e filho como duas listas suspensas separadas:
>
> 
>

3. (Opcional) Se uma ou mais de suas iterações de estado `Map` falharam na execução ou foram interrompidas em um estado interrompido, você poderá ver detalhes sobre a iteração com falha.

Para ver esses detalhes, escolha os números de iteração afetados em Falha ou Interrompido na lista suspensa.

Visualização em tabela — Alterne entre diferentes iterações de estado **`Map`**

Se o estado Mapa tiver cinco iterações, e você quiser visualizar os detalhes de execução da terceira e quarta iterações, faça o seguinte:

1. Escolha o estado `Map` cujos dados de iteração diferentes você deseja visualizar.
2. Na exibição em árvore das iterações de estado `Map`, escolha a linha da iteração chamada nº 2 para a iteração número três. Da mesma forma, escolha a linha chamada nº 3 para a iteração número quatro.

Visualização em tabela — Configure as colunas a serem exibidas

Selecione

.

Em seguida, na caixa de diálogo Preferências, escolha as colunas que você deseja exibir em Selecionar colunas visíveis.

Por padrão, esse modo exibe as colunas Nome, Tipo, Status, Recurso e Iniciado depois.

Visualização em tabela — Filtrar os resultados

Limite a quantidade de informações exibidas aplicando um ou mais filtros com base em uma propriedade, como Status, ou em um intervalo de data e hora. Por exemplo, para visualizar as etapas que falharam na execução, aplique o seguinte filtro:

1. Escolha Filtrar por propriedades ou pesquisar por palavra-chave e escolha Status em Propriedades.
2. Em Operadores, escolha Status =.
3. Escolha Status = Falha.
4. (Opcional) Escolha Limpar filtros para remover os filtros aplicados.

Visualização do evento — Filtre os resultados

Limite a quantidade de informações exibidas aplicando um ou mais filtros com base em uma propriedade, como Tipo, ou em um intervalo de data e hora. Por exemplo, para visualizar as etapas do estado `Task` que falharam na execução, aplique o seguinte filtro:

1. Escolha Filtrar por propriedades ou pesquisar por palavra-chave e escolha Tipo em Propriedades.
2. Em Operadores, escolha Tipo =.
3. Escolha Tipo = TaskFailed.
4. (Opcional) Escolha Limpar filtros para remover os filtros aplicados.

Visualização do evento — inspecione os detalhes de um TaskFailedevento

Escolha o

próximo ao ID de um TaskFailedevento para inspecionar seus detalhes, incluindo entrada, saída e invocação de recursos que aparecem em uma caixa suspensa.

Redriving execuções

Você pode usar redrive para reiniciar as execuções de Fluxos de trabalho padrão que não tenham sido concluídas com êxito nos últimos 14 dias. Isso inclui execuções com falha, abortadas ou expiradas.

Quando você redrive uma execução com falha, ela é retomada a partir da etapa falhou, usando a mesma entrada. Step Functions preserva os resultados e o histórico de execução das etapas bem-sucedidas, que não são executadas novamente quando você redrive uma execução. Por exemplo, digamos que seu fluxo de trabalho contenha dois estados: um estado Pass seguido por um Estado da tarefa. Se a execução do fluxo de trabalho falhar no estado Tarefa e você redrive a execução, esta será reagendada, com uma nova execução do estado Tarefa.

As execuções Redriven usam a mesma definição de máquina de estado e ARN de execução utilizados para a tentativa de execução original. Se a tentativa de execução original tiver sido associada a uma versão, alias ou ambos, a execução redriven estará associada à mesma versão, alias ou ambos. Mesmo que você atualize seu alias para apontar para uma versão diferente, a execução redriven continuará usando a versão associada à tentativa de execução original. Como as execuções redriven usam a mesma definição de máquina de estado, você deverá iniciar uma nova execução se quiser atualizar sua definição de máquina de estado.

Quando você redrive uma execução, o tempo limite do nível da máquina de estado, se definido, é redefinido para 0. Para obter mais informações sobre o tempo limite de máquinas de estado, consulte `TimeoutSeconds`.

redrives de execução são considerados transições de estado. Para ver informações sobre como as transições de estado afetam o faturamento, consulte Preços do Step Functions.

Tópicos

- Elegibilidade de Redrive para execuções malsucedidas
- Comportamento de Redrive de estados individuais
- Permissão do IAM para redrive uma execução
- Redriving execuções no console
- Redriving execuções usando API
- Examinar execuções redriven
- Repetir o comportamento das execuções redriven

Elegibilidade de Redrive para execuções malsucedidas

Você pode redrive execuções se sua tentativa original de execução atender às seguintes condições:

- Você iniciou a execução em 15 de novembro de 2023 ou após essa data. As execuções iniciadas antes dessa data não têm direito ao redrive.
- O status de execução não é `SUCCEEDED`.
- A execução do fluxo de trabalho não excedeu o período de redrivable de 14 dias. O período de Redrivable se refere ao tempo durante o qual você pode redrive uma determinada execução. Esse período começa no dia em que uma máquina de estado conclui sua execução.
- A execução do fluxo de trabalho não excedeu o tempo máximo de abertura de um ano. Para obter informações sobre cotas de execução de máquinas de estado, consulte Cotas relacionadas a execuções de máquina de estado.
- A contagem do histórico de eventos de execução é inferior a 24.999. As execuções Redriven anexam o histórico de eventos delas ao de eventos existentes. Certifique-se de que a execução do fluxo de trabalho contenha menos de 24.999 eventos, para acomodar o evento de `ExecutionRedriven` do histórico e pelo menos um outro evento do histórico.

Comportamento de Redrive de estados individuais

O comportamento de redrive de todos os estados malsucedidos varia de acordo com qual estado falhou em seu fluxo de trabalho. A tabela a seguir descreve o comportamento de redrive para todos os estados.

Nome do estado	Comportamento de Redrive de execução
Pass	Se uma etapa anterior falhar ou a máquina de estado atingir o tempo limite, o estado Passagem será encerrado e não será executado no redrive.
Estado da tarefa	Agenda e inicia o estado Tarefa novamente. Quando você redrive uma execução que faz uma nova execução do estado Tarefa, o `TimeoutSeconds` para o estado, se definido, é redefinido para 0. Para obter mais informaçõ es sobre o tempo limite, consulte estado Tarefa.
Choice	Reavalia as regras do estado Escolha.
Aguardar	Se o estado especificar um `Timestamp` ou `TimestampPath` que se refiram a um timestamp no passado, redrive fará com que o estado Aguardar seja encerrado e vai inserir o estado especificado no campo `Next`.
Succeed	Não redrive execuções de máquina de estado que entrem no estado Êxito.
Fail	Entra novamente no estado Falha e há uma nova falha.
Paralelo	Reagenda e redrives somente as ramificações que falharam ou foram abortadas.

Nome do estado	Comportamento de Redrive de execução
	Se o estado falhar devido a um erro `States.DataLimitExceeded`, o estado Paralelo será executado novamente, incluindo as ramificações que tiveram êxito na tentativa de execução original.
Estado Mapa inline	Reagenda e redrives somente as iterações que falharam ou foram abortadas. Se o estado falhar devido a um erro `States.DataLimitExceeded`, o estado Mapa inline será executado novamente, incluindo as iterações que tiveram êxito na tentativa de execução original.
Estado Mapa distribuído	redrives as execuções malsucedidas do fluxo de trabalho secundário em uma Execução de mapa. Para ter mais informações, consulte Redriving execuções de mapa. Se o estado falhar devido a um erro `States.DataLimitExceeded`, o estado Mapa Distribuído será executado novamente . Isso inclui os fluxos de trabalho secundári os que tiveram êxito na tentativa de execução original.

Permissão do IAM para redrive uma execução

Step Functions precisa da permissão apropriada para redrive uma execução. O exemplo de política do IAM a seguir concede o privilégio mínimo necessário à sua máquina de estado para redriving uma execução. Lembre-se de substituir o texto em *`itálico`* pelas informações específicas do recurso.

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

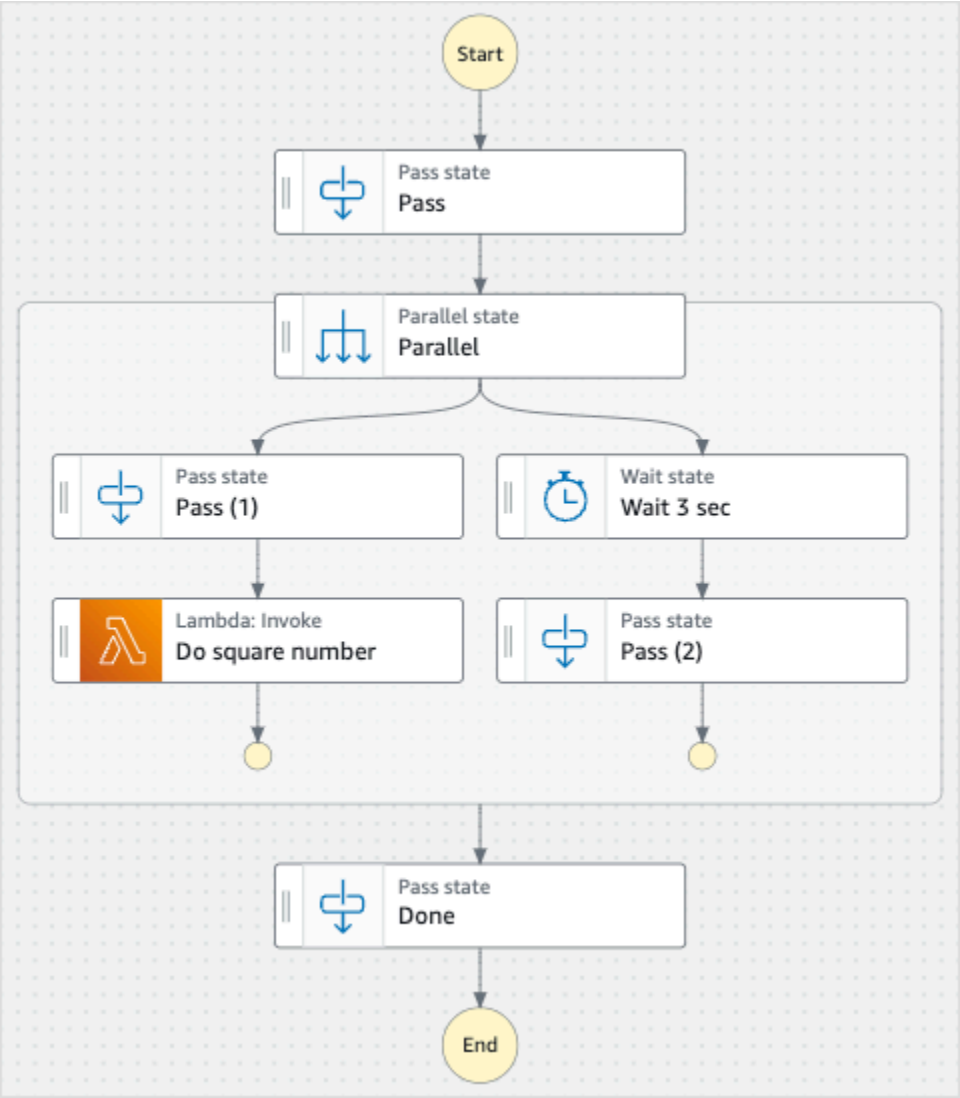
```
        {
            "Effect": "Allow",
            "Action": [
                "states:RedriveExecution"
            ],
            "Resource": "arn:aws:states:us-
east-2:123456789012:execution:myStateMachine:*"
        }
    ]
}
```

Para obter um exemplo da permissão necessária para redrive uma Execução de mapa, consulte Exemplo de política do IAM para redriving um estado Mapa Distribuído.

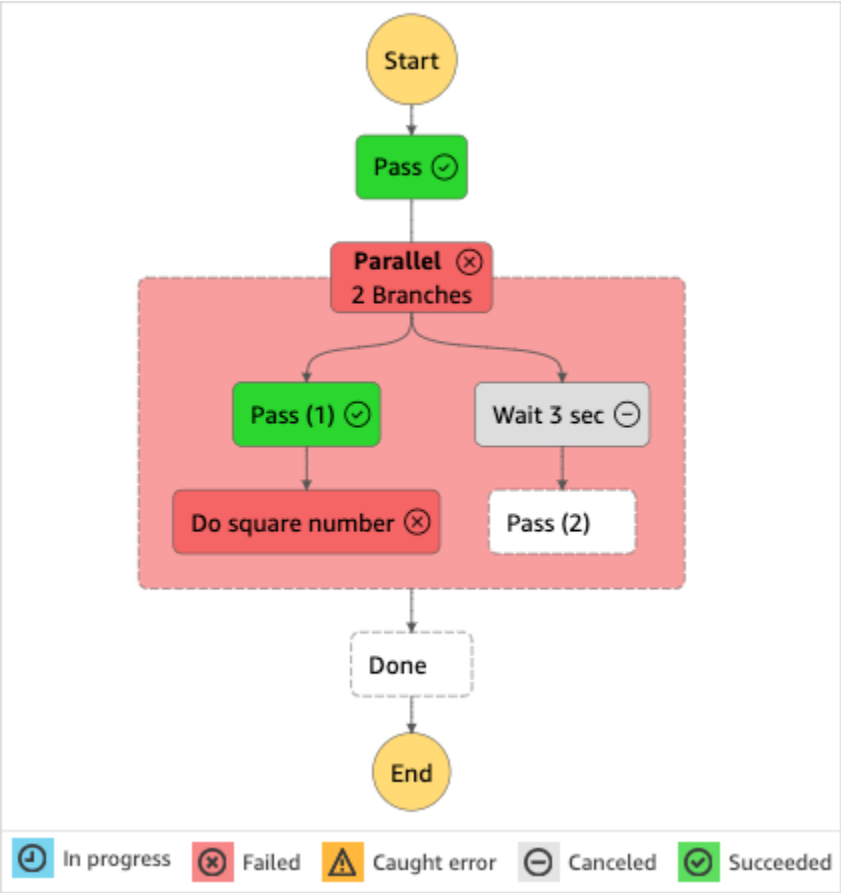
Redriving execuções no console

Você pode redrive execuções elegíveis no console do Step Functions.

Por exemplo, digamos que a imagem a seguir represente o gráfico do fluxo de trabalho da sua máquina de estado.



Imagine que você administre essa máquina de estado. A imagem a seguir mostra o gráfico para a execução de máquina de estado.



Conforme mostrado na imagem, a etapa Invocação Lambda chamada Fazer número quadrado dentro do estado Paralelo retornou um erro. Isso causou uma falha no estado Paralelo. As ramificações cuja execução estava em andamento ou não tinha sido iniciada são interrompidas e a execução da máquina de estado falha.

Para redrive uma execução a partir do console

1. Abra o console do Step Functions e, em seguida, escolha uma máquina de estado existente cuja execução tenha falhado.
2. Na página de detalhes da máquina de estado, em Execuções, escolha uma instância de execução com falha.
3. Selecione Redrive.
4. Na caixa de diálogo Redrive, escolha Redrive execução .

> **Tip**
>
> Se você estiver na página Detalhes da execução de uma execução com falha, faça o seguinte para redrive a execução:
>
> - Escolha Recuperar e, em seguida, selecione Redrive de uma falha.
> - Escolha Ações e, em seguida, selecione Redrive.

Observe que redrive usa a mesma definição de máquina de estado e ARN. A execução é retomada a partir da etapa onde houve a falha na tentativa original. Neste exemplo, a etapa é Fazer número quadrado e a ramificação Aguardar 3 seg dentro do estado Paralelo. Depois de reiniciar a execução dessas etapas malsucedidas no estado Paralelo, o redrive continuará a execução da etapa Concluído.

5. Escolha a execução para abrir a página Detalhes da execução .

   Nessa página, é possível visualizar os resultados da execução redriven. Por exemplo, na seção Resumo da execução, você pode ver a Contagem de Redrive, que representa o número de vezes que uma execução foi redriven. Na seção Eventos, você pode ver os eventos relacionados ao redrive de execução anexados aos da tentativa de execução original. Por exemplo, o evento de `ExecutionRedriven`.

Redriving execuções usando API

Você pode redrive qualificar execuções usando a RedriveExecutionAPI. Essa API reinicia as execuções malsucedidas dos Fluxos de trabalho Padrão a partir da etapa que falhou, foi abortada ou atingiu o tempo limite.

No AWS Command Line Interface (AWS CLI), execute o comando a seguir para redrive uma execução malsucedida da máquina de estado. Lembre-se de substituir o texto em *`itálico`* pelas informações específicas do recurso.

```
aws stepfunctions redrive-execution --execution-arn arn:aws:states:us-
east-2:123456789012:execution:myStateMachine:foo
```

Examinar execuções redriven

Você pode examinar uma redriven execução no console ou usando as APIs: GetExecutionHistorye. DescribeExecution

Examinar as execuções redriven no console

1. Abra o console do Step Functions e, em seguida, escolha uma máquina de estado existente para a qual você redriven uma execução.
2. Abra a página de Detalhes da execução.

   Nessa página, é possível visualizar os resultados da execução redriven. Por exemplo, na seção Resumo da execução, você pode ver a Contagem de Redrive, que representa o número de vezes que uma execução foi redriven. Na seção Eventos, você pode ver os eventos relacionados ao redrive de execução anexados aos da tentativa de execução original. Por exemplo, o evento de `ExecutionRedriven`.

Examinar as execuções redriven usando APIs

Se você tiver redriven uma execução de máquina de estado, poderá usar uma das seguintes APIs para ver detalhes sobre a execução redriven. Lembre-se de substituir o texto em *`itálico`* pelas informações específicas do recurso.

- GetExecutionHistory — Retorna o histórico da execução especificada como uma lista de eventos. Essa API também retorna os detalhes sobre a tentativa redrive de uma execução, se disponível.

  No AWS CLI, execute o comando a seguir.

  ```
  aws stepfunctions get-execution-history --execution-arn arn:aws:states:us-
  east-2:123456789012:execution:myStateMachine:foo
  ```

- DescribeExecution — Fornece informações sobre a execução de uma máquina de estado. Podem ser sobre a máquina de estado associada à execução, entrada e saída da execução, detalhes de redrive da execução, se disponíveis, e metadados de execução relevantes.

  No AWS CLI, execute o comando a seguir.

  ```
  aws stepfunctions describe-execution --execution-arn arn:aws:states:us-
  east-2:123456789012:execution:myStateMachine:foo
  ```

Repetir o comportamento das execuções redriven

Se sua execução redriven executar novamente um estado Estado da tarefa, Paralelo ou Mapa inline, para o qual você definiu novas tentativas, a contagem de novas tentativas para esses estados será redefinida para 0. Isso permite o número máximo de tentativas no redrive. Para uma execução redriven, você pode monitorar tentativas individuais desses estados usando o console.

Para examinar as tentativas individuais de repetição no console

1. Na página Detalhes da execução do console do Step Functions, escolha um estado que foi repetido no redrive.
2. Escolha a guia Tentativas e redrives.
3. Escolha o

   ▶

   ao lado de cada tentativa para visualizar os detalhes. Se a nova tentativa for bem-sucedida, você poderá ver os resultados na Saída que aparece em uma caixa suspensa.

A imagem a seguir mostra um exemplo das novas tentativas realizadas para um estado na tentativa de execução original e o redrives dessa execução. Nesta imagem, três novas tentativas são executadas na tentativa original e na tentativa de redrive a execução. A execução é bem-sucedida na quarta tentativa de redrive e retorna uma saída 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

```
1 ▾ {
2     "Squared": 16
3   }
```

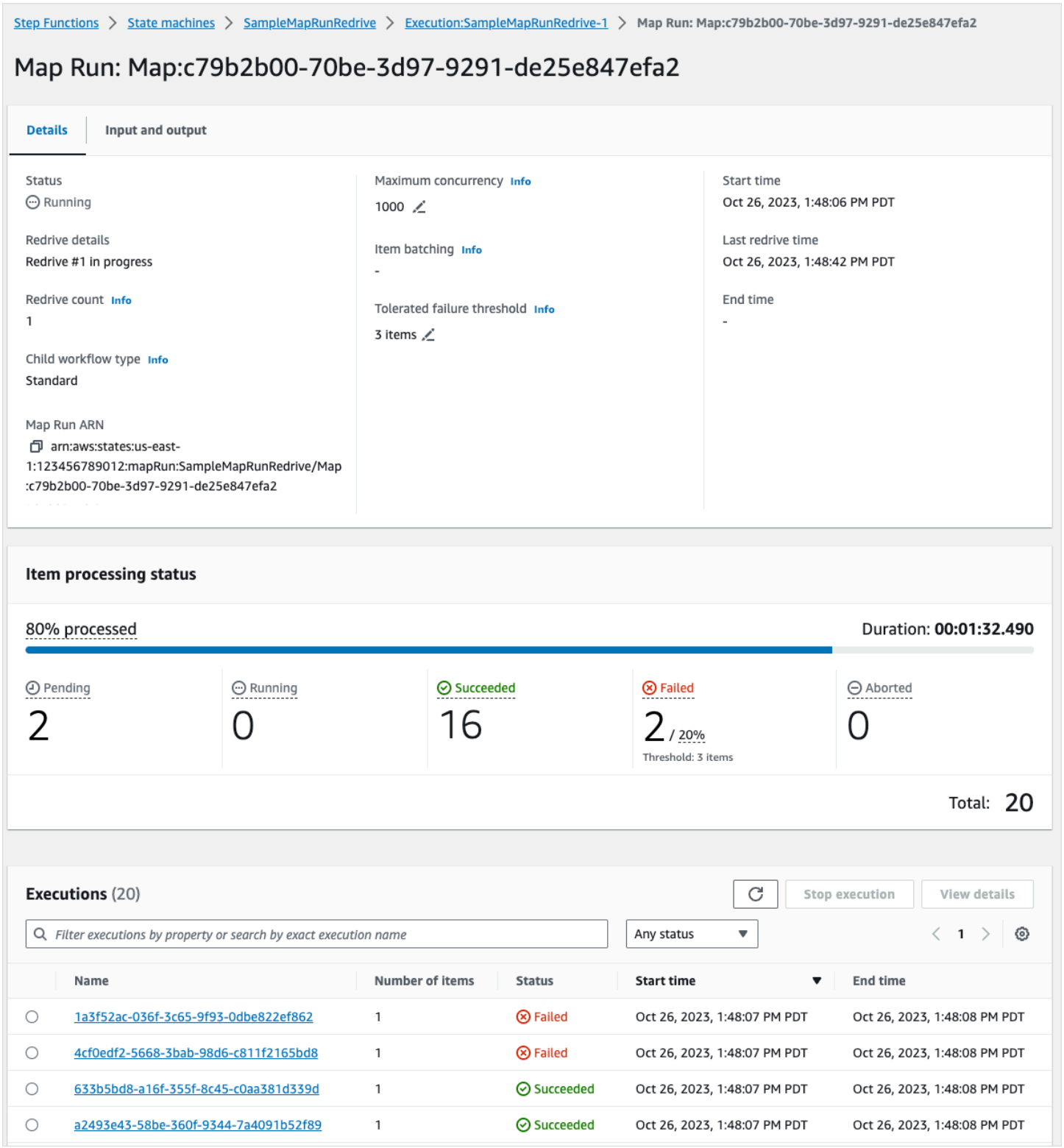
Formatted

Examinando a execução do mapa de uma execução de estado de mapa distribuído

Ao executar um estado `Map` no modo distribuído, o Step Functions cria um recurso de Execução de mapa. Uma Execução de mapa se refere a um conjunto de execuções de fluxo de trabalho secundário que um estado Mapa distribuído inicia e às configurações de runtime que controlam essas execuções. O Step Functions atribui um nome do recurso da Amazon (ARN) à Execução de mapa. Você pode examinar uma Execução de mapa no console do Step Functions. Você também pode invocar a ação da API `DescribeMapRun`. A Map Run também emite métricas para CloudWatch.

O console Step Functions fornece uma página Map Run Details que exibe todas as informações relacionadas à execução de um estado de Mapa Distribuído. Por exemplo, você pode visualizar o status da execução do estado do Mapa Distribuído, o ARN do Map Run e os status dos itens processados nas execuções do fluxo de trabalho secundário iniciadas pelo estado do Mapa Distribuído. Você também pode ver uma lista de todas as execuções do fluxo de trabalho secundário e acessar seus detalhes. Além disso, se sua Execução de Mapa foi redriven, você pode ver os redrive detalhes da Execução de Mapa na Resumo da execução do Map Run seção. Por exemplo, Última redrive vez. O console exibe essas informações em formato de painel.

A página Detalhes da Execução do Mapa contém as seguintes seções:



Conteúdo

- Status de processamento do item
- Lista de execuções
- Redriving execuções de mapa
  - Elegibilidade de Redrive para fluxos de trabalho secundários em uma Execução de mapa
  - Comportamento do redrive de uma execução de fluxo de trabalho secundário
  - Cenários de entrada utilizados no redrive de uma Execução de mapa
  - Permissão do IAM para redrive uma Execução de mapa
  - Redriving uma Execução de mapa no console
  - Redriving Execução de mapa usando a API

Resumo da execução do Map Run

A seção Resumo da Execução da Execução do Mapa aparece na parte superior da página Detalhes da Execução do Mapa. Esta seção fornece uma visão geral dos detalhes de execução do estado do Mapa Distribuído. Essas informações são divididas entre as seguintes guias:

Detalhes

Mostra informações, como o status de execução do estado do Mapa Distribuído, o ARN de Execução do Mapa e o tipo das execuções do fluxo de trabalho secundário iniciadas pelo estado do Mapa Distribuído. Você pode visualizar configurações adicionais, como o limite de falha tolerado para o Map Run e a simultaneidade máxima especificada para execuções de fluxo de trabalho secundário. Você também pode editar essas configurações.

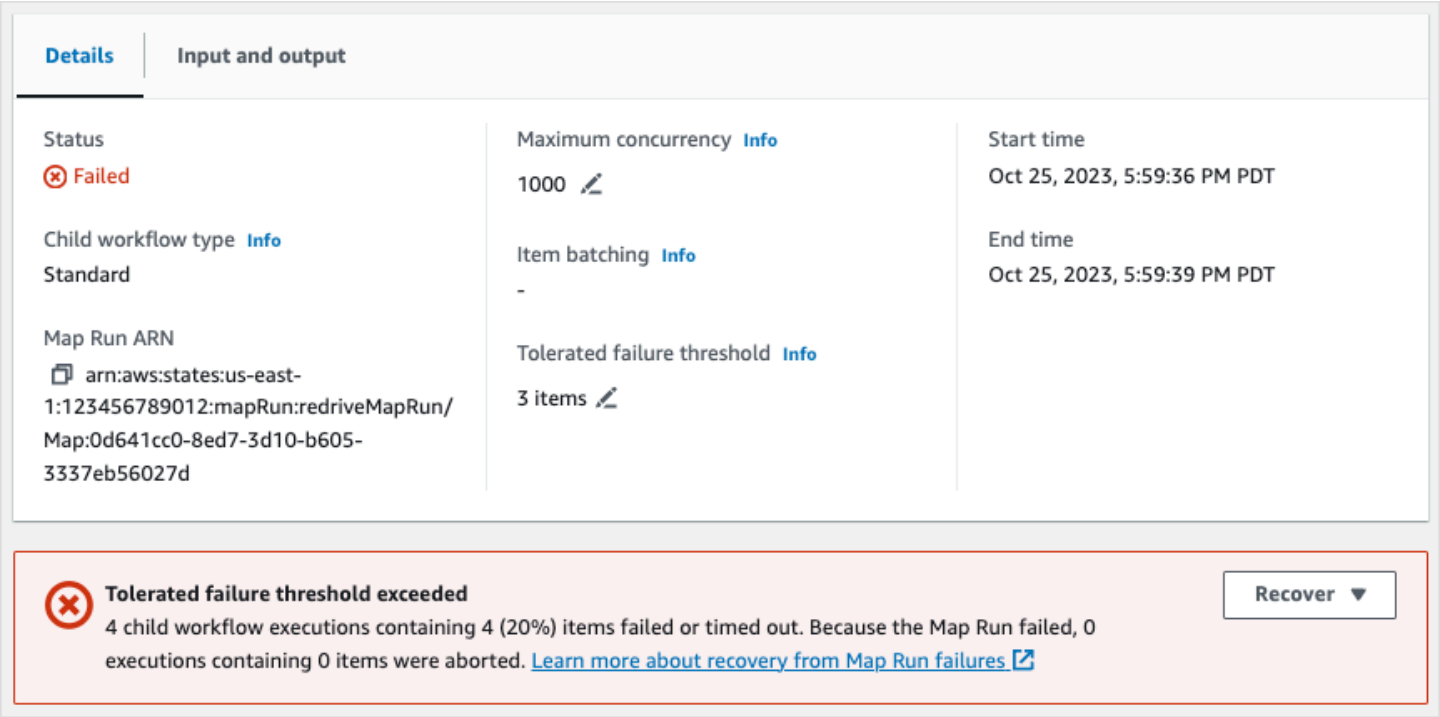
Entrada e saída

Mostra a entrada recebida pelo estado do Mapa Distribuído e a saída correspondente que ele gera. Por exemplo, você pode visualizar o conjunto de dados de entrada e sua localização e os filtros de entrada aplicados aos itens de dados individuais nesse conjunto de dados. Se você exportar a saída da execução do estado do Mapa Distribuído, essa guia mostra o caminho para o bucket do Amazon S3 que contém os resultados da execução. Caso contrário, ele direciona você para a página Detalhes da Execução do fluxo de trabalho principal para visualizar a saída da execução.

Mensagem de erro

Se sua Execução de Mapa falhar, a página Detalhes da Execução de Mapa exibirá uma mensagem de erro com o motivo da falha.

No botão suspenso Recuperar dessa mensagem de erro, você pode executar redrive o fluxo de trabalho secundário malsucedido iniciado por essa Execução de Mapa ou iniciar uma nova execução do fluxo de trabalho principal. Para ter mais informações, consulte Redriving execuções de mapa.



Status de processamento do item

A seção Status de processamento do item exibe o status dos itens processados em uma Execução de Mapa. Por exemplo, Pendente indica que a execução de um fluxo de trabalho secundário ainda não começou a processar o item.

Os status dos itens dependem do status das execuções do fluxo de trabalho secundário que processam os itens. Se a execução de um fluxo de trabalho secundário falhar, atingir o tempo limite ou se um usuário cancelar a execução, o Step Functions não receberá nenhuma informação sobre o resultado do processamento dos itens dentro da execução desse fluxo de trabalho secundário. Todos os itens processados por essa execução compartilham o status da execução do fluxo de trabalho secundário.

Por exemplo, digamos que você queira processar 100 itens em duas execuções secundárias de fluxo de trabalho, em que cada execução processa um lote de 50 itens. Se uma das execuções falhar e a outra for bem-sucedida, você terá 50 itens bem-sucedidos e 50 itens fracassados.

A tabela a seguir explica os tipos de status de processamento disponíveis para todos os itens:

Status	Descrição
Pendente	Indica um item que a execução do fluxo de trabalho secundário não iniciou o processam ento. Se uma execução de mapa parar, falhar ou um usuário cancelar a execução antes do início do processamento de um item, o item permanecerá no status Pendente. Por exemplo, se uma execução de mapa falhar com 10 itens pendentes para serem processad os, esses 10 itens permanecerão no status Pendente.
Running	Indica um item atualmente sendo processado pela execução do fluxo de trabalho secundário.
Sucedido	Indica que a execução do fluxo de trabalho secundário processou o item com êxito. Uma execução bem-sucedida do fluxo de trabalho secundário não pode ter nenhum item com defeito. Se um item no conjunto de dados falhar durante a execução, a execução de todo o fluxo de trabalho secundário falhará.
Com falha	Indica que a execução do fluxo de trabalho secundário falhou ao processar o item ou a execução atingiu o tempo limite. Se qualquer item processado pela execução de um fluxo de trabalho secundário falhar, toda a execução do fluxo de trabalho secundário falhará.

Status	Descrição
	Por exemplo, considere a execução de um fluxo de trabalho secundário que processou 1000 itens. Se algum item desse conjunto de dados falhar durante a execução, o Step Functions considerará que toda a execução do fluxo de trabalho secundário falhou. Quando você executa redriveum mapa, a contagem de itens com esse status é redefinida para 0.

Status	Descrição
Abortado	Indica que a execução do fluxo de trabalho secundário começou a processar o item, mas o usuário cancelou a execução ou o Step Functions interrompeu a execução porque o Map Run falhou. Por exemplo, considere a execução de um fluxo de trabalho secundário em execução que esteja processando 50 itens. Se a execução do mapa parar devido a uma falha ou porque um usuário cancelou a execução, a execução do fluxo de trabalho secundário e o status de todos os 50 itens serão alterados para Abortado. Se você usar uma execução secundária de fluxo de trabalho do tipo Express, não poderá interromper a execução. Quando você usa redriveum Map Run que inicia execuções de fluxo de trabalho secundári o do tipo Express, a contagem de itens com esse status é redefinida para 0. Isso ocorre porque os fluxos de trabalho secundários do Express são reiniciados usando a ação da StartExecutionAPI em vez de serem. redriven

Lista de execuções

A seção Execuções lista todas as execuções do fluxo de trabalho secundário para uma execução de mapa específica. Use o campo Pesquisar pelo nome exato da execução para pesquisar a execução de um fluxo de trabalho secundário específico. Você também pode usar o menu suspenso Qualquer status para filtrar os históricos de execução do fluxo de trabalho secundário por status. Para ver detalhes sobre uma execução específica, selecione a execução de um fluxo de trabalho secundário na lista e escolha o botão Exibir detalhes para abrir a página Detalhes da execução.

> **Important**
>
> A política de retenção para execuções de fluxo de trabalho infantil é de 90 dias. As execuções concluídas de fluxos de trabalho secundários anteriores a esse período de retenção não são exibidas na tabela Execuções. Isso é verdade mesmo se o estado do Mapa Distribuído ou o fluxo de trabalho principal continuar sendo executado por mais tempo do que o período de retenção. Você pode visualizar os detalhes da execução, incluindo os resultados, dessas execuções secundárias do fluxo de trabalho se exportar a saída do estado do Mapa Distribuído para um bucket do Amazon S3 usando. `ResultWriter`

> **Tip**
>
> Escolha o botão Atualizar
>
> 
>
> para ver a lista mais atual de todas as execuções de fluxo de trabalho secundário.

Redriving execuções de mapa

Você pode reiniciar execuções malsucedidas do fluxo de trabalho secundário em uma Execução de mapa redriving o fluxo de trabalho principal. Um fluxo de trabalho principal redriven redrives todos os estados malsucedidos, incluindo o Mapa distribuído. Um fluxo de trabalho principal vai redirecionar estados malsucedidos se não houver nenhum evento `<stateType>Exited` correspondente ao evento `<stateType>Entered` de um estado quando o fluxo de trabalho principal tiver concluído a execução. Por exemplo, se o histórico de eventos não contiver o evento `MapStateExited` de um evento `MapStateEntered`, você poderá redrive o fluxo de trabalho principal para redrive todas as execuções malsucedidas do fluxo de trabalho secundário na Execução de mapa.

Uma Execução de mapa não é iniciada ou apresenta falha na tentativa de execução original quando a máquina de estado não tem a permissão necessária para acessar ItemReader, ResultWriter, ou ambas. Se a Execução de mapa não tiver sido iniciada na tentativa de execução original do fluxo de trabalho principal, redriving o fluxo de trabalho principal iniciará a Execução de mapa pela primeira vez. Para resolver isso, adicione as permissões exigidas à sua função de máquina de estado e, em seguida, redrive o fluxo de trabalho principal. Se você redrive o fluxo de trabalho principal sem adicionar as permissões exigidas, haverá uma tentativa de iniciar uma nova Execução de mapa,

que falhará novamente. Para obter informações sobre as permissões de que você talvez precise, consulte Políticas do IAM para usar o estado Mapa Distribuído.

Tópicos

Elegibilidade de Redrive para fluxos de trabalho secundários em uma Execução de mapa

Você conseguirá redrive execuções malsucedidas do fluxo de trabalho secundário em uma Execução de mapa se as seguintes condições forem atendidas:

- Você iniciou a execução do fluxo de trabalho principal em 15 de novembro de 2023 ou após essa data. As execuções iniciadas antes dessa data não têm direito ao redrive.
- Você não excedeu o limite máximo de mil redrives em uma determinada Execução de mapa. Se você excedeu esse limite, verá o erro `States.Runtime`.
- O fluxo de trabalho principal é redrivable. Se o fluxo de trabalho principal não for redrivable, você não conseguirá redrive as execuções do fluxo de trabalho secundário em uma Execução de mapa. Para obter mais informações sobre a elegibilidade de redrive um fluxo de trabalho, consulte Elegibilidade de Redrive para execuções malsucedidas.
- As execuções do fluxo de trabalho secundário do tipo Padrão em uma Execução de mapa não excederam o limite do histórico de 25 mil eventos de execução. As execuções de fluxo de trabalho secundárias que excederam o limite do histórico de eventos são contabilizadas no limite de falhas tolerado e considera-se que falharam. Para obter mais informações sobre a elegibilidade de redrive de uma execução, consulte Elegibilidade de Redrive para execuções malsucedidas.

Uma nova Execução de mapa é iniciada e a existente não é redriven nos seguintes casos, mesmo que a tentativa original de Execução de mapa tenha falhado:

- A Execução de mapa falhou devido ao erro `States.DataLimitExceeded`.

- A Execução de mapa falhou devido ao erro de interpolação de dados JSON, `States.Runtime`. Por exemplo, você selecionou um nó JSON inexistente em OutputPath.

Uma Execução de mapa pode continuar mesmo após o fluxo de trabalho principal parar ou atingir o tempo limite. Nesses cenários, o redrive isso não acontece imediatamente:

- A Execução de mapa talvez ainda esteja cancelando execuções em andamento de fluxos de trabalho secundários do tipo Padrão ou aguardando que execuções de fluxos de trabalho secundários do tipo expresso sejam concluídas.
- A Execução de mapa talvez ainda esteja gravando resultados no ResultWriter, se você a tiver configurado para exportar resultados.

Nesses casos, a Execução de mapa conclui suas operações antes de tentar redrive.

Comportamento do redrive de uma execução de fluxo de trabalho secundário

As execuções do fluxo de trabalho secundário redriven em uma Execução de mapa exibem o comportamento descrito na tabela a seguir.

Fluxo de trabalho expresso secundário	Fluxo de trabalho padrão secundário
Todas as execuções do fluxo de trabalho secundário que falharam ou atingiram o tempo limite na tentativa de execução original são iniciadas usando a ação da StartExecutionAPI. O primeiro estado em ItemProcessor é executado primeiro.	Todas as execuções de um fluxo de trabalho secundário que tiverem falhado, sido cancelados ou atingirem o tempo limite na tentativa de execução original são redriven usando a ação da API RedriveExecution. Esses fluxos de trabalho secundários são redriven do último estado em ItemProcessor que resultou em sua execução malsucedida.
Execuções malsucedidas sempre podem ser redriven. Isso ocorre porque as execuções do fluxo de trabalho secundário do Express são sempre iniciadas como uma nova execução usando a ação da StartExecution API.	As execuções malsucedidas do fluxo de trabalho padrão secundário nem sempre podem ser redriven. Se a execução não for redrivable, não haverá uma nova tentativa. O último erro ou saída da execução é permanente. Isso pode acontecer quando uma execução

Fluxo de trabalho expresso secundário	Fluxo de trabalho padrão secundário
	excede o limite de 25 mil eventos do histórico ou seu período redrivable ultrapassou 14 dias. A execução do fluxo de trabalho padrão secundário talvez não seja redrivable se a execução do fluxo de trabalho principal tiver sido encerrada dentro de 14 dias, mas a execução do fluxo de trabalho secundário for encerrada antes de 14 dias.
As execuções do fluxo de trabalho expresso secundário usam o mesmo ARN de execução da tentativa de execução original, mas não é possível identificar claramente os redrives individuais.	As execuções do fluxo de trabalho padrão secundário usam o mesmo ARN de execução da tentativa de execução original. Você pode identificar claramente o indivíduo redrives no console e usando APIs, como e. GetExecutionHistoryDescribeExecution Para ter mais informações, consulte Examinar execuções redriven.

Se você tiver uma Execução de mapa redriven e ela tiver atingido seu limite de simultaneidade, as execuções do fluxo de trabalho secundário nela farão a transição para o estado pendente. O status da Execução de mapa também passa para o estado redrive pendente. A execução permanecerá no estado redrive pendente até que o limite de simultaneidade especificado permita a execução de mais execuções de fluxo de trabalho secundário.

Por exemplo, digamos que o limite de simultaneidade de Mapa distribuído em seu fluxo de trabalho seja 3 mil e o número de fluxos de trabalho secundários a serem executados novamente seja de 6 mil. Isso faz com que 3 mil fluxos de trabalho secundários sejam executados em paralelo, enquanto os 3 mil fluxos de trabalho restantes permanecem no estado Redirecionamento pendente. Após a conclusão da execução do primeiro lote de 3 mil fluxos de trabalho secundários, os 3 mil restantes serão executados.

Quando uma Execução de mapa é concluída ou abortada, a contagem de execuções de fluxos de trabalho secundários no estado redrivependente é redefinida para 0.

Cenários de entrada utilizados no redrive de uma Execução de mapa

Dependendo de como você forneceu a entrada para Mapa distribuído na tentativa de execução original, uma Execução de mapa redriven usará a entrada como descrito na tabela a seguir.

Entrada na tentativa de execução original	Entrada utilizada no redrive de Execução de mapa
Entrada passada de um estado anterior ou entrada de execução.	A Execução de mapa redriven usa a mesma entrada.
A entrada passada usando ItemReader a Execução de mapa não iniciaram as execuções do fluxo de trabalho secundário porque uma das seguintes condições é verdadeira: • A Execução do mapa falhou com o erro `States.ItemReaderFailed`. • A Execução do mapa falhou com o erro `States.ResultWriterFailed`. • A execução do fluxo de trabalho principal atingiu o tempo limite ou foi cancelada antes do início da Execução de mapa.	A Execução de mapa redriven usa a entrada no bucket do Amazon S3.
Entrada passada usando ItemReader. A Execução de mapa falhou após o início ou tentativa de iniciar as execuções do fluxo de trabalho secundário.	A Execução de mapa redriven usa a mesma entrada fornecida na tentativa de execução original.

Permissão do IAM para redrive uma Execução de mapa

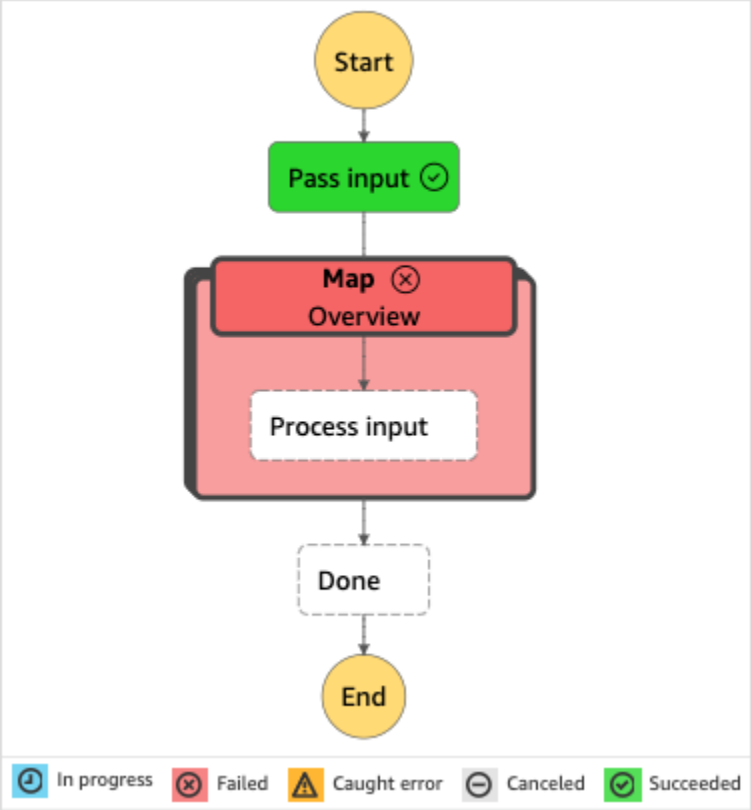
Step Functions precisa da permissão apropriada para redrive uma Execução de mapa. O exemplo de política do IAM a seguir concede o privilégio mínimo necessário à sua máquina de estado para redriving uma Execução de mapa. Lembre-se de substituir o texto em *`itálico`* pelas informações específicas do recurso.

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

```
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:RedriveExecution"
      ],
      "Resource": "arn:aws:states:us-
east-2:123456789012:execution:myStateMachine/myMapRunLabel:*"
    }
  ]
}
```

Redriving uma Execução de mapa no console

A imagem a seguir mostra o gráfico de execução de uma máquina de estado que contém um Mapa distribuído. Essa execução falhou porque a Execução de mapa falhou. Para redrive a Execução de mapa, você deve redrive o fluxo de trabalho principal.



Para redrive uma Execução de mapa no console

1. Abra o console do Step Functions e, em seguida, escolha uma máquina de estado existente que contenha um Mapa distribuído cuja execução tenha falhado.

2. Na página de detalhes da máquina de estado, em Execuções, escolha uma instância de execução com falha dessa máquina.
3. Selecione Redrive.
4. Na caixa de diálogo Redrive, escolha Redrive execução .

> **Tip**
>
> Você também pode redrive uma Execução de mapa na página Detalhes da execução ou Detalhes da execução de mapa.
> Se você estiver na página Detalhes da execução, faça o seguinte para redrive a execução:
>
> - Escolha Recuperar e, em seguida, selecione Redrive de uma falha.
> - Escolha Ações e, em seguida, selecione Redrive.
>
> Se você estiver na página Detalhes da execução de mapa, escolha Recuperar e, em seguida, selecione Redrive de uma falha.

Observe que redrive usa a mesma definição de máquina de estado e ARN. A execução é retomada a partir da etapa onde houve a falha na tentativa original. Neste exemplo, isso ocorre na etapa de Mapa distribuído chamada Mapa e a etapa de Entrada do processo dentro dela. Depois de reiniciar as execuções malsucedidas do fluxo de trabalho secundário da Execução mapa, o redrive continuará a execução para a etapa Concluído.

5. Na página Detalhes da execução, escolha Execução de mapa para ver os detalhes da Execução de mapa redriven.

   Nessa página, é possível visualizar os resultados da execução redriven. Por exemplo, na seção Resumo da execução do Map Run, você pode ver a Contagem de Redrive, que representa o número de vezes que uma Execução de mapa foi redriven. Na seção Eventos, você pode ver os eventos relacionados ao redrive de execução anexados aos da tentativa de execução original. Por exemplo, o evento de `MapRunRedriven`.

Depois de executar redriven um mapa, você pode examinar seus redrive detalhes no console ou usando as ações GetExecutionHistorye DescribeExecutionda API. Para mais informações sobre o exame de uma execução redriven, consulte Examinar execuções redriven.

Redriving Execução de mapa usando a API

Você pode redrive uma Execução de mapa elegível usando a API RedriveExecution no fluxo de trabalho principal. Essa API reinicia as execuções malsucedidas do fluxo de trabalho secundário em uma Execução de mapa.

Na AWS Command Line Interface (AWS CLI), execute o comando a seguir para redrive uma execução malsucedida da máquina de estado. Lembre-se de substituir o texto em *itálico* pelas informações específicas do recurso.

```
aws stepfunctions redrive-execution --execution-arn arn:aws:states:us-
east-2:123456789012:execution:myStateMachine:foo
```

Depois de executar redriven um mapa, você pode examinar seus redrive detalhes no console ou usando a ação da DescribeMapRunAPI. Para examinar os redrive detalhes das execuções do fluxo de trabalho padrão em uma execução de mapa, você pode usar a ação GetExecutionHistoryou DescribeExecutionAPI. Para mais informações sobre o exame de uma execução redriven, consulte the section called “Examinar execuções redriven”.

Você conseguirá examinar os detalhes de redrive das execuções do fluxo de trabalho expresso em uma Execução de mapa no console do Step Functions se tiver ativado o registro no fluxo de trabalho principal. Para ter mais informações, consulte Como registrar usando o CloudWatch Logs.

Tratamento de erros no Step Functions

Todos os estados, exceto `Pass` e `Wait`, podem encontrar erros de runtime. Erros podem ocorrer por diversos motivos, como nos seguintes exemplos:

- Problemas de definição da máquina de estado (por exemplo, nenhuma regra de correspondência em um estado `Choice`)
- Falhas de tarefa (por exemplo, uma exceção em uma função AWS Lambda)
- Problemas transitórios (por exemplo, eventos de partição de rede)

Por padrão, quando um estado relata um erro, o AWS Step Functions faz com que a execução falhe totalmente.

> **Tip**
>
> Para implementar um exemplo de fluxo de trabalho que inclui tratamento de erros em sua Conta da AWS, consulte o Módulo 8 - Tratamento de erros do workshop do AWS Step Functions.

Tópicos

Nomes de erro

O Step Functions identifica os erros na Amazon States Language usando strings que fazem distinção de maiúsculas e minúsculas, conhecidas como nomes de erro. A Amazon States Language define um conjunto de strings integradas que nomeiam erros conhecidos, todos iniciados com o prefixo `States.`.

`States.ALL`

Um curinga que corresponde a qualquer nome de erro conhecido.

> **Note**
>
> Esse tipo de erro não pode capturar o tipo de erro do terminal `States.DataLimitExceeded` e os tipos de erro de runtime. Para ver mais informações sobre esses tipos de regra, consulte `States.DataLimitExceeded` e `States.Runtime`.

`States.DataLimitExceeded`

O Step Functions relata uma exceção de `States.DataLimitExceeded` nas seguintes condições:

- Quando a saída de um conector é maior que a cota de tamanho da carga.
- Quando a saída de um estado é maior que a cota de tamanho da carga.
- Após o processamento de `Parameters`, quando a entrada de um estado for maior que a cota de tamanho da carga.

Para ver mais informações sobre cotas, consulte Cotas.

> **Note**
>
> Esse é um erro de terminal que não pode ser detectado pelo tipo de erro `States.ALL`.

`States.ExceedToleratedFailureThreshold`

Um estado `Map` falhou porque o número de itens com falha excedeu o limite especificado na definição da máquina de estado. Para obter mais informações, consulte Limite de falha tolerado para o estado Mapa Distribuído.

`States.HeartbeatTimeout`

Um estado `Task` não conseguiu enviar uma pulsação por um período maior que o valor de `HeartbeatSeconds`.

> **Note**
>
> Esse erro só está disponível nos campos `Catch` e `Retry`.

`States.IntrinsicFailure`

Uma tentativa de invocar uma função intrínseca em um modelo de carga falhou.

`States.ItemReaderFailed`

Um estado `Map` falhou porque não foi possível ler a partir da fonte do item especificada no campo `ItemReader`. Para obter mais informações, consulte `ItemReader`.

`States.NoChoiceMatched`

Um estado `Choice` falhou em combinar a entrada com as condições definidas na Regra de escolha e uma transição Padrão não foi especificada.

States.ParameterPathFailure

Falha em uma tentativa de substituir um campo, dentro do campo `Parameters` de um estado, cujo nome termina em `.$` usando um caminho.

States.Permissions

Um estado `Task` falhou porque tinha privilégios insuficientes para executar o código especificado.

States.ResultPathMatchFailure

O Step Functions não conseguiu aplicar o campo `ResultPath` de um estado à entrada que o estado recebeu.

States.ResultWriterFailed

Um estado `Map` falhou porque não conseguiu gravar os resultados no destino especificado no campo `ResultWriter`. Para obter mais informações, consulte `ResultWriter`.

States.Runtime

Uma execução falhou devido a alguma exceção que não pôde ser processada. Muitas vezes, isso é causado por erros em tempo de execução, como tentar aplicar `InputPath` ou `OutputPath` em uma carga útil JSON nula. Um erro `States.Runtime` não é recuperável e sempre fará com que a execução falhe. Uma nova tentativa ou captura em `States.ALL` não detectará erros `States.Runtime`.

States.TaskFailed

Um estado `Task` falhou durante a execução. Quando usado em uma nova tentativa ou captura, `States.TaskFailed` age como um curinga que corresponde a qualquer nome de erro conhecido, exceto `States.Timeout`.

States.Timeout

Um estado `Task` que executou por um tempo mais longo que o valor `TimeoutSeconds` ou não conseguiu enviar uma pulsação por um período maior que o valor `HeartbeatSeconds`.

Além disso, se uma máquina de estado for executada por mais tempo do que o valor de `TimeoutSeconds` especificado, a execução falhará com um erro `States.Timeout`.

Os estados podem relatar erros com outros nomes. No entanto, esses nomes de erro não podem começar com o prefixo `States.`.

Como uma melhor prática, verifique se o seu código de produção pode lidar com exceções do serviço do AWS Lambda (`Lambda.ServiceException` e `Lambda.SdkClientException`). Para obter mais informações, consulte Lidar com exceções do serviço Lambda.

> **Note**
>
> Os erros não tratados no Lambda são relatados como `Lambda.Unknown` na saída do erro. Isso inclui erros de falta de memória e tempos limite de função. Você pode combinar com `Lambda.Unknown`, `States.ALL` ou `States.TaskFailed` para lidar com esses erros. Quando o Lambda atinge o número máximo de invocações, o erro é `Lambda.TooManyRequestsException`. Para obter mais informações sobre erros da função do Lambda, consulte Tratamento de erros e novas tentativas automáticas no Guia do desenvolvedor do AWS Lambda.

Nova tentativa após um erro

Os estados `Task`, `Parallel` e `Map` podem ter um campo denominado `Retry`, cujo valor deve ser uma matriz de objetos conhecidos como retriers. Um retrier individual representa determinado número de novas tentativas, geralmente em intervalos de tempo crescentes.

Quando um desses estados relata um erro e há um campo `Retry`, o Step Functions examina os recuperadores na ordem listada na matriz. Quando o nome do erro aparece no valor do campo `ErrorEquals` de um retrier, a máquina de estado faz novas tentativas conforme definido no campo `Retry`.

Se a execução do seu redriven executar novamente um estado Estado da tarefa, Paralelo ou Mapa inline, para o qual você definiu novas tentativas, a contagem de novas tentativas para esses estados será redefinida para 0 a fim de permitir o número máximo de tentativas realizadas em redrive Para uma execução redriven, você pode monitorar tentativas individuais desses estados usando o console. Para obter mais informações, consulte Repetir o comportamento das execuções redriven em Redriving execuções.

Um retrier contém os seguintes campos:

> **Note**
>
> As novas tentativas são tratadas como transições de estado. Para ver informações sobre como as transições de estado afetam o faturamento, consulte Preços do Step Functions.

`ErrorEquals` (obrigatório)

Uma matriz não vazia de strings que correspondem a Nomes de erro. Quando um estado relata um erro, o Step Functions examina os retriers. Quando o nome do erro é exibido na matriz, ele implementa a política de novas tentativas descrita nesse retrier.

`IntervalSeconds` (opcional)

Um inteiro positivo que representa o número de segundos antes da primeira tentativa nova (por padrão, 1). `IntervalSeconds` tem um valor máximo de 99999999.

`MaxAttempts` (opcional)

Um inteiro positivo que representa o número máximo de tentativas novas (por padrão, 3). Se o erro voltar a ocorrer mais vezes do que o especificado, as novas tentativas são interrompidas e o tratamento de erro normal é retomado. Um valor de `0` especifica que o erro nunca será repetido. `MaxAttempts` tem um valor máximo de 99999999.

`BackoffRate` (opcional)

O multiplicador pelo qual o intervalo de nova tentativa denotado por `IntervalSeconds` aumenta após cada nova tentativa. Por padrão, o valor de `BackoffRate` aumenta em `2.0`.

Por exemplo, digamos que o seu `IntervalSeconds` é 3, `MaxAttempts` é 3 e `BackoffRate` é 2. A primeira nova tentativa ocorre três segundos após a ocorrência do erro. A segunda nova tentativa ocorre seis segundos após a primeira. Enquanto a terceira nova tentativa ocorre 12 segundos após a segunda.

`MaxDelaySeconds` (opcional)

Um número inteiro positivo que define o valor máximo, em segundos, até o qual um intervalo de nova tentativa pode aumentar. Esse campo é útil para ser usado com o campo `BackoffRate`. O valor especificado nesse campo limita os tempos de espera exponenciais resultantes do multiplicador da taxa de recuo aplicado a cada nova tentativa consecutiva. Você deve especificar um valor maior que 0 e menor que 31622401 para `MaxDelaySeconds`.

Se você não especificar esse valor, o Step Functions não limitará os tempos de espera entre as novas tentativas.

`JitterStrategy` (opcional)

Uma sequência de caracteres que determina se a oscilação deve ou não ser incluída nos tempos de espera entre novas tentativas consecutivas. A oscilação reduz as novas tentativas simultâneas distribuindo-as em um intervalo de atraso aleatório. Essa string aceita `FULL` ou `NONE` como seus valores. O valor padrão é `NONE`.

Por exemplo, digamos que você tenha definido `MaxAttempts` como 3, `IntervalSeconds` como 2 e `BackoffRate` como 2. A primeira nova tentativa ocorre dois segundos após a ocorrência do erro. A segunda nova tentativa ocorre quatro segundos após a primeira e a terceira ocorre oito segundos após a segunda. Se você definir `JitterStrategy` como `FULL`, o intervalo da primeira nova tentativa será randomizado entre 0 e 2 segundos, o intervalo da segunda nova repetição será randomizado entre 0 e 4 segundos e o intervalo da terceira nova tentativa será randomizado entre 0 e 8 segundos.

Exemplos do campo de nova tentativa

Essa seção inclui os seguintes exemplos do campo `Retry`.

- Retry with BackoffRate
- Retry with MaxDelaySeconds
- Retry all errors except States.Timeout
- Complex retry scenario

> **Tip**
>
> Para implementar um exemplo de fluxo de trabalho de tratamento de erros em sua Conta da AWS, consulte o módulo de Tratamento de erros do workshop do AWS Step Functions.

Exemplo 1 — Nova tentativa com BackOffRate

O exemplo a seguir de uma `Retry` faz duas novas tentativas, com a primeira ocorrendo após uma espera de três segundos. Com base no `BackoffRate` especificado, o Step Functions aumenta o intervalo entre cada nova tentativa até que o número máximo de novas tentativas seja atingido. No

exemplo a seguir, a segunda nova tentativa começa depois de uma espera de três segundos após a primeira nova tentativa.

```
"Retry": [ {
   "ErrorEquals": [ "States.Timeout" ],
   "IntervalSeconds": 3,
   "MaxAttempts": 2,
   "BackoffRate": 1
} ]
```

Exemplo 2 — Nova tentativa com MaxDelaySeconds

O exemplo a seguir faz três novas tentativas e limita o tempo de espera resultante de `BackoffRate` para 5 segundos. A primeira nova tentativa ocorre após uma espera de três segundos. A segunda e a terceira novas tentativas ocorrem depois de uma espera de cinco segundos após a nova tentativa anterior, devido ao limite máximo de tempo de espera definido por `MaxDelaySeconds`.

```
"Retry": [ {
    "ErrorEquals": [ "States.Timeout" ],
    "IntervalSeconds": 3,
    "MaxAttempts": 3,
    "BackoffRate":2,
    "MaxDelaySeconds": 5,
    "JitterStrategy": "FULL"
} ]
```

Sem `MaxDelaySeconds`, a segunda nova tentativa ocorreria seis segundos após a primeira e a terceira ocorreria 12 segundos após a segunda.

Exemplo 3 — Nova tentativa de todos os erros, exceto States.Timeout

O nome reservado `States.ALL` que aparece no campo `ErrorEquals` de um retrier é um curinga que corresponde a qualquer nome de erro. Ele deve aparecer sozinho na matriz `ErrorEquals` e também no último retrier na matriz `Retry`. O nome `States.TaskFailed` também atua como um curinga e corresponde a qualquer erro, exceto `States.Timeout`.

O segundo exemplo de um campo `Retry` faz nova tentativa em relação a qualquer erro, exceto `States.Timeout`.

```
"Retry": [ {
```

```
   "ErrorEquals": [ "States.Timeout" ],
   "MaxAttempts": 0
}, {
   "ErrorEquals": [ "States.ALL" ]
} ]
```

Exemplo 4 — Cenário complexo de nova tentativa

Parâmetros de um retrier em todas as visitas ao retrier no contexto de uma única execução de estado.

Considere o seguinte 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"
   } ]
}
```

Essa tarefa falha quatro vezes sucessivas, produzindo os seguintes nomes de erro: `ErrorA`, `ErrorB`, `ErrorC` e `ErrorB`. O seguinte ocorre em consequência disso:

- Os dois primeiros erros correspondem ao primeiro retrier e provocam esperas de um e dois segundos.
- O terceiro erro corresponde ao segundo retrier e provoca uma espera de cinco segundos.
- O quarto erro também corresponde ao primeiro retrier. No entanto, ele já atingiu o máximo de duas novas tentativas (`MaxAttempts`) para esse erro específico. Portanto, esse retrier falha e a execução redireciona o fluxo de trabalho para o estado `Z` por meio do campo `Catch`.

Estados de fallback

Os estados `Task`, `Map` e `Parallel` podem ter um campo denominado `Catch`. O valor desse campo deve ser uma matriz de objetos, conhecidos como catchers.

Um catcher contém os campos a seguir.

`ErrorEquals` (obrigatório)

Uma matriz não vazia de strings que correspondem a nomes de erros, especificados exatamente como aparecem no campo do retrier de mesmo nome.

`Next` (obrigatório)

Uma string que deve corresponder exatamente a um dos nomes de estado da máquina de estado.

`ResultPath` (opcional)

Um caminho que determina qual entrada o catcher envia ao estado especificado no campo `Next`.

Quando um estado relata um erro e não existe nenhum campo `Retry` ou as novas tentativas não conseguem resolver o erro, o Step Functions examina os catchers na ordem listada na matriz. Quando o nome do erro é exibido no valor do campo `ErrorEquals` de um catcher, a máquina de estado muda para o estado denominado no campo `Next`.

O nome reservado `States.ALL` que aparece em um campo `ErrorEquals` do catcher é um curinga que corresponde a qualquer nome de erro. Ele deve aparecer sozinho na matriz `ErrorEquals` e também no último catcher na matriz `Catch`. O nome `States.TaskFailed` também atua como um curinga e corresponde a qualquer erro, exceto `States.Timeout`.

O exemplo a seguir de um campo `Catch` muda para o estado denominado `RecoveryState` quando uma função do Lambda gera uma exceção Java não tratada. Do contrário, o campo muda para o estado `EndState`.

```
"Catch": [ {
   "ErrorEquals": [ "java.lang.Exception" ],
   "ResultPath": "$.error-info",
   "Next": "RecoveryState"
}, {
   "ErrorEquals": [ "States.ALL" ],
```

```
   "Next": "EndState"
} ]
```

> **Note**
>
> Todo catcher pode especificar vários erros para tratamento.

Error output (Saída de erro)

Quando o Step Functions muda para o estado especificado em um nome de catch, o objeto normalmente contém o campo `Cause`. O valor desse campo é uma descrição de erro humanamente. Esse objeto é conhecido como saída de erro.

Neste exemplo, o primeiro catcher contém um campo `ResultPath`. Ele funciona de modo semelhante a um campo `ResultPath` em um nível superior do estado, o que resulta em duas possibilidades:

- Ele pega os resultados da execução desse estado e substitui toda a entrada do estado ou uma parte dela.
- Ele leva os resultados e os adiciona à entrada. No caso de um erro tratado por um catcher, o resultado da execução do estado é a saída de erro.

Assim, para o primeiro catcher do exemplo, o catcher adicionará a saída de erro à entrada como um campo chamado `error-info` se ainda não houver um campo com esse nome na entrada. Em seguida, o catcher envia toda a entrada para `RecoveryState`. Para o segundo catcher, a saída de erro substitui a entrada e o catcher envia somente a saída de erro para `EndState`.

> **Note**
>
> Se você não especificar o campo `ResultPath`, ele usará como padrão `$`, que seleciona e substitui a entrada completa.

Quando um estado tem os campos `Retry` e `Catch`, o Step Functions usa primeiro qualquer retrier apropriado. Se a política de nova tentativa não solucionar o erro, o Step Functions aplicará a transição correspondente do catcher.

Cargas de causa e integrações de serviços

Um catcher retorna uma carga de string como saída. Ao trabalhar com integrações de serviços, como Amazon Athena ou AWS CodeBuild, talvez você queira converter a string `Cause` em JSON. O exemplo a seguir de um estado `Pass` com funções intrínsecas mostra como converter uma string `Cause` em JSON.

```
"Handle escaped JSON with JSONtoString": {
  "Type": "Pass",
  "Parameters": {
    "Cause.$": "States.StringToJson($.Cause)"
  },
  "Next": "Pass State with Pass Processing"
},
```

Exemplos de máquina de estado que usam Nova tentativa e Detecção

As máquinas de estado definidas nos exemplos a seguir presumem que existem duas funções do Lambda: uma que sempre falha e uma que aguarda tempo suficiente para permitir que haja um tempo limite definido na máquina de estado.

Essa é uma definição de função do Lambda Node.js que sempre falha, retornando a mensagem `error`. Nos exemplos de máquina de estado a seguir, essa função do Lambda é denominada `FailFunction`. Para ver mais informações sobre a criação de uma função do Lambda, consulte a seção Etapa 1: criar uma função do Lambda.

```
exports.handler = (event, context, callback) => {
    callback("error");
};
```

Essa é uma definição de função do Lambda Node.js que hiberna por 10 segundos. Nos exemplos de máquina de estado a seguir, essa função do Lambda é denominada `sleep10`.

> **Note**
>
> Ao criar essa função do Lambda no console do Lambda, lembre-se de alterar o valor Tempo limite na seção Configurações avançadas de 3 segundos (padrão) para 11 segundos.

```
exports.handler = (event, context, callback) => {
```

```
    setTimeout(function(){
    }, 11000);
};
```

Tratamento de falhas usando novas tentativas

Essa máquina de estado usa um campo `Retry` para tentar novamente uma função que falha e gera o nome de erro `HandledError`. Duas novas tentativas dessa função são feitas com um recuo exponencial entre as novas tentativas.

```
{
   "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
      }
   }
}
```

Essa variante usa o código de erro predefinido `States.TaskFailed`, que corresponde a qualquer erro gerado por uma função do Lambda.

```
{
   "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**
>
> Como uma prática recomendada, as tarefas que fazem referência a uma função do Lambda devem lidar com exceções do serviço Lambda. Para obter mais informações, consulte Lidar com exceções do serviço Lambda.

Tratamento de falhas usando detecção

Este exemplo usa um campo `Catch`. Quando uma função do Lambda gera um erro, o erro é detectado e a máquina de estado muda para o 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
      }
   }
```

```
}
```

Essa variante usa o código de erro predefinido `States.TaskFailed`, que corresponde a qualquer erro gerado por uma função do Lambda.

```
{
   "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
      }
   }
}
```

Tratamento de tempo limite usando novas tentativas

Essa máquina de estado usa um campo `Retry` para fazer uma nova tentativa do estado `Task` que atinge o tempo limite, com base no valor de tempo limite especificado em `TimeoutSeconds`. O Step Functions faz duas novas tentativas da invocação da função do Lambda nesse estado `Task`, com um recuo exponencial entre as novas tentativas.

```
{
   "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
      }
   }
}
```

Tratamento de tempo limite usando detecção

Este exemplo usa um campo `Catch`. Quando ocorre um tempo limite, a máquina de estado muda para o 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**
>
> Você pode preservar a entrada de estado e o erro usando `ResultPath`. Consulte Use ResultPath para incluir erro e entrada em um `Catch`.

Invocar AWS Step Functions de outros serviços

Você pode configurar vários outros serviços para invocar máquinas de estado. Com base no tipo de fluxo de trabalho da máquina de estado, você pode invocar máquinas de estado de forma assíncrona ou síncrona. Para invocar máquinas de estado de maneira síncrona, use a chamada de API `StartSyncExecution` ou a integração do Amazon API Gateway com fluxos de trabalho expresso. Com a invocação assíncrona, o Step Functions pausa a execução do fluxo de trabalho até que um token de tarefa seja retornado. No entanto, a espera por um token de tarefa torna o fluxo de trabalho síncrono.

Os serviços que você pode configurar para invocar o Step Functions incluem:

- AWS Lambda, usando a chamada `StartExecution`.
- Amazon API Gateway
- Amazon EventBridge
- AWS CodePipeline
- Mecanismo de regras de AWS IoT
- AWS Step Functions

As invocações do Step Functions são governadas pela cota `StartExecution`. Para obter mais informações, consulte:

- Cotas

Consistência de leitura no Step Functions

As atualizações das máquinas de estado no AWS Step Functions são eventualmente consistentes. Todas as chamadas de `StartExecution` em um lapso de alguns segundos usarão a definição atualizada e `roleArn` (o Nome de recurso da Amazon para o perfil do IAM). Execuções que

iniciarem imediatamente após a chamada de `UpdateStateMachine` podem usar a definição de máquina de estado anterior e `roleArn`.

Para obter mais informações, consulte as informações a seguir.

- `UpdateStateMachine` na Referência de API do AWS Step Functions
- Atualizar um fluxo de trabalho em Começando com AWS Step Functions.

Marcação no Step Functions

O AWS Step Functions oferece suporte à marcação de máquinas de estado (padrão e expresso) e atividades. Pode ajudar a monitorar e gerenciar os custos associados aos recursos, além de fornecer melhor segurança nas políticas do AWS Identity and Access Management (IAM). A marcação de recursos do Step Functions permite que eles sejam gerenciados pelo AWS Resource Groups. Para obter mais informações sobre o Resource Groups, consulte o Guia do usuário do AWS Resource Groups.

Para autorização baseada em tags, os recursos de execução da máquina de estado, conforme mostrado no exemplo a seguir, herdam as tags associadas a uma máquina de estado.

```
arn:<partition>:states:<Region>:<account-id>:execution:<StateMachineName>:<ExecutionId>
```

Ao chamar DescribeExecution ou outras APIs nas quais você especifica o ARN do recurso de execução, o Step Functions usa tags associadas à máquina de estado para aceitar ou negar a solicitação enquanto realiza a autorização baseada em tags. Isso ajuda você a permitir ou negar acesso às execuções no nível da máquina de estado.

Para revisar as restrições relacionadas à marcação de recursos, consulte Restrições relacionadas à marcação.

Tópicos

- Marcação de alocação de custo
- Marcação para segurança
- Como visualizar e gerenciar tags no console do Step Functions
- Gerenciar tags com ações da API do Step Functions

Marcação de alocação de custo

Para organizar e identificar os recursos do Step Functions para alocação de custos, é possível adicionar tags de metadados que identificam um objetivo de uma atividade ou máquina de estado. Isso é especialmente útil quando você tem vários recursos. Você pode usar tags de alocação de custos para organizar sua fatura da AWS para refletir sua própria estrutura de custos. Para isso, cadastre-se para obter a fatura da sua conta da AWS para incluir as chaves e valores das tags. Para obter mais informações, consulte Configuração de um relatório de alocação de custos mensal no Manual do usuário do AWS Billing.

Por exemplo, é possível adicionar tags que representam o centro de custos e o objetivo dos recursos do Step Functions, conforme o seguinte.

Recurso	Chave	Valor
`StateMachine1`	`Cost Center`	`34567`
	`Application`	`Image processing`
`StateMachine2`	`Cost Center`	`34567`
	`Application`	`Rekognition processing`
`Activity1`	`Cost Center`	`12345`
	`Application`	`Legacy database`

Esse esquema de marcação permite que você agrupe duas máquinas de estado executando tarefas relacionadas no mesmo centro de custos e, ao mesmo tempo, marcar uma atividade não relacionada com outra tag de alocação de custos.

Marcação para segurança

O IAM oferece suporte para controlar o acesso a recursos com base em tags. Para controlar o acesso com base em tags, forneça informações sobre as tags de recurso no elemento de condição de uma política do IAM.

Por exemplo, é possível restringir o acesso a todos os recursos do Step Functions que incluam uma tag com a chave `environment` e o 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 obter mais informações, consulte Controlar o acesso usando etiquetas, no Guia do usuário do IAM.

Como visualizar e gerenciar tags no console do Step Functions

O Step Functions permite visualizar e gerenciar as tags para as máquinas de estado no console do Step Functions. Na página Details (Detalhes) de uma máquina de estado, selecione Tags. Aqui você pode visualizar as tags existentes associadas à máquina de estado.

> **Note**
>
> Para gerenciar tags para atividades, consulte Gerenciar tags com ações da API do Step Functions.

Para adicionar ou excluir tags associadas à máquina de estado, selecione o botão Manage Tags (Gerenciar tags).

1. Navegue até a página de detalhes de uma máquina de estado.

2. Selecione Tags ao lado de Executions (Execuções) e Definition (Definição).
3. Selecione Manage tags (Gerenciar tags).
   - Para modificar as tags existentes, edite a Chave e o Valor.
   - Para remover tags existentes, escolha Remove tag (Remover tag).
   - Para adicionar uma nova tag, escolha Add tag (Adicionar tag) e insira uma Key (Chave) e um Value (Valor).
4. Escolha Salvar.

Gerenciar tags com ações da API do Step Functions

Para gerenciar tags usando a API do Step Functions, use as seguintes ações de API:

- `ListTagsForResource`
- `TagResource`
- `UntagResource`

AWS Step Functions Studio de fluxo de trabalho

O Workflow Studio for AWS Step Functions é um designer de fluxo de trabalho visual de baixo código que permite criar fluxos de trabalho sem servidor orquestrando serviços. AWS Usando seu drag-and-drop recurso ou o editor de código integrado, você pode criar e editar fluxos de trabalho, controlar como a entrada e a saída são filtradas ou transformadas para cada estado e configurar o tratamento de erros. Conforme você arrasta e solta estados para criar o fluxo de trabalho, o Workflow Studio valida o trabalho e gera código automaticamente. Você pode revisar o código gerado ou atualizar a definição da máquina de estado no editor de código. Ao terminar, você pode salvar o fluxo de trabalho, executá-lo e examinar os resultados no console do Step Functions. É possível adicionar e modificar visualmente os fluxos de trabalho para orquestrar os vários serviços na aplicação.

Para usar o Step Functions Workflow Studio, você precisará de um Conta da AWS, e credenciais que forneçam as permissões corretas para quaisquer recursos que você queira usar. Para ter mais informações, consulte Pré-requisitos para começar a usar AWS Step Functions.

> **Note**
>
> O Workflow Studio não é compatível com o Internet Explorer 11. Se estiver usando o Internet Explorer 11 e encontrar problemas ao usar o Workflow Studio, tente usar um navegador diferente.

Você pode acessar o Workflow Studio pelo console do Step Functions ao criar ou editar um fluxo de trabalho no Step Functions. Também é possível acessar o Workflow Studio no AWS Application Composer. O Workflow Studio no Application Composer oferece um ambiente visual de IaC que facilita a incorporação de fluxos de trabalho às aplicações sem servidor criadas com ferramentas de IaC, como modelos do AWS CloudFormation. Usando o Workflow Studio no Application Composer, é possível criar fluxos de trabalho usando modelos do AWS CloudFormation. No Application Composer, é possível adicionar um novo fluxo de trabalho, modificar um fluxo de trabalho existente e conectar etapas individuais do fluxo de trabalho a outros recursos da aplicação. O Application Composer cria e atualiza automaticamente as configurações e os recursos do CloudFormation necessários. Isso ajuda você a criar e gerenciar todos os recursos usados nos fluxos de trabalho em um só lugar. Isso também ajuda você a acelerar o caminho desde a prototipagem do fluxo de trabalho até a implantação na produção.

Ao usar o Workflow Studio no Application Composer, também é possível se conectar diretamente ao projeto local. Isso ajuda você a trabalhar no ambiente de desenvolvimento integrado (IDE) junto com a tela visual. Para ter mais informações, consulte Usar o Workflow Studio no Application Composer.

Tópicos

Visão geral da interface

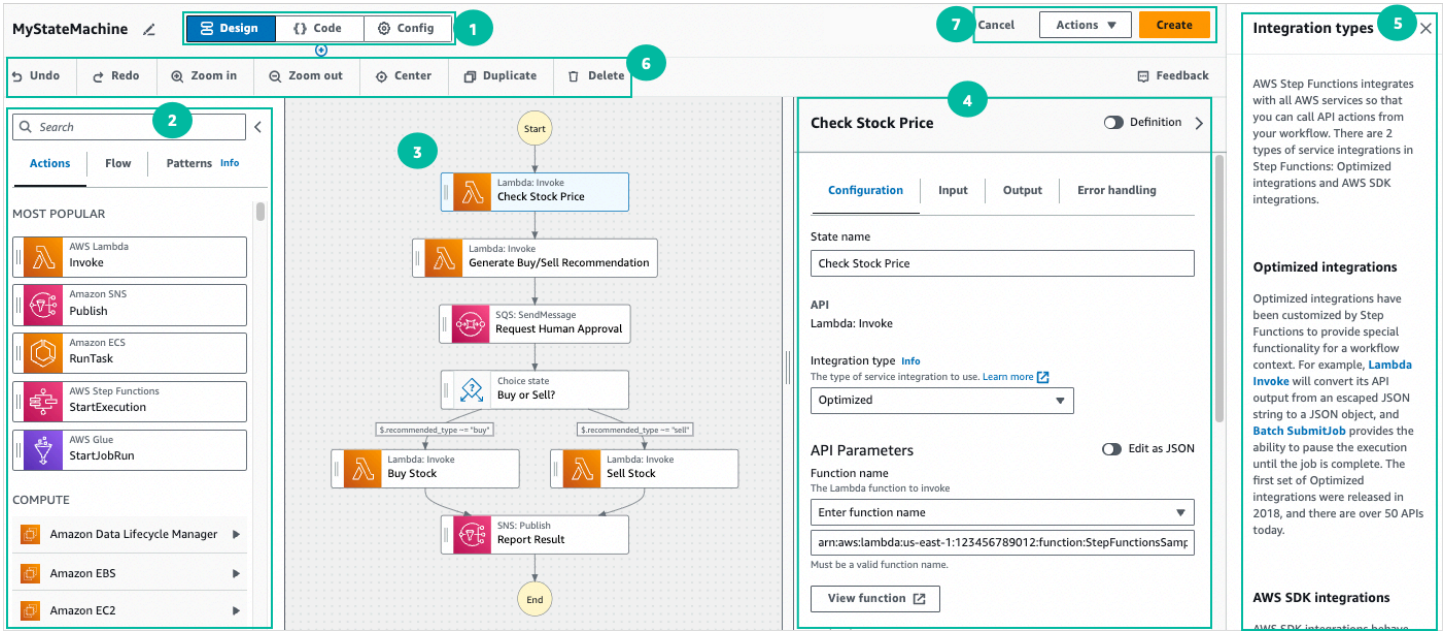
O Workflow Studio for AWS Step Functions é um designer de fluxo de trabalho visual de baixo código que permite criar fluxos de trabalho sem servidor por meio da orquestração. Serviços da AWS Com o recurso de arrastar e soltar, o Workflow Studio facilita a criação, a edição e a visualização dos protótipos de fluxo de trabalho. O Workflow Studio também oferece um editor de código integrado para escrever e editar definições de fluxo de trabalho usando Amazon States Language (ASL) no console do Step Functions.

Para ajudar você a criar e visualizar fluxos de trabalho, editar definições e gerenciar configurações, o Workflow Studio fornece três modos: Design, Código e Configuração. As seguintes seções descrevem esses casos em detalhes.

Neste tópico

Modo de design

O modo Design do Workflow Studio fornece uma interface gráfica para visualizar fluxos de trabalho à medida que você cria protótipos. A imagem a seguir mostra os diferentes componentes disponíveis no modo Design.

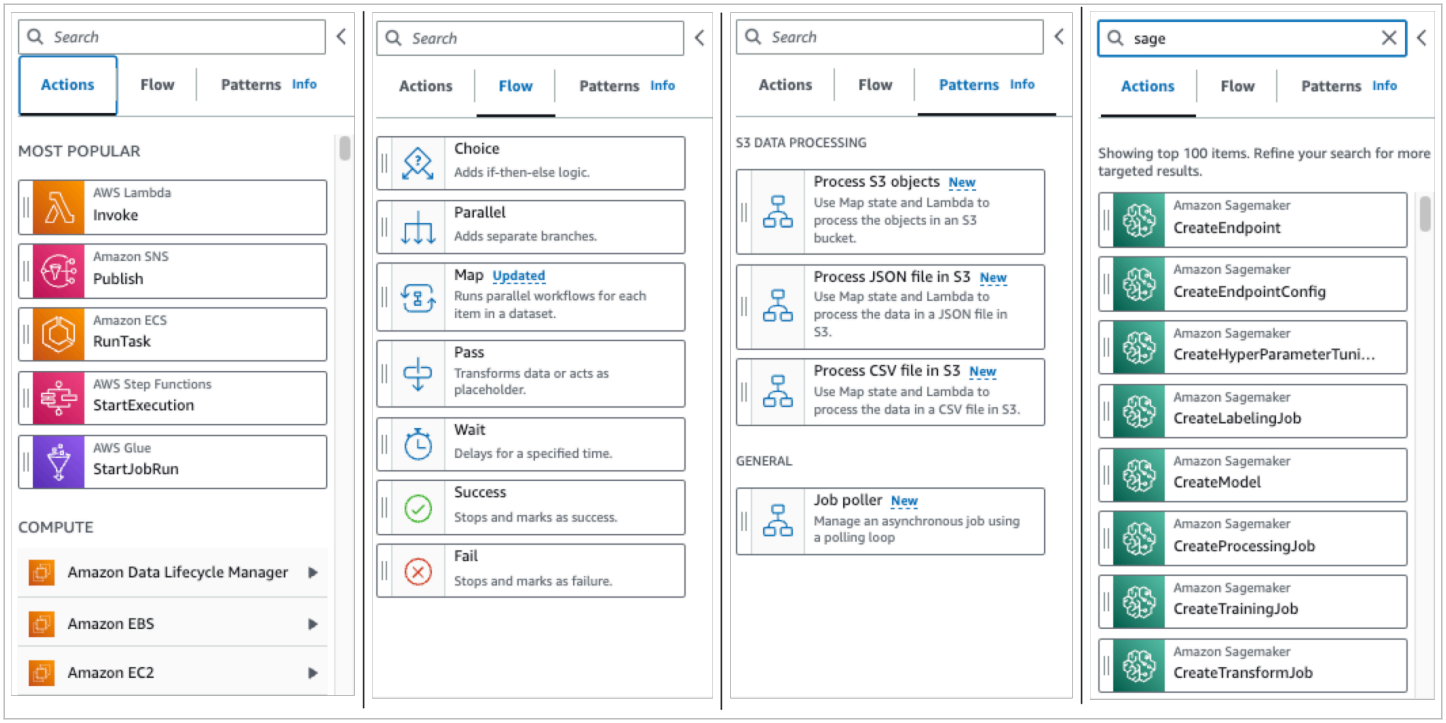


1. Botões de modo — alterne entre os modos Design, Código e Configuração do Workflow Studio usando os botões de modo. Você não pode alternar entre os modos se o JSON na definição de ASL do fluxo de trabalho for inválido.
2. O Navegador de estados contém as três guias seguintes:
   - A guia Ações fornece uma lista de AWS APIs que você pode arrastar e soltar no gráfico do fluxo de trabalho na tela. Cada ação representa um estado Estado da tarefa.
   - A guia Fluxo fornece uma lista de estados de fluxo que você pode arrastar e soltar no gráfico do fluxo de trabalho na tela.
   - A guia Padrões fornece vários ready-to-use blocos de construção reutilizáveis que você pode usar em diversos casos de uso. Por exemplo, você pode usar esses padrões para processar dados de forma iterativa em um bucket do Amazon S3.
3. O Canvas é onde você arrasta e solta estados no gráfico do fluxo de trabalho, altera a ordem dos estados e seleciona estados para configurar ou visualizar.
4. O painel Inspector é onde você pode visualizar e editar as propriedades de qualquer ação selecionada na tela. Ative o botão Definição para visualizar o código do Amazon States Language para o fluxo de trabalho, com o estado atualmente selecionado destacado.

5. Os links de Informações abrem um painel com informações contextuais quando você precisa de ajuda. Esses painéis também incluem links para tópicos relacionados na documentação do Step Functions.
6. Barra de ferramentas de design — contém um conjunto de botões para realizar ações comuns, como desfazer, excluir e ampliar.
7. Botões utilitários — um conjunto de botões para realizar tarefas, como salvar fluxos de trabalho ou exportar definições de ASL em um arquivo JSON ou YAML.

Navegador de estados

O navegador de estados é onde você seleciona os estados para arrastar e soltar no gráfico do fluxo de trabalho. A guia Ações fornece uma lista de AWS APIs e a guia Fluxo fornece uma lista dos estados de fluxo. Embora a guia Padrões forneça vários ready-to-use blocos de construção reutilizáveis que você pode usar em vários casos de uso. Você pode pesquisar todos os estados no Navegador de Estados usando a caixa de pesquisa na parte superior.



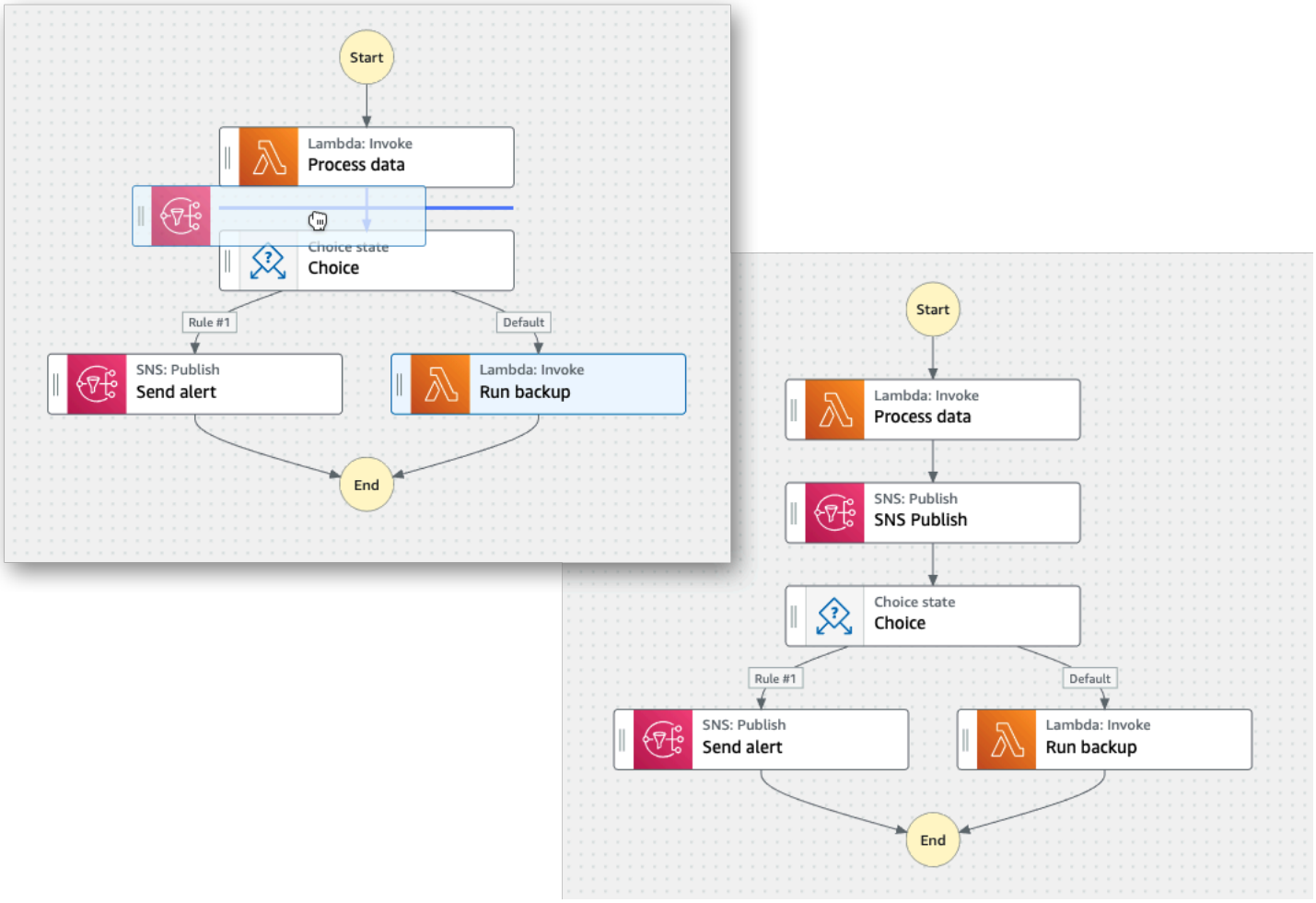
Há sete estados de fluxo que você pode usar para direcionar e controlar o fluxo de trabalho. Todos eles recebem entrada do estado anterior e muitos permitem filtrar a entrada do estado anterior e a saída para o estado seguinte. Os estados de fluxo são:

- Choice: adicione uma escolha entre ramificações de execução ao fluxo de trabalho. Na guia Configuração do Inspector, você pode configurar regras para determinar para qual estado o fluxo de trabalho fará a transição.
- Paralelo: adicione ramificações paralelas de execução ao fluxo de trabalho.
- Mapa: itere etapas dinamicamente para cada elemento de uma matriz de entrada. Ao contrário de um estado de fluxo `Parallel`, um estado `Map` executará as mesmas etapas para várias entradas de uma matriz na entrada de estado.
- Pass: permite que você passe a entrada para a saída. (Opcional) Você pode adicionar dados fixos na saída.
- Aguardar: faça com que o fluxo de trabalho seja pausado por um determinado período de tempo ou até uma hora ou data especificadas.
- Succeed: interrompe o fluxo de trabalho com sucesso.
- Fail: interrompe o fluxo de trabalho com uma falha.

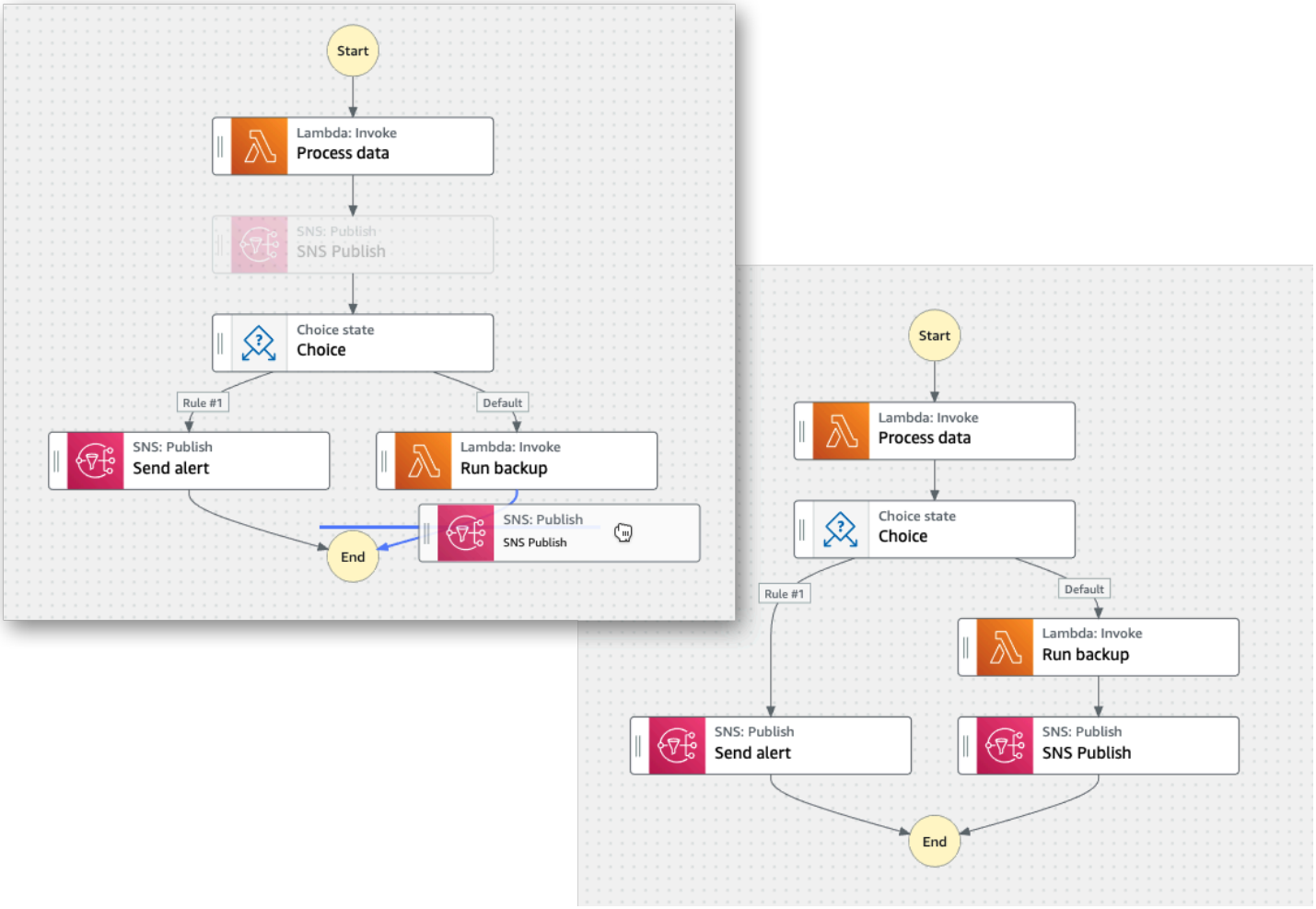
Canvas

Depois de escolher um estado para adicionar ao fluxo de trabalho, arraste-o para a tela e solte-o no gráfico do fluxo de trabalho. Você também pode arrastar e soltar estados para movê-los para lugares diferentes no fluxo de trabalho. Se o fluxo de trabalho for complexo, talvez não seja possível visualizar tudo no painel da tela. Use os controles na parte superior da tela para aumentar ou diminuir o zoom. Para visualizar diferentes partes de um gráfico do fluxo de trabalho, é possível arrastar o gráfico do fluxo de trabalho na tela.

Arraste um estado do fluxo de trabalho da guia Ações ou Fluxo e solte-o no fluxo de trabalho. Uma linha mostra onde ela será colocada no fluxo de trabalho. O novo estado do fluxo de trabalho foi adicionado ao fluxo de trabalho e o código foi gerado automaticamente.

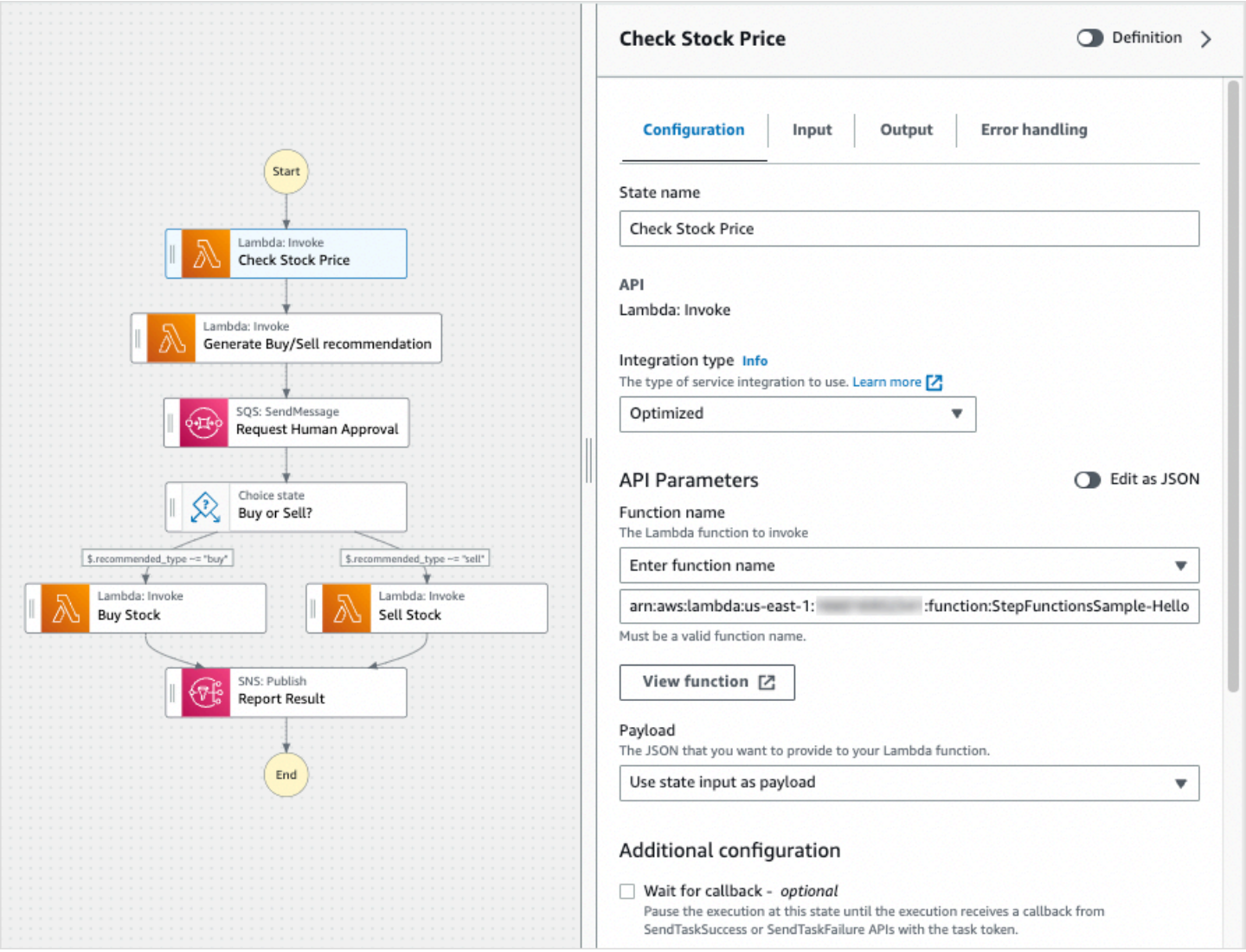


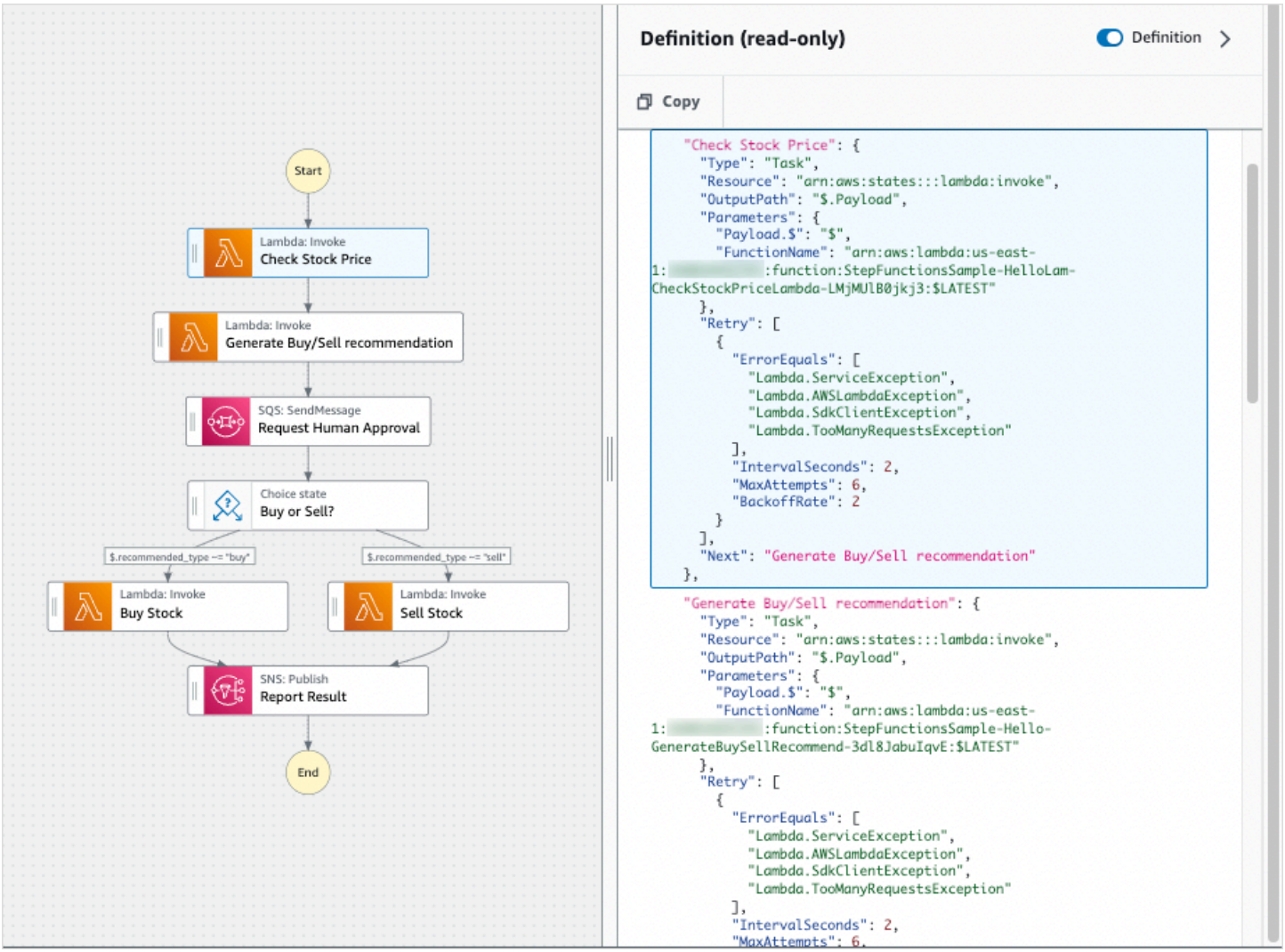
Para alterar a ordem de um estado, é possível arrastá-lo para um local diferente no fluxo de trabalho.



Inspector

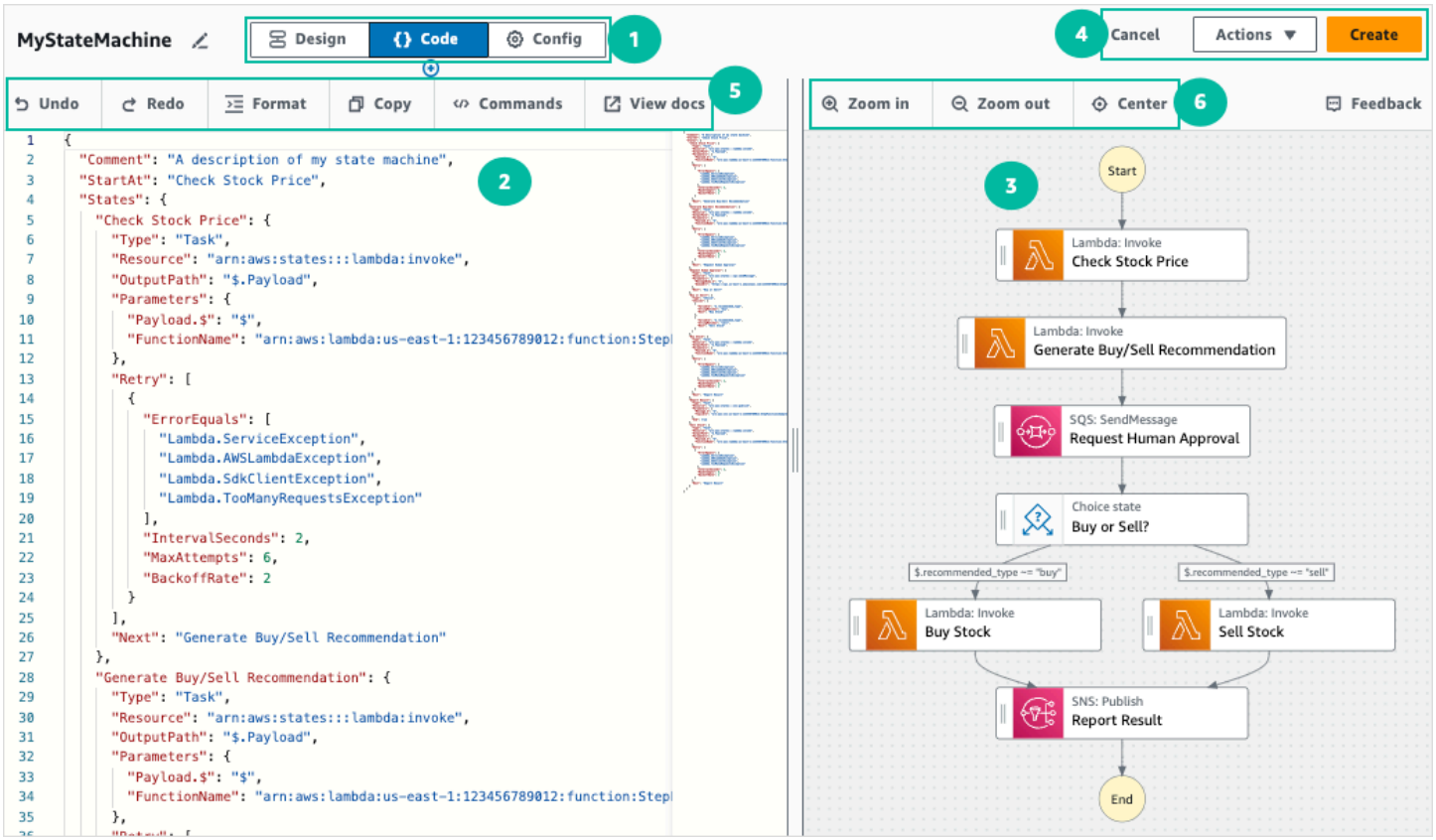
Você pode configurar qualquer estado que você adicionar ao fluxo de trabalho. Escolha o estado que deseja configurar e você verá opções de configuração no painel Inspector. Para ver a definição de ASL gerada automaticamente para o código de fluxo de trabalho, ative a opção Definição. A definição de ASL associada ao estado que você selecionou aparecerá destacada.


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 de código

O modo Código do Workflow Studio fornece um editor de código integrado para visualizar, escrever e editar a definição da Amazon States Language (ASL) dos fluxos de trabalho no console do Step Functions. A imagem a seguir mostra os diferentes componentes disponíveis no modo Código.



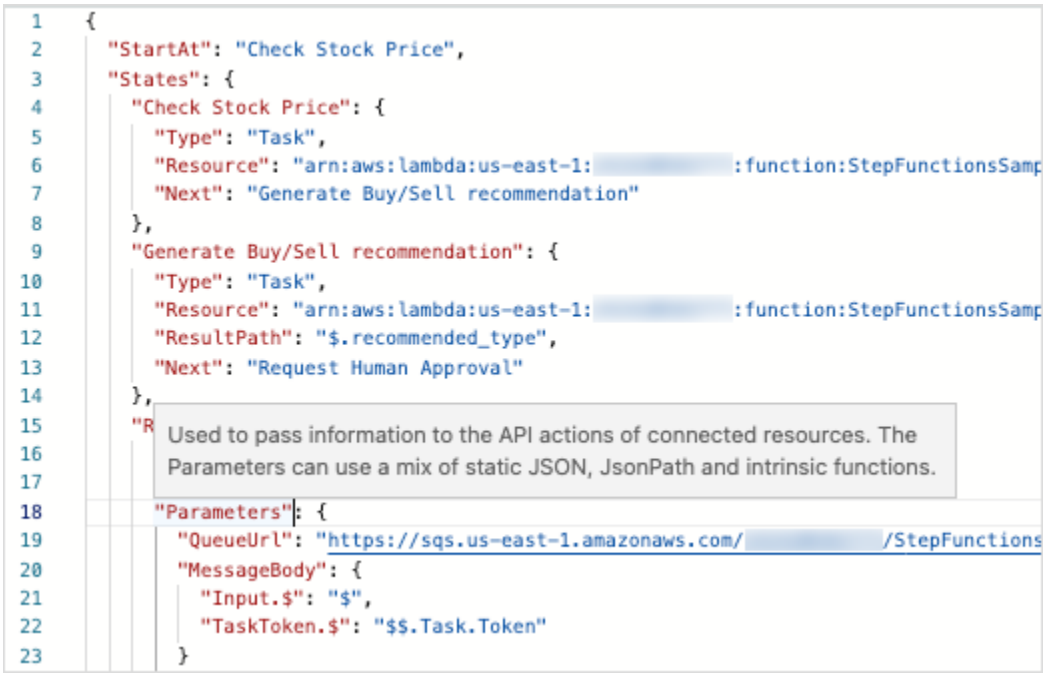
1. Botões de modo — alterne entre os modos Design, Código e Configuração do Workflow Studio usando os botões de modo. Você não pode alternar entre os modos se o JSON na definição de ASL do fluxo de trabalho for inválido.
2. O Editor de código é onde você escreve e edita a definição da ASL dos fluxos de trabalho no Workflow Studio. O editor de código também fornece recursos, como destaque de sintaxe e preenchimento automático.
3. Painel de visualização gráfica — Mostra uma visualização gráfica em tempo real do fluxo de trabalho.
4. Botões utilitários — um conjunto de botões para realizar tarefas, como salvar fluxos de trabalho ou exportar definições de ASL em um arquivo JSON ou YAML.
5. Barra de ferramentas de código — Contém um conjunto de botões para realizar ações comuns, como desfazer uma ação ou formatar o código.
6. Barra de ferramentas do gráfico — Contém um conjunto de botões para realizar ações comuns, como ampliar e reduzir o gráfico do fluxo de trabalho.

Editor de código

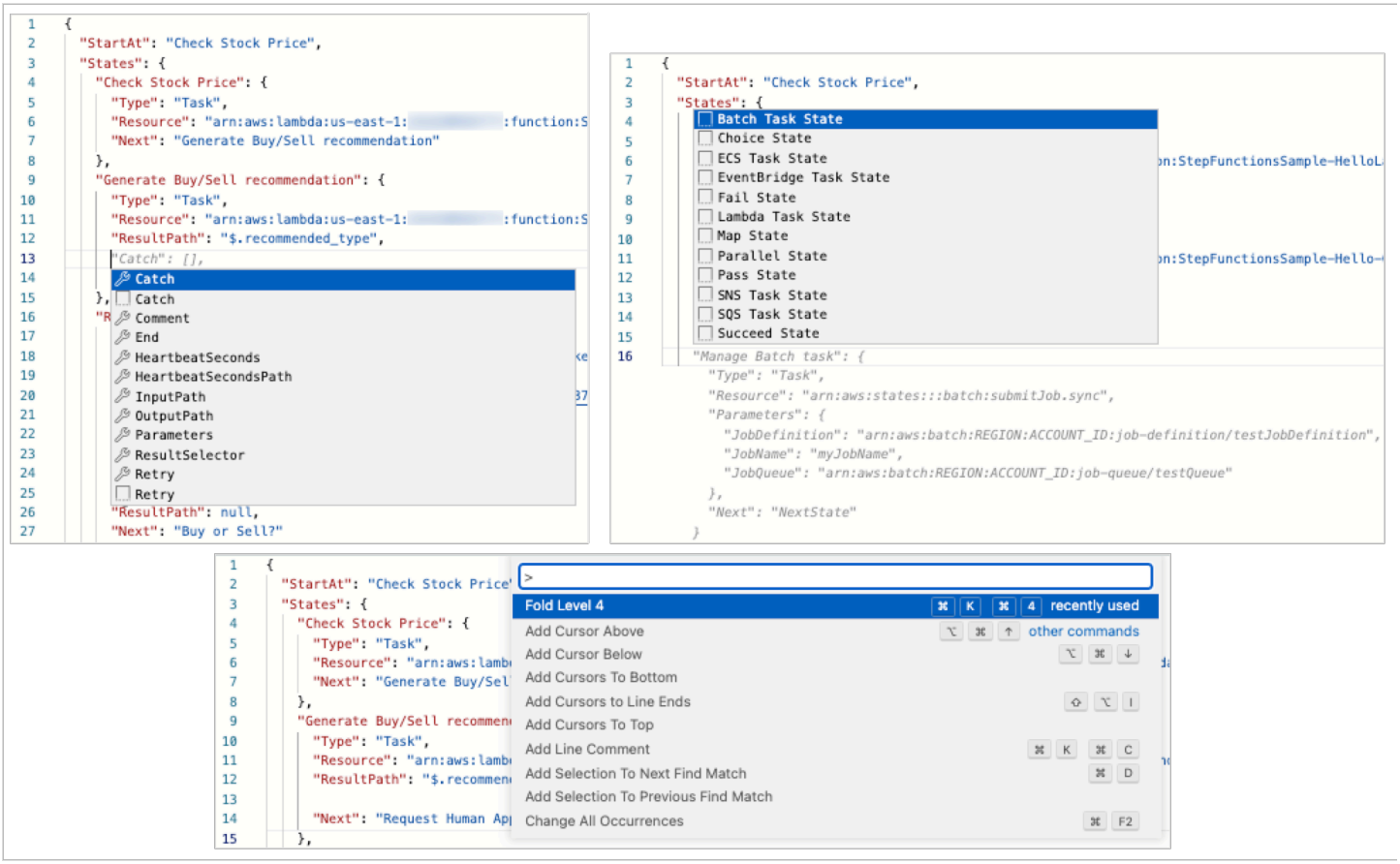
O editor de código fornece uma experiência semelhante à do IDE para escrever e editar definições de fluxo de trabalho usando JSON no Workflow Studio. O editor de código inclui vários recursos, como destaque de sintaxe, sugestões de preenchimento automático, validação de definição da ASL e exibição de ajuda contextual. Conforme você atualiza a definição do fluxo de trabalho, a Painel de visualização gráfica renderiza um gráfico em tempo real do fluxo de trabalho. Você também pode ver o gráfico de fluxo de trabalho atualizado no Modo de design.

Se você selecionar um estado no painel Modo de design ou no painel de visualização gráfica, a definição de ASL desse estado aparecerá destacada no editor de código. A definição de ASL do fluxo de trabalho é atualizada automaticamente se você reordenar, excluir ou adicionar um estado no modo Design ou no painel de visualização gráfica.

Coloque o cursor sobre qualquer campo na definição do fluxo de trabalho para ver a ajuda contextual como uma dica de ferramenta.



As sugestões de preenchimento automático exibem trechos de código dos campos ou estados que você pode incluir nos fluxos de trabalho. Para ver uma lista de campos que você pode incluir em um estado específico, pressione **`Ctrl+Space`**. Para gerar um trecho de código para um novo estado no fluxo de trabalho, pressione **`Ctrl+Space`** após a definição do estado atual. Você também pode pressionar **`F1`** para exibir uma lista de comandos disponíveis.



Painel de visualização gráfica

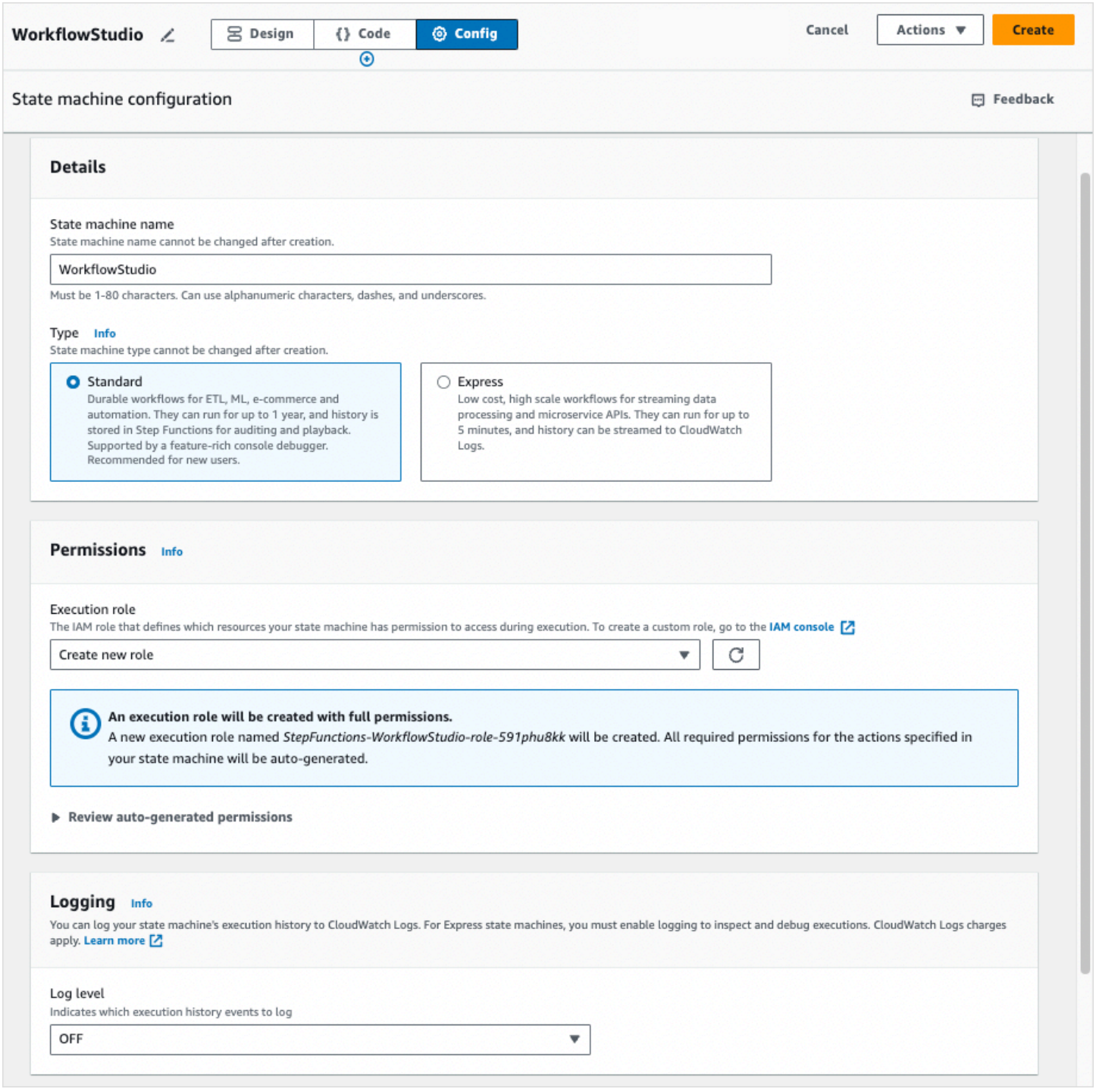
As visualizações de gráfico permitem que você veja a aparência do fluxo de trabalho no formato gráfico. Quando você escreve as definições de fluxo de trabalho no Editor de código do Workflow Studio, o painel de visualização gráfica renderiza um gráfico em tempo real do fluxo de trabalho. Conforme você reordena, exclui ou duplica um estado no painel de visualização gráfica, a definição do fluxo de trabalho no editor de código é atualizada automaticamente. Da mesma forma, à medida que você atualiza as definições de fluxo de trabalho, reordena, exclui ou adiciona um estado no editor de código, a visualização é atualizada automaticamente.

Se o JSON na definição da ASL do fluxo de trabalho for inválido, o painel de visualização gráfica pausará a renderização e exibirá uma mensagem de status na parte inferior do painel.

Modo de configuração

O modo Configuração do Workflow Studio permite gerenciar a configuração das máquinas de estado. Nesse modo, você pode especificar detalhes, como nome e tipo da máquina de estado, permissões do IAM e configuração de registro em log da máquina de estado. Outras configurações adicionais

que você pode especificar nesse modo incluem habilitar o AWS X-Ray rastreamento e publicar uma versão ao criar a máquina de estado. Depois de criar a máquina de estado, você pode editar todas as opções de configuração da máquina de estado, exceto o nome e o tipo da máquina de estado. A imagem a seguir mostra algumas das configurações que você pode especificar no modo Configuração.



Gerenciar a configuração da máquina de estado

Para gerenciar a configuração da máquina de estado, faça o seguinte:

1. Insira um nome para a máquina de estado na caixa Nome da máquina de estado.

> **Tip**
>
> Como alternativa, escolha o ícone de edição ao lado do nome padrão da máquina de estado de MyStateMachine. Em Configuração da máquina de estado, especifique um nome.

> **Important**
>
> Você não pode editar o nome da máquina de estado depois de criá-la.

2. Em Tipo, escolha um tipo de máquina de estado Padrão ou Expresso. Para obter mais informações sobre tipos de máquinas de estado, consulte Comparação entre os fluxos de trabalho padrão e expresso.

> **Important**
>
> Você não pode editar o tipo da máquina de estado depois de criá-la.

3. Em Permissões, selecione o perfil do IAM a ser usado como perfil de execução para a máquina de estado.
   - Criar novo perfil (recomendado): se você selecionar essa opção, o Step Functions criará automaticamente um perfil de execução para as máquinas de estado com os privilégios mínimos necessários ao criar as máquinas de estado. Essas funções do IAM geradas automaticamente são válidas para Região da AWS as quais você cria a máquina de estado.

> **Tip**
>
> Para revisar as permissões que o Step Functions gerará automaticamente para a máquina de estado, escolha Revisar permissões geradas automaticamente.

> **Note**
>
> Se você excluir o perfil do IAM criado pelo Step Functions, não será possível recriá-lo posteriormente. Da mesma forma, se você modificar a função (por exemplo, removendo o Step Functions das entidades principais na política do IAM), o Step Functions não poderá restaurar as configurações originais dela posteriormente.

- Escolha um perfil existente: crie o próprio perfil do IAM para a máquina de estado e, em seguida, escolha-o entre as opções listadas abaixo Escolha um perfil existente. Certifique-se de que a política do perfil inclua as permissões que você gostaria que a máquina de estado assumisse.

  Para obter informações sobre a criação de políticas do IAM, consulte Criação de políticas do IAM, no Manual do usuário do IAM.

- Inserir um ARN de função: especifique o nome do recurso da Amazon (ARN) de um perfil do IAM existente a ser usado para essa máquina de estado. Por exemplo, `arn:aws:iam::123456789012:role/service-role/StepFunctions-WorkflowStudio-role-777f4027`.

4. Em Registro em log, defina o nível de log para a máquina de estado. O Step Functions registra os eventos do histórico de execução com base na seleção. Você pode selecionar uma das seguintes opções:

   - TODOS: todos os tipos de eventos são registrados.
   - ERRO: Todos os tipos de eventos de erro são registrados, como TaskFailed e. ExecutionFailed
   - FATAL: Todos os tipos de eventos de erro fatal são registrados, como ExecutionAborted e. ExecutionFailed
   - DESLIGADO: nenhum tipo de evento é registrado.

   Para mais informações sobre níveis de log, consulte Níveis de log.

5. Em Configuração adicional, defina uma ou mais das seguintes configurações opcionais:

   - Ativar rastreamento de X-Ray: escolha essa caixa de seleção para que o Step Functions envie rastreamentos de X-Ray para execuções de máquinas de estado, mesmo quando um ID de rastreamento não for passado por um serviço upstream. Para ter mais informações, consulte AWS X-Ray e Step Functions.

- Publicar versão na criação: uma versão é um snapshot numerado e imutável de uma máquina de estado que você pode executar. Escolha essa caixa de seleção para publicar uma versão da máquina de estado ao criar a máquina de estado. O Step Functions publica a versão 1 como a primeira revisão da máquina de estado.

  Para obter mais informações sobre versões, consulte Versões de máquina de estado.
- Adicionar nova tag: escolha esta caixa para adicionar tags à máquina de estado. Adicionar tags pode ajudar a monitorar e gerenciar os custos associados aos recursos, além de fornecer melhor segurança nas políticas do IAM. Para obter mais informações sobre tags, consulte Marcação no Step Functions.

6. Escolha Criar.
7. Na caixa de diálogo Confirmar criação do perfil, selecione Confirmar para continuar.

   Você também pode clicar em Exibir configurações da função para retornar ao modo Configurar.

Atalhos de teclado

O Workflow Studio dá suporte aos seguintes atalhos de teclado:

Atalho de teclado	Função
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

Atalho de teclado	Função
Ctrl+D	Duplicate selected state

Como usar o Workflow Studio

Aprenda a criar, editar e executar fluxos de trabalho usando o Step Functions Workflow Studio. Depois que o fluxo de trabalho estiver pronto, você poderá exportá-lo. Você também pode usar o Workflow Studio para prototipagem rápida.

Neste tópico

Criar um fluxo de trabalho

No Workflow Studio, você pode escolher um modelo inicial ou um modelo em branco para criar um fluxo de trabalho do zero. Para modelos em branco, você pode usar o modo Design ou Código para criar o fluxo de trabalho.

Um modelo inicial é um projeto de ready-to-run amostra que cria automaticamente o protótipo e a definição do fluxo de trabalho e implanta todos os AWS recursos relacionados que seu projeto precisa para você. Conta da AWS Você pode usar esses projetos iniciais para implantá-los e executá-los como estão ou usar os protótipos de fluxo de trabalho para se basear neles. Para obter mais informações sobre os modelos iniciais, consulte Projetos de amostra para Step Functions.

Crie um fluxo de trabalho usando modelos iniciais

1. Abra o console do Step Functions e clique em Criar máquina de estado.
2. Na caixa de diálogo Escolher um modelo, faça o seguinte para escolher um projeto de amostra, por exemplo, o projeto de amostra Task Timer:

- Digite **`Task Timer`** na caixa Pesquisa por palavra-chave e escolha Task Timer nos resultados da pesquisa que são retornados.
- Navegue pelos exemplos de projetos listados em Tudo no painel direito e escolha Task Timer.

3. Escolha Próximo para continuar.
4. Step Functions lista o Serviços da AWS usado no projeto de amostra que você selecionou. Também mostra um gráfico de fluxo de trabalho para o projeto de amostra. Implante esse projeto no seu Conta da AWS ou use-o como ponto de partida para criar seus próprios projetos. Com base em como você deseja prosseguir, escolha Executar uma demonstração ou Criar com base nela.
5. Escolha Usar modelo para continuar com a seleção.
6. Execute um destes procedimentos:

   - Se você selecionou Criar com base nela, o Step Functions criará o protótipo do fluxo de trabalho para o projeto de amostra selecionado. O Step Functions não implanta os recursos listados na definição do fluxo de trabalho. No Modo de design, arraste e solte os estados dos Navegador de estados para continuar criando o protótipo de fluxo de trabalho. Ou mude para o Modo de código para atualizar a definição da Amazon States Language (ASL) do fluxo de trabalho.

     > **Important**
     >
     > Lembre-se de atualizar o espaço reservado do nome do recurso da Amazon (ARN) para os recursos usados no projeto de amostra antes de executar o fluxo de trabalho.

   - Se você selecionou Executar uma demonstração, o Step Functions cria um projeto de amostra somente para leitura que usa um AWS CloudFormation modelo para implantar os AWS recursos listados nesse modelo no seu. Conta da AWS

     > **Tip**
     >
     > Para visualizar a definição da máquina de estado do projeto de amostra, escolha Código.

     Quando estiver pronto, escolha Implemente e execute para implantar o projeto de amostra e criar os recursos.

> **Note**
>
> Pode levar até 10 minutos para que esses recursos e as permissões relacionadas ao IAM sejam criados. Enquanto seus recursos estão sendo implantados, você pode abrir o link do CloudFormation Stack ID para ver quais recursos estão sendo provisionados.

> **Important**
>
> Cobranças padrão se aplicam a cada serviço usado no CloudFormation modelo.

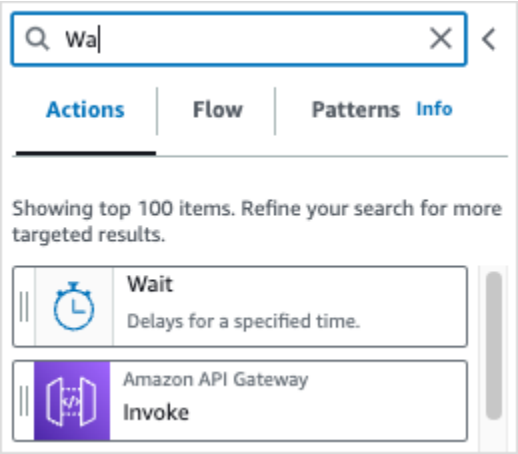
Crie um fluxo de trabalho usando um modelo em branco

1. Abra o console do Step Functions.
2. Escolha Criar uma máquina de estado.
3. Na caixa de diálogo Escolher um modelo, selecione Em branco.
4. Escolha Selecionar. Isso abre o Workflow Studio no Modo de design.

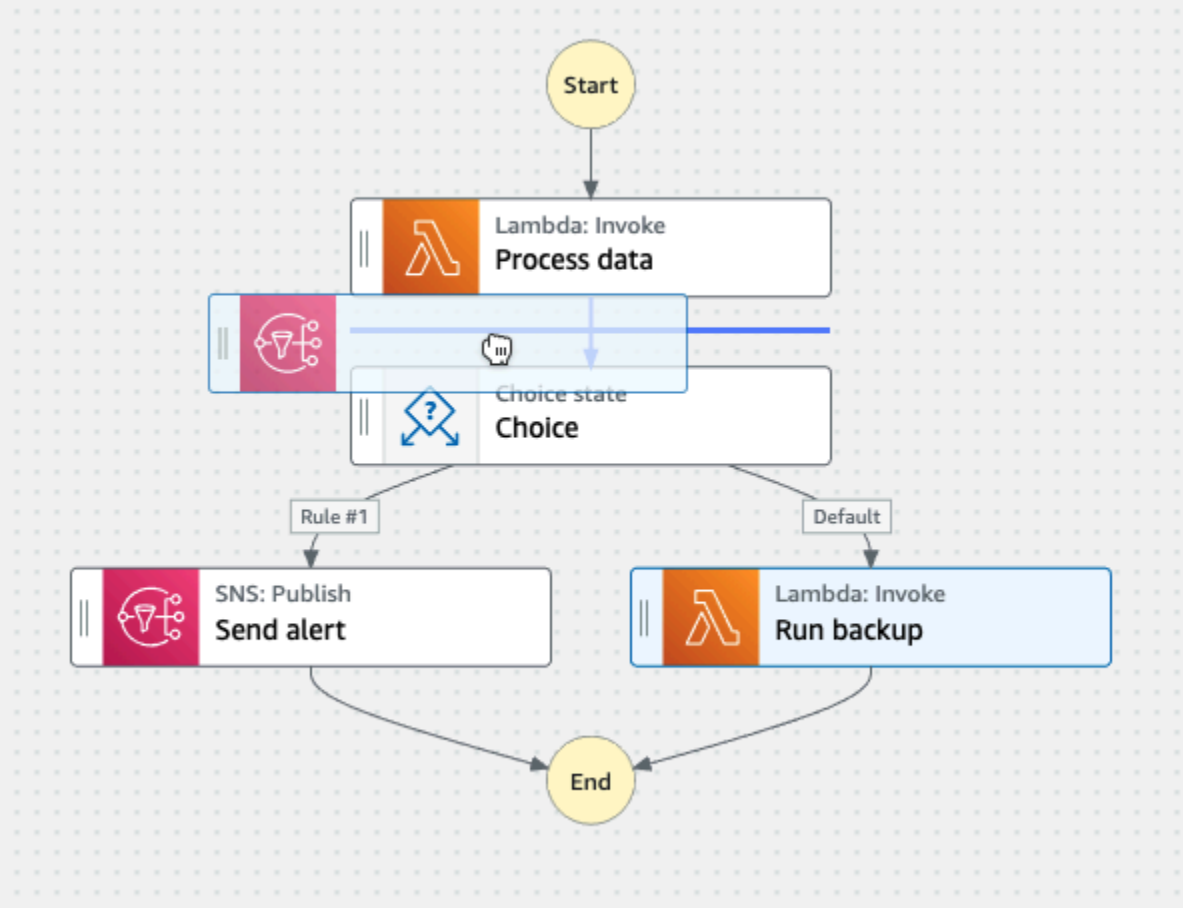
   Agora você pode começar a projetar o fluxo de trabalho no Modo de design ou escrever a definição de fluxo de trabalho no Modo de código.
5. Escolha Configuração para gerenciar a configuração do fluxo de trabalho no Modo de configuração. Por exemplo, forneça um nome para o fluxo de trabalho e escolha o tipo.

Projetar um fluxo de trabalho

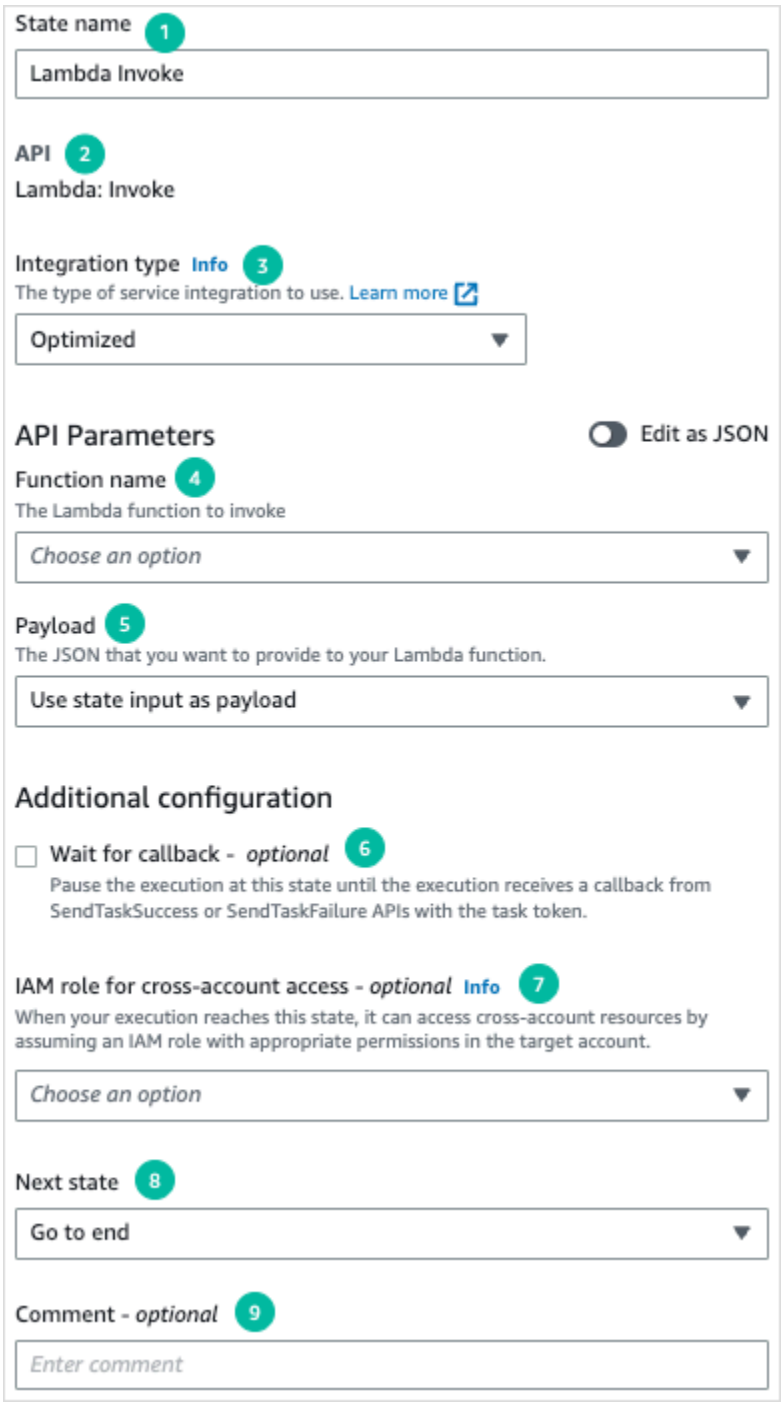
Se você souber o nome do estado que deseja adicionar, use a caixa de pesquisa na parte superior dos Navegador de estados para encontrar esse estado nas guias Ações e Fluxo do Modo de design.



Caso contrário, escolha um estado no navegador de estados e arraste-o e solte-o na tela, colocando-o onde quiser no fluxo de trabalho. Você também pode reordenar estados no fluxo de trabalho arrastando-os para um local diferente no fluxo de trabalho. Quando você arrasta um estado até a tela, uma linha aparece nos locais em que é possível soltá-lo no fluxo de trabalho. Depois que um estado é colocado na tela, o código é gerado automaticamente e adicionado à definição de fluxo de trabalho. Para ver a definição, ative o botão Definição no painel Inspector. Para fazer edições na definição do fluxo de trabalho, escolha o Modo de código que oferece um editor de código integrado.



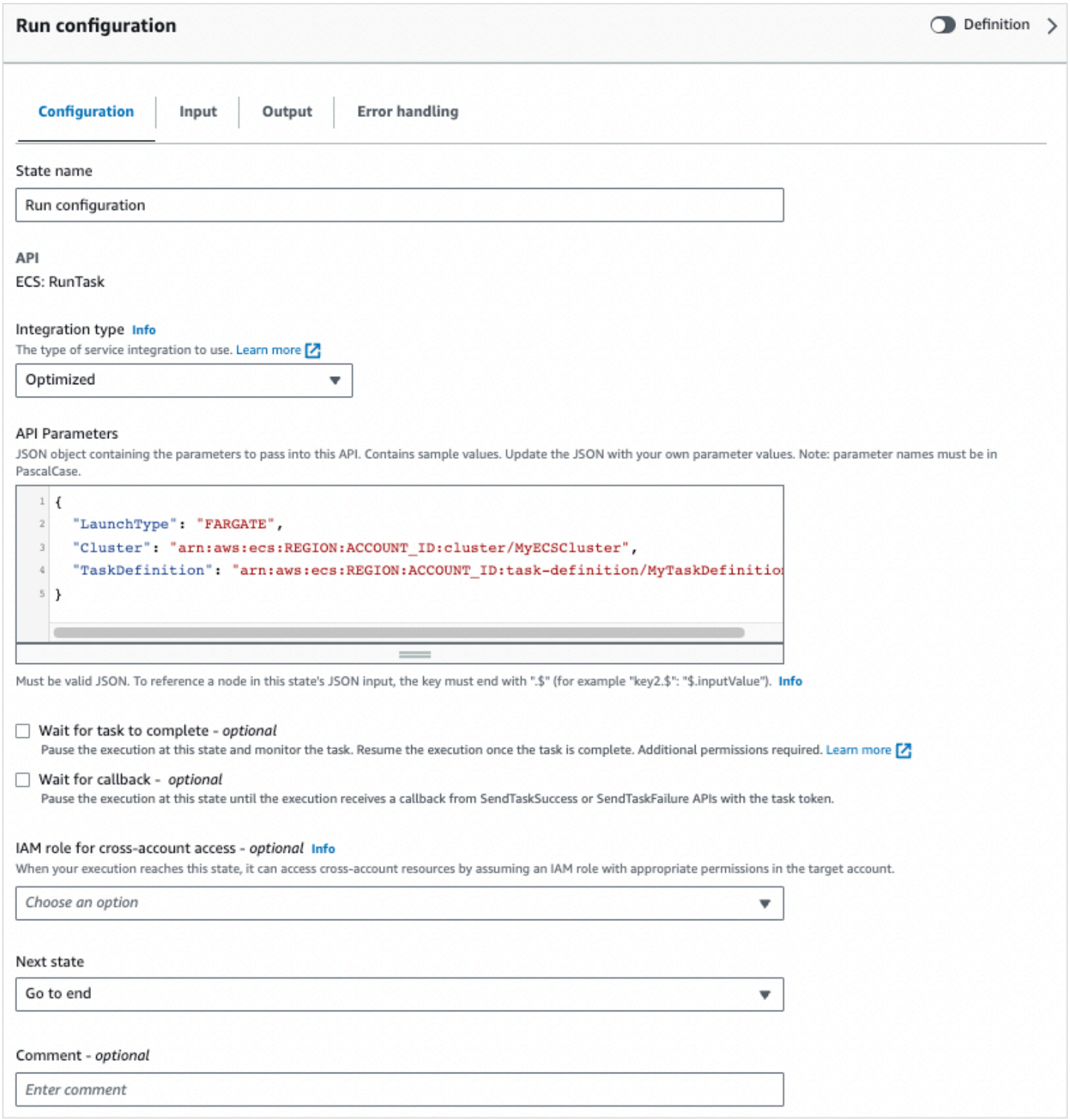
Depois de soltar um estado na tela, configure-o no painel Inspector à direita. Esse painel contém as guias Configuração, Entrada, Saída e Tratamento de erros para cada estado ou ação de API que você coloca na tela. Você configura os estados incluídos nos fluxos de trabalho na guia Configuração. Por exemplo, a guia Configuração da ação da API Invocação Lambda consiste nas seguintes opções:



1. O Nome do estado identifica o estado. Você pode usar o próprio nome ou aceitar o nome padrão gerado.
2. A API mostra a ação da API usada pelo estado.
3. A lista suspensa Tipo de integração fornece opções para escolher o tipo de integração de serviços disponíveis no Step Functions. O tipo de integração que você escolher é usado para chamar ações de API específicas AWS service (Serviço da AWS) do seu fluxo de trabalho.
4. O Nome da função fornece opções para:
   - Inserir o nome da função: você pode inserir o nome da função ou o ARN.
   - Obter nome da função em runtime a partir da entrada de estado: você pode usar essa opção para obter dinamicamente o nome da função da entrada de estado com base no caminho especificado.
   - Selecionar o nome da função: você pode selecionar diretamente das funções disponíveis na conta e região.
5. O Payload permite que você selecione entre as seguintes opções:
   - Usar a entrada de estado como payload: você pode usar essa opção para passar a entrada do estado como a payload fornecida à função do Lambda.
   - Inserir a própria payload: você pode usar essa opção para construir um objeto JSON para passar como payload para a função do Lambda. Esse JSON pode incluir valores estáticos e valores selecionados na entrada de estado.
   - Sem payload: você pode usar essa opção se não quiser passar nenhuma payload para a função do Lambda.
6. (Opcional) Alguns estados terão a opção de selecionar Aguardar a conclusão da tarefa ou Aguardar o retorno da chamada. Quando disponíveis, essas opções selecionam um dos seguintes padrões de integração de serviços:
   - Nenhuma opção selecionada: o Step Functions usará o padrão de integração Resposta de solicitação. O Step Functions aguardará uma resposta HTTP e, em seguida, avançará para o próximo estado. O Step Functions não esperará a conclusão de um trabalho. Quando nenhuma opção estiver disponível, o estado usará esse padrão.
   - Aguardar a conclusão da tarefa: o Step Functions usará o padrão de integração Executar um trabalho (.sync).
   - Aguardar o retorno de chamada: o Step Functions usará o padrão de integração Aguardar um retorno de chamada com um token de tarefa.

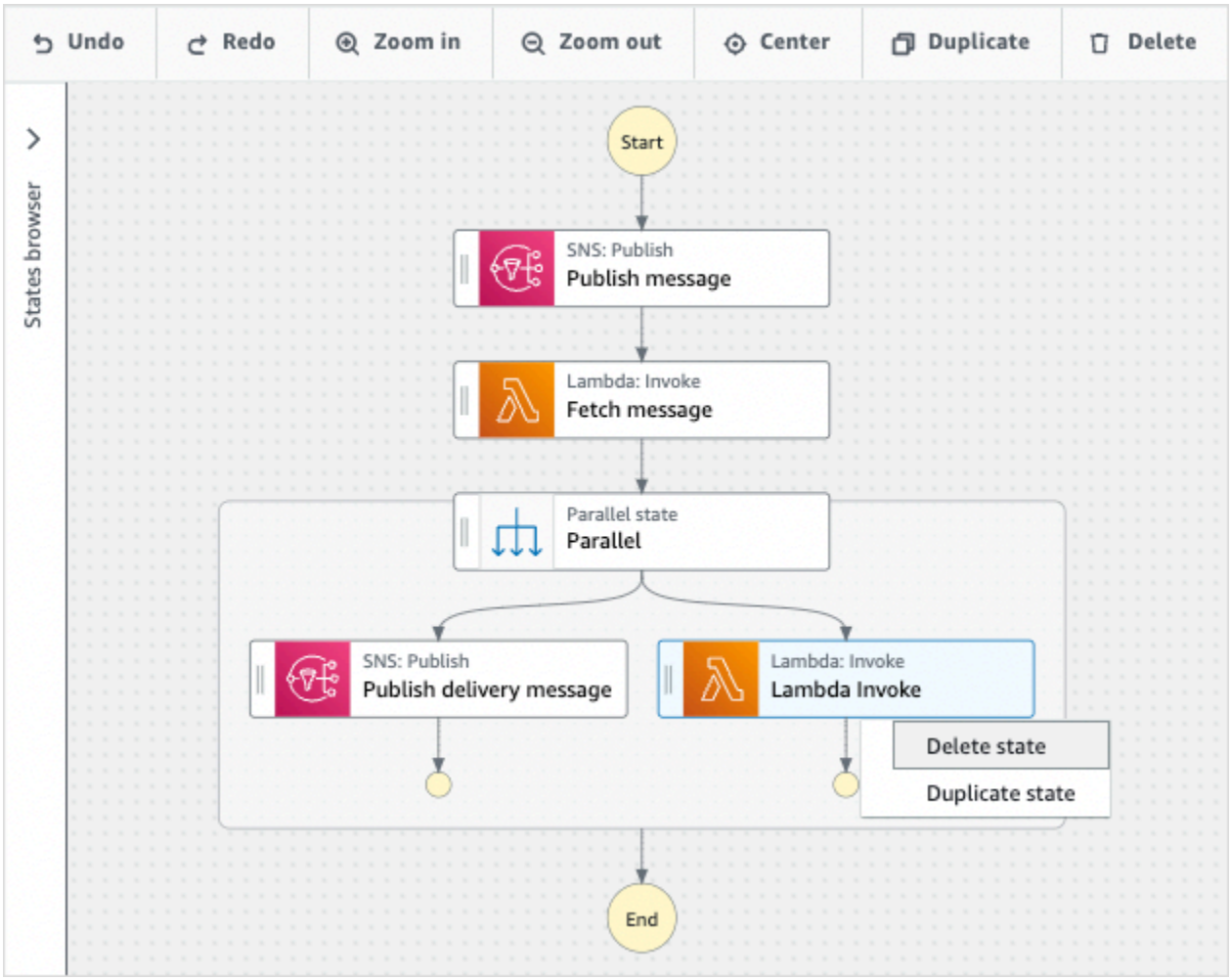
7. (Opcional) Para acessar recursos configurados de forma diferente Contas da AWS em seus fluxos de trabalho, o Step Functions fornece acesso entre contas. O perfil do IAM para acesso entre contas fornece opções para:
   - Fornecer o ARN do perfil do IAM: especifique o perfil do IAM que contém as permissões apropriadas de acesso aos recursos. Esses recursos estão disponíveis em uma conta de destino, Conta da AWS para a qual você faz chamadas entre contas.
   - Obtenha o ARN do perfil do IAM em runtime a partir da entrada do estado: especifique um caminho de referência para um par de valores-chave existente na entrada JSON do estado que contém o perfil do IAM.
8. O próximo estado permite que você selecione o próximo estado para o qual deseja fazer a transição.
9. (Opcional) O campo Comentário pode ser usado para adicionar seu próprio comentário. Isso não afetará o fluxo de trabalho, mas pode ser usado para anotar o fluxo de trabalho.

Alguns estados terão opções de configuração mais genéricas. Por exemplo, a configuração de estado `RunTask` do Amazon ECS contém um campo `API Parameters` preenchido com valores de espaço reservado.



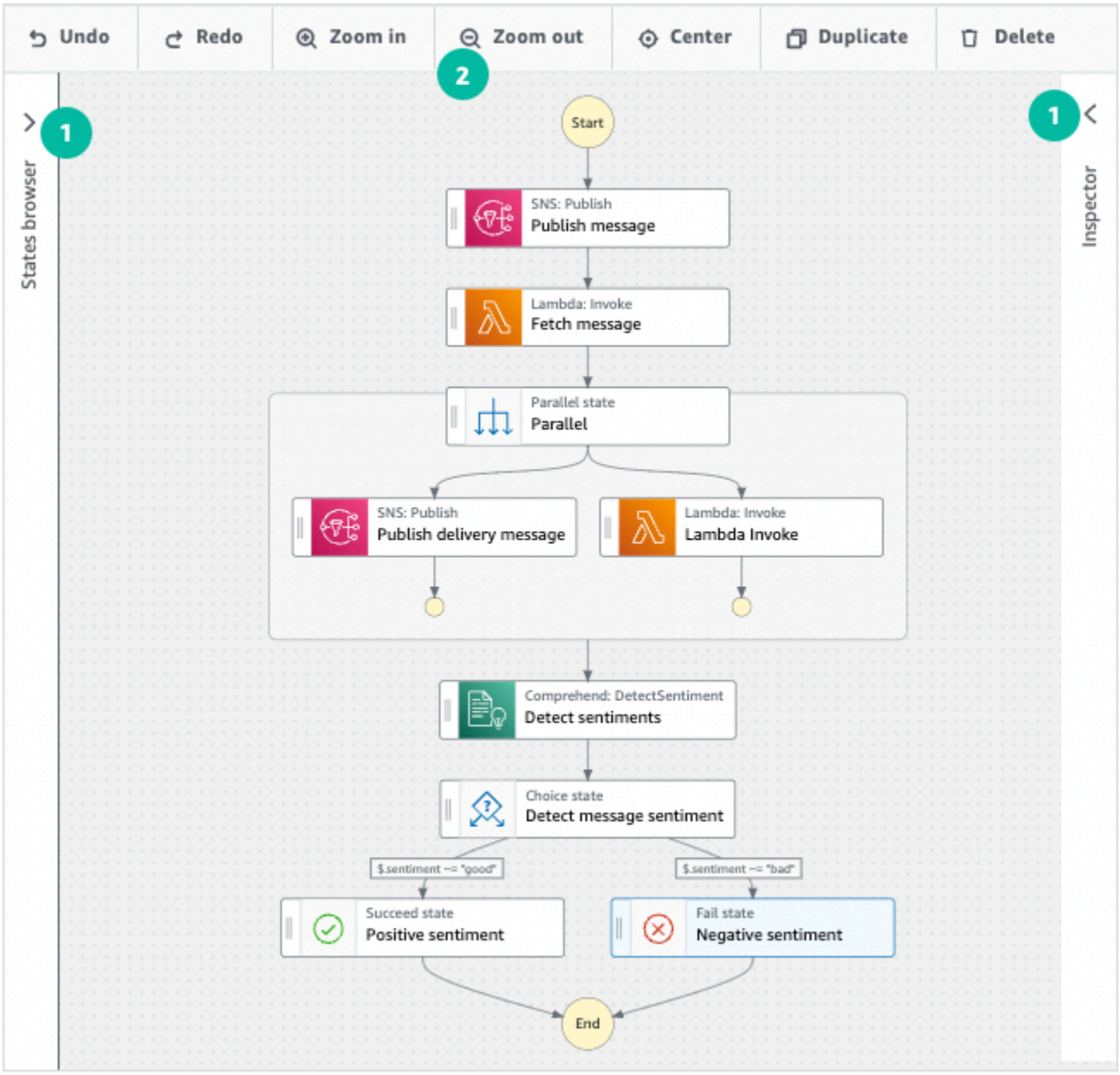
Para esses estados, você pode substituir os valores do espaço reservado por configurações adequadas às suas necessidades.

Para excluir um estado, você pode usar o backspace, clicar com o botão direito do mouse e escolher Excluir estado ou escolher Excluir na barra de ferramentas Design.



À medida que o fluxo de trabalho cresce, ele pode não caber na tela. É possível:

1. Usar os controles nos painéis laterais para redimensionar ou fechar os painéis.
2. Usar a barra de ferramentas Design na parte superior do Canvas para ampliar ou reduzir o gráfico do fluxo de trabalho.



Executar o fluxo de trabalho

Depois de criar ou editar o fluxo de trabalho com o Workflow Studio, você pode executá-lo e visualizar a execução no console do Step Functions.

Para executar um fluxo de trabalho no Workflow Studio

1. No modo Design, Código ou Configuração, escolha Executar.

   A caixa de diálogo Iniciar execução é aberta em uma nova guia.
2. Na caixa de diálogo Iniciar execução, faça o seguinte:

1. (Opcional) Para identificar a execução, insira um nome para ela no campo Nome. Por padrão, o Step Functions cria automaticamente um nome de execução exclusivo.

   > **Note**
   >
   > Step Functions permite criar nomes para máquinas de estado, execuções e atividades, além de rótulos que contenham caracteres não ASCII. Esses nomes não ASCII não funcionam com a Amazon. CloudWatch Para garantir que você possa acompanhar CloudWatch as métricas, escolha um nome que use somente caracteres ASCII.

2. (Opcional) No campo Entrada, insira os valores de entrada no formato JSON para executar o fluxo de trabalho.
3. Selecione Iniciar execução.
4. O console do Step Functions direciona você para uma página em que o título é o ID da execução. Essa página é conhecida como página de Detalhes da execução. Nesta página, você pode revisar os resultados da execução à medida que a execução avança ou após a conclusão.

   Para revisar os resultados da execução, escolha estados individuais na Exibição em gráfico e, em seguida, escolha as guias individuais no painel Detalhes da etapa para visualizar os detalhes de cada estado, incluindo entrada, saída e definição, respectivamente. Para obter detalhes sobre as informações de execução que você pode visualizar na página Detalhes da execução, consulte Página de Detalhes da execução — Visão geral da interface.

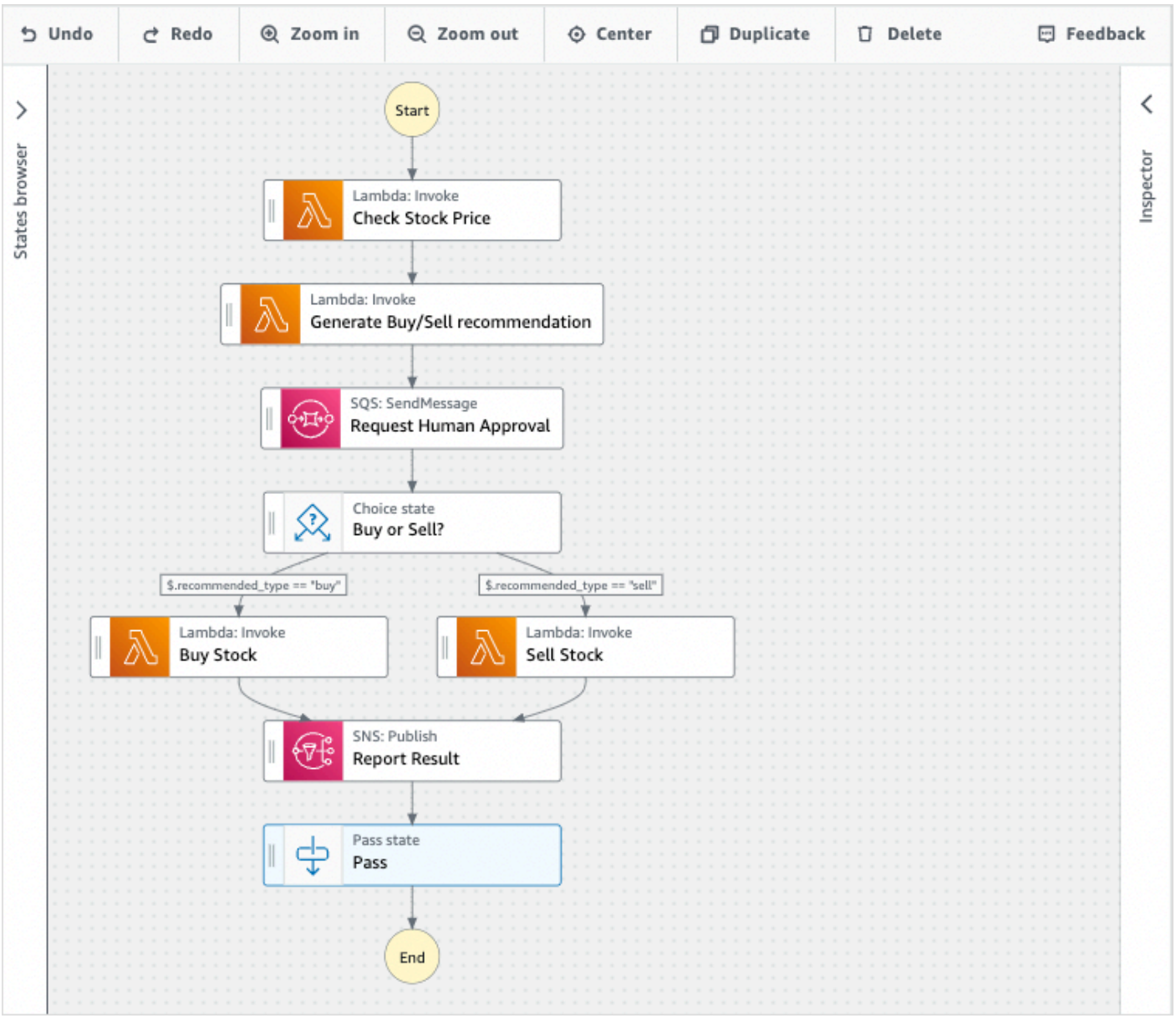
Editar o fluxo de trabalho

Você pode editar visualmente um fluxo de trabalho existente no Modo de design do Workflow Studio. Você também pode editar a definição de fluxo de trabalho no Modo de código do Workflow Studio.

Para editar um fluxo de trabalho existente:

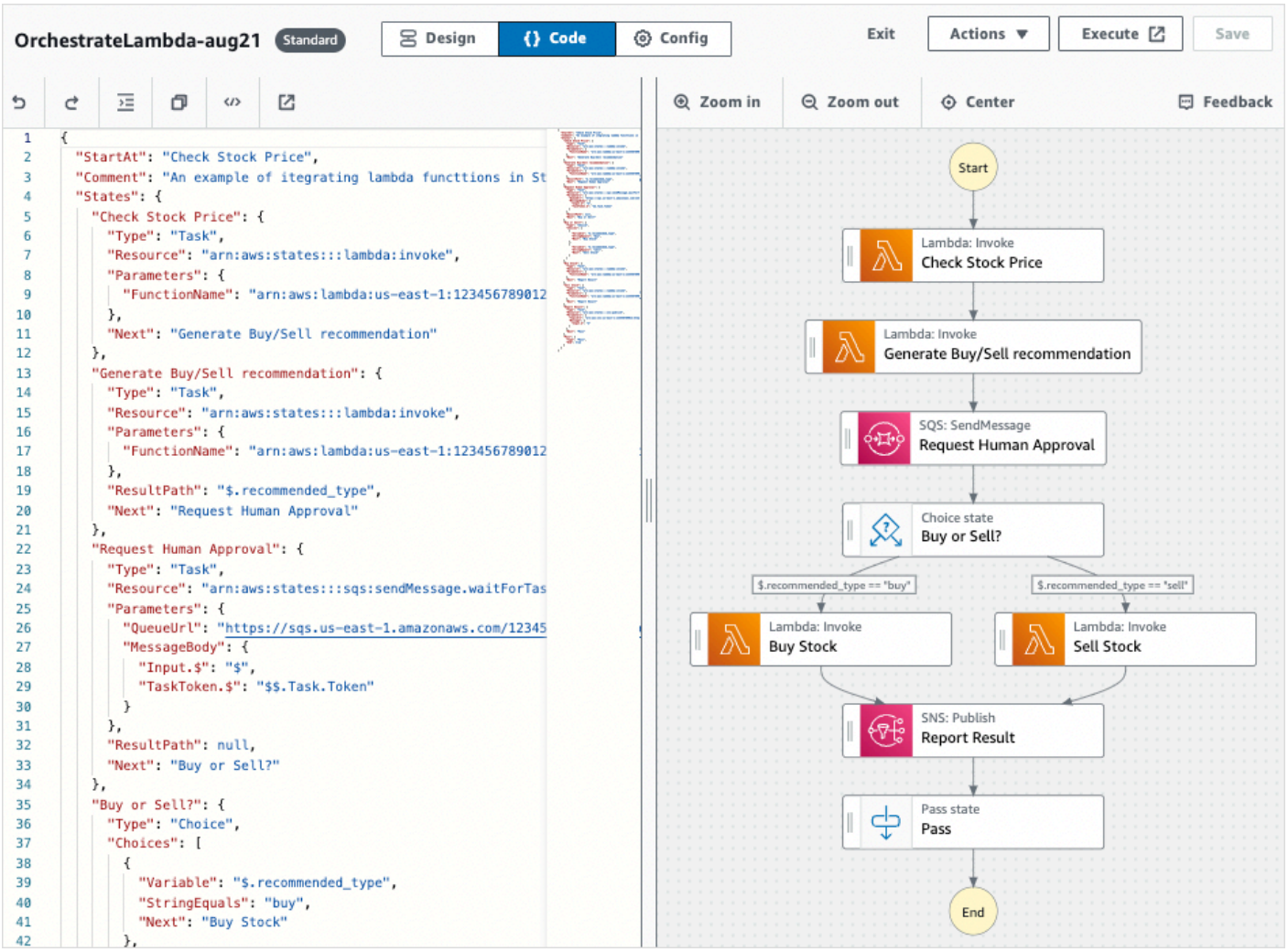
1. Abra o console do Step Functions.
2. Na página Máquinas de estado, escolha o fluxo de trabalho que você deseja editar.
3. Na página Detalhes da máquina de estado, escolha Editar.

4. O fluxo de trabalho é aberto no modo Design do Workflow Studio. Edite o fluxo de trabalho conforme necessário.



> **Note**
>
> Se você encontrar erros no fluxo de trabalho, deverá corrigi-los no modo Design. Você não pode alternar para o modo Código ou Configuração se houver algum erro no fluxo de trabalho.

5. (Opcional) Escolha o botão Código para visualizar ou editar a definição do fluxo de trabalho no Workflow Studio.



6. Quando concluir, selecione Salvar para salvar o fluxo de trabalho atualizado.
7. (Opcional) Para executar o fluxo de trabalho atualizado, escolha Executar. A caixa de diálogo Iniciar execução é aberta em uma nova guia.

Exportar o fluxo de trabalho

Você pode exportar a definição da Amazon States Language (ASL) do fluxo de trabalho e o gráfico do fluxo de trabalho:

1. Escolha o fluxo de trabalho no console do Step Functions.
2. Na página Detalhes da máquina de estado, escolha Editar.
3. (Opcional) O fluxo de trabalho é aberto no modo Design do Workflow Studio. Edite o fluxo de trabalho no modo Design ou alterne para o modo Código.
4. Escolha o botão suspenso Ações e realize uma das seguintes ações, ou ambas:

- Para exportar o gráfico do fluxo de trabalho para um arquivo SVG ou PNG, em Exportar gráfico, selecione o formato desejado.
- Para exportar a definição do fluxo de trabalho como um arquivo JSON ou YAML, em Exportar definição, selecione o formato desejado.

Criar o protótipo de fluxo de trabalho

Você pode usar o Workflow Studio para criar protótipos de novos fluxos de trabalho que contêm recursos de espaço reservado. Também é possível criar fluxos de trabalho usando o Workflow Studio no Application Composer. Para criar um protótipo:

1. Faça login no console do Step Functions.
2. Escolha Criar uma máquina de estado.
3. Na caixa de diálogo Escolher um modelo, selecione Em branco.
4. Escolha Selecionar. Isso abre o Workflow Studio no Modo de design.
5. O modo Design do Workflow Studio é aberto. Crie o fluxo de trabalho no Workflow Studio. Para incluir recursos de espaço reservado:
   a. Escolha o estado para o qual você deseja incluir um recurso de espaço reservado e, em seguida, em Configuração:
      - Para os estados Invocação Lambda, escolha o Nome da função e, em seguida, escolha Inserir nome da função. Você também pode inserir um nome personalizado para sua função.
      - Para os estados de envio de mensagens do Amazon SQS, escolha o URL da fila e, em seguida, escolha Inserir URL da fila. Insira um URL de fila do espaço reservado.
      - Para os estados de publicação do Amazon SNS, em Tópico, escolha um ARN de tópico.
      - Para todos os outros estados listados em Ações, você pode usar a configuração padrão.

      > **Note**
      >
      > Se você encontrar erros no fluxo de trabalho, deverá corrigi-los no modo Design. Você não pode alternar para o modo Código ou Configuração se houver algum erro no fluxo de trabalho.

   b. (Opcional) Para visualizar a definição de ASL gerada automaticamente do fluxo de trabalho, escolha Definição.

c. (Opcional) Para atualizar a definição do fluxo de trabalho no Workflow Studio, escolha o botão Código.

> **Note**
>
> Se você encontrar erros na definição de fluxo de trabalho, deverá corrigi-los no modo Código. Você não pode alternar para o modo Design ou Código se houver algum erro na definição do fluxo de trabalho.

6. (Opcional) Para editar o nome da máquina de estado, escolha o ícone de edição ao lado do nome padrão da máquina de estado MyStateMachinee especifique um nome na caixa Nome da máquina de estado.

   Você também pode alternar para o Modo de configuração para editar o nome da máquina de estado padrão.

7. Especifique outras configurações do fluxo de trabalho, como o tipo de máquina de estado e o perfil de execução.
8. Escolha Criar.

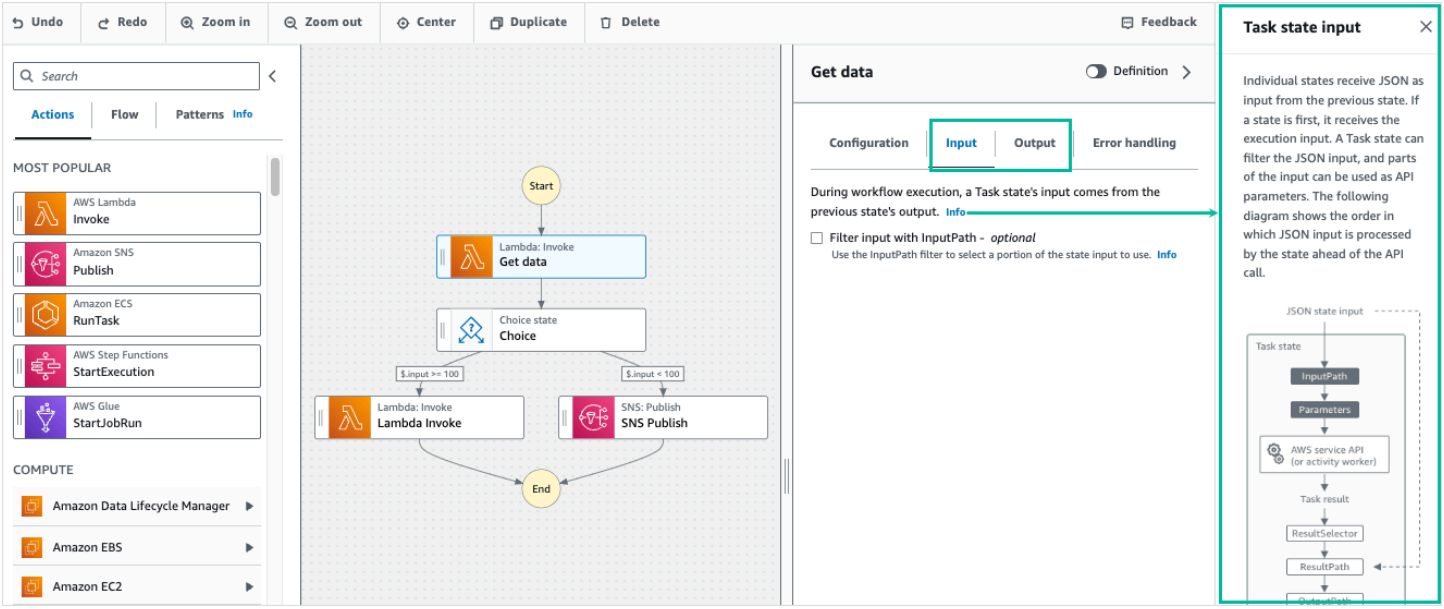
Agora você criou um novo fluxo de trabalho com recursos de espaço reservado que podem ser usados para criar protótipos. Você pode exportar a definição do fluxo de trabalho e o gráfico do fluxo de trabalho.

- Para exportar a definição de fluxo de trabalho como um arquivo JSON ou YAML, no modo Design ou Código, escolha o botão suspenso Ações. Em seguida, em Exportar definição, selecione o formato que você deseja exportar. Você pode usar essa definição exportada como o ponto de partida para o desenvolvimento local com o AWS Toolkit for Visual Studio Code.
- Para exportar o gráfico do fluxo de trabalho para um arquivo SVG ou PNG, no modo Design ou Código, escolha o botão suspenso Ações. Em seguida, em Exportar definição, selecione o formato que você deseja.

Configurar entradas e saídas para os estados

Cada estado toma uma decisão ou executa uma ação com base na entrada que recebe. Na maioria dos casos, ele passa a saída para outros estados. No Workflow Studio, você pode configurar como um estado filtra e manipula os dados de entrada e saída nas guias Entrada e Saída do painel

Inspector. Use os links de Informações para acessar a ajuda contextual ao configurar entradas e saídas.



Para informações detalhadas sobre como o Step Functions processa entrada e saída, consulte Processamento de entrada e saída no Step Functions.

Configurar a entrada para um estado

Cada estado recebe a entrada do estado anterior como JSON. Se quiser filtrar a entrada, você poderá usar o filtro `InputPath` na guia Entrada no painel Inspector. A `InputPath` é uma string, começando com $, que identifica um nó JSON específico. Eles são chamados de caminhos de referência e seguem a JsonPath sintaxe.

Para filtrar a entrada:

- Escolha Filtrar entrada com InputPath.

- Insira um valor válido JsonPathpara o `InputPath` filtro. Por exemplo, **`$.data`**.

O filtro `InputPath` será adicionado ao fluxo de trabalho.

Example Exemplo 1: Usar InputPath filtro no Workflow Studio

Digamos que a entrada para o estado inclua os dados JSON a seguir.

```
{
  "comment": "Example for InputPath",
  "dataset1": {
    "val1": 1,
    "val2": 2,
    "val3": 3
  },
  "dataset2": {
    "val1": "a",
    "val2": "b",
    "val3": "c"
  }
}
```

Para aplicar o `InputPath` filtro, escolha Filtrar entrada com e InputPath, em seguida, insira um caminho de referência apropriado. Se você inserir **`$.dataset2.val1`**, o seguinte JSON será passado como entrada para o estado.

```
{"a"}
```

Um caminho de referência também pode ter uma seleção de valores. Se os dados referenciados forem `{ "a": [1, 2, 3, 4] }`, e você aplicar o caminho de referência `$.a[0:2]` como o filtro `InputPath`, o resultado será como a seguir.

```
[ 1, 2 ]
```

Os estados Paralelo, Mapa e Pass têm uma opção adicional de filtragem de entrada chamada `Parameters` na guia Entrada. Esse filtro entra em vigor após o InputPath filtro e pode ser usado para construir um objeto JSON personalizado que consiste em um ou mais pares de valores-chave. Os valores de cada par podem ser valores estáticos, podem ser selecionados na entrada ou podem ser selecionados a partir do Objeto de contexto com um caminho.

> **Note**
>
> Para determinar que um parâmetro use um caminho para fazer referência a um nó JSON na entrada, o nome do parâmetro deve terminar com `.$`.

Example Exemplo 2: crie uma entrada JSON personalizada para o estado Paralelo

Digamos que os seguintes dados JSON sejam a entrada para um estado Paralelo.

```
{
  "comment": "Example for Parameters",
  "product": {
    "details": {
      "color": "blue",
      "size": "small",
      "material": "cotton"
    },
    "availability": "in stock",
    "sku": "2317",
    "cost": "$23"
  }
}
```

Para selecionar parte dessa entrada e passar pares de valores-chave adicionais com um valor estático, você pode especificar o seguinte no campo Parâmetros, na guia Entrada do estado Paralelo.

```
{
 "comment": "Selecting what I care about.",
 "MyDetails": {
    "size.$": "$.product.details.size",
    "exists.$": "$.product.availability",
    "StaticValue": "foo"
    }
 }
```

Os seguintes dados JSON serão o resultado.

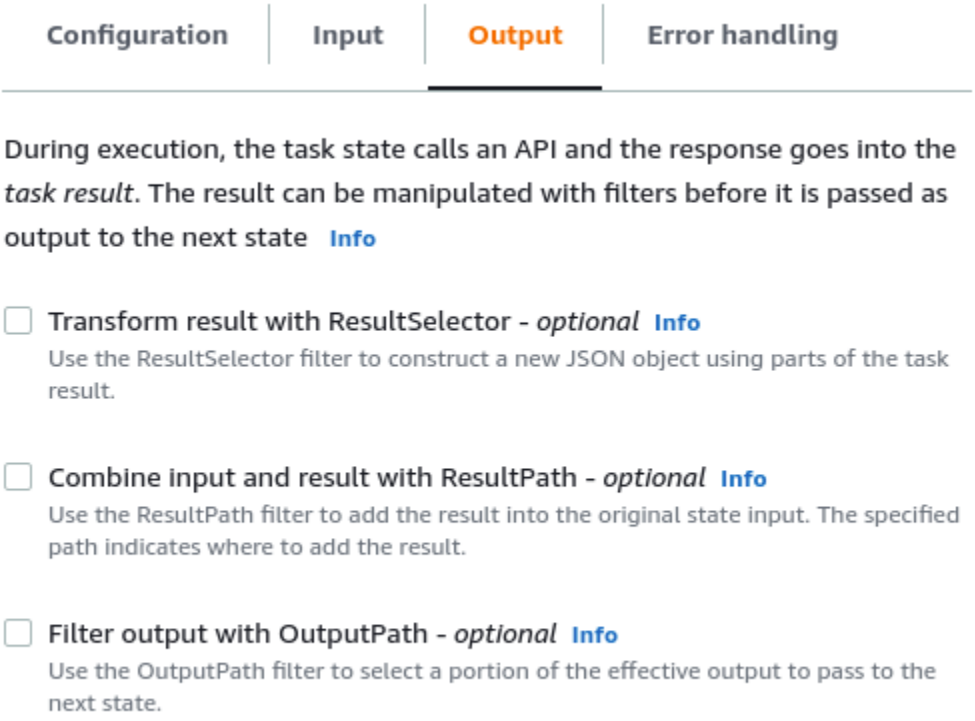
```
{
```

```
  "comment": "Selecting what I care about.",
  "MyDetails": {
    "size": "small",
    "exists": "in stock",
    "StaticValue": "foo"
  }
}
```

Configurar a saída de um estado

Cada estado produz uma saída JSON que pode ser filtrada antes de ser passada para o próximo estado. Há vários filtros disponíveis e cada um afeta a saída de uma maneira diferente. Os filtros de saída disponíveis para cada estado estão listados na guia Saída no painel Inspector. Para estados Estado da tarefa, todos os filtros de saída selecionados são processados nesta ordem:

1. `ResultSelector`: use esse filtro para manipular o resultado do estado. Você pode construir um novo objeto JSON com partes do resultado.
2. `ResultPath`: use esse filtro para selecionar uma combinação da entrada de estado e do resultado da tarefa para passar para a saída.
3. `OutputPath`: use esse filtro para filtrar a saída JSON e escolher quais informações do resultado serão passadas para o próximo estado.



Use ResultSelector

`ResultSelector` é um filtro de saída opcional para os seguintes estados:

- Estados Estado da tarefa, que são todos os estados listados na guia Ações do Navegador de estados.
- Estados Mapa, na guia Fluxo do Navegador de estados.
- Estados Paralelo, na guia Fluxo do Navegador de estados.

`ResultSelector` pode ser usado para construir um objeto JSON personalizado que consiste em um ou mais pares de chave-valor. Os valores de cada par podem ser valores estáticos ou selecionados do resultado do estado com um caminho.

> **Note**
>
> Para especificar que um parâmetro use um caminho para fazer referência a um nó JSON no resultado, o nome do parâmetro deve terminar com `.$`.

Example Exemplo de uso do ResultSelector filtro

Neste exemplo, você usa `ResultSelector` para manipular a resposta da chamada de API do Amazon CreateCluster EMR para um estado do Amazon `CreateCluster` EMR. Veja a seguir o resultado da chamada de API `CreateCluster` do 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 selecionar parte dessas informações e passar um par adicional de valores-chave com um valor estático, especifique o seguinte no ResultSelectorcampo, na guia Saída do estado.

```
{
 "result": "found",
 "ClusterId.$": "$.output.ClusterId",
 "ResourceType.$": "$.resourceType"
 }
```

O uso de `ResultSelector` produz o resultado a seguir.

```
{
 "result": "found",
 "ClusterId": "AKIAIOSFODNN7EXAMPLE",
 "ResourceType": "elasticmapreduce"
```

```
}
```

Use ResultPath

A saída de um estado pode ser uma cópia da entrada, o resultado que ele produz ou uma combinação da entrada e do resultado. Use `ResultPath` para controlar qual combinação desses itens são passadas para o estado de saída. Para obter mais casos de uso de `ResultPath`, consulte ResultPath.

`ResultPath` é um filtro de saída opcional para os seguintes estados:

- Estados Estado da tarefa, que são todos os estados listados na guia Ações do Navegador de estados.
- Estados Mapa, na guia Fluxo do Navegador de estados.
- Estados Paralelo, na guia Fluxo do Navegador de estados.
- Estados Pass, na guia Fluxo do Navegador de estados.

`ResultPath` pode ser usado para adicionar o resultado à entrada do estado original. O caminho especificado indica onde adicionar o resultado.

Example Exemplo de uso do ResultPath filtro

Digamos que os itens seguintes sejam a entrada para um estado Tarefa.

```
{
  "details": "Default example",
  "who": "AWS Step Functions"
}
```

O resultado do estado Tarefa é como a seguir.

```
Hello, AWS Step Functions
```

Você pode adicionar esse resultado à entrada do estado aplicando `ResultPath` e inserindo um caminho de referência que indica onde adicionar o resultado, como `$.taskresult`:

Com esse `ResultPath`, o seguinte é o JSON que é passado como a saída do estado.

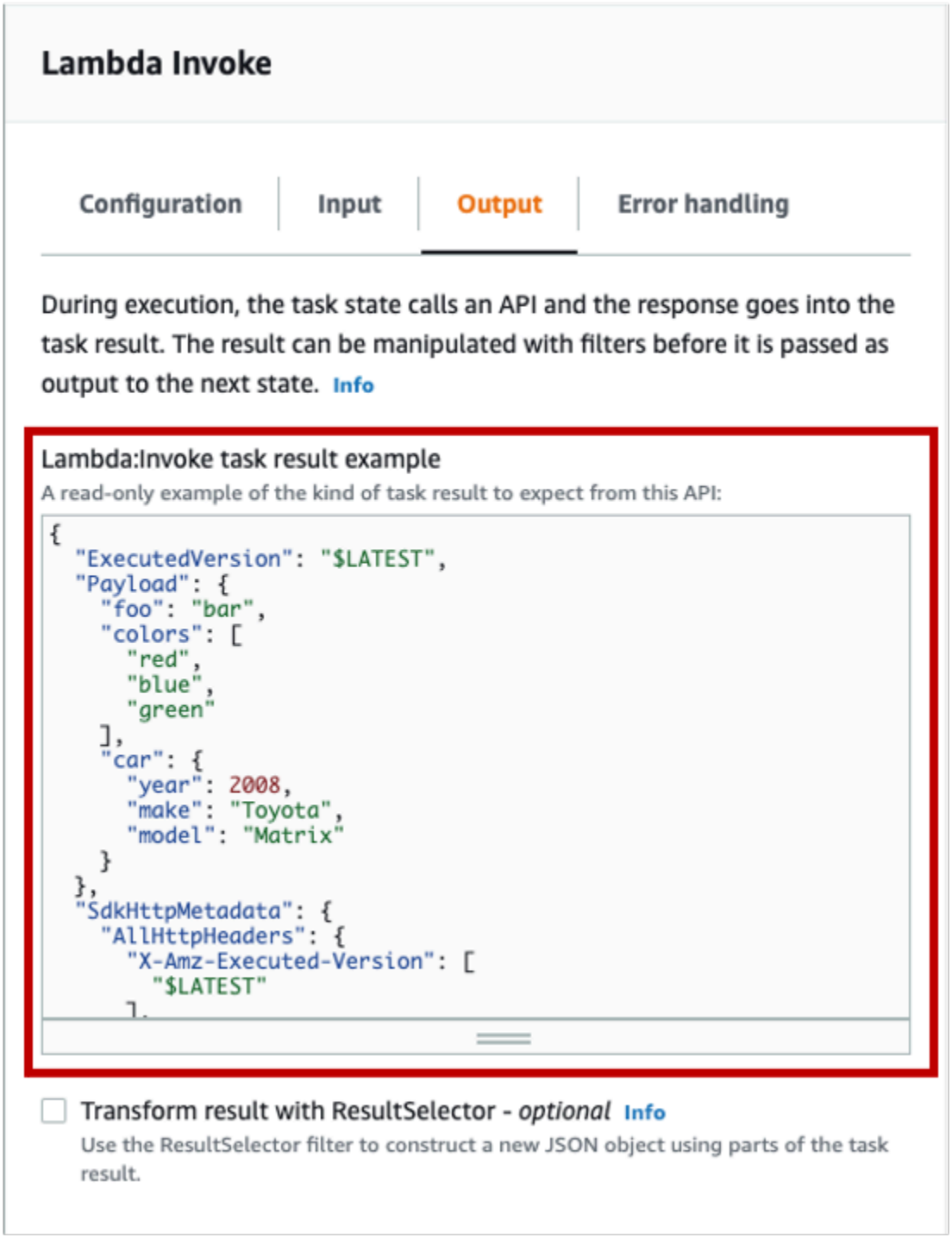
```
{
  "details": "Default example",
  "who": "AWS Step Functions",
  "taskresult": "Hello, AWS Step Functions!"
}
```

Use OutputPath

O filtro `OutputPath` permite filtrar informações indesejadas e transmitir somente a parte do JSON que é importante para você. O `OutputPath` é uma string, que começa com $, que identifica nós no texto JSON.

Example Exemplo de uso do OutputPath filtro

Uma chamada de API Invocação Lambda retorna metadados além da payload, que é o resultado da função do Lambda. Um exemplo da resposta dessa chamada de API é mostrado na guia Saída do estado.



Você pode usar `OutputPath` para filtrar os metadados adicionais. Por padrão, o valor do OutputPathfiltro para estados do Lambda Invoke criados por meio do Workflow Studio é. `$.Payload` Esse valor padrão remove os metadados adicionais e retorna uma saída equivalente à execução direta da função do Lambda.

O exemplo do resultado da tarefa Invocação Lambda e o valor de `$.Payload` para o filtro Output transmitem os seguintes dados JSON como a saída.

```
{
 "foo": "bar",
 "colors": [
      "red",
      "blue",
      "green"
 ],
 "car": {
      "year": 2008,
      "make": "Toyota",
      "model": "Matrix"
 }
}
```

> **Note**
>
> Como o filtro `OutputPath` é o último filtro de saída a entrar em vigor, se você usar filtros de saída adicionais como `ResultSelector` ou `ResultPath`, deverá modificar o valor padrão de `$.Payload` para o filtro `OutputPath` adequadamente.

Perfis de execução no Workflow Studio

Cada máquina de Step Functions estado requer uma função AWS Identity and Access Management (IAM) que concede à máquina de estado permissão para realizar ações Serviços da AWS e recursos ou chamar APIs de terceiros. Esse perfil é chamado de perfil de execução. Esse perfil deve conter políticas do IAM para cada ação, por exemplo, políticas que permitam que a máquina de estado invoque uma função do AWS Lambda, execute um trabalho AWS Batch ou chame a API do Stripe. O Step Functions exige que você forneça um perfil de execução nos seguintes casos:

- Você cria uma máquina de estado no console, nos AWS SDKs ou AWS CLI usando a CreateStateMachineAPI.
- Você testa um estado no console, nos AWS SDKs ou AWS CLI usando a TestStateAPI.

O Workflow Studio tem recursos que facilitam o gerenciamento de perfis de execução para os fluxos de trabalho.

Tópicos

Sobre perfis gerados automaticamente

Quando você cria uma máquina de estado no console do Step Functions, o Workflow Studio pode criar automaticamente um perfil de execução para você que contém as políticas do IAM necessárias. O Workflow Studio analisa a definição de máquina de estado e gera políticas com os privilégios mínimos necessários para executar o fluxo de trabalho.

O Workflow Studio pode gerar políticas do IAM para o seguinte:

- Tarefas HTTP que chamam APIs de terceiros.
- Estados de tarefas que chamam outras pessoas Serviços da AWS usando integrações otimizadas, como LambdaInvoke, DynamoDB GetItem, ou. AWS Glue StartJobRun
- Estados de tarefas que executam fluxos de trabalho aninhados.
- Estados de mapas distribuídos, incluindo políticas para iniciar execuções de fluxos de trabalho secundários, listar buckets do Amazon S3 e ler ou gravar objetos do S3.
- Rastreamento do X-Ray. Cada perfil gerado automaticamente no Workflow Studio contém uma política que concede permissões para a máquina de estado enviar rastreamentos ao X-Ray.
- Como registrar usando o CloudWatch Logs quando o registro em log está habilitado na máquina de estado.

O Workflow Studio não pode gerar IAM políticas para estados de tarefas que chamam outras pessoas Serviços da AWS usando integrações de AWS SDK.

Gerar perfis automaticamente

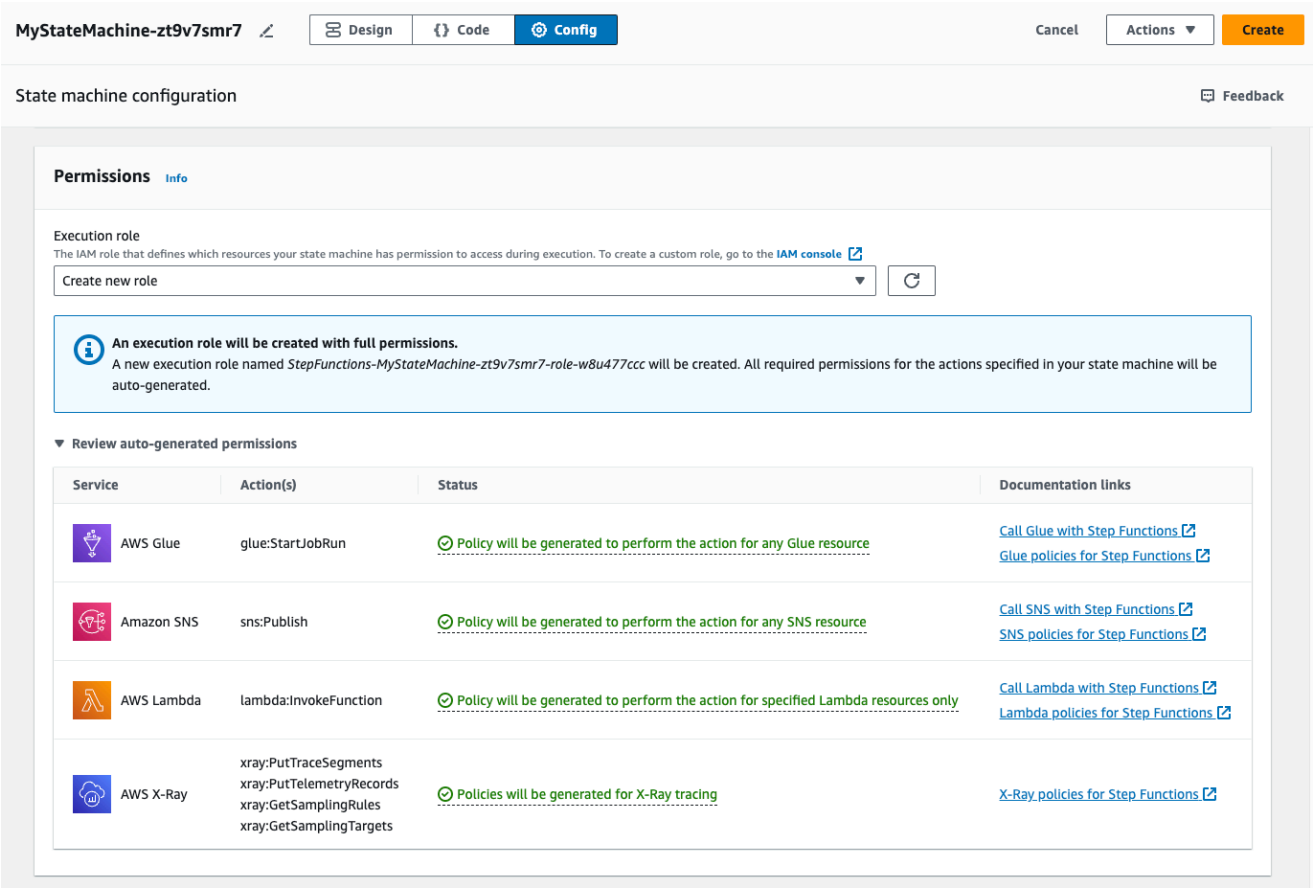
1. Abra o console do Step Functions e clique em Criar máquina de estado.

Também é possível atualizar uma máquina de estado existente. Consulte a Etapa 4 se você estiver atualizando uma máquina de estado.

2. Na caixa de diálogo Escolher um modelo, selecione Em branco.
3. Escolha Selecionar. Isso abre o Workflow Studio no Modo de design.
4. Selecione a guia Config.
5. Role para baixo até a seção Permissões e faça o seguinte:

   a. Em Perfil de execução, mantenha a seleção padrão de Criar perfil.

      O Workflow Studio gera automaticamente todas as políticas do IAM necessárias para cada estado válido na definição da máquina de estado. Ele exibe um banner com a mensagem: Um perfil de execução será criado com todas as permissões.



> **Tip**
>
> Para revisar as permissões que o Workflow Studio vai gerar automaticamente para a máquina de estado, selecione Revisar permissões geradas automaticamente.

> **Note**
>
> Se você excluir o perfil do IAM criado pelo Step Functions, não será possível recriá-lo posteriormente. Da mesma forma, se você modificar a função (por exemplo, removendo o Step Functions das entidades principais na política do IAM), o Step Functions não poderá restaurar as configurações originais dela posteriormente.

Se o Workflow Studio não conseguir gerar todas as políticas do IAM necessárias, ele exibirá um banner com a mensagem Permissões para determinadas ações não podem ser geradas automaticamente. Um perfil do IAM será criado somente com permissões parciais. Para obter informações sobre como adicionar as permissões ausentes, consulte Resolver problemas de geração de perfis.

b. Selecione Criar se você estiver criando uma máquina de estado. Caso contrário, selecione Salvar.

c. Selecione Confirmar na caixa de diálogo exibida.

O Workflow Studio salva a máquina de estado e cria o perfil de execução.

Resolver problemas de geração de perfis

O Workflow Studio não pode gerar automaticamente um perfil de execução com todas as permissões necessárias nos seguintes casos:

- Há erros na máquina de estado. Assegure-se de resolver todos os erros de validação no Workflow Studio. Além disso, solucione todos os erros do servidor encontrados durante o salvamento.
- Sua máquina de estado contém tarefas que usam integrações de AWS SDK. Nesse caso, o Workflow Studio não consegue gerar automaticamente políticas do IAM. O Workflow Studio exibe um banner com a mensagem: Permissões para determinadas ações não podem ser geradas automaticamente. Um perfil do IAM será criado somente com permissões parciais. Na tabela

Revisar permissões geradas automaticamente, selecione o conteúdo em Status para obter mais informações sobre as políticas que faltam no perfil de execução. O Workflow Studio ainda pode gerar um perfil de execução, mas esse perfil não conterá políticas do IAM para todas as ações. Consulte os links em Links de documentação para criar as próprias políticas e adicioná-las ao perfil depois de gerado. Esses links estão disponíveis mesmo depois de salvar a máquina de estado.

Perfil para testar tarefas HTTP no Workflow Studio

É necessário ter um perfil de execução para testar um estado de tarefa HTTP. Se você não tiver um perfil com permissões suficientes, use uma das seguintes opções para criar um perfil:

- Gerar automaticamente um perfil com o Workflow Studio (recomendado): essa é a opção segura. Feche a caixa de diálogo Testar estado e siga as instruções em Gerar perfis automaticamente. Isso exigirá que você primeiro crie ou atualize a máquina de estado e depois volte ao Workflow Studio para testar o estado.
- Use uma função com acesso de administrador — Se você tiver permissões para criar uma função com acesso total a todos os serviços e recursos do AWS, poderá usar essa função para testar qualquer tipo de estado em seu fluxo de trabalho. Para fazer isso, você pode criar uma função de Step Functions serviço e adicionar a AdministratorAccess política a ela no IAM console https://console.aws.amazon.com/iam/.

Perfil para testar uma integração de serviços otimizada no Workflow Studio

É necessário ter um perfil de execução para estados de Tarefa que chamam integrações de serviços otimizadas. Se você não tiver um perfil com permissões suficientes, use uma das seguintes opções para criar um perfil:

- Usar os links da documentação no Workflow Studio para criar as próprias políticas do IAM (recomendado): essa é a opção segura. Feche a caixa de diálogo Testar estado e siga as instruções em Gerar perfis automaticamente. Isso exigirá que você primeiro crie ou atualize a máquina de estado e depois volte ao Workflow Studio para testar o estado.
- Use uma função com acesso de administrador — Se você tiver permissões para criar uma função com acesso total a todos os serviços e recursos do AWS, poderá usar essa função para testar qualquer tipo de estado em seu fluxo de trabalho. Para fazer isso, você pode criar uma função de Step Functions serviço e adicionar a AdministratorAccess política a ela no IAM console https://console.aws.amazon.com/iam/.

Função para testar a integração de um serviço AWS SDK no Workflow Studio

É necessário ter um perfil de execução para estados de Tarefa que chamam integrações de serviços de SDKs da AWS. Se você não tiver um perfil com permissões suficientes, use uma das seguintes opções para criar um perfil:

- Usar os links da documentação no Workflow Studio para criar as próprias políticas do IAM (recomendado): essa é a opção segura. Feche a caixa de diálogo Testar estado e siga as instruções em Gerar perfis automaticamente. Isso exigirá que você primeiro crie ou atualize a máquina de estado e depois volte ao Workflow Studio para testar o estado. Faça o seguinte:
  1. Feche a caixa de diálogo Testar estado.
  2. Selecione a guia Config para visualizar o modo Config.
  3. Role para baixo até a seção Permissões.
  4. O Workflow Studio exibe um banner com a mensagem: Permissões para determinadas ações não podem ser geradas automaticamente. Um perfil do IAM será criado somente com permissões parciais. Selecione Revisar permissões geradas automaticamente.
  5. A tabela Revisar permissões geradas automaticamente exibe uma linha que mostra a ação correspondente ao estado da tarefa que você deseja testar. Consulte os links em Links de documentação para criar as próprias políticas do IAM em um perfil personalizado.
- Use uma função com acesso de administrador — Se você tiver permissões para criar uma função com acesso total a todos os serviços e recursos do AWS, poderá usar essa função para testar qualquer tipo de estado em seu fluxo de trabalho. Para fazer isso, você pode criar uma função de Step Functions serviço e adicionar a AdministratorAccess política a ela no IAM console https://console.aws.amazon.com/iam/.

Perfil para testar estados de fluxo no Workflow Studio

É necessário ter um perfil de execução para testar os estados de fluxo no Workflow Studio. Os estados de fluxo são aqueles estados que direcionam o fluxo de execução, como Choice, Paralelo, Mapa, Pass, Aguardar, Succeed ou Fail. A TestStateAPI não funciona com estados Map ou Parallel. Use uma das seguintes opções para criar um perfil para testar um estado do fluxo:

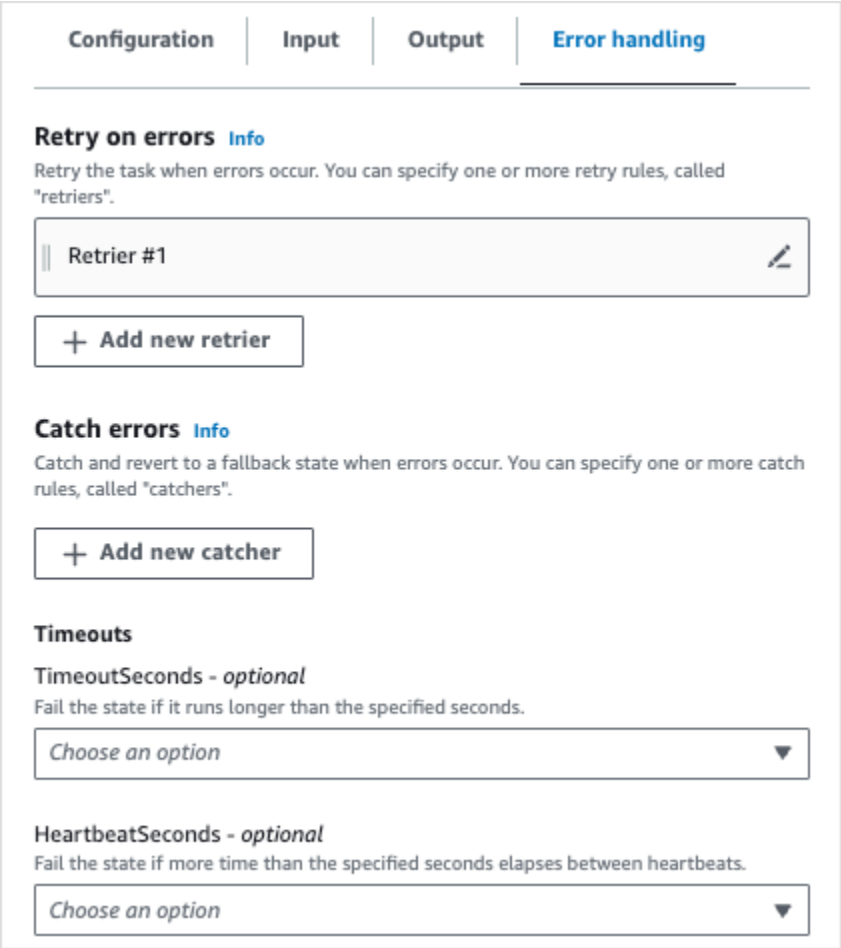
- Use qualquer função em seu Conta da AWS (recomendado) — Os estados de fluxo não exigem nenhuma IAM política específica, porque eles não chamam AWS ações ou recursos. Portanto, você pode usar qualquer IAM função em seu Conta da AWS.

1. Na caixa de diálogo Testar estado, selecione qualquer perfil na lista suspensa Perfil de execução.
2. Se nenhum perfil aparecer na lista suspensa, faça o seguinte:
   a. No console do IAM https://console.aws.amazon.com/iam/, selecione Perfis.
   b. Selecione um perfil na lista e copie o ARN na página de detalhes do perfil. Será necessário fornecer esse ARN na caixa de diálogo Testar estado.
   c. Na caixa de diálogo Testar estado, selecione Inserir um ARN de perfil na lista suspensa Perfil de execução.
   d. Cole o ARN em ARN do perfil.

- Use uma função com acesso de administrador — Se você tiver permissões para criar uma função com acesso total a todos os serviços e recursos do AWS, poderá usar essa função para testar qualquer tipo de estado em seu fluxo de trabalho. Para fazer isso, você pode criar uma função de Step Functions serviço e adicionar a AdministratorAccess política a ela no IAM console https://console.aws.amazon.com/iam/.

Tratamento de erros

Por padrão, quando um estado relata um erro, o Step Functions faz com que a execução do fluxo de trabalho falhe totalmente. Para ações e alguns estados de fluxo, você pode configurar como o Step Functions lida com erros. Mesmo que você tenha configurado o tratamento de erros, alguns erros ainda podem causar uma falha na execução de um fluxo de trabalho. Para ter mais informações, consulte Tratamento de erros no Step Functions. No Workflow Studio, configure o tratamento de erros na guia Tratamento de erros do painel Inspector.



Tentar novamente em caso de erros

Você pode adicionar uma ou mais regras aos estados de ação e ao estado de fluxo Paralelo para repetir a tarefa quando ocorrer um erro. Essas regras são chamadas de retriers. Para adicionar um retrier, escolha o ícone de edição na caixa Retrier nº 1 e configure as opções:

- (Opcional) No campo Comentário, adicione seu comentário. Isso não afetará o fluxo de trabalho, mas pode ser usado para anotar o fluxo de trabalho.
- Coloque o cursor no campo Erros e escolha um erro que acionará o retrier ou insira um nome de erro personalizado. Você pode escolher ou adicionar vários erros.
- (Opcional) Defina um Intervalo. Esse é o tempo em segundos antes que o Step Functions faça a primeira nova tentativa. Novas tentativas ocorrerão em intervalos que você pode configurar com o Máximo de tentativas e a Taxa de recuo.
- (Opcional) Defina o Máximo de tentativas. Esse é o número máximo de novas tentativas antes que o Step Functions cause falha na execução.

- (Opcional) Defina a Taxa de recuo. Esse é um multiplicador que determina em quanto o intervalo de nova tentativa aumentará a cada tentativa.

> **Note**
>
> Nem todas as opções de tratamento de erros estão disponíveis para todos os estados. Invocação Lambda tem um retrier configurado por padrão.

Detecção de erros

Você pode adicionar uma ou mais regras aos estados de ação e aos estados de fluxo Paralelo e Mapa para capturar um erro. Essas regras são chamadas de catchers. Para adicionar um catcher, escolha Adicionar novo catcher e configure as opções:

- (Opcional) No campo Comentário, adicione seu comentário. Isso não afetará o fluxo de trabalho, mas pode ser usado para anotar o fluxo de trabalho.
- Coloque o cursor no campo Erros e escolha um erro que acionará o catcher ou insira um nome de erro personalizado. Você pode escolher ou adicionar vários erros.
- No campo Estado de fallback, escolha um estado de fallback. Esse é o estado para o qual o fluxo de trabalho será movido em seguida, depois que um erro for detectado.
- (Opcional) No ResultPathcampo, adicione um `ResultPath` filtro para adicionar o erro à entrada do estado original. O `ResultPath`deve ser válido JsonPath. Isso será enviado para o estado de fallback.

Tempos limite

Você pode configurar um tempo limite para os estados de ação para definir o número máximo de segundos que o estado pode ser executado antes que ele falhe. Use tempos limite para evitar execuções travadas. Para configurar um tempo limite, insira o número de segundos que o estado deve esperar antes que a execução falhe. Para obter mais informações sobre os tempos limites, consulte `TimeoutSeconds`no estado Estado da tarefa.

HeartbeatSeconds

Você pode configurar um Heartbeat ou uma notificação periódica enviada pela tarefa. Se você definir um intervalo de heartbeat, e o estado não enviar notificações de heartbeat nos intervalos

configurados, a tarefa será marcada como falha. Para configurar uma heartbeat, defina um número inteiro positivo diferente de zero de segundos. Para ter mais informações, consulte `HeartBeatSeconds` no estado Estado da tarefa.

Tutorial: aprenda a usar o Workflow Studio no AWS Step Functions

Neste tutorial, você aprenderá os conceitos básicos de como trabalhar com o Workflow Studio no AWS Step Functions. No Modo de design do Workflow Studio, você criará uma máquina de estados contendo vários estados, incluindo `Pass`, `Choice`, `Fail`, `Wait` e `Parallel`. Você usará o atributo de arrastar e soltar para pesquisar, selecionar e configurar esses estados. Em seguida, você verá a definição do fluxo de trabalho gerada automaticamente pela Amazon States Language (ASL). Você também usará o Modo de código do Workflow Studio para editar a definição do fluxo de trabalho. Após isso, você sairá do Workflow Studio, executará a máquina de estado e analisará os detalhes da execução.

Neste tutorial, você também aprenderá como atualizar a máquina de estado e visualizar as alterações na saída de execução. Por fim, você executará uma etapa de limpeza e excluirá a máquina de estado.

Depois de concluir este tutorial, você saberá como usar o Workflow Studio para criar e configurar fluxos de trabalho usando os modos Design e Código. Você também aprenderá a atualizar, executar e excluir a máquina de estado.

> **Note**
>
> Verifique os pré-requisitos antes de começar este tutorial.

Tópicos

Etapa 1: Navegue até o Workflow Studio

1. Abra o console do Step Functions e clique em Criar máquina de estado.
2. Na caixa de diálogo Escolher um modelo, selecione Em branco.
3. Escolha Selecionar. Isso abre o Workflow Studio no Modo de design.

Etapa 2: Criar uma máquina de estado

No Workflow Studio, a máquina de estado é uma representação gráfica do fluxo de trabalho. Com o Workflow Studio, você pode definir, configurar e examinar as etapas individuais do fluxo de trabalho. Nas etapas a seguir, você usará o Modo de design do Workflow Studio para criar uma máquina de estado.

Para criar uma máquina de estado do

1. Verifique se você está no modo Design do Workflow Studio.
2. No Navegador de estados à esquerda, clique na guia Fluxo. Em seguida, arraste o estado Passagem para o estado vazio Arrastar o primeiro estado para cá.
3. Arraste o estado Escolha da guia Fluxo e solte-o abaixo do estado Passagem.
4. Em Nome do estado, mude o nome Escolha. Neste tutorial, use o nome **`IsHelloWorldExample`**.
5. Arraste outro estado do Pass e solte-o em uma ramificação do IsHelloWorldExampleestado. Em seguida, arraste um estado de falha e solte-o abaixo da outra ramificação do IsHelloWorldExampleestado.
6. Clique em Passagem (1) adicionado e renomeie-o para **`Yes`**. Renomeie o estado Falha para **`No`**.
7. Especifique a lógica de ramificação do IsHelloWorldExampleestado usando a variável booleana. `IsHelloWorldExample`

   Se `IsHelloWorldExample` for `False`, o fluxo de trabalho entrará no estado Não. Caso contrário, o fluxo de trabalho continuará o fluxo de execução no estado Sim.

   Para definir a lógica de ramificação, faça o seguinte:

   a. Escolha o IsHelloWorldExampleestado no eCanvas, em Regras de escolha, escolha o ícone de edição na caixa Regra #1 para definir a regra de primeira escolha.

   b. Escolha Adicionar condições.

c. Na caixa de diálogo Condições da regra 1, digite **`$.IsHelloWorldExample`** em Variável.

d. Em Operador, escolha a opção é igual a.

e. Selecione Constante booleana em Valor e depois escolha verdadeiro na lista suspensa.

f. Clique em Salvar condições.

g. Na lista suspensa Então o próximo estado é:, selecione Sim.

h. Clique em Adicionar nova regra de escolha e depois em Adicionar condições.

i. No campo Regra 2, defina a segunda regra de escolha para quando o valor da variável `IsHelloWorldExample` for falso, repetindo as subetapas de 7.c a 7.f. Para a etapa 7.e, selecione falso em vez de verdadeiro.

j. No campo Regra 2, selecione Não na lista suspensa Então o próximo estado é:.

k. No campo Regra padrão, clique no ícone de edição para definir a regra de escolha padrão e, em seguida, selecione Sim na lista suspensa.

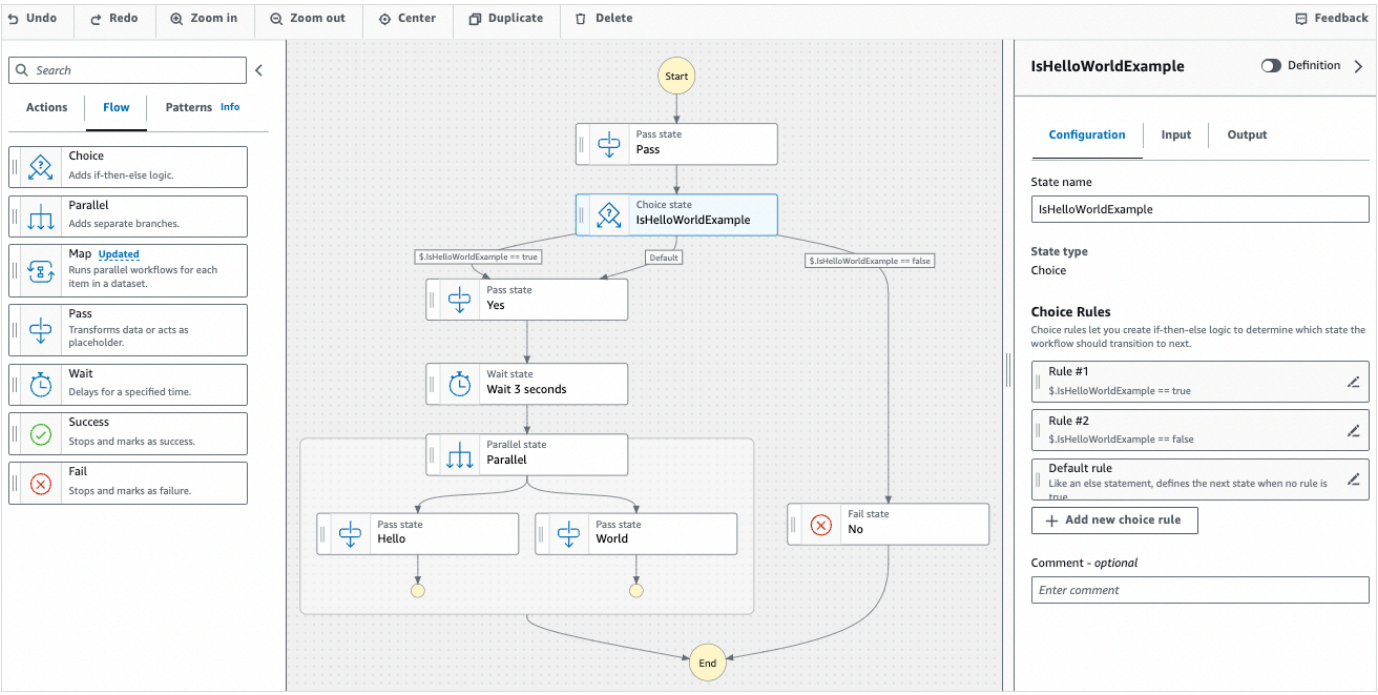
8. Adicione um estado Aguardar após o estado Sim e nomeie-o como **`Wait 3 sec`**. Logo após, configure o tempo de espera para três segundos realizando as etapas a seguir.

   a. Em Opções, não altere a configuração definida em Aguardar um intervalo fixo.

   b. Em Segundos, verifique se a opção Inserir segundos está selecionada e digite **`3`** no campo.

9. Após o estado Aguardar 3 seg, adicione o estado Paralelo. Adicione dois estados Passagem em suas duas ramificações. Nomeie o primeiro estado Passagem como **`Hello`**. Nomeie o segundo estado Passagem como **`World`**.

   O fluxo de trabalho concluído ficará assim:



Etapa 3: Revisar a definição da Amazon States Language gerada automaticamente

Conforme você arrasta e solta os estados da guia Fluxo na tela, o Workflow Studio cria automaticamente a definição da Amazon States Language (ASL) do fluxo de trabalho em tempo real. No painel Inspector, clique no botão de alternância Definição para visualizar essa definição ou mude para o Modo de código para alterá-la conforme necessário. Para obter informações sobre como alterar a definição do fluxo de trabalho, consulte a Etapa 4 deste tutorial.

- (Opcional) Clique em Definição no painel Inspector e veja o fluxo de trabalho da máquina de estado.

  O código de exemplo a seguir mostra a definição da Amazon States Language gerada automaticamente para a máquina de estado `IsHelloWorldExample`. O estado `Choice` adicionado no Workflow Studio é usado para determinar o fluxo de execução com base na lógica de ramificação que você definiu na Etapa 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
            }
          }
        }
      ]
    }
  }
}
```

Etapa 4: Alterar a definição do fluxo de trabalho no modo Código

O modo Código do Workflow Studio fornece um editor de código integrado para visualizar e alterar a definição da ASL dos fluxos de trabalho.

1. Clique em Código para alternar para o modo Código.
2. Após a definição do estado Paralelo, posicione o cursor e pressione **`Enter`**.
3. Pressione **`Ctrl+space`** para ver a lista de estados que podem ser adicionados após o estado Paralelo.
4. Escolha Estado Passagem na lista de opções. O editor de código adiciona um código clichê para o Estado Passagem.
5. A adição desse estado gera erros na definição do fluxo de trabalho. Na definição do estado Paralelo, substitua `"End": true` por **`"Next": "PassState"`**.
6. Na definição do Estado Passagem que você adicionou, faça as seguintes alterações:
   a. Remova o nó Resultado.
   b. Remova `"ResultPath": "$.result"`, e `"Next": "NextState"`.

c. Depois de `"Type": "Pass",`, digite **`"End": true`**.

d. Adicione uma **`,`** após a definição do Estado Passagem.

A definição do fluxo de trabalho agora deve ser semelhante à definição a seguir.

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

Etapa 5: Salvar a máquina de estado

1. Escolha Config more ou escolha o ícone de edição ao lado do nome padrão da máquina de estado de. MyStateMachine Em Configuração da máquina de estado, forneça um nome adequado. Por exemplo, digite **`HelloWorld`**.
2. (Opcional) Especifique outras configurações do fluxo de trabalho, como o tipo de máquina de estado e função de execução. Para este tutorial, mantenha todas as seleções padrão na Configuração da máquina de estado.
3. Escolha Criar.
4. Na caixa de diálogo Confirmar criação do perfil, selecione Confirmar para continuar.

Você também pode clicar em Exibir configurações da função para retornar ao modo Configurar.

Para obter mais informações sobre o modo Config, consulte Modo Configurar do Workflow Studio.

Etapa 6: Executar a máquina de estado

As execuções de máquinas de estado são instâncias em que o fluxo de trabalho é executado para a realização de tarefas.

1. Na página Máquinas de estado, escolha a máquina de HelloWorldestado.
2. Na HelloWorldpágina, escolha Iniciar execução.
3. (Opcional) Para identificar a execução, insira um nome para ela no campo Nome. Por padrão, o Step Functions cria automaticamente um nome de execução exclusivo.

   > **Note**
   >
   > Step Functions permite criar nomes para máquinas de estado, execuções e atividades, além de rótulos que contenham caracteres não ASCII. Esses nomes não ASCII não funcionam com a Amazon. CloudWatch Para garantir que você possa acompanhar CloudWatch as métricas, escolha um nome que use somente caracteres ASCII.

4. No campo Entrada, insira os valores de entrada da execução no formato JSON. Com base na entrada, a variável `IsHelloWorldExample` determina qual fluxo de máquina de estado será executado. Por enquanto, use o seguinte valor de entrada:

   ```
   {
      "IsHelloWorldExample": true
   }
   ```

   > **Note**
   >
   > Embora especificar uma entrada de execução seja opcional, neste tutorial é obrigatório especificar uma entrada de execução semelhante à entrada do exemplo acima. Esse valor de entrada é referenciado no estado `Choice` quando você executa a máquina de estado.

5. Selecione Iniciar execução.

6. O console do Step Functions direciona você para uma página em que o título é o ID da execução. Essa página é conhecida como página de Detalhes da execução. Nesta página, você pode revisar os resultados da execução à medida que a execução avança ou após a conclusão.

   Para revisar os resultados da execução, escolha estados individuais na Exibição em gráfico e, em seguida, escolha as guias individuais no painel Detalhes da etapa para visualizar os detalhes de cada estado, incluindo entrada, saída e definição, respectivamente. Para obter detalhes sobre as informações de execução que você pode visualizar na página Detalhes da execução, consulte Página de Detalhes da execução — Visão geral da interface.

   Para este tutorial, se você inseriu o valor de entrada `"IsHelloWorldExample": true`, deve ver a saída a seguir.

   ```
   {
      "IsHelloWorldExample": true
   },
   {
      "IsHelloWorldExample": true
   }
   ```

Etapa 7: Atualizar a máquina de estado

Quando você atualiza uma máquina de estado, as atualizações são realizadas por meio da consistência posterior. Depois de um curto período, todas as execuções recém-iniciadas refletirão a definição atualizada da máquina de estado. Todas as execuções em andamento no momento serão concluídas de acordo com a definição anterior.

Nesta etapa, você atualizará a máquina de estado no modo Modo de design do Workflow Studio. Você adicionará o campo `Result` no estado Passagem chamado Mundo.

1. Na página com o título do ID da execução, clique em Editar máquina de estado.
2. Verifique se você está no modo Design.
3. Clique no estado Passagem chamado Mundo na tela e, em seguida, escolha Saída.
4. No campo Resultado, digite **`"World has been updated!"`**.
5. Selecione Salvar.
6. (Opcional) Na área de Definição, veja a definição atualizada da Amazon States Language para o fluxo de trabalho.

```
{
      "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. Clique em Executar.
8. Na caixa de diálogo Iniciar execução que é aberta em uma nova guia, digite a seguinte entrada de execução.

```
{
   "IsHelloWorldExample": true
}
```

9. Escolha Start Execution.
10. (Opcional) Na Exibição em gráfico, selecione a etapa Mundo e, em seguida, clique em Saída. A saída é Mundo foi atualizado!

Etapa 8: Limpeza

Para excluir a máquina de estado

1. No menu de navegação, clique em Máquinas de estado.
2. Na página Máquinas de estado, selecione e HelloWorld, em seguida, escolha Excluir.
3. Na caixa de diálogo Excluir máquina de estado, digite **`delete`** para confirmar a exclusão.
4. Escolha Delete.

   Se a exclusão for bem-sucedida, uma barra de status verde será exibida na parte superior da tela. A barra de status verde informa que a máquina de estado foi selecionada para exclusão. A máquina de estado será excluída assim que todas as execuções em andamento pararem de ser executadas.

Para excluir a função de execução

1. Abra a página Perfis do IAM.
2. Escolha o perfil do IAM que o Step Functions criou para você. Por exemplo, StepFunctions-HelloWorld -Role-example.
3. Clique em Excluir função.
4. Depois em Sim, excluir.

Tutoriais do Step Functions

Os tutoriais nesta seção podem ajudar você a entender diferentes aspectos de se trabalhar com o AWS Step Functions.

Para concluir esses tutoriais, você precisa de uma AWS conta. Se você não tiver uma AWS conta, acesse https://aws.amazon.com/ e escolha Criar uma AWS conta.

Tópicos

Como criar uma máquina de estado Step Functions que usa Lambda

Neste tutorial, você criará um fluxo de trabalho de uma única etapa usando AWS Step Functions para invocar uma AWS Lambda função.

> **Note**
>
> O Step Functions é baseado em máquinas e tarefas de estado. Em Step Functions, as máquinas de estado são chamadas de fluxos de trabalho, que são uma série de etapas orientadas por eventos. Cada etapa em um fluxo de trabalho é chamada de estado. Por exemplo, um estado de tarefa representa uma unidade de trabalho que outro AWS serviço executa, como chamar outro AWS service (Serviço da AWS) ou uma API.
> Para obter mais informações, consulte:
>
> - O que AWS Step Functionsé
> - Ligue para outros AWS serviços

O Lambda é adequado para `Task` estados, pois as funções do Lambda têm a tecnologia sem servidor e fáceis de escrever. Você pode escrever código no editor AWS Management Console ou no seu editor favorito. AWS trata dos detalhes de fornecer um ambiente de computação para sua função e executá-la.

Neste tópico:

Etapa 1: criar uma função do Lambda

Sua função do Lambda recebe dados do evento e retorna uma mensagem de saudação.

> **Important**
>
> Certifique-se de que sua função Lambda esteja na mesma AWS conta e AWS região da sua máquina estadual.

1. Abra o console do Lambda e clique em Criar função.
2. Na página Create function, selecione Author from scratch.

3. Em Function name (Nome da função), insira `HelloFunction`.
4. Mantenha todas as seleções-padrão na página e escolha Criar função.
5. Depois que a função do Lambda foi criada, copie o nome do recurso da Amazon (ARN) da função exibido no canto superior direito da página. Para copiar o ARN, clique em

   .

   A seguir está um exemplo de ARN.

   ```
   arn:aws:lambda:us-east-1:123456789012:function:HelloFunction
   ```

6. Copie o código a seguir para a função Lambda na seção Código-fonte da ***HelloFunction***página.

   ```
   export const handler = async(event, context, callback) => {
       callback(null, "Hello from " + event.who + "!");
   };
   ```

   Esse código monta uma saudação usando o campo `who` dos dados de entrada, que são fornecidos pelo `event` objeto passado para sua função. Você poderá adicionar dados de entrada para essa função mais tarde, ao iniciar uma nova execução. O método `callback` retorna saudação montada em sua função.
7. Escolha Implantar.

Etapa 2: testar a função do Lambda

Teste a função do Lambda para vê-la em operação.

1. Escolha Testar.
2. Em Nome do evento, insira `HelloEvent`.
3. Substitua os dados do JSON do Evento pelo seguinte:

   ```
   {
       "who": "AWS Step Functions"
   }
   ```

   A entrada `"who"` corresponde ao campo `event.who` na função do Lambda e completa a saudação. Você vai inserir os mesmos dados de entrada ao executar sua máquina de estado.

4. Escolha Salvar e Testar.
5. Para analisar os resultados do teste, em Execution result (Resultado da execução), expanda Details (Detalhes).

Etapa 3: Criar uma máquina de estado

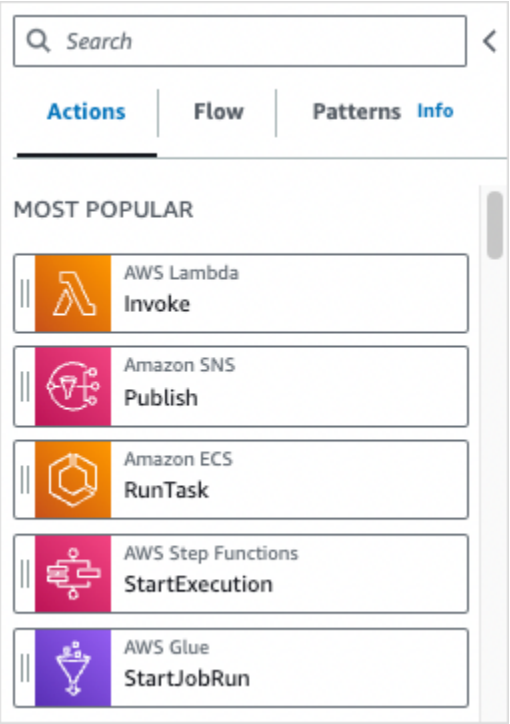
Use o console do Step Functions para criar uma máquina de estado que invoca uma função do Lambda que você criou na Etapa 1.

1. Abra o console do Step Functions e clique em Criar máquina de estado.

   > **Important**
   >
   > Certifique-se de que sua máquina de estado esteja na mesma AWS conta e região da função Lambda que você criou anteriormente.

2. Na caixa de diálogo Escolher um modelo, selecione Em branco.
3. Escolha Selecionar. Isso abre o Workflow Studio no Modo de design.
4. No States browser (Navegador de estados) à esquerda, verifique se você escolheu a guia Ações. Então, faça o seguinte:

   - Arraste e solte a API Invocação AWS Lambda no estado vazio chamado Arrastar primeiro estado aqui.



5. No painel Inspector à direita, configure a função do Lambda:

   a. Na seção Parâmetros da API, escolha a função do Lambda criada antes na lista suspensa Nome da função.

   b. retenha a seleção-padrão na lista suspensa Carga útil.

6. (Opcional) Escolha Definição para ver a definição de Amazon States Language (ASL) da máquina de estado, que é gerada automaticamente com base nas suas seleções nas guias Ações e no painel Inspetor.

7. Especifique um nome para a máquina de estado. Para fazer isso, escolha o ícone de edição ao lado do nome padrão da máquina de estado de MyStateMachine. Em seguida, em Configuração da máquina de estado, insira um nome na caixa Nome da máquina de estado.

   Por exemplo, insira o nome **`LambdaStateMachine`**.

   > **Note**
   >
   > Os nomes das máquinas de estado, execuções e tarefas de atividade não devem exceder oitenta caracteres. Esses nomes devem ser exclusivos para sua conta e AWS região e não devem conter nenhum dos seguintes itens:
   >
   > - Espaço em branco

- Caracteres curinga (`? *`)
- Caracteres de colchete (`< > { } [ ]`)
- Caracteres especiais (`" # % \ ^ | ~ ` $ & , ; : /`)
- caracteres de controle (`\\u0000` - `\\u001f` ou `\\u007f` - `\\u009f`).

Se sua máquina de estado for do tipo Express, você poderá fornecer o mesmo nome para várias execuções da máquina de estado. O Step Functions gera um ARN de execução exclusivo para cada execução de máquina de estado Express, mesmo que várias execuções tenham o mesmo nome.
Step Functions permite criar nomes para máquinas de estado, execuções e atividades, além de rótulos que contenham caracteres não ASCII. Esses nomes não ASCII não funcionam com a Amazon. CloudWatch Para garantir que você possa acompanhar CloudWatch as métricas, escolha um nome que use somente caracteres ASCII.

8. (Opcional) Em Configuração da máquina de estado, especifique outras configurações do fluxo de trabalho, como o tipo de máquina de estado e a função de execução.

   Para este tutorial, mantenha todas as seleções padrão nas Configurações da máquina de estado.

9. Escolha Criar.
10. Na caixa de diálogo Confirmar criação do perfil, selecione Confirmar para continuar.

    Você também pode escolher Exibir configurações do perfil para voltar às Configurações da máquina de estado.

> **Note**
>
> Se você excluir o perfil do IAM criado pelo Step Functions, não será possível recriá-lo posteriormente. Da mesma forma, se você modificar a função (por exemplo, removendo o Step Functions das entidades principais na política do IAM), o Step Functions não poderá restaurar as configurações originais dela posteriormente.

Etapa 4: Executar a máquina de estado

Depois que você criar uma máquina de estado, poderá executá-la.

1. Na página State Machines, escolha LambdaStateMachine.
2. Selecione Iniciar execução.

   A caixa de diálogo Iniciar execução é exibida.
3. (Opcional) Para identificar a execução, insira um nome para ela no campo Nome. Por padrão, o Step Functions cria automaticamente um nome de execução exclusivo.

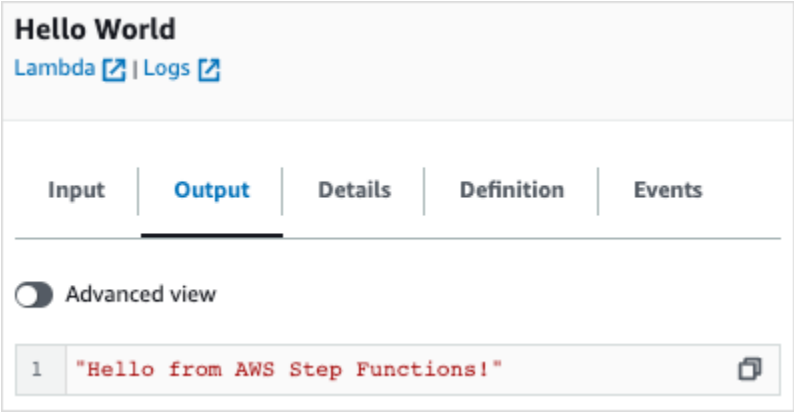
   > **Note**
   >
   > Step Functions permite criar nomes para máquinas de estado, execuções e atividades, além de rótulos que contenham caracteres não ASCII. Esses nomes não ASCII não funcionam com a Amazon. CloudWatch Para garantir que você possa acompanhar CloudWatch as métricas, escolha um nome que use somente caracteres ASCII.
4. Na área Entrada, substitua os dados de exemplo pelo que é exibido a seguir.

   ```
   {
       "who" : "AWS Step Functions"
   }
   ```

   `"who"` é o nome da chave que a função do Lambda usa para obter o nome da pessoa a ser cumprimentada.
5. Escolha Start Execution.

   A execução da máquina de estado começa e uma nova página mostrando a execução em andamento é exibida.
6. O console do Step Functions direciona você para uma página em que o título é o ID da execução. Essa página é conhecida como página de Detalhes da execução. Nesta página, você pode revisar os resultados da execução à medida que a execução avança ou após a conclusão.

   Para revisar os resultados da execução, escolha estados individuais na Exibição em gráfico e, em seguida, escolha as guias individuais no painel Detalhes da etapa para visualizar os detalhes de cada estado, incluindo entrada, saída e definição, respectivamente. Para obter detalhes sobre as informações de execução que você pode visualizar na página Detalhes da execução, consulte Página de Detalhes da execução — Visão geral da interface.



> **Note**
>
> Você também pode transmitir cargas ao invocar o Lambda de uma máquina de estado. Para obter mais informações e exemplos sobre como invocar o Lambda passando a carga no campo `Parameters`, consulte Invocar o Lambda com o Step Functions.

Tratar condições de erro usando uma máquina de estado Step Functions

Neste tutorial, você cria uma máquina de AWS Step Functions estado com um Estados de fallback campo. O `Catch` campo usa uma AWS Lambda função para responder com lógica condicional com base no tipo de mensagem de erro. Essa é uma técnica denominada tratamento de erro de função.

Para obter mais informações, consulte Erros da função AWS Lambda no Node.js no Guia do desenvolvedor do AWS Lambda .

> **Note**
>
> Você pode também criar máquinas de estado que usem Tentar novamente para os limites de tempo ou que usem `Catch` para mudar para um estado específico quando ocorre um erro ou um tempo limite é atingido. Para obter exemplos dessas técnicas de tratamento de erro, consulte Exemplos que usam retry e catch.

Neste tópico:

Etapa 1: Criar uma função do Lambda que apresenta falha

Use uma função do Lambda para simular uma condição de erro.

> **Important**
>
> Certifique-se de que sua função Lambda esteja na mesma AWS conta e AWS região da sua máquina estadual.

1. Abra o AWS Lambda console em https://console.aws.amazon.com/lambda/.
2. Escolha a opção Criar função.
3. Escolha Usar um esquema, insira `step-functions` na caixa de pesquisa e escolha o esquema Throw a custom error (Lançar um erro personalizado).
4. Em Function name (Nome da função), insira `FailFunction`.
5. Para Função, retenha a seleção-padrão (Create a new role with basic Lambda permissions (Criar uma função com permissões básicas do Lambda)).
6. O código a seguir é exibido no painel Código de função do 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!');
};
```

   O objeto `context` retorna a mensagem de erro `This is a custom error!`.
7. Escolha a opção Criar função.

8. Depois que a função do Lambda foi criada, copie o nome do recurso da Amazon (ARN) da função exibido no canto superior direito da página. Para copiar o ARN, clique em

   .

   A seguir está um exemplo de ARN.

   ```
   arn:aws:lambda:us-east-1:123456789012:function:FailFunction
   ```

9. Escolha Implantar.

Etapa 2: testar a função do Lambda

Teste a função do Lambda para vê-la em operação.

1. Na FailFunctionpágina, escolha a guia Teste e, em seguida, escolha Teste. Não é necessário criar um evento de teste.
2. Para analisar os resultados do teste, em Resultado da execução, expanda Detalhes.

Etapa 3: Criar uma máquina de estado com um campo Capturar

Use o console do Step Functions para criar uma máquina de estado que use um estado Estado da tarefa com um campo `Catch`. Adicione uma referência à função do Lambda no estado Tarefa. A máquina de estado invoca a função do Lambda, que falha durante a execução. O Step Functions tenta executar a função duas outras vezes usando recuo exponencial entre novas tentativas.

1. Abra o console do Step Functions e clique em Criar máquina de estado.
2. Na caixa de diálogo Escolher um modelo, selecione Em branco.
3. Escolha Selecionar. Isso abre o Workflow Studio no Modo de design.
4. Selecione Código para abrir o editor de código. No editor de código, você grava e edita a definição de Amazon States Language (ASL) do fluxo de trabalho.
5. Cole o código a seguir, mas substitua o ARN da função do Lambda criada antes no 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
      }
   }
}
```

Essa é uma descrição da máquina de estado usando a Amazon States Language. Ela define um estado `Task` específico denominado `CreateAccount`. Para obter mais informações, consulte Estrutura da máquina de estado.

Para obter mais informações sobre a sintaxe do campo `Retry`, consulte Exemplos de máquina de estado que usam Nova tentativa e Detecção.

> **Note**
>
> Os erros não tratados no Lambda são relatados como `Lambda.Unknown` na saída do erro. Isso inclui out-of-memory erros e tempos limite de função. Você pode combinar com `Lambda.Unknown`, `States.ALL` ou `States.TaskFailed` para lidar com esses erros. Quando o Lambda atinge o número máximo de invocações, o erro é `Lambda.TooManyRequestsException`. Para obter mais informações sobre erros da função do Lambda, consulte Tratamento de erros e novas tentativas automáticas no Guia do desenvolvedor do AWS Lambda .

6. (Opcional) No Painel de visualização gráfica, veja a visualização gráfica em tempo real do seu fluxo de trabalho.
7. Especifique um nome para a máquina de estado. Para fazer isso, escolha o ícone de edição ao lado do nome padrão da máquina de estado de MyStateMachine. Em seguida, em Configuração da máquina de estado, insira um nome na caixa Nome da máquina de estado.

   Para este tutorial, insira **`Catchfailure`**.
8. (Opcional) Em Configuração da máquina de estado, especifique outras configurações do fluxo de trabalho, como o tipo de máquina de estado e a função de execução.

   Para este tutorial, mantenha todas as seleções padrão nas Configurações da máquina de estado.
9. Na caixa de diálogo Confirmar criação do perfil, selecione Confirmar para continuar.

   Você também pode escolher Exibir configurações do perfil para voltar às Configurações da máquina de estado.

> **Note**
>
> Se você excluir o perfil do IAM criado pelo Step Functions, não será possível recriá-lo posteriormente. Da mesma forma, se você modificar a função (por exemplo, removendo o Step Functions das entidades principais na política do IAM), o Step Functions não poderá restaurar as configurações originais dela posteriormente.

Etapa 4: Executar a máquina de estado

Depois que você criar uma máquina de estado, poderá executá-la.

1. Na página Máquinas de estado, escolha Catchfailure.
2. Na página Catchfailure, escolha Iniciar execução. A caixa de diálogo Iniciar execução é exibida.
3. Na caixa de diálogo Iniciar execução, faça o seguinte:

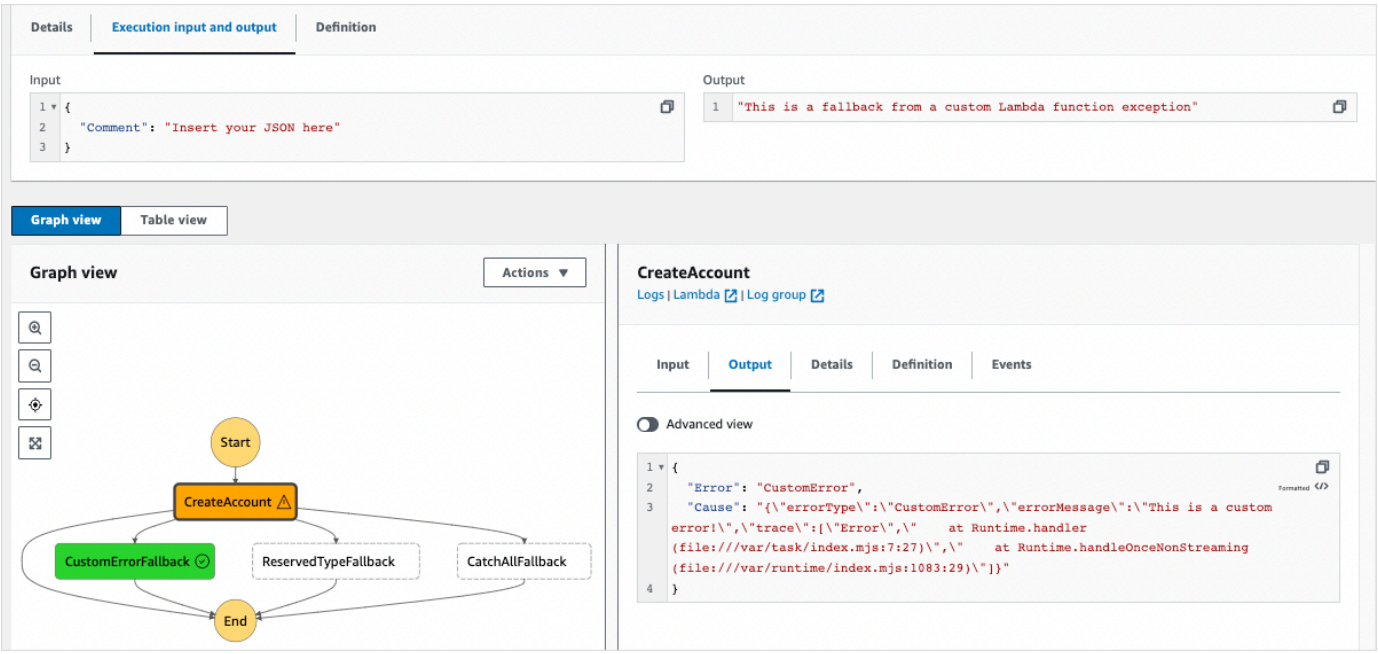
   1. (Opcional) Para identificar a execução, insira um nome para ela no campo Nome. Por padrão, o Step Functions cria automaticamente um nome de execução exclusivo.

      > **Note**
      >
      > Step Functions permite criar nomes para máquinas de estado, execuções e atividades, além de rótulos que contenham caracteres não ASCII. Esses nomes não ASCII não funcionam com a Amazon. CloudWatch Para garantir que você possa acompanhar CloudWatch as métricas, escolha um nome que use somente caracteres ASCII.

   2. (Opcional) No campo Entrada, insira os valores de entrada no formato JSON para executar o fluxo de trabalho.
   3. Selecione Iniciar execução.
   4. O console do Step Functions direciona você para uma página em que o título é o ID da execução. Essa página é conhecida como página de Detalhes da execução. Nesta página, você pode revisar os resultados da execução à medida que a execução avança ou após a conclusão.

      Para revisar os resultados da execução, escolha estados individuais na Exibição em gráfico e, em seguida, escolha as guias individuais no painel Detalhes da etapa para visualizar os detalhes de cada estado, incluindo entrada, saída e definição, respectivamente. Para obter detalhes sobre as informações de execução que você pode visualizar na página Detalhes da execução, consulte Página de Detalhes da execução — Visão geral da interface.

   Por exemplo, para visualizar sua mensagem de erro personalizada, escolha a CreateAccountetapa na visualização do gráfico e, em seguida, escolha a guia Saída.



> **Note**
>
> Você pode preservar a entrada de estado com o erro usando `ResultPath`. Consulte Use ResultPath para incluir erro e entrada em um `Catch`.

Como usar o estado Mapa em linha para repetir uma ação

Este tutorial ajuda você nos conceitos básicos ao usar o estado `Map` no modo em linha. Você usa o estado Mapa em linha em seus fluxos de trabalho para executar uma ação repetidamente. Para obter mais informações sobre o modo Em linha, consulte Estado do mapa no modo Em linha.

Neste tutorial, você usa o estado Mapa em linha para gerar repetidamente identificadores universalmente exclusivos da versão 4 (UUID v4). Você começa criando um fluxo de trabalho que contém dois estados Pass e um estado de Mapa em linha no Workflow Studio. Em seguida, você configura a entrada e a saída, incluindo a matriz JSON de entrada para o estado `Map`. O estado `Map` retorna uma matriz de saída que contém os UUIDs v4 gerados para cada item na matriz de entrada.

Conteúdo

- Etapa 3: Revisar a definição da Amazon States Language gerada automaticamente e salvar o fluxo de trabalho
- Etapa 4: Executar a máquina de estado

Etapa 1: Criar o protótipo do fluxo de trabalho

Nesta e tapa, você cria o protótipo para seu fluxo de trabalho usando o Workflow Studio. O Workflow Studio é um designer visual de fluxo de trabalho disponível no console do Step Functions. Você vai escolher os estados necessários na guia Fluxo e usar o atributo de arrastar e soltar do Workflow Studio para criar o protótipo do fluxo de trabalho.

1. Abra o console do Step Functions e clique em Criar máquina de estado.
2. Na caixa de diálogo Escolher um modelo, selecione Em branco.
3. Escolha Selecionar. Isso abre o Workflow Studio no Modo de design.
4. Na guia Fluxo, arraste o estado Aprovado e solte no estado vazio chamado Arrastar o primeiro estado aqui.
5. Arraste um estado Mapa e solte-o abaixo do estado Aprovado. Altere o nome do estado Mapa para **`Map demo`**.
6. Arraste um segundo estado Aprovado e solte-o dentro do estado Demonstração do mapa.
7. Altere o nome do segundo estado Aprovado como **`Generate UUID`**.

Etapa 2: Configurar entrada e saída

Nesta etapa, você configura a entrada e a saída para todos os estados no protótipo de fluxo de trabalho. Primeiro, você injeta alguns dados fixos no fluxo de trabalho usando o primeiro estado Aprovado. Esse estado Aprovado transmite esses dados como entrada para o estado Demonstração do mapa. Nessa entrada, você especifica o nó que contém a matriz de entrada sobre a qual o estado Demonstração do mapa deve iterar. Em seguida, você define a etapa que o estado Demonstração do mapa deve repetir para gerar os UUIDs v4. Por fim, você configura a saída para retornar para cada repetição.

1. Escolha o primeiro estado Aprovado em seu protótipo de fluxo de trabalho. Na guia Saída, insira o seguinte em Resultado:

```
{
```

```
  "foo": "bar",
  "colors": [
    "red",
    "green",
    "blue",
    "yellow",
    "white"
  ]
}
```

2. Escolha o estado Demonstração do mapa e, na guia Configuração, faça o seguinte:

   a. Escolha Provide a path to items array (Fornecer um caminho para matriz de itens).

   b. Especifique o seguinte caminho de referência para selecionar o nó que contém a matriz de entrada:

   ```
   $.colors
   ```

3. Escolha o estado Generate UUID (Gerar UUID) e, na guia Entrada, faça o seguinte:

   a. Escolha Transform input with Parameters (Transformar entrada com parâmetros).

   b. Insira a seguinte entrada JSON para gerar os UUIDs v4 para cada um dos itens da matriz de entrada. Use a função intrínseca `States.UUID` para gerar os UUIDs.

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

4. Para o estado Generate UUID (Gerar UUID), escolha a guia Saída e faça o seguinte:

   a. Escolha Filtrar saída com OutputPath.

   b. Insira o seguinte caminho de referência para selecionar o nó JSON que contém os itens da matriz de saída:

   ```
   $.uuid
   ```

Etapa 3: Revisar a definição da Amazon States Language gerada automaticamente e salvar o fluxo de trabalho

Conforme você arrasta e solta os estados do painel Fluxo na tela, o Workflow Studio cria automaticamente a definição da Amazon States Language (ASL) do fluxo de trabalho em tempo real. Você pode editar essa definição conforme necessário.

1. (Opcional) Escolha Definição no painel Inspector para ver a definição do Amazon States Language gerada automaticamente do seu fluxo de trabalho.

   > **Tip**
   >
   > Você também pode ver a definição de ASL no Editor de código do Workflow Studio. No editor de código, você também pode editar a definição de ASL do fluxo de trabalho.

   O exemplo a seguir mostra a definição da Amazon States Language gerada automaticamente para seu fluxo de trabalho.

```
{
    "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 um nome para a máquina de estado. Para fazer isso, escolha o ícone de edição ao lado do nome padrão da máquina de estado de MyStateMachine. Em seguida, em Configuração da máquina de estado, insira um nome na caixa Nome da máquina de estado.

   Para este tutorial, insira o nome **`InlineMapDemo`**.

3. (Opcional) Em Configuração da máquina de estado, especifique outras configurações do fluxo de trabalho, como o tipo de máquina de estado e a função de execução.

   Para este tutorial, mantenha todas as seleções padrão na Configuração da máquina de estado.

4. Na caixa de diálogo Confirmar criação do perfil, selecione Confirmar para continuar.

   Você também pode escolher Exibir configurações do perfil para voltar às Configurações da máquina de estado.

> **Note**
>
> Se você excluir o perfil do IAM criado pelo Step Functions, não será possível recriá-lo posteriormente. Da mesma forma, se você modificar a função (por exemplo, removendo o Step Functions das entidades principais na política do IAM), o Step Functions não poderá restaurar as configurações originais dela posteriormente.

Etapa 4: Executar a máquina de estado

As execuções de máquinas de estado são instâncias em que o fluxo de trabalho é executado para a realização de tarefas.

1. Na InlineMapDemopágina, escolha Iniciar execução.
2. Na caixa de diálogo Iniciar execução, faça o seguinte:

   1. (Opcional) Para identificar a execução, insira um nome para ela no campo Nome. Por padrão, o Step Functions gera automaticamente um nome de execução exclusivo.

      > **Note**
      >
      > O Step Functions permite criar nomes de máquina de estado, execuções, atividades e labels que contenham caracteres não ASCII. Esses nomes não ASCII não funcionam com a Amazon. CloudWatch Para garantir que você possa acompanhar CloudWatch as métricas, escolha um nome que use somente caracteres ASCII.

   2. (Opcional) No campo Entrada, insira os valores de entrada no formato JSON para executar o fluxo de trabalho.
   3. Selecione Iniciar execução.
   4. O console do Step Functions direciona você para uma página em que o título é o ID da execução. Essa página é conhecida como página de Detalhes da execução. Nesta página, você pode revisar os resultados da execução à medida que a execução avança ou após a conclusão.

      Para revisar os resultados da execução, escolha estados individuais na Exibição em gráfico e, em seguida, escolha as guias individuais no painel Detalhes da etapa para visualizar os detalhes de cada estado, incluindo entrada, saída e definição, respectivamente. Para obter detalhes sobre as informações de execução que você pode visualizar na página Detalhes da execução, consulte Página de Detalhes da execução — Visão geral da interface.

   Para visualizar a entrada e a saída da execução side-by-side, escolha Entrada e saída de execução. Em Saída, veja a matriz de saída retornada pelo estado `Map`. A seguir, veja um exemplo de matriz de saída:

```
[
```

```
  "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 dados CSV em grande escala usando o Mapa Distribuído

Este tutorial ajuda você a começar a usar o estado `Map` no modo distribuído. Um estado `Map` definido como Distribuído é conhecido como estado Mapa distribuído. Você usa o estado Mapa distribuído em seus fluxos de trabalho para iterar em fontes de dados de grande escala do Amazon S3. O estado `Map` executa cada iteração como uma execução de fluxo de trabalho secundário, o que permite alta simultaneidade. Para obter mais informações sobre o modo distribuído, consulte estado Mapa no modo distribuído.

Neste tutorial, você usa o estado Mapa distribuído para iterar sobre um arquivo CSV em um bucket do Amazon S3. Em seguida, você retorna seu conteúdo com o ARN da execução de um fluxo de trabalho secundário em outro bucket do Amazon S3. Você começa criando um protótipo de fluxo de trabalho no Workflow Studio. Em seguida, você define o modo de processamento do estado `Map` como Distribuído, especifica o arquivo CSV como conjunto de dados e fornece sua localização ao estado `Map`. Você também especifica o tipo de fluxo de trabalho para as execuções do fluxo de trabalho secundário que o estado Mapa distribuído começa como Express.

Além dessas configurações, você também especifica outras configurações, como o número máximo de execuções simultâneas de fluxos de trabalho secundários e o local para exportar o resultado `Map`, para o exemplo de fluxo de trabalho usado neste tutorial.

Conteúdos

- Etapa 6: Revisar a definição da Amazon States Language gerada automaticamente e salvar o fluxo de trabalho
- Etapa 7: Executar a máquina de estado

Pré-requisitos

- Faça upload de um arquivo CSV para bucket do Amazon S3. Você deve definir uma linha de cabeçalho no seu arquivo CSV. Para obter informações sobre os limites de tamanho impostos ao arquivo CSV e como especificar a linha do cabeçalho, consulte Arquivo CSV em um bucket do Amazon S3.
- Crie outro bucket do Amazon S3 e uma pasta dentro desse bucket para a qual exportar o resultado do estado `Map`.

> **Important**
>
> Certifique-se de que seus buckets do Amazon S3 estejam sob a mesma máquina de estado Conta da AWS e Região da AWS sob a mesma.

Etapa 1: Criar o protótipo do fluxo de trabalho

Nesta e tapa, você cria o protótipo para seu fluxo de trabalho usando o Workflow Studio. O Workflow Studio é um designer visual de fluxo de trabalho disponível no console do Step Functions. Você escolhe o estado necessário e a ação da API nas guias Fluxo e Ações, respectivamente. Você vai usar o atributo de arrastar e soltar do Workflow Studio para criar o protótipo do fluxo de trabalho.

1. Abra o console do Step Functions e clique em Criar máquina de estado.
2. Na caixa de diálogo Escolher um modelo, selecione Em branco.
3. Escolha Selecionar. Isso abre o Workflow Studio no Modo de design.
4. Na guia Fluxo, arraste o estado Mapa e solte no estado vazio chamado Arrastar o primeiro estado aqui.
5. Na guia Configuração, em Nome do estado, insira **`Process data`**.
6. Na guia Ações, arraste uma ação da API Invocação AWS Lambda e solte-a dentro do estado Processar dados.
7. Altere o nome do estado Invocar AWS Lambda para **`Process CSV data`**.

Etapa 2: Configurar os campos necessários para o estado Mapa

Nesta etapa, você configura os seguintes campos obrigatórios do estado Mapa Distribuído (Mapa distribuído):

- ItemReader— Especifica o conjunto de dados e sua localização a partir da qual o `Map` estado pode ler a entrada.
- ItemProcessor: especifica os seguintes valores:
  - `ProcessorConfig`: define o `Mode` e o `ExecutionType` como `DISTRIBUTED` e `EXPRESS`, respectivamente. Isso define o modo de processamento do estado `Map` e o tipo de fluxo de trabalho para execuções de fluxo de trabalho secundário que o estado Mapa distribuído inicia.
  - `StartAt`: o primeiro estado no fluxo de trabalho do Mapa.
  - `States`: define o fluxo de trabalho Mapear, que é um conjunto de etapas a serem repetidas em cada execução do fluxo de trabalho secundário.
- ResultWriter— Especifica a localização do Amazon S3 onde o Step Functions grava os resultados do estado do Distributed Map.

> **Important**
>
> Certifique-se de que o bucket do Amazon S3 que você usa para exportar os resultados de uma execução de mapa esteja abaixo da mesma máquina de estado Conta da AWS e na Região da AWS mesma. Caso contrário, a execução da sua máquina de estado falhará com o erro `States.ResultWriterFailed`.

Para configurar os campos obrigatórios:

1. Escolha o estado Process data (Dados processo) e, na guia Configuração, faça o seguinte:
   a. Para Modo de processamento, escolha Distribuído.
   b. Em Item source (Origem do item), escolha Amazon S3 e CSV file in S3 (Arquivo CSV no S3) na lista suspensa S3 item source (Origem do item do S3).
   c. Faça o seguinte para especificar a localização do seu arquivo CSV no Amazon S3:
      i. Para Objeto do S3, selecione Enter bucket and key (Inserir bucket e chave) na lista suspensa.

ii. Para Bucket, insira o nome do bucket do Amazon S3 que contém o arquivo CSV. Por exemplo, **`sourceBucket`**.

iii. Em Chave, insira o nome do objeto Amazon S3 no qual você salvou o arquivo CSV. Você também deve especificar o nome do arquivo CSV nesse campo. Por exemplo, **`csvDataset/ratings.csv`**.

d. Para arquivos CSV, também é necessário especificar a local do cabeçalho da coluna. Para fazer isso, escolha Configuração adicional e, para CSV header location (Localização do cabeçalho CSV), retenha a seleção-padrão de First row (Primeira linha) se a primeira linha do seu arquivo CSV for o cabeçalho. Caso contrário, escolha Given (Fornecido) para especificar o cabeçalho na definição da máquina de estado. Para ter mais informações, consulte `ReaderConfig`.

e. Para o Child execution type (Tipo de execução secundária), escolha Express.

2. Em Local de exportação, para exportar os resultados da Execução do mapa para um local específico do Amazon S3, escolha Exportar a saída do estado do mapa para o Amazon S3.

3. Faça o seguinte:

a. Para Bucket do S3, escolha Enter bucket name and prefix (Inserir nome e prefixo do bucket) na lista suspensa.

b. Para Bucket, insira o nome do bucket do Amazon S3 para o qual você quer exportar os resultados. Por exemplo, **`mapOutputs`**.

c. Em Prefixo, insira o nome da pasta na qual você deseja salvar os resultados. Por exemplo, **`resultData`**.

Etapa 3: Configurar opções adicionais

Além das configurações necessárias para um estado Mapa Distribuído (Mapa Distribuído), você também pode especificar outras opções. Isso pode incluir o número máximo de execuções simultâneas de fluxos de trabalho secundários e o local para o qual exportar o resultado do estado `Map`.

1. Escolha o estado Process data (Processar dados). Em seguida, em Origem do item, escolha Configuração adicional.

2. Faça o seguinte:

a. Escolha Modificar itens com ItemSelector para especificar uma entrada JSON personalizada para cada execução do fluxo de trabalho secundário.

b. Insira a seguinte entrada JSON:

```
{
  "index.$": "$$.Map.Item.Index",
  "value.$": "$$.Map.Item.Value"
}
```

Para obter informações sobre como criar uma entrada personalizada, consulte `ItemSelector`.

3. Nas Configurações de tempo de execução, em Limite de simultaneidade, especifique o número de execuções simultâneas de fluxo de trabalho secundário que o estado Mapa distribuído pode iniciar. Por exemplo, digite **`100`**.
4. Abra uma nova janela ou guia no seu navegador e conclua a configuração da função do Lambda que você usará nesse fluxo de trabalho, conforme explicado em Etapa 4: Configurar uma função do Lambda.

Etapa 4: Configurar uma função do Lambda

> **Important**
>
> Certifique-se de que sua função Lambda esteja sob a Região da AWS mesma que sua máquina de estado.

1. Abra o console do Lambda e clique em Criar função.
2. Na página Create function, selecione Author from scratch.
3. Na seção Informações básicas, configure a função do Lambda:

a. Em Function name (Nome da função), insira **`distributedMapLambda`**.

b. Em Runtime, selecione Node.js 16.x.

c. Mantenha todas as seleções padrão e escolha Criar função.

d. Depois de criar a função do Lambda, copie o nome do recurso da Amazon (ARN) da função exibido no canto superior direito da página. Você precisará

fornecer isso em seu protótipo de fluxo de trabalho. Para copiar o ARN, clique em

.

A seguir está um exemplo de ARN.

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

4. Copie o código a seguir para a função Lambda e cole-o na seção Código-fonte da 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. Escolha Implantar. Depois que sua função for implantada, escolha Testar para ver a saída da sua função do Lambda.

Etapa 5: Atualizar o protótipo do fluxo de trabalho

No console do Step Functions, você atualizará seu fluxo de trabalho para adicionar o ARN da função do Lambda.

1. Volte para a guia ou janela em que você criou o protótipo do fluxo de trabalho.
2. Escolha a etapa Process CSV data (Processar dados CSV) e, na guia Configuração, faça o seguinte:

   a. Em Tipo de integração, escolha Otimizado.

   b. Para Nome da função, insira o nome da função do Lambda. Escolha a função na lista suspensa exibida ou escolha Enter function name (Inserir nome da função) e forneça o ARN da função do Lambda.

Etapa 6: Revisar a definição da Amazon States Language gerada automaticamente e salvar o fluxo de trabalho

Conforme você arrasta e solta os estados das guias Ação e Fluxo para a tela, o Workflow Studio cria automaticamente a definição da Amazon States Language do fluxo de trabalho em tempo real. Você pode editar essa definição conforme necessário.

1. (Opcional) Clique em Definição no painel Inspector e veja a definição da máquina de estado.

> **Tip**
>
> Você também pode ver a definição de ASL no Editor de código do Workflow Studio. No editor de código, você também pode editar a definição de ASL do fluxo de trabalho.

O código de exemplo a seguir mostra a definição da Amazon States Language gerada automaticamente para seu fluxo de trabalho.

```
{
  "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 um nome para a máquina de estado. Para fazer isso, escolha o ícone de edição ao lado do nome padrão da máquina de estado de MyStateMachine. Em seguida, em Configuração da máquina de estado, insira um nome na caixa Nome da máquina de estado.

   Para este tutorial, insira o nome **`DistributedMapDemo`**.

3. (Opcional) Em Configuração da máquina de estado, especifique outras configurações do fluxo de trabalho, como o tipo de máquina de estado e a função de execução.

   Para este tutorial, mantenha todas as seleções padrão na Configuração da máquina de estado.

4. Na caixa de diálogo Confirmar criação do perfil, selecione Confirmar para continuar.

Você também pode escolher Exibir configurações do perfil para voltar às Configurações da máquina de estado.

> **Note**
>
> Se você excluir o perfil do IAM criado pelo Step Functions, não será possível recriá-lo posteriormente. Da mesma forma, se você modificar a função (por exemplo, removendo o Step Functions das entidades principais na política do IAM), o Step Functions não poderá restaurar as configurações originais dela posteriormente.

Etapa 7: Executar a máquina de estado

Uma execução é uma instância da máquina de estado em que você executa o fluxo de trabalho para realizar tarefas.

1. Na DistributedMapDemopágina, escolha Iniciar execução.
2. Na caixa de diálogo Iniciar execução, faça o seguinte:
    1. (Opcional) Para identificar a execução, insira um nome para ela no campo Nome. Por padrão, o Step Functions gera automaticamente um nome de execução exclusivo.

        > **Note**
        >
        > O Step Functions permite criar nomes de máquina de estado, execuções, atividades e labels que contenham caracteres não ASCII. Esses nomes não ASCII não funcionam com a Amazon. CloudWatch Para garantir que você possa acompanhar CloudWatch as métricas, escolha um nome que use somente caracteres ASCII.

    2. (Opcional) No campo Entrada, insira os valores de entrada no formato JSON para executar o fluxo de trabalho.
    3. Selecione Iniciar execução.
    4. O console do Step Functions direciona você para uma página em que o título é o ID da execução. Essa página é conhecida como página de Detalhes da execução. Nesta página, você pode revisar os resultados da execução à medida que a execução avança ou após a conclusão.

Para revisar os resultados da execução, escolha estados individuais na Exibição em gráfico e, em seguida, escolha as guias individuais no painel Detalhes da etapa para visualizar os detalhes de cada estado, incluindo entrada, saída e definição, respectivamente. Para obter detalhes sobre as informações de execução que você pode visualizar na página Detalhes da execução, consulte Página de Detalhes da execução — Visão geral da interface.

Por exemplo, escolha o estado `Map` e escolha Execução do mapa para abrir a página Detalhes da execução do mapa. Nessa página, você pode visualizar todos os detalhes da execução do estado Mapa distribuído e as execuções do fluxo de trabalho secundário que ele iniciou. Para obter informações sobre essa página, consulte Examinando o Map Run.

Como processar um lote inteiro de dados em uma função do Lambda

Neste tutorial, você usa o campo ItemBatcher do estado Mapa distribuído para processar um lote inteiro de itens dentro de uma função do Lambda. Cada lote contém até três itens. O estado Mapa distribuído inicia quatro execuções secundárias do fluxo de trabalho, em que cada execução processa três itens, enquanto uma execução processa um só item. A execução de cada fluxo de trabalho secundário invoca uma função do Lambda que itera os itens individuais presentes no lote.

Você vai criar uma máquina de estado que executa a multiplicação em uma matriz de números inteiros. Digamos que a matriz de números inteiros que você fornece como entrada é `[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]` e o fator de multiplicação é `7`. Então, a matriz resultante formada após a multiplicação desses números inteiros por um fator de sete será `[7, 14, 21, 28, 35, 42, 49, 56, 63, 70]`.

Tópicos

- Etapa 1: Criar a máquina de estado
- Etapa 2: Criar a função do Lambda
- Etapa 3: Executar a máquina de estado

Etapa 1: Criar a máquina de estado

Nesta etapa, você cria o protótipo de fluxo de trabalho da máquina de estado que passa um lote inteiro de dados para a função do Lambda que você vai criar na Etapa 2.

- Use a definição a seguir para criar uma máquina de estado usando o console do Step Functions. Para obter informações sobre como criar uma máquina de estado, consulte Etapa 1: Criar o protótipo do fluxo de trabalho no tutorial Introdução ao uso do estado Mapa Distribuído.

  Nessa máquina de estado, você define um estado Mapa Distribuído que aceita uma matriz de 10 números inteiros como entrada e passa essa matriz para uma função do Lambda em lotes do 3. A função do Lambda itera os itens individuais presentes no lote e retorna uma matriz de saída chamada `multiplied`. A matriz de saída contém o resultado da multiplicação realizada nos itens passados na matriz de entrada.

  > **Important**
  >
  > Substitua o Nome do recurso da Amazon (ARN) da função do Lambda no código a seguir pelo ARN da função que você vai criar na Etapa 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"
    }
  }
}
```

Etapa 2: Criar a função do Lambda

Nesta etapa, você vai criar a função do Lambda que processa todos os itens passados no lote.

> **Important**
>
> Certifique-se de que sua função Lambda esteja sob a Região da AWS mesma que sua máquina de estado.

Para criar a função do Lambda

1. Use o console do Lambda para criar uma função do Lambda do Python 3.9 chamada **`ProcessEntireBatch`**. Para obter informações sobre como criar uma função do Lambda, consulte Etapa 4: Configurar a função do Lambda no tutorial Introdução ao uso do estado Mapa Distribuído.
2. Copie o código a seguir para a função do Lambda e cole-o na seção Origem do código da função do 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. Depois de criar a função do Lambda, copie o ARN da função exibido no canto superior direito da página. Para copiar o ARN, clique no

   .

   Veja a seguir um exemplo de ARN, em que *`function-name`* é o nome da função do Lambda (nesse caso, `ProcessEntireBatch`):

   ```
   arn:aws:lambda:us-east-1:123456789012:function:function-name
   ```

   Você precisará fornecer a função do ARN na máquina de estado criada na Etapa 1.
4. Escolha Implantar para implantar essas alterações.

Etapa 3: Executar a máquina de estado

Quando você executa a máquina de estado, o estado Mapa Distribuído inicia quatro execuções secundárias do fluxo de trabalho, em que cada execução processa três itens, enquanto uma execução processa um só item.

O exemplo a seguir mostra os dados que uma das execuções do fluxo de trabalho secundário passa para a função do `ProcessEntireBatch`.

```
{
  "BatchInput": {
    "MyMultiplicationFactor": 7
  },
  "Items": [1, 2, 3]
}
```

Com essa entrada, o exemplo a seguir mostra a matriz de saída chamada `multiplied` que a função do Lambda retorna.

```
{
  "statusCode": 200,
  "multiplied": [7, 14, 21]
}
```

A máquina de estado retorna a seguinte saída que contém quatro matrizes chamadas `multiplied` para as quatro execuções do fluxo de trabalho secundário. Essas matrizes contêm os resultados da multiplicação dos itens de entrada individuais.

```
[
  {
    "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 os itens da matriz retornados em uma só matriz de saída, use o campo ResultSelector. Defina esse campo dentro do estado Mapa Distribuído para encontrar todas as matrizes `multiplied`, extrair todos os itens dentro dessas matrizes e combiná-los em uma só matriz de saída.

Para usar o campo `ResultSelector`, atualize a definição da máquina de estado como mostra o exemplo a seguir.

```
{
  "StartAt": "Pass",
  "States": {
    ...
    ...
    "Map": {
      "Type": "Map",
      ...
      ...
      "ItemsPath": "$.MyItems",
      "ResultSelector": {
        "multiplied.$": "$..multiplied[*]"
      }
    }
  }
}
```

A máquina de estado atualizada retorna uma matriz de saída consolidada, como mostra o exemplo a seguir.

```
{
  "multiplied": [7, 14, 21, 28, 35, 42, 49, 56, 63, 70]
}
```

Como processar de itens de dados individuais com uma função do Lambda

Neste tutorial, você usa o campo ItemBatcher do estado Mapa Distribuído (Mapa distribuído) para processar um lote inteiro de itens dentro de uma função do Lambda. O estado Mapa Distribuído (Mapa distribuído) inicia quatro execuções do fluxo de trabalho secundário. Cada um desses fluxos de trabalho secundários executa um estado Mapa inline. Para cada iteração, o estado Mapa inline invoca uma função do Lambda e passa um só item do lote para a função. A função do Lambda então processa o item e retorna o resultado.

Você vai criar uma máquina de estado que executa a multiplicação em uma matriz de números inteiros. Digamos que a matriz de números inteiros que você fornece como entrada é `[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]` e o fator de multiplicação é `7`. Então, a matriz resultante formada após a multiplicação desses números inteiros por um fator de sete será `[7, 14, 21, 28, 35, 42, 49, 56, 63, 70]`.

Tópicos

Etapa 1: Criar a máquina de estado

Nesta etapa, você cria o protótipo de fluxo de trabalho da máquina de estado que passa um só item de um lote de itens a cada invocação da função do Lambda que você vai criar na Etapa 2.

- Use a definição a seguir para criar uma máquina de estado usando o console do Step Functions. Para obter informações sobre como criar uma máquina de estado, consulte Etapa 1: Criar o protótipo do fluxo de trabalho no tutorial Introdução ao uso do estado Mapa Distribuído.

  Nessa máquina de estado, você define um estado de Mapa distribuído que aceita uma matriz de 10 números inteiros como entrada e passa esses itens de matriz para as execuções do fluxo de trabalho secundário em lotes. Cada execução de fluxo de trabalho secundário recebe um lote de três itens como entrada e executa um estado Mapa inline. Cada iteração do estado Mapa inline invoca uma função do Lambda e passa um só item do lote para a função. Essa função então multiplica o item por um fator de 7 e retorna o resultado.

A saída da execução de cada fluxo de trabalho secundário é uma matriz JSON que contém o resultado da multiplicação de cada um dos itens passados.

> **Important**
>
> Substitua o Nome do recurso da Amazon (ARN) da função do Lambda no código a seguir pelo ARN da função que você vai criar na Etapa 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"
        }
      }
    }
  }
```

```
}
```

Etapa 2: Criar a função do Lambda

Nesta etapa, você vai criar a função do Lambda que processa cada item passado do lote.

> **Important**
>
> Certifique-se de que sua função Lambda esteja sob a Região da AWS mesma que sua máquina de estado.

Para criar a função do Lambda

1. Use o console do Lambda para criar uma função do Lambda do Python 3.9 chamada **`ProcessSingleItem`**. Para obter informações sobre como criar uma função do Lambda, consulte Etapa 4: Configurar a função do Lambda no tutorial Introdução ao uso do estado Mapa Distribuído.
2. Copie o código a seguir para a função do Lambda e cole-o na seção Origem do código da função do 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. Depois de criar a função do Lambda, copie o ARN da função exibido no canto superior direito da página. Para copiar o ARN, clique no  .

Veja a seguir um exemplo de ARN, em que *function-name* é o nome da função do Lambda (nesse caso, `ProcessSingleItem`):

```
arn:aws:lambda:us-east-1:123456789012:function:function-name
```

Você precisará fornecer a função do ARN na máquina de estado criada na Etapa 1.

4. Escolha Implantar para implantar essas alterações.

Etapa 3: Executar a máquina de estado

Quando você executa a máquina de estado, o estado Mapa Distribuído inicia quatro execuções secundárias do fluxo de trabalho, em que cada execução processa três itens, enquanto uma execução processa um só item.

O exemplo a seguir mostra os dados passados a uma das invocações de função `ProcessSingleItem` dentro de uma execução de fluxo de trabalho secundário.

```
{
  "MyMultiplicationFactor": 7,
  "MyItem": 1
}
```

Com essa entrada, o exemplo a seguir mostra a saída que a função do Lambda retorna.

```
{
  "statusCode": 200,
  "multiplied": 7
}
```

Veja a seguir um exemplo de saída da matriz JSON para uma das execuções do fluxo de trabalho secundário.

```
[
  {
    "statusCode": 200,
    "multiplied": 7
  },
  {
```

```
    "statusCode": 200,
    "multiplied": 14
  },
  {
    "statusCode": 200,
    "multiplied": 21
  }
]
```

A máquina de estado retorna a seguinte saída que contém quatro matrizes para as quatro execuções do fluxo de trabalho secundário. Essas matrizes contêm os resultados da multiplicação dos itens de entrada individuais.

Por fim, a saída da máquina de estado é uma matriz chamada `multiplied` que combina todos os resultados de multiplicação retornados para as quatro execuções do fluxo de trabalho secundário.

```
[
  [
    {
      "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 os resultados de multiplicação retornados pelas execuções do fluxo de trabalho secundário em uma só matriz de saída, você pode usar o campo ResultSelector. Defina esse campo dentro do estado Mapa Distribuído (Mapa Distribuído) para encontrar todos os resultados, extrair os resultados individuais e combiná-los em uma só matriz de saída chamada `multiplied`.

Para usar o campo `ResultSelector`, atualize a definição da máquina de estado como mostra o exemplo a seguir.

```
{
  "StartAt": "Pass",
  "States": {
    ...
    ...
    "Map": {
      "Type": "Map",
      ...
      ...
      "ItemBatcher": {
        "MaxItemsPerBatch": 3,
```

```
        "BatchInput": {
          "MyMultiplicationFactor.$": "$.MyMultiplicationFactor"
        }
      },
      "ItemsPath": "$.MyItems",
      "ResultSelector": {
        "multiplied.$": "$..multiplied"
      }
    }
  }
}
```

A máquina de estado atualizada retorna uma matriz de saída consolidada, como mostra o exemplo a seguir.

```
{
  "multiplied": [7, 14, 21, 28, 35, 42, 49, 56, 63, 70]
}
```

Iniciar a execução de uma máquina de estado em resposta a eventos do Amazon S3

É possível executar uma máquina de estado do AWS Step Functions em resposta a uma regra do Amazon EventBridge.

Este tutorial mostra a você como configurar uma máquina de estado como destino para uma regra do Amazon EventBridge. Essa regra iniciará a execução da máquina de estados quando forem adicionados arquivos a um bucket do Amazon Simple Storage Service (Amazon S3).

Para uma aplicação prática, você pode iniciar uma máquina de estado que executa operações em arquivos que você adiciona ao bucket, como a criação de miniaturas ou a execução de análises do Amazon Rekognition em arquivos de imagem e de vídeo.

Neste tutorial, você começa a execução de uma máquina de estado `Helloworld` fazendo o upload de um arquivo para um bucket do Amazon S3. Em seguida, você revisa o exemplo de entrada dessa execução para identificar as informações incluídas na entrada da notificação de eventos do Amazon S3 entregue ao EventBridge.

Tópicos

Pré-requisito: criar uma máquina de estado

Antes de configurar uma máquina de estado como um destino do Amazon EventBridge, você deve criar uma máquina de estado.

- Para criar uma máquina de estado básica, use o tutorial Criar uma máquina de estado que usa uma função do Lambda.
- Se você já tiver uma máquina de estado `Helloworld`, vá para a próxima etapa.

Etapa 1: Criar um bucket no Amazon S3

Agora que você tem uma máquina de estado `Helloworld`, precisa criar um bucket do Amazon S3 que armazene seus arquivos. Na Etapa 3 deste tutorial, você vai configurar uma regra de modo que, quando um arquivo for carregado para esse bucket, o EventBridge acionará uma execução da máquina de estado.

1. Navegue até o console do Amazon S3 e escolha Criar bucket para criar o bucket no qual armazenar seus arquivos e acione uma regra de evento do Amazon S3.
2. Insira um Bucket name (Nome de bucket), como *`username`*`-sfn-tutorial`.

   > **Note**
   >
   > Os nomes de bucket devem ser exclusivos em todos os nomes de bucket existentes em todas as Regiões da AWS no Amazon S3. Use seu próprio *`nome de usuário`* para tornar esse nome exclusivo. É necessário criar todos os recursos na mesma região da AWS.

3. Mantenha todas as seleções padrão na página e escolha Criar bucket.

Etapa 2: Habilitar a notificação de eventos do Amazon S3 com o EventBridge

Depois de criar o bucket do Amazon S3, configure-o para enviar eventos para o EventBridge sempre que determinados eventos acontecerem no bucket do S3, como uploads de arquivos.

1. Acesse o console do Amazon S3.
2. Na lista Buckets, escolha o nome do bucket para o qual você deseja habilitar eventos.
3. Escolha Properties (Propriedades).
4. Role a página para baixo para ver a seção Notificações de eventos e escolha Editar na subseção Amazon EventBridge.
5. Em Enviar notificações para o Amazon EventBridge para todos os eventos deste bucket, escolha Ativado.
6. Escolha Save changes (Salvar alterações).

> **Note**
>
> Depois que você habilitar o EventBridge, leva cerca de cinco minutos para que as alterações sejam implementadas.

Etapa 3: Criar uma regra do Amazon EventBridge

Quando você tiver uma máquina de estado, tiver criado o bucket do Amazon S3 e configurado-o para enviar notificações de eventos para o EventBridge, crie uma regra do EventBridge.

> **Note**
>
> Você deve configurar a regra do EventBridge na mesma Região da AWS que o bucket do Amazon S3.

Como criar a regra do

1. No console do Amazon EventBridge, escolha Criar regra.

> **Tip**
>
> Como alternativa, no painel de navegação no console do Eventbridge, escolha Regras em Barramentos e Criar regra.

2. Insira um Nome (por exemplo, *S3Step Functions*) e, opcionalmente, uma Descrição para a regra.
3. Para Barramento de eventos e Tipo de regra, mantenha as seleções padrão.
4. Clique em Próximo. Isso abre a página Criar padrão de evento.
5. Role para baixo até a seção Padrão de eventos e faça o seguinte:
   a. Em Origem do evento, retenha a seleção-padrão de Eventos da AWS ou de parceiros do EventBridge.
   b. Em Serviço da AWS, escolha Serviço de armazenamento simples (S3).
   c. Para Tipo de evento, escolha Notificação de eventos do Amazon S3.
   d. Selecione Eventos específicos e Objeto criado.
   e. Selecione Buckets específicos por nome e digite o nome do bucket criado na Etapa 1 (*username*`-sfn-tutorial`) para armazenar seus arquivos.
   f. Clique em Próximo. Isso abre a página Selecionar destinos.

Para criar o destino

1. Em Destino 1, retenha a seleção-padrão do serviço da AWS.
2. Na lista suspensa Selecionar um destino, selecione Máquina de estado do Step Functions.
3. Na lista Máquina de estado, selecione a máquina de estado que você criou antes (por exemplo, `Helloworld`).
4. Mantenha todas as seleções padrão na página e escolha Próximo. A página Configurar tags é aberta.
5. Escolha Next (Próximo) novamente. A página Revisar e criar é aberta.
6. Analise os detalhes da regra e escolha Criar regra.

   A regra é criada e a página Regras é exibida, listando todas as regras para o Amazon EventBridge.

Etapa 4: Testar a regra

Agora que tudo está no lugar, teste a adição de um arquivo ao bucket do Amazon S3 e examine a entrada da execução da máquina de estado.

1. Adicione um arquivo ao bucket do Amazon S3.

   Navegue até o console do Amazon S3, selecione o bucket que você criou para armazenar arquivos (*username*`-sfn-tutorial`) e selecione Fazer upload.
2. Adicione um arquivo, *`test.png`* por exemplo, e escolha Carregar.

   Isso inicia uma execução de sua máquina de estado, enviando informações do AWS CloudTrail como a entrada.
3. Verifique a execução de sua máquina de estado.

   Navegue até o console do Step Functions e selecione a máquina de estado usada na regra do Amazon EventBridge (`Helloworld`).
4. Selecione a execução mais recente da máquina de estado e expanda a seção Entrada de execução.

   Esta entrada inclui informações como o nome do bucket e o nome do objeto. Em um caso de uso do mundo real, uma máquina de estado pode usar essa entrada para executar ações nesse objeto.

Exemplo de entrada de execução

O exemplo a seguir mostra uma entrada típica para a execução da 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"
    }
  }
```

Criar uma API do Step Functions usando o API Gateway

Você pode usar o Amazon API Gateway para associar suas AWS Step Functions APIs a métodos em uma API do API Gateway. Quando uma solicitação HTTPS for enviada para um método de API, o API Gateway invoca as ações de sua API do Step Functions.

Este tutorial mostra como criar uma API que usa um único recurso e o método `POST` para se comunicar com a ação de API `StartExecution`. Você usará o console AWS Identity and Access Management (IAM) para criar uma função para o API Gateway. Em seguida, você usará o console do API Gateway para criar uma API do API Gateway, criar um recurso e um método e mapear o método para a ação de API `StartExecution`. Ao final, você implantará e testará sua API.

> **Note**
>
> Embora o Amazon API Gateway possa iniciar uma execução do Step Functions chamando `StartExecution`, você deve chamar `DescribeExecution` para obter o resultado.

Tópicos

- Etapa 1: Criar um perfil do IAM para o API Gateway
- Etapa 2: Criar a API no API Gateway
- Etapa 3: Testar e implantar a API do API Gateway

Etapa 1: Criar um perfil do IAM para o API Gateway

Antes de criar a API do API Gateway, você precisa permitir que o API Gateway chame ações de API do Step Functions.

Para configurar permissões para o API Gateway

1. Faça login no console do IAM e escolha Perfis, Criar perfil.
2. Na página Select trusted entity (Selecionar entidade confiável), faça o seguinte:
   a. Para o Tipo de entidade confiável, retenha a seleção-padrão de AWS service (Serviço da AWS).
   b. Em Caso de uso, escolha API Gateway na lista suspensa.
3. Selecione API Gateway e escolha Próximo.
4. Na página Adicionar permissões, escolha Próximo.
5. (Opcional) Na página Nomear, revisar e criar, insira detalhes, como o nome da função. Por exemplo, digite **`APIGatewayToStepFunctions`**.
6. Selecione Criar função.

   O perfil do IAM é exibido na lista de perfis.
7. Escolha o nome da sua função e anote o Role ARN (ARN da função), conforme mostrado no exemplo a seguir.

   ```
   arn:aws:iam::123456789012:role/APIGatewayToStepFunctions
   ```

Para anexar uma política ao perfil do IAM

1. Na página Roles (Funções), procure sua função (`APIGatewayToStepFunctions`) e escolha-a.
2. Na guia Permissões, escolha Adicionar permissões e Anexar políticas.
3. Na página Anexar política, procure `AWSStepFunctionsFullAccess`, escolha a política e escolha Adicionar permissões.

Etapa 2: Criar a API no API Gateway

Depois de criar o perfil do IAM, poderá criar uma API personalizada do API Gateway.

Para criar a API

1. Abra o console do Amazon API Gateway e, depois, selecione Criar API.
2. Na página Escolher um tipo de API, no painel API REST, escolha Criar.
3. Na página Criar API REST, selecione Nova API e, em seguida, insira ***StartExecutionAPI*** como nome da API.
4. Mantenha o Tipo de endpoint da API como Regional e, depois, selecione Criar API.

Para criar um recurso

1. Na página Recursos da ***StartExecutionAPI***, escolha Criar recurso.
2. Na página Criar recurso, insira **`execution`** para Nome do recurso e, depois, selecione Criar recurso.

Para criar um método POST

1. Selecione o recurso /execution e, depois, Criar método.
2. Em Tipo de método, selecione `POST`.
3. Em Tipo de integração, selecione Serviço da AWS .
4. Em Região da AWS, selecione uma região na lista.

> **Note**
>
> Para as Regiões que dão suporte ao Step Functions, consulte Regiões compatíveis.

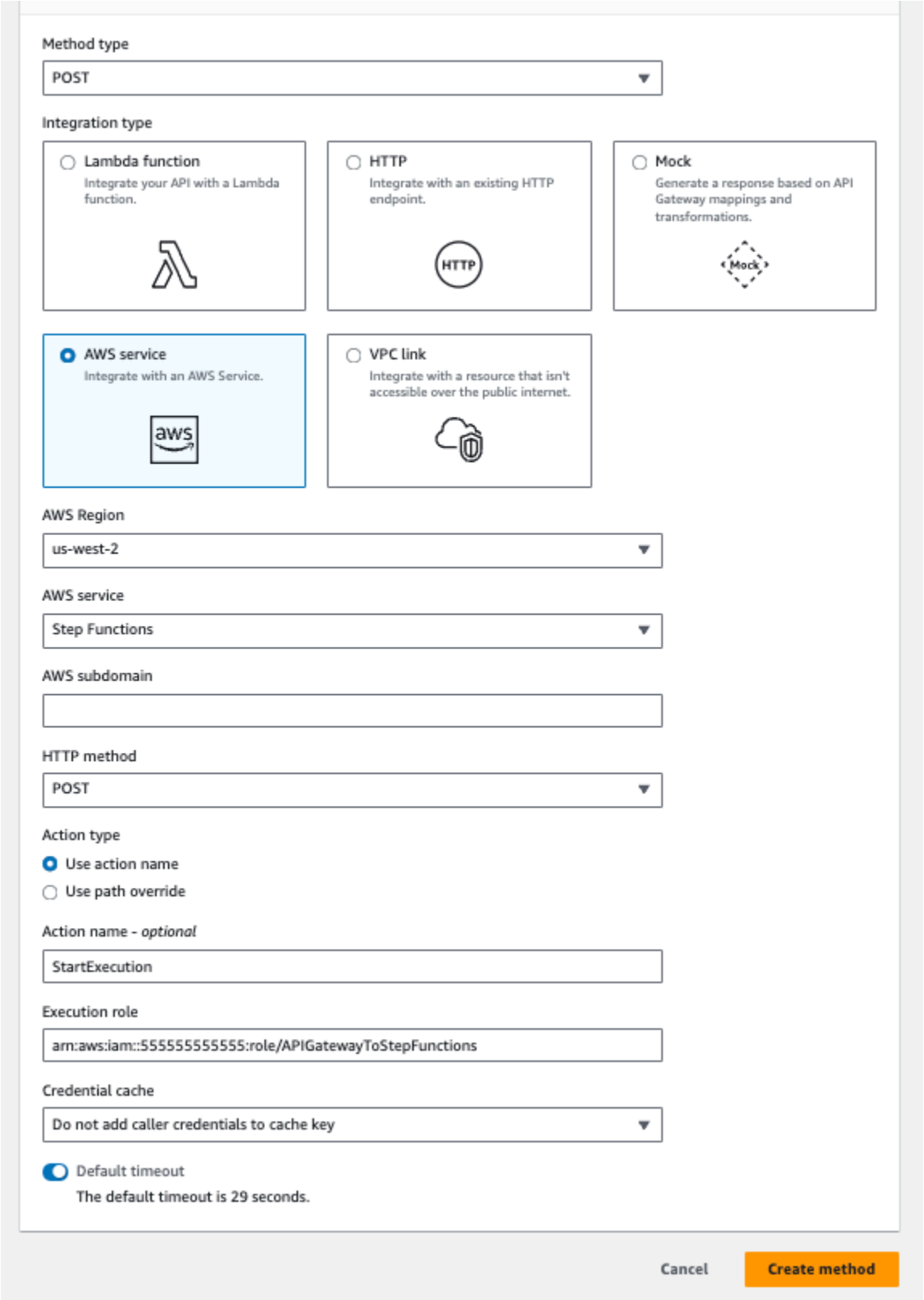
5. Em AWS service (Serviço da AWS), selecione Step Functions na lista.
6. Mantenha o subdomínio da AWS em branco.
7. Em Método HTTP, selecione POST na lista.

> **Note**
>
> Todas as ações da API do Step Functions usam o método HTTP `POST`.

8. Em Tipo de ação, selecione Usar nome da ação.

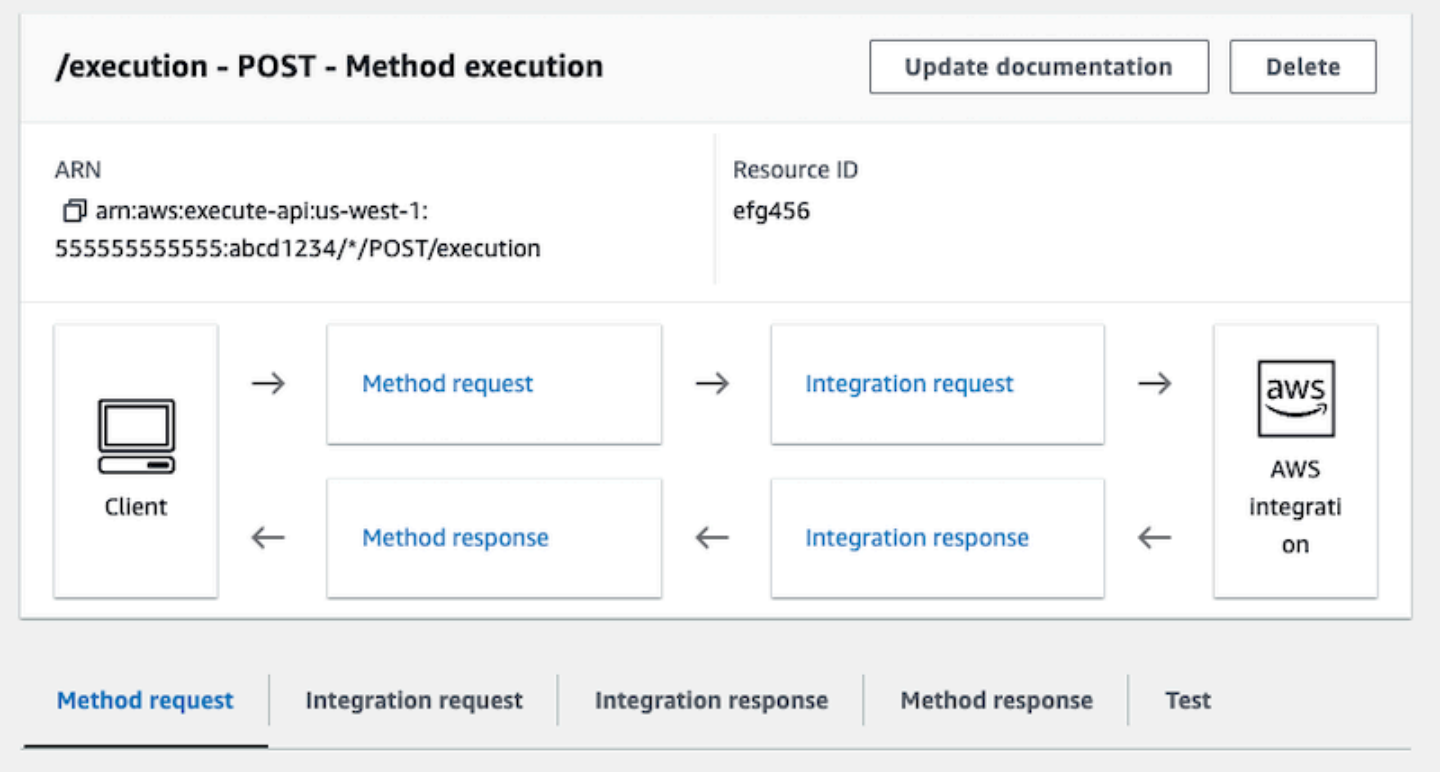
9. Em Nome da ação, insira **`StartExecution`**.
10. Em Função de execução, insira o ARN do perfil do IAM criado antes, conforme mostrado no exemplo a seguir.

```
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. Mantenha as opções padrão para Cache de credenciais e Tempo limite padrão e, depois, selecione Salvar.

O mapeamento visual entre o API Gateway e o Step Functions é exibido na página /execução – POST – Método de execução.



Etapa 3: Testar e implantar a API do API Gateway

Assim que você tiver criado a API, teste-a e implante-a.

Para testar a comunicação entre o API Gateway e o Step Functions

1. Na página /execution - POST - Execução do método, selecione Testar. Talvez seja necessário selecionar o botão de seta para a direita para mostrar a guia.
2. Na guia /execution – POST – Teste de método, copie os parâmetros da solicitação a seguir na seção Corpo da solicitação usando o ARN de uma máquina de estado existente (ou criar uma máquina de estado que use uma função do Lambda) e, depois, selecione Testar.

```
{
   "input": "{}",
   "name": "MyExecution",
```

```
   "stateMachineArn": "arn:aws:states:us-
east-1:123456789012:stateMachine:HelloWorld"
}
```

Para obter mais informações, consulte Request Syntax (Sintaxe da solicitação) `StartExecution` na Referência da API AWS Step Functions .

> **Note**
>
> Se não quiser incluir o ARN da máquina de estado no corpo da chamada do API Gateway, será possível configurar um modelo de mapeamento na guia Solicitação de integração, conforme mostrado no exemplo a seguir.
>
> ```
> {
>     "input": "$util.escapeJavaScript($input.json('$'))",
>     "stateMachineArn": "$util.escapeJavaScript($stageVariables.arn)"
> }
> ```
>
> Com essa abordagem, você pode especificar ARNs de diferentes máquinas de estado com base no estágio de desenvolvimento (por exemplo, `dev`, `test` e `prod`). Para obter mais informações sobre como especificar variáveis de estágio em um modelo de mapeamento, consulte `$stageVariables` no Guia do desenvolvedor do API Gateway.

3. A execução começa e o ARN da execução e a respectiva data de referência de época são exibidos em Corpo da resposta.

```
{
   "executionArn": "arn:aws:states:us-
east-1:123456789012:execution:HelloWorld:MyExecution",
   "startDate": 1486768956.878
}
```

> **Note**
>
> Você pode visualizar a execução escolhendo sua máquina de estado no console do AWS Step Functions.

Para implantar sua API

1. Na página Recursos da ***StartExecutionAPI***, escolha Implantar API.
2. Em Estágio, selecione Novo estágio.
3. Em Stage name (Nome do estágio), insira **alpha**.
4. (Opcional) Em Description (Descrição), insira uma descrição.
5. Escolha Implantar.

Para testar sua implantação

1. Na página Estágios da ***StartExecutionAPI***, expanda alpha,/, /execution, POST e escolha o método POST.
2. Em Substituições de método, selecione o ícone de cópia para copiar o URL de invocação da API. O URL completo deve ser semelhante ao exemplo a seguir.

```
https://a1b2c3d4e5.execute-api.us-east-1.amazonaws.com/alpha/execution
```

3. Na linha de comando, execute o comando `curl` usando o ARN de sua máquina de estado e chame o URL de sua implantação, conforme mostrado no exemplo a seguir.

```
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
```

O ARN de execução e sua data de epoch são retornados, conforme mostrado no exemplo a seguir.

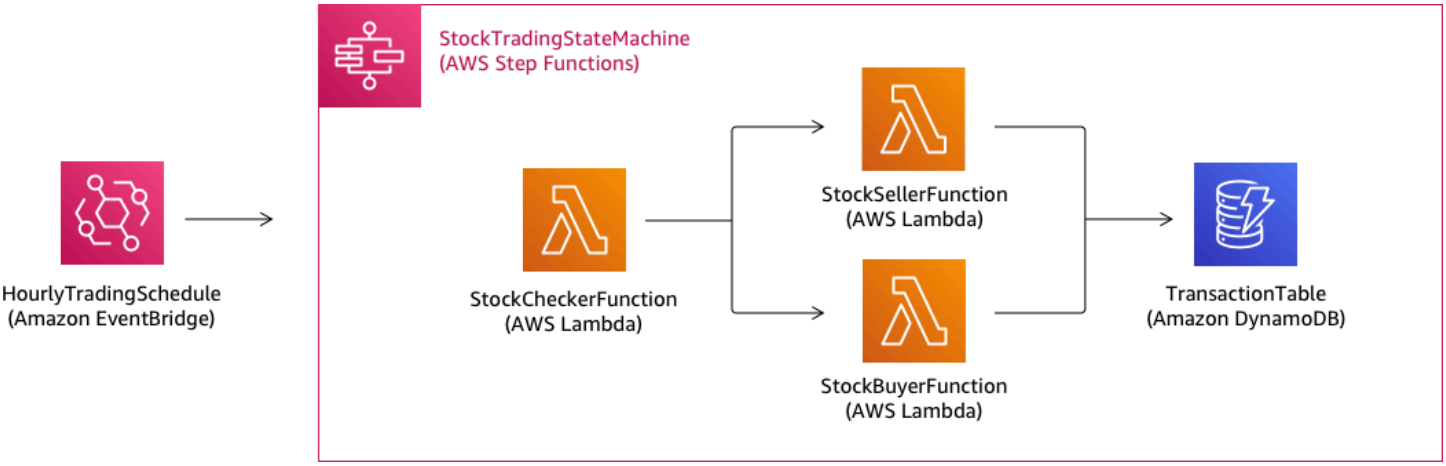
```
{"executionArn":"arn:aws:states:us-
east-1:123456789012:execution:HelloWorld:MyExecution","startDate":1.486772644911E9}
```

> **Note**
>
> Se você receber um erro “Token de autenticação ausente”, verifique se o URL de invocação termina com /execution.

Criar uma máquina de estado do Step Functions usando AWS SAM

Neste guia, você faz download, compila e implanta um aplicativo de exemplo do AWS SAM que contém uma máquina de estado do AWS Step Functions. Este aplicativo cria um fluxo de trabalho de negociação de ações simulado que é executado em uma programação predefinida (observe que a programação está desabilitada por padrão para evitar cobranças).

O diagrama a seguir mostra os componentes deste aplicativo:



Veja a seguir uma visualização dos comandos executados para criar o aplicativo de exemplo. Para obter mais detalhes sobre cada um desses comandos, consulte as seções posteriores nesta 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
```

Pré-requisitos

Neste guia, é pressuposto que você concluiu as etapas da seção Instalar a CLI do AWS SAM no sistema operacional. Pressupõem-se que você fez o seguinte:

1. Crie uma conta da AWS.
2. Permissões do IAM configuradas.
3. Instalou o Homebrew. Observação: o Homebrew é apenas um pré-requisito para o Linux e o macOS.
4. Instalou a CLI do AWS SAM. Observação: verifique se você tem a versão 0.52.0 ou posterior. É possível verificar a versão executando o comando `sam --version`.

Etapa 1: Fazer download de um aplicativo de exemplo do AWS SAM

Comando a ser executado:

```
sam init
```

Siga as instruções na tela para selecionar o seguinte:

1. Modelo: modelos de Início rápido do AWS
2. Linguagem: Python, Ruby, NodeJS, Go, Java ou .NET
3. Nome do projeto: (nome de sua escolha, o padrão é `sam-app`)
4. Aplicação de início rápido: fluxo de trabalho em várias etapas

O que AWS SAM está fazendo:

Este comando cria um diretório com o nome fornecido para o prompt "Nome do projeto" (o padrão é `sam-app`). O conteúdo específico do diretório dependerá da linguagem que você escolher.

Veja a seguir o conteúdo do diretório ao escolher um dos tempos de execução do 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
```

Existem dois arquivos especialmente interessantes que você pode conferir:

- `template.yaml`: contém o modelo do AWS SAM que define os recursos da AWS do seu aplicativo.
- `statemachine/stockTrader.asl.json`: contém a definição de máquina de estado do aplicativo, escrita em Amazon States Language.

É possível ver a seguinte entrada no arquivo `template.yaml`, que aponta para o arquivo de definição da máquina de estado:

```
    Properties:
      DefinitionUri: statemachine/stock_trader.asl.json
```

Pode ser útil manter a definição da máquina de estado como um arquivo separado, em vez de incorporá-la ao modelo AWS SAM. Por exemplo, rastrear alterações na definição da máquina de estado será mais fácil se você não incluir a definição no modelo. Você pode usar o Workflow Studio para criar e manter a definição da máquina de estado e exportar a definição do console diretamente para o arquivo de especificação da Amazon States Language sem mesclá-la ao modelo.

Para obter mais informações sobre o aplicativo de exemplo, consulte o arquivo `README.md` no diretório do projeto.

Etapa 2: Criar o aplicativo

Comando a ser executado:

Primeiro, mude para o diretório do projeto (ou seja, o diretório onde o arquivo `template.yaml` do aplicativo de exemplo está localizado; por padrão é `sam-app`) e execute este comando:

```
sam build
```

Exemplos de resultado:

```
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
```

O que AWS SAM está fazendo:

A CLI do AWS SAM vem com abstrações para vários runtimes do Lambda para criar as dependências e copia todos os artefatos de compilação em pastas de preparação para que tudo esteja pronto para ser empacotado e implantado. O comando `sam build` compila quaisquer dependências que o aplicativo tem e copia os artefatos de compilação para pastas em `.aws-sam/build`.

Etapa 3: Implantar o aplicativo na Nuvem AWS

Comando a ser executado:

```
sam deploy --guided
```

Siga as instruções na tela. Só é possível responder com `Enter` para aceitar as opções padrão fornecidas na experiência interativa.

O que AWS SAM está fazendo:

Este comando implanta o aplicativo na Nuvem AWS. Ele usa os artefatos de implantação criados com o comando `sam build`, empacota-os e faz upload deles para um bucket do Amazon S3

criado pela CLI do AWS SAM e implanta o aplicativo usando o AWS CloudFormation. Na saída do comando de implantação, é possível ver as alterações que estão sendo feitas na pilha do AWS CloudFormation.

É possível verificar se a máquina de estado de exemplo do Step Functions foi implantada com êxito seguindo estas etapas:

1. Entre no AWS Management Console e abra o console do Step Functions em https://console.aws.amazon.com/states/.
2. Na navegação à esquerda, escolha Máquinas de estado.
3. Encontre e escolha a nova máquina de estado na lista. Ela será StockTradingStateMachine-*<unique-hash>*.
4. Escolha a guia Definição.

Agora é necessário ver uma representação visual da máquina de estado. É possível verificar se a representação visual corresponde à definição da máquina de estado encontrada no arquivo `statemachine/stockTrader.asl.json` do diretório do projeto.

Solução de problemas

Erro da CLI do SAM: "no such option: --guided" (não existe essa opção: --guided)

Ao executar `sam deploy`, você verá o seguinte erro:

```
Error: no such option: --guided
```

Isso significa que você está usando uma versão mais antiga da CLI do AWS SAM que não é compatível com o parâmetro `--guided`. Para corrigir isso, é possível atualizar a versão da CLI do AWS SAM para 0.33.0 ou posterior ou omitir o parâmetro `--guided` do comando `sam deploy`.

Erro da CLI do SAM: "Failed to create managed resources: Unable to locate credentials" (Falha ao criar recursos gerenciados: não é possível localizar credenciais)

Ao executar `sam deploy`, você verá o seguinte erro:

```
Error: Failed to create managed resources: Unable to locate credentials
```

Isso significa que você não configurou credenciais da AWS para permitir que a CLI do AWS SAM faça chamadas de serviço da AWS. Para corrigir isso, é necessário configurar credenciais da AWS. Para obter mais informações, consulte Configurar credenciais do AWS no Guia do desenvolvedor do AWS Serverless Application Model.

Limpar

Se você não precisar mais dos recursos da AWS criados durante a execução deste tutorial, poderá removê-los excluindo a pilha do AWS CloudFormation que você implantou.

Para excluir a pilha do AWS CloudFormation criada com este tutorial usando o AWS Management Console, siga estas etapas:

1. Faça login no AWS Management Console e abra o console AWS CloudFormation em https://console.aws.amazon.com/cloudformation.
2. No painel de navegação à esquerda, selecioneStacks (Pilhas).
3. Na lista de pilhas, escolha sam-app (ou o nome da pilha criada).
4. Clique em Excluir.

Quando concluído, o status da pilha será alterado para DELETE_COMPLETE.

Como alternativa, é possível excluir a pilha do AWS CloudFormation executando o seguinte comando da AWS CLI:

```
aws cloudformation delete-stack --stack-name sam-app --region region
```

Verificar pilha excluída

Para os dois métodos de exclusão da pilha do AWS CloudFormation, é possível verificar se ela foi excluída acessando https://console.aws.amazon.com/cloudformation, escolhendo Pilhas no painel de navegação esquerdo e escolhendo Excluído no menu suspenso à direita da caixa de texto de pesquisa. Você verá o nome da pilha sam-app (ou o nome da pilha criada) na lista de pilhas excluídas.

Como criar uma máquina de estado de Atividade usando o Step Functions

Este tutorial mostra como criar uma máquina de estado baseada em atividade que usa Java e o AWS Step Functions. As atividades permitem que você controle o código de operador que é executado em outro lugar em sua máquina de estado. Para obter uma visão geral, consulte Atividades em Como funciona o Step Functions.

Para concluir este tutorial, você precisará do seguinte:

- O SDK para Java. O exemplo de atividade neste tutorial é um aplicativo Java que usa o AWS SDK for Java para se comunicar com AWS.
- AWS credenciais no ambiente ou no arquivo de AWS configuração padrão. Para obter mais informações, consulte Configurar suas AWS credenciais no Guia do AWS SDK for Java desenvolvedor.

Tópicos

Etapa 1: Criar uma atividade

Você deve informar o Step Functions sobre a atividade cujo operador (um programa) que você deseja criar. O Step Functions responde com um nome do recurso da Amazon (ARN) que estabelece uma identidade para a atividade. Use essa identidade para coordenar as informações passadas entre sua máquina de estado e o operador.

> **Important**
>
> Certifique-se de que sua tarefa de atividade esteja na mesma AWS conta da sua máquina de estado.

1. No console do Step Functions, no painel de navegação à esquerda, escolha Atividades.
2. Escolha Create activity (Criar atividade).
3. Insira um Nome para a atividade, como *get-greeting* por exemplo, e escolha Criar atividade.
4. Quando sua tarefa de atividade for criada, anote seu ARN, conforme mostrado no exemplo a seguir.

```
arn:aws:states:us-east-1:123456789012:activity:get-greeting
```

Etapa 2: Criar uma máquina de estado

Crie uma máquina de estado que determine quando sua atividade é invocada e quando o operador deve executar seu trabalho principal, coletar os respectivos resultados e retorná-los. Para criar a máquina de estado, você vai usar o Editor de código do Workflow Studio.

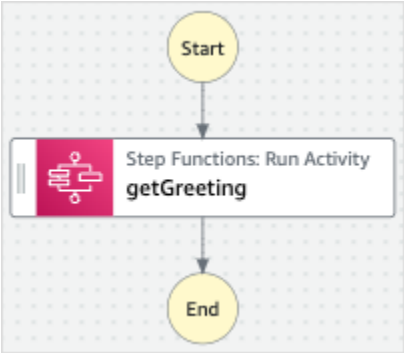
1. No console do Step Functions, no painel de navegação à esquerda, escolha Máquinas de estado.
2. Na página Máquinas de estado, selecione Criar máquina de estado.
3. Na caixa de diálogo Escolher um modelo, selecione Em branco.
4. Escolha Selecionar. Isso abre o Workflow Studio no Modo de design.
5. Neste tutorial, você vai escrever a definição da Amazon States Language (ASL) da máquina de estado no editor de código. Para isso, clique em Código.
6. Remova o código clichê existente e cole o código a seguir. Substitua o ARN de exemplo nesse código pelo ARN da tarefa de atividade que você criou anteriormente no campo de `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
    }
  }
```

```
}
```

Essa é a descrição de sua máquina de estado usando o Amazon States Language (ASL). Ela define um estado `Task` específico denominado `getGreeting`. Para obter mais informações, consulte Estrutura da máquina de estado.

7. No Painel de visualização gráfica, o gráfico do fluxo de trabalho para a definição de ASL que você adicionou deve ser semelhante ao gráfico a seguir.



8. Especifique um nome para a máquina de estado. Para fazer isso, escolha o ícone de edição ao lado do nome padrão da máquina de estado de MyStateMachine. Em seguida, em Configuração da máquina de estado, insira um nome na caixa Nome da máquina de estado.

   Para este tutorial, insira o nome **`ActivityStateMachine`**.

9. (Opcional) Em Configuração da máquina de estado, especifique outras configurações do fluxo de trabalho, como o tipo de máquina de estado e a função de execução.

   Para este tutorial, mantenha todas as seleções padrão nas Configurações da máquina de estado.

   Se você já criou um perfil do IAM com as permissões corretas para a máquina de estado e deseja usá-lo, em Permissões, clique em Escolher um perfil existente e selecione uma função na lista. Ou selecione Inserir um ARN de função e forneça o ARN para esse perfil do IAM.

10. Na caixa de diálogo Confirmar criação do perfil, selecione Confirmar para continuar.

    Você também pode escolher Exibir configurações do perfil para voltar às Configurações da máquina de estado.

> **Note**
>
> Se você excluir o perfil do IAM criado pelo Step Functions, não será possível recriá-lo posteriormente. Da mesma forma, se você modificar a função (por exemplo, removendo

o Step Functions das entidades principais na política do IAM), o Step Functions não poderá restaurar as configurações originais dela posteriormente.

Etapa 3: Implementar um operador

Crie um operador. Um operador é um programa que é responsável por:

- Sonde o Step Functions para atividades usando a ação da API `GetActivityTask`.
- Executar o trabalho da atividade usando seu código (por exemplo, o método `getGreeting()` no código a seguir).
- Retornar os resultados usando as ações de API `SendTaskSuccess`, `SendTaskFailuree` `SendTaskHeartbeat`.

> **Note**
>
> Para um exemplo mais completo de um operador de atividade, consulte Exemplo de operador de atividade em Ruby. Esse exemplo fornece uma implementação baseada em melhores práticas, que você pode usar como referência para seu operador de atividade. O código implementa um padrão de produtor-consumidor com um número configurável de threads para agentes de sondagem e operadores de atividade.

Para implementar o operador

1. Crie um arquivo chamado `GreeterActivities.java`.
2. Adicione a ele o código a seguir.

```
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**
>
> Nesse exemplo, a classe `EnvironmentVariableCredentialsProvider` assume as variáveis de ambiente `AWS_ACCESS_KEY_ID` (ou `AWS_ACCESS_KEY`) e `AWS_SECRET_KEY` (ou `AWS_SECRET_ACCESS_KEY`). Para obter mais informações sobre como fornecer as credenciais necessárias à fábrica, consulte AWSCredentialsProvidera Referência da AWS SDK for Java API e Configurar AWS credenciais e região para desenvolvimento no Guia do AWS SDK for Java desenvolvedor.
> Por padrão, o AWS SDK aguardará até 50 segundos para receber dados do servidor para qualquer operação. A operação `GetActivityTask` é uma operação de sondagem longa que aguardará até 60 segundos para a próxima tarefa disponível. Para evitar o recebimento de um `SocketTimeoutException` erro, SocketTimeout defina para 70 segundos.

3. Na lista de parâmetros do construtor `GetActivityTaskRequest().withActivityArn()`, substitua o valor `ACTIVITY_ARN` pelo ARN da tarefa de atividade que você criou anteriormente.

Etapa 4: Executar a máquina de estado

Ao iniciar a execução da máquina de estado, o operador examina atividades no Step Functions, executa seu trabalho (usando a entrada que você fornece) e retorna os devidos resultados.

1. Na ***ActivityStateMachine***página, escolha Iniciar execução.

   A caixa de diálogo Iniciar execução é exibida.

2. Na caixa de diálogo Iniciar execução, faça o seguinte:

   a. (Opcional) Para identificar a execução, insira um nome para ela no campo Nome. Por padrão, o Step Functions cria automaticamente um nome de execução exclusivo.

> **Note**
>
> Step Functions permite criar nomes para máquinas de estado, execuções e atividades, além de rótulos que contenham caracteres não ASCII. Esses nomes não ASCII não funcionam com a Amazon. CloudWatch Para garantir que você possa acompanhar CloudWatch as métricas, escolha um nome que use somente caracteres ASCII.

b. Na caixa Entrada, insira a seguinte entrada JSON para executar seu fluxo de trabalho.

```
{
  "who": "AWS Step Functions"
}
```

c. Selecione Iniciar execução.

d. O console do Step Functions direciona você para uma página em que o título é o ID da execução. Essa página é conhecida como página de Detalhes da execução. Nesta página, você pode revisar os resultados da execução à medida que a execução avança ou após a conclusão.

   Para revisar os resultados da execução, escolha estados individuais na Exibição em gráfico e, em seguida, escolha as guias individuais no painel Detalhes da etapa para visualizar os detalhes de cada estado, incluindo entrada, saída e definição, respectivamente. Para obter detalhes sobre as informações de execução que você pode visualizar na página Detalhes da execução, consulte Página de Detalhes da execução — Visão geral da interface.

Etapa 5: Executar e interromper o operador

Para que o operador examine atividades em sua máquina de estado, você precisa executá-lo.

1. Na linha de comando, vá até o diretório em que você criou `GreeterActivities.java`.
2. Para usar o AWS SDK, adicione o caminho completo dos `third-party` diretórios `lib` e às dependências do seu arquivo de compilação e ao seu Java. `CLASSPATH` Para obter mais informações, consulte Como fazer download e extrair o SDK no Guia do desenvolvedor de AWS SDK for Java .
3. Compile o arquivo.

```
$ javac GreeterActivities.java
```

4. Execute o arquivo.

```
$ java GreeterActivities
```

5. No console do Step Functions, vá até a página Detalhes da execução.
6. Quando a execução for concluída, examine os resultados da execução.
7. Interrompa o operador.

Repita um loop com o Lambda

Neste tutorial, você irá implementar um padrão de design que usa uma máquina de estado e uma função de AWS Lambda para iterar um loop um número específico de vezes.

Use esse padrão de design sempre que precisar para controlar o número de loops em uma máquina de estado. Essa implementação pode ajudar você a dividir grandes tarefas ou execuções de longa execução em partes menores. Ela também será útil no encerramento de uma execução após um número específico de eventos. Você pode usar uma implementação semelhante para encerrar e reiniciar periodicamente uma execução de longa duração para evitar exceder as cotas de serviço para AWS Step Functions AWS Lambda, ou outros serviços. AWS

Antes de começar, leia o tutorial do Como criar uma máquina de estado Step Functions que usa Lambda para garantir que você esteja familiarizado com o uso do Lambda e do Step Functions juntos.

Etapa 1: Criar uma função do Lambda para iterar uma contagem

Ao usar uma função do Lambda, você pode controlar o número de iterações de um loop na sua máquina de estado. A função do Lambda a seguir recebe os valores de entrada para `count`, `index` e `step`. Ela retorna esses valores com um `index` atualizado e um valor booliano chamado `continue`. A função do Lambda definirá `continue` como `true` se `index` for menor que `count`.

A seguir, a máquina de estado implementa um estado de `Choice` que executa certa lógica de aplicação se `continue` for `true`, ou é encerrada se for `false`.

Para criar a função do Lambda

1. Faça login no console do Lambda e escolha Criar função.
2. Na página Create function, selecione Author from scratch.
3. Na seção Informações básicas, configure a função do Lambda da seguinte maneira:
   a. Em Function name (Nome da função), insira `Iterator`.
   b. Em Runtime (Tempo de execução), selecione Node.js.
   c. Em Alterar a função de execução padrão, escolha Create a new role with basic Lambda permissions (Criar uma função com permissões básicas do Lambda).
   d. Escolha a opção Criar função.
4. Copie o código a seguir para a função Lambda na fonte do código.

```
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
  })
}
```

   Esse código aceita os valores de entrada `count`, `index` e `step`. Ele incrementa o `index` pelo valor da `step` e retorna esses valores e o booliano `continue`. O valor de `continue` é `true` se `index` for inferior a `count`.
5. Escolha Implantar.

Etapa 2: Testar a função do Lambda

Execute a sua função do Lambda com valores numéricos para vê-la em operação. Você pode fornecer valores de entrada para sua função Lambda que imitam uma iteração.

Para testar a função do Lambda

1. Escolha Testar.
2. Na caixa de diálogo Configurar evento de teste, insira `TestIterator` na caixa Nome do evento.
3. Substitua os dados de exemplo pelo seguinte:

```
{
  "Comment": "Test my Iterator function",
  "iterator": {
    "count": 10,
    "index": 5,
    "step": 1
  }
}
```

   Esses valores imitam o que pode ser proveniente de sua máquina de estado durante uma iteração. A função Lambda incrementará o índice e retornará `true` `continue` quando o índice for menor que. `count` Para este teste, o índice já foi incrementado para `5`. O teste será `index` incrementado 6 e definido como`continue`. `true`
4. Escolha Criar.
5. Escolha Testar para testar sua função Lambda.

   Os resultados do teste são exibidos na guia Resultados da execução.
6. Escolha a guia Resultados da execução para ver a saída.

```
{
  "index": 6,
  "step": 1,
  "count": 10,
  "continue": true
}
```

> **Note**
>
> Se você configurar 9 e `index` testar novamente, os `index` incrementos para`10`, e `continue` serão`false`.

Etapa 3: Criar uma máquina de estado

> Antes de sair do console Lambda...
>
> Copie o ARN da função Lambda. Cole-o em uma nota. Você precisará dele na próxima etapa.

Em seguida, você criará uma máquina de estados com os seguintes estados:

- `ConfigureCount`— Define valores padrão para `countindex`, `step` e.
- `Iterator`— Refere-se à função Lambda que você criou anteriormente, passando os valores configurados em. `ConfigureCount`
- `IsCountReached`— Um estado de escolha que continua o loop ou continua até o `Done` estado, com base no valor retornado de sua `Iterator` função.
- `ExampleWork`— Um esboço para o trabalho que precisa ser feito. Neste exemplo, o fluxo de trabalho tem um `Pass` estado, mas em uma solução real, você provavelmente usaria um`Task`.
- `Done`— Estado final do seu fluxo de trabalho.

Para criar a máquina de estado no console:

1. Abra o console do Step Functions e escolha Create a state machine (Criar uma máquina de estado).

   > **Important**
   >
   > Sua máquina de estado deve estar na mesma AWS conta e região da sua função Lambda.

2. Selecione o modelo em branco.
3. No painel Código, cole o seguinte JSON que define a máquina de estado.

   Para obter mais informações sobre a Amazon States Language, consulte Estrutura da máquina 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. Substitua o `Iterator Resource` campo pelo ARN da função `Iterator` Lambda que você criou anteriormente.
5. Selecione Config e insira um nome para sua máquina de estado, como. *`IterateCount`*

> **Note**
>
> Os nomes das máquinas de estado, execuções e tarefas de atividade não devem exceder oitenta caracteres. Esses nomes devem ser exclusivos para sua conta e AWS região e não devem conter nenhum dos seguintes itens:
>
> - Espaço em branco
> - Caracteres curinga (`? *`)
> - Caracteres de colchete (`< > { } [ ]`)
> - Caracteres especiais (`" # % \ ^ | ~ ` $ & , ; : /`)
> - caracteres de controle (`\\u0000` - `\\u001f` ou `\\u007f` - `\\u009f`).
>
> Se sua máquina de estado for do tipo Express, você poderá fornecer o mesmo nome para várias execuções da máquina de estado. O Step Functions gera um ARN de execução exclusivo para cada execução de máquina de estado Express, mesmo que várias execuções tenham o mesmo nome.
> Step Functions permite criar nomes para máquinas de estado, execuções e atividades, além de rótulos que contenham caracteres não ASCII. Esses nomes não ASCII não funcionam com a Amazon. CloudWatch Para garantir que você possa acompanhar CloudWatch as métricas, escolha um nome que use somente caracteres ASCII.

6. Para Tipo, aceite o valor padrão de Padrão. Em Permissões, escolha Criar nova função.
7. Escolha Criar e, em seguida, Confirme as criações da função.

Etapa 4: Iniciar uma nova execução

Assim que criar sua máquina de estado, poderá iniciar uma execução.

1. Na IterateCountpágina, escolha Iniciar execução.

2. (Opcional) Para identificar a execução, insira um nome para ela no campo Nome. Por padrão, o Step Functions cria automaticamente um nome de execução exclusivo.

   > **Note**
   >
   > Step Functions permite criar nomes para máquinas de estado, execuções e atividades, além de rótulos que contenham caracteres não ASCII. Esses nomes não ASCII não funcionam com a Amazon. CloudWatch Para garantir que você possa acompanhar CloudWatch as métricas, escolha um nome que use somente caracteres ASCII.

3. Escolha Start Execution.

   Uma nova execução de sua máquina de estado inicia-se mostrando sua execução em andamento.
   Visualização do gráfico da máquina de estado, mostrando o estado do Iterador em azul para indicar o status em andamento.

   A execução é incrementada em etapas, fazendo o acompanhamento da contagem com a função do Lambda. Em cada iteração, ela executa o exemplo mencionado no estado `ExampleWork` em sua máquina de estado.

   Quando a contagem atingir o número especificado no estado `ConfigureCount` em sua máquina de estado, a execução fechará a iteração e será encerrada.
   Visualização do gráfico da máquina de estado, mostrando o estado do Iterador e o estado Concluído em verde para indicar que ambos foram bem-sucedidos.

Execuções contínuas de fluxo de trabalho de longa duração como uma nova execução

O AWS Step Functions é projetado para executar fluxos de trabalho que têm uma duração finita e número de etapas. As execuções têm uma duração máxima de um ano e um máximo de 25.000 eventos (consulte Cotas).

Para execuções de longa duração, para evitar atingir a cota fixa de 25 mil entradas no histórico de eventos de execução, recomendamos que você inicie a execução de um novo fluxo de trabalho diretamente do estado `Task` de uma máquina de estado. Isso permite que você divida seus fluxos de trabalho em máquinas de estado menores e continue o trabalho em andamento em

uma nova execução. Para iniciar essas execuções de fluxo de trabalho, chame a ação da API `StartExecution` do seu estado `Task` e passe os parâmetros necessários.

Como alternativa, você também pode implementar um padrão que usa uma função do Lambda para iniciar uma nova execução da sua máquina de estado para dividir o trabalho contínuo em várias execuções de fluxo de trabalho.

Este tutorial mostra as duas abordagens para continuar as execuções do fluxo de trabalho sem exceder as service quotas.

Tópicos

Usar uma ação da API Step Functions para continuar uma nova execução (recomendado)

O Step Functions pode iniciar essas execuções de fluxo de trabalho chamando a própria API como um serviço integrado. Recomendamos que você adote essa abordagem para evitar exceder as service quotas para execuções de longa duração.

Etapa 1: Criar uma máquina de estado de execução longa

Crie uma máquina de estado de longa duração que você deseja iniciar do estado `Task` de uma máquina de estado diferente. Para este tutorial, use a máquina de estado que usa uma função do Lambda.

> **Note**
>
> Copie o nome e o Nome do recurso da Amazon dessa máquina de estado em um arquivo de texto para uso posterior.

Etapa 2: Criar uma máquina de estado para chamar a ação da API do Step Functions

Para iniciar execuções de fluxo de trabalho usando um estado **`Task`**

1. Abra o console do Step Functions e clique em Criar máquina de estado.

2. Na caixa de diálogo Escolher um modelo, selecione Em branco.
3. Escolha Selecionar. Isso abre o Workflow Studio no Modo de design.
4. Na guia Ações, arraste a ação da StartExecutionAPI e solte-a no estado vazio chamado Arraste o primeiro estado aqui.
5. Escolha o StartExecutionestado e faça o seguinte na guia Configuração emModo de design:

   a. Renomeie o estado para **`Start nested execution`**.
   b. Em Tipo de integração, escolha SDK AWS – novo na lista suspensa.
   c. Em Parâmetros da API, faça o seguinte:

      i. Para `StateMachineArn`, substitua o nome de recurso da Amazon de amostra pelo ARN da sua máquina de estado. Por exemplo, insira o ARN da máquina de estado que usa o Lambda.
      ii. Para o nó `Input`, substitua o texto do espaço reservado pelo seguinte valor:

```
"Comment": "Starting workflow execution using a Step Functions API action"
```

      iii. As entradas nos Parâmetros da API `devem ser semelhantes às seguintes:

```
{
  "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) Escolha Definição no painel Inspector para ver a definição do Amazon States Language (ASL) gerada automaticamente do seu fluxo de trabalho.

   > **Tip**
   >
   > Você também pode ver a definição de ASL no Editor de código do Workflow Studio. No editor de código, você também pode editar a definição de ASL do fluxo de trabalho.

7. Especifique um nome para a máquina de estado. Para fazer isso, escolha o ícone de edição ao lado do nome padrão da máquina de estado de MyStateMachine. Em seguida, em Configuração da máquina de estado, insira um nome na caixa Nome da máquina de estado.

Para este tutorial, insira o nome **`ParentStateMachine`**.

8. (Opcional) Em Configuração da máquina de estado, especifique outras configurações do fluxo de trabalho, como o tipo de máquina de estado e a função de execução.

   Para este tutorial, mantenha todas as seleções padrão nas Configurações da máquina de estado.

   Se você já criou um perfil do IAM com as permissões corretas para a máquina de estado e deseja usá-lo, em Permissões, clique em Escolher um perfil existente e selecione uma função na lista. Ou selecione Inserir um ARN de função e forneça o ARN para esse perfil do IAM.

9. Na caixa de diálogo Confirmar criação do perfil, selecione Confirmar para continuar.

   Você também pode escolher Exibir configurações do perfil para voltar às Configurações da máquina de estado.

   > **Note**
   >
   > Se você excluir o perfil do IAM criado pelo Step Functions, não será possível recriá-lo posteriormente. Da mesma forma, se você modificar a função (por exemplo, removendo o Step Functions das entidades principais na política do IAM), o Step Functions não poderá restaurar as configurações originais dela posteriormente.

Etapa 3: Atualizar a política do IAM;

Para garantir que sua máquina de estado tenha permissões para iniciar a execução da máquina de estado que usa uma função do Lambda, você precisa anexar uma política em linha ao perfil do IAM da sua máquina de estado. Para mais informações, consulte Incorporar políticas em linha no Guia do usuário do IAM.

1. Na ParentStateMachinepágina, escolha o ARN da função do IAM para navegar até a página Funções do IAM da sua máquina de estado.
2. Atribua uma permissão apropriada à função IAM do ParentStateMachinepara que ela possa iniciar a execução de outra máquina de estado. Para atribuir a permissão, faça o seguinte:

   a. Na página Perfis do IAM, escolha Adicionar permissões e depois Criar política em linha.

   b. Na página Criar política, escolha a guia JSON.

c. Substitua o texto pela política a seguir.

```
{
  "Version": "2012-10-17",
  "Statement": [
      {
          "Effect": "Allow",
          "Action": [
              "states:StartExecution"
          ],
          "Resource": [
              "arn:aws:states:us-
east-2:123456789012:stateMachine:LambdaStateMachine"
          ]
      }
  ]
}
```

d. Escolha Revisar política.

e. Especifique um nome para a política e escolha Criar política.

Etapa 4: Executar a máquina de estado

As execuções de máquinas de estado são instâncias em que o fluxo de trabalho é executado para a realização de tarefas.

1. Na ParentStateMachinepágina, escolha Iniciar execução.

   A caixa de diálogo Iniciar execução é exibida.

2. Na caixa de diálogo Iniciar execução, faça o seguinte:

   a. (Opcional) Para identificar a execução, insira um nome para ela no campo Nome. Por padrão, o Step Functions cria automaticamente um nome de execução exclusivo.

   > **Note**
   >
   > Step Functions permite criar nomes para máquinas de estado, execuções e atividades, além de rótulos que contenham caracteres não ASCII. Esses nomes não ASCII não funcionam com a Amazon. CloudWatch Para garantir que você

possa acompanhar CloudWatch as métricas, escolha um nome que use somente caracteres ASCII.

b. (Opcional) No campo Entrada, insira os valores de entrada no formato JSON para executar o fluxo de trabalho.

c. Selecione Iniciar execução.

d. O console do Step Functions direciona você para uma página em que o título é o ID da execução. Essa página é conhecida como página de Detalhes da execução. Nesta página, você pode revisar os resultados da execução à medida que a execução avança ou após a conclusão.

   Para revisar os resultados da execução, escolha estados individuais na Exibição em gráfico e, em seguida, escolha as guias individuais no painel Detalhes da etapa para visualizar os detalhes de cada estado, incluindo entrada, saída e definição, respectivamente. Para obter detalhes sobre as informações de execução que você pode visualizar na página Detalhes da execução, consulte Página de Detalhes da execução — Visão geral da interface.

3. Abra a LambdaStateMachinepágina e observe uma nova execução acionada pelo ParentStateMachine.

Usar uma função do Lambda para continuar uma nova execução

Você pode criar uma máquina de estado que usa uma função do Lambda para iniciar uma nova execução antes que a execução atual termine. Usar essa abordagem para continuar seu trabalho em andamento em uma nova execução permite que você tenha uma máquina de estado que pode separar trabalhos maiores em menores fluxos de trabalho, ou para ter uma máquina de estado que roda indefinidamente.

Este tutorial se baseia no conceito de usar uma função do Lambda externa para modificar seu fluxo de trabalho, que foi demonstrado no tutorial Repita um loop com o Lambda. Você usa a mesma função do Lambda (`Iterator`) para iterar um loop por um número de vezes específico. Além disso, você cria outra função do Lambda para iniciar uma nova execução do seu fluxo de trabalho e para diminuir uma contagem cada vez que iniciar uma nova execução. Ao definir o número de execuções na entrada, essa máquina de estado encerra e reinicia uma execução um número de vezes específico.

A máquina de estado que você criará implementa os seguintes estados.

State	Finalidade
`ConfigureCount`	Um estado `Pass` que configura os valores `count`, `index` e `step` que são usados pela função do Lambda do `Iterator` para percorrer iterações de trabalho.
`Iterator`	Um estado `Task` que faz referência à função do Lambda do `Iterator`.
`IsCountReached`	Um estado `Choice` que usa um valor booliano da função `Iterator` para decidir se a máquina de estado deve continuar o trabalho de exemplo, ou mover para o estado `ShouldRestart` .
`ExampleWork`	Um estado `Pass` que representa o estado `Task` que deve executar trabalhos em uma implementação real.
`ShouldRestart`	Um estado `Choice` que usa o valor `executionCount` para decidir se deve terminar uma execução e iniciar outra, ou simplesmente finaliza.
`Restart`	Um estado `Task` que usa uma função do Lambda para iniciar uma nova execução de sua máquina de estado. Como a função `Iterator`, essa função também diminui uma contagem. O estado `Restart` passa o valor diminuído da contagem para a entrada da nova execução.

Pré-requisitos

Antes de começar, leia o tutorial do Como criar uma máquina de estado Step Functions que usa Lambda para garantir que você esteja familiarizado com o uso do Lambda e do Step Functions juntos.

Tópicos

- Etapa 1: Criar uma função do Lambda para iterar uma contagem
- Etapa 2: Criar uma função do Lambda Reiniciar para iniciar uma nova execução do Step Functions
- Etapa 3: Criar uma máquina de estado
- Etapa 4: Atualizar a política do IAM

- Etapa 5: Executar a máquina de estado

Etapa 1: Criar uma função do Lambda para iterar uma contagem

> **Note**
>
> Se tiver concluído o tutorial do Repita um loop com o Lambda, você poderá ignorar esta etapa e usar essa função do Lambda.

Esta seção e o tutorial do Repita um loop com o Lambda mostra como usar uma função do Lambda para rastrear uma contagem, por exemplo, o número de iterações de um loop em sua máquina de estado.

A função do Lambda a seguir recebe os valores de entrada para `count`, `index` e `step`. Ela retorna esses valores com um `index` atualizado e um booliano chamado `continue`. A função do Lambda definirá `continue` como `true` se `index` for menor que `count`.

Sua máquina de estado implementa um estado de `Choice` que executa certa lógica de aplicação se `continue` for `true`, ou move-se para `ShouldRestart` se `continue` for `false`.

Criar a função do Lambda Iterar

1. Abra o console do Lambda e escolha Create function (Criar função).
2. Na página Create function, selecione Author from scratch.
3. Na seção Informações básicas, configure a função do Lambda da seguinte maneira:
   a. Em Function name (Nome da função), insira `Iterator`.
   b. Em Runtime, selecione Node.js 16.x.
   c. Mantenha todas as seleções padrão na página e escolha Criar função.

      Quando a função do Lambda for criada, anote o respectivo Nome do recurso da Amazon (ARN) que aparece no canto superior direito da página, por exemplo:

      ```
      arn:aws:lambda:us-east-1:123456789012:function:Iterator
      ```
4. Copie o código a seguir para a função do Lambda na seção Origem do código da página ***`Iterador`*** no console do 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
  })
}
```

Esse código aceita os valores de entrada `count`, `index` e `step`. Ele incrementa o `index` pelo valor de `step` e retorna esses valores e o booliano de `continue`. O valor de `continue` é `true` se `index` for inferior a `count`.

5. Escolha Implantar para implantar o código.

Testar a função do Lambda Iterar

Para visualizar a função `Iterate` funcionando, execute-a com valores numéricos. Você pode fornecer valores de entrada para a função do Lambda que imitam uma iteração para verificar o resultado obtido com valores de entrada específicos.

Para testar a função do Lambda

1. Na caixa de diálogo Configure test event, escolha Create new test event e digite `TestIterator` para Event name.
2. Substitua os dados de exemplo pelo seguinte:

```
{
  "Comment": "Test my Iterator function",
  "iterator": {
    "count": 10,
    "index": 5,
    "step": 1
  }
```

```
}
```

Esses valores imitam o que pode ser proveniente de sua máquina de estado durante uma iteração. A função do Lambda incrementa o índice e retorna `continue` como `true`. Quando o índice não é menor que `count`, ele retorna `continue` como `false`. Para este teste, o índice já foi incrementado para 5. Os resultados devem incrementar o `index` para 6 e definir `continue` como `true`.

3. Escolha Criar.
4. Na página ***`Iterator`*** em seu console Lambda, TestIteratorverifique se está listado e escolha Testar.

   Os resultados do teste são exibidos na parte superior da página. Escolha Details e verifique o resultado.

```
{
  "index": 6,
  "step": 1,
  "count": 10,
  "continue": true
}
```

> **Note**
>
> Se você definir `index` como 9 para este teste, o `index` será incrementado para `10` e `continue` será `false`.

Etapa 2: Criar uma função do Lambda Reiniciar para iniciar uma nova execução do Step Functions

1. Abra o console do Lambda e escolha Create function (Criar função).
2. Na página Create function, selecione Author from scratch.
3. Na seção Informações básicas, configure a função do Lambda da seguinte maneira:
   a. Em Function name (Nome da função), insira `Restart`.
   b. Em Runtime, selecione Node.js 16.x.
4. Mantenha todas as seleções padrão na página e escolha Criar função.

Quando a função do Lambda for criada, anote o respectivo Nome do recurso da Amazon (ARN) que aparece no canto superior direito da página, por exemplo:

```
arn:aws:lambda:us-east-1:123456789012:function:Iterator
```

5. Copie o código a seguir para a função do Lambda na seção Origem do código da página ***Reiniciar*** no console do Lambda.

   O código a seguir diminui uma contagem do número de execuções e inicia uma nova execução de sua máquina de estado, incluindo o valor diminuído.

```
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. Escolha Implantar para implantar o código.

Etapa 3: Criar uma máquina de estado

Agora que você criou suas duas funções do Lambda, crie uma máquina de estado. Nesta máquina de estado, os estados `ShouldRestart` e `Restart` são como você divide seu trabalho em várias execuções.

Example ShouldRestart Estado de escolha

O trecho a seguir mostra o estado `ShouldRestartChoice`. Esse estado determina se você deve ou não reiniciar a execução.

```
"ShouldRestart": {
"Type": "Choice",
"Choices": [
  {
    "Variable": "$.restart.executionCount",
    "NumericGreaterThan": 1,
    "Next": "Restart"
  }
],
```

O valor `$.restart.executionCount` é incluído na entrada da execução inicial. Ele é reduzido cada vez que a função `Restart` é chamada e colocado na entrada para cada execução subsequente.

Example Estado Task de Restart

O trecho a seguir mostra o estado `RestartTask`. Esse estado usa a função do Lambda que você criou anteriormente para reiniciar a execução e para diminuir a contagem para rastrear o número de execuções restantes para iniciar.

```
"Restart": {
  "Type": "Task",
  "Resource": "arn:aws:lambda:us-east-1:123456789012:function:Restart",
  "Next": "Done"
},
```

Para criar a máquina de estado

1. Abra o console do Step Functions e clique em Criar máquina de estado.

   > **Important**
   >
   > Certifique-se de que sua máquina de estado esteja na mesma AWS conta e região das funções Lambda que você criou anteriormente na Etapa 1 e na Etapa 2.

2. Na caixa de diálogo Escolher um modelo, selecione Em branco.

3. Escolha Selecionar. Isso abre o Workflow Studio no Modo de design.
4. Neste tutorial, você escreverá a definição da Amazon States Language (ASL) da máquina de estado no Editor de código. Para isso, clique em Código.
5. Remova o código clichê existente e cole o código a seguir. Lembre-se de substituir os ARNs nesse código pelos ARNs das funções do Lambda que você criou.

```
{
    "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 um nome para a máquina de estado. Para fazer isso, escolha o ícone de edição ao lado do nome padrão da máquina de estado de MyStateMachine. Em seguida, em Configuração da máquina de estado, insira um nome na caixa Nome da máquina de estado.

   Para este tutorial, insira o nome **`ContinueAsNew`**.

7. (Opcional) Em Configuração da máquina de estado, especifique outras configurações do fluxo de trabalho, como o tipo de máquina de estado e a função de execução.

   Para este tutorial, mantenha todas as seleções padrão nas Configurações da máquina de estado.

   Se você já criou um perfil do IAM com as permissões corretas para a máquina de estado e deseja usá-lo, em Permissões, clique em Escolher um perfil existente e selecione uma função na lista. Ou selecione Inserir um ARN de função e forneça o ARN para esse perfil do IAM.

8. Na caixa de diálogo Confirmar criação do perfil, selecione Confirmar para continuar.

Você também pode escolher Exibir configurações do perfil para voltar às Configurações da máquina de estado.

> **Note**
>
> Se você excluir o perfil do IAM criado pelo Step Functions, não será possível recriá-lo posteriormente. Da mesma forma, se você modificar a função (por exemplo, removendo o Step Functions das entidades principais na política do IAM), o Step Functions não poderá restaurar as configurações originais dela posteriormente.

9. Salve o Nome do recurso da Amazon (ARN) dessa máquina de estado em um arquivo de texto. Você precisará fornecer o ARN e, ao mesmo tempo, permissão à função do Lambda para iniciar uma nova execução do Step Functions.

Etapa 4: Atualizar a política do IAM

Para garantir que a função do Lambda tenha permissões para iniciar uma nova execução do Step Functions, anexe uma política em linha para o perfil do IAM que você usa para sua função do Lambda `Restart`. Para mais informações, consulte Incorporar políticas em linha no Guia do usuário do IAM.

> **Note**
>
> Você pode atualizar a linha `Resource` no exemplo anterior para referenciar o ARN de sua máquina de estado `ContinueAsNew`. Isso restringe a política para que ela só possa iniciar uma execução dessa máquina de estado específico.

```
{
 "Version": "2012-10-17",
 "Statement": [
     {
         "Sid": "VisualEditor0",
         "Effect": "Allow",
         "Action": [
             "states:StartExecution"
         ],
         "Resource": "arn:aws:states:us-east-2:123456789012stateMachine:ContinueAsNew"
```

```
    }
 ]
}
```

Etapa 5: Executar a máquina de estado

Para iniciar uma execução, forneça a mesma entrada que inclui o ARN da máquina de estado e um `executionCount` para quantas vezes ela deve iniciar uma nova execução.

1. Na ContinueAsNewpágina, escolha Iniciar execução.

   A caixa de diálogo Iniciar execução é exibida.

2. Na caixa de diálogo Iniciar execução, faça o seguinte:

   a. (Opcional) Para identificar a execução, insira um nome para ela no campo Nome. Por padrão, o Step Functions cria automaticamente um nome de execução exclusivo.

   > **Note**
   >
   > Step Functions permite criar nomes para máquinas de estado, execuções e atividades, além de rótulos que contenham caracteres não ASCII. Esses nomes não ASCII não funcionam com a Amazon. CloudWatch Para garantir que você possa acompanhar CloudWatch as métricas, escolha um nome que use somente caracteres ASCII.

   b. Na caixa Entrada, insira a seguinte entrada JSON para executar seu fluxo de trabalho.

   ```
   {
     "restart": {
       "StateMachineArn": "arn:aws:states:us-
   east-1:123456789012:stateMachine:ContinueAsNew",
       "executionCount": 4
     }
   }
   ```

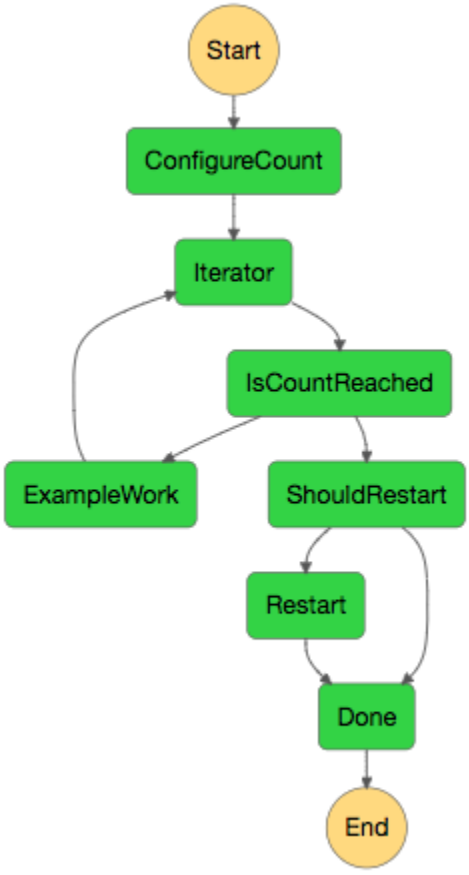
   c. Atualize o campo `StateMachineArn` com o ARN para sua máquina de estado `ContinueAsNew`.

   d. Selecione Iniciar execução.

e. O console do Step Functions direciona você para uma página em que o título é o ID da execução. Essa página é conhecida como página de Detalhes da execução. Nesta página, você pode revisar os resultados da execução à medida que a execução avança ou após a conclusão.

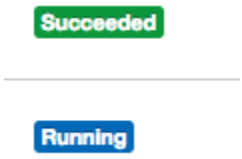
   Para revisar os resultados da execução, escolha estados individuais na Exibição em gráfico e, em seguida, escolha as guias individuais no painel Detalhes da etapa para visualizar os detalhes de cada estado, incluindo entrada, saída e definição, respectivamente. Para obter detalhes sobre as informações de execução que você pode visualizar na página Detalhes da execução, consulte Página de Detalhes da execução — Visão geral da interface.

   A Exibição em gráfico exibe a primeira das quatro execuções. Antes de concluir, ele passará pelo estado `Restart` e iniciará uma nova execução.



   Conforme essa execução é concluída, você poderá ver a próxima execução que estiver em andamento. Selecione o ContinueAsNewlink na parte superior para ver a lista de

execuções. Você deve ver a execução recém-fechada e uma execução contínua que a função do Lambda `Restart` iniciou.



Quando todas as execuções forem concluídas, você verá quatro execuções bem-sucedidas na lista. A primeira execução iniciada exibe o nome que você escolheu, e as execuções subsequentes têm um nome gerado.

Implantar um projeto de aprovação humana de exemplo

Este tutorial mostra como implantar um projeto de aprovação humana que permite que uma execução do AWS Step Functions pause durante uma tarefa e aguarde a resposta de um usuário a um e-mail. O fluxo de trabalho segue para o próximo estado depois que o usuário aprova a tarefa para prosseguir.

A implantação da AWS CloudFormation pilha incluída neste tutorial criará todos os recursos necessários, incluindo:

- Recursos do Amazon API Gateway
- E AWS Lambda funções
- Uma máquina de AWS Step Functions estado
- Um tópico do e-mail do Amazon Simple Notification Service
- AWS Identity and Access Management Funções e permissões relacionadas

> **Note**
>
> Você precisará fornecer um endereço de e-mail válido ao qual tenha acesso ao criar a AWS CloudFormation pilha.

Para obter mais informações, consulte Trabalhando com CloudFormation modelos e o `AWS::StepFunctions::StateMachine` recurso no Guia AWS CloudFormation do usuário.

Tópicos

Etapa 1: criar um AWS CloudFormation modelo

1. Copie o código de exemplo na seção AWS CloudFormation Código-fonte do modelo.

2. Cole a fonte do AWS CloudFormation modelo em um arquivo na sua máquina local.

   Neste exemplo, o arquivo é chamado de `human-approval.yaml`.

Etapa 2: Criar uma pilha

1. Faça login no console do AWS CloudFormation.
2. Selecione Criar pilha e Com novos recursos (padrão).
3. Na página Create a stack (Criar uma pilha), faça o seguinte:

a. Na seção Pré-requisito – Preparar modelo, verifique se a opção O modelo está pronto está selecionada.

b. Na seção Especificar modelo, escolha Fazer upload de um arquivo de modelo e escolha Escolher arquivo para carregar o `human-approval.yaml` arquivo que você criou anteriormente e que inclui o código-fonte do modelo.

4. Escolha Próximo.

5. Na página Specify stack details (Especificar detalhes da pilha), faça o seguinte:

a. Em Nome da pilha, informe um nome para sua pilha.

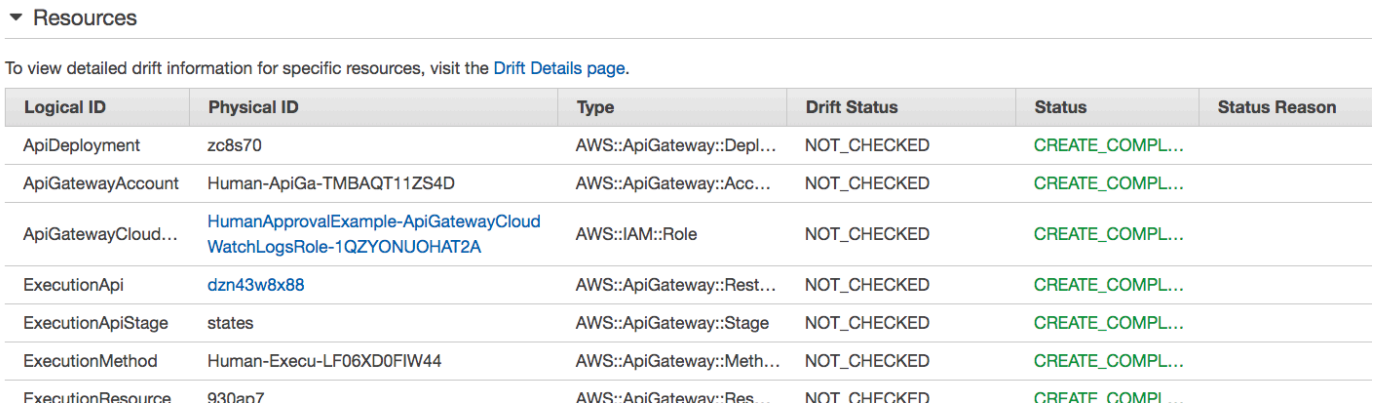
b. Em Parâmetros, insira um endereço de e-mail válido. Você vai usar esse endereço de e-mail para assinar o tópico do Amazon SNS.

6. Selecione Próximo e Próximo novamente.

7. Na página de revisão, escolha Eu reconheço que isso AWS CloudFormation pode criar recursos do IAM e, em seguida, escolha Criar.

   AWS CloudFormation começa a criar sua pilha e exibe o status CREATE_IN_PROGRESS. Quando o processo estiver concluído, AWS CloudFormation exibirá o status CREATE_COMPLETE.

8. (Opcional) Para exibir os recursos em sua pilha, selecione a pilha e escolha a guia Resources.



Etapa 3: Aprovar a assinatura do Amazon SNS

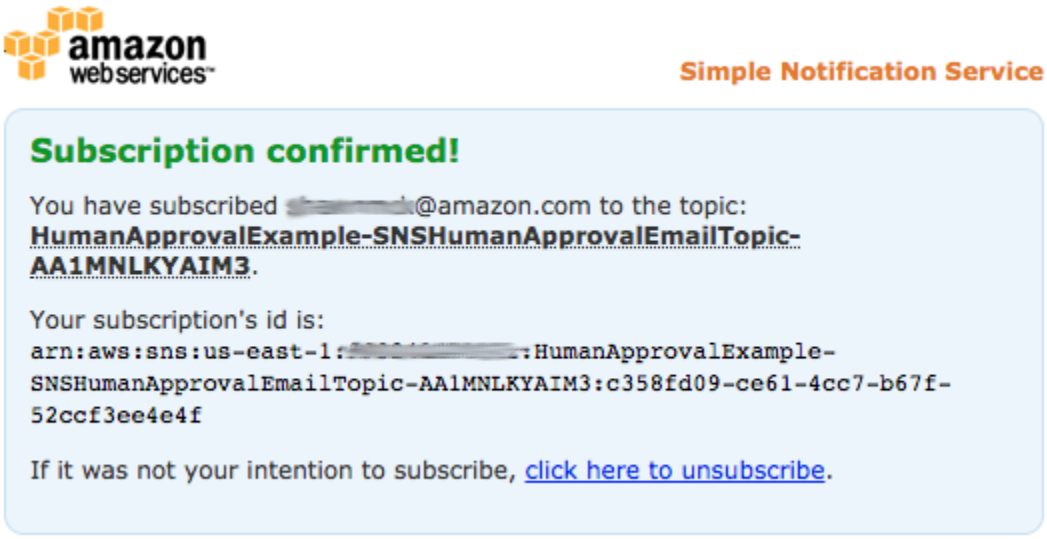
Quando o tópico do Amazon SNS for criado, você receberá um e-mail solicitando a confirmação da assinatura.

1. Abra a conta de e-mail que você forneceu ao criar a AWS CloudFormation pilha.

2. Abra a mensagem AWS Notification - Subscription Confirmation (Notificação da AWS – confirmação da assinatura) recebida de no-reply@sns.amazonaws.com

   O e-mail listará o Nome do recurso da Amazon para o tópico do Amazon SNS e um link de confirmação.

3. Clique no link confirm subscription (confirmar assinatura).



Etapa 4: Executar a máquina de estado

1. Na HumanApprovalLambdaStateMachinepágina, escolha Iniciar execução.

   A caixa de diálogo Iniciar execução é exibida.

2. Na caixa de diálogo Iniciar execução, faça o seguinte:

   a. (Opcional) Para identificar a execução, insira um nome para ela no campo Nome. Por padrão, o Step Functions cria automaticamente um nome de execução exclusivo.

   > **Note**
   >
   > Step Functions permite criar nomes para máquinas de estado, execuções e atividades, além de rótulos que contenham caracteres não ASCII. Esses nomes não ASCII não funcionam com a Amazon. CloudWatch Para garantir que você possa acompanhar CloudWatch as métricas, escolha um nome que use somente caracteres ASCII.

b. Na caixa Entrada, insira a seguinte entrada JSON para executar seu fluxo de trabalho.

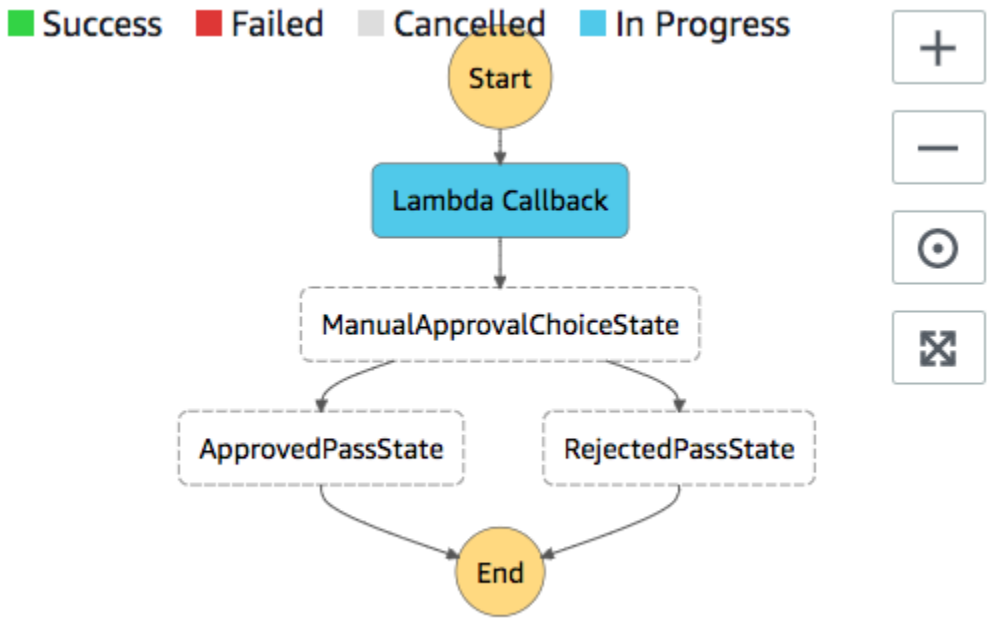
```
{
    "Comment": "Testing the human approval tutorial."
}
```

c. Selecione Iniciar execução.

A execução da máquina de ApprovalTestestado é iniciada e pausada na tarefa Lambda Callback.

d. O console do Step Functions direciona você para uma página em que o título é o ID da execução. Essa página é conhecida como página de Detalhes da execução. Nesta página, você pode revisar os resultados da execução à medida que a execução avança ou após a conclusão.

Para revisar os resultados da execução, escolha estados individuais na Exibição em gráfico e, em seguida, escolha as guias individuais no painel Detalhes da etapa para visualizar os detalhes de cada estado, incluindo entrada, saída e definição, respectivamente. Para obter detalhes sobre as informações de execução que você pode visualizar na página Detalhes da execução, consulte Página de Detalhes da execução — Visão geral da interface.

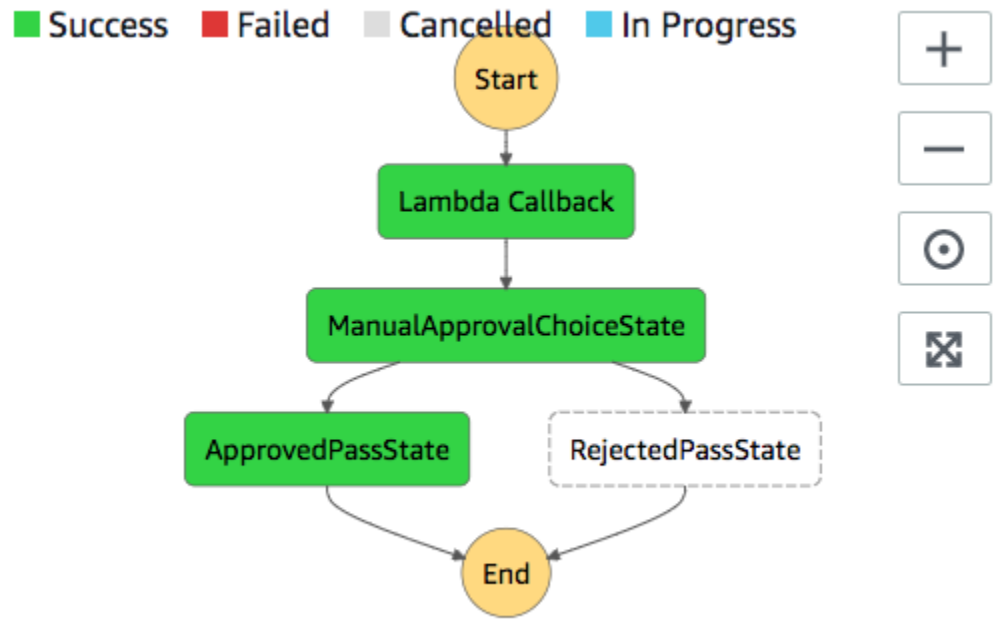


3. Na conta de e-mail que você usou para o tópico do Amazon SNS anteriormente, abra a mensagem com o assunto Aprovação obrigatória de. AWS Step Functions

A mensagem inclui URLs separados para Approve (Aprovar) e Reject (Rejeitar).

4. Selecione o URL para Approve (Aprovar).

   O fluxo de trabalho seguirá com base na sua escolha.



AWS CloudFormation Código-fonte do modelo

Use esse AWS CloudFormation modelo para implantar um exemplo de fluxo de trabalho do processo de aprovação humana.

```
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
```

Visualizar rastreamentos do X-Ray no Step Functions

Neste tutorial, você aprenderá a usar o X-Ray para rastrear erros que ocorrem ao executar uma máquina de estado. Você pode usar o AWS X-Ray para visualizar os componentes da máquina de estado, identificar gargalos de desempenho e solucionar problemas de solicitações que resultaram em erros. Neste tutorial, você criará várias funções do Lambda que geram erros aleatórios que podem ser rastreados e analisados usando o X-Ray.

O tutorial Como criar uma máquina de estado Step Functions que usa Lambda orienta você na criação de uma máquina de estado que chama uma função do Lambda. Se você concluiu esse tutorial, vá para a Etapa 2 e use o perfil do AWS Identity and Access Management (IAM) criado anteriormente.

Tópicos

Etapa 1: Criar um perfil do IAM para o Lambda

Ambos AWS Lambda AWS Step Functions podem executar código e acessar AWS recursos (por exemplo, dados armazenados em buckets do Amazon S3). Para manter a segurança, você deve permitir que o Lambda e o Step Functions acessem esses recursos.

O Lambda exige que você atribua uma função AWS Identity and Access Management (IAM) ao criar uma função Lambda, da mesma forma que o Step Functions exige que você atribua uma função do IAM ao criar uma máquina de estado.

Para criar uma função vinculada ao serviço, você usa o console do IAM.

Para criar uma função (console)

1. Faça login AWS Management Console e abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação do console do IAM, escolha Perfis. Então, escolha Criar perfil.
3. Escolha o tipo de perfil de serviço da AWS e depois selecione Lambda.
4. Escolha o caso de uso do Lambda. Casos de uso são definidos pelo serviço para incluir a política de confiança exigida pelo serviço. Então, escolha Próximo: Permissões.
5. Selecione uma ou mais políticas de permissões a serem anexadas à função (por exemplo, `AWSLambdaBasicExecutionRole`). Consulte Modelo de permissões do AWS Lambda.

   Selecione a caixa ao lado da política que atribui as permissões que você deseja que a função tenha e, em seguida, escolha Próximo: Revisar.
6. Insira um Role name.
7. (Opcional) Em Descrição da função, edite a descrição para a nova função vinculada ao serviço.
8. Reveja a função e escolha Create role (Criar função).

Etapa 2: Criar uma função do Lambda

A função do Lambda gerará erros aleatoriamente ou atingirá o tempo limite, produzindo dados de exemplo para visualização no X-Ray.

> **Important**
>
> Certifique-se de que sua função Lambda esteja na mesma AWS conta e AWS região da sua máquina estadual.

1. Abra o console do Lambda e clique em Criar função.
2. Na seção Criar função, selecione Criar do zero.
3. Na seção Informações básicas, configure a função do Lambda:
   a. Em Function name (Nome da função), insira `TestFunction1`.
   b. Em Runtime, selecione Node.js 18.x.
   c. Em Role (Função), selecione Choose an existing role (Escolher uma função existente).
   d. Em Função existente, selecione a função do Lambda que você criou anteriormente.

      > **Note**
      >
      > Se o perfil do IAM que você criou não aparecer na lista, será necessário esperar alguns minutos até que ele seja exibido no Lambda.

   e. Escolha a opção Criar função.

      Ao criar a função do Lambda, anote o nome do recurso da Amazon (ARN) exibido no canto superior direito da página. Por exemplo: .

      ```
      arn:aws:lambda:us-east-1:123456789012:function:TestFunction1
      ```

4. Copie o código a seguir para a função Lambda na seção Código da função da 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 }
};
```

Esse código cria falhas programadas aleatoriamente que serão usadas para gerar exemplos de erros na máquina de estado que podem ser visualizados e analisados usando rastreamentos do X-Ray.

5. Selecione Salvar.

Etapa 3: Criar mais duas funções do Lambda

Crie mais duas funções do Lambda.

1. Repita a Etapa 2 para criar mais duas funções do Lambda. Em Nome da função da primeira função, insira `TestFunction2`. Em Nome da função da segunda função, insira `TestFunction3`.
2. No console do Lambda, verifique se agora são exibidas três funções do Lambda, `TestFunction1`, `TestFunction2` e `TestFunction3`.

Etapa 4: Criar uma máquina de estado

Nesta etapa, você usará o console do Step Functions para criar uma máquina de estado com três estados `Task`. Cada estado `Task` fará referência a uma das três funções do Lambda.

1. Abra o console do Step Functions e clique em Criar máquina de estado.

> **Important**
>
> Certifique-se de que sua máquina de estado esteja na mesma AWS conta e região das funções Lambda que você criou anteriormente nas etapas 2 e 3.

2. Na caixa de diálogo Escolher um modelo, selecione Em branco.
3. Escolha Selecionar. Isso abre o Workflow Studio no Modo de design.
4. Neste tutorial, você escreverá a definição da Amazon States Language (ASL) da máquina de estado no Editor de código. Para isso, clique em Código.

5. Remova o código clichê existente e cole o código a seguir. Na definição do estado Tarefa, lembre-se de substituir os ARNs de exemplo pelos ARNs das funções do Lambda que você criou.

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

Essa é uma descrição da máquina de estado usando a Amazon States Language. Ela define três estados `Task` chamados `CallTestFunction1`, `CallTestFunction2` e `CallTestFunction3`. Cada uma chama uma das três funções do Lambda. Para obter mais informações, consulte Estrutura da máquina de estado.

6. Especifique um nome para a máquina de estado. Para fazer isso, escolha o ícone de edição ao lado do nome padrão da máquina de estado de MyStateMachine. Em seguida, em Configuração da máquina de estado, insira um nome na caixa Nome da máquina de estado.

   Para este tutorial, insira o nome **`TraceFunctions`**.

7. (Opcional) Em Configuração da máquina de estado, especifique outras configurações do fluxo de trabalho, como o tipo de máquina de estado e a função de execução.

   Para este tutorial, em Configurações adicionais, selecione Ativar rastreamento do X-Ray. Mantenha todas as outras seleções padrão nas configurações da máquina de estado.

   Se você já criou um perfil do IAM com as permissões corretas para a máquina de estado e deseja usá-lo, em Permissões, clique em Escolher um perfil existente e selecione uma função na lista. Ou selecione Inserir um ARN de função e forneça o ARN para esse perfil do IAM.

8. Na caixa de diálogo Confirmar criação do perfil, selecione Confirmar para continuar.

   Você também pode escolher Exibir configurações do perfil para voltar às Configurações da máquina de estado.

> **Note**
>
> Se você excluir o perfil do IAM criado pelo Step Functions, não será possível recriá-lo posteriormente. Da mesma forma, se você modificar a função (por exemplo, removendo o Step Functions das entidades principais na política do IAM), o Step Functions não poderá restaurar as configurações originais dela posteriormente.

Etapa 5: Executar a máquina de estado

As execuções de máquinas de estado são instâncias em que o fluxo de trabalho é executado para a realização de tarefas.

1. Na ***TraceFunctions***página, escolha Iniciar execução.

   A página New execution é exibida.

2. Na caixa de diálogo Iniciar execução, faça o seguinte:

   a. (Opcional) Para identificar a execução, insira um nome para ela no campo Nome. Por padrão, o Step Functions cria automaticamente um nome de execução exclusivo.

> **Note**
>
> Step Functions permite criar nomes para máquinas de estado, execuções e atividades, além de rótulos que contenham caracteres não ASCII. Esses nomes não ASCII não funcionam com a Amazon. CloudWatch Para garantir que você possa acompanhar CloudWatch as métricas, escolha um nome que use somente caracteres ASCII.

   b. Selecione Iniciar execução.

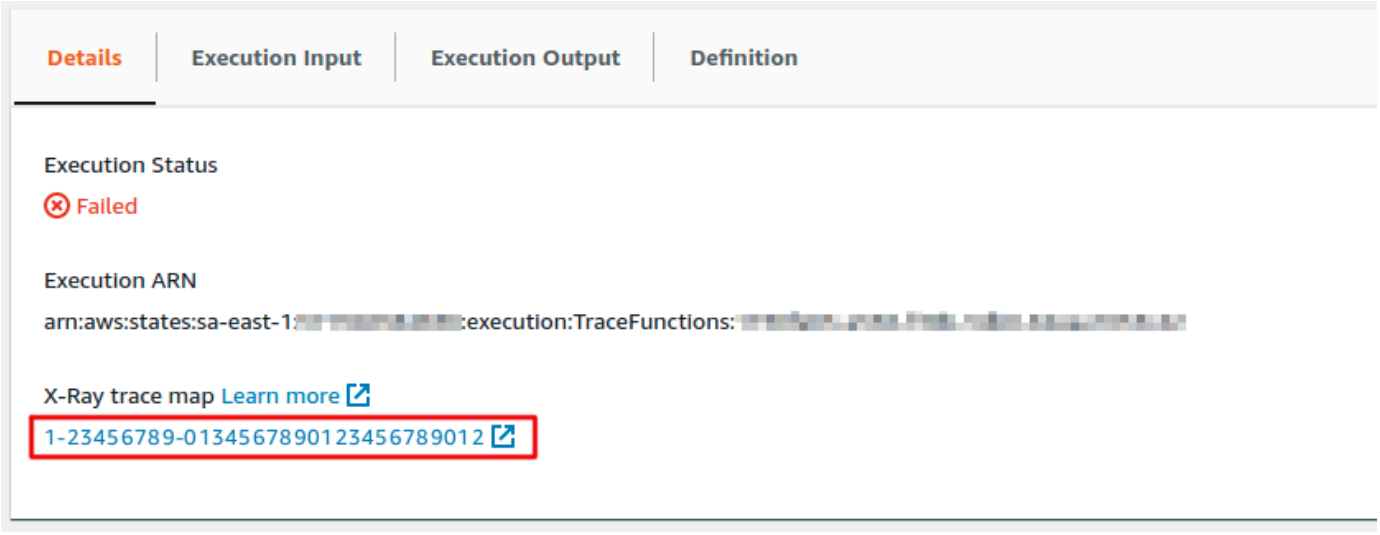
   c. O console do Step Functions direciona você para uma página em que o título é o ID da execução. Essa página é conhecida como página de Detalhes da execução. Nesta página,

você pode revisar os resultados da execução à medida que a execução avança ou após a conclusão.

Para revisar os resultados da execução, escolha estados individuais na Exibição em gráfico e, em seguida, escolha as guias individuais no painel Detalhes da etapa para visualizar os detalhes de cada estado, incluindo entrada, saída e definição, respectivamente. Para obter detalhes sobre as informações de execução que você pode visualizar na página Detalhes da execução, consulte Página de Detalhes da execução — Visão geral da interface.

Execute várias execuções (pelo menos três).

3. Depois que as execuções terminarem, acesse o link do mapa de rastreamento do X-Ray. Você pode visualizar o rastreamento enquanto a execução ainda está em andamento, porém, é interessante ver primeiro os resultados da execução antes de visualizar o mapa de rastreamento do X-Ray.



4. Visualize o mapa de serviço para identificar onde estão ocorrendo erros, conexões com alta latência ou rastreamentos de solicitações com erros. Neste exemplo, você pode ver quanto tráfego cada função está recebendo. A `TestFunction2` foi chamada com mais frequência do que a `TestFunction3` e a `TestFunction1` foi chamada duas vezes mais do que `TestFunction2`.

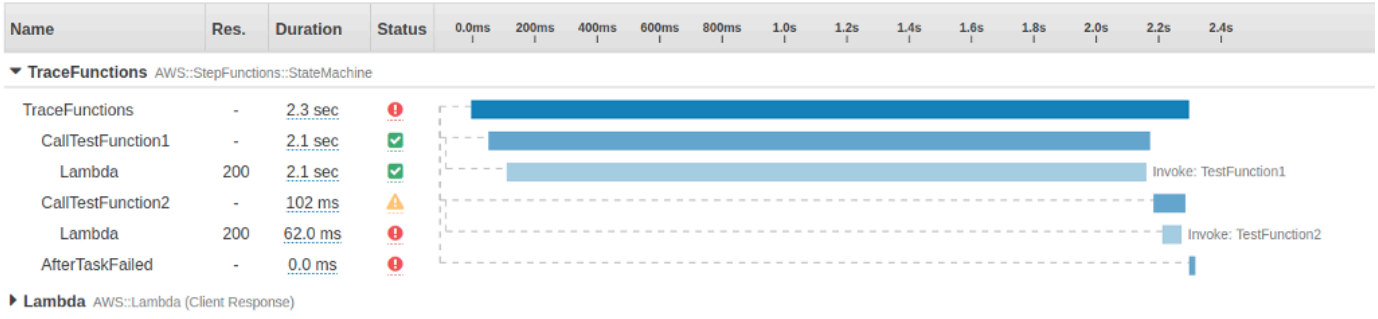
   O mapa de serviço indica a integridade de cada nó atribuindo cores a ele com base no índice de chamadas bem-sucedidas em relação a erros e falhas:

   - Verde para chamadas bem-sucedidas
   - Vermelho para falhas do servidor (erros da série 500)

- Amarelo para erros de clientes (erros da série 400)
- Roxo para erros de controle de utilização (429, muitas solicitações)

Você também pode escolher um nó de serviço para visualizar as solicitações desse nó ou uma borda entre dois nós para visualizar as solicitações que percorreram essa conexão.

5. Veja o mapa de rastreamento do X-Ray para ver o que aconteceu em cada execução. A visualização da linha do tempo exibe uma hierarquia de segmentos e subsegmentos. A primeira entrada na lista é o segmento, que representa todos os dados registrados pelo serviço para uma única solicitação. Os subsegmentos estão abaixo do segmento. Este exemplo mostra os subsegmentos registrados pelas funções do Lambda.



Para obter mais informações sobre como entender os rastreamentos do X-Ray e usar o X-Ray com o Step Functions, consulte o AWS X-Ray e Step Functions

Reúna informações do bucket do Amazon S3 usando integrações de serviços AWS SDK

Este tutorial mostra como realizar uma integração do AWS SDK com o Amazon Simple Storage Service. A máquina de estado que você cria neste tutorial reúne informações sobre seus buckets do Amazon S3 e lista seus buckets junto com as informações de versão de cada bucket na região atual.

Tópicos

Etapa 1: Criar a máquina de estado

Usando o console do Step Functions, você vai criar uma máquina de estado que inclui um estado `Task` para listar todos os buckets do Amazon S3 na conta e na região atuais. Você então vai adicionar outro estado `Task` que invoca a API `HeadBucket` para verificar se o bucket retornado está acessível na região atual. Se o bucket não estiver acessível, a chamada de API `HeadBucket` retornará o erro `S3.S3Exception`. Você incluirá um bloco `Catch` para capturar essa exceção e um estado `Pass` como estado alternativo.

1. Abra o console do Step Functions e clique em Criar máquina de estado.
2. Na caixa de diálogo Escolher um modelo, selecione Em branco.
3. Escolha Selecionar. Isso abre o Workflow Studio no Modo de design.
4. Neste tutorial, você escreverá a definição da Amazon States Language (ASL) da máquina de estado no Editor de código. Para isso, clique em Código.
5. Remova o código clichê existente e cole a seguinte definição 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 um nome para a máquina de estado. Para fazer isso, escolha o ícone de edição ao lado do nome padrão da máquina de estado de MyStateMachine. Em seguida, em Configuração da máquina de estado, insira um nome na caixa Nome da máquina de estado.

   Para este tutorial, insira o nome **`Gather-S3-Bucket-Info-Standard`**.

7. (Opcional) Em Configuração da máquina de estado, especifique outras configurações do fluxo de trabalho, como o tipo de máquina de estado e a função de execução.

   Mantenha todas as seleções padrão nas configurações da máquina de estado.

   Se você já criou um perfil do IAM com as permissões corretas para a máquina de estado e deseja usá-lo, em Permissões, clique em Escolher um perfil existente e selecione uma função na lista. Ou selecione Inserir um ARN de função e forneça o ARN para esse perfil do IAM.

8. Na caixa de diálogo Confirmar criação do perfil, selecione Confirmar para continuar.

   Você também pode escolher Exibir configurações do perfil para voltar às Configurações da máquina de estado.

   > **Note**
   >
   > Se você excluir o perfil do IAM criado pelo Step Functions, não será possível recriá-lo posteriormente. Da mesma forma, se você modificar a função (por exemplo, removendo o Step Functions das entidades principais na política do IAM), o Step Functions não poderá restaurar as configurações originais dela posteriormente.

   Na Etapa 2, você vai adicionar as permissões ausentes à função de máquina de estado.

Etapa 2: Adicionar permissões do perfil do IAM necessárias

Para coletar informações sobre os buckets do Amazon S3 em sua região atual, forneça à sua máquina de estado as permissões necessárias para acessar os buckets do Amazon S3.

1. Na página da máquina de estado, escolha ARN do perfil do IAM para abrir a página Funções da função da máquina de estado.
2. Escolha Adicionar permissões e depois Criar política em linha.
3. Escolha a guia JSON e cole as permissões a seguir no editor JSON.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "s3:ListAllMyBuckets",
                "s3:ListBucket",
                "s3:GetBucketVersioning"
            ],
            "Resource": "*"
        }
    ]
}
```

4. Escolha Revisar política.
5. No campo Política de revisão, em Nome da política, insira **`s3-bucket-permissions`**.
6. Escolha Criar política.

Etapa 3: Iniciar a execução de uma máquina de estado padrão

1. Na página Gather-S3-Bucket-Info-Standard, escolha Iniciar execução.
2. Na caixa de diálogo Iniciar execução, faça o seguinte:

    a. (Opcional) Para identificar a execução, insira um nome para ela no campo Nome. Por padrão, o Step Functions cria automaticamente um nome de execução exclusivo.

> **Note**
>
> Step Functions permite criar nomes para máquinas de estado, execuções e atividades, além de rótulos que contenham caracteres não ASCII. Esses nomes não ASCII não funcionam com a Amazon. CloudWatch Para garantir que você possa acompanhar CloudWatch as métricas, escolha um nome que use somente caracteres ASCII.

b. Selecione Iniciar execução.

c. O console do Step Functions direciona você para uma página em que o título é o ID da execução. Essa página é conhecida como página de Detalhes da execução. Nesta página, você pode revisar os resultados da execução à medida que a execução avança ou após a conclusão.

   Para revisar os resultados da execução, escolha estados individuais na Exibição em gráfico e, em seguida, escolha as guias individuais no painel Detalhes da etapa para visualizar os detalhes de cada estado, incluindo entrada, saída e definição, respectivamente. Para obter detalhes sobre as informações de execução que você pode visualizar na página Detalhes da execução, consulte Página de Detalhes da execução — Visão geral da interface.

Etapa 4: Executar uma execução de máquina de estado Express

1. Crie uma máquina de estado expresso usando a definição de máquina de estado apresentada na Etapa 1. Inclua também as permissões de perfil do IAM necessárias, conforme explicado na Etapa 2.

   > **Tip**
   >
   > Para diferenciar da máquina padrão criada antes, dê à máquina de estado expresso o nome **`Gather-S3-Bucket-Info-Express`**.

2. Na página Gather-S3-Bucket-Info-Standard, escolha Iniciar execução.
3. Na caixa de diálogo Iniciar execução, faça o seguinte:

   a. (Opcional) Para identificar a execução, insira um nome para ela no campo Nome. Por padrão, o Step Functions cria automaticamente um nome de execução exclusivo.

> **Note**
>
> Step Functions permite criar nomes para máquinas de estado, execuções e atividades, além de rótulos que contenham caracteres não ASCII. Esses nomes não ASCII não funcionam com a Amazon. CloudWatch Para garantir que você possa acompanhar CloudWatch as métricas, escolha um nome que use somente caracteres ASCII.

b. Selecione Iniciar execução.

c. O console do Step Functions direciona você para uma página em que o título é o ID da execução. Essa página é conhecida como página de Detalhes da execução. Nesta página, você pode revisar os resultados da execução à medida que a execução avança ou após a conclusão.

   Para revisar os resultados da execução, escolha estados individuais na Exibição em gráfico e, em seguida, escolha as guias individuais no painel Detalhes da etapa para visualizar os detalhes de cada estado, incluindo entrada, saída e definição, respectivamente. Para obter detalhes sobre as informações de execução que você pode visualizar na página Detalhes da execução, consulte Página de Detalhes da execução — Visão geral da interface.

Ferramentas de desenvolvedor

Os seguintes recursos contêm informações adicionais sobre como criar fluxos de trabalho de tecnologia sem servidor e trabalhar com máquinas de estado:

- AWS CDK
- AWS Kit de ferramentas para VS Code

Os tópicos abaixo contêm informações que ensinam como criar, testar e depurar máquinas de estado.

Tópicos

- Opções de desenvolvimento
- AWS Step Functions e AWS SAM
- Usar o Workflow Studio no Application Composer
- Como criar uma máquina de estado Lambda para o Step Functions usando o AWS CloudFormation
- Criar uma máquina de estado do Lambda para o Step Functions usando o AWS CDK
- Criando uma API REST do API Gateway com o Synchronous Express State Machine usando o AWS CDK
- AWS Step Functions SDK de ciência de dados para Python
- Implantação de máquinas de estado usando o Terraform

Opções de desenvolvimento

Você pode implementar suas máquinas de AWS Step Functions estado de várias maneiras, como usar o console, os SDKs ou uma versão local do Step Functions para testes e desenvolvimento.

Tópicos

- Console do Step Functions
- AWS SDKs
- Fluxos de trabalho Padrão e Expressos

- API de serviço de HTTPS
- Ambientes de desenvolvimento
- Endpoints
- AWS CLI
- Step Functions Local
- AWS Toolkit for Visual Studio Code
- AWS Serverless Application Model e Step Functions
- Terraform e Step Functions
- Suporte ao formato de definição

Console do Step Functions

Você pode definir uma máquina de estado usando o console do Step Functions. Você pode escrever máquinas de estado complexas na nuvem sem usar um ambiente de desenvolvimento local usando AWS Lambda para fornecer código para suas tarefas. Depois de escrito, você pode usar o console do Step Functions para definir a máquina de estado usando a Amazon States Language.

O tutorial Como criar uma máquina de estado Lambda usa essa técnica para criar uma máquina de estado simples, executá-la e visualizar os resultados.

Simulador de fluxo de dados

Você pode projetar, implementar e depurar fluxos de trabalho no console do Step Functions. Você também pode controlar o fluxo de dados nos fluxos de trabalho com o processamento de dados de entrada e saída do `JsonPath`. Use o simulador de fluxo de dados no console do Step Functions para aprender como as informações fluem de um estado para outro e para entender como filtrar e manipular dados. Essa ferramenta simula cada um dos campos que o Step Functions usa para processar dados, como `InputPath`, `Parameters`, `ResultSelector`, `OutputPath` e `ResultPath`.

Para ter mais informações, consulte Simulador de fluxo de dados.

AWS SDKs

O Step Functions é compatível com AWS os SDKs para Java, .NET, Ruby, PHP, Python (Boto 3) JavaScript, Go e C++. Esses SDKs oferecem uma forma conveniente de usar as ações de API HTTPS do Step Functions em várias linguagens de programação.

Você pode desenvolver máquinas de estado, atividades ou iniciadores de máquina de estado usando as ações de API expostas por essas bibliotecas de SDK. Você pode também acessar as operações de visibilidade usando essas bibliotecas para desenvolver ferramentas próprias de monitoramento e de relatório do Step Functions.

Para usar o Step Functions com outros AWS serviços, consulte a documentação de referência dos AWS SDKs e ferramentas atuais da Amazon Web Services.

> **Note**
>
> O Step Functions só oferece suporte a endpoints HTTPS.

Fluxos de trabalho Padrão e Expressos

Ao criar uma nova máquina de estado, você deve selecionar um código `Type` de Padrão ou Expresso. Em ambos os casos, você define a máquina de estado usando a Amazon States Language. Suas execuções de máquina de estado se comportarão de forma diferente, dependendo de qual Type (Tipo) você selecionar. O Tipo escolhido não pode ser alterado depois que a máquina de estado for criada.

Consulte Como registrar usando o CloudWatch Logs Para mais informações.

API de serviço de HTTPS

O Step Functions fornece operações de serviço que são acessíveis por meio de solicitações HTTPS. Você pode usar essas operações para se comunicar diretamente com o Step Functions e desenvolver bibliotecas próprias em qualquer linguagem que possa se comunicar com o Step Functions por meio de HTTPS.

Você pode desenvolver máquinas de estado, operadores ou iniciadores de máquina de estado usando as ações de API do serviço. Você pode também acessar as operações de visibilidade usando as ações de API para desenvolver ferramentas próprias de monitoramento e de relatório.

Para obter informações detalhadas sobre ações de API, consulte a Referência de API do AWS Step Functions.

Ambientes de desenvolvimento

Você precisa configurar um ambiente de desenvolvimento que seja compatível com a linguagem de programação que planeja usar.

Por exemplo, para desenvolver para Step Functions usando Java, você deve instalar um ambiente de desenvolvimento Java, como o AWS SDK for Java, em cada uma de suas estações de trabalho de desenvolvimento. Se usar o Eclipse IDE para Java Developers, você deverá instalar também o AWS Toolkit for Eclipse. Esse plug-in do Eclipse adiciona recursos úteis para o desenvolvimento na AWS.

Quando a linguagem de programação exige um ambiente de runtime, é necessário configurar esse ambiente em cada computador em que esses processos são executados.

Endpoints

Para reduzir a latência e armazenar dados em um local que atenda aos seus requisitos, o Step Functions fornece endpoints em diferentes AWS regiões.

Cada endpoint no Step Functions é totalmente independente. Uma máquina de estado ou atividade só existe na região em que ela foi criada. As máquinas de estado e as atividades criadas em uma região não compartilham dados nem atributos com as que são criadas em outra região. Por exemplo, você pode registrar uma máquina de estado nomeada `STATES-Flows-1` em duas regiões diferentes. A máquina de estado `STATES-Flows-1` em uma região não compartilhará dados ou atributos com a máquina de estado `STATES-Flow-1` na outra região.

Para obter uma lista dos endpoints do Step Functions, consulte Regiões e endpoints do AWS Step Functions na Referência geral da AWS.

AWS CLI

Você pode acessar muitos recursos do Step Functions a partir do AWS Command Line Interface (AWS CLI). AWS CLI É uma alternativa ao uso do console Step Functions ou, em alguns casos, à programação usando as ações da API Step Functions. Por exemplo, você pode usar a AWS CLI para criar uma máquina de estado e, depois, listar as máquinas de estado existentes.

Você pode usar comandos do Step Functions na AWS CLI para iniciar e gerenciar execuções, sondar atividades, registrar heartbeats de tarefas e assim por diante. Para obter uma lista completa de comandos do Step Functions, descrições dos argumentos disponíveis e exemplos que mostram o uso, consulte a Referência de comando da AWS CLI .

AWS CLI os comandos seguem de perto a Amazon States Language, então você pode usá-los AWS CLI para aprender sobre as ações da API Step Functions. Você pode usar também o que já conhece sobre API para fazer protótipos de código ou executar ações do Step Functions por meio da linha de comando.

Step Functions Local

Para fins de testes e desenvolvimento, é possível instalar e executar o Step Functions na máquina local. Com o Step Functions Local, é possível iniciar uma execução em qualquer máquina.

A versão local do Step Functions pode invocar AWS Lambda funções, tanto na execução local AWS quanto durante a execução local. Você também pode coordenar outros AWS serviços suportados. Para ter mais informações, consulte Testando máquinas de estado localmente.

> **Note**
>
> O Step Functions Local usa contas fictícias para funcionar.

AWS Toolkit for Visual Studio Code

É possível usar o VS Code para interagir com máquinas de estado remotas e desenvolver máquinas de estado localmente. É possível criar ou atualizar máquinas de estado, listar máquinas de estado existentes, executá-las e fazer download delas. O VS Code também permite que você crie máquinas de estado com base em modelos e veja uma visualização da máquina de estado, além de fornecer trechos de código, preenchimento de código e validação de código.

Para obter mais informações, consulte o Guia AWS Toolkit for Visual Studio Code do usuário

AWS Serverless Application Model e Step Functions

O Step Functions é integrado ao AWS Serverless Application Model, o que permite integrar fluxos de trabalho com funções, APIs e eventos do Lambda para criar aplicativos sem servidor.

Você também pode usar a AWS SAM CLI em conjunto com o AWS Toolkit for Visual Studio Code como parte de uma experiência integrada.

Para ter mais informações, consulte AWS Step Functions e AWS SAM.

Terraform e Step Functions

O Terraform by HashiCorp é uma estrutura para criar aplicativos usando infraestrutura como código (IaC). Com o Terraform, você pode criar máquinas de estado e usar recursos, como visualizar implantações de infraestrutura e criar modelos reutilizáveis. Os modelos do Terraform ajudam você a manter e reutilizar o código dividindo-o em partes menores.

Para ter mais informações, consulte Implantação de máquinas de estado usando o Terraform.

Suporte ao formato de definição

O Step Functions oferece uma variedade de ferramentas que permitem que você forneça definições de máquina de estado em diferentes formatos. Uma definição de Amazon States Language (ASL) que especifica os detalhes da máquina de estado pode ser fornecida como uma string ou como um objeto serializado usando JSON ou YAML.

> **Note**
>
> O YAML permite comentários de linha única. Quaisquer comentários YAML fornecidos na parte de definição da máquina de estado de um modelo não serão transferidos para a definição do recurso criado. Em vez disso, você pode usar a propriedade `Comment` na definição da máquina de estado. Para obter mais informações, consulte a página do Estrutura da máquina de estado.

As tabelas a seguir mostram quais ferramentas são compatíveis com as definições baseadas em ASL.

Suporte ao formato de definição por ferramenta

	JSON	YAML	Amazon States Language colocada em strings		
Console do Step Functions	✓				

	JSON	YAML	Amazon States Language colocada em strings		
API de serviço de HTTPS			✓		
AWS CLI			✓		
Step Functions Local			✓		
AWS Toolkit for Visual Studio Code	✓	✓			
AWS SAM	✓	✓			
AWS CloudFormation	✓	✓	✓		

> **Note**
>
> AWS CloudFormation e AWS SAM também permitem que você carregue suas definições de máquina de estado para o Amazon S3 no formato JSON ou YAML e forneça a localização da definição no Amazon S3 no modelo. Isso pode melhorar a legibilidade dos modelos quando a definição da máquina de estado for complexa. Para obter mais informações, consulte a página AWS::StepFunctions::StateMachine S3Location.

Os AWS CloudFormation modelos de exemplo a seguir mostram como você pode fornecer a mesma definição de máquina de estado usando 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 e AWS SAM

Você pode usar a AWS SAM CLI em conjunto com o AWS Toolkit for Visual Studio Code como parte de uma experiência integrada para criar máquinas de estado localmente. É possível criar uma aplicação de tecnologia sem servidor com o AWS SAM e compilar a máquina de estado no IDE do VS Code. Depois, você pode validar, empacotar e implantar recursos. Opcionalmente, você também pode publicar no AWS Serverless Application Repository.

> **Tip**
>
> Para implantar um exemplo de aplicativo sem servidor que inicia um fluxo de trabalho do Step Functions usando o AWS SAM seu Conta da AWS, consulte o Módulo 11 - Deploy with AWS SAM do The AWS Step Functions Workshop.

Tópicos

Por que usar Step Functions com AWS SAM?

Ao usar o Step Functions com AWS SAM você pode:

- Comece a usar um modelo AWS SAM de amostra.
- Compilar a máquina de estado no aplicativo sem servidor.
- Usar a substituição de variáveis para substituir ARNs na máquina de estado no momento da implantação.

  O AWS CloudFormation é compatível com `DefinitionSubstitutions` que permitem adicionar referências dinâmicas na definição de fluxo de trabalho a um valor fornecido no modelo do CloudFormation. É possível adicionar referências dinâmicas incluindo substituições na definição de fluxo de trabalho usando a notação `${dollar_sign_brace}`. Você também precisa definir essas referências dinâmicas na `DefinitionSubstitutions` propriedade do StateMachine recurso em seu CloudFormation modelo. Essas substituições são substituídas por valores reais durante o processo de criação da pilha do CloudFormation. Para ter mais informações, consulte DefinitionSubstitutions em AWS SAM modelos.
- Especifique a função da sua máquina de estado usando modelos AWS SAM de política.
- Inicie execuções de máquinas de estado com o API Gateway, EventBridge eventos ou em um cronograma dentro do seu AWS SAM modelo.

Integração do Step Functions com a especificação do AWS SAM

É possível usar os Modelos de política do AWS SAM para adicionar permissões à máquina de estado. Com essas permissões, você pode orquestrar as funções do Lambda e outros recursos da AWS para formar fluxos de trabalho complexos e robustos.

Integração do Step Functions à CLI do SAM

O Step Functions é integrado à AWS SAM CLI. Use isso para desenvolver rapidamente uma máquina de estado no aplicativo sem servidor.

Experimente o Criar uma máquina de estado do Step Functions usando AWS SAM tutorial para aprender a usar AWS SAM para criar máquinas de estado.

As funções de AWS SAM CLI suportadas incluem:

Comando da CLI	Descrição
sam init	Inicializa um aplicativo sem servidor com um modelo. AWS SAM Pode ser usado com um modelo do SAM para o Step Functions.
sam validate	Valida um AWS SAM modelo.
sam package	Empacota um AWS SAM aplicativo. Cria um arquivo ZIP do código e das dependências e faz upload dele no Amazon S3. Depois, ele retorna uma cópia do modelo do AWS SAM , substituindo referências a artefatos locais pelo local do Amazon S3 onde o comando fez upload dos artefatos.
sam deploy	Implanta um AWS SAM aplicativo.
sam publish	Publique um AWS SAM aplicativo no AWS Serverless Application Repository. Esse comando usa um AWS SAM modelo empacotado e publica o aplicativo na região especificada.

> **Note**
>
> Ao usar o AWS SAM local, você pode emular o Lambda e o API Gateway localmente. No entanto, você não pode emular Step Functions localmente usando o. AWS SAM

DefinitionSubstitutions em AWS SAM modelos

É possível definir máquinas de estado usando modelos do CloudFormation com o AWS SAM. Ao usar o AWS SAM, é possível definir a máquina de estado em linha no modelo ou em um arquivo separado. O modelo do AWS SAM a seguir inclui uma máquina de estado que simula um fluxo de trabalho de negociação de ações. Essa máquina de estado invoca três funções do Lambda para conferir o preço de uma ação e determinar se deve comprar ou vender a ação. Essa transação é então registrada em uma tabela do Amazon DynamoDB. Os ARNs das funções do Lambda e da tabela do DynamoDB no modelo a seguir são especificados usando `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
```

O código a seguir é a definição da máquina de estado no arquivo `stock_trader.asl.json` usado no tutorial Criar uma máquina de estado do Step Functions usando AWS SAM. Essa definição da máquina de estado contém várias `DefinitionSubstitutions` indicadas pela notação `${dollar_sign_brace}`. Por exemplo, em vez de especificar um ARN de função estática do Lambda para a tarefa `Check Stock Value`, a substituição `${StockCheckerFunctionArn}` é usada. Essa substituição é definida na propriedade DefinitionSubstitutions do modelo. `DefinitionSubstitutions` é um mapa de pares de chave-valor para o recurso de máquina de estado. Em`DefinitionSubstitutions`, $ {StockCheckerFunctionArn} mapeia para o ARN do `StockCheckerFunction` recurso usando a função CloudFormation intrínseca. `!GetAtt` Ao implantar o modelo do AWS SAM, as `DefinitionSubstitutions` do modelo são substituídas pelos valores reais.

```
{
    "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
        }
    }
}
```

Próximas etapas

Você pode aprender mais sobre o uso do Step Functions AWS SAM com os seguintes recursos:

- Conclua o Criar uma máquina de estado do Step Functions usando AWS SAM tutorial para criar uma máquina de estado com AWS SAM o.

- Especifique um AWS::Serverless::StateMachinerecurso.
- Localize os Modelos de política do AWS SAM a serem usados.
- Use o AWS Toolkit for Visual Studio Code com o Step Functions.
- Analise a referência da CLI do AWS SAM para saber mais sobre os recursos disponíveis no AWS SAM.

Também é possível projetar e criar fluxos de trabalho na infraestrutura como código (IaC) usando criadores visuais, como o Workflow Studio no Application Composer. Para ter mais informações, consulte Usar o Workflow Studio no Application Composer.

Usar o Workflow Studio no Application Composer

O AWS Application Composer é um criador visual que ajuda a desenvolver modelos do AWS SAM e do AWS CloudFormation usando uma interface gráfica simples. Com o Application Composer, você vai criar uma arquitetura de aplicação arrastando, agrupando e conectando os Serviços da AWS em uma tela. Depois, o Application Composer criará um modelo de infraestrutura como código (IaC) com base no projeto que você pode usar para implantar a aplicação com a interface de linha de comandos do AWS SAM (CLI do AWS SAM) ou o CloudFormation. Para saber mais sobre o Application Composer, consulte O que é o Application Composer.

O Workflow Studio está disponível no Application Composer para ajudar a projetar e criar fluxos de trabalho. O Workflow Studio no Application Composer oferece um ambiente visual de IaC que facilita a incorporação de fluxos de trabalho às aplicações sem servidor criadas com ferramentas de IaC, como modelos do CloudFormation. Ao usar o Workflow Studio no Application Composer, ele conecta as etapas individuais do fluxo de trabalho aos recursos da AWS e gera as configurações dos recursos em um modelo do AWS SAM. Ele também adiciona as permissões do IAM necessárias para que o fluxo de trabalho seja executado. Usando o Workflow Studio no Application Composer, é possível criar protótipos de aplicações e transformá-los em aplicações prontas para produção.

Ao usar o Workflow Studio no Application Composer, é possível alternar entre a tela do Application Composer e o Workflow Studio.

Tópicos

- Usar o Workflow Studio no Application Composer para criar fluxos de trabalho sem servidor
- Referenciar recursos dinamicamente usando substituições de definição do CloudFormation no Workflow Studio

- Conectar as tarefas de integração de serviços às placas de componentes aprimoradas
- Importar projetos existentes e sincronizá–los localmente
- Recursos indisponíveis do Workflow Studio no AWS Application Composer

Usar o Workflow Studio no Application Composer para criar fluxos de trabalho sem servidor

1. Abra o console do Application Composer e selecione Criar projeto para criar um projeto.
2. No campo de pesquisa na paleta Recursos, insira **`state machine`**.
3. Arraste o recurso Máquina de estado do Step Functions para a tela.
4. Selecione Editar no Workflow Studio para editar o recurso de máquina de estado.

   A animação a seguir mostra como mudar para o Workflow Studio para editar a definição de máquina de estado.
   Uma animação que ilustra como usar o Workflow Studio no Application Composer.

   A integração com o Workflow Studio para editar recursos de máquinas de estado criados no Application Composer só está disponível para o recurso `AWS::Serverless::StateMachine`. Essa integração não está disponível para modelos que usam o recurso `AWS::StepFunctions::StateMachine`.

Referenciar recursos dinamicamente usando substituições de definição do CloudFormation no Workflow Studio

No Workflow Studio, é possível usar substituições de definição do CloudFormation na definição de fluxo de trabalho para referenciar dinamicamente os recursos definidos no modelo de IaC. É possível adicionar substituições de espaço reservado à definição de fluxo de trabalho usando a notação `${dollar_sign_brace}`. Elas são substituídas por valores reais durante o processo de criação da pilha do CloudFormation. Para obter mais informações sobre substituição de definições, consulte DefinitionSubstitutions em AWS SAM modelos.

A animação a seguir mostra como adicionar substituições de espaço reservado para os recursos na definição de máquina de estado.
Uma animação que ilustra como referenciar dinamicamente recursos, como funções do AWS Lambda e substituições de definição ao usar o Workflow Studio no Application Composer.

Conectar as tarefas de integração de serviços às placas de componentes aprimoradas

É possível conectar as tarefas que chamam integrações de serviços otimizadas às placas de componentes aprimoradas na tela do Application Composer. Esse procedimento associa todas as substituições de espaço reservado especificadas pela notação `${dollar_sign_brace}` na definição do fluxo de trabalho e a propriedade `DefinitionSubstitution` do recurso `StateMachine`. Também adiciona as políticas do AWS SAM apropriadas à máquina de estado.

Se você associar tarefas otimizadas de integração de serviços a placas de componentes padrão, a linha de conexão não aparecerá na tela Application Composer.

A animação a seguir mostra como conectar uma tarefa otimizada a uma placa de componente aprimorado e visualizar as alterações no Change Inspector.
Uma animação que ilustra como conectar tarefas que chamam integrações de serviços otimizadas a placas de componentes aprimoradas ao usar o Workflow Studio no Application Composer.

Não é possível conectar integrações de SDKs da AWS no estado de Tarefa com placas de componentes aprimoradas ou integrações de serviços otimizadas com placas de componentes padrão. Para essas tarefas, é possível associar as substituições no painel Propriedades do recurso na tela do Application Composer e adicionar políticas ao modelo do AWS SAM.

> **Tip**
>
> Também é possível associar substituições de espaço reservado para a máquina de estado em Substituições de definição no painel Propriedades do recurso. Ao fazer isso, é necessário adicionar as permissões necessárias para o AWS service (Serviço da AWS) que o estado da Tarefa chama no perfil de execução da máquina de estado. Para obter informações sobre as permissões necessárias a um perfil de execução, consulte Perfis de execução no Workflow Studio.

A animação a seguir mostra como você pode atualizar manualmente o mapeamento de substituição de espaço reservado no painel Propriedades do recurso.
Uma animação que ilustra como atualizar manualmente o mapeamento de substituição de espaço reservado no painel Propriedades do recurso ao usar o Workflow Studio no Application Composer.

Importar projetos existentes e sincronizá–los localmente

É possível abrir projetos existentes do CloudFormation e do AWS SAM no Application Composer para visualizá-los para entender melhor e modificar os projetos. Usando o atributo de sincronização local do Application Composer, é possível sincronizar e salvar automaticamente os arquivos de modelo e código na máquina de compilação local. O uso do modo de sincronização local pode complementar os fluxos de desenvolvimento existentes. Assegure-se de que o navegador seja compatível com a API do File System Access, que permite a uma aplicação da web ler, gravar e salvar arquivos no sistema de arquivos local. Recomendamos usar o Google Chrome ou o Microsoft Edge.

Recursos indisponíveis do Workflow Studio no AWS Application Composer

Ao usar o Workflow Studio no Application Composer, alguns recursos do Workflow Studio não estão disponíveis. Além disso, a seção Parâmetros da API disponível no painel Inspector é compatível com substituições de definição do CloudFormation. É possível adicionar as substituições no Modo de código usando a notação `${dollar_sign_brace}`. Para obter mais informações sobre essa notação, consulte DefinitionSubstitutions em AWS SAM modelos.

A lista a seguir descreve os recursos do Workflow Studio que não estão disponíveis ao usar o Workflow Studio no Application Composer:

- Modelos iniciais: os modelos iniciais são exemplos de projetos prontos para execução que criam automaticamente os protótipos e as definições do fluxo de trabalho. Esses modelos implantam na Conta da AWS todos os recursos da AWS relacionados de que o projeto precisa.
- Modo de configuração: esse modo permite gerenciar a configuração das máquinas de estado. É possível atualizar as configurações da máquina de estado nos modelos de IaC ou usar o painel Propriedades do recurso na tela do Application Composer. Para obter informações sobre a atualização de configurações no painel Propriedades do recurso, consulte Conectar as tarefas de integração de serviços às placas de componentes aprimoradas.
- API TestState
- Opção para importar ou exportar definições de fluxo de trabalho no botão suspenso Ações do Workflow Studio. Em vez disso, no menu do Application Composer, selecione Abrir > Pasta do projeto. Certifique-se de ter habilitado o modo de sincronização local para salvar automaticamente as alterações na tela do Application Composer diretamente na máquina local.
- Botão Executar. Ao usar o Workflow Studio no Application Composer, o Application Composer gera o código IaC para o fluxo de trabalho. Portanto, primeiro será necessário implantar o modelo.

Depois, execute o fluxo de trabalho no console ou por meio da AWS Command Line Interface (AWS CLI).

Como criar uma máquina de estado Lambda para o Step Functions usando o AWS CloudFormation

Este tutorial mostra como criar uma AWS Lambda função básica usando AWS CloudFormation o. Você usará o AWS CloudFormation console e um modelo YAML para criar a pilha (funções do IAM, a função Lambda e a máquina de estado). Em seguida, você usará o AWS Step Functions console para iniciar a execução da máquina de estado.

Para obter mais informações, consulte Trabalhando com CloudFormation modelos e o `AWS::StepFunctions::StateMachine` recurso no Guia AWS CloudFormation do usuário.

Tópicos

Etapa 1: configurar seu AWS CloudFormation modelo

Antes de usar os modelos de exemplo, é necessário entender como declarar as diferentes partes de um modelo do AWS CloudFormation .

Tópicos

Para criar uma função do IAM para Lambda

Defina a política de confiança associada ao perfil do IAM para a função do Lambda. Os exemplos a seguir definem uma política de confiança usando YAML ou 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"
                      }
                  ]
              }
          }
```

Criar uma função do Lambda

Defina as propriedades a seguir da função do Lambda que imprimirão a mensagem `Hello World`.

> **Important**
>
> Certifique-se de que sua função Lambda esteja na mesma AWS conta e AWS região da sua máquina estadual.

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 criar um perfil do IAM para a execução da máquina de estado

Defina a política de confiança associada ao perfil do IAM para a execução da 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 criar uma máquina de estado do Lambda

Defina a máquina de estado do 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"
                    ]
                }
            }
        }
```

Etapa 2: usar o AWS CloudFormation modelo para criar uma máquina de estado Lambda

Depois de entender os componentes do AWS CloudFormation modelo, você pode juntá-los e usar o modelo para criar uma AWS CloudFormation pilha.

Para criar a máquina de estado do Lambda

1. Copie os dados de exemplo a seguir em um arquivo chamado `MyStateMachine.yaml` para o exemplo de YAML ou `MyStateMachine.json` para 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

```
{
```

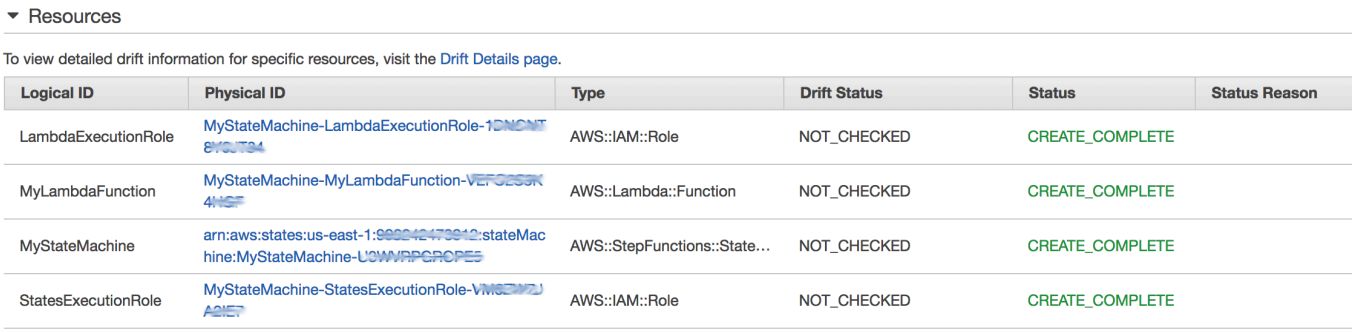
```
   "AWSTemplateFormatVersion": "2010-09-09",
   "Description": "An example template with an IAM role for a Lambda state
machine.",
   "Resources": {
       "LambdaExecutionRole": {
           "Type": "AWS::IAM::Role",
           "Properties": {
               "AssumeRolePolicyDocument": {
                   "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 o console do AWS CloudFormation e escolha Create Stack (Criar pilha).
3. Na página Select Template (Selecionar modelo), selecione Upload a template to Amazon S3 (Fazer upload de um modelo para o Amazon S3). Escolha seu arquivo `MyStateMachine` e, em seguida, Next.
4. Na página Specify Details (Especificar detalhes), em Stack Name (Nome da pilha), insira `MyStateMachine` e escolha Next (Próximo).
5. Na página Options (Opções), escolha Next (Avançar).
6. Na página Revisão, escolha Eu reconheço que o AWS CloudFormation pode criar recursos do IAM. e, em seguida, escolha Criar.

   AWS CloudFormation começa a criar a `MyStateMachine` pilha e exibe o status CREATE_IN_PROGRESS. Quando o processo é concluído, o AWS CloudFormation exibe o status CREATE_COMPLETE.
7. (Opcional) Para exibir os recursos em sua pilha, selecione a pilha e escolha a guia Resources.



▼ 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	

Etapa 3: iniciar a execução de uma máquina de estado

Assim que você criar a máquina de estado do Lambda, poderá iniciar uma execução.

Para iniciar a execução da máquina de estado

1. Abra o console Step Functions e escolha o nome da máquina de estado que você criou usando AWS CloudFormation.
2. Na página ***MyStateMachine-ABCDEFGHIJ1K***, escolha Nova execução.

   A página New execution é exibida.
3. (Opcional) Para identificar a execução, insira um nome para ela no campo Nome. Por padrão, o Step Functions cria automaticamente um nome de execução exclusivo.

   > **Note**
   >
   > Step Functions permite criar nomes para máquinas de estado, execuções e atividades, além de rótulos que contenham caracteres não ASCII. Esses nomes não ASCII não funcionam com a Amazon. CloudWatch Para garantir que você possa acompanhar CloudWatch as métricas, escolha um nome que use somente caracteres ASCII.
4. Escolha Start Execution.

   Uma nova execução de sua máquina de estado inicia-se e uma nova página mostrando a execução em andamento é exibida.
5. (Opcional) Na seção Execution Details (Detalhes da execução), examine o Execution Status (Status da execução) e os timestamps Started (Iniciado) e Closed (Fechado).
6. Para visualizar os resultados de sua execução, selecione Output (Saída).

Criar uma máquina de estado do Lambda para o Step Functions usando o AWS CDK

Este tutorial mostra como criar uma máquina de estado do AWS Step Functions que contém uma função do AWS Lambda usando o AWS Cloud Development Kit (AWS CDK). AWS CDKÉ uma estrutura de infraestrutura como código (IAC) que permite definir a AWS infraestrutura usando uma linguagem de programação completa. Você pode escrever uma aplicação em uma das linguagens com suporte do CDK contendo uma ou mais pilhas. Em seguida, você pode sintetizá-lo em um AWS CloudFormation modelo e implantá-lo em sua AWS conta. Usaremos esse método para definir uma máquina de Step Functions estado contendo uma Lambda função a e, em seguida, usaremos o AWS Management Console para executar a máquina de estado.

Antes de começar este tutorial, você deve configurar seu ambiente de desenvolvimento do AWS CDK conforme descrito em Introdução aos AWS CDK – pré-requisitos no Guia do desenvolvedor do AWS Cloud Development Kit (AWS CDK) . Então instale o AWS CDK com o seguinte comando na AWS CLI:

```
npm install -g aws-cdk
```

Este tutorial produz o mesmo resultado que the section called “Criando uma máquina de estado Lambda usando AWS CloudFormation”. Porém, neste tutorial, o AWS CDK não exige que você crie nenhuma função do IAM; o AWS CDK faz isso por você. A versão do AWS CDK também inclui uma etapa Succeed para ilustrar como adicionar mais etapas à sua máquina de estado.

> **Tip**
>
> Para implantar um exemplo de aplicativo sem servidor que inicia um Step Functions fluxo de trabalho usando AWS CDK with TypeScript to your Conta da AWS, consulte o Módulo 10 - Implantar com AWS CDK do The AWS Step Functions Workshop.

Tópicos

- Próximas etapas

Etapa 1: configurar o projeto do AWS CDK

1. No diretório inicial ou em outro diretório, se preferir, execute o comando a seguir para criar um diretório para a nova aplicação AWS CDK.

   > **Important**
   >
   > Dê ao diretório o nome `step`. O modelo de aplicação do AWS CDK usa o nome do diretório para gerar nomes para arquivos e classes de origem. Se você escolher outro nome, sua aplicação não combinará com este tutorial.

   TypeScript

   ```
   mkdir step && cd step
   ```

   JavaScript

   ```
   mkdir step && cd step
   ```

   Python

   ```
   mkdir step && cd step
   ```

   Java

   ```
   mkdir step && cd step
   ```

   C#

   Verifique se você instalou o .NET versão 6.0 ou superior. Para obter informações, consulte Versões compatíveis.

   ```
   mkdir step && cd step
   ```

2. Inicialize o aplicativo usando o comando cdk init. Especifique o modelo (“aplicação”) e a linguagem de programação desejados conforme mostrado nos exemplos a seguir.

   TypeScript

   ```
   cdk init --language typescript
   ```

   JavaScript

   ```
   cdk init --language javascript
   ```

   Python

   ```
   cdk init --language python
   ```

   Depois que o projeto foi inicializado, ative o ambiente virtual do projeto e instale as dependências de linha de base do AWS CDK.

   ```
   source .venv/bin/activate
   python -m pip install -r requirements.txt
   ```

   Java

   ```
   cdk init --language java
   ```

   C#

   ```
   cdk init --language csharp
   ```

Etapa 2: Usar o AWS CDK para criar uma máquina de estado

Primeiro, vamos apresentarem as partes individuais do código que definem a função do Lambda e a máquina de estado do Step Functions. Em seguida, explicaremos como colocá-los juntos em sua aplicação AWS CDK. Por fim, você verá como sintetizar e implantar esses recursos.

Como criar uma função do Lambda

O código AWS CDK a seguir define a função Lambda, fornecendo seu código-fonte em linha.

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)
});
```

Você pode ver, neste exemplo de código curto:

- O nome lógico da função, `MyLambdaFunction`.
- O código-fonte da função, incorporado como uma string no código-fonte da aplicação AWS CDK.
- Outros atributos da função, como o runtime a ser usado (Node 18.x), o ponto de entrada da função e um tempo limite.

Para criar uma máquina de estado do

Nossa máquina de estados tem dois estados: uma função do Lambda, tarefa e um estado do Succeed. A função exige criar um Step Functions the section called “Tarefa” que invoca nossa função. Esse estado de tarefa é usado como a primeira etapa na máquina de estado. O estado de sucesso é adicionado à máquina de estado usando o método `next()` do estado da tarefa. O código a seguir primeiro invoca a função chamada `MyLambdaTask` e depois, usa o método `next()` para definir um estado de sucesso chamado `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 criar e implantar a aplicação AWS CDK

No projeto AWS CDK que acaba de ser criado, edite o arquivo que contém a definição de pilha para que seja parecido com o código de exemplo a seguir. Você vai reconhecer as definições da função do Lambda e da máquina de estado do Step Functions das seções anteriores.

1. Atualize a pilha como mostrado nos exemplos a seguir.

   TypeScript

   Atualize o `lib/step-stack.ts` com o código a seguir.

```
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

   Atualize o `lib/step-stack.js` com o código a seguir.

```
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

Atualize o `step/step_stack.py` com o código a seguir.

```
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

Atualize o `src/main/java/com.myorg/StepStack.java` com o código a seguir.

```
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#

Atualize o `scr/Step/StepStack.cs` com o código a seguir.

```
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. Salve o arquivo de origem e execute o comando `cdk synth` no diretório principal da aplicação.

   O AWS CDK executa a aplicação e sintetiza um modelo do AWS CloudFormation com base nela. O AWS CDK então exibe o modelo.

   > **Note**
   >
   > Se você TypeScript costumava criar seu AWS CDK projeto, a execução do `cdk synth` comando pode retornar o seguinte erro.
   >
   > ```
   > TSError: # Unable to compile TypeScript:
   > bin/step.ts:7:33 - error TS2554: Expected 2 arguments, but got 3.
   > ```
   >
   > Modifique o arquivo `bin/step.ts` conforme mostrado no exemplo a seguir para resolver esse erro.
   >
   > ```
   > #!/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 implantar a função do Lambda e a máquina de estado do Step Functions em sua conta da AWS , emita `cdk deploy`. Você deverá aprovar as políticas do IAM que foram AWS CDK geradas.

Etapa 3: Iniciar a execução de uma máquina de estado

Assim que você criar sua máquina de estado, poderá iniciar sua execução.

Para iniciar a execução da máquina de estado

1. Abra o console do Step Functions e escolha o nome da máquina de estado que você criou usando o AWS CDK.
2. Na página da máquina de estado, escolha Iniciar execução.

   A caixa de diálogo Iniciar execução é exibida.
3. (Opcional) Para identificar a execução, insira um nome para ela no campo Nome. Por padrão, o Step Functions cria automaticamente um nome de execução exclusivo.

   > **Note**
   >
   > Step Functions permite criar nomes para máquinas de estado, execuções e atividades, além de rótulos que contenham caracteres não ASCII. Esses nomes não ASCII não funcionam com a Amazon. CloudWatch Para garantir que você possa acompanhar CloudWatch as métricas, escolha um nome que use somente caracteres ASCII.
4. Escolha Start Execution.

   A execução da máquina de estado começa e uma nova página mostrando a execução em andamento é exibida.
5. O console do Step Functions direciona você para uma página em que o título é o ID da execução. Essa página é conhecida como página de Detalhes da execução. Nesta página, você pode revisar os resultados da execução à medida que a execução avança ou após a conclusão.

Para revisar os resultados da execução, escolha estados individuais na Exibição em gráfico e, em seguida, escolha as guias individuais no painel Detalhes da etapa para visualizar os detalhes de cada estado, incluindo entrada, saída e definição, respectivamente. Para obter detalhes sobre as informações de execução que você pode visualizar na página Detalhes da execução, consulte Página de Detalhes da execução — Visão geral da interface.

Etapa 4: Limpeza

Depois de testar sua máquina de estado, recomendamos remover a máquina de estado e a função do Lambda relacionada para liberar recursos na sua Conta da AWS. Execute o comando `cdk destroy` no diretório principal da aplicação para remover sua máquina de estado.

Próximas etapas

Para saber mais sobre como desenvolver a AWS infraestrutura usandoAWS CDK, consulte o Guia do AWS CDK desenvolvedor.

Para obter informações sobre como criar aplicações do AWS CDK na sua linguagem preferencial, consulte:

TypeScript

- Trabalhando com AWS CDK em TypeScript

JavaScript

- Trabalhando com AWS CDK em JavaScript

Python

- Trabalhar com o AWS CDK no Python

Java

- Trabalhar com o AWS CDK no Java

C#

- Trabalhar com o AWS CDK no C#

Para obter mais informações sobre os módulos da AWS Construct Library usados neste tutorial, consulte as seguintes visões gerais de referência de AWS CDK API:

- aws-lambda
- aws-stepfunctions
- aws-stepfunctions-tasks

Criando uma API REST do API Gateway com o Synchronous Express State Machine usando o AWS CDK

Este tutorial mostra como criar uma API REST do API Gateway com a Máquina de estado expresso síncrona como integração de backend usando o AWS Cloud Development Kit (AWS CDK). Este tutorial usará a estrutura `StepFunctionsRestApi` para conectar a Máquina de estado ao API Gateway. A estrutura `StepFunctionsRestApi` vai configurar um mapeamento padrão de entrada/saída e a API REST do API Gateway, com as permissões necessárias e um método HTTP "ANY". AWS CDK É uma estrutura de infraestrutura como código (IAC) que permite definir a AWS infraestrutura usando uma linguagem de programação completa. Você escreve um aplicativo em uma das linguagens suportadas pelo CDK, contendo uma ou mais pilhas, depois o sintetiza em um AWS CloudFormation modelo e o implanta em sua conta. AWS Vamos usá-lo para definir uma API REST do API Gateway, que é integrada ao Synchronous Express State Machine como back-end, e usaremos o AWS Management Console para iniciar a execução.

Antes de iniciar este tutorial, configure seu ambiente de AWS CDK desenvolvimento conforme descrito em Introdução aos AWS CDK - Pré-requisitos e, em seguida, instale o emitindo: AWS CDK

```
npm install -g aws-cdk
```

Tópicos

- Etapa 1: Configurar o projeto do AWS CDK
- Etapa 2: Use o AWS CDK para criar uma API REST do API Gateway com integração de back-end do Synchronous Express State Machine
- Etapa 3: Testar o API Gateway
- Etapa 4: Limpeza

Etapa 1: Configurar o projeto do AWS CDK

Primeiro, crie um diretório para seu novo AWS CDK aplicativo e inicialize o projeto.

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
```

Depois que o projeto for inicializado, ative o ambiente virtual do projeto e instale as dependências básicas AWS CDK do projeto.

```
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**
>
> Dê ao diretório o nome `stepfunctions-rest-api`. O modelo de AWS CDK aplicativo usa o nome do diretório para gerar nomes para classes e arquivos de origem. Se você escolher outro nome, sua aplicação não combinará com este tutorial.

Agora, instale os módulos da biblioteca de construção para o 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 o `pom.xml` do projeto para adicionar as seguintes dependências dentro do contêiner `<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>
```

O Maven instala automaticamente essas dependências na próxima vez que você criar a aplicação. Para criar, inicie `mvn compile` ou use o comando Build (Criar) do Java IDE.

C#

```
dotnet add src/StepfunctionsRestApi package Amazon.CDK.AWS.Stepfunctions
dotnet add src/StepfunctionsRestApi package Amazon.CDK.AWS.APIGateway
```

Você também pode instalar os pacotes indicados usando a NuGet GUI do Visual Studio, disponível em Tools > NuGet Package Manager > Manage NuGet Packages for Solution.

Depois de instalar os módulos, você pode usá-los em seu AWS CDK aplicativo importando os pacotes a seguir.

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

Adicione o seguinte a `import` dentro de `stepfunctions-rest-api.go`.

```
"github.com/aws/aws-cdk-go/awscdk/awsapigateway"
"github.com/aws/aws-cdk-go/awscdk/awsstepfunctions"
```

Etapa 2: Use o AWS CDK para criar uma API REST do API Gateway com integração de back-end do Synchronous Express State Machine

Primeiro, apresentaremos as partes individuais do código que definem a Synchronous Express State Machine e a API REST do API Gateway e, em seguida, explicaremos como reuni-las em seu AWS CDK aplicativo. Então você verá como sintetizar e implantar esses recursos.

> **Note**
>
> A Máquina de Estado que mostraremos aqui será uma Máquina de Estado simples com um estado `Pass`.

Para criar uma máquina de estado Express

Esse é o AWS CDK código que define uma máquina de estado simples com um `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,
})
```

Você pode ver, neste pequeno trecho:

- A definição da máquina chamada `PassState`, que é um estado `Pass`.
- O nome lógico da Máquina do estado, `MyStateMachine`.
- A definição da máquina é usada como a definição da máquina de estado.
- O Tipo de máquina de estado é definido como `EXPRESS` porque `StepFunctionsRestApi` só permitirá uma máquina de estado do Synchronous Express.

Para criar a API REST do API Gateway usando a estrutura **`StepFunctionsRestApi`**

Vamos usar a estrutura `StepFunctionsRestApi` para criar a API REST do API Gateway com as permissões necessárias e mapeamento padrão de entrada/saída.

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 criar e implantar a aplicação AWS CDK

No AWS CDK projeto que você criou, edite o arquivo contendo a definição da pilha para ficar parecido com o código abaixo. Você vai reconhecer as definições da máquina de estado do Step Functions e do API Gateway apresentadas acima.

TypeScript

Atualizar `lib/stepfunctions-rest-api-stack.ts` para ler conforme mostrado a seguir.

```
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

Atualizar `lib/stepfunctions-rest-api-stack.js` para ler conforme mostrado a seguir.

```
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

Atualizar `stepfunctions_rest_api/stepfunctions_rest_api_stack.py` para ler conforme mostrado a seguir.

```
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

Atualizar `src/main/java/com.myorg/StepfunctionsRestApiStack.java` para ler conforme mostrado a seguir.

```
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#

Atualizar `src/StepfunctionsRestApi/StepfunctionsRestApiStack.cs` para ler conforme mostrado a seguir.

```
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

Atualizar `stepfunctions-rest-api.go` para ler conforme mostrado a seguir.

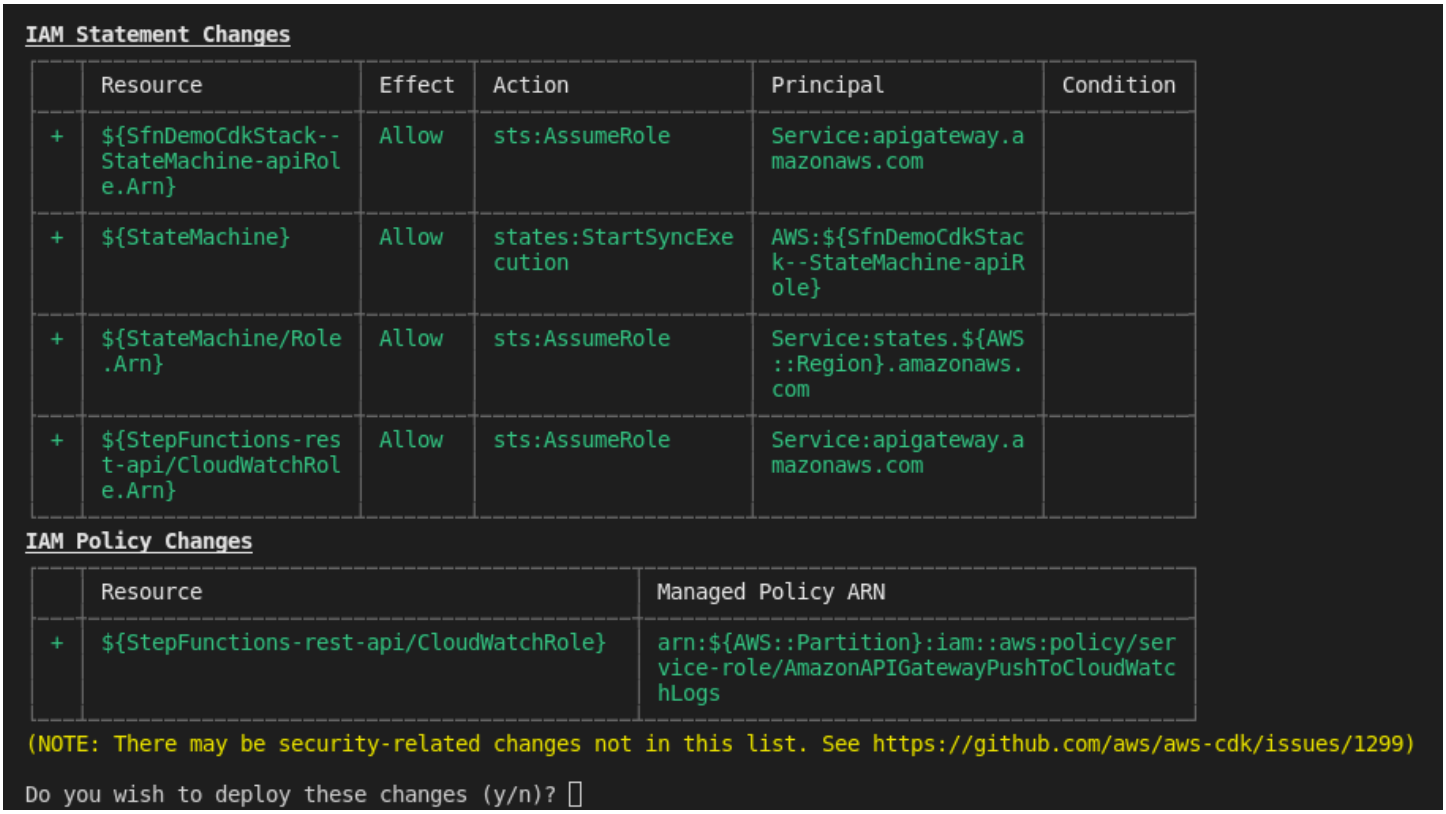
```
package main
import (
    "github.com/aws/aws-cdk-go/awscdk"
    "github.com/aws/aws-cdk-go/awscdk/awsapigateway"
    "github.com/aws/aws-cdk-go/awscdk/awsstepfunctions"
    "github.com/aws/constructs-go/constructs/v3"
    "github.com/aws/jsii-runtime-go"
)
```

```
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")),
    // }
}
```

Salve o arquivo de origem e emita `cdk synth` no diretório principal da aplicação. O AWS CDK executa o aplicativo e sintetiza um AWS CloudFormation modelo a partir dele e, em seguida, exibe o modelo.

Para realmente implantar o Amazon API Gateway e a máquina de AWS Step Functions estado em sua conta da AWS, emita`cdk deploy`. Você deverá aprovar as políticas do IAM que foram AWS CDK geradas. As políticas que estão sendo criadas serão mais ou menos assim:



IAM Statement Changes

	Resource	Effect	Action	Principal	Condition
+	${SfnDemoCdkStack--StateMachine-apiRole.Arn}	Allow	sts:AssumeRole	Service:apigateway.amazonaws.com	
+	${StateMachine}	Allow	states:StartSyncExecution	AWS:${SfnDemoCdkStack--StateMachine-apiRole}	
+	${StateMachine/Role.Arn}	Allow	sts:AssumeRole	Service:states.${AWS::Region}.amazonaws.com	
+	${StepFunctions-rest-api/CloudWatchRole.Arn}	Allow	sts:AssumeRole	Service:apigateway.amazonaws.com	

IAM Policy Changes

	Resource	Managed Policy ARN
+	${StepFunctions-rest-api/CloudWatchRole}	arn:${AWS::Partition}:iam::aws:policy/service-role/AmazonAPIGatewayPushToCloudWatchLogs

(NOTE: There may be security-related changes not in this list. See https://github.com/aws/aws-cdk/issues/1299)

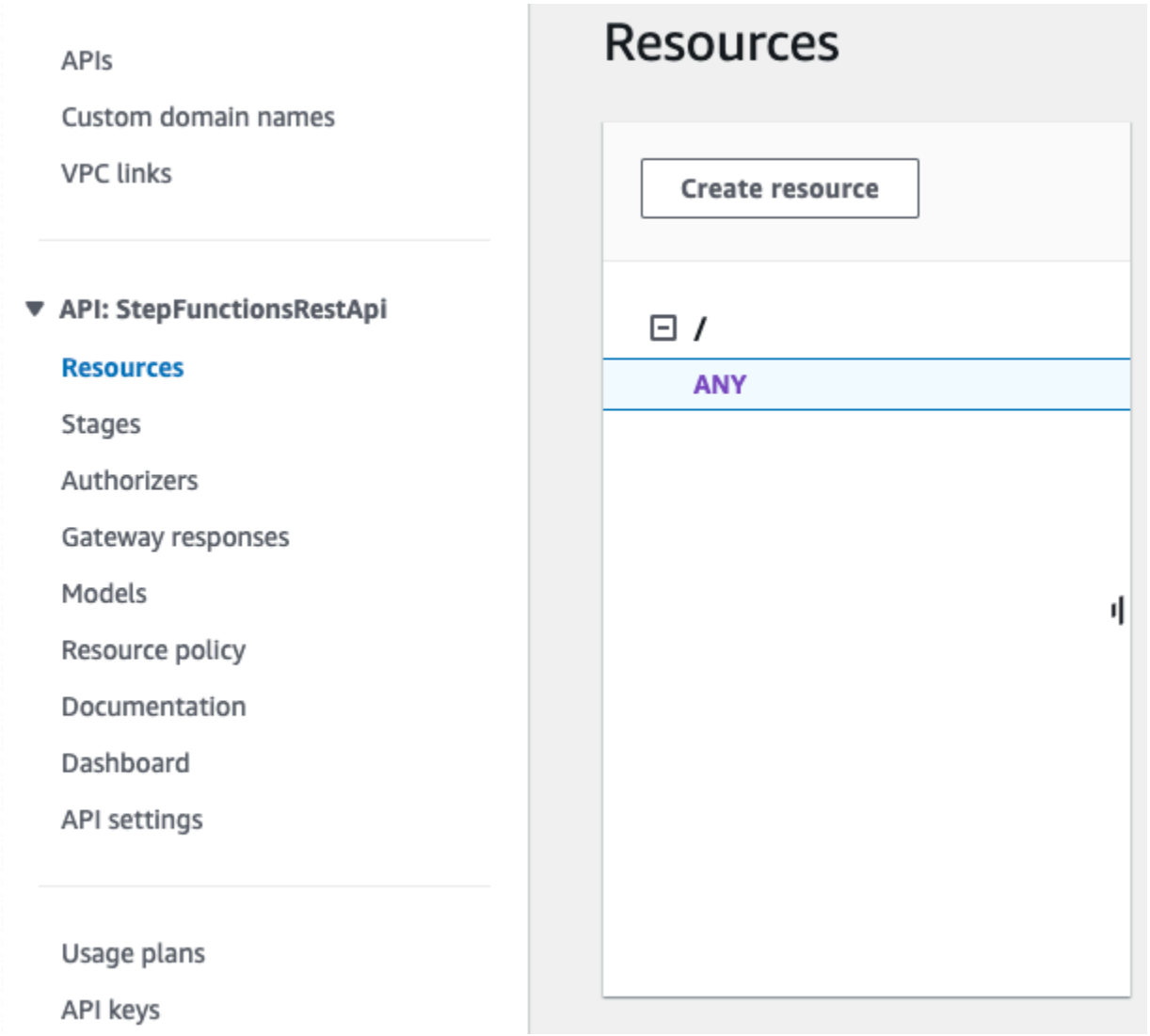
Do you wish to deploy these changes (y/n)?

Etapa 3: Testar o API Gateway

Depois de criar sua API REST do API Gateway com a máquina de estado síncrona expressa como integração de back-end, você pode testar o API Gateway.

Para testar o API Gateway implantado usando o console do API Gateway

1. Abra o console do Amazon API Gateway e faça login.
2. Selecione a API REST denominada `StepFunctionsRestApi`.
3. No painel Recursos, selecione o método `ANY`.



4. Selecione a guia Testar. Talvez seja necessário selecionar o botão de seta para a direita para mostrar a guia.
5. Em Method (Método), selecione POST.
6. Em Corpo da solicitação, copie os parâmetros da solicitação a seguir.

```
{
    "key": "Hello"
}
```

7. Escolha Test (Testar). As informações a seguir serão exibidas:
   - Request (Solicitação) é o caminho do recurso que foi chamado para o método.
   - Status é o código de status HTTP da resposta.

- Latency (Latência) é o tempo entre a recepção da solicitação do autor da chamada e a resposta retornada.
- Corpo da resposta é o corpo de resposta HTTP.
- Cabeçalhos de resposta são os cabeçalhos de resposta HTTP.
- O log mostra as entradas simuladas do Amazon CloudWatch Logs que teriam sido gravadas se esse método fosse chamado fora do console do API Gateway.

> **Note**
>
> Embora as entradas de CloudWatch registros sejam simuladas, os resultados da chamada do método são reais.

A saída do Corpo da resposta deve ser mais ou menos assim:

```
"Hello"
```

> **Tip**
>
> Experimente o API Gateway com métodos diferentes e uma entrada inválida para ver a saída do erro. É recomendável alterar a máquina de estado para procurar uma chave específica e, durante o teste, fornecer a chave errada para provocar uma falha na execução da máquina de estado e gerar uma mensagem de erro na saída do Corpo da resposta.

Para testar a API implantada usando cURL

1. Abra uma janela do terminal.
2. Copie o seguinte comando cURL e cole-o na janela do terminal, substituindo `<api-id>` pelo ID de API da sua API e `<region>` pela região em que a API foi implantada.

```
curl -X POST\
 'https://<api-id>.execute-api.<region>.amazonaws.com/prod' \
 -d '{"key":"Hello"}' \
 -H 'Content-Type: application/json'
```

A saída do Corpo da resposta deve ser mais ou menos assim:

```
"Hello"
```

> **Tip**
>
> Experimente o API Gateway com métodos diferentes e uma entrada inválida para ver a saída do erro. Talvez você queira alterar a máquina de estado para procurar uma chave específica e, durante o teste, fornecer a chave errada para falhar na execução da Máquina de Estado e gerar uma mensagem de erro na saída do Corpo de Resposta.

Etapa 4: Limpeza

Ao terminar de testar seu API Gateway, você pode derrubar tanto a máquina de estado quanto o API Gateway usando o AWS CDK. Inicie `cdk destroy` no diretório principal da aplicação.

AWS Step Functions SDK de ciência de dados para Python

O AWS Step Functions Data Science SDK é uma biblioteca de código aberto para cientistas de dados. Com esse SDK, você pode criar fluxos de trabalho que processam e publicam modelos de aprendizado de máquina usando Step SageMaker Functions. Você também pode criar fluxos de trabalho de aprendizado de máquina em várias etapas em Python que orquestram a AWS infraestrutura em grande escala, sem precisar provisionar e integrar os serviços separadamente. AWS

O AWS Step Functions Data Science SDK fornece uma API do Python que pode criar e invocar fluxos de trabalho do Step Functions. Você pode gerenciar e executar esses fluxos de trabalho diretamente no Python, assim como em cadernos Jupyter.

Além de criar fluxos de trabalho prontos para produção diretamente em Python, o SDK de Ciência de AWS Step Functions Dados permite que você copie esse fluxo de trabalho, experimente novas opções e, em seguida, coloque o fluxo de trabalho refinado em produção.

Para obter mais informações sobre o SDK de Ciência de AWS Step Functions Dados, consulte o seguinte:

- Projeto no Github
- Documentação do SDK

- Os seguintes exemplos de notebooks, que estão disponíveis nas instâncias do notebook Jupyter no SageMaker console e no projeto relacionado: GitHub
  - `hello_world_workflow.ipynb`
  - `machine_learning_workflow_abalone.ipynb`
  - `training_pipeline_pytorch_mnist.ipynb`

Implantação de máquinas de estado usando o Terraform

O Terraform da HashiCorp é uma estrutura para criar aplicativos usando infraestrutura como código (IaC). Com o Terraform, você pode criar máquinas de estado e usar recursos, como visualizar implantações de infraestrutura e criar modelos reutilizáveis. Os modelos do Terraform ajudam você a manter e reutilizar o código dividindo-o em partes menores.

Se você estiver familiarizado com o Terraform, poderá seguir o ciclo de vida de desenvolvimento descrito neste tópico como um modelo para criar e implantar suas máquinas de estado no Terraform. Se não estiver familiarizado com o Terraform, recomendamos que primeiro conclua o workshop Introdução ao Terraform em AWS para se familiarizar com o Terraform.

> **Tip**
>
> Para implantar um exemplo de uma máquina de estado construída usando o Terraform em sua Conta da AWS, consulte o módulo Gerenciando máquinas de estado com infraestrutura como código do The AWS Step Functions Workshop.

Neste tópico

Pré-requisitos

Antes de começar, conclua os seguintes pré-requisitos:

- Instale o Terraform na máquina. Para obter informações sobre a instalação do Terraform, consulte Instalar o Terraform.

- Instale o Step Functions Local em sua máquina. Recomendamos que você instale a imagem do Docker do Step Functions Local para poder usar o Step Functions Local. Para obter mais informações, consulte Testando máquinas de estado localmente.
- Instalar a CLI do AWS SAM. Para informações sobre a instalação, consulte a Instalação da CLI do AWS SAM no Guia do desenvolvedor AWS Serverless Application Model.
- Instale o AWS Toolkit for Visual Studio Code para visualizar o diagrama do fluxo de trabalho de suas máquinas de estado. Para informações sobre instalação, consulte Instalação do AWS Toolkit for Visual Studio Code no Guia do usuário da AWS Toolkit for Visual Studio Code.

Ciclo de vida de desenvolvimento de máquinas de estado com o Terraform

O procedimento a seguir explica como você pode usar um protótipo de máquina de estado criado usando o Workflow Studio no console Step Functions como ponto de partida para o desenvolvimento local com o Terraform e o AWS Toolkit for Visual Studio Code.

Para ver o exemplo completo que discute o desenvolvimento da máquina de estado com o Terraform e apresenta as melhores práticas em detalhes, consulte Melhores práticas para criar projetos Terraform do Step Functions.

Para iniciar o ciclo de vida de desenvolvimento de uma máquina de estado com o Terraform

1. Faça bootstrap de um novo projeto do Terraform com o seguinte comando.

   ```
   terraform init
   ```

2. Abra o console Step Functions para criar um protótipo para sua máquina de estado.
3. No Workflow Studio, faça o seguinte:

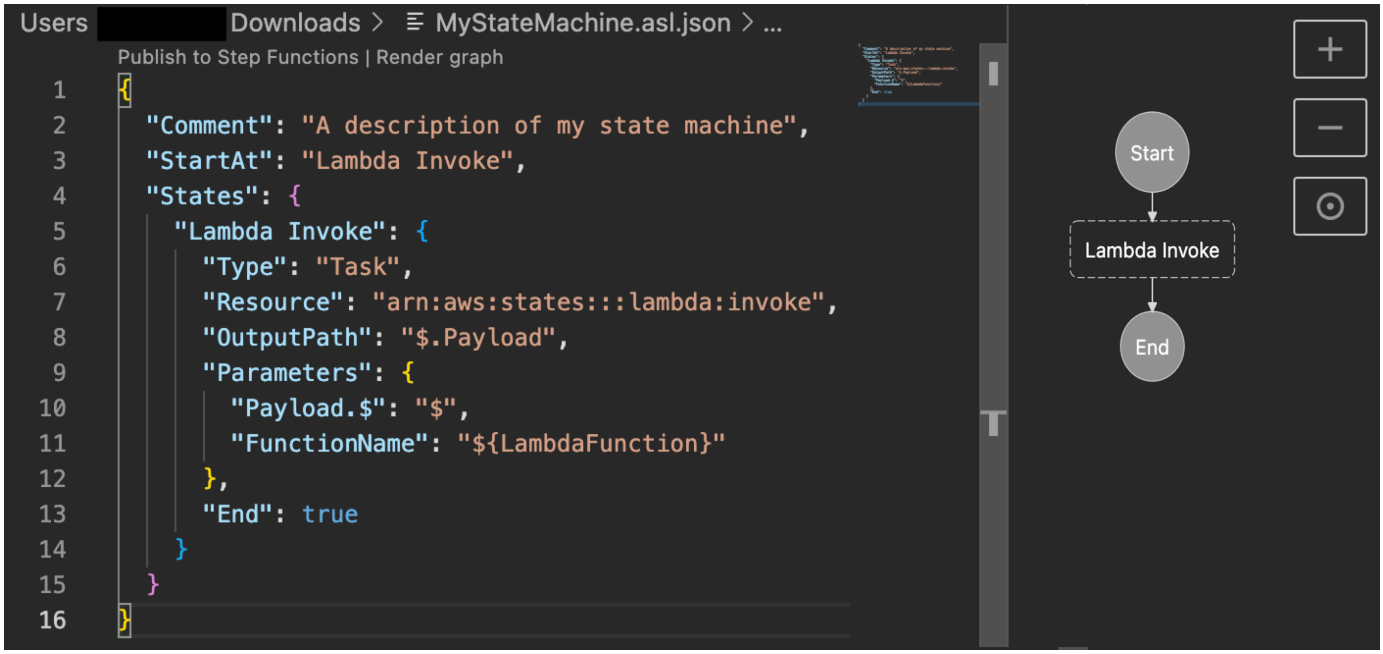
   a. Crie seu protótipo de fluxo de trabalho.

   b. Exporte a definição Amazon States Language (ASL) de seu fluxo de trabalho. Para fazer isso, escolha a lista suspensa Importar/Exportar e selecione Exportar definição JSON.
4. Salve a definição de ASL exportada no diretório do seu projeto.

   Você passa a definição de ASL exportada como um parâmetro de entrada para o recurso do `aws_sfn_state_machine` Terraform que usa a função `templatefile`. Essa função é usada dentro do campo de definição que passa a definição de ASL exportada e quaisquer substituições de variáveis.

> **Tip**
>
> Como o arquivo de definição ASL pode conter blocos de texto longos, recomendamos que você evite o método EOF em linha. Isso facilita a substituição de parâmetros na definição da máquina de estado.

5. (Opcional) Atualize a definição de ASL em seu IDE e visualize suas alterações usando o AWS Toolkit for Visual Studio Code.



Para evitar exportar continuamente sua definição e refatorá-la em seu projeto, recomendamos que você faça atualizações localmente em seu IDE e acompanhe essas atualizações com o Git.

6. Teste seu fluxo de trabalho usando o Step Functions Local.

> **Tip**
>
> Você também pode testar localmente as integrações de serviços com funções do Lambda e APIs do API Gateway em sua máquina de estado usando a CLI do AWS SAM Local.

7. Visualize sua máquina de estado e outros recursos AWS antes de implantá-la. Para fazer isso, execute o comando a seguir.

```
terraform plan
```

8. Implante sua máquina de estado a partir do seu ambiente local ou por meio de pipelines de CI/CD usando o comando a seguir.

```
terraform apply
```

9. (Opcional) Limpe seus recursos e exclua a máquina de estado usando o comando a seguir.

```
terraform destroy
```

Perfis e políticas do IAM para sua máquina de estado

Use as políticas de integração de serviços do Terraform para adicionar as permissões necessárias do IAM à sua máquina de estado, por exemplo, permissão para invocar funções do Lambda. Você também pode definir perfis e políticas explícitas e associá-las à sua máquina de estado.

O exemplo de política do IAM a seguir concede à sua máquina de estado acesso para invocar uma função do Lambda chamada *myFunction*.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "lambda:InvokeFunction"
      ],
      "Resource": "arn:aws:lambda:us-east-1:123456789012:function:myFunction"
    }
  ]
}
```

Também recomendamos usar a fonte de dados `aws_iam_policy_document` ao definir políticas do IAM para suas máquinas de estado no Terraform. Isso ajuda a verificar se sua política está malformada e a substituir quaisquer recursos por variáveis.

O exemplo de política do IAM a seguir usa a fonte de dados `aws_iam_policy_document` e concede à sua máquina de estado acesso para invocar uma função do Lambda chamada *`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 padrões arquitetônicos AWS mais avançados implantados com o Terraform, consulte exemplos do Terraform na Serverless Land Workflows Collection.

Testes e depuração

O Step Functions oferece maneiras diferentes de testar e depurar máquinas de estado. Por exemplo, é possível testar e depurar as máquinas de estado no console, usar a API TestState para testar um estado individual ou usar o Step Functions Local para testar máquinas de estado localmente.

Usando a API TestState você vai fornecer a definição de um único estado e executá-la. É possível testar um estado único sem criar uma máquina de estado nem atualizar uma existente.

O Step Functions Local é uma versão para download do Step Functions que permite desenvolver e testar aplicações usando uma versão do Step Functions em execução no próprio ambiente de desenvolvimento. Com o Step Functions Local, é possível executar as máquinas de estado para testar os fluxos de dados de entrada e de saída, integrações a serviços compatíveis e muito mais no ambiente de desenvolvimento local.

Tópicos

Usando a TestState API para testar um estado

A TestStateAPI aceita a definição de um único estado e a executa. É possível testar um estado sem criar uma máquina de estado nem atualizar uma existente.

Usando a TestState API, você pode testar o seguinte:

- O fluxo de dados de processamento de entrada e de saída de um estado.
- Uma AWS service (Serviço da AWS) integração com outras Serviços da AWS solicitações e respostas
- A solicitação e a resposta de uma tarefa HTTP.

Para testar um estado, também é possível usar o console do Step Functions, a AWS Command Line Interface (AWS CLI) ou o SDK.

A API `TestState` assume um perfil do IAM que deve conter as permissões do IAM necessárias para os recursos acessados pelo estado. Para obter informações sobre as permissões necessárias a um estado, consulte IAMpermissões para usar a TestState API.

Tópicos

Considerações sobre o uso da API TestState

Usando a TestStateAPI, você pode testar somente um estado por vez. Os estados que você pode testar incluem os seguintes:

- Todos os Tipos de tarefa, exceto Atividades
- Pass
- Aguardar
- Choice
- Succeed
- Fail

Ao usar a API `TestState`, tenha em mente as considerações a seguir.

- A TestState API não inclui suporte para o seguinte:
  - Estados Estado da tarefa que usem os seguintes tipos de recurso:
    - Atividades
    - Padrões de integração de serviço do tipo `.sync` ou `.waitForTaskToken`
  - Estado Paralelo
  - Estado Mapa
- Um teste pode ser executado por até cinco minutos. Se um teste exceder essa duração, ele falhará com o erro `States.Timeout`.

Usando níveis de inspeção na TestState API

Para testar um estado usando a TestStateAPI, você fornece a definição desse estado. O teste então exibe uma saída. Para cada estado, é possível especificar a quantidade de detalhes que você deseja visualizar nos resultados do teste. Esses detalhes fornecem informações adicionais sobre o estado que você está testando. Por exemplo, se você usou algum filtro de processamento de dados de entrada e de saída, como `InputPath` ou `ResultPath` em um estado, poderá visualizar os resultados intermediários e finais do processamento de dados.

O Step Functions oferece os seguintes níveis para especificar os detalhes a serem visualizados:

- INFO
- DEBUG
- TRACE

Todos esses níveis também exibem os campos `status` e `nextState`. O `status` indica o status da execução do estado. Por exemplo, `SUCCEEDED`, `FAILED`, `RETRIABLE` e `CAUGHT_ERROR`. `nextState` indica o nome do próximo estado para o qual fazer a transição. Se você não definiu o próximo estado na definição, esse campo exibirá um valor vazio.

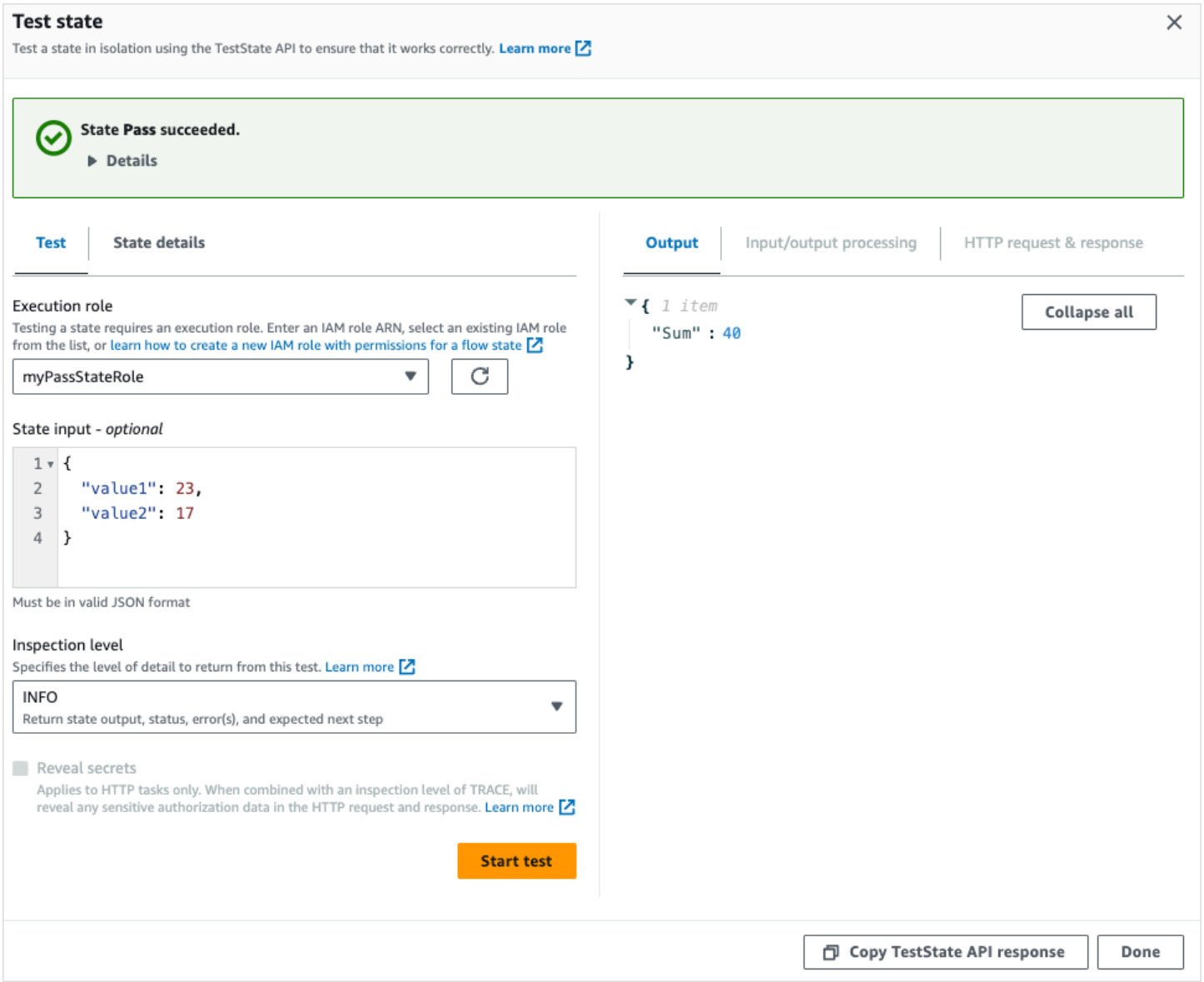
Para obter informações sobre como testar um estado usando esses níveis de inspeção no console do Step Functions e na AWS CLI, consulte Testar um estado (console) e Testar um estado usando a AWS CLI.

Nível de inspeção INFO

Se o teste for bem-sucedido, esse nível mostrará a saída do estado. Se o teste falhar, esse nível mostrará a saída do erro. Por padrão, o Step Functions define o Nível de inspeção como INFO se você não especificar um nível.

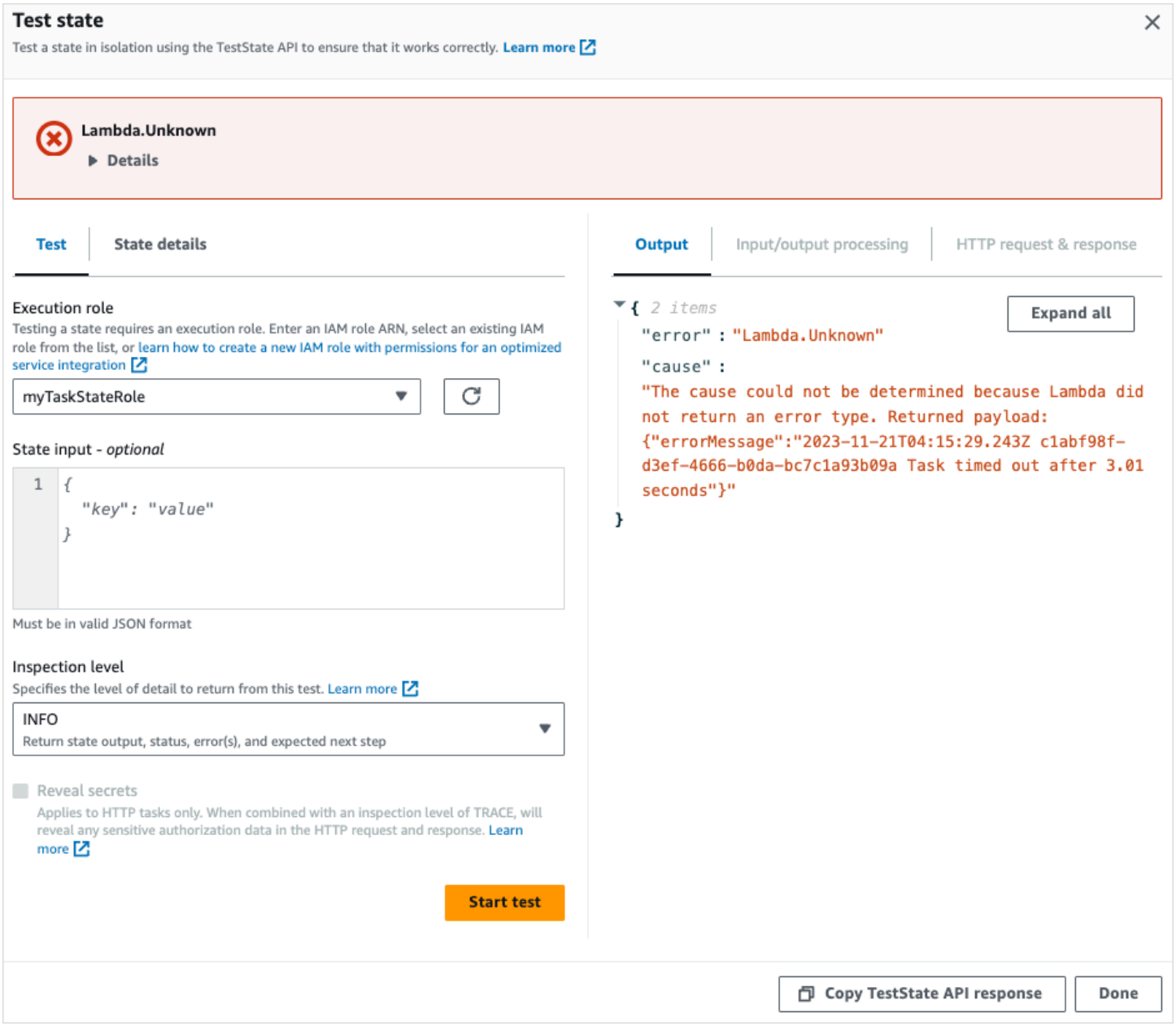
Exemplo de teste com o nível INFO bem-sucedido

A imagem a seguir mostra um teste bem-sucedido para um estado Aprovado. O Nível de inspeção desse estado é definido como INFO e a saída do estado é exibida na guia Saída.



Exemplo de teste com o nível INFO com falha

A imagem a seguir mostra um teste que falhou em um estado de Tarefa quando o Nível de inspeção está definido como INFO. A guia Saída mostra a saída do erro que inclui o nome e uma explicação detalhada da causa desse erro.



Nível de inspeção DEBUG

Se o teste for bem-sucedido, esse nível mostrará a saída do estado e o resultado do processamento dos dados de entrada e de saída.

Se o teste falhar, esse nível mostrará a saída do erro. Esse nível mostra os resultados intermediários do processamento de dados até o ponto de falha. Por exemplo, digamos que você testou um estado de Tarefa que invoque uma função do Lambda. Imagine que você tenha aplicado os filtros InputPath, Parâmetros, ResultPath e OutputPath ao estado de Tarefa. Digamos que a invocação tenha falhado.

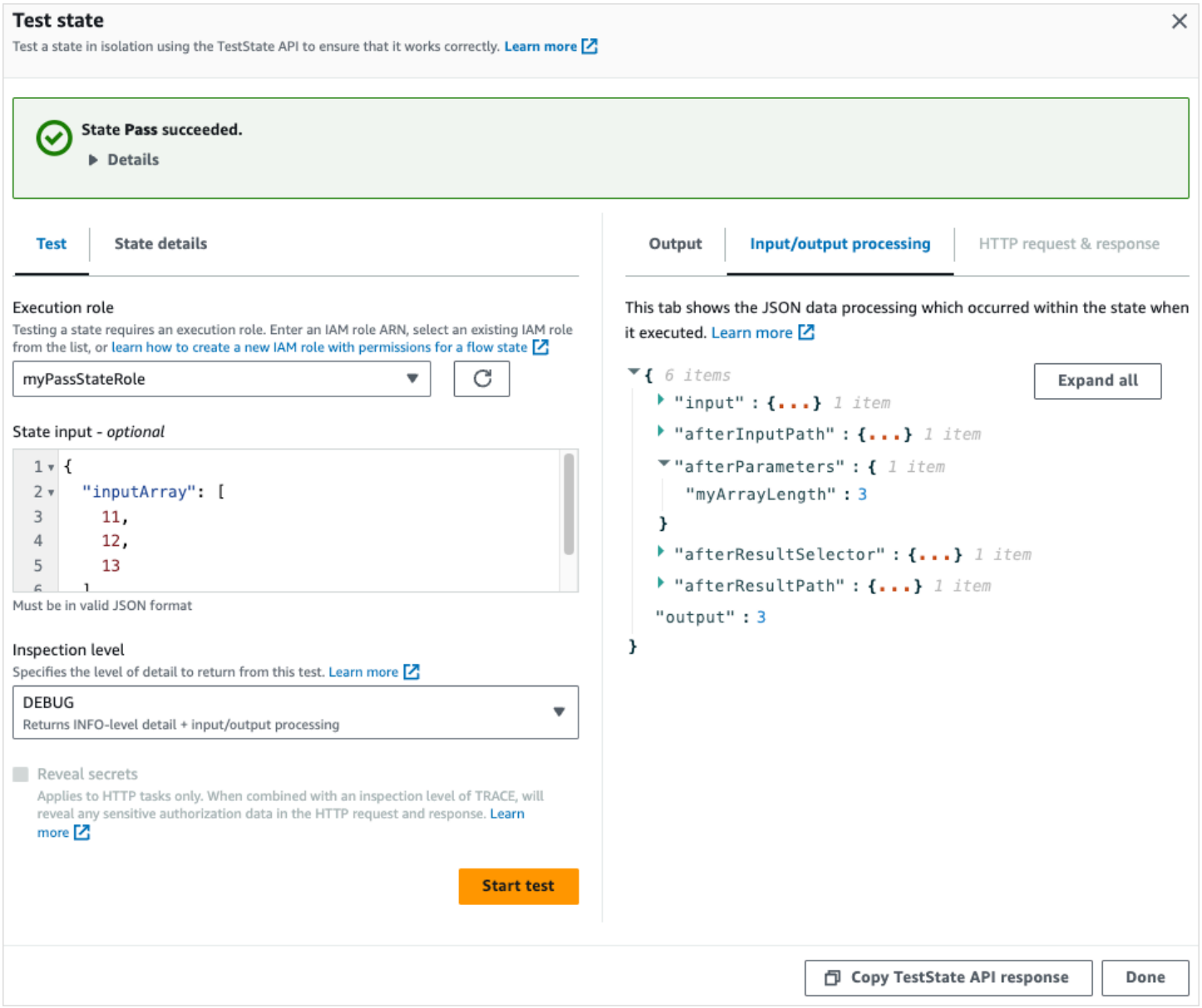
Nesse caso, o nível `DEBUG` mostra os resultados do processamento de dados com base na aplicação dos filtros na seguinte ordem:

- `input`: entrada de estado bruto.
- `afterInputPath`: entrada após o Step Functions aplicar o filtro `InputPath`.
- `afterParameters`: a entrada efetiva após o Step Functions aplicar o filtro `Parameters`.

As informações de diagnóstico disponíveis nesse nível podem ajudar a solucionar problemas relacionados a uma integração de serviços ou ao fluxo de processamento de dados de entrada e de saída definido.

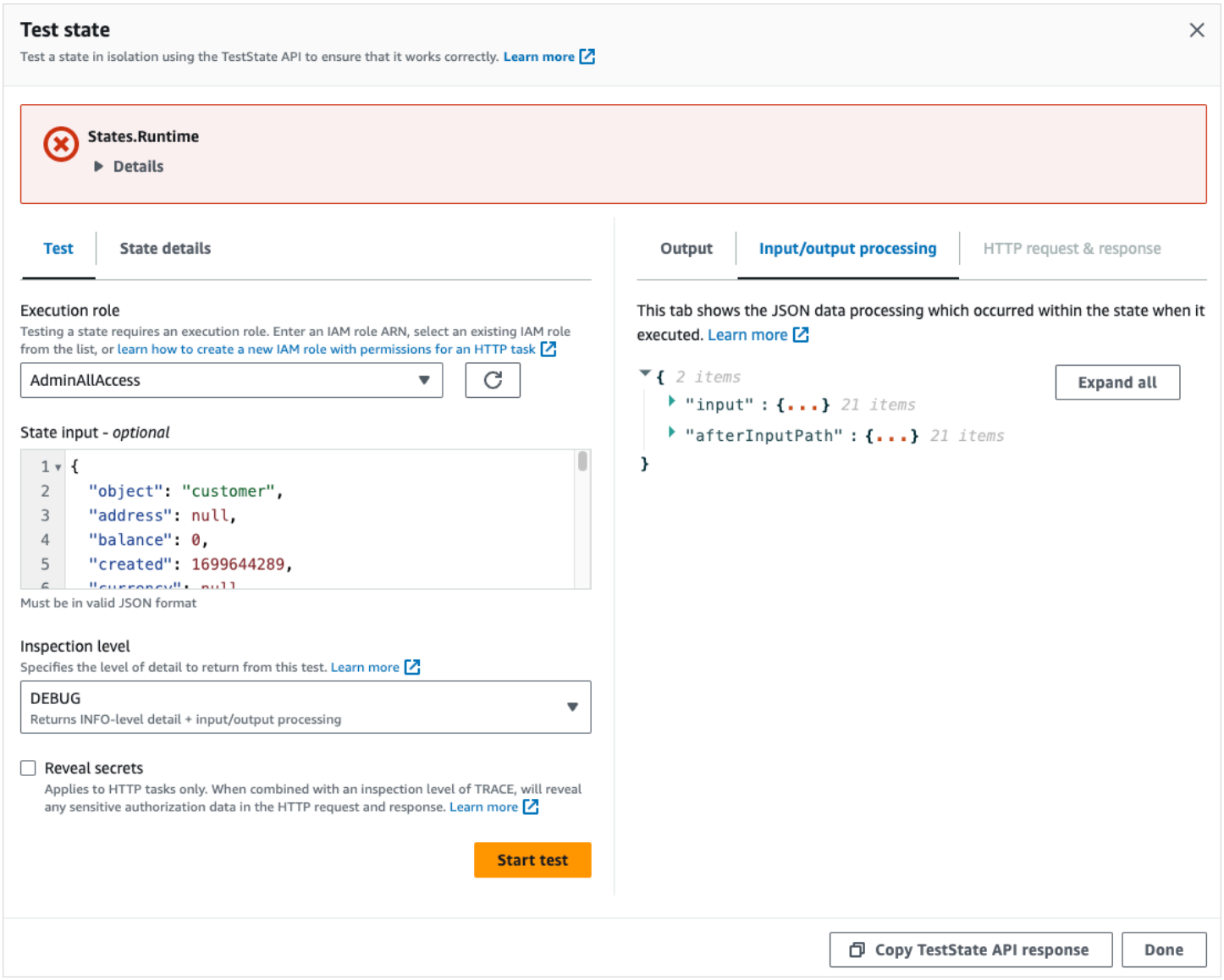
Exemplo de teste com nível DEBUG bem-sucedido

A imagem a seguir mostra um teste bem-sucedido para um estado Aprovado. O Nível de inspeção para esse estado é definido como DEBUG. A guia Processamento de entrada/saída na imagem a seguir mostra o resultado da aplicação de `Parameters` na entrada fornecida para esse estado.



Exemplo de teste com o nível DEBUG com falha

A imagem a seguir mostra um teste que falhou em um estado de Tarefa quando o Nível de inspeção está definido como DEBUG. A guia Processamento de entrada/saída na imagem a seguir mostra o resultado do processamento de dados de entrada e de saída para o estado até o ponto da falha.



Nível de inspeção TRACE

O Step Functions oferece o nível TRACE para testar uma tarefa HTTP. Esse nível exibe informações sobre a solicitação HTTP feita pelo Step Functions e a resposta que uma API de terceiros exibe. A resposta pode conter informações, como cabeçalhos e corpo da solicitação. Além disso, é possível visualizar a saída do estado e o resultado do processamento de dados de entrada e de saída nesse nível.

Se o teste falhar, esse nível mostrará a saída do erro.

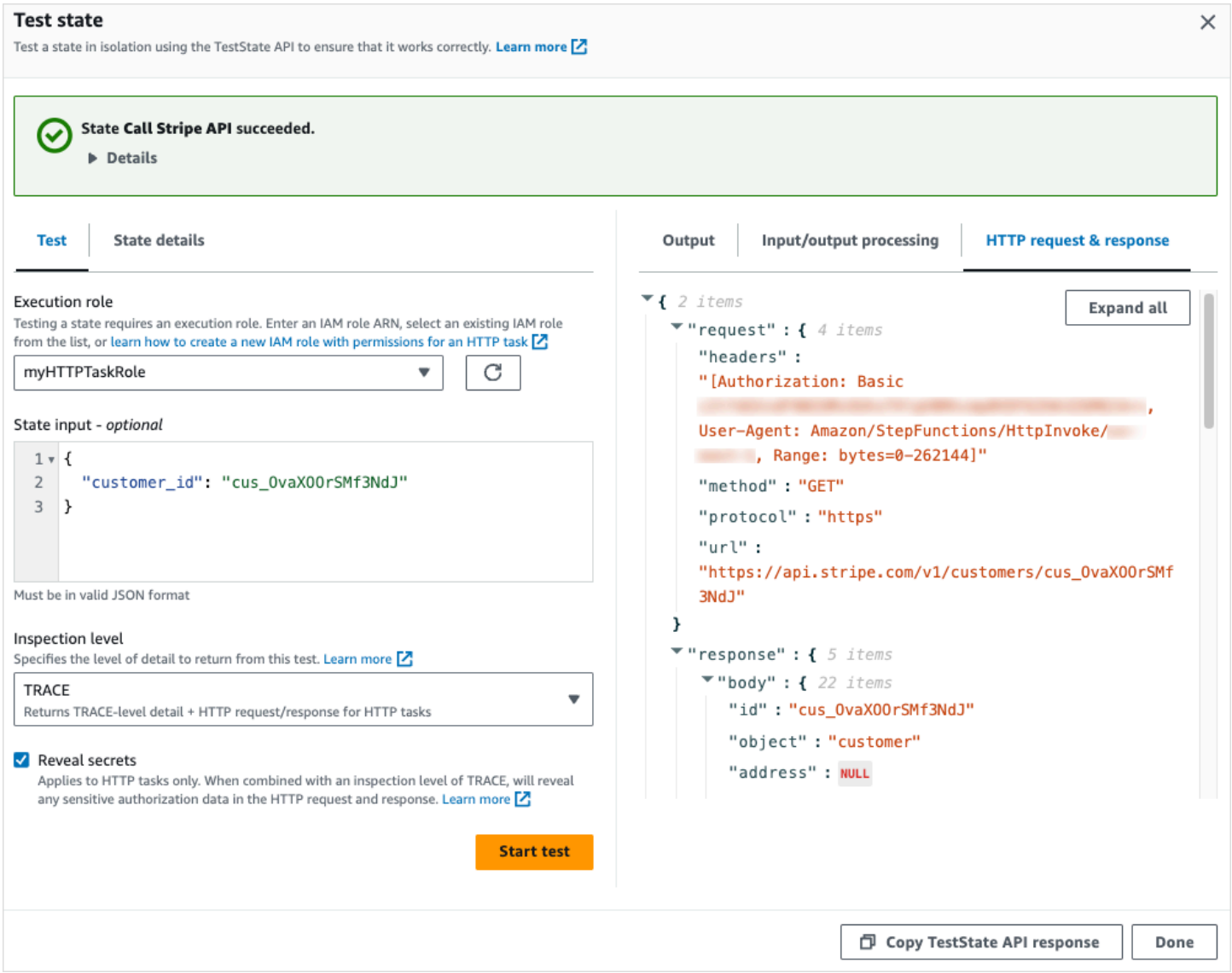
Esse nível é aplicável somente para tarefas HTTP. O Step Functions vai gerar um erro se você usar esse nível para outros tipos de estado.

Ao definir o nível de inspeção como TRACE, você também pode visualizar os segredos incluídos na EventBridge conexão. Para fazer isso, você deve definir o `revealSecrets` parâmetro como `true` na TestStateAPI. Além disso, você deve garantir que o IAM usuário que chama a TestState API tenha permissão para a `states:RevealSecrets` ação. Para ver um exemplo de política do IAM que concede a permissão `states:RevealSecrets`, consulte IAMpermissões para usar a TestState API. Sem essa permissão, o Step Functions gera um erro de acesso negado.

Se você definir o parâmetro `revealSecrets` como `false`, o Step Functions omitirá todos os segredos nos dados de solicitação e resposta HTTP.

Exemplo de teste com o nível TRACE bem-sucedido

A imagem a seguir mostra um teste bem-sucedido para uma tarefa HTTP. O Nível de inspeção para esse estado é definido como TRACE. A guia Solicitação e resposta HTTP na imagem a seguir mostra o resultado da chamada de API de terceiros.



IAMpermissões para usar a TestState API

O usuário do IAM que chama a API `TestState` deve ter permissões para realizar as ações `states:TestState` e `iam:PassRole`. Além disso, se você definir o parâmetro [revealSecrets] como `true`, deverá garantir que o usuário do IAM tenha permissões para realizar a ação `states:RevealSecrets`. Sem essa permissão, o Step Functions gera um erro de acesso negado.

Também é necessário garantir que o perfil de execução contenha as permissões do IAM necessárias para os recursos que o estado está acessando. Para obter informações sobre as permissões necessárias a um estado, consulte Managing execution roles.

O exemplo de política do IAM a seguir define as permissões `states:TestState`, `iam:PassRole` e `states:RevealSecrets`.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:TestState",
        "states:RevealSecrets",
        "iam:PassRole"
      ],
      "Resource": "*"
    }
  ]
}
```

Testar um estado (console)

É possível testar um estado no console e conferir a saída do estado ou o fluxo de processamento de dados de entrada e de saída. Para uma tarefa HTTP, é possível testar a solicitação e a resposta HTTP brutas.

Como testar um estado

1. Abra o console do Step Functions.
2. Selecione Criar uma máquina de estado para começar a criar uma máquina de estado ou escolha uma máquina de estado existente.
3. No Modo de design do Workflow Studio, selecione um estado que deseja testar.
4. Selecione Testar estado no painel Inspector do Workflow Studio.
5. Na caixa de diálogo Testar estado, faça o seguinte:

    a. Em Perfil de execução, selecione um perfil de execução para testar o estado. Tenha as permissões do IAM necessárias para o estado que deseja testar.

    b. (Opcional) Forneça todas as entradas JSON de que o estado selecionado precise para o teste.

    c. Em Nível de inspeção, selecione uma das seguintes opções com base nos valores que você deseja visualizar:

    - INFO: mostrará a saída do estado na guia Saída se o teste for bem-sucedido. Se o teste falhar, INFO mostrará a saída do erro que inclui o nome e uma explicação detalhada da

causa desse erro. Por padrão, o Step Functions definirá o Nível de inspeção como INFO se você não especificar um nível.

- DEBUG: mostrará a saída do estado e o resultado do processamento dos dados de entrada e de saída se o teste for bem-sucedido. Se o teste falhar, DEBUG mostrará a saída do erro que inclui o nome e uma explicação detalhada da causa desse erro.
- TRACE: mostra a solicitação e a resposta HTTP brutas e é útil para verificar cabeçalhos, parâmetros de consulta e outros detalhes específicos da API. Essa opção só está disponível para a tarefa HTTP.

  Também é possível selecionar Revelar segredos. Em combinação com TRACE, essa configuração permite que você veja os dados confidenciais que a conexão com o EventBridge insere, como chaves de API. A identidade do usuário do IAM que você usa para acessar o console deve ter permissão para realizar a ação `states:RevealSecrets`. Sem essa permissão, o Step Functions gera um erro de acesso negado ao iniciar o teste. Para ver um exemplo de política do IAM que concede essas permissões `states:RevealSecrets`, consulte IAMpermissões para usar a TestState API.

d. Selecione Iniciar teste.

Testar um estado usando a AWS CLI

Você pode testar um estado compatível usando a TestStateAPI noAWS CLI. A API aceita a definição de um estado e a executa.

Para cada estado, é possível especificar a quantidade de detalhes que você deseja visualizar nos resultados do teste. Esses detalhes fornecem informações adicionais sobre a execução do estado, incluindo o resultado do processamento de dados de entrada e de saída e as informações de solicitação e resposta HTTP. Os exemplos a seguir mostram os diferentes níveis de inspeção que você pode especificar para a TestState API. Lembre-se de substituir o texto em *`itálico`* pelas informações específicas do recurso.

Esta seção contém os seguintes exemplos que descrevem como usar os diferentes níveis de inspeção fornecidos pelo Step Functions na AWS CLI:

- Usar o nível de inspeção INFO
- Usar o nível de inspeção DEBUG
- Usar o nível de inspeção TRACE

- Usando o utilitário jq in AWS CLI para filtrar e imprimir a resposta HTTP que a TestState API retorna

Exemplo 1: Usar o nível de inspeção INFO para testar um estado de escolha

Para testar um estado usando o `INFO` InspectionLevel no AWS CLI, execute o `test-state` comando conforme mostrado no exemplo a seguir.

```
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}'
```

Este exemplo usa um estado de escolha para determinar o caminho de execução do estado com base na entrada numérica fornecida. Por padrão, o Step Functions definirá o `inspectionLevel` como `INFO` se você não definir um nível.

O Step Functions exibe a saída a seguir.

```
{
    "output": "{\"number\": 2}",
    "nextState": "Equals 2",
    "status": "SUCCEEDED"
}
```

Exemplo 2: Usar o Nível de inspeção DEBUG para depurar o processamento de dados de entrada e de saída em um estado Aprovado

Para testar um estado usando o `DEBUG` InspectionLevel no AWS CLI, execute o `test-state` comando conforme mostrado no exemplo a seguir.

```
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
```

Este exemplo usa um estado Pass para mostrar como o Step Functions filtra e manipula os dados JSON de entrada usando os filtros de processamento de dados de entrada e de saída. Este exemplo usa estes filtros: `InputPath`, `Parâmetros`, `ResultPath` e `OutputPath`.

O Step Functions exibe a saída a seguir.

```
{
    "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"
}
```

Exemplo 3: Usar o Nível de inspeção TRACE e RevealSecrets para inspecionar a solicitação HTTP enviada a uma API de terceiros

Para testar uma tarefa HTTP usando o TRACE InspectionLevel junto com o parâmetro revealSecrets no AWS CLI, execute o `test-state` comando conforme mostrado no exemplo a seguir.

```
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
```

Este exemplo testa se a tarefa HTTP chama a API de terceiros especificada, `https://httpbin.org/`. Ele também mostra os dados de solicitação e resposta HTTP para a chamada de 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"
```

```
}
```

Exemplo 4: Usando o utilitário jq para filtrar e imprimir a resposta que a TestState API retorna

A TestState API retorna dados JSON como sequências de caracteres de escape em sua resposta. O AWS CLI exemplo a seguir estende o Exemplo 3 e usa o `jq` utilitário para filtrar e imprimir a resposta HTTP que a TestState API retorna em um formato legível por humanos. Para obter informações `jq` e instruções de instalação, 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'
```

O exemplo a seguir mostra a saída exibida em formato legível.

```
{
  "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"
}
```

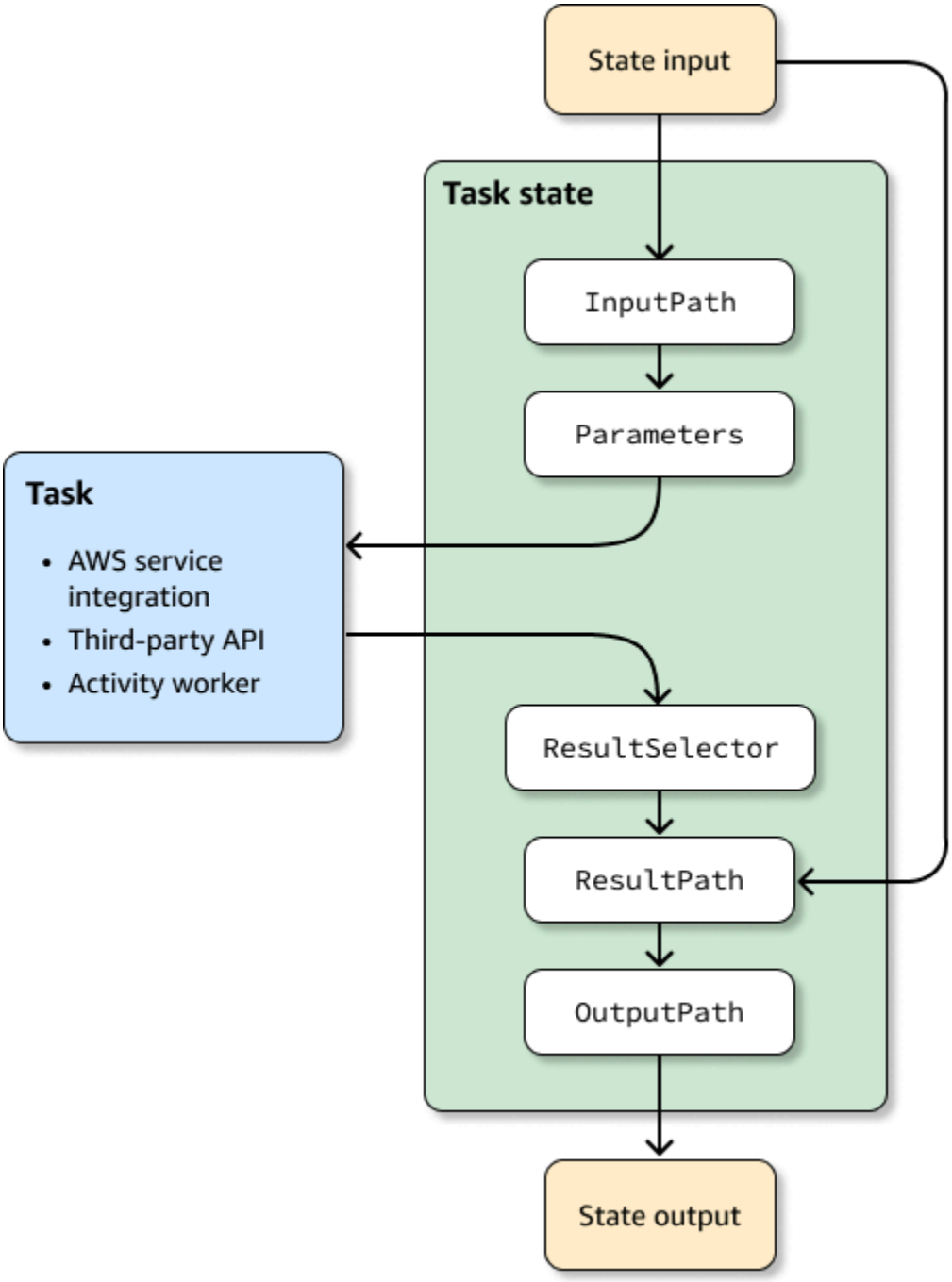
Testar e depurar o fluxo de dados de entrada e de saída.

A API `TestState` é útil para testar e depurar os dados que fluem pelo fluxo de trabalho. Esta seção fornece alguns conceitos-chave e explica como usar o TestState para essa finalidade.

Principais conceitos

No Step Functions, o processo de filtrar e manipular dados JSON à medida que eles passam pelos estados na máquina de estado é chamado de processamento de entrada e de saída. Para obter informações sobre como isso funciona, consulte Processamento de entrada e saída no Step Functions.

Todos os tipos de estado no Amazon States Language (ASL) (Tarefa, Paralelo, Mapa, Aprovar, Aguardar, Escolha, Êxito e Falha) compartilham um conjunto de campos comuns para filtrar e manipular os dados JSON que passam por eles. Esses campos são: InputPath, Parâmetros, ResultSelector, ResultPath e OutputPath. O suporte para cada campo varia de acordo com os estados. Em runtime, o Step Functions aplica cada campo em uma ordem específica. O diagrama a seguir mostra a ordem na qual esses campos são aplicados aos dados em um estado de Tarefa:



A lista a seguir descreve a ordem de aplicação dos campos de processamento de entrada e de saída mostrados no diagrama.

1. A entrada de estado são os dados JSON transmitidos para o estado atual a partir de um estado anterior.
2. O InputPath filtra uma parte da entrada de estado bruto.
3. O Parâmetros configura o conjunto de valores a serem passados para a Tarefa.

4. A tarefa executa o trabalho e exibe um resultado.
5. O ResultSelector seleciona um conjunto de valores a ser excluído do resultado da tarefa.
6. O ResultPath combina o resultado com a entrada de estado bruto ou substitui o resultado por ela.
7. O OutputPath filtra uma parte da saída para passar para o próximo estado.
8. A entrada de estado são os dados JSON transmitidos do estado atual para o próximo estado.

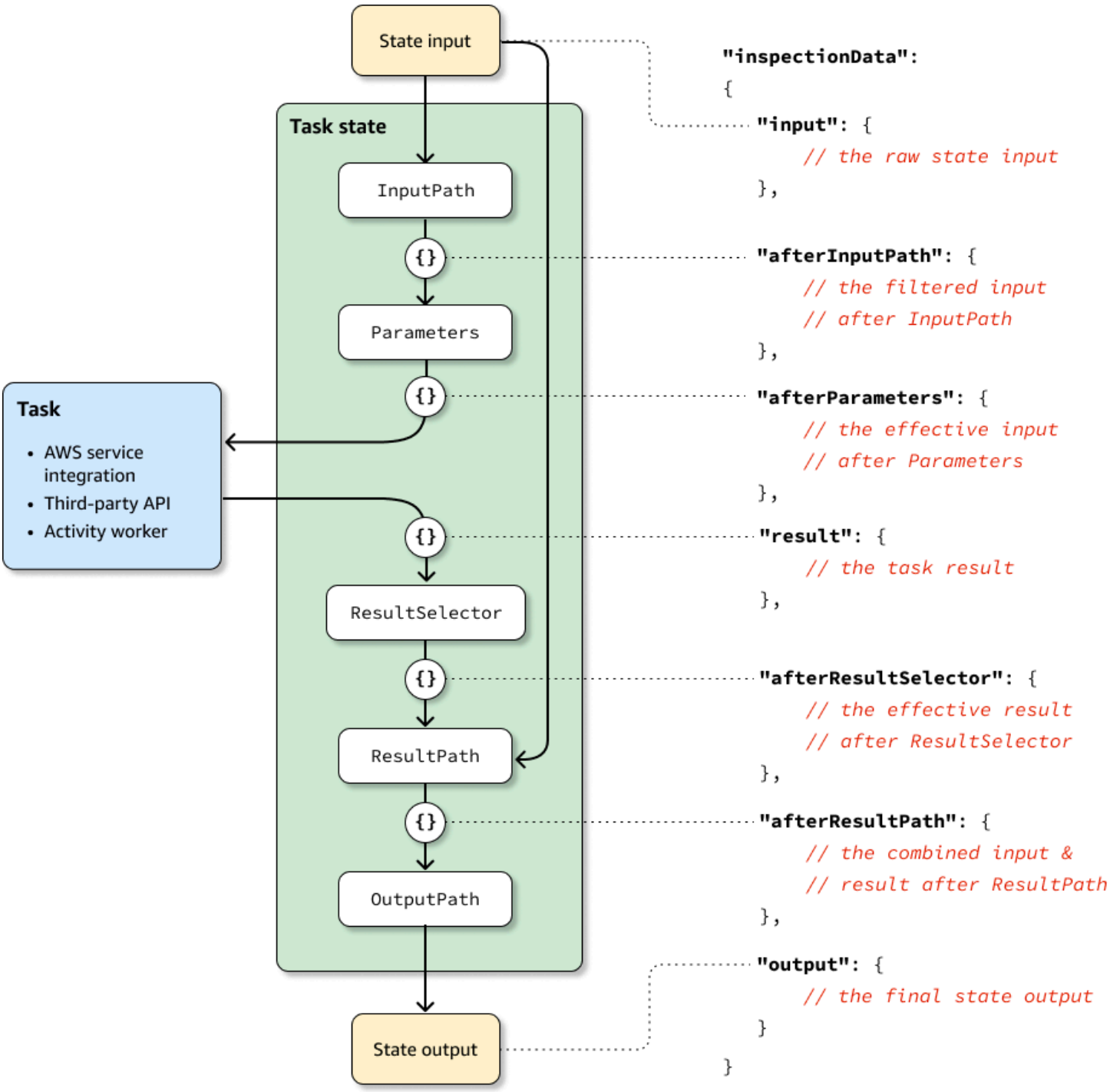
Esses campos de processamento de entrada e de saída são opcionais. Se você não usar nenhum desses campos na definição do estado, a tarefa consumirá a entrada do estado bruto e exibirá o resultado da tarefa como a saída do estado.

Usando TestState para inspecionar o processamento de entrada e saída

Ao chamar a API `TestState` e definir o parâmetro `inspectionLevel` como `DEBUG`, a resposta da API incluirá um objeto chamado `inspectionData`. Esse objeto contém campos para ajudar a inspecionar como os dados foram filtrados ou manipulados no estado durante a execução. O exemplo a seguir mostra o objeto `inspectionData` para o estado de Tarefa.

```
"inspectionData":   {
  "input": string,
  "afterInputPath": string,
  "afterParameters": string,
  "result": string,
  "afterResultSelector": string,
  "afterResultPath": string,
  "output": string
}
```

Neste exemplo, cada campo que contém o prefixo `after` mostra os dados após a aplicação de um campo específico. Por exemplo, `afterInputPath` mostra o efeito da aplicação do campo `InputPath` para filtrar a entrada de estado bruto. O diagrama a seguir associa cada campo Definição de ASL ao campo correspondente no objeto `inspectionData`:



Para exemplos de uso da TestState API para depurar o processamento de entrada e saída, consulte o seguinte:

- Testar um estado usando o nível de inspeção DEBUG no console do Step Functions
- Testando um estado usando o nível de inspeção DEBUG no AWS CLI

Testando máquinas de estado localmente

O Step Functions Local AWS é uma versão para download do Step Functions que permite desenvolver e testar aplicativos usando uma versão do Step Functions em execução em seu próprio ambiente de desenvolvimento. A versão local do Step Functions pode invocar funções do AWS Lambda na AWS e em execução localmente. Também é possível coordenar outros serviços compatíveis da AWS.

> **Note**
>
> O Step Functions Local usa contas fictícias para funcionar.

Ao executar o Step Functions Local, você pode usar uma das seguintes formas de invocar integrações de serviços:

- Configurando endpoints locais para AWS Lambda e outros serviços. Para obter mais informações sobre os endpoints com suporte, consulte Definindo opções de configuração para Step Functions Local.
- Fazer chamadas diretamente para um serviço AWS do Step Functions Local.
- Simulando a resposta das integrações de serviços. Para obter mais informações sobre as integrações de serviços simuladas, consulte Usando integrações de serviços simulados.

O Step Functions Local AWS está disponível como um pacote JAR ou uma imagem do Docker independente que é executada no Microsoft Windows, Linux, macOS e outras plataformas compatíveis com Java ou Docker.

> **Warning**
>
> A versão disponível para download do AWS Step Functions destina-se a ser usada somente para testes e não deve ser usada para processar informações confidenciais.

> **Tip**
>
> Certifique-se de usar o Step Functions Local versão 1.12.0 ou superior para poder incluir todas as funções intrínsecas em seus fluxos de trabalho.

Os tópicos a seguir descrevem como você pode configurar o Step Functions Local usando o Docker e o arquivo JAR e executar o Step Functions Local para trabalhar com AWS Lambda, AWS Serverless Application Model(AWS SAM) CLI Local ou outros serviços compatíveis.

Tópicos

Configuração do Step Functions local (versão para download) e Docker

A imagem do Docker do Step Functions Local permite que você comece a usar rapidamente o Step Functions Local usando uma imagem do Docker com todas as dependências necessárias. A imagem do Docker permite incluir o Step Functions Local em compilações em contêineres e como parte dos testes de integração contínua.

Para obter a imagem do Docker para o Step Functions Local, acesse https://hub.docker.com/r/amazon/aws-stepfunctions-local, ou digite o comando `pull` do Docker.

```
docker pull amazon/aws-stepfunctions-local
```

Para iniciar a versão para download do Step Functions no Docker, execute o seguinte comando `run` do Docker

```
docker run -p 8083:8083 amazon/aws-stepfunctions-local
```

Para interagir com o AWS Lambda ou outros serviços compatíveis, é necessário configurar suas credenciais e outras opções de configuração primeiro. Para obter informações, consulte os tópicos a seguir:

- Definindo opções de configuração para Step Functions Local
- Credenciais e configuração para o Docker

Configuração do Step Functions Local (versão para download) - Versão Java

A versão disponível para download do AWS Step Functions é fornecida como um arquivo JAR executável e como uma imagem do Docker. O aplicativo Java é executado no Windows, Linux, macOS e outras plataformas compatíveis com Java. Além do Java, será necessário instalar a AWS Command Line Interface (AWS CLI). Para obter mais informações sobre a instalação e a configuração da AWS CLI, consulte o Guia do usuário da AWS Command Line Interface.

Como configurar e executar o Step Functions no computador

1. Faça download do Step Functions usando os seguintes links.

Links para fazer download	Soma de verificação
.tar.gz	.tar.gz.md5
.zip	.zip.md5

2. Extraia o arquivo `.zip`.
3. Teste o download e visualize as informações da versão.

```
$ java -jar StepFunctionsLocal.jar -v
Step Function Local
Version: 1.0.0
Build: 2019-01-21
```

4. (Opcional) Visualize uma lista de comandos disponíveis.

```
$ java -jar StepFunctionsLocal.jar -h
```

5. Para iniciar o Step Functions em seu computador, abra uma janela de prompt de comando, vá até o diretório onde você extraiu o `StepFunctionsLocal.jar` e insira o seguinte comando.

```
java -jar StepFunctionsLocal.jar
```

6. Para acessar o Step Functions em execução localmente, use o parâmetro `--endpoint-url`. Por exemplo, ao usar a AWS CLI, você especificaria os comandos do Step Functions assim:

```
aws stepfunctions --endpoint-url http://localhost:8083 command
```

> **Note**
>
> Por padrão, o Step Functions Local usa uma conta de teste e credenciais locais e a Região da AWS é definida como Leste dos EUA (Norte da Virgínia). Para usar o Step Functions Local com o AWS Lambda ou outros serviços compatíveis, é necessário configurar suas credenciais e sua região.
>
> Se você usar fluxos de trabalho expressos com o Step Functions Local, o histórico de execuções será armazenado em um arquivo de log. Ele não está logado no CloudWatch Logs. O caminho do arquivo de log será baseado no ARN do grupo de logs do CloudWatch Logs fornecido ao criar a máquina de estado local. O arquivo de log será armazenado no `/aws/states/log-group-name/${execution_arn}.log` relativo ao local em que você estiver executando o Step Functions Local. Por exemplo, se o ARN de execução for:
>
> ```
> arn:aws:states:us-east-1:123456789012:express:test:example-ExpressLogGroup-wJalrXUtnFEMI
> ```
>
> o arquivo de log será:
>
> ```
> aws/states/log-group-name/arn:aws:states:us-east-1:123456789012:express:test:example-ExpressLogGroup-wJalrXUtnFEMI.log
> ```

Definindo opções de configuração para Step Functions Local

Ao iniciar o AWS Step Functions Local usando o arquivo JAR, é possível definir as opções de configuração usando a AWS Command Line Interface (AWS CLI) ou incluindo-as no ambiente do sistema. Para o Docker, você deve especificar essas opções em um arquivo ao qual faz referência ao iniciar o Step Functions Local.

Opções de configuração

Quando você configura o contêiner do Step Functions Local para usar um endpoint de substituição, como Lambda Endpoint e Batch Endpoint, e faz chamadas para esse endpoint, o Step Functions Local não usa as credenciais que você especifica. Definir essas substituições de endpoint é opcional.

Opção	Linha de comando	Ambiente
Conta	-account, --aws-account	AWS_ACCOUNT_ID
Região	-region, --aws-region	AWS_DEFAULT_REGION
Aguardar escala de tempo	-waitTimeScale, --wait-time-scale	WAIT_TIME_SCALE
Endpoint do Lambda	-lambdaEndpoint, --lambda-endpoint	LAMBDA_ENDPOINT
Endpoint do Batch	-batchEndpoint, --batch-endpoint	BATCH_ENDPOINT
Endpoint do DynamoDB	-dynamoDBEndpoint, --dynamodb-endpoint	DYNAMODB_ENDPOINT
Endpoint do ECS	-ecsEndpoint, --ecs-endpoint	ECS_ENDPOINT
Endpoint do Glue	-glueEndpoint, --glue-endpoint	GLUE_ENDPOINT
Endpoint do SageMaker	-sageMakerEndpoint, --sagemaker-endpoint	SAGE_MAKER_ENDPOINT
Endpoint do SQS	-sqsEndpoint, --sqs-endpoint	SQS_ENDPOINT
Endpoint do SNS	-snsEndpoint, --sns-endpoint	SNS_ENDPOINT
Endpoint do Step Functions	-stepFunctionsEndpoint, --step-functions-endpoint	STEP_FUNCTIONS_ENDPOINT

Credenciais e configuração para o Docker

Para configurar o Step Functions Local para o Docker, crie o seguinte arquivo: `aws-stepfunctions-local-credentials.txt`.

Esse arquivo contém suas credenciais e outras opções de configuração. O seguinte pode ser usado como modelo ao criar o arquivo `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
```

Depois de configurar suas credenciais e opções de configuração em `aws-stepfunctions-local-credentials.txt`, inicie o Step Functions com o seguinte comando.

```
docker run -p 8083:8083 --env-file aws-stepfunctions-local-credentials.txt amazon/aws-stepfunctions-local
```

> **Note**
>
> É recomendável usar o nome DNS especial `host.docker.internal`, que é resolvido para o endereço IP interno que o host usa, como `http://host.docker.internal:8000`. Para obter mais informações, consulte a documentação do Docker para Mac e Windows em Recursos de rede no Docker Desktop para Mac e Recursos de rede no Docker Desktop para Windows, respectivamente.

Executando o Step Functions Local em seu computador

Use a versão local do Step Functions para configurar, desenvolver e testar máquinas de estado em seu computador.

Executar uma máquina de estado HelloWorld localmente

Depois de executar o Step Functions localmente com a AWS Command Line Interface (AWS CLI), é possível iniciar uma execução da máquina de estado.

1. Crie uma máquina de estado a partir da AWS CLI fazendo o escape da definição de máquina de estado.

```
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**
>
> O `role-arn` não é usado para o Step Functions Local, mas é necessário incluí-lo com a sintaxe adequada. É possível usar o nome de recurso da Amazon (ARN) do exemplo anterior.

Se você criou a máquina de estado com êxito, o Step Functions responde com a data de criação e o ARN da máquina de estado.

```
{
    "creationDate": 1548454198.202,
    "stateMachineArn": "arn:aws:states:us-
east-1:123456789012:stateMachine:HelloWorld"
}
```

2. Inicie uma execução usando o ARN da máquina de estado que você criou.

```
aws stepfunctions --endpoint-url http://localhost:8083 start-execution --state-
machine-arn arn:aws:states:us-east-1:123456789012:stateMachine:HelloWorld
```

Step Functions Local com AWS SAM CLI Local

É possível usar a versão local do Step Functions com uma versão local do AWS Lambda. Para configurar isso, é necessário instalar e configurar o AWS SAM.

Para obter informações sobre como configurar e executar o AWS SAM, consulte o seguinte:

- Configurar o AWS SAM
- Iniciar o AWS SAM CLI Local

Assim que o Lambda estiver em execução no sistema local, será possível iniciar o Step Functions Local. No diretório em que você extraiu os arquivos JAR locais do Step Functions, inicie o Step Functions Local e use o parâmetro `--lambda-endpoint` para configurar o endpoint Lambda local.

```
java -jar StepFunctionsLocal.jar --lambda-endpoint http://127.0.0.1:3001 command
```

Para obter mais informações sobre como executar o Step Functions Local com o AWS Lambda, consulte Testando o Step Functions e AWS SAM CLI Local.

Testando o Step Functions e AWS SAM CLI Local

Com o AWS Step Functions e o AWS Lambda em execução em sua máquina local, é possível testar sua máquina de estado e as funções do Lambda sem implantar o código na AWS.

Para obter mais informações, consulte os tópicos a seguir:

- Testando máquinas de estado localmente
- Configurar o AWS SAM

Tópicos

- Etapa 1: configurar o AWS SAM
- Etapa 2: Testar o AWS SAM CLI Local
- Etapa 3: Iniciar o AWS SAM CLI Local
- Etapa 4: Iniciar o Step Functions Local
- Etapa 5: Criar uma máquina de estado que faz referência à sua função AWS SAM CLI Local

- Etapa 6: Iniciar uma execução da máquina de estado local

Etapa 1: configurar o AWS SAM

A CLI Local do AWS Serverless Application Model (AWS SAM) exige que a AWS Command Line Interface, o AWS SAM e o Docker sejam instalados.

1. Instalar a CLI do AWS SAM.

   > **Note**
   >
   > Antes de instalar a CLI do AWS SAM, é necessário instalar a AWS CLI e o Docker. Consulte os Pré-requisitos para instalar a CLI do AWS SAM.

2. Consulte a documentação do Início rápido do AWS SAM. Siga as etapas para fazer o seguinte:

   1. Inicializar o aplicativo
   2. Testar o aplicativo localmente

   Isso cria um diretório `sam-app` e cria um ambiente que inclui uma função Olá, mundo do Lambda baseada em Python.

Etapa 2: Testar o AWS SAM CLI Local

Agora que você instalou o AWS SAM e criou a função Olá, mundo do Lambda, pode testá-la. No diretório `sam-app`, insira o comando a seguir:

```
sam local start-api
```

Isso executa uma instância local da sua função do Lambda. Você deve ver saída semelhante a:

```
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 um navegador e insira o seguinte:

```
http://127.0.0.1:3000/hello
```

Isso exibirá uma resposta semelhante a:

```
{"message": "hello world", "location": "72.21.198.66"}
```

Insira CTRL+C para finalizar a API do Lambda.

Etapa 3: Iniciar o AWS SAM CLI Local

Agora que você testou se a função funciona, inicie o AWS SAM CLI Local. No diretório `sam-app`, insira o comando a seguir:

```
sam local start-lambda
```

Isso inicia o AWS SAM CLI Local e fornece o endpoint a ser usado, semelhante à saída a seguir:

```
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)
```

Etapa 4: Iniciar o Step Functions Local

Arquivo JAR

Se estiver usando a versão `.jar` do arquivo do Step Functions Local, inicie o Step Functions e especifique o endpoint do Lambda. No diretório onde você extraiu os arquivos `.jar`, insira o seguinte comando:

```
java -jar StepFunctionsLocal.jar --lambda-endpoint http://localhost:3001
```

Quando o Step Functions Local for iniciado, ele verificará o ambiente e as credenciais configuradas no seu arquivo `~/.aws/credentials`. Por padrão, ele é iniciado usando um ID de usuário fictício e é listado 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

Se estiver usando a versão do Docker do Step Functions Local, execute-o com o comando a seguir:

```
docker run -p 8083:8083 amazon/aws-stepfunctions-local
```

Para obter informações sobre a instalação da versão do Docker do Step Functions, consulte Configuração do Step Functions local (versão para download) e Docker.

> **Note**
>
> Você poderá especificar o endpoint por meio da linha de comando ou definindo variáveis do ambiente se executar o Step Functions a partir do arquivo `.jar`. Para a versão do Docker, você deve especificar os endpoints e as credenciais em um arquivo de texto. Consulte Definindo opções de configuração para Step Functions Local.

Etapa 5: Criar uma máquina de estado que faz referência à sua função AWS SAM CLI Local

Assim que o Step Functions Local estiver em execução, crie uma máquina de estado que faz referência à `HelloWorldFunction` inicializada no Etapa 1: configurar o 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"
```

Isso criará uma máquina de estado e fornecerá um nome do recurso da Amazon (ARN) que pode ser usado para iniciar uma execução.

```
{
    "creationDate": 1548805711.403,
    "stateMachineArn": "arn:aws:states:us-east-1:123456789012:stateMachine:HelloWorld"
}
```

Etapa 6: Iniciar uma execução da máquina de estado local

Depois de criar uma máquina de estado, inicie uma execução. Você precisará referenciar o ARN do endpoint e da máquina de estado ao usar o seguinte comando **`aws stepfunctions`**:

```
aws stepfunctions --endpoint http://localhost:8083 start-execution --state-machine
 arn:aws:states:us-east-1:123456789012:stateMachine:HelloWorld --name test
```

Isso inicia uma execução chamada `test` em sua máquina de estado `HelloWorld`.

```
{
    "startDate": 1548810641.52,
    "executionArn": "arn:aws:states:us-east-1:123456789012:execution:HelloWorld:test"
}
```

Agora que o Step Functions está sendo executado localmente, você pode interagir com ele usando a AWS CLI. Por exemplo, para obter informações sobre essa execução, use o seguinte comando:

```
aws stepfunctions --endpoint http://localhost:8083 describe-execution --execution-arn
 arn:aws:states:us-east-1:123456789012:execution:HelloWorld:test
```

Chamar `describe-execution` para uma execução fornece detalhes mais completos, semelhante ao resultado abaixo:

```
{
```

```
    "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": "{}"
}
```

Usando integrações de serviços simulados

No Step Functions Local, você pode testar os caminhos de execução de suas máquinas de estado sem realmente chamar serviços integrados usando integrações de serviços simuladas. Para configurar suas máquinas de estado para usar integrações de serviços simuladas, crie um arquivo de configuração simulado. Nesse arquivo, você define a saída desejada de suas integrações de serviços como respostas simuladas e as execuções que usam essas respostas para simular um caminho de execução como casos de teste.

Ao fornecer o arquivo de configuração simulado ao Step Functions Local, você pode testar as chamadas de integração de serviços executando máquinas de estado que usam as respostas simuladas especificadas nos casos de teste em vez de fazer chamadas reais de integração de serviços.

> **Note**
>
> Se você não especificar respostas de integração de serviços simulados no arquivo de configuração simulado, o Step Functions Local invocará a integração de AWS serviços usando o endpoint que você configurou ao configurar o Step Functions Local. Para obter informações sobre como configurar endpoints para Step Functions Local, consulte Definindo opções de configuração para Step Functions Local.

Tópicos

- Etapa 3: executar testes de integração de serviços simulados
- Arquivo de configuração para integrações de serviços simulados

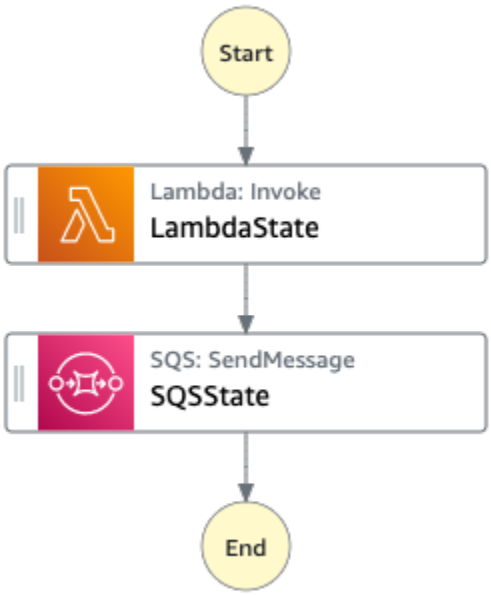
Principais conceitos neste tópico

Este tópico usa vários conceitos definidos na lista a seguir:

- Integrações de serviços simulados: refere-se aos estados de tarefas configurados para usar respostas simuladas em vez de realizar chamadas de serviço reais.
- Respostas simuladas: refere-se aos dados simulados que os estados da tarefa podem ser configurados para usar.
- Casos de teste: refere-se às execuções de máquinas de estado configuradas para usar integrações de serviços simuladas.
- Arquivo de configuração simulado: refere-se ao arquivo de configuração simulado que contém JSON, que define integrações de serviços simulados, respostas simuladas e casos de teste.

Etapa 1: especificar integrações de serviços simulados em um arquivo de configuração simulado

Você pode testar o AWS SDK do Step Functions e as integrações de serviços otimizadas usando o Step Functions Local. A imagem a seguir mostra a máquina de estado definida na guia Definição de máquina de estado:



Para fazer isso, você deve criar um arquivo de configuração simulado contendo seções conforme definido em Apresentando a estrutura da configuração simulada.

1. Crie um arquivo chamado `MockConfigFile.json` para configurar testes com integrações de serviços simuladas.

   O exemplo a seguir mostra um arquivo de configuração simulado referenciando uma máquina de estados com dois estados definidos chamados `LambdaState` e `SQSState`.

   Mock configuration file example

   Veja a seguir um exemplo de um arquivo de configuração simulado que demonstra como simular respostas da invocação de uma função do Lambda e do envio de uma mensagem para o Amazon SQS. Neste exemplo, a máquina de estado `LambdaSQSIntegration` contém três casos de teste chamados `HappyPath`, `RetryPath` e `HybridPath` que simulam os estados `Task` chamados `LambdaState` e `SQSState`. Esses estados usam as respostas de serviço simuladas `MockedLambdaSuccess`, `MockedSQSSuccess` e `MockedLambdaRetry`. Essas respostas de serviço simuladas são definidas na seção `MockedResponses` do arquivo.

```
{
  "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

Veja a seguir um exemplo de uma definição de máquina de estado chamada `LambdaSQSIntegration`, que define dois estados de tarefas de integração de serviços chamados `LambdaState` e `SQSState`. `LambdaState` contém uma política de repetição baseada em `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
    }
  }
}
```

Você pode executar a definição da máquina de estado `LambdaSQSIntegration` referenciada no arquivo de configuração simulado usando um dos seguintes casos de teste:

- `HappyPath` - Este teste simula a saída de `LambdaState` e `SQSState` usando `MockedLambdaSuccess` e `MockedSQSSuccess` respectivamente.
  - O `LambdaState` retorna o seguinte valor:

```
"0":{
  "Return":{
    "StatusCode":200,
    "Payload":{
      "StatusCode":200,
      "body":"Hello from Lambda!"
    }
  }
}
```

  - O `SQSState` retorna o seguinte valor:

```
"0":{
  "Return":{
    "MD5OfMessageBody":"3bcb6e8e-7h85-4375-b0bc-1a59812c6e51",
    "MessageId":"3bcb6e8e-8b51-4375-b0bc-1a59812c6e51"
  }
}
```

- `RetryPath` - Este teste simula a saída de `LambdaState` e `SQSState` usando `MockedLambdaRetry` e `MockedSQSSuccess` respectivamente. Além disso, o `LambdaState` está configurado para realizar quatro tentativas de repetição. As respostas simuladas para essas tentativas são definidas e indexadas no estado `MockedLambdaRetry`.
  - A tentativa inicial termina com uma falha de tarefa contendo uma causa e uma mensagem de erro, conforme mostrado no exemplo a seguir:

```
"0":{
```

```
  "Throw": {
    "Error": "Lambda.ResourceNotReadyException",
    "Cause": "Lambda resource is not ready."
  }
}
```

- A primeira e a segunda tentativas terminam com uma falha de tarefa contendo uma causa e uma mensagem de erro, conforme mostrado no exemplo a seguir:

```
"1-2":{
  "Throw": {
    "Error": "Lambda.TimeoutException",
    "Cause": "Lambda timed out."
  }
}
```

- A terceira tentativa termina com uma tarefa bem-sucedida contendo o resultado do estado da seção Payload na resposta simulada do Lambda.

```
"3":{
  "Return": {
    "StatusCode": 200,
    "Payload": {
      "StatusCode": 200,
      "body": "Hello from Lambda!"
    }
  }
}
```

> **Note**
>
> - Para estados com uma política de repetição, o Step Functions Local esgotará as tentativas de repetição definidas na política até receber uma resposta bem-sucedida. Isso significa que você deve indicar simulações para novas tentativas com números de tentativas consecutivos e deve abranger todas as tentativas antes de retornar uma resposta bem-sucedida.
> - Se você não especificar uma resposta simulada para uma tentativa de repetição específica, por exemplo, tentar novamente “3”, a execução da máquina de estado falhará.

- `HybridPath` - Este teste simula a saída de `LambdaState`. Depois de `LambdaState` ser executado com sucesso e receber dados simulados como resposta, `SQSState` executa uma chamada de serviço real para o recurso especificado na produção.

Para obter informações sobre como iniciar execuções de teste com integrações de serviços simuladas, consulte Etapa 3: executar testes de integração de serviços simulados.

2. Certifique-se de que a estrutura das respostas simuladas esteja em conformidade com a estrutura das respostas de serviço reais que você recebe ao fazer chamadas de serviço integradas. Para obter informações sobre os requisitos estruturais para respostas simuladas, consulte Configurando integrações de serviços simuladas.

   No exemplo anterior do arquivo de configuração simulado, as respostas simuladas são definidas em `MockedLambdaSuccess` e `MockedLambdaRetry` está em conformidade com a estrutura das respostas reais que são retornadas da chamada `HelloFromLambda`.

   > **Important**
   >
   > As respostas do serviço AWS podem variar em estrutura entre os diferentes serviços. O Step Functions Local não valida se as estruturas de resposta simuladas estão em conformidade com as estruturas reais de resposta do serviço. Você deve garantir que suas respostas simuladas estejam em conformidade com as respostas reais antes do teste. Para revisar a estrutura das respostas de serviço, você pode realizar as chamadas de serviço reais usando Step Functions ou visualizar a documentação desses serviços.

Etapa 2: forneça ao Step Functions Local o arquivo de configuração simulado

É possível fornecer o arquivo de configuração simulado ao Step Functions Local de uma das seguintes maneiras:

Docker

> **Note**
>
> Se você estiver usando a versão Docker do Step Functions Local, poderá fornecer o arquivo de configuração simulado usando somente uma variável de ambiente. Além

disso, você deve montar o arquivo de configuração simulado no contêiner do Step Functions Local na inicialização do servidor.

Monte o arquivo de configuração simulado em qualquer diretório dentro do contêiner do Step Functions Local. Em seguida, defina uma variável de ambiente chamada `SFN_MOCK_CONFIG` que contenha o caminho para o arquivo de configuração simulado no contêiner. Esse método permite que o arquivo de configuração simulado tenha qualquer nome, desde que a variável de ambiente contenha o caminho e o nome do arquivo.

O comando a seguir mostra o formato para iniciar a imagem do 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
```

O comando a seguir mostra o comando para iniciar a imagem do 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 uma das seguintes maneiras para fornecer o arquivo de configuração simulado ao Step Functions Local:

- Coloque o arquivo de configuração simulado no mesmo diretório que `Step FunctionsLocal.jar`. Ao usar esse método, você deve nomear o arquivo de configuração simulado `MockConfigFile.json`.
- Na sessão que executa o Step Functions Local, defina uma variável de ambiente chamada `SFN_MOCK_CONFIG` para o caminho completo do arquivo de configuração simulado. Esse método permite que o arquivo de configuração simulado tenha qualquer nome, desde que a variável de ambiente contenha o caminho e o nome do arquivo. No exemplo a seguir, a variável `SFN_MOCK_CONFIG` é definida para apontar para um arquivo de configuração simulado chamado `EnvSpecifiedMockConfig.json`, localizado no diretório `/home/workspace`.

```
export SFN_MOCK_CONFIG="/home/workspace/EnvSpecifiedMockConfig.json"
```

> **Note**
>
> - Se você não fornecer a variável de ambiente `SFN_MOCK_CONFIG` para o Step Functions Local, por padrão, ele tentará ler um arquivo de configuração simulado chamado `MockConfigFile.json` no diretório a partir do qual você iniciou o Step Functions Local.
> - Se você colocar o arquivo de configuração simulado no mesmo diretório que `Step FunctionsLocal.jar` e definir a variável de ambiente `SFN_MOCK_CONFIG`, o Step Functions Local lerá o arquivo especificado pela variável de ambiente.

Etapa 3: executar testes de integração de serviços simulados

Depois de criar e fornecer um arquivo de configuração simulado ao Step Functions Local, execute a máquina de estado configurada no arquivo de configuração simulado usando integrações de serviços simuladas. Em seguida, verifique os resultados da execução usando uma ação de API.

1. Crie uma máquina de estado com base na definição mencionada anteriormente no arquivo de configuração 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. Execute a máquina de estado usando integrações de serviços simuladas.

   Para usar o arquivo de configuração simulado, faça uma chamada de API `StartExecution` em uma máquina de estado configurada no arquivo de configuração simulado. Para fazer isso, anexe o sufixo, #*`test_name`*, ao ARN da máquina de estado usado por `StartExecution`. *`test_name`* é um caso de teste, configurado para a máquina de estado no mesmo arquivo de configuração simulado.

   O comando a seguir é um exemplo que usa a máquina de estado `LambdaSQSIntegration` e a configuração simulada. Neste exemplo, a máquina de estado `LambdaSQSIntegration` é executada usando o teste `HappyPath` definido em Etapa 1: especificar integrações de serviços simulados em um arquivo de configuração simulado. O teste `HappyPath` contém a configuração da execução para lidar com chamadas simuladas de integração de serviços que os estados `LambdaState` e `SQSState` fazem usando as respostas de serviço simuladas `MockedLambdaSuccess` e `MockedSQSSuccess`.

   ```
   aws stepfunctions start-execution \
       --endpoint http://localhost:8083 \
       --name executionWithHappyPathMockedServices \
       --state-machine arn:aws:states:us-
   east-1:123456789012:stateMachine:LambdaSQSIntegration#HappyPath
   ```

3. Veja a resposta de execução da máquina de estado.

   A resposta à chamada `StartExecution` usando um teste simulado de integração de serviços é a mesma resposta à chamada `StartExecution` normal, que retorna o ARN da execução e a data de início.

   Veja a seguir um exemplo de resposta à chamada `StartExecution` usando o teste simulado de integração de serviços:

   ```
   {
      "startDate":"2022-01-28T15:03:16.981000-05:00",
      "executionArn":"arn:aws:states:us-
   east-1:123456789012:execution:LambdaSQSIntegration:executionWithHappyPathMockedServices"
   }
   ```

4. Verifique os resultados da execução fazendo uma chamada de API `ListExecutions`, `DescribeExecution` ou `GetExecutionHistory`.

```
aws stepfunctions get-execution-history \
    --endpoint http://localhost:8083 \
    --execution-arn arn:aws:states:us-
east-1:123456789012:execution:LambdaSQSIntegration:executionWithHappyPathMockedServices
```

O exemplo a seguir demonstra partes de uma resposta à chamada `GetExecutionHistory` usando o ARN de execução do exemplo de resposta mostrado na etapa 2. Neste exemplo, a saída de `LambdaState` e `SQSState` são os dados simulados definidos no arquivo de configuração simulado `MockedLambdaSuccess` e `MockedSQSSuccess`. Além disso, os dados simulados são usados da mesma forma que os dados retornados pela execução de chamadas reais de integração de serviços seriam usados. Além disso, neste exemplo, a saída de `LambdaState` é passada 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
                }
            }
        },
        ...
    ]
}
```

Arquivo de configuração para integrações de serviços simulados

Para usar integrações de serviços simuladas, você primeiro deve criar um arquivo de configuração simulado chamado `MockConfigFile.json` contendo suas configurações simuladas. Em seguida, forneça ao Step Functions Local o arquivo de configuração simulado. Esse arquivo de configuração define casos de teste, que contêm estados simulados que usam respostas simuladas de integração

de serviços. A seção a seguir contém informações sobre a estrutura da configuração simulada que inclui os estados simulados e as respostas simuladas:

Tópicos

Apresentando a estrutura da configuração simulada

Uma configuração simulada é um objeto JSON contendo os campos de nível superior a seguir:

- `StateMachines` - Os campos desse objeto representam máquinas de estado configuradas para usar integrações de serviços simuladas.
- `MockedResponse` - Os campos desse objeto representam respostas simuladas para chamadas de integração de serviços.

A seguir está um exemplo de um arquivo de configuração simulado que inclui uma definição `StateMachine` e `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!"
```

```
          }
        }
      }
    }
  }
}
```

Referência de campo de configuração simulada

As seções a seguir explicam os campos de objeto de nível superior que você deve definir em sua configuração simulada.

- StateMachines
- MockedResponses

StateMachines

O objeto `StateMachines` define quais máquinas de estado usarão integrações de serviços simuladas. A configuração de cada máquina de estado é representada como um campo de nível superior de `StateMachines`. O nome do campo é o nome da máquina de estado e o valor é um objeto contendo um único campo chamado `TestCases`, cujos campos representam casos de teste dessa máquina de estado.

A sintaxe a seguir mostra uma máquina de estado com dois casos de teste:

```
"MyStateMachine": {
  "TestCases": {
    "HappyPath": {
      ...
    },
    "SadPath": {
      ...
    }
  }
```

TestCases

Os campos de `TestCases` representam casos de teste individuais para a máquina de estado. O nome de cada caso de teste deve ser exclusivo por máquina de estado e o valor de cada caso de teste é um objeto que especifica uma resposta simulada a ser usada para estados de tarefas na máquina de estado.

O exemplo a seguir de um `TestCase` vincula dois estados `Task` a dois `MockedResponses`:

```
"HappyPath": {
  "SomeTaskState": "SomeMockedResponse",
  "AnotherTaskState": "AnotherMockedResponse"
}
```

MockedResponses

`MockedResponses` é um objeto que contém vários objetos de resposta simulados com nomes de campo exclusivos. Um objeto de resposta simulada define o resultado bem-sucedido ou a saída de erro para cada invocação de um estado de tarefa simulado. Você especifica o número da invocação usando strings de números inteiros individuais, como “0”, “1”, “2” e “3”, ou um intervalo inclusivo de números inteiros, como “0-1”, “2-3”.

Ao simular uma tarefa, você deve especificar uma resposta simulada para cada invocação. Uma resposta deve conter um único campo chamado `Return` ou `Throw` cujo valor seja o resultado ou a saída de erro para a invocação simulada da Tarefa. Se você não especificar uma resposta simulada, a execução da máquina de estado falhará.

A seguir está um exemplo de um `MockedResponse` com objetos `Throw` e `Return`. Neste exemplo, nas primeiras três vezes em que a máquina de estado é executada, a resposta retornada é a especificada em `"0-2"`, e na quarta vez em que a máquina de estado é executada, a resposta retornada é a especificada em `"3"`.

```
"SomeMockedResponse": {
  "0-2": {
    "Throw": {
      ...
    }
  },
  "3": {
    "Return": {
      ...
    }
  }
}
```

> **Note**
>
> Se você estiver usando um estado `Map` e quiser garantir respostas previsíveis para o estado `Map`, defina o valor de `maxConcurrency` como 1. Se você definir um valor maior que 1, o Step Functions Local executará várias iterações simultaneamente, o que fará com que a ordem geral de execução dos estados nas iterações seja imprevisível. Isso pode ainda fazer com que o Step Functions Local use diferentes respostas simuladas para estados de iteração de uma execução para a próxima.

Return

`Return` é representado como um campo dos objetos `MockedResponse`. Ele especifica o resultado bem-sucedido de um estado de tarefa simulado.

Veja a seguir um exemplo de um objeto `Return` que contém uma resposta simulada para chamar `Invoke` em uma função do Lambda:

```
"Return": {
  "StatusCode": 200,
  "Payload": {
    "StatusCode": 200,
    "body": "Hello from Lambda!"
  }
}
```

Throw

`Throw` é representado como um campo dos objetos `MockedResponse`. Ele especifica a saída de erro de uma tarefa com falha. O valor de `Throw` deve ser um objeto contendo campos `Error` e `Cause` com valores de string. Além disso, o valor da string que você especifica no campo `Error` do `MockConfigFile.json` deve corresponder aos erros tratados nas seções `Retry` e `Catch` da sua máquina de estado.

Veja a seguir um exemplo de um objeto `Throw` que contém uma resposta simulada para chamar `Invoke` em uma função do Lambda:

```
"Throw": {
  "Error": "Lambda.TimeoutException",
  "Cause": "Lambda timed out."
```

```
}
```

Configurando integrações de serviços simuladas

Você pode simular qualquer integração de serviços usando o Step Functions Local. No entanto, o Step Functions Local não impõe que as simulações sejam iguais às APIs reais. Uma tarefa simulada nunca chamará o endpoint do serviço. Se você não especificar uma resposta simulada, uma tarefa tentará chamar os endpoints do serviço. Além disso, o Step Functions Local gerará automaticamente um token de tarefa quando você simular uma tarefa usando o `.waitForTaskToken`.

Práticas recomendadas do Step Functions

As melhores práticas a seguir para implementar fluxos de trabalho do AWS Step Functions podem ajudar você a otimizar a performance de suas implementações.

Tópicos

Usar tempos limite para evitar travar execuções

Por padrão, a Amazon States Language não especifica tempos limite para definições de máquinas de estado. Sem um tempo limite explícito, o Step Functions frequentemente se baseia somente em uma resposta de um operador de atividade para saber que uma tarefa foi concluída. Se algo der errado e o campo `TimeoutSeconds` não estiver especificado para um estado `Activity` ou `Task`, uma execução ficará travada, aguardando uma resposta que nunca chegará.

Para evitar essa situação, ao criar uma `Task` em sua máquina de estado, especifique um tempo limite razoável. Por exemplo:

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

Se você usar um retorno de chamada com um token de tarefa (.waitForTaskToken), recomendamos usar pulsações e adicionar o campo `HeartbeatSeconds` à definição do estado `Task`. Você pode

definir `HeartbeatSeconds` para ser menor que o tempo limite da tarefa, de modo que, se o seu fluxo de trabalho falhar com um erro de heartbeat, você saberá que isso se deve à falha da tarefa, e não ao fato de a tarefa demorar muito para ser concluída.

```
{
  "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 ver mais informações, consulte Estado da tarefa na documentação da Amazon States Language.

> **Note**
>
> Você pode definir um tempo limite para sua máquina de estado usando o campo `TimeoutSeconds` na definição da Amazon States Language. Para obter mais informações, consulte Estrutura da máquina de estado.

Use ARNs do Amazon S3 em vez de transmitir grandes cargas

As execuções que transmitem grandes cargas de dados entre os estados podem ser encerradas. Se os dados que você está transmitindo entre estados puderem crescer para mais de 256 KB, use o Amazon Simple Storage Service (Amazon S3) para armazenar os dados e analise o nome do recurso da Amazon (ARN) do bucket no parâmetro `Payload` para obter o nome e o valor da chave do bucket. Como alternativa, ajuste a implementação para passar cargas menores em suas execuções.

No exemplo a seguir, uma máquina de estado transmite a entrada para uma função AWS Lambda, que processa um arquivo JSON em um bucket do Amazon S3. Depois de executar essa máquina de estado, a função do Lambda lê o conteúdo do arquivo JSON e retorna o conteúdo do arquivo como saída.

Criar a função do Lambda

A função do Lambda a seguir chamada *pass-large-payload* lê o conteúdo de um arquivo JSON armazenado em um bucket específico do Amazon S3.

> **Note**
>
> Depois de criar essa função do Lambda, forneça a permissão apropriada ao perfil do IAM para ler de um bucket do Amazon S3. Por exemplo, anexe a permissão AmazonS3ReadOnlyAccess ao perfil da função do 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
```

Criar a máquina de estado

A máquina de estado a seguir invoca a função do Lambda que você criou 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
      }
   }
}
```

Em vez de transmitir uma grande quantidade de dados na entrada, é possível salvar esses dados em um bucket do Amazon S3 e transmitir o nome do recurso da Amazon (ARN) do bucket no parâmetro `Payload` para obter o nome e valor da chave do bucket. Assim, a função do Lambda pode usar esse ARN para acessar os dados diretamente. Veja a seguir um exemplo de entrada para uma execução de máquina de estado, onde os dados estão armazenados em `data.json` em um bucket do Amazon S3 denominado *`large-payload-json`*.

```
{
  "key": "data.json",
  "bucket": "arn:aws:s3:::large-payload-json"
}
```

Evitar atingir a cota do histórico

O AWS Step Functions tem uma cota fixa de 25.000 entradas no histórico de eventos de execução. Quando uma execução atinge 24.999 eventos, ela aguarda a ocorrência do próximo evento.

- Se o número do evento 25.000 for `ExecutionSucceeded`, a execução será concluída com êxito.
- Se o número do evento 25.000 não for `ExecutionSucceeded`, o evento `ExecutionFailed` será registrado e a execução da máquina de estado falhará por ter atingido o limite do histórico

Para evitar atingir essa cota em execuções de longa duração, você pode tentar uma das seguintes soluções alternativas:

- Usar o estado Mapa no modo Distribuído. Nesse modo, o estado `Map` executa cada iteração como uma execução de fluxo de trabalho secundário, o que permite um processamento simultâneo de até 10.000 execuções paralelas de fluxo de trabalho secundário. Cada execução de fluxo de trabalho secundário tem seu próprio histórico de execução separado do fluxo de trabalho principal.
- Iniciar uma nova execução de máquina de estado diretamente do estado `Task` de uma execução em andamento. Para iniciar essas execuções de fluxo de trabalho aninhadas, use a ação da API `StartExecution` do Step Functions na máquina de estado principal juntamente com os parâmetros necessários. Para ver mais informações sobre o uso de fluxos de trabalho aninhados, consulte Iniciar execuções de fluxo de trabalho usando um estado Tarefa ou o tutorial Usar uma ação da API do Step Functions para continuar uma nova execução.

  > **Tip**
  >
  > Para implementar um exemplo de fluxo de trabalho aninhado em seu Conta da AWS, consulte Módulo 13 - Fluxos de trabalho expressos aninhados.

- Implemente um padrão que usa uma função do AWS Lambda capaz de iniciar uma nova execução de sua máquina de estado para dividir o trabalho em andamento em várias execuções de fluxo de trabalho. Para mais informações, consulte o tutorial Usar uma função do Lambda para continuar uma nova execução.

Lidar com exceções do serviço Lambda

O AWS Lambda pode apresentar erros de serviço temporários ocasionalmente. Nesse caso, a invocação do Lambda resulta em um erro 500, como `ClientExecutionTimeoutException`, `ServiceException`, `AWSLambdaException` ou `SdkClientException`. Como uma prática recomendada, lide com essas exceções de maneira proativa em sua máquina de estado para executar `Retry` da invocação da função do Lambda ou para executar `Catch` do erro.

Erros do Lambda são relatados como `Lambda.ErrorName`. Para repetir um erro de exceção do serviço Lambda;, você pode usar o código `Retry` a seguir:

```
"Retry": [ {
   "ErrorEquals": [ "Lambda.ClientExecutionTimeoutException",
 "Lambda.ServiceException", "Lambda.AWSLambdaException", "Lambda.SdkClientException"],
   "IntervalSeconds": 2,
   "MaxAttempts": 6,
   "BackoffRate": 2
} ]
```

> **Note**
>
> Os erros não tratados no Lambda são relatados como `Lambda.Unknown` na saída do erro. Isso inclui erros de falta de memória e tempos limite de função. Você pode combinar com `Lambda.Unknown`, `States.ALL` ou `States.TaskFailed` para lidar com esses erros. Quando o Lambda atinge o número máximo de invocações, o erro é `Lambda.TooManyRequestsException`. Para obter mais informações sobre erros da função do Lambda, consulte Tratamento de erros e novas tentativas automáticas no Guia do desenvolvedor do AWS Lambda.

Para obter mais informações, consulte as informações a seguir.

- Nova tentativa após um erro
- Tratar condições de erro usando uma máquina de estado Step Functions
- Erros de invocação do Lambda

Evitar latência ao fazer uma sondagem de tarefas de atividade

A API `GetActivityTask` foi criada para fornecer um `taskToken` exatamente uma vez. Se um `taskToken` é descartado ao se comunicar com um operador de atividade, várias solicitações `GetActivityTask` podem ser bloqueadas por 60 segundos de espera por uma resposta até que `GetActivityTask` expire.

Se você tem apenas um pequeno número de sondagens aguardando uma resposta, é possível que todas as solicitações sejam colocadas em fila depois da solicitação bloqueada e interrompidas. No

entanto, se você tiver um grande número de sondagens pendentes para cada nome do recurso da Amazon (ARN) de atividade e uma porcentagem de suas solicitações estiver paralisada em espera, haverá várias outras que ainda poderão obter um `taskToken` e começar a processar os trabalhos.

Para sistemas de produção, recomendamos pelo menos 100 sondagens abertas por ARN de atividade em cada momento específico. Se uma sondagem é bloqueada, e uma parte dessas sondagens é colocada em fila depois dela, há várias outras solicitações que receberão um `taskToken` para processar trabalhos enquanto a solicitação `GetActivityTask` está bloqueada.

Para evitar esses tipos de problemas de latência ao fazer uma sondagem de tarefas:

- Implemente seus agentes de sondagem como threads separados do trabalho na sua implementação do operador de atividade.
- Tenha pelo menos 100 sondagens abertas por ARN de atividade em cada momento específico.

> **Note**
>
> Dimensionar para 100 sondagens abertas por ARN pode ser caro. Por exemplo, 100 sondagens de funções do Lambda por ARN custam 100 vezes mais caro que ter uma única função do Lambda com 100 threads de sondagem. Para reduzir a latência e minimizar os custos, use uma linguagem de E/S assíncrona e implemente vários threads de sondagem por operador. Para obter um operador de atividade de exemplo em que os threads do agente de sondagem são separados de threads do trabalho, consulte Exemplo de operador de atividade em Ruby.

Para obter mais informações sobre atividades e operadores de atividade, consulte Atividades.

Escolher fluxos de trabalho padrão ou expresso

O AWS Step Functions oferece fluxos de trabalho padrão como o tipo de fluxo de trabalho padrão, com a opção de escolher fluxos de trabalho expresso.

É possível escolher fluxos de trabalho padrão quando precisar de fluxos de trabalho de longa duração, duráveis e auditáveis ou fluxos de trabalho expressos para cargas de trabalho de processamento de eventos de alto volume. Suas execuções de máquina de estado se comportarão de forma diferente, dependendo de qual `Type` você selecionar. O `Type` que você escolher não pode ser alterado depois que sua máquina de estado for criada.

- Para ver informações detalhadas sobre as diferenças entre fluxos de trabalho padrão e expresso, consulte Comparação entre os fluxos de trabalho padrão e expresso.
- Para ver informações sobre como otimizar custos ao criar fluxos de trabalho de tecnologia sem servidor usando o Step Functions, consulte Otimização de custos usando o fluxos de trabalho expressos.

Restrições de tamanho de políticas de recursos do Amazon CloudWatch Logs

As políticas de recursos do CloudWatch Logs são limitadas a 5.120 caracteres. Quando o CloudWatch Logs detecta que uma política se aproxima desse limite de tamanho, ele ativa automaticamente grupos de logs que começam com `/aws/vendedlogs/`.

Ao criar uma máquina de estado com o registro em log habilitado, o Step Functions deve atualizar a política de recursos do CloudWatch Logs com o grupo de logs especificado. Para evitar que o limite de tamanho de políticas de recursos do CloudWatch Logs seja atingido, insira o prefixo `/aws/vendedlogs/` nos nomes de grupos de logs do CloudWatch Logs. Os nomes dos grupos de logs criados no console do Step Functions têm o prefixo `/aws/vendedlogs/states`. Para ver mais informações, consulte Habilitar o registro em log de determinados serviços da AWS.

Se você não conseguir acessar o CloudWatch Logs, consulte Unable to access the CloudWatch Logs para obter informações.

Usando AWS Step Functions com outros serviços

Saiba como coordenar outras APIs Serviços da AWS ou chamar outras APIs de terceiros com. AWS Step Functions

Tópicos

Ligue para outros AWS serviços

Com as integrações de AWS serviços, você pode chamar ações de API e coordenar execuções diretamente do seu fluxo de trabalho. Você pode usar as integrações do AWS SDK do Step Functions para chamar qualquer um dos mais de duzentos AWS serviços diretamente de sua máquina de estado, dando acesso a mais de nove mil ações de API. Ou você pode usar as integrações otimizadas do Step Functions, que foram personalizadas para fornecer funcionalidades especiais para o fluxo de trabalho. Algumas ações de API estão disponíveis nos dois tipos de integração. Quando possível, recomendamos usar a integração otimizada.

Coordene esses serviços diretamente de um estado `Task` no Amazon States Language. Por exemplo, usando o Step Functions, é possível chamar outros serviços para:

- Invoque uma AWS Lambda função.
- Execute um AWS Batch trabalho e, em seguida, execute ações diferentes com base nos resultados.
- Inserir ou obter um item do Amazon DynamoDB.
- Executar uma tarefa do Amazon Elastic Container Service (Amazon ECS) e aguardar que ela seja concluída.
- Publicar em um tópico do Amazon Simple Notification Service (Amazon SNS).

- Enviar uma mensagem do Amazon Simple Queue Service (Amazon SQS).
- Gerencie um trabalho para AWS Glue nossa Amazon SageMaker.
- Criar fluxos de trabalho para executar trabalhos do Amazon EMR.
- Inicie a execução AWS Step Functions de um fluxo de trabalho.

Integrações otimizadas

As integrações otimizadas foram personalizadas pelo Step Functions para fornecer funcionalidades especiais para um contexto de fluxo de trabalho. Por exemplo, o Lambda `Invoke` converte sua saída de API de um JSON de escape em um objeto JSON. O AWS Batch`SubmitJob` permite pausar a execução até que o trabalho seja concluído. O primeiro conjunto de integrações otimizadas foi lançado em 2018 e agora existem mais de cinquenta APIs.

AWS Integrações de SDK

AWS As integrações do SDK funcionam exatamente como uma chamada de API padrão usando o AWS SDK. Eles oferecem a capacidade de chamar mais de nove mil APIs em mais de duzentos AWS serviços diretamente da definição de sua máquina de estado.

Suporte ao padrão de integração

Os fluxos de trabalho padrão e os fluxos de trabalho expressos oferecem suporte às mesmas integrações, mas não aos mesmos padrões de integração.

- O suporte ao padrão de integrações otimizadas é diferente para cada integração.
- Os fluxos de trabalho expressos não oferecem suporte a Run a Job (.sync) nem a Wait for Callback (). waitForTaskSímbolo).
- Para ter mais informações, consulte Comparação entre os fluxos de trabalho padrão e expresso.

Standard Workflows

Integrações de serviços compatíveis

	Serviço	Resposta de solicitação	Executar um trabalho (`.sync`)	Aguardar o retorno de chamada (`.waitForTaskToken`)
Integrações otimizadas	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	✓		✓
	AWS Elemental MediaConvert	✓	✓	

	Serviço	Resposta de solicitação	Executar um trabalho (`.sync`)	Aguardar o retorno de chamada (`.waitForTaskToken`)
	Amazon SageMaker	✓	✓	
	Amazon SNS	✓		✓
	Amazon SQS	✓		✓
	AWS Step Functions	✓	✓	✓
AWS Integrações de SDK	Mais de duzentas	✓		✓

Express Workflows

Integrações de serviços compatíveis

	Serviço	Resposta de solicitação	Executar um trabalho (`.sync`)	Aguardar o retorno de chamada (`.waitForTaskToken`)
Integrações otimizadas	Amazon API Gateway	✓		
	Amazon Athena	✓		
	AWS Batch	✓		
	Amazon Bedrock	✓		

	Serviço	Resposta de solicitação	Executar um trabalho (`.sync`)	Aguardar o retorno de chamada (`.waitForTaskToken`)
	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	✓		
	AWS Step Functions	✓		

	Serviço	Resposta de solicitação	Executar um trabalho (`.sync`)	Aguardar o retorno de chamada (`.waitForTaskToken`)
AWS Integrações de SDK	Mais de duzentas	✓		

Acesso entre contas

O Step Functions fornece acesso entre contas a recursos configurados de forma diferente Contas da AWS em seus fluxos de trabalho. Usando as integrações de serviços Step Functions, você pode invocar qualquer AWS recurso entre contas, mesmo que isso AWS service (Serviço da AWS) não ofereça suporte a políticas baseadas em recursos ou chamadas entre contas.

Para ter mais informações, consulte Acessando recursos em outros Contas da AWS em seus fluxos de trabalho.

AWS Integrações de serviços SDK

AWS Step Functions se integra com Serviços da AWS, permitindo que você chame as ações de API de cada serviço a partir do seu fluxo de trabalho. Você pode usar as integrações do AWS SDK do Step Functions para chamar quase todas as ações AWS service (Serviço da AWS) de API da sua máquina de estado. Você também pode usar as integrações otimizadas do Step Functions, que foram personalizadas para fornecer funcionalidades especiais para o fluxo de trabalho.

Alguns serviços ou SDKs podem não estar disponíveis como integrações de AWS SDK, temporária ou permanentemente. Os serviços lançados recentemente podem não ter interações com o SDK disponíveis até uma atualização posterior. Alguns serviços exigem configuração personalizada, como a especificação de um endpoint específico do cliente, que pode ser mais adequado para uma integração otimizada. Outros SDKs não são adequados para uso em um fluxo de trabalho, como aqueles para streaming de áudio ou vídeo. Por fim, alguns serviços podem ser retidos até passarem por determinadas validações internas realizadas pelo Step Functions.

> **Tip**
>
> Para implantar um exemplo de fluxo de trabalho que usa integrações do AWS SDK no seu Conta da AWS, consulte o Módulo 9 - Integrações de serviços do AWS SDK no The Workshop. AWS Step Functions

Tópicos

Usando integrações de serviços AWS do SDK

Para usar as integrações do AWS SDK, você especifica o nome do serviço e a chamada da API e, opcionalmente, um padrão de integração do serviço.

> **Note**
>
> - Os parâmetros em Step Functions são expressos em PascalCase, mesmo se a API do serviço nativo estiver no CamelCase. Por exemplo, é possível usar a ação da API do Step Functions `startSyncExecution` e especificar o parâmetro como `StateMachineArn`.
> - Para ações de API que aceitam parâmetros enumerados, como a ação da API `DescribeLaunchTemplateVersions` para o Amazon EC2, especifique uma versão plural do nome do parâmetro. Por exemplo, especifique `Filters` para o parâmetro `Filter.N` da ação da API `DescribeLaunchTemplateVersions`.

Você pode chamar os serviços do AWS SDK diretamente da Amazon States Language no `Resource` campo de um estado de tarefa. Para fazer isso, use a sintaxe a seguir.

`arn:aws:states:::aws-sdk:`*`serviceName`*`:`*`apiAction`*`.`*`[serviceIntegrationPattern]`*

Por exemplo, para o Amazon EC2, você pode usar `arn:aws:states:::aws-sdk:ec2:describeInstances`. Isso retornaria a saída conforme definida para a chamada da API describeInstances do Amazon EC2.

Se uma integração do AWS SDK encontrar um erro, o campo Erro resultante será composto pelo nome do serviço e pelo nome do erro, separados por um caractere de ponto final:. *`ServiceName`*`.`*`ErrorName`* Tanto o nome do serviço quanto o nome do erro terão o estilo Pascal case. Você também pode ver o nome do serviço, em letras minúsculas, no campo Recurso do estado da tarefa. Você pode encontrar os nomes dos possíveis erros na documentação de referência da API do serviço de destino.

Por exemplo, você pode usar a integração do `arn:aws:states:::aws-sdk:acmpca:deleteCertificateAuthority` AWS SDK. A Referência da API do AWS Private Certificate Authority indica que a ação da API `DeleteCertificateAuthority` pode resultar em um `ResourceNotFoundException`, por exemplo. Para lidar com esse erro, você deve especificar o erro `AcmPca.ResourceNotFoundException` nos retriers ou catchers do estado da tarefa. Para obter mais informações sobre como lidar com erros no Step Functions, consulte Tratamento de erros no Step Functions.

O Step Functions não pode gerar automaticamente políticas do IAM para integrações do AWS SDK. Depois de criar a máquina de estado, você precisará navegar até o console do IAM e configurar as políticas de função. Consulte Políticas do IAM para serviços integrados Para mais informações.

Veja o Reúna informações do bucket do Amazon S3 usando integrações de serviços AWS SDK tutorial para ver um exemplo de como usar as integrações do AWS SDK.

Integrações de serviços AWS SDK compatíveis

A tabela a seguir lista as integrações de serviços do AWS SDK suportadas pelo Step Functions. A coluna do recurso do estado da *`Task`* lista a sintaxe para chamar uma ação de API específica ao usar a integração para o serviço especificado na coluna Nome do serviço à esquerda. A coluna Data de suporte fornece informações sobre as datas em que a integração de serviços foi compatível. Além disso, a coluna Prefixo de exceção, à direita, lista os prefixos de exceção para cada integração de serviço. Esses prefixos de exceção estão presentes nas exceções que são geradas quando você executa erroneamente uma AWS integração de serviços SDK com Step Functions.

Note

- Os serviços marcados com *** têm ações de API que não são compatíveis com o Step Functions no momento. Para obter informações sobre as ações que não são compatíveis com um serviço, consulte a tabela Ações de API não compatíveis para serviços compatíveis.
- Para obter informações sobre as atualizações feitas em cada lançamento para expandir o suporte às integrações do AWS SDK, consulte. Registro de alterações para integrações de AWS SDK compatíveis

Important

O suporte à ação da API é lançado trimestralmente. As atualizações de ações já suportadas, como novos parâmetros, podem não estar imediatamente disponíveis.

Nome do serviço	Recurso do estado da **`Task`**	Data de suporte	Prefixo de exceção
AWS AppFabric	`arn:aws:states:::aws-sdk:appfabric: [apiAction]`	18 de janeiro de 2024	AppFabric
B2B Data Interchange	`arn:aws:states:::aws-sdk:b2bi: [apiAction]`	18 de janeiro de 2024	B2Bi
Exportações de dados da AWS	`arn:aws:states:::aws-sdk:bcmdataexports: [apiAction]`	18 de janeiro de 2024	BcmDataExports
Amazon Bedrock	`arn:aws:states:::aws-sdk:bedrock: [apiAction]`	18 de janeiro de 2024	Bedrock

Nome do serviço	Recurso do estado da **`Task`**	Data de suporte	Prefixo de exceção
Amazon Bedrock Agents	`arn:aws:states:::aws-sdk:bedrockagent: [apiAction]`	18 de janeiro de 2024	BedrockAgent
Amazon Bedrock Runtime Agents	`arn:aws:states:::aws-sdk:bedrockagentruntime: [apiAction]`	18 de janeiro de 2024	BedrockAgentRuntime
Amazon Bedrock Runtime	`arn:aws:states:::aws-sdk:bedrockruntime: [apiAction]`	18 de janeiro de 2024	BedrockRuntime
AWS Clean Rooms	`arn:aws:states:::aws-sdk:cleanroomsml: [apiAction]`	18 de janeiro de 2024	CleanRoomsML
Amazon CloudFront KeyValueStore	`arn:aws:states:::aws-sdk:cloudfrontkeyvaluestore: [apiAction]`	18 de janeiro de 2024	CloudFrontKeyValueStore
CodeGuru Security	`arn:aws:states:::aws-sdk:codegurusecurity: [apiAction]`	18 de janeiro de 2024	CodeGuruSecurity
Hub de Otimização de Custos da AWS	`arn:aws:states:::aws-sdk:costoptimizationhub: [apiAction]`	18 de janeiro de 2024	CostOptimizationHub
Amazon DataZone	`arn:aws:states:::aws-sdk:datazone: [apiAction]`	18 de janeiro de 2024	DataZone

Nome do serviço	Recurso do estado da **`Task`**	Data de suporte	Prefixo de exceção
Amazon EKS Auth	`arn:aws:states:::aws-sdk:eksauth: [apiAction]`	18 de janeiro de 2024	EksAuth
AWS Entity Resolution	`arn:aws:states:::aws-sdk:entityresolution: [apiAction]`	18 de janeiro de 2024	EntityResolution
Nível gratuito da AWS	`arn:aws:states:::aws-sdk:freetier: [apiAction]`	18 de janeiro de 2024	FreeTier
Amazon Inspector Scan	`arn:aws:states:::aws-sdk:inspectorscan: [apiAction]`	18 de janeiro de 2024	InspectorScan
AWS Launch Wizard	`arn:aws:states:::aws-sdk:launchwizard: [apiAction]`	18 de janeiro de 2024	LaunchWizard
Amazon Managed Blockchain Query	`arn:aws:states:::aws-sdk:managedblockchainquery: [apiAction]`	18 de janeiro de 2024	ManagedBlockchainQuery
AWS Elemental MediaPackage V2	`arn:aws:states:::aws-sdk:mediapackagev2: [apiAction]`	18 de janeiro de 2024	MediaPackageV2
AWS HealthImaging	`arn:aws:states:::aws-sdk:medicalimaging: [apiAction]`	18 de janeiro de 2024	MedicalImaging
Network Manager	`arn:aws:states:::aws-sdk:networkmanager: [apiAction]`	18 de janeiro de 2024	NetworkManager

Nome do serviço	Recurso do estado da **Task**	Data de suporte	Prefixo de exceção
AWS Payment Cryptography	`arn:aws:states:::aws-sdk:paymentcryptography:` *`[apiAction]`*	18 de janeiro de 2024	PaymentCryptography
AWS Payment Cryptography Data	`arn:aws:states:::aws-sdk:paymentcryptographydata:` *`[apiAction]`*	18 de janeiro de 2024	PaymentCryptographyData
AWS Private CA Connector for Active Directory	`arn:aws:states:::aws-sdk:pcaconnectorad:` *`[apiAction]`*	18 de janeiro de 2024	PcaConnectorAd
Amazon Q Business	`arn:aws:states:::aws-sdk:qbusiness:` *`[apiAction]`*	18 de janeiro de 2024	QBusiness
Amazon Q Connect	`arn:aws:states:::aws-sdk:qconnect:` *`[apiAction]`*	18 de janeiro de 2024	QConnect
AWS re:Post	`arn:aws:states:::aws-sdk:repostspace:` *`[apiAction]`*	18 de janeiro de 2024	Repostspace
Amazon Timestream Query	`arn:aws:states:::aws-sdk:timestreamquery:` *`[apiAction]`*	18 de janeiro de 2024	TimestreamQuery
Amazon Timestream Write	`arn:aws:states:::aws-sdk:timestreamwrite:` *`[apiAction]`*	18 de janeiro de 2024	TimestreamWrite

Nome do serviço	Recurso do estado da **`Task`**	Data de suporte	Prefixo de exceção
Trusted Advisor	`arn:aws:states:::aws-sdk:trustedadvisor:` *`[apiAction]`*	18 de janeiro de 2024	TrustedAdvisor
Verified Permissions	`arn:aws:states:::aws-sdk:verifiedpermissions:` *`[apiAction]`*	18 de janeiro de 2024	VerifiedPermissions
Amazon WorkSpaces Thin Client	`arn:aws:states:::aws-sdk:workspacesthinclient:` *`[apiAction]`*	18 de janeiro de 2024	WorkSpacesThinClient
AWS CloudTrail Data	`arn:aws:states:::aws-sdk:cloudtraildata:` *`[apiAction]`*	16 de junho de 2023	CloudTrailData
Amazon CloudWatch Internet Monitor	`arn:aws:states:::aws-sdk:internetmonitor:` *`[apiAction]`*	16 de junho de 2023	InternetMonitor
Amazon Interactive Video Service RealTime	`arn:aws:states:::aws-sdk:ivsrealtime:` *`[apiAction]`*	16 de junho de 2023	IvsRealTime
AWS IoT TwinMaker	`arn:aws:states:::aws-sdk:iottwinmaker:` *`[apiAction]`*	16 de junho de 2023	IoTTwinMaker
Amazon OpenSearch Ingestion	`arn:aws:states:::aws-sdk:osis:` *`[apiAction]`*	16 de junho de 2023	Osis

Nome do serviço	Recurso do estado da **`Task`**	Data de suporte	Prefixo de exceção
AWS Telco Network Builder	`arn:aws:states:::aws-sdk: tnb:[apiAction]`	16 de junho de 2023	Tnb
Amazon VPC Lattice	`arn:aws:states:::aws-sdk:vpclattice: [apiAction]`	16 de junho de 2023	VpcLattice
Amazon Chime Media Pipelines	`arn:aws:states:::aws-sdk:chimesdkmediapipelines: [apiAction]`	17 de fevereiro de 2023	ChimeSdkMediaPipelines
Amazon Chime Voice	`arn:aws:states:::aws-sdk:chimesdkvoice: [apiAction]`	17 de fevereiro de 2023	ChimeSdkVoice
Amazon CodeCatalyst	`arn:aws:states:::aws-sdk:codecatalyst: [apiAction]`	17 de fevereiro de 2023	CodeCatalyst
Amazon Connect Cases	`arn:aws:states:::aws-sdk:connectcases: [apiAction]`	17 de fevereiro de 2023	ConnectCases
Amazon DocumentDB Elastic Clusters	`arn:aws:states:::aws-sdk:docdbelastic: [apiAction]`	17 de fevereiro de 2023	DocDbElastic
Amazon EMR Serverless	`arn:aws:states:::aws-sdk:emrserverless: [apiAction]`	17 de fevereiro de 2023	EmrServerless
Amazon IVS Chat	`arn:aws:states:::aws-sdk:ivs: [apiAction]`	17 de fevereiro de 2023	Ivs

Nome do serviço	Recurso do estado da **`Task`**	Data de suporte	Prefixo de exceção
Amazon Kendra Intelligent Ranking	`arn:aws:states:::aws-sdk:kendraranking: [apiAction]`	17 de fevereiro de 2023	KendraRanking
AWS HealthOmics	`arn:aws:states:::aws-sdk:omics: [apiAction]`	17 de fevereiro de 2023	Omics
Amazon Redshift Serverless	`arn:aws:states:::aws-sdk:redshiftserverless: [apiAction]`	17 de fevereiro de 2023	RedshiftServerless
Amazon Security Lake	`arn:aws:states:::aws-sdk:securitylake: [apiAction]`	17 de fevereiro de 2023	SecurityLake
AWS Backup Storage	`arn:aws:states:::aws-sdk:backupstorage: [apiAction]`	17 de fevereiro de 2023	BackupStorage
AWS Clean Rooms	`arn:aws:states:::aws-sdk:cleanrooms: [apiAction]`	17 de fevereiro de 2023	CleanRooms
AWS Control Tower	`arn:aws:states:::aws-sdk:controltower: [apiAction]`	17 de fevereiro de 2023	ControlTower
AWS Health	`arn:aws:states:::aws-sdk:health: [apiAction]`	17 de fevereiro de 2023	Health
AWS IoT FleetWise	`arn:aws:states:::aws-sdk:iotfleetwise: [apiAction]`	17 de fevereiro de 2023	IotFleetWise

Nome do serviço	Recurso do estado da **`Task`**	Data de suporte	Prefixo de exceção
AWS Mainframe Modernization	`arn:aws:states:::aws-sdk:m2:` *`[apiAction]`*	17 de fevereiro de 2023	M2
Orquestrador do AWS Migration Hub	`arn:aws:states:::aws-sdk:migrationhuborchestrator:` *`[apiAction]`*	17 de fevereiro de 2023	Migration HubOrchestrator
AWS Private 5G	`arn:aws:states:::aws-sdk:privatenetworks:` *`[apiAction]`*	17 de fevereiro de 2023	PrivateNetworks
Explorador de recursos da AWS	`arn:aws:states:::aws-sdk:resourceexplorer2:` *`[apiAction]`*	17 de fevereiro de 2023	ResourceExplorer2
AWS SimSpace Weaver	`arn:aws:states:::aws-sdk:simspaceweaver:` *`[apiAction]`*	17 de fevereiro de 2023	SimSpaceWeaver
AWS Support App	`arn:aws:states:::aws-sdk:supportapp:` *`[apiAction]`*	17 de fevereiro de 2023	SupportApp
CloudWatch Observability Access Manager	`arn:aws:states:::aws-sdk:oam:` *`[apiAction]`*	17 de fevereiro de 2023	Oam
EventBridge Pipes	`arn:aws:states:::aws-sdk:pipes:` *`[apiAction]`*	17 de fevereiro de 2023	Pipes
EventBridge Scheduler	`arn:aws:states:::aws-sdk:scheduler:` *`[apiAction]`*	17 de fevereiro de 2023	Scheduler

Nome do serviço	Recurso do estado da **`Task`**	Data de suporte	Prefixo de exceção
IAM Roles Anywhere	`arn:aws:states:::aws-sdk:rolesanywhere: [apiAction]`	17 de fevereiro de 2023	RolesAnywhere
Kinesis Video WebRTC Storage	`arn:aws:states:::aws-sdk:kinesisvideowebrtcstorage: [apiAction]`	17 de fevereiro de 2023	KinesisVideoWebRtcStorage
License Manager Linux Subscriptions	`arn:aws:states:::aws-sdk:licensemanagerlinuxsubscriptions: [apiAction]`	17 de fevereiro de 2023	LicenseManagerLinuxSubscriptions
License Manager User Subscriptions	`arn:aws:states:::aws-sdk:licensemanagerusersubscriptions: [apiAction]`	17 de fevereiro de 2023	LicenseManagerUserSubscriptions
OpenSearch Serverless	`arn:aws:states:::aws-sdk:opensearchserverless: [apiAction]`	17 de fevereiro de 2023	OpenSearchServerless
Route 53 ARC Zonal Shift	`arn:aws:states:::aws-sdk:arczonalshift: [apiAction]`	17 de fevereiro de 2023	ArcZonalShift
SageMaker Geospatial	`arn:aws:states:::aws-sdk:sagemakergeospatial: [apiAction]`	17 de fevereiro de 2023	SageMaker Geospatial
SageMaker Metrics	`arn:aws:states:::aws-sdk:sagemakermetrics: [apiAction]`	17 de fevereiro de 2023	SageMakerMetrics

Nome do serviço	Recurso do estado da **`Task`**	Data de suporte	Prefixo de exceção
Systems Manager for SAP	`arn:aws:states:::aws-sdk:ssmsap:` *`[apiAction]`*	17 de fevereiro 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 setembro de 2021	Amplify
AWS App Mesh	`arn:aws:states:::aws-sdk:appmesh:` *`[apiAction]`*	30 de setembro de 2021	AppMesh
AWS App Runner	`arn:aws:states:::aws-sdk:apprunner:` *`[apiAction]`*	30 de setembro de 2021	AppRunner
AWS AppConfig	`arn:aws:states:::aws-sdk:appconfig:` *`[apiAction]`*	30 de setembro 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 setembro de 2021	AppSync
AWS Application Discovery Service	`arn:aws:states:::aws-sdk:applicationdiscovery:` *`[apiAction]`* [***] —	30 de setembro de 2021	ApplicationDiscovery

Nome do serviço	Recurso do estado da **`Task`**	Data de suporte	Prefixo de exceção
AWS Application Migration Service	`arn:aws:states:::aws-sdk:mgn:` *`[apiAction]`*	30 de setembro de 2021	Mgn
AWS Audit Manager	`arn:aws:states:::aws-sdk:auditmanager:` *`[apiAction]`*	30 de setembro de 2021	AuditManager
AWS Auto Scaling Plans	`arn:aws:states:::aws-sdk:autoscalingplans:` *`[apiAction]`*	30 de setembro de 2021	AutoScalingPlans
AWS Backup	`arn:aws:states:::aws-sdk:backup:` *`[apiAction]`*	30 de setembro de 2021	Backup
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 setembro de 2021	Batch
AWS Billing Conductor	`arn:aws:states:::aws-sdk:billingconductor:` *`[apiAction]`*	26 de julho de 2022	Billingconductor
AWS Budgets	`arn:aws:states:::aws-sdk:budgets:` *`[apiAction]`*	30 de setembro de 2021	Budgets
AWS Certificate Manager	`arn:aws:states:::aws-sdk:acm:` *`[apiAction]`*	30 de setembro de 2021	Acm

Nome do serviço	Recurso do estado da **`Task`**	Data de suporte	Prefixo de exceção
AWS Private Certificate Authority	`arn:aws:states:::aws-sdk:acmpca: [apiAction]`	30 de setembro de 2021	AcmPca
AWS Cloud Map	`arn:aws:states:::aws-sdk:servicediscovery: [apiAction]`	30 de setembro de 2021	ServiceDiscovery
AWS Cloud9	`arn:aws:states:::aws-sdk:cloud9: [apiAction]`	30 de setembro de 2021	Cloud9
AWS CloudFormation	`arn:aws:states:::aws-sdk:cloudformation: [apiAction]`	30 de setembro de 2021	CloudFormation
AWS CloudHSM	`arn:aws:states:::aws-sdk:cloudhsm: [apiAction]`	30 de setembro de 2021	CloudHsm
AWS CloudHSM	`arn:aws:states:::aws-sdk:cloudhsmv2: [apiAction]`	30 de setembro de 2021	CloudHsmV2
AWS CloudTrail	`arn:aws:states:::aws-sdk:cloudtrail: [apiAction]`	30 de setembro 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 setembro de 2021	CodeBuild

Nome do serviço	Recurso do estado da **`Task`**	Data de suporte	Prefixo de exceção
AWS CodeCommit	`arn:aws:states:::aws-sdk:codecommit:` *`[apiAction]`*	30 de setembro de 2021	CodeCommit
AWS CodeDeploy	`arn:aws:states:::aws-sdk:codedeploy:` *`[apiAction]`* ***	30 de setembro de 2021	CodeDeploy
AWS CodePipeline	`arn:aws:states:::aws-sdk:codepipeline:` *`[apiAction]`*	30 de setembro de 2021	CodePipeline
AWS CodeStar	`arn:aws:states:::aws-sdk:codestar:` *`[apiAction]`*	30 de setembro de 2021	CodeStar
AWS CodeStar	`arn:aws:states:::aws-sdk:codestarnotifications:` *`[apiAction]`*	30 de setembro de 2021	CodestarNotifications
AWS CodeStar	`arn:aws:states:::aws-sdk:codestarconnections:` *`[apiAction]`*	30 de setembro de 2021	CodeStarConnections
AWS Compute Optimizer	`arn:aws:states:::aws-sdk:computeoptimizer:` *`[apiAction]`*	30 de setembro de 2021	ComputeOptimizer
AWS Config	`arn:aws:states:::aws-sdk:config:` *`[apiAction]`*	30 de setembro de 2021	Config

Nome do serviço	Recurso do estado da **`Task`**	Data de suporte	Prefixo de exceção
AWS Cost Explorer Service	`arn:aws:states:::aws-sdk:costexplorer:` *`[apiAction]`*	30 de setembro de 2021	CostExplorer
AWS Cost and Usage Report	`arn:aws:states:::aws-sdk:costandusagereport:` *`[apiAction]`*	30 de setembro de 2021	CostAndUsageReport
AWS Data Exchange	`arn:aws:states:::aws-sdk:dataexchange:` *`[apiAction]`*	30 de setembro de 2021	DataExchange
AWS Data Pipeline	`arn:aws:states:::aws-sdk:datapipeline:` *`[apiAction]`*	30 de setembro de 2021	DataPipeline
AWS DataSync	`arn:aws:states:::aws-sdk:datasync:` *`[apiAction]`*	30 de setembro de 2021	DataSync
AWS Database Migration Service	`arn:aws:states:::aws-sdk:databasemigration:` *`[apiAction]`*	30 de setembro de 2021	DatabaseMigration
AWS Device Farm	`arn:aws:states:::aws-sdk:devicefarm:` *`[apiAction]`*	30 de setembro de 2021	DeviceFarm
AWS Direct Connect	`arn:aws:states:::aws-sdk:directconnect:` *`[apiAction]`* ***	30 de setembro de 2021	DirectConnect
AWS Directory Service	`arn:aws:states:::aws-sdk:directory:` *`[apiAction]`*	30 de setembro de 2021	Directory

Nome do serviço	Recurso do estado da **Task**	Data de suporte	Prefixo de exceção
AWS EC2 Instance Connect	`arn:aws:states:::aws-sdk:ec2instanceconnect:` *[apiAction]*	30 de setembro de 2021	Ec2InstanceConnect
AWS Elastic Beanstalk	`arn:aws:states:::aws-sdk:elasticbeanstalk:` *[apiAction]*	30 de setembro de 2021	ElasticBeanstalk
AWS Elemental MediaLive	`arn:aws:states:::aws-sdk:medialive:` *[apiAction]*	30 de setembro de 2021	MediaLive
AWS Elemental MediaPackage	`arn:aws:states:::aws-sdk:mediapackage:` *[apiAction]* *** —	30 de setembro de 2021	MediaPackage
AWS Elemental MediaPackage VOD	`arn:aws:states:::aws-sdk:mediapackagevod:` *[apiAction]*	30 de setembro de 2021	MediaPackageVod
AWS Elemental MediaStore	`arn:aws:states:::aws-sdk:mediastore:` *[apiAction]*	30 de setembro de 2021	MediaStore
AWS Fault Injection Service	`arn:aws:states:::aws-sdk:fis:` *[apiAction]*	30 de setembro de 2021	Fis
AWS Firewall Manager	`arn:aws:states:::aws-sdk:fms:` *[apiAction]*	30 de setembro de 2021	Fms
AWS Glue	`arn:aws:states:::aws-sdk:glue:` *[apiAction]*	30 de setembro de 2021	Glue

Nome do serviço	Recurso do estado da **Task**	Data de suporte	Prefixo de exceção
AWS Glue DataBrew	`arn:aws:states:::aws-sdk:databrew: [apiAction]`	30 de setembro de 2021	DataBrew
AWS IoT Greengrass	`arn:aws:states:::aws-sdk:greengrass: [apiAction]`	30 de setembro de 2021	Greengrass
AWS Ground Station	`arn:aws:states:::aws-sdk:groundstation: [apiAction]`	30 de setembro de 2021	GroundStation
AWS Identity and Access Management	`arn:aws:states:::aws-sdk:iam: [apiAction]`	30 de setembro de 2021	Iam
AWS IoT	`arn:aws:states:::aws-sdk:iot: [apiAction]` ***	30 de setembro de 2021	Iot
AWS IoT 1-Click	`arn:aws:states:::aws-sdk:iot1clickprojects: [apiAction]`	30 de setembro de 2021	Iot1ClickProjects
AWS IoT Analytics	`arn:aws:states:::aws-sdk:iotanalytics: [apiAction]`	30 de setembro de 2021	IoTAnalytics
AWS IoT Core Device Advisor	`arn:aws:states:::aws-sdk:iotdeviceadvisor: [apiAction]` ***	30 de setembro de 2021	IotDeviceAdvisor
AWS IoT Events	`arn:aws:states:::aws-sdk:ioteventsevents: [apiAction]`	30 de setembro de 2021	IotEvents

Nome do serviço	Recurso do estado da **`Task`**	Data de suporte	Prefixo de exceção
Dados do AWS IoT Events	`arn:aws:states:::aws-sdk:ioteventsdata: [apiAction]`	30 de setembro de 2021	IotEventsData
AWS IoT Fleet Hub	`arn:aws:states:::aws-sdk:iotfleethub: [apiAction]`	30 de setembro de 2021	IoTFleetHub
AWS IoT Greengrass Version 2	`arn:aws:states:::aws-sdk:greengrassv2: [apiAction]`	30 de setembro de 2021	GreengrassV2
Dados de trabalhos do AWS IoT Plane	`arn:aws:states:::aws-sdk:iotjobsdataplane: [apiAction]`	30 de setembro de 2021	IotJobsDataPlane
AWS IoT Secure Tunneling	`arn:aws:states:::aws-sdk:iotsecuretunneling: [apiAction]`	30 de setembro de 2021	IoTSecure Tunneling
AWS IoT SiteWise	`arn:aws:states:::aws-sdk:iotsitewise: [apiAction]`	30 de setembro de 2021	IoTSiteWise
AWS IoT Wireless	`arn:aws:states:::aws-sdk:iotwireless: [apiAction]`	30 de setembro de 2021	IotWireless
AWS Key Management Service	`arn:aws:states:::aws-sdk:kms: [apiAction]`	30 de setembro de 2021	Kms
AWS Lake Formation	`arn:aws:states:::aws-sdk:lakeformation: [apiAction]`	30 de setembro de 2021	LakeFormation

Nome do serviço	Recurso do estado da **Task**	Data de suporte	Prefixo de exceção
AWS Lambda	`arn:aws:states:::aws-sdk:lambda: [apiAction]` ***	30 de setembro de 2021	Lambda
AWS License Manager	`arn:aws:states:::aws-sdk:licensemanager: [apiAction]`	30 de setembro de 2021	LicenseManager
AWS Marketplace	`arn:aws:states:::aws-sdk:marketplacecatalog: [apiAction]`	30 de setembro de 2021	MarketplaceCatalog
AWS Marketplace Commerce Analytics	`arn:aws:states:::aws-sdk:marketplacecommerceanalytics: [apiAction]`	30 de setembro de 2021	MarketplaceCommerceAnalytics
AWS Marketplace Entitlement Service	`arn:aws:states:::aws-sdk:marketplaceentitlement: [apiAction]`	30 de setembro de 2021	MarketplaceEntitlement
AWS Elemental MediaTailor	`arn:aws:states:::aws-sdk:mediatailor: [apiAction]`	30 de setembro de 2021	MediaTailor
AWS Migration Hub	`arn:aws:states:::aws-sdk:migrationhub: [apiAction]`	30 de setembro de 2021	MigrationHub
AWS Migration Hub Config	`arn:aws:states:::aws-sdk:migrationhubconfig: [apiAction]`	30 de setembro de 2021	MigrationHubConfig

Nome do serviço	Recurso do estado da **`Task`**	Data de suporte	Prefixo de exceção
Recomendações Estratégicas do 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 setembro de 2021	
AWS Network Firewall	`arn:aws:states:::aws-sdk:networkfirewall:` *`[apiAction]`*	30 de setembro de 2021	NetworkFirewall
AWS OpsWorks	`arn:aws:states:::aws-sdk:opsworks:` *`[apiAction]`*	30 de setembro de 2021	OpsWorks
AWS OpsWorks CM	`arn:aws:states:::aws-sdk:opsworkscm:` *`[apiAction]`*	30 de setembro de 2021	OpsWorksCm
AWS Organizations	`arn:aws:states:::aws-sdk:organizations:` *`[apiAction]`*	30 de setembro de 2021	Organizations
AWS Outposts	`arn:aws:states:::aws-sdk:outposts:` *`[apiAction]`*	30 de setembro de 2021	Outposts
AWS Panorama	`arn:aws:states:::aws-sdk:panorama:` *`[apiAction]`*	19 de abril de 2022	Panorama

Nome do serviço	Recurso do estado da **Task**	Data de suporte	Prefixo de exceção
Amazon Relational Database Service Performance Insights	`arn:aws:states:::aws-sdk:pi:` *[apiAction]*	30 de setembro de 2021	Pi
AWS Price List	`arn:aws:states:::aws-sdk:pricing:` *[apiAction]*	30 de setembro de 2021	Pricing
Amazon Relational Database Service	`arn:aws:states:::aws-sdk:rdsdata:` *[apiAction]* ***	30 de setembro de 2021	RdsData
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 setembro de 2021	Ram
AWS Resource Groups	`arn:aws:states:::aws-sdk:resourcegroups:` *[apiAction]*	30 de setembro de 2021	ResourceGroups
AWS Resource Groups Tagging API	`arn:aws:states:::aws-sdk:resourcegroupstaggingapi:` *[apiAction]*	30 de setembro de 2021	ResourceGroupsTaggingApi
AWS RoboMaker	`arn:aws:states:::aws-sdk:robomaker:` *[apiAction]*	30 de setembro de 2021	RoboMaker

Nome do serviço	Recurso do estado da **Task**	Data de suporte	Prefixo de exceção
AWS IAM Identity Center	`arn:aws:states:::aws-sdk:identitystore: [apiAction]`	30 de setembro de 2021	Identitystore
IAM Identity Center OIDC	`arn:aws:states:::aws-sdk:ssooidc: [apiAction]`	30 de setembro de 2021	SsoOidc
AWS Secrets Manager	`arn:aws:states:::aws-sdk:secretsmanager: [apiAction]`	30 de setembro de 2021	SecretsManager
AWS Security Token Service	`arn:aws:states:::aws-sdk:sts: [apiAction]` *** —	30 de setembro de 2021	Sts
AWS Security Hub	`arn:aws:states:::aws-sdk:securityhub: [apiAction]`	30 de setembro de 2021	SecurityHub
AWS Server Migration Service	`arn:aws:states:::aws-sdk:sms: [apiAction]`	30 de setembro de 2021	Sms
AWS Service Catalog	`arn:aws:states:::aws-sdk:servicecatalog: [apiAction]`	30 de setembro de 2021	ServiceCatalog
AWS Service Catalog AppRegistry	`arn:aws:states:::aws-sdk:servicecatalogappregistry: [apiAction]`	30 de setembro de 2021	ServiceCatalogAppRegistry
AWS Shield	`arn:aws:states:::aws-sdk:shield: [apiAction]` *** —	30 de setembro de 2021	Shield

Nome do serviço	Recurso do estado da **`Task`**	Data de suporte	Prefixo de exceção
AWS Signer	`arn:aws:states:::aws-sdk:signer:` *`[apiAction]`*	30 de setembro de 2021	Signer
IAM Identity Center	`arn:aws:states:::aws-sdk:sso:` *`[apiAction]`*	30 de setembro de 2021	Sso
IAM Identity Center Admin	`arn:aws:states:::aws-sdk:ssoadmin:` *`[apiAction]`*	30 de setembro de 2021	SsoAdmin
AWS Step Functions	`arn:aws:states:::aws-sdk:sfn:` *`[apiAction]`*	30 de setembro de 2021	Sfn
AWS Storage Gateway	`arn:aws:states:::aws-sdk:storagegateway:` *`[apiAction]`*	30 de setembro de 2021	StorageGateway
AWS Support	`arn:aws:states:::aws-sdk:support:` *`[apiAction]`*	30 de setembro 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 setembro de 2021	Transfer
AWS WAF	`arn:aws:states:::aws-sdk:waf:` *`[apiAction]`*	30 de setembro de 2021	Waf

Nome do serviço	Recurso do estado da **`Task`**	Data de suporte	Prefixo de exceção
AWS WAF Regional	`arn:aws:states:::aws-sdk:wafregional: [apiAction]`	30 de setembro de 2021	WafRegional
AWS WAFV2	`arn:aws:states:::aws-sdk:wafv2: [apiAction]`	30 de setembro de 2021	Wafv2
AWS Well-Architected Tool	`arn:aws:states:::aws-sdk:wellarchitected: [apiAction]`	30 de setembro de 2021	WellArchitected
AWS X-Ray	`arn:aws:states:::aws-sdk:xray: [apiAction]`	30 de setembro de 2021	XRay
AWS Marketplace Metering Service	`arn:aws:states:::aws-sdk:marketplacemetering: [apiAction]`	30 de setembro de 2021	MarketplaceMetering
AWS Serverless Application Repository	`arn:aws:states:::aws-sdk:serverlessapplicationrepository: [apiAction]`	30 de setembro de 2021	ServerlessApplicationRepository
AWS Identity and Access Management Access Analyzer	`arn:aws:states:::aws-sdk:accessanalyzer: [apiAction]`	30 de setembro de 2021	AccessAnalyzer
Alexa for Business	`arn:aws:states:::aws-sdk:alexaforbusiness: [apiAction]`	30 de setembro de 2021	AlexaForBusiness

Nome do serviço	Recurso do estado da **`Task`**	Data de suporte	Prefixo de exceção
Amazon API Gateway	`arn:aws:states:::aws-sdk:apigateway:` *`[apiAction]`*	30 de setembro de 2021	ApiGateway
Amazon API Gateway	`arn:aws:states:::aws-sdk:apigatewayv2:` *`[apiAction]`*	30 de setembro de 2021	ApiGatewayV2
Amazon AppIntegrations	`arn:aws:states:::aws-sdk:appintegrations:` *`[apiAction]`*	30 de setembro de 2021	AppIntegrations
Amazon AppStream 2.0	`arn:aws:states:::aws-sdk:appstream:` *`[apiAction]`*	30 de setembro de 2021	AppStream
Amazon AppFlow	`arn:aws:states:::aws-sdk:appflow:` *`[apiAction]`*	30 de setembro de 2021	Appflow
Amazon Athena	`arn:aws:states:::aws-sdk:athena:` *`[apiAction]`*	30 de setembro de 2021	Athena
Amazon Augmented AI	`arn:aws:states:::aws-sdk:sagemakera2iruntime:` *`[apiAction]`*	30 de setembro de 2021	SageMaker A2IRuntime
Amazon Braket	`arn:aws:states:::aws-sdk:braket:` *`[apiAction]`*	30 de setembro de 2021	Braket
Amazon Chime	`arn:aws:states:::aws-sdk:chime:` *`[apiAction]`*	30 de setembro de 2021	Chime

Nome do serviço	Recurso do estado da **`Task`**	Data de suporte	Prefixo de exceção
Amazon Chime Meetings	`arn:aws:states:::aws-sdk:chimesdkmeetings: [apiAction]`	19 de abril de 2022	ChimeSdkMeetings
Amazon Cloud Directory	`arn:aws:states:::aws-sdk:clouddirectory: [apiAction]`	30 de setembro de 2021	CloudDirectory
Amazon CloudFront	`arn:aws:states:::aws-sdk:cloudfront: [apiAction]`	30 de setembro de 2021	CloudFront
Amazon CloudSearch	`arn:aws:states:::aws-sdk:cloudsearch: [apiAction]`	30 de setembro de 2021	CloudSearch
Amazon CloudWatch	`arn:aws:states:::aws-sdk:cloudwatch: [apiAction]`	30 de setembro de 2021	CloudWatch
Amazon CloudWatch Application Insights	`arn:aws:states:::aws-sdk:applicationinsights: [apiAction]`	30 de setembro 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 setembro de 2021	CloudWatchLogs
Amazon CloudWatch RUM	`arn:aws:states:::aws-sdk:rum: [apiAction]`	19 de abril de 2022	Rum

Nome do serviço	Recurso do estado da **`Task`**	Data de suporte	Prefixo de exceção
Amazon CloudWatch Synthetics	`arn:aws:states:::aws-sdk:synthetics:` *`[apiAction]`*	30 de setembro de 2021	Synthetics
Amazon CodeGuru Profiler	`arn:aws:states:::aws-sdk:codeguruprofiler:` *`[apiAction]`*	30 de setembro de 2021	CodeGuruProfiler
Amazon CodeGuru Reviewer	`arn:aws:states:::aws-sdk:codegurureviewer:` *`[apiAction]`*	30 de setembro de 2021	CodeGuruReviewer
Amazon Cognito	`arn:aws:states:::aws-sdk:cognitoidentity:` *`[apiAction]`*	30 de setembro de 2021	CognitoIdentity
Amazon Cognito Identity Provider	`arn:aws:states:::aws-sdk:cognitoidentityprovider:` *`[apiAction]`*	30 de setembro de 2021	CognitoIdentityProvider
Amazon Cognito Sync	`arn:aws:states:::aws-sdk:cognitosync:` *`[apiAction]`*	30 de setembro de 2021	CognitoSync
Amazon Comprehend	`arn:aws:states:::aws-sdk:comprehend:` *`[apiAction]`*	30 de setembro de 2021	Comprehend
Amazon Comprehend Medical	`arn:aws:states:::aws-sdk:comprehendmedical:` *`[apiAction]`* ***	30 de setembro de 2021	ComprehendMedical
Amazon Connect Contact Lens	`arn:aws:states:::aws-sdk:connectcontactlens:` *`[apiAction]`*	30 de setembro de 2021	ConnectContactLens

Nome do serviço	Recurso do estado da **`Task`**	Data de suporte	Prefixo de exceção
Amazon Connect Participant Service	`arn:aws:states:::aws-sdk:connectparticipant: [apiAction]`	30 de setembro de 2021	ConnectParticipant
Amazon Connect	`arn:aws:states:::aws-sdk:connect: [apiAction]`	30 de setembro 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 setembro de 2021	Dlm
Amazon Detective	`arn:aws:states:::aws-sdk:detective: [apiAction]`	30 de setembro de 2021	Detective
Amazon DevOps Guru	`arn:aws:states:::aws-sdk:devopsguru: [apiAction]`	30 de setembro de 2021	DevOpsGuru
Amazon DocumentDB (with MongoDB compatibility)	`arn:aws:states:::aws-sdk:docdb: [apiAction]`	30 de setembro de 2021	DocDb
Amazon DynamoDB	`arn:aws:states:::aws-sdk:dynamodb: [apiAction]`	30 de setembro de 2021	DynamoDb

Nome do serviço	Recurso do estado da **`Task`**	Data de suporte	Prefixo de exceção
Amazon DynamoDB Streams	`arn:aws:states:::aws-sdk:dynamodbstreams: [apiAction]`	30 de setembro de 2021	DynamoDbStreams
Amazon EC2 Container Registry	`arn:aws:states:::aws-sdk:ecr: [apiAction]`	30 de setembro de 2021	Ecr
Amazon EC2 Container Service	`arn:aws:states:::aws-sdk:ecs: [apiAction]`	30 de setembro de 2021	Ecs
Amazon EC2 Systems Manager	`arn:aws:states:::aws-sdk:ssm: [apiAction]`	30 de setembro de 2021	Ssm
Amazon EMR	`arn:aws:states:::aws-sdk:emrcontainers: [apiAction]` ***	30 de setembro de 2021	EmrContainers
Amazon ElastiCache	`arn:aws:states:::aws-sdk:elasticache: [apiAction]`	30 de setembro de 2021	ElastiCache
Amazon Elastic Inference	`arn:aws:states:::aws-sdk:elasticinference: [apiAction]`	30 de setembro de 2021	ElasticInference
Amazon Elastic Block Store	`arn:aws:states:::aws-sdk:ebs: [apiAction]`	30 de setembro de 2021	Ebs
Amazon Elastic Compute Cloud	`arn:aws:states:::aws-sdk:ec2: [apiAction]`	30 de setembro de 2021	Ec2

Nome do serviço	Recurso do estado da **`Task`**	Data de suporte	Prefixo de exceção
Amazon Elastic Container Registry Public	`arn:aws:states:::aws-sdk:ecrpublic: [apiAction]`	30 de setembro de 2021	EcrPublic
Amazon Elastic File System	`arn:aws:states:::aws-sdk:efs: [apiAction]` ***	30 de setembro de 2021	Efs
Amazon Elastic Kubernetes Service	`arn:aws:states:::aws-sdk:eks: [apiAction]`	30 de setembro de 2021	Eks
Amazon EMR	`arn:aws:states:::aws-sdk:emr: [apiAction]`	30 de setembro de 2021	Emr
Amazon Elastic Transcoder	`arn:aws:states:::aws-sdk:elastictranscoder: [apiAction]` ***	30 de setembro de 2021	ElasticTranscoder
Amazon OpenSearch Service	`arn:aws:states:::aws-sdk:elasticsearch: [apiAction]`	30 de setembro de 2021	Elasticsearch
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 setembro de 2021	EventBridge
Amazon FSx	`arn:aws:states:::aws-sdk:fsx: [apiAction]`	30 de setembro de 2021	FSx

Nome do serviço	Recurso do estado da **`Task`**	Data de suporte	Prefixo de exceção
Amazon Forecast Query	`arn:aws:states:::aws-sdk:forecastquery:` *`[apiAction]`*	30 de setembro de 2021	Forecastquery
Amazon Forecast Service	`arn:aws:states:::aws-sdk:forecast:` *`[apiAction]`*	30 de setembro de 2021	Forecast
Amazon Fraud Detector	`arn:aws:states:::aws-sdk:frauddetector:` *`[apiAction]`*	30 de setembro de 2021	FraudDetector
Amazon GameLift	`arn:aws:states:::aws-sdk:gamelift:` *`[apiAction]`*	30 de setembro de 2021	Amazon GameLift
Amazon GameSparks	`arn:aws:states:::aws-sdk:gamesparks:` *`[apiAction]`*	27 de julho de 2022	GameSparks
Amazon S3 Glacier	`arn:aws:states:::aws-sdk:glacier:` *`[apiAction]`*	30 de setembro de 2021	Glacier
Amazon GuardDuty	`arn:aws:states:::aws-sdk:guardduty:` *`[apiAction]`*	30 de setembro de 2021	GuardDuty
AWS HealthLake	`arn:aws:states:::aws-sdk:healthlake:` *`[apiAction]`*	30 de setembro de 2021	HealthLake
Amazon Honeycode	`arn:aws:states:::aws-sdk:honeycode:` *`[apiAction]`*	30 de setembro de 2021	Honeycode

Nome do serviço	Recurso do estado da **`Task`**	Data de suporte	Prefixo de exceção
Amazon Inspector	`arn:aws:states:::aws-sdk:inspector: [apiAction]`	30 de setembro 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 setembro de 2021	Ivs
Amazon Kendra	`arn:aws:states:::aws-sdk:kendra: [apiAction]`	30 de setembro de 2021	Kendra
Amazon Kinesis	`arn:aws:states:::aws-sdk:kinesis: [apiAction]` ***	30 de setembro de 2021	Kinesis
Amazon Kinesis Analytics	`arn:aws:states:::aws-sdk:kinesisanalytics: [apiAction]`	30 de setembro de 2021	KinesisAnalytics
Amazon Kinesis Analytics V2	`arn:aws:states:::aws-sdk:kinesisanalyticsv2: [apiAction]`	30 de setembro de 2021	KinesisAnalyticsV2
Amazon Kinesis Firehose	`arn:aws:states:::aws-sdk:firehose: [apiAction]`	30 de setembro de 2021	Firehose
Amazon Kinesis Video Signaling Channels	`arn:aws:states:::aws-sdk:kinesisvideosignaling: [apiAction]`	30 de setembro de 2021	KinesisVideoSignaling

Nome do serviço	Recurso do estado da **`Task`**	Data de suporte	Prefixo de exceção
Amazon Kinesis Video Streams	`arn:aws:states:::aws-sdk:kinesisvideo: [apiAction]`	30 de setembro de 2021	KinesisVideo
Amazon Kinesis Video Streams Archived Media	`arn:aws:states:::aws-sdk:kinesisvideoarchivedmedia: [apiAction]`	30 de setembro de 2021	KinesisVideoArchivedMedia
Amazon Kinesis video stream	`arn:aws:states:::aws-sdk:kinesisvideomedia: [apiAction]`	30 de setembro de 2021	KinesisVideoMedia
Amazon Lex Model Building Service	`arn:aws:states:::aws-sdk:lexmodelbuilding: [apiAction]`	30 de setembro de 2021	LexModelBuilding
Amazon Lex Model Building Service V2	`arn:aws:states:::aws-sdk:lexmodelsv2: [apiAction]`	30 de setembro de 2021	LexModelsV2
Amazon Lex	`arn:aws:states:::aws-sdk:lexruntime: [apiAction]`	30 de setembro de 2021	LexRuntime
Amazon Lex Runtime V2	`arn:aws:states:::aws-sdk:lexruntimev2: [apiAction]` ***	30 de setembro de 2021	LexRuntimeV2
Amazon Lightsail	`arn:aws:states:::aws-sdk:lightsail: [apiAction]`	30 de setembro de 2021	Lightsail
Amazon Location Service	`arn:aws:states:::aws-sdk:location: [apiAction]`	30 de setembro de 2021	Location

Nome do serviço	Recurso do estado da **`Task`**	Data de suporte	Prefixo de exceção
Amazon Lookout for Equipment	`arn:aws::states:::aws-sdk:lookoutequipment: [apiAction]`	30 de setembro de 2021	LookoutEquipment
Amazon Lookout for Metrics	`arn:aws:states:::aws-sdk:lookoutmetrics: [apiAction]`	30 de setembro de 2021	LookoutMetrics
Amazon Lookout for Vision	`arn:aws:states:::aws-sdk:lookoutvision: [apiAction]`	30 de setembro de 2021	LookoutVision
Amazon MQ	`arn:aws:states:::aws-sdk:mq: [apiAction]`	30 de setembro de 2021	Mq
Amazon Macie	`arn:aws:states:::aws-sdk:macie: [apiAction]`	30 de setembro de 2021	
Amazon Macie 2	`arn:aws:states:::aws-sdk:macie2: [apiAction]`	30 de setembro de 2021	Macie2
Amazon Managed Blockchain	`arn:aws:states:::aws-sdk:managedblockchain: [apiAction]`	30 de setembro 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 setembro de 2021	Amp

Nome do serviço	Recurso do estado da **`Task`**	Data de suporte	Prefixo de exceção
Amazon Managed Streaming for Apache Kafka	`arn:aws:states:::aws-sdk:kafka:` *`[apiAction]`*	30 de setembro 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 setembro de 2021	Mwaa
Amazon Mechanical Turk	`arn:aws:states:::aws-sdk:mturk:` *`[apiAction]`*	30 de setembro de 2021	MTurk
Amazon MemoryDB for Redis	`arn:aws:states:::aws-sdk:memorydb:` *`[apiAction]`*	19 de abril de 2022	MemoryDB
Amazon Nimble Studio	`arn:aws:states:::aws-sdk:nimble:` *`[apiAction]`*	30 de setembro de 2021	Nimble
Amazon Personalize	`arn:aws:states:::aws-sdk:personalize:` *`[apiAction]`*	30 de setembro de 2021	Personalize
Amazon Personalize Events	`arn:aws:states:::aws-sdk:personalizeevents:` *`[apiAction]`*	30 de setembro de 2021	PersonalizeEvents
Amazon Personalize Runtime	`arn:aws:states:::aws-sdk:personalizeruntime:` *`[apiAction]`*	30 de setembro de 2021	PersonalizeRuntime

Nome do serviço	Recurso do estado da **`Task`**	Data de suporte	Prefixo de exceção
Amazon Pinpoint	`arn:aws:states:::aws-sdk:pinpoint: [apiAction]`	30 de setembro de 2021	Pinpoint
Amazon Pinpoint Email Service	`arn:aws:states:::aws-sdk:pinpointemail: [apiAction]`	30 de setembro de 2021	PinpointEmail
Amazon Pinpoint SMS and Voice Service	`arn:aws:states:::aws-sdk:pinpointsmsvoice: [apiAction]`	30 de setembro de 2021	PinpointSmsVoice
Amazon Pinpoint SMS and Voice V2 Service	`arn:aws:states:::aws-sdk:pinpointsmsvoicev2: [apiAction]`	27 de julho de 2022	PinpointSmsVoiceV2
Amazon Polly	`arn:aws:states:::aws-sdk:polly: [apiAction]`	30 de setembro de 2021	Polly
Amazon QLDB	`arn:aws:states:::aws-sdk:qldb: [apiAction]`	30 de setembro de 2021	Qldb
Amazon QLDB Session	`arn:aws:states:::aws-sdk:qldbsession: [apiAction]`	30 de setembro de 2021	QldbSession
Amazon QuickSight	`arn:aws:states:::aws-sdk:quicksight: [apiAction]`	30 de setembro de 2021	QuickSight
Amazon Redshift	`arn:aws:states:::aws-sdk:redshift: [apiAction]`	30 de setembro de 2021	Redshift

Nome do serviço	Recurso do estado da **`Task`**	Data de suporte	Prefixo de exceção
Amazon Redshift Data API	`arn:aws:states:::aws-sdk:redshiftdata: [apiAction]`	30 de setembro de 2021	RedshiftData
Amazon Rekognition	`arn:aws:states:::aws-sdk:rekognition: [apiAction]`	30 de setembro de 2021	Rekognition
Amazon Relational Database Service	`arn:aws:states:::aws-sdk:rds: [apiAction]`	30 de setembro de 2021	Rds
Amazon Route 53	`arn:aws:states:::aws-sdk:route53: [apiAction]`	30 de setembro de 2021	Route53
Amazon Route 53 Recovery Control Config	`arn:aws:states:::aws-sdk:route53recoverycontrolconfig: [apiAction]`	30 de setembro de 2021	Route53RecoveryControlConfig
Amazon Route 53 Domains	`arn:aws:states:::aws-sdk:route53domains: [apiAction]`	30 de setembro de 2021	Route53Domains
Amazon Route 53 Resolver	`arn:aws:states:::aws-sdk:route53resolver: [apiAction]`	30 de setembro de 2021	Route53Resolver
Amazon S3 on Outposts	`arn:aws:states:::aws-sdk:s3outposts: [apiAction]`	30 de setembro de 2021	S3Outposts

Nome do serviço	Recurso do estado da **`Task`**	Data de suporte	Prefixo de exceção
Amazon SageMaker Runtime Feature Store Runtime	`arn:aws:states:::aws-sdk:sagemakerfeaturestoreruntime:` *`[apiAction]`*	30 de setembro de 2021	SageMakerRuntimeFeatureStoreRuntime
Amazon SageMaker Runtime Runtime	`arn:aws:states:::aws-sdk:sagemakerruntime:` *`[apiAction]`*	30 de setembro de 2021	SageMakerRuntimeRuntime
Amazon SageMaker	`arn:aws:states:::aws-sdk:sagemaker:` *`[apiAction]`*	30 de setembro de 2021	SageMakerRuntime
Amazon SageMaker Edge Manager	`arn:aws:states:::aws-sdk:sagemakeredge:` *`[apiAction]`*	30 de setembro de 2021	SagemakerEdge
Amazon Simple Email Service	`arn:aws:states:::aws-sdk:ses:` *`[apiAction]`*	30 de setembro de 2021	Ses
Amazon Simple Email Service V2	`arn:aws:states:::aws-sdk:sesv2:` *`[apiAction]`*	30 de setembro de 2021	SesV2
Amazon Simple Notification Service	`arn:aws:states:::aws-sdk:sns:` *`[apiAction]`*	30 de setembro de 2021	Sns
Amazon Simple Queue Service	`arn:aws:states:::aws-sdk:sqs:` *`[apiAction]`*	30 de setembro de 2021	Sqs
Amazon Simple Storage Service	`arn:aws:states:::aws-sdk:s3:` *`[apiAction]`* ***	30 de setembro de 2021	S3

Nome do serviço	Recurso do estado da **`Task`**	Data de suporte	Prefixo de exceção
Amazon Simple Workflow Service	`arn:aws:states:::aws-sdk:swf:` *`[apiAction]`*	30 de setembro de 2021	Swf
Amazon Textract	`arn:aws:states:::aws-sdk:textract:` *`[apiAction]`*	30 de setembro de 2021	Textract
Amazon Transcribe	`arn:aws:states:::aws-sdk:transcribe:` *`[apiAction]`*	30 de setembro de 2021	Transcribe
Amazon Translate	`arn:aws:states:::aws-sdk:translate:` *`[apiAction]`*	30 de setembro de 2021	Translate
Amazon WorkDocs	`arn:aws:states:::aws-sdk:workdocs:` *`[apiAction]`*	30 de setembro de 2021	WorkDocs
Amazon WorkMail	`arn:aws:states:::aws-sdk:workmail:` *`[apiAction]`*	30 de setembro de 2021	WorkMail
Amazon WorkMail Message Flow	`arn:aws:states:::aws-sdk:workmailmessageflow:` *`[apiAction]`*	30 de setembro de 2021	WorkMailMessageFlow
Amazon WorkSpaces	`arn:aws:states:::aws-sdk:workspaces:` *`[apiAction]`*	30 de setembro de 2021	WorkSpaces
Amazon WorkSpaces Web	`arn:aws:states:::aws-sdk:workspacesweb:` *`[apiAction]`*	19 de abril de 2022	WorkSpacesWeb

Nome do serviço	Recurso do estado da **`Task`**	Data de suporte	Prefixo de exceção
Amplify	`arn:aws:states:::aws-sdk:amplifybackend: [apiAction]`	30 de setembro 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 setembro de 2021	ApplicationAutoScaling
Amazon EC2 Auto Scaling	`arn:aws:states:::aws-sdk:autoscaling: [apiAction]`	30 de setembro de 2021	Auto Scaling
CodeArtifact	`arn:aws:states:::aws-sdk:codeartifact: [apiAction]`	30 de setembro de 2021	Codeartifact
DynamoDB Accelerator	`arn:aws:states:::aws-sdk:dax: [apiAction]`	30 de setembro de 2021	Dax
EC2 Image Builder	`arn:aws:states:::aws-sdk:imagebuilder: [apiAction]`	30 de setembro de 2021	Imagebuilder
AWS Elastic Disaster Recovery	`arn:aws:states:::aws-sdk:drs: [apiAction]`	19 de abril de 2022	Drs

Nome do serviço	Recurso do estado da **`Task`**	Data de suporte	Prefixo de exceção
Elastic Load Balancing	`arn:aws:states:::aws-sdk:elasticloadbalancing:` *`[apiAction]`*	30 de setembro de 2021	ElasticLoadBalancing
Elastic Load Balancing V2	`arn:aws:states:::aws-sdk:elasticloadbalancingv2:` *`[apiAction]`*	30 de setembro de 2021	ElasticLoadBalancingV2
MediaConnect	`arn:aws:states:::aws-sdk:mediaconnect:` *`[apiAction]`*	30 de setembro de 2021	MediaConnect
Amazon S3 Control	`arn:aws:states:::aws-sdk:s3control:` *`[apiAction]`* ***	30 de setembro 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 setembro de 2021	Savingsplans
Amazon EventBridge Schema Registry	`arn:aws:states:::aws-sdk:schemas:` *`[apiAction]`*	30 de setembro de 2021	Schemas
Service Quotas	`arn:aws:states:::aws-sdk:servicequotas:` *`[apiAction]`*	30 de setembro de 2021	ServiceQuotas

Nome do serviço	Recurso do estado da **`Task`**	Data de suporte	Prefixo de exceção
AWS Snowball	`arn:aws:states:::aws-sdk:snowball:` *`[apiAction]`*	30 de setembro de 2021	Snowball

Ações de API não compatíveis para serviços compatíveis

A tabela a seguir lista as ações de API não suportadas para integrações de serviços AWS do SDK. A coluna direita contém as ações de API que atualmente não são compatíveis com o serviço listado na coluna da esquerda.

Nome do serviço	Ação da API não compatível
AWS Application Discovery Service	• `DescribeExportConfigurations` • `ExportConfigurations`
Amazon Bedrock	• `InvokeModelWithResponseStream`
Agentes do Amazon Bedrock Runtime	• `InvokeAgent`
AWS CodeDeploy	• `BatchGetDeploymentInstances` • `GetDeploymentInstance` • `ListDeploymentInstances` • `SkipWaitTimeForInstanceTermination`
Amazon Comprehend Medical	• `DetectEntities`
AWS Direct Connect	• `AllocateConnectionOnInterconnect` • `DescribeConnectionLoa` • `DescribeConnectionsOnInterconnect` • `DescribeInterconnectLoa`

Nome do serviço	Ação da API não compatível
Amazon Elastic File System	• `CreateTags`
Amazon Elastic Transcoder	• `TestRole`
Amazon EMR	• `DescribeJobFlows`
AWS IoT	• `AttachPrincipalPolicy` • `ListPrincipalPolicies` • `DetachPrincipalPolicy` • `ListPolicyPrincipals` • `DetachPrincipalPolicy`
AWS IoT Conselheiro de dispositivos principais	• `ListTestCases`
Amazon Kinesis	• `SubscribeToShard`
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`

Integrações de serviços SDK obsoletas AWS

As seguintes integrações de serviços do AWS SDK agora estão obsoletas:

- AWS Mobile
- Amazon Macie
- AWS IoT RoboRunner

Integrações otimizadas para o Step Functions

Os tópicos a seguir incluem as APIs, os parâmetros e a sintaxe de solicitação/resposta compatíveis no Amazon States Language para coordenar outros serviços. AWS Os tópicos também fornecem exemplos de código. Você pode chamar os serviços de integrações otimizadas diretamente da Amazon States Language no campo `Resource` de um estado `Task`.

Você pode usar três padrões de integração de serviços:

- Solicitar uma resposta (padrão) - aguarde a resposta HTTP e vá para o próximo estado
- Run a Job (`.sync`) - aguarde a conclusão do trabalho
- Aguarde o retorno de chamada (`.waitForTaskToken`) - pause um fluxo de trabalho até que um token de tarefa seja retornado

Os fluxos de trabalho padrão e os fluxos de trabalho expressos oferecem suporte às mesmas integrações, mas não aos mesmos padrões de integração.

- O suporte ao padrão de integrações otimizadas é diferente para cada integração.
- Os fluxos de trabalho expressos não oferecem suporte a Run a Job (.sync) nem a Wait for Callback (). waitForTaskSímbolo).
- Para ter mais informações, consulte Comparação entre os fluxos de trabalho padrão e expresso.

Standard Workflows

Integrações de serviços compatíveis

	Serviço	Resposta de solicitação	Executar um trabalho (`.sync`)	Aguardar o retorno de chamada (`.waitForTaskToken`)
Integrações otimizadas	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	✓		✓
	AWS Elemental MediaConvert	✓	✓	

	Serviço	Resposta de solicitação	Executar um trabalho (`.sync`)	Aguardar o retorno de chamada (`.waitForTaskToken`)
	Amazon SageMaker	✓	✓	
	Amazon SNS	✓		✓
	Amazon SQS	✓		✓
	AWS Step Functions	✓	✓	✓
AWS Integrações de SDK	Mais de duzentas	✓		✓

Express Workflows

Integrações de serviços compatíveis

	Serviço	Resposta de solicitação	Executar um trabalho (`.sync`)	Aguardar o retorno de chamada (`.waitForTaskToken`)
Integrações otimizadas	Amazon API Gateway	✓		
	Amazon Athena	✓		
	AWS Batch	✓		
	Amazon Bedrock	✓		

	Serviço	Resposta de solicitação	Executar um trabalho (`.sync`)	Aguardar o retorno de chamada (`.waitForTaskToken`)
	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	✓		
	AWS Step Functions	✓		

	Serviço	Resposta de solicitação	Executar um trabalho (`.sync`)	Aguardar o retorno de chamada (`.waitForTaskToken`)
AWS Integrações de SDK	Mais de duzentas	✓		

Chame o API Gateway com o Step Functions

Step Functions pode controlar determinados AWS serviços diretamente do Amazon States Language (ASL). Para saber mais, consulte Como trabalhar com outros serviços e Transmitir parâmetros para uma API de serviço.

> Como a integração do API Gateway otimizado é diferente da integração do AWS SDK do API Gateway
>
> - `apigateway:invoke:`não tem equivalente na integração de serviços do AWS SDK. Em vez disso, o serviço Optimized API Gateway chama o endpoint do API Gateway diretamente.

É possível usar o Amazon API Gateway para criar, publicar, manter e monitorar APIs de HTTP ou REST. Para fazer a integração com o API Gateway, você define um estado `Task` no Step Functions que chama diretamente um endpoint de HTTP do API Gateway ou de REST do API Gateway, sem escrever código ou depender de outra infraestrutura. Uma definição de estado `Task` inclui todas as informações necessárias para a chamada da API. Também é possível selecionar métodos de autorização diferentes.

Suporte ao recurso API Gateway

A integração do API Gateway do Step Functions é compatível com alguns, mas não todos os recursos do API Gateway. Para obter uma lista mais detalhada dos recursos compatíveis, consulte o seguinte:

- Compatível com as integrações da API REST do API Gateway e do API HTTP do API Gateway do Step Functions:
  - Autorizadores: IAM (usando Singature Version 4), No Auth, Lambda Authorizers (baseados em parâmetros de solicitação e baseados em tokens com cabeçalho personalizado)
  - Tipos de API: regional
  - Gerenciamento de API: nomes de domínio de API do API Gateway, estágio da API, caminho, parâmetros de consulta, corpo da solicitação
- Compatível com a integração da API HTTP do API Gateway do Step Functions. A integração da API REST do API Gateway do Step Functions, que fornece a opção de APIs otimizadas para o Edge, não é compatível.
- Não compatível com a integração do API Gateway do Step Functions:
  - Autorizadores: Amazon Cognito, Native Open ID Connect/OAuth 2.0, cabeçalho de autorização para autorizadores Lambda baseados em tokens
  - Tipos de API: privado
  - Gerenciamento de API: nomes de domínio personalizados

Para obter mais informações sobre API Gateway e as APIs HTTP e REST, consulte o seguinte:

- A página Conceitos do Amazon API Gateway.
- Como escolher entre APIs HTTP e REST no Guia do desenvolvedor do API Gateway.

Formato de solicitação

Quando você cria a definição de estado `Task`, o Step Functions valida os parâmetros, cria a URL necessária para realizar a chamada e, em seguida, chama a API. A resposta inclui o código de status HTTP, cabeçalhos e corpo da resposta. O formato da solicitação tem parâmetros obrigatórios e opcionais.

Parâmetros de solicitação obrigatórios

- `ApiEndpoint`
  - Tipo: `String`
  - O nome do host de um URL do API Gateway. O formato é *`<API ID>`*`.execute-api.`*`<region>`*`.amazonaws.com`.

    O ID da API só pode conter uma combinação dos seguintes caracteres alfanuméricos: `0123456789abcdefghijklmnopqrstuvwxyz`
- `Method`
  - Tipo: `Enum`
  - O método HTTP, que deve ser um dos seguintes:
    - `GET`
    - `POST`
    - `PUT`
    - `DELETE`
    - `PATCH`
    - `HEAD`
    - `OPTIONS`

Parâmetros de solicitação opcionais

- `Headers`
  - Tipo: `JSON`
  - Os cabeçalhos HTTP permitem uma lista de valores associados à mesma chave.
- `Stage`
  - Tipo: `String`
  - O nome do estágio em que a API é implantada no API Gateway. É opcional para qualquer API HTTP que use o estágio `$default`.
- `Path`
  - Tipo: `String`
  - Parâmetros de caminho que são anexados após o endpoint da API.
- `QueryParameters`

  - Tipo: `JSON`
  - As strings de consulta só permitem uma lista de valores associados à mesma chave.
- `RequestBody`
  - Tipo: `JSON` ou `String`
  - O corpo da solicitação HTTP. O tipo pode ser um objeto `JSON` ou `String`. `RequestBody` só é compatível com os métodos de HTTP `PATCH`, `POST` e `PUT`.
- `AllowNullValues`
  - Tipo: `BOOLEAN` — valor padrão: `false`
  - Com a configuração padrão, quaisquer valores nulos no estado de entrada da solicitação não serão enviados para sua API. No exemplo a seguir, o `category` campo não será incluído na solicitação, a menos que `AllowNullValues` esteja definido `true` na definição da sua máquina de estado.

```
{
    "NewPet": {
        "type": "turtle",
        "price": 123,
        "category": null
    }
}
```

> **Note**
>
> Por padrão, campos com valores nulos no estado de entrada da solicitação não serão enviados para sua API. Você pode forçar o envio de valores nulos para sua API `AllowNullValues` configurando como `true` na definição da sua máquina de estado.

- `AuthType`
  - Tipo: `JSON`
  - O método de autenticação. O método padrão é `NO_AUTH`. Os valores permitidos são:
    - `NO_AUTH`
    - `IAM_ROLE`
    - `RESOURCE_POLICY`

    Consulte Autenticação e autorização para obter mais informações.

> **Note**
>
> Por questões de segurança, as seguintes chaves de cabeçalho HTTP não são permitidas atualmente:
>
> - Qualquer coisa prefixada com `X-Forwarded`, `X-Amz` ou `X-Amzn`.
> - `Authorization`
> - `Connection`
> - `Content-md5`
> - `Expect`
> - `Host`
> - `Max-Forwards`
> - `Proxy-Authenticate`
> - `Server`
> - `TE`
> - `Transfer-Encoding`
> - `Trailer`
> - `Upgrade`
> - `Via`
> - `Www-Authenticate`

O exemplo de código a seguir mostra como invocar o API Gateway usando o 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"
    }
}
```

Autenticação e autorização

Você pode usar os seguintes métodos de autenticação:

- Sem autorização: chame a API diretamente sem nenhum método de autorização.
- Perfil do IAM: com esse método, o Step Functions assume o papel da máquina de estado, assina a solicitação com Signature Version 4 (SigV4) e chama a API.
- Política de recursos: o Step Functions autentica a solicitação e, em seguida, chama a API. Você deve anexar uma política de recursos à API que especifique o seguinte:
  1. A máquina de estado que invocará o API Gateway.

     > **Important**
     >
     > Você deve especificar a máquina de estado para limitar o acesso a ela. Caso contrário, qualquer máquina de estado que autentique a solicitação do API Gateway com a autenticação Política de recursos na API receberá acesso.

  2. Esse Step Functions é o serviço que chama o API Gateway: `"Service": "states.amazonaws.com"`.
  3. O recurso que você deseja acessar, incluindo:
     - A *região*.
     - O *ID da conta* na região especificada.
     - O *ID da API*.
     - O *nome do estágio*.
     - O *HTTP-VERB* (método).
     - O *especificador do caminho do recurso*.

  Para ver um exemplo de política de recursos, consulte Políticas do IAM para Step Functions e API Gateway.

Para obter mais informações sobre o formato do recurso, consulte Formato do recurso de permissões para executar a API no API Gateway no Guia do desenvolvedor do API Gateway.

> **Note**
>
> As políticas de recursos são compatíveis somente com a API REST.

Padrões de integração de serviço

A integração do API Gateway oferece suporte a dois padrões de integração de serviços:

- Resposta de solicitação, que é o padrão de integração padrão. Ele permite que o Step Functions avance para a próxima etapa imediatamente após receber uma resposta HTTP.
- Aguardar um retorno de chamada com um token de tarefa (`.waitForTaskToken`), que espera até que um token de tarefa seja retornado com uma carga útil. Para usar o `.waitForTaskToken` padrão, anexe .wait ForTaskToken ao final do campo Recurso da definição de sua tarefa, conforme mostrado no exemplo a seguir:

```
{
    "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 saída

Os seguintes parâmetros de saída são fornecidos:

Nome	Tipo	Description
`ResponseBody`	`JSON` ou `String`	O corpo da resposta da chamada de API.
`Headers`	`JSON`	Os cabeçalhos de resposta.
`StatusCode`	`Integer`	O código de status HTTP da resposta.
`StatusText`	`String`	O texto do status da resposta.

Um exemplo de resposta:

```
{
    "ResponseBody": {
        "myBills": []
    },
    "Headers": {
        "key": ["value1", "value2"]
    },
    "StatusCode": 200,
    "StatusText": "OK"
}
```

Tratamento de erros

Quando ocorre um erro, `error` e `cause` são retornados da seguinte forma:

- Se o código de status HTTP estiver disponível, o erro será retornado no formato `ApiGateway.<HTTP Status Code>`.
- Se o código de status HTTP estiver indisponível, o erro será retornado no formato `ApiGateway.<Exception>`.

Em ambos os casos, `cause` é retornado como uma string.

O seguinte exemplo mostra uma resposta em que ocorreu um erro:

```
{
    "error": "ApiGateway.403",
    "cause": "{\"message\":\"Missing Authentication Token\"}"
}
```

> **Note**
>
> Um código de status 2XX indica sucesso e nenhum erro será retornado. Todos os outros códigos de status ou exceções lançadas resultarão em um erro.

Para obter mais informações, consulte:

- Conceitos do Amazon API Gateway no Guia do desenvolvedor do API Gateway.
- Políticas do IAM para o Amazon API Gateway
- Um exemplo de projeto que mostra como Fazer uma chamada para o API Gateway

Conceitos do Amazon API Gateway no Guia do desenvolvedor do API Gateway.

Chame o Athena com o Step Functions

Step Functions pode controlar determinados AWS serviços diretamente do Amazon States Language (ASL). Para saber mais, consulte Como trabalhar com outros serviços e Transmitir parâmetros para uma API de serviço.

> **Como a integração otimizada do Athena é diferente da integração do SDK do Athena AWS**
>
> - O padrão de integração Executar um trabalho (.sync) é compatível.
> - Não há otimizações para o padrão de integração Resposta de solicitação.
> - O padrão de integração Aguardar um retorno de chamada com um token de tarefa não é compatível.

A integração do AWS Step Functions serviço com o Amazon Athena permite que você use Step Functions para iniciar e interromper a execução de consultas e obter resultados de consultas.

Usando o Step Functions, você pode executar consultas de dados específicas ou agendadas e recuperar os resultados direcionados aos data lakes do S3. Como o Athena não utiliza servidor, não há infraestrutura para configurar ou gerenciar, e você paga apenas pelas consultas executadas.

Para fazer a integração AWS Step Functions com o Amazon Athena, você usa as APIs de integração de serviços fornecidas pelo Athena.

As APIs de integração de serviços são iguais às APIs Athena correspondentes. Nem todas as APIs oferecem suporte a todos os padrões de integração, como mostrado na tabela a seguir.

API	Resposta de solicitação	Executar um trabalho (.sync)
`StartQueryExecution`	✓	✓
`StopQueryExecution`	✓	
`GetQueryExecution`	✓	
`GetQueryResults`	✓	

APIs do Amazon Athena compatíveis:

> **Note**
>
> Há uma cota para o tamanho máximo de dados de entrada ou resultado para uma tarefa no Step Functions. Isso restringe você a 256 KB de dados como uma string codificada em UTF-8 quando você envia ou recebe dados de outro serviço. Consulte Cotas relacionadas a execuções de máquina de estado.

- `StartQueryExecution`
  - Sintaxe da solicitação
  - Parâmetros compatíveis:
    - `ClientRequestToken`
    - `ExecutionParameters`
    - `QueryExecutionContext`
    - `QueryString`

    - ResultConfiguration
    - WorkGroup
  - Response syntax
- StopQueryExecution
  - Sintaxe da solicitação
  - Parâmetros compatíveis:
    - QueryExecutionId
- GetQueryExecution
  - Sintaxe da solicitação
  - Parâmetros compatíveis:
    - QueryExecutionId
  - Response syntax
- GetQueryResults
  - Sintaxe da solicitação
  - Parâmetros compatíveis:
    - MaxResults
    - NextToken
    - QueryExecutionId
  - Response syntax

Veja a seguir um estado de Tarefa que inicia uma consulta do 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 obter informações sobre como configurar IAM permissões ao usar Step Functions com outros AWS serviços, consulte Políticas do IAM para serviços integrados.

Gerencie AWS Batch com Step Functions

Step Functions pode controlar determinados AWS serviços diretamente do Amazon States Language (ASL). Para saber mais, consulte Como trabalhar com outros serviços e Transmitir parâmetros para uma API de serviço.

> Como a AWS Batch integração otimizada é diferente da integração do AWS BatchAWS SDK
>
> - O padrão de integração Executar um trabalho (.sync) está disponível.
>
> Observe que não há otimizações para os padrões de integração Resposta de solicitação ou Aguardar um retorno de chamada com um token de tarefa.

AWS Batch APIs compatíveis:

- `SubmitJob`
  - Sintaxe da solicitação
  - Parâmetros compatíveis:
    - `ArrayProperties`
    - `ContainerOverrides`
    - `DependsOn`
    - `JobDefinition`
    - `JobName`
    - `JobQueue`
    - `Parameters`
    - `RetryStrategy`
    - `Timeout`
    - `Tags`
  - Sintaxe da resposta

> **Os parâmetros em Step Functions são expressos em PascalCase**
>
> Mesmo que a API de serviço nativa esteja no CamelCase, por exemplo, a `startSyncExecution` ação da API, você especifica parâmetros PascalCase em, como:. `StateMachineArn`

O seguinte inclui um `Task` estado que envia um AWS Batch trabalho e aguarda sua conclusão.

```
{
  "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 obter informações sobre como configurar IAM permissões ao usar Step Functions com outros AWS serviços, consulte[Políticas do IAM para serviços integrados](#).

Chamar o Amazon Bedrock com o Step Functions

Step Functions pode controlar determinados AWS serviços diretamente do Amazon States Language (ASL). Para saber mais, consulte Como trabalhar com outros serviços e Transmitir parâmetros para uma API de serviço.

Tópicos

- APIs de integração de serviço do Amazon Bedrock
- Definição do estado da tarefa para integração do Amazon Bedrock

APIs de integração de serviço do Amazon Bedrock

Para integrar o AWS Step Functions ao Amazon Bedrock, é possível usar as APIs a seguir. Essas APIs são semelhantes às APIs correspondentes do Amazon Bedrock, com algumas diferenças nos campos da solicitação que são passados.

A tabela a seguir descreve as diferenças entre cada API de integração de serviço e a API do Amazon Bedrock correspondente.

APIs de integração de serviço do Amazon Bedrock e APIs correspondentes do Amazon Bedrock

API de integração de serviço do Amazon Bedrock	API do Amazon Bedrock correspondente	Diferenças
InvokeModel Invoca o modelo do Amazon Bedrock especificado para executar a inferência usando a entrada fornecida no corpo da solicitação. Você vai usa `InvokeModel` para executar inferência para modelos de texto, modelos de imagem e modelos de incorporação.	InvokeModel	O corpo da solicitação da API de integração do serviço Amazon Bedrock inclui os parâmetros adicionais a seguir. • `Body`: especifica os dados de entrada no formato especificado no cabeçalho da solicitação do tipo de conteúdo. O `Body` contém parâmetros específicos do modelo de destino.

API de integração de serviço do Amazon Bedrock	API do Amazon Bedrock correspondente	Diferenças
		Se você usar a API `InvokeModel`, deverá especificar o parâmetro `Body`. O Step Functions não valida a entrada fornecida em `Body`. Ao especificar o `Body` usando a integração otimizada do Amazon Bedrock, é possível especificar uma carga útil de até 256 KB. Se a carga útil exceder 256 KB, recomendamos usar `Input`. • `Input`: especifica a fonte da qual recuperar os dados de entrada. Esse campo opcional é específico da integração otimizada do Amazon Bedrock ao Step Functions. Nesse campo, é possível especificar um `S3Uri`. É possível especificar `Body` nos Parâmetros ou `Input`, mas não ambos. Quando você especifica `Input` sem especificar `ContentType`, o tipo de conteúdo da fonte de dados

API de integração de serviço do Amazon Bedrock	API do Amazon Bedrock correspondente	Diferenças
		de entrada se torna o valor de `ContentType` . • `Output`: especifica o destino em que a resposta da API é gravada. Esse campo opcional é específico da integração otimizada do Amazon Bedrock ao Step Functions. Nesse campo, é possível especificar um `S3Uri`. Se você especificar esse campo, o corpo da resposta da API será substituído por uma referência ao local do Amazon S3 da saída original. O exemplo a seguir mostra a sintaxe InvokeModel da API para Amazon Bedrock integração. `{ "ModelId": String, // required "Accept": String, // default: application/json "ContentType": String, // default: application/json "Input": { // not from Bedrock API "S3Uri": String`

API de integração de serviço do Amazon Bedrock	API do Amazon Bedrock correspondente	Diferenças

		<pre> },
		    "Output": { // not from Bedrock API
		        "S3Uri": String
		    }
		}</pre>

CreateModelCustomizationJob Cria um trabalho de ajuste fino para personalizar um modelo básico.	CreateModelCustomizationJob	Nenhum
CreateModelCustomizationJob.sync Cria um trabalho de ajuste fino para personalizar um modelo básico.	CreateModelCustomizationJob	Nenhum

Para obter informações sobre como configurar IAM permissões ao usar Step Functions com outros AWS serviços, consultePolíticas do IAM para serviços integrados.

Definição do estado da tarefa para integração do Amazon Bedrock

A definição do estado de Tarefa a seguir mostra como se integrar ao Amazon Bedrock nas máquinas de estado. Este exemplo mostra um estado de Tarefa que extrai o resultado completo da invocação do modelo especificado pelo caminho, `result_one`. Isso se baseia em Parâmetros de inferência para modelos básicos. Este exemplo usa o grande modelo de linguagem (LLM) 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 implantar um exemplo de uma máquina de estado que se integra Amazon Bedrock à sua Conta da AWS, consulteRealizar o encadeamento de prompts de IA com o Amazon Bedrock.

Chamada AWS CodeBuild com Step Functions

Step Functions pode controlar determinados AWS serviços diretamente do Amazon States Language (ASL). Para saber mais, consulte Como trabalhar com outros serviços e Transmitir parâmetros para uma API de serviço.

> **Como a CodeBuild integração otimizada é diferente da integração do CodeBuild AWS SDK**
>
> - O padrão de integração Executar um trabalho (.sync) é compatível.
> - Depois de chamar `StopBuild` ou`StopBuildBatch`, a compilação ou o lote de compilação não podem ser excluídos imediatamente até que algum trabalho interno seja concluído CodeBuild para finalizar o estado da compilação ou compilações. Se você tentar usar `BatchDeleteBuilds` ou `DeleteBuildBatch` durante esse período, a compilação ou o lote de compilação não poderão ser excluídos. As integrações de serviços otimizadas de `BatchDeleteBuilds` e `DeleteBuildBatch` incluem uma nova tentativa interna para simplificar o caso de uso de exclusão imediata após a interrupção.

A integração do AWS Step Functions serviço com AWS CodeBuild permite que você use Step Functions para acionar, interromper e gerenciar compilações e compartilhar relatórios de criação.

Com o Step Functions, é possível projetar e executar pipelines de integração contínua para validar as alterações de software para aplicações.

Nem todas as APIs oferecem suporte a todos os padrões de integração, como mostrado na tabela a seguir.

API	Resposta de solicitação	Executar um trabalho (.sync)
StartBuild	✓	✓
StopBuild	✓	
BatchDeleteBuilds	✓	
BatchGetReports	✓	
StartBuildBatch	✓	✓
StopBuildBatch	✓	
RetryBuildBatch	✓	✓
DeleteBuildBatch	✓	

> **Os parâmetros em Step Functions são expressos em PascalCase**
>
> Mesmo que a API de serviço nativa esteja no CamelCase, por exemplo, a `startSyncExecution` ação da API, você especifica parâmetros PascalCase em, como:. `StateMachineArn`

CodeBuild APIs e sintaxe suportadas:

- `StartBuild`
  - Sintaxe da solicitação
  - Parâmetros compatíveis:
    - `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

- Sintaxe da resposta

- StopBuild
  - Sintaxe da solicitação

- Parâmetros compatíveis:
  - `Id`
- Sintaxe da resposta

- `BatchDeleteBuilds`
  - Sintaxe da solicitação
  - Parâmetros compatíveis:
    - `Ids`
  - Sintaxe da resposta
- `BatchGetReports`
  - Sintaxe da solicitação
  - Parâmetros compatíveis:
    - `ReportArns`
  - Sintaxe da resposta
- `StartBuildBatch`
  - Sintaxe da solicitação
  - Parâmetros compatíveis:
    - `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

- Sintaxe da resposta

- StopBuildBatch
  - Sintaxe da solicitação
  - Parâmetros compatíveis:
    - Id
  - Sintaxe da resposta
- RetryBuildBatch
  - Sintaxe da solicitação
  - Parâmetros compatíveis:
    - Id
    - IdempotencyToken
    - RetryType
  - Sintaxe da resposta

- `DeleteBuildBatch`
  - Sintaxe da solicitação
  - Parâmetros compatíveis:
    - `Id`
  - Sintaxe da resposta

> **Note**
>
> É possível usar o operador de descida recursiva `JSONPath` (`..`) para `BatchDeleteBuilds`. Isso retorna uma matriz e permite que você transforme o campo `Arn` de `StartBuild` em um parâmetro plural `Ids`, como mostrado no exemplo a seguir.
>
> ```
> "BatchDeleteBuilds": {
>     "Type": "Task",
>     "Resource": "arn:aws:states:::codebuild:batchDeleteBuilds",
>     "Parameters": {
>         "Ids.$": "$.Build..Arn"
>     },
>     "Next": "MyNextState"
> },
> ```

Para obter informações sobre como configurar IAM permissões ao usar Step Functions com outros AWS serviços, consultePolíticas do IAM para serviços integrados.

Chamar as APIs do DynamoDB com o Step Functions

Step Functions pode controlar determinados AWS serviços diretamente do Amazon States Language (ASL). Para saber mais, consulte Como trabalhar com outros serviços e Transmitir parâmetros para uma API de serviço.

> **Note**
>
> Há uma cota para o tamanho máximo de dados de entrada ou resultado para uma tarefa no Step Functions. Isso restringe você a 256 KB de dados como uma string codificada em

UTF-8 quando você envia ou recebe dados de outro serviço. Consulte Cotas relacionadas a execuções de máquina de estado.

Como a integração otimizada do DynamoDB é diferente da integração do SDK do DynamoDB AWS

- Não há otimização para o padrão de integração Resposta de solicitação.
- O padrão de integração Aguardar um retorno de chamada com um token de tarefa não é compatível.
- Somente as ações `GetItem`, `PutItem`, `UpdateItem` e `DeleteItem` da API estão disponíveis por meio de integração otimizada. Outras ações de API, como as que `CreateTable`estão disponíveis usando a integração do AWS DynamoDB SDK.

APIs e sintaxe do Amazon DynamoDB compatíveis:

- `GetItem`
  - Sintaxe da solicitação
  - Parâmetros compatíveis:
    - `Key`
    - `TableName`
    - `AttributesToGet`
    - `ConsistentRead`
    - `ExpressionAttributeNames`
    - `ProjectionExpression`
    - `ReturnConsumedCapacity`
  - Sintaxe da resposta
- `PutItem`
  - Sintaxe da solicitação
  - Parâmetros compatíveis:
    - `Item`
    - `TableName`

    - ConditionalOperator
    - ConditionExpression
    - Expected
    - ExpressionAttributeNames
    - ExpressionAttributeValues
    - ReturnConsumedCapacity
    - ReturnItemCollectionMetrics
    - ReturnValues
  - Sintaxe da resposta
- DeleteItem
  - Sintaxe da solicitação
  - Parâmetros compatíveis:
    - Key
    - TableName
    - ConditionalOperator
    - ConditionExpression
    - Expected
    - ExpressionAttributeNames
    - ExpressionAttributeValues
    - ReturnConsumedCapacity
    - ReturnItemCollectionMetrics
    - ReturnValues
  - Sintaxe da resposta
- UpdateItem
  - Sintaxe da solicitação
  - Parâmetros compatíveis:
    - Key
    - TableName
    - AttributeUpdates
    - ConditionalOperator

- ConditionExpression
- Expected
- ExpressionAttributeNames
- ExpressionAttributeValues
- ReturnConsumedCapacity
- ReturnItemCollectionMetrics
- ReturnValues
- UpdateExpression

- Sintaxe da resposta

> **Os parâmetros em Step Functions são expressos em PascalCase**
>
> Mesmo que a API de serviço nativa esteja no CamelCase, por exemplo, a `startSyncExecution` ação da API, você especifica parâmetros PascalCase em, como:. `StateMachineArn`

O seguinte é um estado `Task` que recupera uma mensagem do 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 esse estado em um exemplo funcional, consulte o projeto de amostra Transferir registros de dados (Lambda, DynamoDB, Amazon SQS).

Para obter informações sobre como configurar IAM permissões ao usar Step Functions com outros AWS serviços, consultePolíticas do IAM para serviços integrados.

Gerenciar tarefas do Amazon ECS ou do Fargate com o Step Functions

Step Functions pode controlar determinados AWS serviços diretamente do Amazon States Language (ASL). Para saber mais, consulte Como trabalhar com outros serviços e Transmitir parâmetros para uma API de serviço.

> Como a integração otimizada do Amazon ECS/Fargate é diferente da integração do Amazon ECS ou do Fargate SDK AWS
>
> - O padrão de integração Executar um trabalho (.sync) é compatível.
> - O `ecs:runTask` pode retornar uma resposta HTTP 200, mas ter um campo `Failures` não vazio da seguinte forma:
>   - Solicitar resposta: retorne a resposta e não falhe na tarefa. Isso é o mesmo que não haver otimização.
>   - Executar um trabalho ou um token de tarefa: se um campo `Failures` não vazio for encontrado, a tarefa falhará com um erro `AmazonECS.Unknown`.

APIs e sintaxe compatíveis do Amazon ECS/Fargate:

> Os parâmetros em Step Functions são expressos em PascalCase
>
> Mesmo que a API de serviço nativa esteja no CamelCase, por exemplo, a `startSyncExecution` ação da API, você especifica parâmetros PascalCase em, como:. `StateMachineArn`

- `RunTask` inicia uma nova tarefa usando a definição de tarefa especificada.
  - Sintaxe da solicitação
  - Parâmetros compatíveis:
    - `Cluster`
    - `Group`
    - `LaunchType`

- NetworkConfiguration
- Overrides
- PlacementConstraints
- PlacementStrategy
- PlatformVersion
- PropagateTags
- TaskDefinition
- EnableExecuteCommand

- Sintaxe da resposta

Como transmitir dados para uma tarefa do Amazon ECS

Step Functions pode controlar determinados AWS serviços diretamente do Amazon States Language (ASL). Para saber mais, consulte Como trabalhar com outros serviços e Transmitir parâmetros para uma API de serviço.

É possível usar o comando `overrides` para substituir o comando padrão de um contêiner e transmitir a entrada para as tarefas do Amazon ECS. Consulte `ContainerOverride`. No exemplo, usamos JsonPath para passar valores para a `Task` da entrada para o `Task` estado.

Veja a seguir um estado `Task` que executa uma tarefa do Amazon ECS e aguarda até que ela seja concluída.

```
{
 "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
    }
  }
}
```

A linha `"Command.$": "$.commands"` em `ContainerOverrides` transmite os comandos da entrada do estado para o contêiner.

Para o exemplo anterior, cada um dos comandos será transmitido como uma substituição de contêiner se a entrada para a execução for a seguinte.

```
{
  "commands": [
    "test command 1",
    "test command 2",
    "test command 3"
  ]
}
```

Veja a seguir um estado `Task` que executa uma tarefa do Amazon ECS e aguarda o retorno do token da tarefa. Consulte Aguardar um retorno de chamada com um token de tarefa.

```
{
   "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 obter informações sobre como configurar IAM permissões ao usar Step Functions com outros AWS serviços, consulte Políticas do IAM para serviços integrados.

Chamar o Amazon EKS com o Step Functions

Step Functions pode controlar determinados AWS serviços diretamente do Amazon States Language (ASL). Para saber mais, consulte Como trabalhar com outros serviços e Transmitir parâmetros para uma API de serviço.

> Como a integração otimizada do Amazon EKS é diferente da integração do Amazon EKS AWS SDK
>
> - O padrão de integração Executar um trabalho (.sync) é compatível.
> - Não há otimizações para o padrão de integração Resposta de solicitação.
> - O padrão de integração Aguardar um retorno de chamada com um token de tarefa não é compatível.

Para obter informações sobre como configurar IAM permissões ao usar Step Functions com outros AWS serviços, consulte Políticas do IAM para serviços integrados.

O Step Functions fornece dois tipos de APIs de integração de serviços para integração com o Amazon Elastic Kubernetes Service. Uma delas permite que você use as APIs do Amazon EKS para criar e gerenciar um cluster do Amazon EKS. A outra permite que você interaja com seu cluster usando a API Kubernetes e execute trabalhos como parte do fluxo de trabalho do seu aplicativo. Você pode usar as integrações da API do Kubernetes com clusters do Amazon EKS criados usando Step Functions, com clusters do Amazon EKS criados pela ferramenta eksctl ou pelo console do

Amazon EKS ou métodos similares. Para obter mais informações, consulte Criação de um cluster para o Amazon EKS no Guia do usuário do Amazon EKS.

> **Note**
>
> A integração do Step Functions EKS oferece suporte somente às APIs do Kubernetes com acesso público ao endpoint. Por padrão, os endpoints do servidor da API de clusters EKS têm acesso público. Para obter mais informações, consulte o Controle de acesso ao endpoint do cluster do Amazon EKS no Guia do Usuário do Amazon EKS.

O Step Functions não encerra automaticamente um cluster do Amazon EKS se a execução for interrompida. Se sua máquina de estado parar antes do encerramento do cluster Amazon EKS, seu cluster poderá continuar funcionando indefinidamente e poderá acumular cobranças adicionais. Para evitar isso, certifique-se de que qualquer cluster do Amazon EKS que você criar seja encerrado corretamente. Para obter mais informações, consulte:

- Excluir um cluster no Guia do Usuário do Amazon EKS.
- Executar um trabalho (.sync) nos Padrões de Integração de Serviço.

> **Note**
>
> Há uma cota para o tamanho máximo de dados de entrada ou resultado para uma tarefa no Step Functions. Isso restringe você a 256 KB de dados como uma string codificada em UTF-8 quando você envia ou recebe dados de outro serviço. Consulte Cotas relacionadas a execuções de máquina de estado.

Integrações da API do Kubernetes

O Step Functions é compatível com as seguintes APIs do Kubernetes:

RunJob

A integração do serviço `eks:runJob` permite que você execute um trabalho no cluster do Amazon EKS. A variante `eks:runJob.sync` permite que você aguarde a conclusão do trabalho e, opcionalmente, recupere os registros.

Seu servidor da API do Kubernetes deve conceder permissões para o perfil do IAM usado pela sua máquina de estado. Para ter mais informações, consulte Permissões.

Para o padrão Executar um trabalho (`.sync`), o status do trabalho é determinado por meio de pesquisa. O Step Functions pesquisa inicialmente a uma taxa de aproximadamente 1 pesquisa por minuto. Por fim, essa taxa diminui para aproximadamente 1 pesquisa a cada 5 minutos. Se você precisar de pesquisas mais frequentes ou precisar de mais controle sobre a estratégia de pesquisa, poderá usar a integração `eks:call` para consultar o status do trabalho.

A integração `eks:runJob` é específica para `batch/v1` Tarefas no Kubernetes. Para obter mais informações, consulte Tarefas na documentação do Kubernetes. Se você quiser gerenciar outros recursos do Kubernetes, incluindo recursos personalizados, use a integração de serviços `eks:call`. Você pode usar o Step Functions para criar ciclos de pesquisa, conforme demonstrado no projeto de exemplo de the section called “Pesquisa de status de trabalho (AWS Batch Lambda,) ”.

Os parâmetros compatíveis incluem:

- `ClusterName`: o nome do cluster do Amazon EKS que você quer chamar.
  - `Type`: `String`
  - Obrigatório: sim
- `CertificateAuthority`: os dados de certificado codificados em Base64 necessários para se comunicar com o cluster. Você pode obter esse valor no console do Amazon EKS ou usando a DescribeClusterAPI do Amazon EKS.
  - `Type`: `String`
  - Obrigatório: sim
- `Endpoint`: o URL do endpoint para o servidor da API do Kubernetes. Você pode obter esse valor no console do Amazon EKS ou usando a DescribeClusterAPI do Amazon EKS.
  - `Type`: `String`
  - Obrigatório: sim
- `Namespace`: o namespace em que a tarefa será executada. Se não for fornecido, o namespace `default` será usado.
  - `Type`: `String`
  - Obrigatório: não
- `Job`: A definição da Tarefa do Kubernetes. Consulte Tarefas na documentação do Kubernetes.
  - `Type`: `JSON` ou `String`

  - Obrigatório: sim
- `LogOptions`: um conjunto de opções para controlar a recuperação opcional de logs. Aplicável somente se o padrão de integração do serviço Executar um trabalho (.sync) for usado para aguardar a conclusão do trabalho.
  - `Type`: `JSON`
  - Obrigatório: não
  - Os registros são incluídos na resposta abaixo da chave `logs`. Pode haver vários pods na tarefa, cada uma com vários contêineres.

```
{
  ...
  "logs": {
    "pods": {
      "pod1": {
        "containers": {
          "container1": {
            "log": <log>
          },
          ...
        }
      },
      ...
    }
  }
```

  - A recuperação do log é realizada com base no melhor esforço. Se houver um erro ao recuperar um log, no lugar do campo `log` estarão os campos `error` e `cause`.
- `LogOptions.RetrieveLogs`: habilite a recuperação de log após a conclusão da tarefa. Por padrão, os logs não são recuperados.
  - `Type`: `Boolean`
  - Obrigatório: não
- `LogOptions.RawLogs`: se `RawLogs` for definido como verdadeiro, os logs serão retornados como strings brutas sem a tentativa de analisá-los em JSON. Por padrão, os logs são desserializados em JSON, se possível. Em alguns casos, essa análise pode introduzir alterações indesejadas, como limitação da precisão de números contendo muitos dígitos.
  - `Type`: `Boolean`
  - Obrigatório: não

- `LogOptions.LogParameters`: a API Read Log da API do Kubernetes é compatível com parâmetros de consulta para controlar a recuperação de logs. Por exemplo, você pode usar `tailLines` ou `limitBytes` para limitar o tamanho dos logs recuperados e permanecer dentro da cota de tamanho de dados do Step Functions. Para obter mais informações, consulte a seção Log de leitura da Referência da API do Kubernetes.
  - `Type:` Mapa de `String` até `List of Strings`
  - Obrigatório: não
  - Exemplo:

```
"LogParameters": {
  "tailLines": [ "6" ]
}
```

O exemplo a seguir inclui um estado `Task` que executa um trabalho, aguarda sua conclusão e, em seguida, recupera os logs do trabalho:

```
{
  "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`

A integração do serviço `eks:call` permite que você use a API do Kubernetes para ler e gravar objetos de recursos do Kubernetes por meio de um endpoint da API do Kubernetes.

Seu servidor da API do Kubernetes deve conceder permissões para o perfil do IAM usado pela sua máquina de estado. Para ter mais informações, consulte Permissões.

Para obter mais informações sobre as operações disponíveis, consulte a Referência de API do Kubernetes.

Os parâmetros compatíveis para `Call` incluem:

- `ClusterName`: o nome do cluster do Amazon EKS que você quer chamar.
  - `Type`: string
  - Obrigatório: Sim

- `CertificateAuthority`: os dados de certificado codificados em Base64 necessários para se comunicar com o cluster. Você pode obter esse valor no console do Amazon EKS ou usando a DescribeClusterAPI do Amazon EKS.
  - `Type`: `String`
  - Obrigatório: Sim
- `Endpoint`: o URL do endpoint para o servidor da API do Kubernetes. Você pode encontrar esse valor no console do Amazon EKS ou usando a DescribeCluster API do Amazon EKS.
  - `Type`: `String`
  - Obrigatório: Sim
- `Method`: o método HTTP da sua solicitação. Um destes: `GET`, `POST`, `PUT`, `DELETE`, `HEAD` ou `PATCH`.
  - `Type`: `String`
  - Obrigatório: Sim
- `Path`: o caminho HTTP da operação da API REST do Kubernetes.
  - `Type`: `String`
  - Obrigatório: Sim
- `QueryParameters`: os parâmetros de consulta HTTP da operação da API REST do Kubernetes.
  - `Type:` Mapa de `String` até `List of Strings`
  - Obrigatório: não
  - Exemplo:

```
"QueryParameters": {
  "labelSelector": [ "job-name=example-job" ]
}
```

- `RequestBody`: o corpo da mensagem HTTP da operação da API REST do Kubernetes.
  - `Type`: `JSON` ou `String`
  - Obrigatório: não

Veja a seguir um estado `Task` que utiliza `eks:call` para listar os pods pertencentes à tarefa `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
    }
  }
}
```

Veja a seguir um estado `Task` que utiliza `eks:call` para excluir a tarefa `example-job` e define o `propagationPolicy` para garantir que os pods da tarefa também sejam excluídos.

```
{
  "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
```

```
    }
  }
}
```

APIs compatíveis do Amazon EKS

As APIs e a sintaxe compatíveis do Amazon EKS incluem:

- `CreateCluster`
  - Sintaxe da solicitação
  - Sintaxe da resposta

    Quando um cluster do Amazon EKS é criado usando a integração do serviço `eks:createCluster`, o perfil do IAM é adicionado à tabela de autorização de RBAC do Kubernetes como o administrador (com permissões do system:masters). Inicialmente, somente a entidade do IAM pode fazer chamadas para o servidor da API do Kubernetes. Para obter mais informações, consulte:

    - Gerenciar usuários ou funções do IAM para o seu cluster no Guia do Usuário do Amazon EKS
    - A Permissões seção

    O Amazon EKS usa funções vinculadas ao serviço que contém as permissões que o Amazon EKS considera necessárias para chamar outros serviços em seu nome. Se essas funções vinculadas ao serviço ainda não existirem em sua conta, você deverá adicionar a permissão `iam:CreateServiceLinkedRole` ao perfil do IAM usado pelo Step Functions. Para obter mais informações, consulte Uso de Funções Vinculadas ao Serviço no Guia do Usuário do Amazon EKS.

    O perfil do IAM usado pelo Step Functions deve ter permissões `iam:PassRole` para aprovar o perfil do IAM do cluster para o Amazon EKS. Para obter mais informações, consulte o perfil do IAM do cluster do Amazon EKS no Guia do Usuário do Amazon EKS.
- `DeleteCluster`
  - Sintaxe da solicitação
  - Sintaxe da resposta

    Você deve excluir todos os perfis ou grupos de nós do Fargate antes de excluir um cluster.
- `CreateFargateProfile`
  - Sintaxe da solicitação

- Sintaxe da resposta

  O Amazon EKS usa funções vinculadas ao serviço que contém as permissões que o Amazon EKS considera necessárias para chamar outros serviços em seu nome. Se essas funções vinculadas ao serviço ainda não existirem em sua conta, você deverá adicionar a permissão `iam:CreateServiceLinkedRole` ao perfil do IAM usado pelo Step Functions. Para obter mais informações, consulte Uso de Funções Vinculadas ao Serviço no Guia do Usuário do Amazon EKS.

  O Amazon EKS no Fargate pode não estar disponível em todas as regiões. Para obter informações sobre a disponibilidade por região, consulte a seção sobre o Fargate no Guia do usuário do Amazon EKS.

  O perfil do IAM usado pelo Step Functions deve ter permissões `iam:PassRole` para aprovar o perfil do IAM de execução do pod para o Amazon EKS. Para obter mais informações, consulte Função de execução de pods no Guia do Usuário do Amazon EKS.

- `DeleteFargateProfile`
  - Sintaxe da solicitação
  - Sintaxe da resposta
- `CreateNodegroup`
  - Sintaxe da solicitação
  - Sintaxe da resposta

  O Amazon EKS usa funções vinculadas ao serviço que contém as permissões que o Amazon EKS considera necessárias para chamar outros serviços em seu nome. Se essas funções vinculadas ao serviço ainda não existirem em sua conta, você deverá adicionar a permissão `iam:CreateServiceLinkedRole` ao perfil do IAM usado pelo Step Functions. Para obter mais informações, consulte Uso de Funções Vinculadas ao Serviço no Guia do Usuário do Amazon EKS.

  O perfil do IAM usado pelo Step Functions deve ter permissões `iam:PassRole` para aprovar o perfil do IAM do nó para o Amazon EKS. Para obter mais informações, consulte Uso de Funções Vinculadas ao Serviço no Guia do Usuário do Amazon EKS.

- `DeleteNodegroup`
  - Sintaxe da solicitação
  - Sintaxe da resposta

Veja a seguir um `Task` que cria um cluster do 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
    }
  }
}
```

Veja a seguir um estado `Task` que exclui um cluster do Amazon EKS.

```
{
  "StartAt": "DeleteCluster.sync",
  "States": {
    "DeleteCluster.sync": {
      "Type": "Task",
      "Resource": "arn:aws:states:::eks:deleteCluster.sync",
      "Parameters": {
        "Name": "MyCluster"
      },
      "End": true
    }
  }
}
```

Veja a seguir um estado `Task` que cria um perfil do 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
    }
  }
}
```

Veja a seguir um estado `Task` que exclui um perfil do Fargate.

```
{
  "StartAt": "DeleteFargateProfile.sync",
  "States": {
    "DeleteFargateProfile.sync": {
      "Type": "Task",
      "Resource": "arn:aws:states:::eks:deleteFargateProfile.sync",
      "Parameters": {
        "ClusterName": "MyCluster",
        "FargateProfileName": "MyFargateProfile"
      },
      "End": true
    }
  }
}
```

Veja a seguir um estado `Task` que cria um grupo de nós.

```
{
  "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
    }
  }
}
```

Veja a seguir um estado `Task` que exclui um grupo de nós.

```
{
  "StartAt": "DeleteNodegroup.sync",
  "States": {
    "DeleteNodegroup.sync": {
      "Type": "Task",
      "Resource": "arn:aws:states:::eks:deleteNodegroup.sync",
      "Parameters": {
        "ClusterName": "MyCluster",
        "NodegroupName": "MyNodegroup"
      },
      "End": true
    }
  }
}
```

Permissões

Quando um cluster do Amazon EKS é criado usando a integração do serviço `eks:createCluster`, o perfil do IAM é adicionado à tabela de autorização RBAC do Kubernetes como o administrador, com permissões do `system:masters`. Inicialmente, somente a entidade do IAM pode fazer chamadas para o servidor da API do Kubernetes. Por exemplo, você não poderá usar o kubectl para interagir com seu servidor de API do Kubernetes, a menos que assuma a mesma função da sua máquina de estado Step Functions ou configure o Kubernetes para conceder permissões a entidades adicionais do IAM. Para obter mais informações, consulte Gerenciar usuários ou funções do IAM para o seu cluster no Guia do Usuário do Amazon EKS.

Você pode adicionar permissão para entidades adicionais do IAM, como usuários ou funções, adicionando-as ao namespace `aws-auth` `ConfigMap` no kube-system. Se você estiver criando seu cluster a partir do Step Functions, use a integração de serviços `eks:call`.

Veja a seguir um estado `Task` que cria `aws-auth` `ConfigMap` e concede permissão `system:masters` ao usuário `arn:aws:iam::123456789012:user/my-user` e ao perfil do 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**
>
> Você pode ver o ARN de um perfil do IAM exibido em um formato que inclui o caminho / service-role/, como `arn:aws:iam::123456789012:role/service-role/my-role`. Esse token do caminho service-role não deve ser incluído ao listar a função em `aws-auth`

Quando seu cluster for criado pela primeira vez, o `aws-auth` `ConfigMap` não existirá, mas será adicionado automaticamente se você criar um perfil do Fargate. Você pode recuperar o valor atual de`aws-auth`, adicionar as permissões adicionais e PUT uma nova versão. Normalmente, é mais fácil criar `aws-auth` antes do perfil do Fargate.

Se seu cluster foi criado fora do Step Functions, você pode configurar o kubectl para se comunicar com seu servidor de API do Kubernetes. Em seguida, crie um novo `aws-auth` `ConfigMap` usando `kubectl apply -f aws-auth.yaml` ou edite um que já exista usando`kubectl edit -n kube-system configmap/aws-auth`. Para obter mais informações, consulte:

- Crie um kubeconfig para o Amazon EKS no Guia do Usuário do Amazon EKS.
- Gerenciar usuários ou funções do IAM para o seu cluster no Guia do Usuário do Amazon EKS.

Se seu perfil do IAM não tiver permissões suficientes no Kubernetes, as integrações dos serviços `eks:call` ou `eks:runJob` falharão com o seguinte erro:

```
Error:
EKS.401

Cause:
{
  "ResponseBody": {
    "kind": "Status",
    "apiVersion": "v1",
    "metadata": {},
    "status": "Failure",
    "message": "Unauthorized",
    "reason": "Unauthorized",
    "code": 401
  },
  "StatusCode": 401,
  "StatusText": "Unauthorized"
}
```

Chamar o Amazon EMR com o Step Functions

Step Functions pode controlar determinados AWS serviços diretamente do Amazon States Language (ASL). Para saber mais, consulte Como trabalhar com outros serviços e Transmitir parâmetros para uma API de serviço.

> **Como a integração otimizada do Amazon EMR é diferente da integração do Amazon EMR SDK AWS**
>
> A integração otimizada do serviço Amazon EMR tem um conjunto personalizado de APIs que engloba as APIs subjacentes do Amazon EMR, descritas abaixo. Por isso, ele difere significativamente da integração de serviços do Amazon EMR AWS SDK. Além disso, o padrão de integração Executar um trabalho (.sync) é compatível.

Para se integrar AWS Step Functions ao Amazon EMR, você usa as APIs de integração de serviços do Amazon EMR fornecidas. As APIs de integração de serviço são semelhantes às APIs correspondentes do Amazon EMR, com algumas diferenças nos campos que são passados e nas respostas retornadas.

O Step Functions não encerra automaticamente um cluster do Amazon EMR se a execução for interrompida. Se sua máquina de estado parar antes do encerramento do cluster do Amazon EMR, seu cluster poderá continuar funcionando indefinidamente e poderá acumular cobranças adicionais. Para evitar isso, certifique-se de que qualquer cluster do Amazon EMR criado por você seja encerrado corretamente. Para obter mais informações, consulte:

- Controle o encerramento do cluster no Guia do Usuário do Amazon EMR.
- A seção Executar um trabalho (.sync) de padrões de integração de serviços.

> **Note**
>
> A partir de `emr-5.28.0`, você pode especificar o parâmetro `StepConcurrencyLevel` ao criar um cluster para permitir que várias etapas sejam executadas em paralelo em um

único cluster. Você pode usar os estados de `Map` e `Parallel` do Step Functions para enviar trabalho em paralelo ao cluster.

A disponibilidade da integração do serviço do Amazon EMR está sujeita à disponibilidade de APIs do Amazon EMR. Verifique a documentação do Amazon EMR quanto a limitações em regiões especiais.

> **Note**
>
> Para integração com o Amazon EMR, o Step Functions tem uma frequência de pesquisa de trabalhos de codificação rígida de 60 segundos para os primeiros 10 minutos e 300 segundos depois disso.

A tabela a seguir descreve as diferenças entre cada API de integração de serviço e a API correspondente do Amazon EMR.

APIs de integração de serviços do Amazon EMR e APIs correspondentes do Amazon EMR

API de integração de serviços do Amazon EMR	API do EMR correspondente	Diferenças
createCluster Cria e inicia a execução de um cluster (fluxo de trabalho). O Amazon EMR está vinculado diretamente a um tipo exclusivo de perfil do IAM conhecido como função vinculada ao serviço. Para que `createCluster` e `createCluster.sync` funcionem, você deve ter configurado as permissões necessárias para criar a função vinculada ao serviço `AWSServiceRoleForE`	runJobFlow	`createCluster` usa a mesma sintaxe de solicitação que runJobFlow, exceto pelo seguinte: • O campo `Instances.KeepJobFlowAliveWhenNoSteps` é obrigatório e deve ter o valor booliano `TRUE`. • O campo `Steps` não é permitido. • O campo `Instances.InstanceFleets[index].Name` deve ser fornecido e deve ser

API de integração de serviços do Amazon EMR	API do EMR correspondente	Diferenças

`MRCleanup` . Para obter mais informações sobre isso, incluindo uma instrução que você pode adicionar à política de permissões IAM, consulte Usar a função vinculada ao serviço para o Amazon EMR.		exclusivo se a API opcional do conector `modifyInstanceFleetByName` for usada.
		• O campo `Instances.InstanceGroups[index].Name` deve ser fornecido e deve ser exclusivo se a API `modifyInstanceGroupByName` opcional for usada.
		
		A resposta é a seguinte:
		
		`{ "ClusterId": "string" }`
		
		O Amazon EMR usa o seguinte:
		
		`{ "JobFlowId": "string" }`

createCluster.sync Cria e inicia a execução de um cluster (fluxo de trabalho).	runJobFlow	O mesmo que `createCluster` , mas espera que o cluster atinja o estado `WAITING`.

API de integração de serviços do Amazon EMR	API do EMR correspondente	Diferenças

setClusterTerminationProteção
 	
Bloqueia um cluster (fluxo de trabalho) para que as instâncias do EC2 no cluster não possam ser encerradas por intervenção do usuário, uma chamada de API ou erro de fluxo de trabalho.
	setTerminationProtection	A solicitação usa o seguinte:
		
		`{ "ClusterId": "string" }`
		
		O Amazon EMR usa o seguinte:
		
		`{ "JobFlowIds": ["string"] }`

terminateCluster
 	
Desliga um cluster (fluxo de trabalho).
	terminateJobFlows	A solicitação usa o seguinte:
		
		`{ "ClusterId": "string" }`
		
		O Amazon EMR usa o seguinte:
		
		`{ "JobFlowIds": ["string"] }`

terminateCluster.sync Desliga um cluster (fluxo de trabalho).	terminateJobFlows	O mesmo que `terminateCluster` , mas aguarda o encerramento do cluster.

API de integração de serviços do Amazon EMR	API do EMR correspondente	Diferenças

addStep
 	
Adiciona uma nova etapa a um cluster em execução.
 	
Se preferir, você também poderá especificar o parâmetro `ExecutionRoleArn` ao usar essa API.
	addJobFlowEtapas	A solicitação usa a chave `"ClusterId"` . O Amazon EMR usa o `"JobFlowId"` . A solicitação usa uma única etapa.
		
		`{ "Step": <"StepConfig object"> }`
		
		O Amazon EMR usa o seguinte:
		
		`{ "Steps": [<StepConfig objects>] }`
		
		A resposta é a seguinte:
		
		`{ "StepId": "string" }`
		
		O Amazon EMR retorna o seguinte:
		
		`{ "StepIds": [<strings>] }`

API de integração de serviços do Amazon EMR	API do EMR correspondente	Diferenças
addStep.sync Adiciona uma nova etapa a um cluster em execução. Se preferir, você também poderá especificar o parâmetro `ExecutionRoleArn` ao usar essa API.	addJobFlowEtapas	O mesmo que `addStep`, mas aguarda a etapa ser concluída .

API de integração de serviços do Amazon EMR	API do EMR correspondente	Diferenças

cancelStep
 	
Cancela uma etapa pendente em um cluster em execução.
	cancelSteps	A solicitação usa o seguinte:
		
		`{ "StepId": "string" }`
		
		O Amazon EMR usa o seguinte:
		
		`{ "StepIds": [<strings >] }`
		
		A resposta é a seguinte:
		
		`{ "CancelStepsInfo": <CancelStepsInfo object> }`
		
		O Amazon EMR usa o seguinte:
		
		`{ "CancelStepsInfoLi st": [<CancelStepsInfo objects>] }`

API de integração de serviços do Amazon EMR	API do EMR correspondente	Diferenças
modifyInstanceFleetByName Modifica as capacidades sob demanda e spot de destino para a frota de instâncias com o `InstanceFleetName` especificado.	modifyInstanceFleet	O pedido é o mesmo que para `modifyInstanceFleet` , exceto pelo seguinte: • O campo `Instance.InstanceFleetId` não é permitido. • Em tempo de execução, o `InstanceFleetId` é determinado automaticamente pela integração do serviço chamando `ListInstanceFleets` e analisando o resultado.

API de integração de serviços do Amazon EMR	API do EMR correspondente	Diferenças

modifyInstanceGroupByName
 	
Modifica o número de nós e as configurações de um grupo de instâncias.
	modifyInstanceGroups	O pedido é o seguinte:
		
		`{ "ClusterId": "string", "InstanceGroup": <InstanceGroupModifyConfig object> }`
		
		O Amazon EMR usa uma lista:
		
		`{ "ClusterId": ["string"], "InstanceGroups": [<InstanceGroupModifyConfig objects>] }`
		
		Dentro do objeto `InstanceGroupModifyConfig`, o campo `InstanceGroupId` não é permitido.
		
		Um novo campo, `InstanceGroupName`, foi adicionado. Em tempo de execução, o `InstanceGroupId` é determinado automaticamente pela integração do serviço chamando `ListInstanceGroups` e analisando o resultado.

Veja a seguir um estado `Task` que cria um cluster.

```
"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
```

```
}
```

Veja a seguir um estado `Task` que habilita a proteção contra encerramento.

```
"Enable_Termination_Protection": {
    "Type": "Task",
    "Resource": "arn:aws:states:::elasticmapreduce:setClusterTerminationProtection",
    "Parameters": {
        "ClusterId.$": "$.ClusterId",
        "TerminationProtected": true
    },
    "End": true
}
```

Veja a seguir um estado `Task` que envia uma etapa para um cluster.

```
"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
```

```
}
```

Veja a seguir um estado `Task` que cancela uma etapa.

```
"Cancel_Step_One": {
    "Type": "Task",
    "Resource": "arn:aws:states:::elasticmapreduce:cancelStep",
    "Parameters": {
        "ClusterId.$": "$.ClusterId",
        "StepId.$": "$.AddStepsResult.StepId"
    },
    "End": true
}
```

Veja a seguir um estado `Task` que encerra um cluster.

```
"Terminate_Cluster": {
    "Type": "Task",
    "Resource": "arn:aws:states:::elasticmapreduce:terminateCluster.sync",
    "Parameters": {
        "ClusterId.$": "$.ClusterId"
    },
    "End": true
}
```

Veja a seguir um estado `Task` que expande ou reduz um cluster para um grupo de instâncias.

```
"ModifyInstanceGroupByName": {
    "Type": "Task",
    "Resource": "arn:aws:states:::elasticmapreduce:modifyInstanceGroupByName",
    "Parameters": {
        "ClusterId": "j-1234567890123",
        "InstanceGroupName": "MyCoreGroup",
        "InstanceGroup": {
            "InstanceCount": 8
        }
    },
    "End": true
}
```

Veja a seguir um estado `Task` que expande ou reduz um cluster para uma frota de instâncias.

```
"ModifyInstanceFleetByName": {
    "Type": "Task",
    "Resource": "arn:aws:states:::elasticmapreduce:modifyInstanceFleetByName",
    "Parameters": {
        "ClusterId": "j-1234567890123",
        "InstanceFleetName": "MyCoreFleet",
        "InstanceFleet": {
            "TargetOnDemandCapacity": 8,
            "TargetSpotCapacity": 0
        }
    },
    "End": true
}
```

Para obter informações sobre como configurar IAM permissões ao usar Step Functions com outros AWS serviços, consultePolíticas do IAM para serviços integrados.

Ligue para o Amazon EMR no EKS com AWS Step Functions

Step Functions pode controlar determinados AWS serviços diretamente do Amazon States Language (ASL). Para saber mais, consulte Como trabalhar com outros serviços e Transmitir parâmetros para uma API de serviço.

> Como a integração otimizada do Amazon EMR no EKS é diferente da integração do Amazon EMR no EKS SDK AWS
>
> - O padrão de integração Executar um trabalho (.sync) é compatível.
> - Não há otimizações para o padrão de integração Resposta de solicitação.
> - O padrão de integração Aguardar um retorno de chamada com um token de tarefa não é compatível.

> **Note**
>
> Para integração com o Amazon EMR, o Step Functions tem uma frequência de pesquisa de trabalhos de codificação rígida de 60 segundos para os primeiros 10 minutos e 300 segundos depois disso.

Para fazer a integração AWS Step Functions com o Amazon EMR no EKS, use as APIs de integração de serviços do Amazon EMR no EKS. As APIs de integração de serviços são iguais às APIs correspondentes do Amazon EMR no EKS, mas nem todas as APIs oferecem suporte a todos os padrões de integração, conforme mostrado na tabela a seguir.

API	Resposta de solicitação	Executar um trabalho (.sync)
CreateVirtualCluster	✓	
DeleteVirtualCluster	✓	✓
StartJobRun	✓	✓

Compatível com as APIs do Amazon EMR no EKS:

> **Note**
>
> Há uma cota para o tamanho máximo de dados de entrada ou resultado para uma tarefa no Step Functions. Isso restringe você a 256 KB de dados como uma string codificada em UTF-8 quando você envia ou recebe dados de outro serviço. Consulte Cotas relacionadas a execuções de máquina de estado.

- `CreateVirtualCluster`
  - Sintaxe da solicitação
  - Parâmetros compatíveis
  - Sintaxe da resposta
- `DeleteVirtualCluster`
  - Sintaxe da solicitação
  - Parâmetros compatíveis
  - Sintaxe da resposta
- `StartJobRun`
  - Sintaxe da solicitação
  - Parâmetros compatíveis

- Sintaxe da resposta

Veja a seguir um estado `Task` que cria um cluster 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
}
```

Veja a seguir um `Task` estado que envia um trabalho a um cluster virtual e aguarde até que ele seja concluído.

```
"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
}
```

Veja a seguir um `Task` estado que exclui um cluster virtual e aguarde a conclusão da exclusão.

```
"Delete_Virtual_Cluster": {
  "Type": "Task",
  "Resource": "arn:aws:states:::emr-containers:deleteVirtualCluster.sync",
  "Parameters": {
    "Id.$": "$.VirtualClusterId"
  },
  "End": true
}
```

Para obter informações sobre como configurar IAM permissões ao usar Step Functions com outros AWS serviços, consulte Políticas do IAM para serviços integrados.

Chamar o Amazon EMR Serverless com o Step Functions

Step Functions pode controlar determinados AWS serviços diretamente do Amazon States Language (ASL). Para saber mais, consulte Como trabalhar com outros serviços e Transmitir parâmetros para uma API de serviço.

> Como a integração otimizada do EMR Serverless é diferente da integração do AWS SDK do EMR Serverless
>
> - A integração otimizada de serviços EMR Serverless tem um conjunto personalizado de APIs que agrupam as APIs EMR Serverless subjacentes. Por causa dessa personalização, a EMR Serverless integração otimizada difere significativamente da integração do serviço EMR Serverless AWS SDK. Além disso, a integração otimizada do EMR Serverless é compatível com o padrão de integração do Executar um trabalho (.sync).
> - O padrão de integração Aguardar um retorno de chamada com um token de tarefa não é compatível.

Neste tópico

APIs de integração de serviço do EMR Serverless

Para integrar o AWS Step Functions ao EMR Serverless, use as seis APIs de integração de serviço do EMR Serverless apresentadas a seguir. As APIs de integração de serviço são semelhantes às APIs correspondentes do EMR Serverless, com algumas diferenças nos campos que são passados e nas respostas retornadas.

A tabela a seguir descreve as diferenças entre cada API de integração de serviço e a API do EMR Serverless correspondente.

APIs de integração de serviço do EMR Serverless e APIs correspondentes do EMR Serverless

API de integração de serviço do EMR Serverless	API do EMR Serverless correspondente	Diferenças
createApplication Cria um aplicativo. O EMR Serverless está vinculado a um tipo exclusivo de função do IAM conhecida como função vinculada ao serviço. Para que `createApplication` e `createApplication.sync` funcionem, você deve ter configurado as permissões necessárias para criar a função vinculada ao serviço `AWSServiceRoleForAmazonEMRServerless`. Para obter mais informações sobre isso, incluindo uma instrução de que você pode adicionar à política de permissões do IAM, consulte Usar funções vinculadas ao serviço para o EMR Serverless.	CreateApplication	Nenhum
createApplication.sync Cria um aplicativo.	CreateApplication	Não há diferenças entre as solicitações e as respostas da API do EMR Serverless e da API de integração de serviços do EMR Serverless. No entanto, createApplication.

API de integração de serviço do EMR Serverless	API do EMR Serverless correspondente	Diferenças
		sync espera que o aplicativo alcance o estado de `CREATED`.
startApplication Inicia um aplicativo especificado e inicializa a capacidade inicial do aplicativo, se configurado.	StartApplication	A resposta da API do EMR Serverless não contém nenhum dado, mas a resposta da API de integração de serviços do EMR Serverless inclui os seguintes dados. `{ "ApplicationId": "string" }`
startApplication.sync Inicia um aplicativo especificado e inicializa a capacidade inicial do aplicativo, se configurado.	StartApplication	A resposta da API do EMR Serverless não contém nenhum dado, mas a resposta da API de integração de serviços do EMR Serverless inclui os seguintes dados. `{ "ApplicationId": "string" }` Além disso, startApplication.sync espera que o aplicativo alcance o estado de `STARTED`.

API de integração de serviço do EMR Serverless	API do EMR Serverless correspondente	Diferenças
stopApplication Interrompe um aplicativo especificado e libera a capacidade inicial, se configurada. Todas as tarefas programadas e em execução devem ser concluídas ou canceladas antes de interromper um aplicativo.	StopApplication	A resposta da API do EMR Serverless não contém nenhum dado, mas a resposta da API de integração de serviços do EMR Serverless inclui os seguintes dados. `{ "ApplicationId": "string" }`
stopApplication.sync Interrompe um aplicativo especificado e libera a capacidade inicial, se configurada. Todas as tarefas programadas e em execução devem ser concluídas ou canceladas antes de interromper um aplicativo.	StopApplication	A resposta da API do EMR Serverless não contém nenhum dado, mas a resposta da API de integração de serviços do EMR Serverless inclui os seguintes dados. `{ "ApplicationId": "string" }` Além disso, stopApplication.sync espera que o aplicativo alcance o estado de `STOPPED`.

API de integração de serviço do EMR Serverless	API do EMR Serverless correspondente	Diferenças
deleteApplication Exclui um aplicativo. Um aplicativo deve estar no estado `STOPPED` ou `CREATED` para ser excluído.	DeleteApplication	A resposta da API do EMR Serverless não contém nenhum dado, mas a resposta da API de integração de serviços do EMR Serverless inclui os seguintes dados. `{ "ApplicationId": "string" }`
deleteApplication.sync Exclui um aplicativo. Um aplicativo deve estar no estado `STOPPED` ou `CREATED` para ser excluído.	DeleteApplication	A resposta da API do EMR Serverless não contém nenhum dado, mas a resposta da API de integração de serviços do EMR Serverless inclui os seguintes dados. `{ "ApplicationId": "string" }` Além disso, stopApplication.sync espera que o aplicativo alcance o estado de `TERMINATED` .
startJobRun Inicia uma execução de tarefa.	StartJobRun	Nenhum

API de integração de serviço do EMR Serverless	API do EMR Serverless correspondente	Diferenças
startJobRun.sync Inicia uma execução de tarefa.	StartJobRun	Não há diferenças entre as solicitações e as respostas da API do EMR Serverless e da API de integração de serviços do EMR Serverless. No entanto, startJobRun.sync espera que o aplicativo alcance o `SUCCESS` estado.
cancelJobRun Cancela uma execução de tarefa.	CancelJobRun	Nenhum
cancelJobRun.sync Cancela uma execução de tarefa.	CancelJobRun	Não há diferenças entre as solicitações e as respostas da API do EMR Serverless e da API de integração de serviços do EMR Serverless. No entanto, cancelJobRun.sync espera que o aplicativo alcance o `CANCELLED` estado.

Casos de uso da integração do EMR sem servidor

Para a integração otimizada de serviços do EMR Serverless, recomendamos que você crie um único aplicativo e, em seguida, use esse aplicativo para executar várias tarefas. Por exemplo, em uma única máquina de estado, você pode incluir várias startJobRunsolicitações, todas usando o mesmo aplicativo. Os exemplos de estado de Estado da tarefa a seguir mostram casos de uso para integrar as APIs do EMR Serverless com o Step Functions. Para obter informações sobre outros casos de uso do EMR Serverless, consulte O que é o Amazon EMR Serverless.

> **Tip**
>
> Para implantar um exemplo de uma máquina de estado que se integra EMR Serverless para executar várias tarefas em sua Conta da AWS, consulteExecutar uma tarefa do EMR Serverless.

- Cria uma aplicação
- Inicia o aplicativo
- Interrompe o aplicativo
- Deleta o aplicativo
- Inicia uma tarefa em um aplicativo
- Cancela uma tarefa em um aplicativo

Para obter informações sobre como configurar IAM permissões ao usar Step Functions com outros AWS serviços, consultePolíticas do IAM para serviços integrados.

Nos exemplos mostrados nos casos de uso a seguir, substitua o texto em *itálico* pelas informações específicas do seu recurso. Por exemplo, *yourApplicationId*substitua pelo ID do seu EMR Serverless aplicativo, como`00yv7iv71inak893`.

Cria uma aplicação

O exemplo de estado da Tarefa a seguir cria um aplicativo usando a API de integração do serviço 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
}
```

Inicia o aplicativo

O exemplo de estado da Tarefa a seguir inicia um aplicativo usando a API de integração do serviço startApplication.sync.

```
"Start_Application": {
    "Type": "Task",
    "Resource": "arn:aws:states:::emr-serverless:startApplication.sync",
    "Parameters": {
        "ApplicationId": "yourApplicationId"
    },
    "End": true
}
```

Interrompe o aplicativo

O exemplo de estado da Tarefa a seguir interrompe um aplicativo usando a API de integração do serviço stopApplication.sync.

```
"Stop_Application": {
    "Type": "Task",
    "Resource": "arn:aws:states:::emr-serverless:stopApplication.sync",
    "Parameters": {
        "ApplicationId": "yourApplicationId"
    },
    "End": true
}
```

Deleta o aplicativo

O exemplo de estado da Tarefa a seguir deleta um aplicativo usando a API de integração do serviço deleteApplication.sync.

```
"Delete_Application": {
    "Type": "Task",
    "Resource": "arn:aws:states:::emr-serverless:deleteApplication.sync",
    "Parameters": {
        "ApplicationId": "yourApplicationId"
    },
    "End": true
}
```

Inicia uma tarefa em um aplicativo

O exemplo de estado de tarefa a seguir inicia um trabalho em um aplicativo usando a API de integração de serviços 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
}
```

Cancela uma tarefa em um aplicativo

O exemplo de estado da tarefa a seguir cancela um trabalho em um aplicativo usando a API de integração de serviços cancelJobRun.sync.

```
"Cancel_Job": {
    "Type": "Task",
    "Resource": "arn:aws:states:::emr-serverless:cancelJobRun.sync",
    "Parameters": {
        "ApplicationId.$": "$.ApplicationId",
        "JobRunId.$": "$.JobRunId"
    },
    "End": true
}
```

Chamada EventBridge com Step Functions

Step Functions pode controlar determinados AWS serviços diretamente do Amazon States Language (ASL). Para saber mais, consulte Como trabalhar com outros serviços e Transmitir parâmetros para uma API de serviço.

> Como a EventBridge integração otimizada é diferente da integração do EventBridge AWS SDK
>
> - O ARN de execução e o ARN da máquina de estado são automaticamente anexados ao campo `Resources` de cada `PutEventsRequestEntry`.
> - Se a resposta de `PutEvents` apresentar um valor diferente de zero `FailedEntryCount`, então o estado `Task` falhará com o erro `EventBridge.FailedEntry`.

Para obter informações sobre como configurar IAM permissões ao usar Step Functions com outros AWS serviços, consultePolíticas do IAM para serviços integrados.

O Step Functions fornece uma API de integração de serviços para integração com a Amazon EventBridge. Isso permite a criação de aplicativos orientados a eventos enviando eventos personalizados diretamente dos fluxos de trabalho do Step Functions.

Para usar a `PutEvents` API, você precisará criar uma EventBridge regra em sua conta que corresponda ao padrão específico dos eventos que você enviará. Por exemplo, você pode:

- Crie uma função Lambda em sua conta que receba e imprima um evento que corresponda a uma EventBridge regra.
- Crie uma EventBridge regra em sua conta no barramento de eventos padrão que corresponda a um padrão de evento específico e tenha como alvo a função Lambda.

Para obter mais informações, consulte:

- Adicionar EventBridge eventos da Amazon PutEvents no Guia do EventBridge usuário.
- Aguardar um retorno de chamada com um token de tarefa nos Padrões de Integração de Serviço.

> **Note**
>
> Há uma cota para o tamanho máximo de dados de entrada ou resultado para uma tarefa no Step Functions. Isso restringe você a 256 KB de dados como uma string codificada em UTF-8 quando você envia ou recebe dados de outro serviço. Consulte Cotas relacionadas a execuções de máquina de estado.

EventBridge API compatível

A EventBridge API e a sintaxe suportadas incluem:

- `PutEvents`
  - Sintaxe da solicitação
  - Parâmetros compatíveis:
    - `Entries`
  - Sintaxe da resposta

Veja a seguir um `Task` que envia um evento personalizado:

```
{
  "Type": "Task",
  "Resource": "arn:aws:states:::events:putEvents",
  "Parameters": {
    "Entries": [
      {
        "Detail": {
          "Message": "MyMessage"
        },
        "DetailType": "MyDetailType",
        "EventBusName": "MyEventBus",
        "Source": "my.source"
      }
    ]
  },
  "End": true
}
```

Tratamento de erros

A `PutEvents` API aceita uma matriz de entradas como entrada e, em seguida, retorna uma matriz de entradas de resultados. Desde que a ação `PutEvents` tenha sido bem-sucedida, `PutEvents` retornará uma resposta HTTP 200, mesmo se uma ou mais entradas falharem. O `PutEvents` retorna o número de entradas com falha no campo `FailedEntryCount`.

O Step Functions verifica se o `FailedEntryCount` é maior que zero. Se for maior que zero, o Step Functions falhará no estado com o erro `EventBridge.FailedEntry`. Isso permite que você use o tratamento de erros incorporado do Step Functions nos estados da tarefa para capturar ou tentar novamente quando houver falhas nas entradas, em vez de precisar usar um estado adicional para analisar a `FailedEntryCount` da resposta.

> **Note**
>
> Se você implementou a idempotência e pode tentar novamente com segurança em todas as entradas, é possível usar a lógica de repetição do Step Functions. O Step Functions não remove entradas bem-sucedidas da matriz de entrada de `PutEvents` antes de tentar novamente. Em vez disso, ele tenta novamente com a matriz original de entradas.

Gerencie AWS Glue trabalhos com Step Functions

Step Functions pode controlar determinados AWS serviços diretamente do Amazon States Language (ASL). Para saber mais, consulte Como trabalhar com outros serviços e Transmitir parâmetros para uma API de serviço.

> **Note**
>
> Como a AWS Glue integração otimizada é diferente da integração do AWS GlueAWS SDK
>
> - O padrão de integração Executar um trabalho (.sync) está disponível.
> - O `JobName` campo é extraído da solicitação e inserido na resposta, que normalmente contém apenas `JobRunID`.

AWS Glue API suportada:

- `StartJobRun`

> **Os parâmetros em Step Functions são expressos em PascalCase**
>
> Mesmo que a API de serviço nativa esteja no CamelCase, por exemplo, a `startSyncExecution` ação da API, você especifica parâmetros PascalCase em, como:. `StateMachineArn`

O seguinte inclui um `Task` estado que inicia um AWS Glue trabalho.

```
"Glue StartJobRun": {
      "Type": "Task",
      "Resource": "arn:aws:states:::glue:startJobRun.sync",
      "Parameters": {
        "JobName": "GlueJob-JTrRO5l98qMG"
      },
      "Next": "ValidateOutput"
    },
```

Para obter informações sobre como configurar IAM permissões ao usar Step Functions com outros AWS serviços, consulte Políticas do IAM para serviços integrados.

Gerencie AWS Glue DataBrew trabalhos com Step Functions

Step Functions pode controlar determinados AWS serviços diretamente do Amazon States Language (ASL). Para saber mais, consulte Como trabalhar com outros serviços e Transmitir parâmetros para uma API de serviço.

Você pode usar a DataBrew integração para adicionar etapas de limpeza e normalização de dados aos seus fluxos de trabalho de análise e aprendizado de máquina.

DataBrew API suportada:

- `StartJobRun`

O seguinte inclui um `Task` estado que inicia um trabalho de solicitação-resposta DataBrew.

```
"DataBrew StartJobRun": {
            "Type": "Task",
            "Resource": "arn:aws:states:::databrew:startJobRun",
```

```
            "Parameters": {
               "Name": "sample-proj-job-1"
            },
            "Next": "NEXT_STATE"
          },
```

O seguinte inclui um `Task` estado que inicia um DataBrew trabalho de sincronização.

```
"DataBrew StartJobRun": {
           "Type": "Task",
           "Resource": "arn:aws:states:::databrew:startJobRun.sync",
           "Parameters": {
              "Name": "sample-proj-job-1"
           },
           "Next": "NEXT_STATE"
          },
```

Para obter informações sobre como configurar IAM permissões ao usar Step Functions com outros AWS serviços, consulte[Políticas do IAM para serviços integrados](#).

Invocar o Lambda com o Step Functions

Step Functions pode controlar determinados AWS serviços diretamente do [Amazon States Language](#) (ASL). Para saber mais, consulte [Como trabalhar com outros serviços](#) e [Transmitir parâmetros para uma API de serviço](#).

> Como a integração otimizada do Lambda é diferente da integração do SDK do Lambda AWS
>
> - O campo `Payload` da resposta é analisado de Json escapado para Json.
> - Se a resposta possuir o campo `FunctionError` ou uma exceção for gerada na função do Lambda, a tarefa falhará.

Para obter mais informações sobre o gerenciamento da entrada, saída e dos resultados do estado, consulte [Processamento de entrada e saída no Step Functions](#).

AWS Lambda APIs compatíveis:

- Invoke
  - Sintaxe da solicitação
  - Parâmetros compatíveis
    - ClientContext
    - FunctionName
    - InvocationType
    - Qualifier
    - Payload
  - Sintaxe da resposta

> **Os parâmetros em Step Functions são expressos em PascalCase**
>
> Mesmo que a API de serviço nativa esteja no CamelCase, por exemplo, a `startSyncExecution` ação da API, você especifica parâmetros PascalCase em, como:. `StateMachineArn`

Veja a seguir um estado `Task` que invoca uma função do 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
      }
   }
}
```

Veja a seguir um estado `Task` que implementa o padrão de integração do serviço retorno de chamada.

```
{
```

```
   "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
      }
   }
}
```

Ao invocar uma função do Lambda, a execução espera a conclusão da função. Se você invocar a função do Lambda com uma tarefa de retorno de chamada, o tempo limite de pulsação não começará a ser contado até que a função do Lambda tenha concluído a execução e retornado um resultado. Enquanto a função do Lambda for executada, o tempo limite de pulsação não será aplicado.

Também é possível chamar o Lambda de forma assíncrona usando o parâmetro `InvocationType`, conforme mostrado no exemplo a seguir.

> **Note**
>
> Para invocações assíncronas de funções do Lambda, o período do tempo limite de pulsação começa imediatamente.

```
{

  "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
    }
  }
}
```

Quando o resultado da `Task` é retornado, a saída da função é aninhada em um dicionário de metadados. Por exemplo: .

```
{

   "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, você pode invocar uma função do Lambda especificando o ARN da função diretamente no campo “Recurso”. Ao invocar uma função do Lambda dessa forma, você não pode especificar a `.waitForTaskToken` e o resultado da tarefa contém somente a saída da função.

```
{
   "StartAt":"CallFunction",
```

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

É possível invocar determinada versão ou alias da função do Lambda especificando essas opções no ARN no campo `Resource`. Consulte a seguir na documentação do Lambda:

- Versionamento do AWS Lambda
- AWS Lambda pseudônimos

Para obter informações sobre como configurar IAM permissões ao usar Step Functions com outros AWS serviços, consulte Políticas do IAM para serviços integrados.

Gerencie AWS Elemental MediaConvert com Step Functions

> **Experimente com Step Functions e MediaConvert**
>
> Saiba como usar a integração MediaConvert otimizada em um fluxo de trabalho que detecta e remove barras coloridas SMTPE de tamanho desconhecido desde o início de um videoclipe. Leia a postagem do blog de 12 de abril de 2024: Fluxos de trabalho de baixo código com AWS Elemental MediaConvert

Step Functions pode controlar determinados AWS serviços diretamente do Amazon States Language (ASL). Para saber mais, consulte Como trabalhar com outros serviços e Transmitir parâmetros para uma API de serviço.

> **Como a integração otimizada é diferente da integração padrão do AWS SDK**
>
> - O padrão de integração Executar um trabalho (.sync) está disponível.
> - Sem otimizações Resposta de solicitação ou padrões de Aguardar um retorno de chamada com um token de tarefa integração.

MediaConvert APIs compatíveis:

- `CreateJob`
  - Sintaxe da solicitação
  - Parâmetros compatíveis:
    - `Role` (obrigatório)
    - `Settings` (obrigatório)
    - `CreateJobRequest` (Opcional)
  - Sintaxe de resposta — veja o esquema CreateJobResponse

O seguinte inclui um `Task` estado que envia um MediaConvert trabalho e aguarda sua conclusão.

```
{
    "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 obter informações sobre como configurar IAM permissões ao usar Step Functions com MediaConvert, consulte[Políticas do IAM para AWS Elemental MediaConvert](#).

> **Os parâmetros em Step Functions são expressos em PascalCase**
>
> Mesmo que a API de serviço nativa esteja no CamelCase, por exemplo, a `startSyncExecution` ação da API, você especifica parâmetros PascalCase em, como:. `StateMachineArn`

Gerencie SageMaker com Step Functions

Step Functions pode controlar determinados AWS serviços diretamente do Amazon States Language (ASL). Para saber mais, consulte Como trabalhar com outros serviços e Transmitir parâmetros para uma API de serviço.

> **Como a SageMaker integração otimizada é diferente da integração do SageMaker AWS SDK**
>
> - O padrão de integração Executar um trabalho (.sync) é compatível.
> - Não há otimizações para o padrão de integração Resposta de solicitação.
> - O padrão de integração Aguardar um retorno de chamada com um token de tarefa não é compatível.

SageMaker APIs e sintaxe suportadas:

- `CreateEndpoint`
  - Sintaxe da solicitação
  - Parâmetros compatíveis:
    - `EndpointConfigName`
    - `EndpointName`
    - `Tags`
  - Sintaxe da resposta
- `CreateEndpointConfig`
  - Sintaxe da solicitação
  - Parâmetros compatíveis:
    - `EndpointConfigName`
    - `KmsKeyId`

    - ProductionVariants
    - Tags
  - Sintaxe da resposta
- CreateHyperParameterTuningJob

> **Note**
>
> Essa ação da API é compatível com o padrão de integração .sync.

  - Sintaxe da solicitação
  - Parâmetros compatíveis:
    - HyperParameterTuningJobConfig
    - HyperParameterTuningJobName
    - Tags
    - TrainingJobDefinition
    - WarmStartConfig
  - Sintaxe da resposta
- CreateLabelingJob

> **Note**
>
> Essa ação da API é compatível com o padrão de integração .sync.

  - Sintaxe da solicitação
  - Parâmetros compatíveis:
    - HumanTaskConfig
    - InputConfig
    - LabelAttributeName
    - LabelCategoryConfigS3Uri
    - LabelingJobAlgorithmsConfig
    - LabelingJobName

    - OutputConfig
    - RoleArn
    - StoppingConditions
    - Tags
  - Sintaxe da resposta
- CreateModel
  - Sintaxe da solicitação
  - Parâmetros compatíveis:
    - Containers
    - EnableNetworkIsolation
    - ExecutionRoleArn
    - ModelName
    - PrimaryContainer
    - Tags
    - VpcConfig
- CreateProcessingJob

> **Note**
>
> Essa ação da API é compatível com o padrão de integração .sync.

  - Sintaxe da solicitação
  - Parâmetros compatíveis:
    - AppSpecification
    - Environment
    - ExperimentConfig
    - NetworkConfig
    - ProcessingInputs
    - ProcessingJobName
    - ProcessingOutputConfig
    - ProcessingResources

    - RoleArn
    - StoppingCondition
    - Tags
  - Sintaxe da resposta
- CreateTrainingJob

> **Note**
>
> Essa ação da API é compatível com o padrão de integração .sync.

  - Sintaxe da solicitação
  - Parâmetros compatíveis:
    - AlgorithmSpecification
    - HyperParameters
    - InputDataConfig
    - OutputDataConfig
    - ResourceConfig
    - RoleArn
    - StoppingCondition
    - Tags
    - TrainingJobName
    - VpcConfig
  - Sintaxe da resposta
- CreateTransformJob

> **Note**
>
> Essa ação da API é compatível com o padrão de integração .sync.

> **Note**
>
> AWS Step Functions não criará automaticamente uma política para`CreateTransformJob`. É necessário anexar uma política em linha à função criada. Para obter mais informações, veja este exemplo de política do IAM: `CreateTrainingJob`.

- Sintaxe da solicitação
- Parâmetros compatíveis:
  - `BatchStrategy`
  - `Environment`
  - `MaxConcurrentTransforms`
  - `MaxPayloadInMB`
  - `ModelName`
  - `Tags`
  - `TransformInput`
  - `TransformJobName`
  - `TransformOutput`
  - `TransformResources`
- Sintaxe da resposta

- `UpdateEndpoint`
  - Sintaxe da solicitação
  - Parâmetros compatíveis:
    - `EndpointConfigName`
    - `EndpointName`
  - Sintaxe da resposta

SageMaker Exemplo de Transform Job

O seguinte inclui um `Task` estado que cria uma tarefa de SageMaker transformação da Amazon, especificando a localização `DataSource` do Amazon S3 para e. `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 Exemplo de Job de Treinamento

O seguinte inclui um `Task` estado que cria um trabalho de SageMaker treinamento na 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 Exemplo de Job de Etiquetagem

O seguinte inclui um `Task` estado que cria um trabalho de SageMaker etiquetagem na 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 Exemplo de Trabalho de Processamento

O seguinte inclui um `Task` estado que cria um trabalho de SageMaker processamento da 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 obter informações sobre como configurar IAM permissões ao usar Step Functions com outros AWS serviços, consulte Políticas do IAM para serviços integrados.

Chamar o Amazon SNS com o Step Functions

Step Functions pode controlar determinados AWS serviços diretamente do Amazon States Language (ASL). Para saber mais, consulte Como trabalhar com outros serviços e Transmitir parâmetros para uma API de serviço.

> **Como a integração otimizada do Amazon SNS é diferente da integração do Amazon AWS SNS SDK**
>
> Não há otimizações para os padrões de integração Resposta de solicitação ou Aguardar um retorno de chamada com um token de tarefa.

APIs compatíveis do Amazon SNS:

> **Note**
>
> Há uma cota para o tamanho máximo de dados de entrada ou resultado para uma tarefa no Step Functions. Isso restringe você a 256 KB de dados como uma string codificada em UTF-8 quando você envia ou recebe dados de outro serviço. Consulte Cotas relacionadas a execuções de máquina de estado.

- `Publish`
  - Sintaxe da solicitação
  - Parâmetros compatíveis
    - `Message`
    - `MessageAttributes`
    - `MessageStructure`
    - `PhoneNumber`
    - `Subject`

- `TargetArn`
- `TopicArn`

- Sintaxe da resposta

> **Os parâmetros em Step Functions são expressos em PascalCase**
>
> Mesmo que a API de serviço nativa esteja no CamelCase, por exemplo, a `startSyncExecution` ação da API, você especifica parâmetros PascalCase em, como:. `StateMachineArn`

Veja a seguir um estado `Task` que publica em um tópico do 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
    }
  }
}
```

Aprovação de valores dinâmicos. Você pode modificar o exemplo acima para aprovar dinamicamente um atributo dessa carga JSON:

```
{
"input": {
  "message": "Hello world"
},
"SNSDetails": {
  "attribute1": "some value",
  "attribute2": "some other value",
}
}
```

Anexe o `.$` ao campo `StringValue`:

```
"MessageAttributes": {
  "my_attribute_no_1": {
      "DataType": "String",
      "StringValue.$": "$.SNSDetails.attribute1"
  },
  "my_attribute_no_2": {
      "DataType": "String",
      "StringValue.$": "$.SNSDetails.attribute2"
  }
```

Veja a seguir um estado `Task` que publica em um tópico do Amazon SNS e aguarda o retorno do token da tarefa. Consulte Aguardar um retorno de chamada com um token de tarefa.

```
{
   "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 obter informações sobre como configurar IAM permissões ao usar Step Functions com outros AWS serviços, consulte Políticas do IAM para serviços integrados.

Chamar o Amazon SQS com o Step Functions

Step Functions pode controlar determinados AWS serviços diretamente do Amazon States Language (ASL). Para saber mais, consulte Como trabalhar com outros serviços e Transmitir parâmetros para uma API de serviço.

> Como a integração otimizada do Amazon SQS é diferente da integração do Amazon SQS SDK AWS
>
> Não há otimizações para os padrões de integração Resposta de solicitação ou Aguardar um retorno de chamada com um token de tarefa.

APIs compatíveis do Amazon SQS:

> Note
>
> Há uma cota para o tamanho máximo de dados de entrada ou resultado para uma tarefa no Step Functions. Isso restringe você a 256 KB de dados como uma string codificada em UTF-8 quando você envia ou recebe dados de outro serviço. Consulte Cotas relacionadas a execuções de máquina de estado.

- `SendMessage`

  Parâmetros compatíveis:

  - `DelaySeconds`
  - `MessageAttribute`
  - `MessageBody`
  - `MessageDeduplicationId`
  - `MessageGroupId`

  - QueueUrl
- Response syntax

> **Os parâmetros em Step Functions são expressos em PascalCase**
>
> Mesmo que a API de serviço nativa esteja no CamelCase, por exemplo, a `startSyncExecution` ação da API, você especifica parâmetros PascalCase em, como:. `StateMachineArn`

Veja a seguir um estado `Task` que envia uma mensagem do 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
    }
  }
}
```

Veja a seguir um estado `Task` que publica em uma fila do Amazon SQS e aguarda o retorno do token da tarefa. Consulte Aguardar um retorno de chamada com um token de tarefa.

```
{
   "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 saber mais sobre o recebimento de mensagens no Amazon SQS, consulte Receber e Excluir Sua Mensagem no Guia do Desenvolvedor do Amazon Simple Queue Service.

Para obter informações sobre como configurar IAM permissões ao usar Step Functions com outros AWS serviços, consultePolíticas do IAM para serviços integrados.

Gerencie AWS Step Functions execuções como um serviço integrado

O Step Functions se integra à sua própria API como uma integração de serviço. Isso permite que o Step Functions inicie uma nova execução de uma máquina de estado diretamente do estado da tarefa de uma execução em andamento. Ao criar novos fluxos de trabalho, use execuções de fluxo de trabalho aninhado para reduzir a complexidade dos fluxos de trabalho principais e reutilizar processos comuns.

Como a integração do Optimized Step Functions é diferente da integração do AWS SDK do Step Functions

- O padrão de integração Executar um trabalho (.sync) está disponível.

Observe que não há otimizações para os padrões de integração Resposta de solicitação ou Aguardar um retorno de chamada com um token de tarefa.

Para mais informações, consulte:

- Iniciar execuções de uma tarefa
- Como trabalhar com outros serviços
- Transmitir parâmetros para uma API de serviço

APIs e sintaxe compatíveis com o Step Functions:

- `StartExecution`
  - Sintaxe da solicitação
  - Parâmetros compatíveis
    - `Input`
    - `Name`
    - `StateMachineArn`
  - Sintaxe da resposta

Veja a seguir um estado `Task` que inicia uma execução de outra máquina de estado e aguarda até que ela seja concluída.

```
{
   "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
}
```

Veja a seguir um estado `Task` que inicia uma execução de outra 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
}
```

Veja a seguir um estado `Task` que implementa o padrão de integração do serviço retorno de chamada.

```
{
   "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 associar a execução de um fluxo de trabalho aninhado à execução principal que o iniciou, transmita um parâmetro especialmente nomeado que inclua o ID de execução obtido do objeto de contexto. Ao iniciar uma execução aninhada, use um parâmetro chamado `AWS_STEP_FUNCTIONS_STARTED_BY_EXECUTION_ID`. Transmita o ID de execução anexando `.$` ao nome do parâmetro e fazendo referência ao ID no objeto de contexto com `$$.Execution.Id`. Para ter mais informações, consulte Acessar o 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
}
```

Máquinas de estado aninhadas retornam o seguinte:

Recurso	Saída
startExecution.sync	String
startExecution.sync:2	JSON

Ambos aguardarão a conclusão da máquina de estado aninhado, mas eles retornam formatos diferentes de `Output`. Por exemplo, se você criar uma função do Lambda que retorna o objeto `{ "MyKey": "MyValue" }`, você obteria as seguintes respostas:

Para startExecution.sync:

```
{
   <other fields>
   "Output": "{ \"MyKey\": \"MyValue\" }"
}
```

Para startExecution.sync:2:

```
{
   <other fields>
   "Output": {
      "MyKey": "MyValue"
   }
}
```

Configurando permissões do IAM para máquinas de estado aninhado

Uma máquina de estado principal determina se uma máquina de estado secundária concluiu a execução usando pesquisas e eventos. A enquete requer permissão, `states:DescribeExecution` enquanto os eventos enviados EventBridge para o Step Functions exigem permissões para `events:PutTargets``events:PutRule`, e. `events:DescribeRule` Se essas permissões estiverem ausentes em seu perfil do IAM, pode haver um atraso até que uma máquina de estado principal tome conhecimento da conclusão da execução da máquina de estado secundária.

Para uma máquina de estado que chama `StartExecution` para uma única execução de fluxo de trabalho aninhado, use uma política do IAM que limita as permissões a essa máquina de estado.

Para obter mais informações, consulte Permissões do IAM para o Step Functions.

Chamar APIs de terceiros

Uma tarefa HTTP é um tipo de estado Estado da tarefa que possibilita chamar qualquer API pública de terceiros, como Salesforce e Stripe, nos fluxos de trabalho. Para chamar uma API de terceiros, use o estado Tarefa com o recurso `arn:aws:states:::http:invoke`. Depois, forneça os detalhes da configuração do endpoint da API, como o URL da API, o método a ser utilizado e os detalhes da autenticação.

Se você usa o Workflow Studio para criar a máquina de estado que contém uma tarefa HTTP, o Workflow Studio gera automaticamente um perfil de execução com as políticas do IAM para a tarefa HTTP. Para ter mais informações, consulte Perfil para testar tarefas HTTP no Workflow Studio.

Tópicos

- Definição da tarefa HTTP
- Campos da tarefa HTTP
- Autenticação para uma tarefa HTTP
- Mesclar a conexão com o EventBridge e os dados da definição de tarefa HTTP
- Aplicar a codificação em URL no corpo da solicitação
- Permissões do IAM para executar uma tarefa HTTP
- Exemplo de tarefa HTTP
- Testar uma tarefa HTTP

- Respostas da tarefa HTTP não compatíveis

Definição da tarefa HTTP

A definição de ASL representa uma tarefa HTTP com o recurso `http:invoke`. A definição da tarefa HTTP a seguir invoca uma API do Stripe que exibe uma lista de todos os 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 da tarefa HTTP

Uma tarefa HTTP inclui os seguintes campos na definição.

`Resource` (obrigatório)

Para especificar um tipo de tarefa, forneça o ARN no campo `Resource`. Para uma tarefa HTTP, você vai especificar o campo `Resource` da forma a seguir.

```
"Resource": "arn:aws:states:::http:invoke"
```

`Parameters` (obrigatório)

Contém os campos `ApiEndpoint`, `Method` e `ConnectionArn` que fornecem informações sobre a API de terceiros a ser chamada. `Parameters` também contém campos opcionais, como `Headers` e `QueryParameters`.

Você pode especificar uma combinação de JSON estático e JsonPathsintaxe como `Parameters` no `Parameters` campo. Para ter mais informações, consulte Transmitir parâmetros para uma API de serviço.

`ApiEndpoint` (obrigatório)

Especifica o URL da API de terceiros a ser chamada. Para acrescentar parâmetros de consulta ao URL, use o campo `QueryParameters`. O exemplo a seguir mostra como chamar uma API do Stripe para obter a lista de todos os clientes.

```
"ApiEndpoint":"https://api.stripe.com/v1/customers"
```

Você também pode especificar um caminho de referência usando a JsonPathsintaxe para selecionar o nó JSON que contém o URL da API de terceiros. Por exemplo, digamos que você queira chamar uma das APIs do Stripe usando o ID de um cliente específico. Imagine que você tenha fornecido a entrada de estado a seguir.

```
{
    "customer_id": "1234567890",
    "name": "John Doe"
}
```

Para recuperar os detalhes do ID desse cliente usando uma API do Stripe, especifique o `ApiEndpoint` conforme mostrado no exemplo a seguir. Este exemplo usa uma função intrínseca e um caminho de referência.

```
"ApiEndpoint.$":"States.Format('https://api.stripe.com/v1/customers/{}',
 $.customer_id)"
```

Em runtime, o Step Functions resolve o valor de `ApiEndpoint` da forma a seguir.

```
https://api.stripe.com/v1/customers/1234567890
```

`Method` (obrigatório)

Especifica o método HTTP a ser utilizado para chamar uma API de terceiros. É possível especificar um desses métodos na tarefa HTTP: GET, POST, PUT, DELETE, PATCH, OPTIONS ou HEAD.

Por exemplo, para usar o método GET, especifique o campo `Method` da forma a seguir.

```
"Method": "GET"
```

Também é possível usar um caminho de referência para especificar o método em runtime. Por exemplo, **`"Method.$": "$.myHTTPMethod"`**.

`Authentication` (obrigatório)

Contém o campo `ConnectionArn` que especifica como autenticar uma chamada de API de terceiros. O Step Functions é compatível com autenticação para um `ApiEndpoint` especificado usando o recurso de conexão com o Amazon EventBridge.

`ConnectionArn` (obrigatório)

Especifica o ARN da conexão com o EventBridge.

Uma tarefa HTTP requer uma EventBridge conexão, que gerencia com segurança as credenciais de autenticação de um provedor de API. Uma conexão especifica o tipo de autorização e as credenciais a serem utilizadas para autorizar uma API de terceiros. Usar uma conexão ajuda a evitar segredos de codificação rígida, como chaves de API, na definição da máquina de estado. Em uma conexão, também é possível especificar os parâmetros `Headers`, `QueryParameters` e `RequestBody`.

Ao criar uma EventBridge conexão, você fornece seus detalhes de autenticação. Para obter mais informações sobre como a autenticação funciona para uma tarefa HTTP, consulte Autenticação para uma tarefa HTTP.

O exemplo a seguir mostra como especificar o campo `Authentication` na definição da tarefa HTTP.

```
"Authentication": {
  "ConnectionArn": "arn:aws:events:us-east-2:123456789012:connection/
Stripe/81210c42-8af1-456b-9c4a-6ff02fc664ac"
}
```

`Headers` (Opcional)

Fornece contexto adicional e metadados ao endpoint da API. É possível especificar cabeçalhos como uma string ou uma matriz JSON.

É possível especificar cabeçalhos na conexão com o EventBridge e no campo `Headers` em uma tarefa HTTP. É recomendável não incluir detalhes de autenticação dos provedores de API no campo `Headers`. Recomendamos incluir esses detalhes na conexão com o EventBridge.

O Step Functions adiciona os cabeçalhos da conexão com o EventBridge aos cabeçalhos na especificados na definição da tarefa HTTP. Se as mesmas chaves de cabeçalho estiverem presentes na definição e na conexão, o Step Functions usará os valores correspondentes especificados na conexão com o EventBridge nesses cabeçalhos. Para obter mais informações sobre como o Step Functions realiza a mesclagem de dados, consulte Mesclar a conexão com o EventBridge e os dados da definição de tarefa HTTP.

O exemplo a seguir especifica um cabeçalho que será incluído em uma chamada de API de terceiros: `content-type`.

```
"Headers": {
  "content-type": "application/json"
}
```

Também é possível usar um caminho de referência para especificar os cabeçalhos em runtime. Por exemplo, **`"Headers.$": "$.myHTTPHeaders"`**.

O Step Functions define os cabeçalhos `User-Agent`, `Range` e `Host`. O Step Functions define o valor do cabeçalho `Host` com base na API que você está chamando. Veja a seguir um exemplo de valor desses cabeçalhos.

```
User-Agent: Amazon|StepFunctions|HttpInvoke|us-east-1,
Range: bytes=0-262144,
Host: api.stripe.com
```

Não é possível usar os cabeçalhos a seguir na definição da tarefa HTTP. Se você usar esses cabeçalhos, a tarefa HTTP falhará com o erro `States.Runtime`.

- A-IM
- Accept-Charset
- Accept-Datetime
- Accept-Encoding
- Cache-Control
- Conexão
- Content-Encoding
- Conteúdo-MD5
- Data

- Expect
- Encaminhado
- De
- Host
- HTTP2-Settings
- If-Match
- If-Modified-Since
- If-None-Match
- If-Range
- If-Unmodified-Since
- Max-Forwards
- Origem
- Pragma
- Proxy-Authorization
- Referer
- Servidor
- TE
- Trailer
- Transfer-Encoding
- Upgrade
- Via
- Aviso
- x-forwarded-*
- x-amz-*
- x-amzn-*

`QueryParameters` (Opcional)

Insere pares de chave-valor no final de um URL de API. É possível especificar parâmetros de consulta como uma string, uma matriz JSON ou um objeto JSON. O Step Functions codifica automaticamente os parâmetros de consulta em URL ao chamar uma API de terceiros.

Por exemplo, digamos que você queira chamar a API do Stripe para pesquisar clientes que realizam transações em dólares americanos (USD). Imagine que você tenha fornecido `QueryParameters` a seguir como a entrada de estado.

```
"QueryParameters": {
  "currency": "usd"
}
```

Em runtime, o Step Functions anexa o `QueryParameters` ao URL da API da forma a seguir.

```
https://api.stripe.com/v1/customers/search?currency=usd
```

Também é possível usar um caminho de referência para especificar os parâmetros de consulta em runtime. Por exemplo, **`"QueryParameters.$": "$.myQueryParameters"`**.

Se você tiver especificado parâmetros de consulta na conexão com o EventBridge, o Step Functions adicionará esses parâmetros aos parâmetros de consulta especificados na definição da tarefa HTTP. Se as mesmas chaves de parâmetros de consulta estiverem presentes na definição e na conexão, o Step Functions usará os valores correspondentes especificados na conexão com o EventBridge nesses cabeçalhos. Para obter mais informações sobre como o Step Functions realiza a mesclagem de dados, consulte Mesclar a conexão com o EventBridge e os dados da definição de tarefa HTTP.

`Transform` (Opcional)

Contém os cabeçalhos `RequestBodyEncoding` e `RequestEncodingOptions`. Por padrão, o Step Functions envia o corpo da solicitação como dados JSON a um endpoint da API.

Se o provedor de API aceitar corpos de solicitação `form-urlencoded`, use o campo `Transform` para especificar a codificação de URL para os corpos de solicitação. Também é necessário especificar o cabeçalho `content-type` como `application/x-www-form-urlencoded`. Depois, o Step Functions codifica automaticamente o corpo da solicitação em URL.

`RequestBodyEncoding`

Especifica a codificação do corpo da solicitação em URL. É possível especificar um destes valores: `NONE` ou `URL_ENCODED`.

- `NONE`: o corpo da solicitação HTTP será o JSON serializado do campo `RequestBody`. Este é o valor padrão.

- `URL_ENCODED`: o corpo da solicitação HTTP serão os dados do formulário codificados em URL do campo `RequestBody`.

`RequestEncodingOptions`

Determina a opção de codificação a ser usada para matrizes no corpo da solicitação, se você definir `RequestBodyEncoding` como `URL_ENCODED`.

O Step Functions é compatível com as opções de codificação de matriz a seguir. Para obter mais informações sobre essas opções e exemplos, consulte Aplicar a codificação em URL no corpo da solicitação.

- `INDICES`: codifica matrizes usando o valor do índice dos elementos da matriz. Por padrão, o Step Functions usa essa opção de codificação.
- `REPEAT`: repete uma chave para cada item em uma matriz.
- `COMMAS`: codifica todos os valores em uma chave como uma lista de valores delimitada por vírgula.
- `BRACKETS`: repete uma chave para cada item em uma matriz e acrescenta um colchete, [], à chave para indicar que é uma matriz.

O exemplo a seguir define a codificação em URL dos dados do corpo da solicitação. Também especifica o uso da opção de codificação `COMMAS` para matrizes no corpo da solicitação.

```
"Transform": {
  "RequestBodyEncoding": "URL_ENCODED",
  "RequestEncodingOptions": {
    "ArrayFormat": "COMMAS"
  }
}
```

`RequestBody` (Opcional)

Aceita dados JSON fornecidos na entrada de estado. Em`RequestBody`, você pode especificar uma combinação de JSON estático e JsonPathsintaxe. Por exemplo, vamos supor que você forneça a seguinte entrada de estado:

```
{
    "CardNumber": "1234567890",
    "ExpiryDate": "09/25"
}
```

Para usar esses valores de `CardNumber` e `ExpiryDate` no corpo da solicitação em runtime, é possível especificar os dados JSON a seguir no corpo da solicitação.

```
"RequestBody": {
  "Card": {
    "Number.$": "$.CardNumber",
    "Expiry.$": "$.ExpiryDate",
    "Name": "John Doe",
    "Address": "123 Any Street, Any Town, USA"
  }
}
```

Se a API de terceiros a ser chamada exigir corpos de solicitação `form-urlencoded`, será necessário especificar a codificação em URL dos dados do corpo da solicitação. Para ter mais informações, consulte Aplicar a codificação em URL no corpo da solicitação.

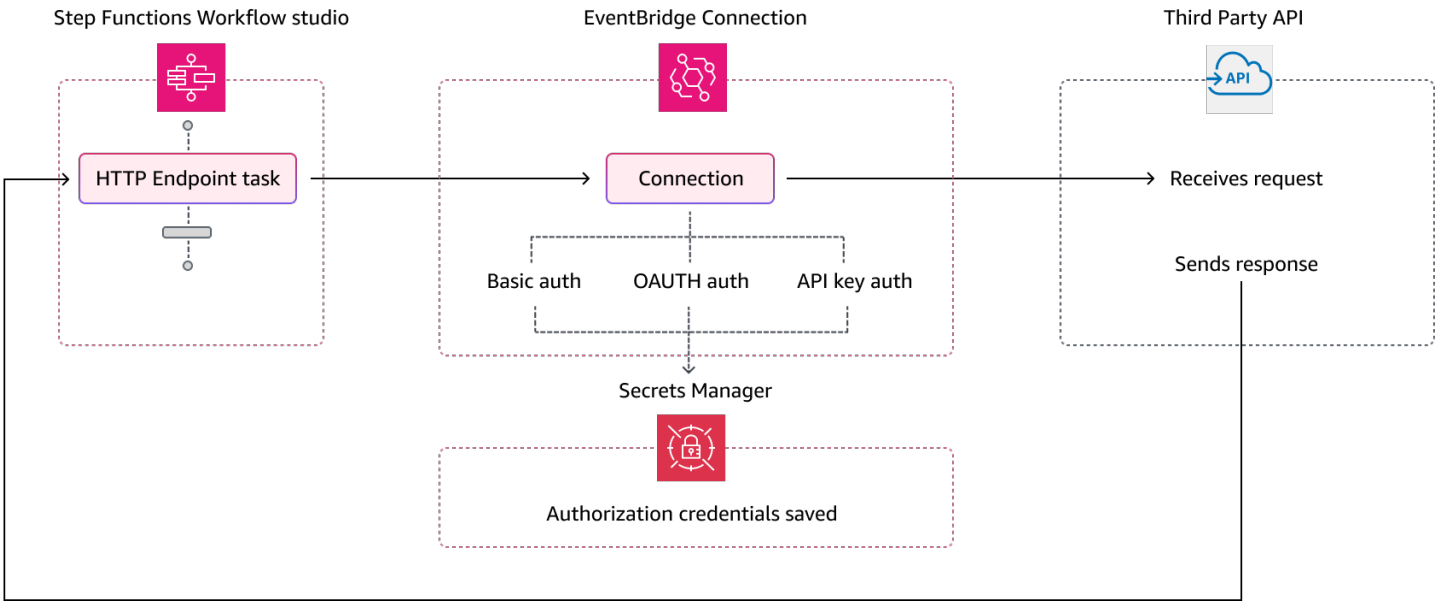
Autenticação para uma tarefa HTTP

Uma tarefa HTTP requer uma EventBridge conexão, que gerencia com segurança as credenciais de autenticação de um provedor de API. Uma conexão especifica o tipo de autorização e as credenciais a serem utilizadas para autorizar uma API de terceiros. Usar uma conexão ajuda a evitar segredos de codificação rígida, como chaves de API, na definição da máquina de estado. Uma EventBridge conexão é compatível com os esquemas de autorização Basic, OAuth e API Key.

Ao criar uma EventBridge conexão, você fornece seus detalhes de autenticação. Também é possível incluir o cabeçalho, o corpo e os parâmetros de consulta necessários para autorização com uma API. É necessário incluir o ARN da conexão em qualquer tarefa HTTP que chame uma API de terceiros.

Quando você cria uma conexão e adiciona parâmetros de autorização, EventBridge cria uma entrada secreta AWS Secrets Manager. Nesse segredo, EventBridge armazena os parâmetros de conexão e autorização em um formato criptografado. Para criar ou atualizar uma conexão com sucesso, você deve usar uma Conta da AWS que tenha permissão para usar o Secrets Manager. Para obter mais informações sobre as IAM permissões que sua máquina de estado precisa para acessar uma EventBridge conexão, consultePermissões do IAM para executar uma tarefa HTTP.

A imagem a seguir mostra como o Step Functions processa a autenticação para chamadas de API de terceiros usando uma conexão com o EventBridge.



Mesclar a conexão com o EventBridge e os dados da definição de tarefa HTTP

Ao invocar uma tarefa HTTP, é possível especificar dados na conexão com o EventBridge e na definição da tarefa HTTP. Esses dados incluem os parâmetros `Headers`, `QueryParameters` e `RequestBody`. Antes de chamar uma API de terceiros, o Step Functions mescla o corpo da solicitação com os parâmetros do corpo da conexão em todos os casos, exceto se o corpo da solicitação for uma string e os parâmetros do corpo da conexão não estiverem vazios. Nesse caso, a tarefa HTTP falhará com o erro `States.Runtime`.

Se houver alguma chave duplicada especificada na definição da tarefa HTTP e na conexão com o EventBridge, o Step Functions substituirá os valores na tarefa HTTP pelos valores na conexão.

A lista a seguir descreve como o Step Functions mescla dados antes de chamar uma API de terceiros:

- Cabeçalhos: o Step Functions adiciona todos os cabeçalhos especificados na conexão com os cabeçalhos no campo `Headers` da tarefa HTTP. Se houver um conflito entre as chaves do cabeçalho, o Step Functions usará os valores especificados na conexão para esses cabeçalhos. Por exemplo, se você especificou o cabeçalho `content-type` na definição da tarefa HTTP e na conexão com o EventBridge, o Step Functions usará o valor do cabeçalho `content-type` especificado na conexão.
- Parâmetros da consulta: o Step Functions adiciona todos os parâmetros da consulta especificados na conexão aos parâmetros da consulta no campo `QueryParameters` da tarefa HTTP. Se

houver um conflito entre as chaves de parâmetros da consulta, o Step Functions usará os valores especificados na conexão para esses cabeçalhos. Por exemplo, se você especificou o cabeçalho da consulta `maxItems` na definição da tarefa HTTP e na conexão com o EventBridge, o Step Functions usará o valor do parâmetro da consulta `maxItems` especificado na conexão.

- Body parameters (Parâmetros do corpo)
  - O Step Functions adiciona todos os valores do corpo da solicitação especificados na conexão com o corpo da solicitação no campo `RequestBody` da tarefa HTTP. Se houver um conflito entre as chaves do corpo da solicitação, o Step Functions usará os valores especificados na conexão para o corpo da solicitação. Por exemplo, digamos que você tenha especificado um campo `Mode` no `RequestBody` da definição da tarefa HTTP e da conexão com o EventBridge. O Step Functions usará o valor do campo `Mode` especificado na conexão.
  - Se você especificar o corpo da solicitação como uma string em vez de um objeto JSON e a conexão com o EventBridge também contiver o corpo da solicitação, o Step Functions não poderá mesclar o corpo da solicitação especificado nesses dois locais. A tarefa HTTP falha com o erro `States.Runtime`.

  O Step Functions aplica todas as transformações e serializa o corpo da solicitação depois de concluir a mesclagem do corpo da solicitação.

O exemplo a seguir define os campos `Headers`, `QueryParameters` e `RequestBody` na tarefa HTTP e na conexão com o EventBridge.

Definição da tarefa 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"
        }
      }
    }
  }
}
```

Conexão com o 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"
        }
```

```
      ]
    }
  }
}
```

Neste exemplo, chaves duplicadas são especificadas na tarefa HTTP e na conexão com o EventBridge. Portanto, o Step Functions substitui os valores na tarefa HTTP pelos valores na conexão. O trecho de código a seguir mostra a solicitação HTTP que o Step Functions envia à API de terceiros.

```
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"}
```

Aplicar a codificação em URL no corpo da solicitação

Por padrão, o Step Functions envia o corpo da solicitação como dados JSON a um endpoint da API. Se o provedor de API de terceiros exigir corpos da solicitação `form-urlencoded`, será necessário especificar a codificação em URL para os corpos da solicitação. Depois, o Step Functions vai codificar automaticamente o corpo da solicitação com base na opção de codificação em URL selecionada.

Você vai especificar a codificação em URL usando o campo `Transform`. Esse campo contém o campo `RequestBodyEncoding` que especifica se a codificação em URL será aplicada ou não aos corpos da solicitação. Ao especificar o campo `RequestBodyEncoding`, o Step Functions converte o corpo da solicitação JSON em corpo da solicitação `form-urlencoded` antes de chamar a API de terceiros. Também é necessário especificar o cabeçalho `content-type` como `application/x-www-form-urlencoded`, pois as APIs que aceitam dados codificados em URL esperam o cabeçalho `content-type`.

Para codificar matrizes no corpo da solicitação, o Step Functions oferece as opções de codificação de matriz a seguir.

- `INDICES`: repete uma chave para cada item em uma matriz e acrescenta um colchete, [], à chave para indicar que é uma matriz. Esse colchete contém o índice do elemento da matriz. Adicionar o índice ajuda a especificar a ordem dos elementos da matriz. Por padrão, o Step Functions usa essa opção de codificação.

  Por exemplo, se o corpo da solicitação contiver a matriz a seguir.

  ```
  {"array": ["a","b","c","d"]}
  ```

  O Step Functions codifica essa matriz para a string a seguir.

  ```
  array[0]=a&array[1]=b&array[2]=c&array[3]=d
  ```

- `REPEAT`: repete uma chave para cada item em uma matriz.

  Por exemplo, se o corpo da solicitação contiver a matriz a seguir.

  ```
  {"array": ["a","b","c","d"]}
  ```

  O Step Functions codifica essa matriz para a string a seguir.

  ```
  array=a&array=b&array=c&array=d
  ```

- `COMMAS`: codifica todos os valores em uma chave como uma lista de valores delimitada por vírgula.

  Por exemplo, se o corpo da solicitação contiver a matriz a seguir.

  ```
  {"array": ["a","b","c","d"]}
  ```

  O Step Functions codifica essa matriz para a string a seguir.

  ```
  array=a,b,c,d
  ```

- `BRACKETS`: repete uma chave para cada item em uma matriz e acrescenta um colchete, [], à chave para indicar que é uma matriz.

  Por exemplo, se o corpo da solicitação contiver a matriz a seguir.

  ```
  {"array": ["a","b","c","d"]}
  ```

O Step Functions codifica essa matriz para a string a seguir.

```
array[]=a&array[]=b&array[]=c&array[]=d
```

Permissões do IAM para executar uma tarefa HTTP

O perfil de execução da máquina de estado deve ter as permissões `states:InvokeHTTPEndpoint`, `events:RetrieveConnectionCredentials`, `secretsmanager:GetSecretValue` e `secretsmanager:DescribeSecret` para que uma tarefa HTTP chame uma API de terceiros. O exemplo de política do IAM a seguir concede os privilégios mínimos necessários ao perfil da máquina de estado para chamar APIs do Stripe. Essa política do IAM também concede permissão à função de máquina de estado para acessar uma EventBridge conexão específica, incluindo o segredo dessa conexão que está armazenado no 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/*"
        }
    ]
}
```

Exemplo de tarefa HTTP

A definição de máquina de estado a seguir mostra uma tarefa HTTP que inclui os parâmetros `Headers`, `QueryParameters`, `Transform` e `RequestBody`. A tarefa HTTP chama uma API do Stripe, https://api.stripe.com/v1/invoices, para gerar uma fatura. A tarefa HTTP também especifica a codificação em URL para o corpo da solicitação usando a opção de codificação `INDICES`.

Verifique se você criou uma conexão com o EventBridge. O exemplo a seguir mostra uma conexão criada usando o tipo de autenticação básica.

```
{
    "Type": "BASIC",
    "AuthParameters": {
        "BasicAuthParameters": {
            "Password": "myPassword",
            "Username": "myUsername"
         },
    }
}
```

Lembre-se de substituir o texto em *`itálico`* pelas informações específicas do 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 executar essa máquina de estado, forneça o ID do cliente como entrada, conforme mostrado no seguinte exemplo:

```
{
    "customer_id": "1234567890"
}
```

O exemplo a seguir mostra a solicitação HTTP que o Step Functions envia à API do 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
```

Testar uma tarefa HTTP

Você pode usar a TestStateAPI por meio do console, do SDK ou do AWS CLI para testar uma tarefa HTTP. O procedimento a seguir descreve como usar a TestState API no Step Functions console. É possível testar iterativamente os detalhes da solicitação, da resposta e da autenticação da API até que a tarefa HTTP esteja funcionando conforme o esperado.

Testar o estado de uma tarefa HTTP no console do Step Functions

1. Abra o console do Step Functions.
2. Selecione Criar uma máquina de estado para começar a criar uma máquina de estado ou escolha uma máquina de estado que contenha uma tarefa HTTP.

   Consulte a Etapa 4 se você estiver testando a tarefa em uma máquina de estado existente.
3. No Modo de design do Workflow Studio, configure visualmente uma tarefa HTTP. Também é possível selecionar o modo Código para copiar e colar a definição da máquina de estado do ambiente de desenvolvimento local.
4. No Modo de design, selecione Testar estado no painel Inspector do Workflow Studio.
5. Na caixa de diálogo Testar estado, faça o seguinte:

   a. Em Perfil de execução, selecione um perfil de execução para testar o estado. Se você não tiver um perfil com permissões suficientes para uma tarefa HTTP, consulte Perfil para testar tarefas HTTP no Workflow Studio para criar um perfil.

   b. (Opcional) Forneça todas as entradas JSON de que o estado selecionado precise para o teste.

   c. Em nível de inspeção, mantenha a seleção padrão de INFO. Esse nível mostra o status da chamada de API e a saída do estado. Isso é útil para conferir rapidamente a resposta da API.

   d. Selecione Iniciar teste.

   e. Se o teste for bem-sucedido, a saída do estado aparecerá no lado direito da caixa de diálogo Testar estado. Se o teste falhar, um erro será exibido.

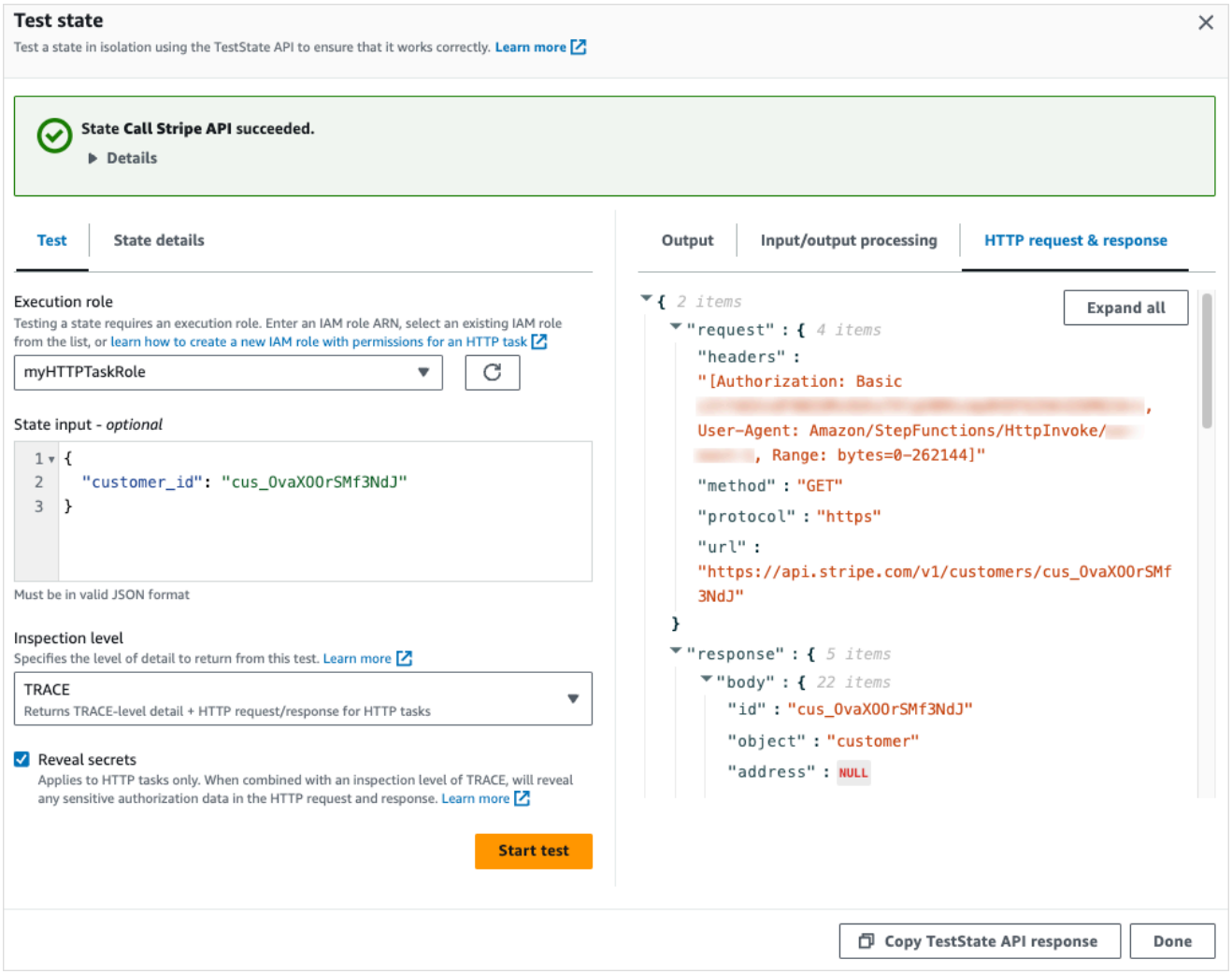
      Na guia Detalhes do estado da caixa de diálogo, é possível ver a definição do estado e um link para a conexão com o EventBridge.

   f. Altere o nível de inspeção para TRACE. Esse nível mostra a solicitação e a resposta HTTP brutas e é útil para verificar cabeçalhos, parâmetros de consulta e outros detalhes específicos da API.

   g. Marque a caixa de seleção Revelar segredos. Em combinação com TRACE, essa configuração permite que você veja os dados confidenciais que a conexão com o EventBridge insere, como chaves de API. A identidade do usuário do IAM que você usa para acessar o console deve ter permissão para realizar a ação `states:RevealSecrets`. Sem essa permissão, o Step Functions gera um erro de acesso negado ao iniciar

o teste. Para ver um exemplo de política do IAM que concede essas permissões `states:RevealSecrets`, consulte IAMpermissões para usar a TestState API.

A imagem a seguir mostra um teste bem-sucedido para uma tarefa HTTP. O Nível de inspeção para esse estado é definido como TRACE. A guia Solicitação e resposta HTTP na imagem a seguir mostra o resultado da chamada de API de terceiros.



h. Selecione Iniciar teste.

i. Se o teste for bem-sucedido, será possível ver os detalhes HTTP na guia Solicitação e resposta HTTP.

Respostas da tarefa HTTP não compatíveis

Uma tarefa HTTP falhará com o erro `States.Runtime` se uma das seguintes condições for verdadeira para a resposta exibida:

- A resposta contém um cabeçalho do tipo de conteúdo de `application/octet-stream`, `image/*`, `video/*` ou `audio/*`.
- A resposta não pode ser lida como uma string válida. Por exemplo, dados binários ou de imagem.

Padrões de integração de serviço

AWS Step Functions se integra aos serviços diretamente no Amazon States Language. É possível controlar esses serviços da AWS usando três padrões de integração de serviço:

- Chame um serviço e deixe que o Step Functions avance para o próximo estado imediatamente após obter uma resposta HTTP.
- Chame um serviço e faça o Step Functions aguardar a conclusão de uma tarefa.
- Chame um serviço com um token da tarefa e faça com que o Step Functions aguarde até que esse token retorne com uma carga.

Cada um desses padrões de integração de serviços é controlado pelo modo como você cria um URI no campo `"Resource"` da sua definição de tarefa.

Maneiras para chamar um serviço integrado

- Resposta de solicitação
- Executar um trabalho (.sync)
- Aguardar um retorno de chamada com um token de tarefa

Para obter informações sobre a configuração AWS Identity and Access Management (IAM) para serviços integrados, consulte Políticas do IAM para serviços integrados.

Resposta de solicitação

Quando você especifica um serviço na string `"Resource"` do seu estado de tarefa, e você fornece apenas o recurso, o Step Functions aguarda uma resposta HTTP e, então, avança para o próximo estado. O Step Functions não esperará a conclusão de um trabalho.

O exemplo a seguir mostra como publicar um tópico do 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 exemplo se refere à API Publish do Amazon SNS. O fluxo de trabalho seguirá para o próximo estado após a chamada da API `Publish`.

> **Tip**
>
> Para implantar um exemplo de fluxo de trabalho que usa o padrão de integração do serviço de Solicitação de Resposta no seu Conta da AWS, consulte o Módulo 2 - Resposta de Solicitação do AWS Step Functions Workshop.

Executar um trabalho (.sync)

Para serviços integrados, como AWS Batch o Amazon ECS, o Step Functions pode aguardar a conclusão de uma solicitação antes de passar para o próximo estado. Para que o Step Functions aguarde, especifique o campo `"Resource"` na definição do seu estado de tarefa com o sufixo `.sync` anexado depois do URI do recurso.

Por exemplo, ao enviar um AWS Batch trabalho, use o `"Resource"` campo na definição da máquina de estado, conforme mostrado neste exemplo.

```
"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"
}
```

A parte `.sync` anexada ao nome do recurso da Amazon (ARN) do recurso indica que o Step Functions aguarda até que o trabalho seja concluído. Depois de chamar `submitJob` do AWS Batch ,

o fluxo de trabalho é pausado. Quando o trabalho é concluído, o Step Functions avança para o próximo estado. Para obter mais informações, consulte o projeto AWS Batch de amostra:Gerenciar uma tarefa em lote (AWS Batch, Amazon SNS).

Se uma tarefa usando esse padrão de integração de serviço (`.sync`) for cancelada e o Step Functions não conseguir cancelar a tarefa, você poderá incorrer em cobranças adicionais do serviço integrado. Uma tarefa pode ser cancelada se:

- A execução da máquina de estado for interrompida.
- Uma ramificação diferente de um estado paralelo falha com um erro não detectado.
- Uma iteração de um estado Mapa falha com um erro não detectado.

O Step Functions fará o possível para cancelar a tarefa. Por exemplo, se uma tarefa `states:startExecution.sync` do Step Functions for cancelada, ela chamará a ação da API `StopExecution` do Step Functions. No entanto, é possível que o Step Functions não consiga cancelar a tarefa. Os motivos para isso incluem, entre outros:

- Sua função de execução do IAM não tem permissão para fazer a chamada de API correspondente.
- Ocorreu uma interrupção temporária do serviço.

Quando você usa o padrão de integração do serviço `.sync`, o Step Functions usa uma pesquisa que consome a cota e os eventos atribuídos para monitorar o status de um trabalho. Para `.sync` invocações na mesma conta, o Step Functions usa EventBridge eventos e pesquisa as APIs que você especifica no estado. `Task` Para invocações `.sync` entre contas, o Step Functions usa apenas pesquisa. Por exemplo, for`states:StartExecution.sync`, Step Functions realiza pesquisas na DescribeExecutionAPI e usa sua cota atribuída.

> **Tip**
>
> Para implantar um exemplo de fluxo de trabalho que usa o padrão de integração do serviço Run a Job (.sync) no seu Conta da AWS, consulte o Módulo 3 - Executar um trabalho (.sync) do The AWS Step Functions Workshop.

Para ver uma lista de quais serviços integrados oferecem suporte à espera da conclusão de um trabalho (`.sync`), consulte Integrações otimizadas para o Step Functions.

> **Note**
>
> As integrações de serviços que usam o padrão `.sync` exigem permissões adicionais do IAM. Para ter mais informações, consulte Políticas do IAM para serviços integrados.

Em alguns casos, você pode querer que o Step Functions continue com o fluxo de trabalho antes que o trabalho seja totalmente concluído. Você pode fazer isso da mesma forma que ao usar o padrão de integração de serviços Aguardar um retorno de chamada com um token de tarefa. Para fazer isso, passe um token de tarefa para seu trabalho e, em seguida, retorne-o usando uma chamada de API `SendTaskSuccess` ou `SendTaskFailure`. O Step Functions usará os dados fornecidos nessa chamada para concluir a tarefa, parar de monitorar o trabalho e continuar o fluxo de trabalho.

Aguardar um retorno de chamada com um token de tarefa

As tarefas de retorno de chamada oferecem uma maneira de pausar um fluxo de trabalho até o retorno do token da tarefa. Uma tarefa pode precisar aguardar uma aprovação humana, integrar-se a terceiros ou chamar sistemas legados. Para tarefas como essas, você pode pausar o Step Functions até que a execução do fluxo de trabalho atinja a cota de serviço de um ano (consulte Cotas relacionadas aos controles de utilização de estado) e aguardar a conclusão de um processo ou fluxo de trabalho externo. Para essas situações, o Step Functions permite que você passe um token de tarefa para as integrações de serviços do AWS SDK e também para algumas integrações de serviços otimizados. A tarefa será pausada até o retorno de um token com uma chamada `SendTaskSuccess` ou `SendTaskFailure`.

Se um estado `Task` usando o token da tarefa de retorno de chamada expirar, um novo token aleatório será gerado. Você pode acessar os tokens da tarefa a partir do objeto de contexto.

> **Note**
>
> Um token de tarefa deve conter pelo menos um caractere e não pode exceder 1024 caracteres.

Para usar `.waitForTaskToken` com uma integração do AWS SDK, a API que você usa deve ter um campo de parâmetro no qual colocar o token da tarefa.

> **Note**
>
> Você deve passar tokens de tarefas dos diretores da mesma AWS conta. Os tokens não funcionarão se você os enviar de diretores em uma AWS conta diferente.

> **Tip**
>
> Para implantar um exemplo de fluxo de trabalho que usa um padrão de integração do serviço de token de tarefa de retorno de chamada no seu Conta da AWS, consulte o Módulo 4 - Aguarde um retorno de chamada com o token de tarefa do AWS Step Functions workshop.

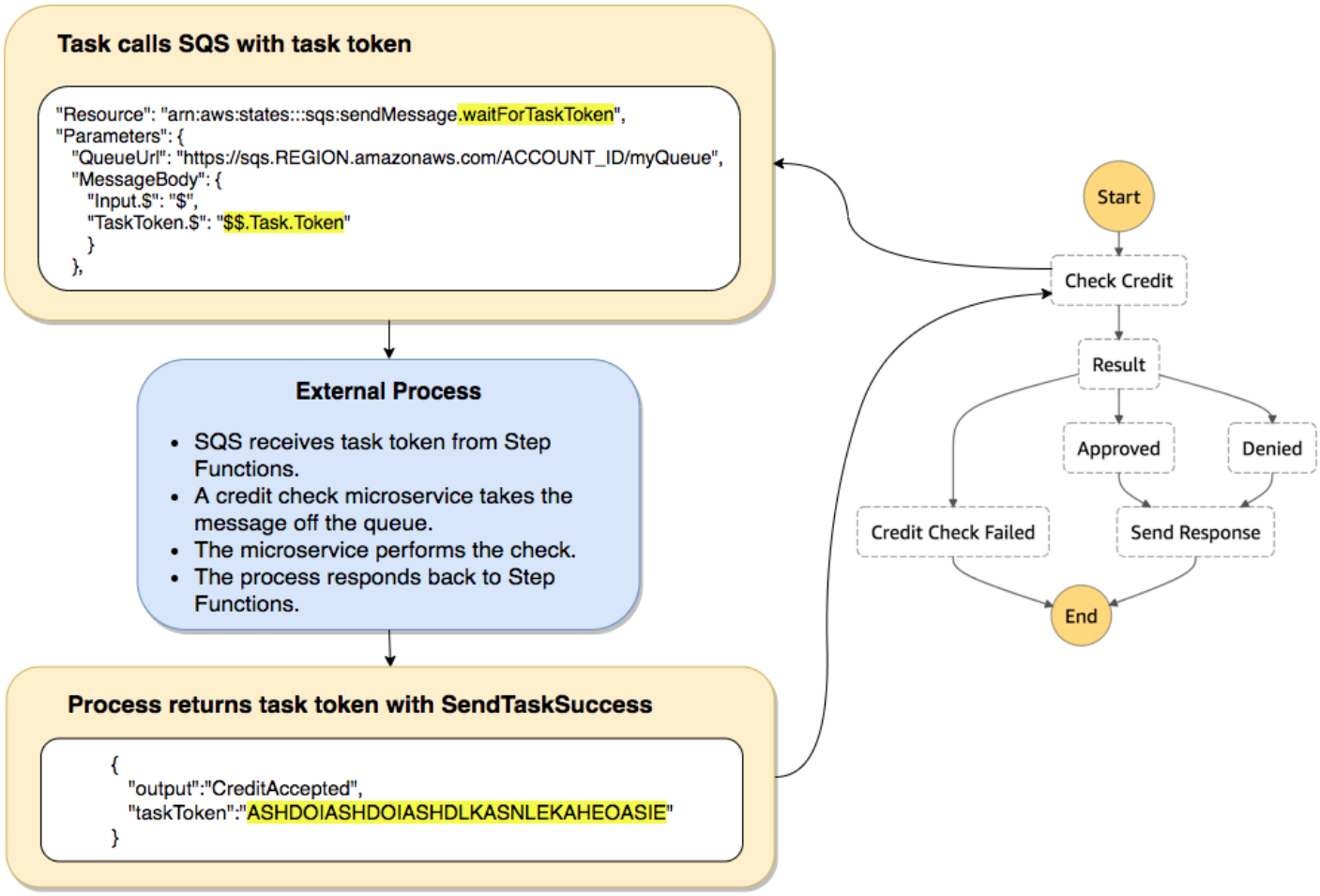
Para ver uma lista de quais serviços integrados oferecem suporte à espera de um token de tarefa (`.waitForTaskToken`), consulte Integrações otimizadas para o Step Functions.

Tópicos

- Exemplo de token de tarefa
- Obter um token para o objeto de contexto
- Configurar um tempo limite de pulsação para uma tarefa de espera

Exemplo de token de tarefa

Neste exemplo, um fluxo de trabalho do Step Functions precisa se integrar a um microsserviço externo para realizar uma verificação de crédito como parte de um fluxo de trabalho de aprovação. O Step Functions publica uma mensagem do Amazon SQS que inclui um token de tarefa como parte da mensagem. Um sistema externo se integra ao Amazon SQS e obtém a mensagem da fila. Quando isso é concluído, ele retorna o resultado e o token da tarefa original. O Step Functions então continua com seu fluxo de trabalho.



O campo `"Resource"` da definição da tarefa que se refere ao Amazon SQS inclui o `.waitForTaskToken` anexado no 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"
}
```

Isso informa ao Step Functions para pausar e aguardar o token da tarefa. Quando você especifica um recurso usando `.waitForTaskToken`, o token da tarefa pode ser acessado no campo `"Parameters"` da definição de estado com uma designação de caminho especial (`$$.Task.Token`). O primeiro `$$.` define que o caminho acesse o objeto de contexto e obtenha o token da tarefa para a tarefa atual em uma execução que está em andamento.

Após a conclusão, o serviço externo chama `SendTaskSuccess` ou `SendTaskFailure` com `taskToken` incluído. Somente depois disso, o fluxo de trabalho segue para o próximo estado.

> **Note**
>
> Para evitar aguardar indefinidamente caso um processo não envie o token da tarefa com `SendTaskSuccess` ou `SendTaskFailure`, consulte Configurar um tempo limite de pulsação para uma tarefa de espera.

Obter um token para o objeto de contexto

O objeto de contexto é um objeto JSON interno com informações sobre a sua execução. Assim como a entrada de estado, ele pode ser acessado com um caminho do campo `"Parameters"` durante uma execução. Quando acessado dentro de uma definição de tarefa, ele inclui informações sobre a execução específica, incluindo o token da tarefa.

```
{
    "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"
    }
}
```

É possível acessar o token da tarefa usando um caminho especial dentro do campo `"Parameters"` da sua definição de tarefa. Para acessar o objeto de contexto ou entrada, primeiro especifique que o parâmetro será um caminho anexando um `.$` ao nome do parâmetro. O seguinte especifica os nós de entrada e do objeto de contexto em uma especificação `"Parameters"`.

```
"Parameters": {
   "Input.$": "$",
   "TaskToken.$": "$$.Task.Token"
 },
```

Em ambos os casos, anexar `.$` ao nome de parâmetro informa ao Step Functions para aguardar um caminho. No primeiro caso, `"$"` é um caminho que inclui a entrada completa. No segundo caso, `$$.` especifica que o caminho acessará o objeto de contexto, e `$$.Task.Token` define o parâmetro como o valor do token da tarefa no objeto de contexto de uma execução que está em andamento.

No exemplo do Amazon SQS, `.waitForTaskToken` no campo `"Resource"` informa ao Step Functions para aguardar o retorno do token da tarefa. O parâmetro `"TaskToken.$": "$$.Task.Token"` transmite esse token como parte da mensagem do 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 obter mais informações sobre o objeto de contexto, consulte Objeto de contexto na seção Processamento de entrada e saída deste guia.

Configurar um tempo limite de pulsação para uma tarefa de espera

Uma tarefa que aguarda um token de tarefa aguardará até que a execução atinja a cota de serviço de um ano (consulte Cotas relacionadas aos controles de utilização de estado). Para evitar execuções presas, configure um intervalo de tempo limite de pulsação na sua definição de máquina de estado. Use o campo `HeartbeatSeconds` para especificar o intervalo do tempo limite.

```
{
  "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
    }
  }
}
```

Nesta definição de máquina de estado, uma tarefa envia uma mensagem ao Amazon SQS e aguarda que um processo externo retorne a chamada com o token de tarefa fornecido. O campo `"HeartbeatSeconds": 600` define o intervalo de tempo limite de pulsação como 10 minutos. A tarefa aguardará o retorno do token de tarefa com uma dessas ações de API:

- `SendTaskSuccess`
- `SendTaskFailure`
- `SendTaskHeartbeat`

Se a tarefa de espera não receber um token válido em até 10 minutos, ela falhará com um nome de erro `States.Timeout`.

Para obter mais informações, consulte o exemplo de projeto de tarefa de retorno de chamada: Exemplo de Padrão de Retorno de Chamada (Amazon SQS, Amazon SNS, Lambda) .

Transmitir parâmetros para uma API de serviço

Use o campo `Parameters` em um estado `Task` para controlar os parâmetros que serão transmitidos a uma API de serviço.

Dentro do campo `Parameters`, você deve usar a forma plural dos parâmetros da matriz em uma ação de API. Por exemplo, se você usar o campo Filtro da ação de API `DescribeSnapshots` para integração com o Amazon EC2, será necessário definir o campo como `Filters`. Se você não usar o formato plural, o Step Functions retornará o seguinte erro:

```
The field Filter is not supported by Step Functions.
```

Transmitir JSON estático como parâmetros

É possível incluir um objeto JSON diretamente na definição da máquina de estado para transmiti-lo como um parâmetro a um recurso.

Por exemplo, para definir o parâmetro `RetryStrategy` para a API `SubmitJob` do AWS Batch, você pode incluir nos parâmetros o seguinte:

```
"RetryStrategy": {
  "attempts": 5
}
```

Também é possível transmitir vários parâmetros com JSON estático. Para um exemplo mais completo, veja a seguir os campos `Resource` e `Parameters` da especificação de uma tarefa que publica em um tópico do Amazon SNS chamado *`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"
```

```
        }
      }
  },
```

Transmitir a entrada de estado como parâmetros usando caminhos

É possível transmitir partes da entrada de estado como parâmetros usando caminhos. Um caminho é uma string que começa com $ e é usada para identificar componentes em texto JSON. Os caminhos do Step Functions usam a sintaxe JsonPath.

Para especificar que um parâmetro use um caminho, termine o nome do parâmetro com `.$`. Por exemplo, se sua entrada de estado contiver texto em um nó chamado `message`, você poderá transmitir esse texto como parâmetro usando um caminho.

Considere a seguinte entrada de estado:

```
{
  "comment": "A message in the state input",
  "input": {
    "message": "foo",
    "otherInfo": "bar"
  },
  "data": "example"
}
```

Para transmitir o valor do nó denominado `message` como parâmetro, especifique a seguinte sintaxe:

```
"Parameters": {"myMessage.$": "$.input.message"},
```

Em seguida, o Step Functions transmite o valor `foo` como um parâmetro.

Para ver mais informações sobre como usar parâmetros no Step Functions, consulte o seguinte:

- Processamento de entrada e saída
- InputPath, Parâmetros e ResultSelector

Passar nós do objeto de contexto como parâmetros

Além do conteúdo estático e de nós do estado de entrada, é possível passar nós do objeto de contexto como parâmetros. O objeto de contexto é formado por dados JSON dinâmicos que existem

durante uma execução da máquina de estado. Ele inclui informações sobre sua máquina de estado e a execução atual. É possível acessar o objeto de contexto usando um caminho no campo `"Parameters"` de uma definição de estado.

Para obter mais informações sobre o objeto de contexto e como acessar os dados de um campo `"Parameters"`, consulte o seguinte:

- Objeto de contexto
- Acessar o objeto de contexto
- Obter um token para o objeto de contexto

Registro de alterações para integrações de AWS SDK compatíveis

A tabela a seguir resume quando os serviços foram inicialmente integrados ao Step Functions e quando sua API de integração foi atualizada mais recentemente. Para obter detalhes sobre o uso de integrações, consulteAWS Integrações de serviços SDK.

> **Important**
>
> O suporte à ação da API é lançado trimestralmente. As atualizações de ações já suportadas, como novos parâmetros, podem não estar imediatamente disponíveis.

Serviço	Suporte inicial	Atualizado	
AWS AppFabric	18 de janeiro de 2024		
B2B Data Interchange	18 de janeiro de 2024		
Exportações de dados da AWS	18 de janeiro de 2024		
Amazon Bedrock	18 de janeiro de 2024		
Amazon Bedrock Agents	18 de janeiro de 2024		

Serviço	Suporte inicial	Atualizado	
Amazon Bedrock Runtime Agents	18 de janeiro de 2024		
Amazon Bedrock Runtime	18 de janeiro de 2024		
Amazon CloudFront KeyValueStore	18 de janeiro de 2024		
Amazon CodeGuru Security	18 de janeiro de 2024		
Hub de Otimização de Custos da AWS	18 de janeiro de 2024		
Amazon DataZone	18 de janeiro de 2024		
Amazon EKS Auth	18 de janeiro de 2024		
AWS Entity Resolution	18 de janeiro de 2024		
Nível gratuito da AWS	18 de janeiro de 2024		
Amazon Inspector Scan	18 de janeiro de 2024		
AWS Launch Wizard	18 de janeiro de 2024		
Amazon Managed Blockchain Query	18 de janeiro de 2024		
AWS Elemental MediaPackage V2	18 de janeiro de 2024		
AWS HealthImaging	18 de janeiro de 2024		
Network Manager	18 de janeiro de 2024		

Serviço	Suporte inicial	Atualizado	
AWS Payment Cryptography	18 de janeiro de 2024		
AWS Payment Cryptography Data	18 de janeiro de 2024		
AWS Private CA Connector for Active Directory	18 de janeiro de 2024		
Amazon Q Business	18 de janeiro de 2024		
Amazon Q Connect	18 de janeiro de 2024		
AWS re:Post	18 de janeiro de 2024		
Amazon Timestream Query	18 de janeiro de 2024		
Amazon Timestream Write	18 de janeiro de 2024		
Trusted Advisor	18 de janeiro de 2024		
Verified Permissions	18 de janeiro de 2024		
Amazon WorkSpaces Thin Client	18 de janeiro de 2024		
AWS CloudTrail Data	16 de junho de 2023		
Amazon CloudWatch Internet Monitor	16 de junho de 2023		
Amazon Interacti ve Video Service RealTime	16 de junho de 2023		

Serviço	Suporte inicial	Atualizado	
AWS IoT TwinMaker	16 de junho de 2023		
Amazon OpenSearch Ingestion	16 de junho de 2023		
AWS Telco Network Builder	16 de junho de 2023		
Amazon VPC Lattice	16 de junho de 2023		
AWS Backup Storage	17 de fevereiro de 2023		
Amazon Chime Media Pipelines	17 de fevereiro de 2023		
Amazon Chime Voice	17 de fevereiro de 2023		
AWS Clean Rooms	17 de fevereiro de 2023	18 de janeiro de 2024	
Amazon CodeCatalyst	17 de fevereiro de 2023		
Amazon Connect Cases	17 de fevereiro de 2023		
AWS Control Tower	17 de fevereiro de 2023		
Amazon DocumentDB Elastic Clusters	17 de fevereiro de 2023		
Amazon EMR Serverless	17 de fevereiro de 2023		

Serviço	Suporte inicial	Atualizado	
Amazon IVS Chat	17 de fevereiro de 2023		
Amazon Kendra Intelligent Ranking	17 de fevereiro de 2023		
AWS HealthOmics	17 de fevereiro de 2023		
Amazon Redshift Serverless	17 de fevereiro de 2023		
Amazon Security Lake	17 de fevereiro de 2023		
AWS Health	17 de fevereiro de 2023		
AWS IoT FleetWise	17 de fevereiro de 2023		
AWS IoT RoboRunner	17 de fevereiro de 2023		
AWS Mainframe Modernization	17 de fevereiro de 2023		
Orquestrador do AWS Migration Hub	17 de fevereiro de 2023		
AWS Private 5G	17 de fevereiro de 2023		
Explorador de recursos da AWS	17 de fevereiro de 2023		
AWS SimSpace Weaver	17 de fevereiro de 2023		

Serviço	Suporte inicial	Atualizado	
AWS Support App	17 de fevereiro de 2023		
CloudWatch Observability Access Manager	17 de fevereiro de 2023		
EventBridge Pipes	17 de fevereiro de 2023		
EventBridge Scheduler	17 de fevereiro de 2023		
IAM Roles Anywhere	17 de fevereiro de 2023		
Kinesis Video WebRTC Storage	17 de fevereiro de 2023		
License Manager Linux Subscriptions	17 de fevereiro de 2023		
License Manager User Subscriptions	17 de fevereiro de 2023		
OpenSearch Serverless	17 de fevereiro de 2023		
Route 53 ARC Zonal Shift	17 de fevereiro de 2023		
SageMaker Geospatial	17 de fevereiro de 2023		
SageMaker Metrics	17 de fevereiro de 2023		

Serviço	Suporte inicial	Atualizado	
Systems Manager for SAP	17 de fevereiro de 2023		
AWS Account Management	19 de abril de 2022		
AWS Amplify	30 de setembro de 2021		
AWS App Mesh	30 de setembro de 2021		
AWS App Runner	30 de setembro de 2021	17 de fevereiro de 2023	
AWS AppConfig	30 de setembro de 2021		
AWS AppConfig Data	19 de abril de 2022		
AWS AppSync	30 de setembro de 2021	17 de fevereiro de 2023	
AWS Application Discovery Service	30 de setembro de 2021		
AWS Application Migration Service	30 de setembro de 2021		
AWS Audit Manager	30 de setembro de 2021		
AWS Auto Scaling Plans	30 de setembro de 2021		
AWS Backup	30 de setembro de 2021	17 de fevereiro de 2023	

Serviço	Suporte inicial	Atualizado	
AWS Backup gateway	19 de abril de 2022	17 de fevereiro de 2023	
AWS Batch	30 de setembro de 2021	17 de fevereiro de 2023	
AWS Billing Conductor	26 de julho de 2022	17 de fevereiro de 2023	
AWS Budgets	30 de setembro de 2021		
AWS Certificate Manager	30 de setembro de 2021		
AWS Private Certificate Authority	30 de setembro de 2021		
AWS Cloud Map	30 de setembro de 2021		
AWS Cloud9	30 de setembro de 2021		
AWS CloudFormation	30 de setembro de 2021	17 de fevereiro de 2023	
AWS CloudHSM	30 de setembro de 2021		
AWS CloudHSM	30 de setembro de 2021		
AWS CloudTrail	30 de setembro de 2021	17 de fevereiro de 2023	
AWS Cloud Control	19 de abril de 2022		

Serviço	Suporte inicial	Atualizado	
AWS CodeBuild	30 de setembro de 2021		
AWS CodeCommit	30 de setembro de 2021	17 de fevereiro de 2023	
AWS CodeDeploy	30 de setembro de 2021		
AWS CodePipeline	30 de setembro de 2021		
AWS CodeStar	30 de setembro de 2021		
AWS CodeStar	30 de setembro de 2021		
AWS CodeStar	30 de setembro de 2021		
AWS Compute Optimizer	30 de setembro de 2021	17 de fevereiro de 2023	
AWS Config	30 de setembro de 2021	26 de julho de 2022	
AWS Cost Explorer Service	30 de setembro de 2021	17 de fevereiro de 2023	
AWS Cost and Usage Report	30 de setembro de 2021		
AWS Data Exchange	30 de setembro de 2021	26 de julho de 2022	
AWS Data Pipeline	30 de setembro de 2021		

Serviço	Suporte inicial	Atualizado	
AWS DataSync	30 de setembro de 2021	26 de julho de 2022	
AWS Database Migration Service	30 de setembro de 2021		
AWS Device Farm	30 de setembro de 2021		
AWS Direct Connect	30 de setembro de 2021		
AWS Directory Service	30 de setembro de 2021	17 de fevereiro de 2023	
AWS EC2 Instance Connect	30 de setembro de 2021		
AWS Elastic Beanstalk	30 de setembro de 2021		
AWS Elemental MediaLive	30 de setembro de 2021		
AWS Elemental MediaPackage	30 de setembro de 2021		
AWS Elemental MediaPackage VOD	30 de setembro de 2021		
AWS Elemental MediaStore	30 de setembro de 2021		
AWS Fault Injection Service	30 de setembro de 2021		
AWS Firewall Manager	30 de setembro de 2021	17 de fevereiro de 2023	

Serviço	Suporte inicial	Atualizado	
AWS Glue	30 de setembro de 2021	17 de fevereiro de 2023	
AWS Glue DataBrew	30 de setembro de 2021		
AWS IoT Greengrass	30 de setembro de 2021		
AWS Ground Station	30 de setembro de 2021	17 de fevereiro de 2023	
AWS Identity and Access Management	30 de setembro de 2021		
AWS IoT	30 de setembro de 2021	17 de fevereiro de 2023	
AWS IoT 1-Click	30 de setembro de 2021		
AWS IoT Analytics	30 de setembro de 2021		
AWS IoT Core Device Advisor	30 de setembro de 2021		
AWS IoT Events	30 de setembro de 2021		
Dados do AWS IoT Events	30 de setembro de 2021		
AWS IoT Fleet Hub	30 de setembro de 2021		
AWS IoT Greengrass Version 2	30 de setembro de 2021		

Serviço	Suporte inicial	Atualizado	
AWS IoT Jobs Data Plane	30 de setembro de 2021		
AWS IoT Secure Tunneling	30 de setembro de 2021		
AWS IoT SiteWise	30 de setembro de 2021	17 de fevereiro de 2023	
AWS IoT Things Graph	30 de setembro de 2021		
AWS IoT Wireless	30 de setembro de 2021	17 de fevereiro de 2023	
AWS Key Management Service	30 de setembro de 2021	26 de julho de 2022	
AWS Lake Formation	30 de setembro de 2021	17 de fevereiro de 2023	
AWS Lambda	30 de setembro de 2021	17 de fevereiro de 2023	
AWS License Manager	30 de setembro de 2021	17 de fevereiro de 2023	
AWS Marketplace	30 de setembro de 2021	17 de fevereiro de 2023	
AWS Marketplace Commerce Analytics	30 de setembro de 2021		
AWS Marketplace Entitlement Service	30 de setembro de 2021		
AWS Elemental MediaTailor	30 de setembro de 2021	26 de julho de 2022	

Serviço	Suporte inicial	Atualizado	
AWS Migration Hub	30 de setembro de 2021		
AWS Migration Hub Config	30 de setembro de 2021		
Recomendações Estratégicas do AWS Migration Hub	19 de abril de 2022	17 de fevereiro de 2023	
AWS Mobile	30 de setembro de 2021		
AWS Network Firewall	30 de setembro de 2021		
AWS OpsWorks	30 de setembro de 2021		
AWS OpsWorks CM	30 de setembro de 2021		
AWS Organizations	30 de setembro de 2021	17 de fevereiro de 2023	
AWS Outposts	30 de setembro de 2021		
AWS Panorama	19 de abril de 2022	17 de fevereiro de 2023	
Amazon Relational Database Service Performance Insights	30 de setembro de 2021		
AWS Price List	30 de setembro de 2021		

Serviço	Suporte inicial	Atualizado	
Amazon Relational Database Service	30 de setembro de 2021		
AWS Resilience Hub	19 de abril de 2022		
AWS Resource Access Manager	30 de setembro de 2021		
AWS Resource Groups	30 de setembro de 2021	17 de fevereiro de 2023	
AWS Resource Groups Tagging API	30 de setembro de 2021		
AWS RoboMaker	30 de setembro de 2021		
AWS IAM Identity Center	30 de setembro de 2021	17 de fevereiro de 2023	
AWS SSO OIDC	30 de setembro de 2021		
AWS Secrets Manager	30 de setembro de 2021		
AWS Security Token Service	30 de setembro de 2021		
AWS Security Hub	30 de setembro de 2021		
AWS Server Migration Service	30 de setembro de 2021		
AWS Service Catalog	30 de setembro de 2021		

Serviço	Suporte inicial	Atualizado	
AWS Service Catalog AppRegistry	30 de setembro de 2021	17 de fevereiro de 2023	
AWS Shield	30 de setembro de 2021		
AWS Signer	30 de setembro de 2021		
AWS IAM Identity Center	30 de setembro de 2021		
AWS IAM Identity Center Admin	30 de setembro de 2021		
AWS Step Functions	30 de setembro de 2021	17 de fevereiro de 2023	
AWS Storage Gateway	30 de setembro de 2021		
AWS Support	30 de setembro de 2021		
AWS Transfer Family	30 de setembro de 2021	17 de fevereiro de 2023	
AWS WAF	30 de setembro de 2021		
AWS WAF Regional	30 de setembro de 2021		
AWS WAFV2	30 de setembro de 2021		
AWS Well-Architected Tool	30 de setembro de 2021	17 de fevereiro de 2023	

Serviço	Suporte inicial	Atualizado	
AWS X-Ray	30 de setembro de 2021	17 de fevereiro de 2023	
AWS Marketplace Metering Service	30 de setembro de 2021		
AWS Serverless Application Repository	30 de setembro de 2021		
AWS Identity and Access Management Access Analyzer	30 de setembro de 2021		
Alexa for Business	30 de setembro de 2021		
Amazon API Gateway	30 de setembro de 2021	17 de fevereiro de 2023	
Amazon API Gateway	30 de setembro de 2021		
Amazon AppIntegrations	30 de setembro de 2021		
Amazon AppStream 2.0	30 de setembro de 2021		
Amazon AppFlow	30 de setembro de 2021	17 de fevereiro de 2023	
Amazon Athena	30 de setembro de 2021	17 de fevereiro de 2023	
Amazon Augmented AI	30 de setembro de 2021		

Serviço	Suporte inicial	Atualizado	
Amazon Braket	30 de setembro de 2021		
Amazon Chime	30 de setembro de 2021		
Amazon Chime Meetings	19 de abril de 2022	17 de fevereiro de 2023	
Amazon Cloud Directory	30 de setembro de 2021		
Amazon CloudFront	30 de setembro de 2021	17 de fevereiro de 2023	
Amazon CloudSearch	30 de setembro de 2021		
Amazon CloudWatch	30 de setembro de 2021	17 de fevereiro de 2023	
Amazon CloudWatch Application Insights	30 de setembro de 2021		
CloudWatch Evidently	19 de abril de 2022		
Amazon CloudWatch Logs	30 de setembro de 2021		
Amazon CloudWatch RUM	19 de abril de 2022	17 de fevereiro de 2023	
Amazon CloudWatch Synthetics	30 de setembro de 2021		
Amazon CodeGuru Profiler	30 de setembro de 2021		

Serviço	Suporte inicial	Atualizado	
Amazon CodeGuru Reviewer	30 de setembro de 2021		
Amazon Cognito	30 de setembro de 2021		
Amazon Cognito Identity Provider	30 de setembro de 2021		
Amazon Cognito Sync	30 de setembro de 2021		
Amazon Comprehend	30 de setembro de 2021	17 de fevereiro de 2023	
Amazon Comprehend Medical	30 de setembro de 2021		
Amazon Connect Contact Lens	30 de setembro de 2021		
Amazon Connect Participant Service	30 de setembro de 2021		
Amazon Connect	30 de setembro de 2021	17 de fevereiro 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 setembro de 2021		
Amazon Detective	30 de setembro de 2021		

Serviço	Suporte inicial	Atualizado	
Amazon DevOps Guru	30 de setembro de 2021	26 de julho de 2022	
Amazon DocumentDB (with MongoDB compatibility)	30 de setembro de 2021		
Amazon DynamoDB	30 de setembro de 2021	17 de fevereiro de 2023	
Amazon DynamoDB Streams	30 de setembro de 2021		
Amazon EC2 Container Registry	30 de setembro de 2021		
Amazon EC2 Container Service	30 de setembro de 2021	17 de fevereiro de 2023	
Amazon EC2 Systems Manager	30 de setembro de 2021	17 de fevereiro de 2023	
Amazon EMR	30 de setembro de 2021	17 de fevereiro de 2023	
Amazon ElastiCache	30 de setembro de 2021		
Amazon Elastic Inference	30 de setembro de 2021		
Amazon Elastic Block Store	30 de setembro de 2021		
Amazon Elastic Compute Cloud	30 de setembro de 2021	17 de fevereiro de 2023	

Serviço	Suporte inicial	Atualizado	
Amazon Elastic Container Registry Public	30 de setembro de 2021		
Amazon Elastic File System	30 de setembro de 2021		
Amazon Elastic Kubernetes Service	30 de setembro de 2021	17 de fevereiro de 2023	
Amazon EMR	30 de setembro de 2021		
Amazon Elastic Transcoder	30 de setembro de 2021		
Amazon OpenSearch Service	30 de setembro de 2021	17 de fevereiro de 2023	
Amazon OpenSearch Service	19 de abril de 2022	17 de fevereiro de 2023	
Amazon EventBridge	30 de setembro de 2021	17 de fevereiro de 2023	
Amazon FSx	30 de setembro de 2021	17 de fevereiro de 2023	
Amazon Forecast Query	30 de setembro de 2021	17 de fevereiro de 2023	
Amazon Forecast Service	30 de setembro de 2021	17 de fevereiro de 2023	
Amazon Fraud Detector	30 de setembro de 2021		

Serviço	Suporte inicial	Atualizado	
Amazon GameLift	30 de setembro de 2021	17 de fevereiro de 2023	
Amazon GameSparks	26 de julho de 2022		
Amazon S3 Glacier	30 de setembro de 2021		
Amazon GuardDuty	30 de setembro de 2021		
AWS HealthLake	30 de setembro de 2021		
Amazon Honeycode	30 de setembro de 2021		
Amazon Inspector	30 de setembro de 2021		
Amazon Inspector V2	19 de abril de 2022		
Amazon Interactive Video Service	30 de setembro de 2021		
Amazon Kendra	30 de setembro de 2021		
Amazon Kinesis	30 de setembro de 2021		
Amazon Kinesis Analytics	30 de setembro de 2021		
Amazon Kinesis Analytics V2	30 de setembro de 2021		

Serviço	Suporte inicial	Atualizado	
Amazon Kinesis Firehose	30 de setembro de 2021		
Amazon Kinesis Video Signaling Channels	30 de setembro de 2021		
Amazon Kinesis Video Streams	30 de setembro de 2021	17 de fevereiro de 2023	
Amazon Kinesis Video Streams Archived Media	30 de setembro de 2021		
Amazon Kinesis video stream	30 de setembro de 2021		
Amazon Lex Model Building Service	30 de setembro de 2021		
Amazon Lex Model Building Service V2	30 de setembro de 2021	17 de fevereiro de 2023	
Amazon Lex	30 de setembro de 2021		
Amazon Lex Runtime V2	30 de setembro de 2021		
Amazon Lightsail	30 de setembro de 2021	17 de fevereiro de 2023	
Amazon Location Service	30 de setembro de 2021	17 de fevereiro de 2023	
Amazon Lookout for Equipment	30 de setembro de 2021		

Serviço	Suporte inicial	Atualizado	
Amazon Lookout for Metrics	30 de setembro de 2021	17 de fevereiro de 2023	
Amazon Lookout for Vision	30 de setembro de 2021		
Amazon MQ	30 de setembro de 2021		
Amazon Macie	30 de setembro de 2021		
Amazon Macie 2	30 de setembro de 2021	17 de fevereiro de 2023	
Amazon Managed Blockchain	30 de setembro de 2021	17 de fevereiro de 2023	
Amazon Managed Grafana	19 de abril de 2022	17 de fevereiro de 2023	
Amazon Managed Service for Prometheus	30 de setembro de 2021	17 de fevereiro de 2023	
Amazon Managed Streaming for Apache Kafka	30 de setembro de 2021	17 de fevereiro de 2023	
Amazon Managed Streaming for Apache Kafkax	19 de abril de 2022		
Amazon Managed Workflows for Apache Airflow	30 de setembro de 2021		

Serviço	Suporte inicial	Atualizado	
Amazon Mechanical Turk	30 de setembro de 2021		
Amazon MemoryDB for Redis	19 de abril de 2022	17 de fevereiro de 2023	
Amazon Nimble Studio	30 de setembro de 2021		
Amazon Personalize	30 de setembro de 2021	17 de fevereiro de 2023	
Amazon Personalize Events	30 de setembro de 2021		
Amazon Personalize Runtime	30 de setembro de 2021		
Amazon Pinpoint	30 de setembro de 2021		
Amazon Pinpoint Email Service	30 de setembro de 2021		
Amazon Pinpoint SMS and Voice Service	30 de setembro de 2021		
Amazon Pinpoint SMS and Voice V2 Service	26 de julho de 2022		
Amazon Polly	30 de setembro de 2021		
Amazon QLDB	30 de setembro de 2021		

Serviço	Suporte inicial	Atualizado	
Amazon QLDB Session	30 de setembro de 2021		
Amazon QuickSight	30 de setembro de 2021	17 de fevereiro de 2023	
Amazon Redshift	30 de setembro de 2021		
Amazon Redshift Data API	30 de setembro de 2021		
Amazon Rekognition	30 de setembro de 2021	17 de fevereiro de 2023	
Amazon Relational Database Service	30 de setembro de 2021	17 de fevereiro de 2023	
Amazon Route 53	30 de setembro de 2021		
Amazon Route 53 Recovery Control Config	30 de setembro de 2021	26 de julho de 2022	
Amazon Route 53 Domains	30 de setembro de 2021	17 de fevereiro de 2023	
Amazon Route 53 Resolver	30 de setembro de 2021		
Amazon S3 on Outposts	30 de setembro de 2021	26 de julho de 2022	
Amazon SageMaker Runtime Feature Store Runtime	30 de setembro de 2021		

Serviço	Suporte inicial	Atualizado	
Amazon SageMaker Runtime Runtime	30 de setembro de 2021		
Amazon SageMaker	30 de setembro de 2021	17 de fevereiro de 2023	
Amazon SageMaker Edge Manager	30 de setembro de 2021		
Amazon Simple Email Service	30 de setembro de 2021		
Amazon Simple Email Service V2	30 de setembro de 2021	17 de fevereiro de 2023	
Amazon Simple Notification Service	30 de setembro de 2021	17 de fevereiro de 2023	
Amazon Simple Queue Service	30 de setembro de 2021	17 de fevereiro de 2023	
Amazon Simple Storage Service	30 de setembro de 2021	17 de fevereiro de 2023	
Amazon Simple Workflow Service	30 de setembro de 2021		
Amazon Textract	30 de setembro de 2021	17 de fevereiro de 2023	
Amazon Transcribe	30 de setembro de 2021		
Amazon Translate	30 de setembro de 2021	17 de fevereiro de 2023	
Amazon WorkDocs	30 de setembro de 2021	17 de fevereiro de 2023	

Serviço	Suporte inicial	Atualizado	
Amazon WorkMail	30 de setembro de 2021	17 de fevereiro de 2023	
Amazon WorkMail Message Flow	30 de setembro de 2021		
Amazon WorkSpaces	30 de setembro de 2021	17 de fevereiro de 2023	
Amazon WorkSpaces Web	19 de abril de 2022	17 de fevereiro de 2023	
Amplify	30 de setembro de 2021		
Amplify UI Builder	19 de abril de 2022	17 de fevereiro de 2023	
Application Auto Scaling	30 de setembro de 2021		
Amazon EC2 Auto Scaling	30 de setembro de 2021	17 de fevereiro de 2023	
CodeArtifact	30 de setembro de 2021		
DynamoDB Accelerator	30 de setembro de 2021		
EC2 Image Builder	30 de setembro de 2021		
AWS Elastic Disaster Recovery	19 de abril de 2022	17 de fevereiro de 2023	
Elastic Load Balancing	30 de setembro de 2021		

Serviço	Suporte inicial	Atualizado	
Elastic Load Balancing V2	30 de setembro de 2021		
MediaConnect	30 de setembro de 2021		
Amazon S3 Control	30 de setembro de 2021	17 de fevereiro de 2023	
Recycle Bin for Amazon EBS	19 de abril de 2022	17 de fevereiro de 2023	
Savings Plans	30 de setembro de 2021		
Amazon EventBridge Schema Registry	30 de setembro de 2021		
Service Quotas	30 de setembro de 2021		
AWS Snowball	30 de setembro de 2021		

Projetos de amostra para Step Functions

No console AWS Step Functions, você pode escolher um dos seguintes modelos iniciais para implantar máquinas de estado no seu Contas da AWS. Esses modelos iniciais são exemplos de projetos prontos para execução que criam automaticamente o protótipo e a definição do fluxo de trabalho e todos os recursos AWS relacionados ao projeto.

Você pode usar esses projetos de amostra para implantá-los e executá-los como estão ou usar os protótipos de fluxo de trabalho para desenvolvê-los. Se você se basear nesses projetos, o Step Functions cria o protótipo do fluxo de trabalho, mas não implanta os recursos listados na definição do fluxo de trabalho.

Ao implantar os projetos de amostra, eles oferecem uma máquina de estado totalmente funcional e criam os recursos relacionados para que a máquina de estado funcione. Quando você cria um projeto de amostra, o Step Functions usa o AWS CloudFormation para criar os recursos relacionados referenciados pela máquina de estado.

Tópicos

- Crie um AWS CodeBuild projeto (CodeBuild, Amazon SNS)
- Pré-processar dados e treinar um modelo de Machine Learning
- Exemplo de orquestração do Lambda
- Iniciar uma consulta do Athena
- Executar várias consultas (Amazon Athena, Amazon SNS)
- Consulte grandes conjuntos de dados (Amazon Athena, Amazon S3, Amazon SNS AWS Glue)
- Mantenha os dados atualizados (Amazon Athena, Amazon S3,) AWS Glue
- Gerenciar um cluster do Amazon EKS
- Fazer uma chamada para o API Gateway
- Chamar um microsserviço em execução no Fargate usando a integração do API Gateway
- Envie um evento personalizado para EventBridge
- Invocar fluxos de trabalho expresso síncronos
- Executar fluxos de trabalho ETL/ELT usando o Amazon Redshift (Lambda, API de dados do Amazon Redshift)
- Uso Step Functions e AWS Batch com tratamento de erros
- Distribuir um AWS Batch emprego
- AWS Batch com Lambda
- Realizar o encadeamento de prompts de IA com o Amazon Bedrock

Gerenciar uma tarefa em lote (AWS Batch, Amazon SNS)

Este projeto de exemplo demonstra como enviar um trabalho do AWS Batch e, depois, enviar uma notificação do Amazon SNS com base no êxito ou na falha do trabalho. A implantação desse projeto de exemplo criará uma máquina de estado do AWS Step Functions, um trabalho do AWS Batch e um tópico do Amazon SNS.

Nesse projeto, o Step Functions usa uma máquina de estado para chamar o trabalho AWS Batch de maneira síncrona. Em seguida, ele aguarda até o êxito ou a falha do trabalho e envia um tópico do Amazon SNS com uma mensagem sobre o êxito ou a falha do trabalho.

Etapa 1: Criar a máquina de estado e provisionar os recursos

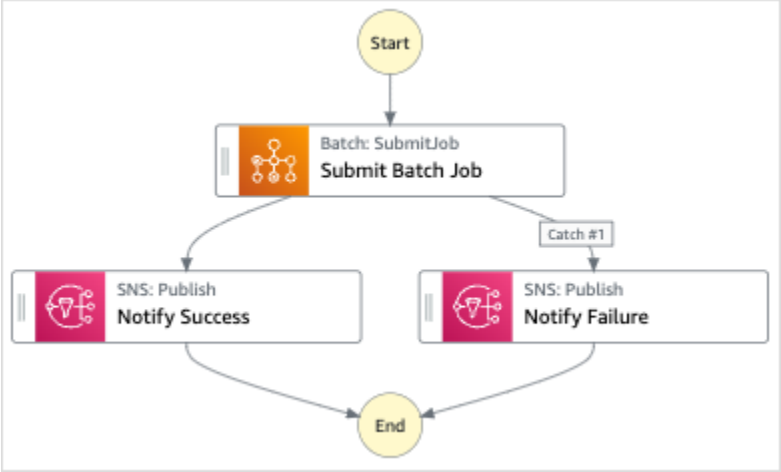
1. Abra o console do Step Functions e clique em Criar máquina de estado.

2. Digite **`Manage a batch job`** na caixa de pesquisa e escolha Gerenciar uma tarefa em lotes nos resultados da pesquisa que são retornados.
3. Escolha Próximo para continuar.
4. Step Functions lista o Serviços da AWS usado no projeto de amostra que você selecionou. Também mostra um gráfico de fluxo de trabalho para o projeto de amostra. Implante esse projeto no seu Conta da AWS ou use-o como ponto de partida para criar seus próprios projetos. Com base em como você deseja prosseguir, escolha Executar uma demonstração ou Criar com base nela.

   Este projeto de exemplo implementa os recursos a seguir.

   - Um AWS Batch emprego
   - Um tópico do Amazon SNS
   - Uma máquina de AWS Step Functions estado
   - Funções relacionadas AWS Identity and Access Management (IAM)

   A imagem a seguir mostra o gráfico do fluxo de trabalho para o projeto de exemplo Gerenciar uma tarefa em lote:



5. Escolha Usar modelo para continuar com a seleção.
6. Execute um destes procedimentos:

   - Se você selecionou Criar com base nela, o Step Functions criará o protótipo do fluxo de trabalho para o projeto de amostra selecionado. O Step Functions não implanta os recursos listados na definição do fluxo de trabalho.

No Modo de design do Workflow Studio, arraste e solte os estados do Navegador de estados para continuar criando seu protótipo de fluxo de trabalho. Ou mude para o Modo de código que fornece um editor de código integrado semelhante ao VS Code para atualizar a definição Amazon States Language (ASL) de sua máquina de estado no console Step Functions. Para obter mais informações sobre o uso do Workflow Studio para criar suas máquinas de estado, consulte Como usar o Workflow Studio.

> **Important**
>
> Lembre-se de atualizar o espaço reservado do nome do recurso da Amazon (ARN) para os recursos usados no projeto de amostra antes de executar o fluxo de trabalho.

- Se você selecionou Executar uma demonstração, o Step Functions cria um projeto de amostra somente para leitura que usa um AWS CloudFormation modelo para implantar os AWS recursos listados nesse modelo no seu. Conta da AWS

  > **Tip**
  >
  > Para visualizar a definição da máquina de estado do projeto de amostra, escolha Código.

  Quando estiver pronto, escolha Implemente e execute para implantar o projeto de amostra e criar os recursos.

  Pode levar até 10 minutos para que esses recursos e as permissões relacionadas ao IAM sejam criados. Enquanto seus recursos estão sendo implantados, você pode abrir o link do CloudFormation Stack ID para ver quais recursos estão sendo provisionados.

  Depois que todos os recursos do projeto de exemplo forem criados, você poderá ver o novo projeto de exemplo listado na página Máquinas de estado.

  > **Important**
  >
  > Taxas padrão podem ser aplicadas a cada serviço usado no CloudFormation modelo.

Etapa 2: Executar a máquina de estado

1. Na página Máquinas de estado, escolha seu projeto de exemplo.
2. Na página do projeto de exemplo, escolha Iniciar execução.
3. Na caixa de diálogo Iniciar execução, faça o seguinte:

    1. (Opcional) Para identificar a execução, insira um nome para ela no campo Nome. Por padrão, o Step Functions cria automaticamente um nome de execução exclusivo.

        > **Note**
        >
        > Step Functions permite criar nomes para máquinas de estado, execuções e atividades, além de rótulos que contenham caracteres não ASCII. Esses nomes não ASCII não funcionam com a Amazon. CloudWatch Para garantir que você possa acompanhar CloudWatch as métricas, escolha um nome que use somente caracteres ASCII.

    2. (Opcional) No campo Entrada, insira os valores de entrada no formato JSON para executar o fluxo de trabalho.

        Se você optar por Executar uma demonstração, não precisará fornecer nenhuma entrada de execução.

        > **Note**
        >
        > Se o projeto de demonstração que você implementou contiver dados de entrada de execução pré-preenchidos, use essa entrada para executar a máquina de estado.

    3. Selecione Iniciar execução.
    4. O console do Step Functions direciona você para uma página em que o título é o ID da execução. Essa página é conhecida como página de Detalhes da execução. Nesta página, você pode revisar os resultados da execução à medida que a execução avança ou após a conclusão.

        Para revisar os resultados da execução, escolha estados individuais na Exibição em gráfico e, em seguida, escolha as guias individuais no painel Detalhes da etapa para visualizar os detalhes de cada estado, incluindo entrada, saída e definição, respectivamente. Para obter

detalhes sobre as informações de execução que você pode visualizar na página Detalhes da execução, consulte Página de Detalhes da execução — Visão geral da interface.

Exemplo de código da máquina de estado

A máquina de estado neste projeto de amostra se integra AWS Batch ao Amazon SNS passando parâmetros diretamente para esses recursos.

Navegue por este exemplo de máquina de estado para ver como o Step Functions controla AWS Batch o Amazon SNS conectando-se ao Amazon Resource Name (ARN) no `Resource` campo e passando `Parameters` para a API do serviço.

Para obter mais informações sobre como AWS Step Functions controlar outros AWS serviços, consulteUsando AWS Step Functions com outros serviços.

```
{
  "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
    }
  }
}
```

Exemplo do IAM

Esse exemplo de política AWS Identity and Access Management (IAM) gerado pelo projeto de amostra inclui o menor privilégio necessário para executar a máquina de estado e os recursos relacionados. Recomendamos que você inclua apenas as permissões que forem necessárias em suas políticas do 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 obter informações sobre como configurar o IAM ao usar Step Functions com outros AWS serviços, consulte[Políticas do IAM para serviços integrados](#).

Gerenciar uma tarefa de contêiner (Amazon ECS, Amazon SNS)

Este projeto de exemplo demonstra como executar uma tarefa do AWS Fargate e, depois, enviar uma notificação do Amazon SNS com base no êxito ou na falha do trabalho. A implementação desse projeto de exemplo criará uma máquina de estado do AWS Step Functions, um cluster do Fargate e um tópico do Amazon SNS.

Nesse projeto, o Step Functions usa uma máquina de estado para chamar a tarefa Fargate de maneira síncrona. Em seguida, ele aguarda até o êxito ou a falha da tarefa e envia um tópico do Amazon SNS com uma mensagem sobre o êxito ou a falha do trabalho.

Etapa 1: Criar a máquina de estado e provisionar os recursos

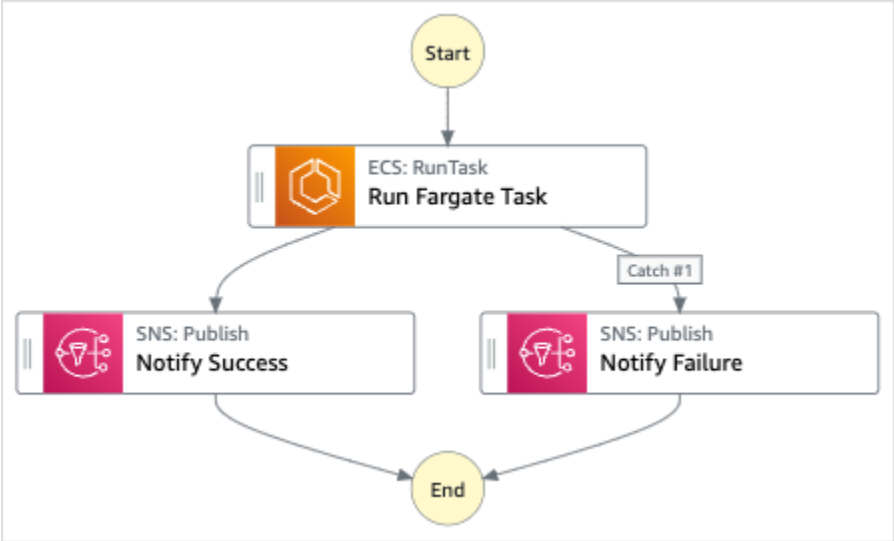
1. Abra o console do Step Functions e clique em Criar máquina de estado.
2. Digite **`Manage a container task`** no campo de pesquisa e escolha Gerenciar uma tarefa de contêiner nos resultados da pesquisa que são retornados.
3. Escolha Próximo para continuar.

4. Step Functions lista o Serviços da AWS usado no projeto de amostra que você selecionou. Também mostra um gráfico de fluxo de trabalho para o projeto de amostra. Implante esse projeto no seu Conta da AWS ou use-o como ponto de partida para criar seus próprios projetos. Com base em como você deseja prosseguir, escolha Executar uma demonstração ou Criar com base nela.

   Este projeto de exemplo implementa os recursos a seguir.

   - Um AWS Fargate cluster
   - Um tópico do Amazon SNS
   - Uma máquina de AWS Step Functions estado
   - Funções relacionadas AWS Identity and Access Management (IAM)

   A imagem a seguir mostra o gráfico do fluxo de trabalho do projeto de exemplo de Gerenciar uma tarefa de contêiner:



5. Escolha Usar modelo para continuar com a seleção.
6. Execute um destes procedimentos:

   - Se você selecionou Criar com base nela, o Step Functions criará o protótipo do fluxo de trabalho para o projeto de amostra selecionado. O Step Functions não implanta os recursos listados na definição do fluxo de trabalho.

     No Modo de design do Workflow Studio, arraste e solte os estados do Navegador de estados para continuar criando seu protótipo de fluxo de trabalho. Ou mude para o Modo de código que fornece um editor de código integrado semelhante ao VS Code para atualizar a definição

Amazon States Language (ASL) de sua máquina de estado no console Step Functions. Para obter mais informações sobre o uso do Workflow Studio para criar suas máquinas de estado, consulte Como usar o Workflow Studio.

> **Important**
>
> Lembre-se de atualizar o espaço reservado do nome do recurso da Amazon (ARN) para os recursos usados no projeto de amostra antes de executar o fluxo de trabalho.

- Se você selecionou Executar uma demonstração, o Step Functions cria um projeto de amostra somente para leitura que usa um AWS CloudFormation modelo para implantar os AWS recursos listados nesse modelo no seu. Conta da AWS

  > **Tip**
  >
  > Para visualizar a definição da máquina de estado do projeto de amostra, escolha Código.

  Quando estiver pronto, escolha Implemente e execute para implantar o projeto de amostra e criar os recursos.

  Pode levar até 10 minutos para que esses recursos e as permissões relacionadas ao IAM sejam criados. Enquanto seus recursos estão sendo implantados, você pode abrir o link do CloudFormation Stack ID para ver quais recursos estão sendo provisionados.

  Depois que todos os recursos do projeto de exemplo forem criados, você poderá ver o novo projeto de exemplo listado na página Máquinas de estado.

  > **Important**
  >
  > Taxas padrão podem ser aplicadas a cada serviço usado no CloudFormation modelo.

Etapa 2: Executar a máquina de estado

1. Na página Máquinas de estado, escolha seu projeto de exemplo.
2. Na página do projeto de exemplo, escolha Iniciar execução.

3. Na caixa de diálogo Iniciar execução, faça o seguinte:

   1. (Opcional) Para identificar a execução, insira um nome para ela no campo Nome. Por padrão, o Step Functions cria automaticamente um nome de execução exclusivo.

      > **Note**
      >
      > Step Functions permite criar nomes para máquinas de estado, execuções e atividades, além de rótulos que contenham caracteres não ASCII. Esses nomes não ASCII não funcionam com a Amazon. CloudWatch Para garantir que você possa acompanhar CloudWatch as métricas, escolha um nome que use somente caracteres ASCII.

   2. (Opcional) No campo Entrada, insira os valores de entrada no formato JSON para executar o fluxo de trabalho.

      Se você optar por Executar uma demonstração, não precisará fornecer nenhuma entrada de execução.

      > **Note**
      >
      > Se o projeto de demonstração que você implementou contiver dados de entrada de execução pré-preenchidos, use essa entrada para executar a máquina de estado.

   3. Selecione Iniciar execução.
   4. O console do Step Functions direciona você para uma página em que o título é o ID da execução. Essa página é conhecida como página de Detalhes da execução. Nesta página, você pode revisar os resultados da execução à medida que a execução avança ou após a conclusão.

      Para revisar os resultados da execução, escolha estados individuais na Exibição em gráfico e, em seguida, escolha as guias individuais no painel Detalhes da etapa para visualizar os detalhes de cada estado, incluindo entrada, saída e definição, respectivamente. Para obter detalhes sobre as informações de execução que você pode visualizar na página Detalhes da execução, consulte Página de Detalhes da execução — Visão geral da interface.

Exemplo de código da máquina de estado

A máquina de estado neste projeto de amostra se integra AWS Fargate ao Amazon SNS passando parâmetros diretamente para esses recursos. Explore essa máquina de estado de exemplo para ver como o Step Functions usa uma máquina de estado para chamar a tarefa do Fargate de maneira síncrona, aguarda até que a tarefa seja bem-sucedida ou falhe e envia um tópico do Amazon SNS com uma mensagem informando se o trabalho foi bem-sucedido ou falhou.

Para obter mais informações sobre como AWS Step Functions controlar outros AWS serviços, consulteUsando AWS Step Functions com outros serviços.

```
{
  "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
    }
  }
}
```

Exemplo do IAM

Esse exemplo de política AWS Identity and Access Management (IAM) gerado pelo projeto de amostra inclui o menor privilégio necessário para executar a máquina de estado e os recursos relacionados. É uma prática recomendada para incluir somente as permissões necessárias nas suas políticas do 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 obter informações sobre como configurar o IAM ao usar Step Functions com outros AWS serviços, consulte[Políticas do IAM para serviços integrados](#).

Transferir registros de dados (Lambda, DynamoDB, Amazon SQS)

Este exemplo de projeto demonstra como ler iterativamente itens de uma tabela Amazon DynamoDB e enviar esses itens para uma fila Amazon SQS usando uma máquina de estado Step Functions. A

implantação deste exemplo de projeto criará uma máquina de estado do Step Functions, uma tabela do DynamoDB, uma função AWS Lambda e uma fila do Amazon SQS.

Neste projeto, Step Functions usa a função Lambda para preencher a tabela DynamoDB. A máquina de estado também usa um loop `for` para ler cada uma das entradas e, em seguida, envia cada entrada para uma fila Amazon SQS.

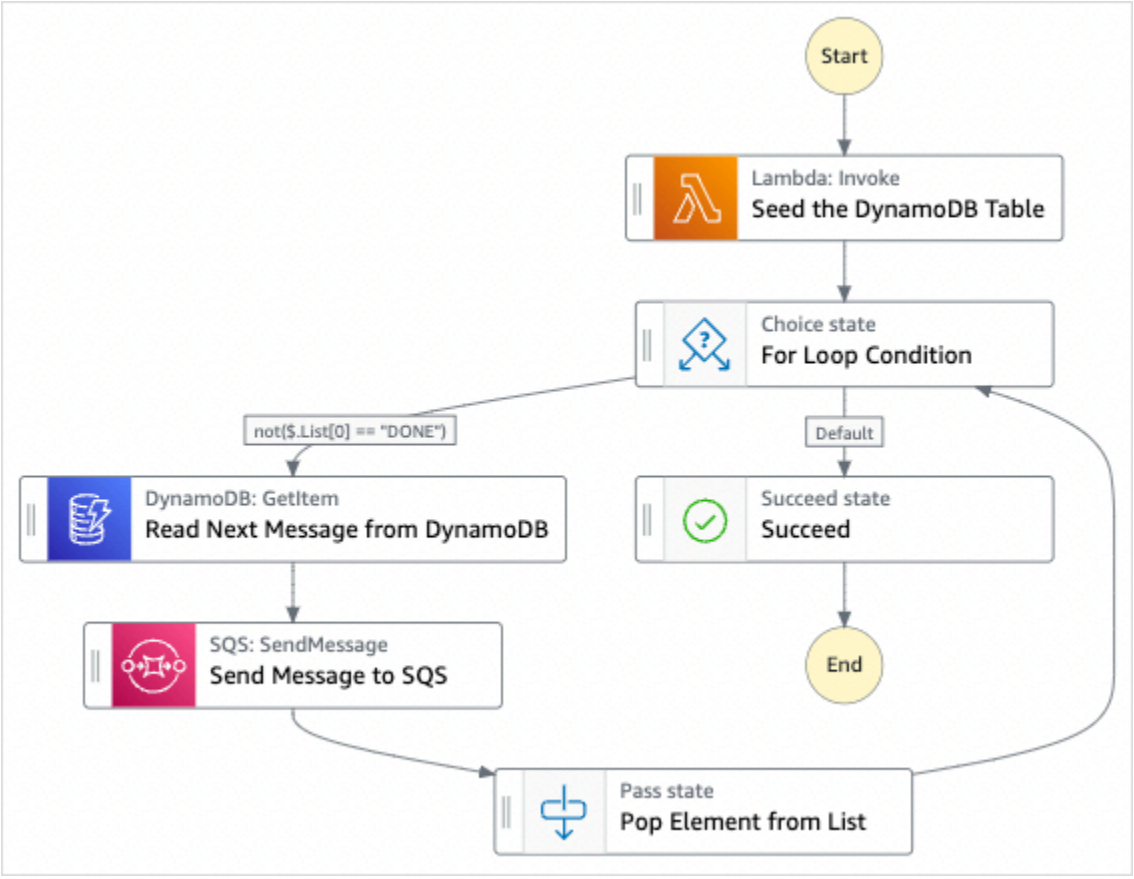
Etapa 1: Criar a máquina de estado e provisionar os recursos

1. Abra o console do Step Functions e clique em Criar máquina de estado.
2. Digite **`Transfer data records`** na caixa de pesquisa e escolha Transferir registros de dados nos resultados da pesquisa que são retornados.
3. Escolha Próximo para continuar.
4. Step Functions lista o Serviços da AWS usado no projeto de amostra que você selecionou. Também mostra um gráfico de fluxo de trabalho para o projeto de amostra. Implante esse projeto no seu Conta da AWS ou use-o como ponto de partida para criar seus próprios projetos. Com base em como você deseja prosseguir, escolha Executar uma demonstração ou Criar com base nela.

   Este projeto de exemplo implementa os recursos a seguir.

   - Uma função do Lambda para propagar a tabela do DynamoDB
   - Uma fila do Amazon SQS
   - Uma tabela do DynamoDB
   - Uma máquina de AWS Step Functions estado
   - Funções relacionadas AWS Identity and Access Management (IAM)

   A imagem a seguir mostra o gráfico do fluxo de trabalho do exemplo de projeto do Transferir registros de dados:



5. Escolha Usar modelo para continuar com a seleção.
6. Execute um destes procedimentos:

   - Se você selecionou Criar com base nela, o Step Functions criará o protótipo do fluxo de trabalho para o projeto de amostra selecionado. O Step Functions não implanta os recursos listados na definição do fluxo de trabalho.

     No Modo de design do Workflow Studio, arraste e solte os estados do Navegador de estados para continuar criando seu protótipo de fluxo de trabalho. Ou mude para o Modo de código que fornece um editor de código integrado semelhante ao VS Code para atualizar a definição Amazon States Language (ASL) de sua máquina de estado no console Step Functions. Para obter mais informações sobre o uso do Workflow Studio para criar suas máquinas de estado, consulte Como usar o Workflow Studio.

     > **Important**
     >
     > Lembre-se de atualizar o espaço reservado do nome do recurso da Amazon (ARN) para os recursos usados no projeto de amostra antes de executar o fluxo de trabalho.

- Se você selecionou Executar uma demonstração, o Step Functions cria um projeto de amostra somente para leitura que usa um AWS CloudFormation modelo para implantar os AWS recursos listados nesse modelo no seu. Conta da AWS

  > **Tip**
  >
  > Para visualizar a definição da máquina de estado do projeto de amostra, escolha Código.

  Quando estiver pronto, escolha Implemente e execute para implantar o projeto de amostra e criar os recursos.

  Pode levar até 10 minutos para que esses recursos e as permissões relacionadas ao IAM sejam criados. Enquanto seus recursos estão sendo implantados, você pode abrir o link do CloudFormation Stack ID para ver quais recursos estão sendo provisionados.

  Depois que todos os recursos do projeto de exemplo forem criados, você poderá ver o novo projeto de exemplo listado na página Máquinas de estado.

  > **Important**
  >
  > Taxas padrão podem ser aplicadas a cada serviço usado no CloudFormation modelo.

Etapa 2: Executar a máquina de estado

1. Na página Máquinas de estado, escolha seu projeto de exemplo.
2. Na página do projeto de exemplo, escolha Iniciar execução.
3. Na caixa de diálogo Iniciar execução, faça o seguinte:

   1. (Opcional) Para identificar a execução, insira um nome para ela no campo Nome. Por padrão, o Step Functions cria automaticamente um nome de execução exclusivo.

      > **Note**
      >
      > Step Functions permite criar nomes para máquinas de estado, execuções e atividades, além de rótulos que contenham caracteres não ASCII. Esses nomes não

ASCII não funcionam com a Amazon. CloudWatch Para garantir que você possa acompanhar CloudWatch as métricas, escolha um nome que use somente caracteres ASCII.

2. (Opcional) No campo Entrada, insira os valores de entrada no formato JSON para executar o fluxo de trabalho.

   Se você optar por Executar uma demonstração, não precisará fornecer nenhuma entrada de execução.

   > **Note**
   >
   > Se o projeto de demonstração que você implementou contiver dados de entrada de execução pré-preenchidos, use essa entrada para executar a máquina de estado.

3. Selecione Iniciar execução.
4. O console do Step Functions direciona você para uma página em que o título é o ID da execução. Essa página é conhecida como página de Detalhes da execução. Nesta página, você pode revisar os resultados da execução à medida que a execução avança ou após a conclusão.

   Para revisar os resultados da execução, escolha estados individuais na Exibição em gráfico e, em seguida, escolha as guias individuais no painel Detalhes da etapa para visualizar os detalhes de cada estado, incluindo entrada, saída e definição, respectivamente. Para obter detalhes sobre as informações de execução que você pode visualizar na página Detalhes da execução, consulte Página de Detalhes da execução — Visão geral da interface.

Exemplo de código da máquina de estado

A máquina de estado deste projeto de exemplo se integra ao DynamoDB e ao Amazon SQS transmitindo parâmetros diretamente para esses recursos.

Navegue por este exemplo de máquina de estado para ver como o Step Functions controla o DynamoDB e o Amazon SQS conectando-se ao nome do recurso da Amazon (ARN) no campo `Resource` e passando `Parameters` para a API de serviço.

Para obter mais informações sobre como AWS Step Functions controlar outros AWS serviços, consulteUsando AWS Step Functions com outros serviços.

```
{
  "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 obter mais informações sobre como passar parâmetros e gerenciar resultados, consulte o seguinte:

- Transmitir parâmetros para uma API de serviço
- ResultPath

Exemplo do IAM

Esse exemplo de política AWS Identity and Access Management (IAM) gerado pelo projeto de amostra inclui o menor privilégio necessário para executar a máquina de estado e os recursos relacionados. É uma prática recomendada para incluir somente as permissões necessárias nas suas políticas do 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 obter informações sobre como configurar o IAM ao usar Step Functions com outros AWS serviços, consulte[Políticas do IAM para serviços integrados](#).

Pesquisa de status de trabalho (AWS Batch Lambda,)

Esse projeto de amostra cria uma pesquisa de AWS Batch empregos. Ele implementa uma máquina de AWS Step Functions estado usada AWS Lambda para criar um loop de `Wait` estado que verifica um AWS Batch trabalho.

Esse projeto de amostra cria e configura todos os recursos para que seu fluxo de trabalho do Step Functions envie um AWS Batch trabalho e espere que esse trabalho seja concluído antes de terminar com êxito.

> **Note**
>
> Também é possível implementar esse padrão sem usar uma função do Lambda. Para obter informações sobre o controle AWS Batch direto, consulteUsando AWS Step Functions com outros serviços.

Esse projeto de amostra cria a máquina de estado, duas funções Lambda e uma AWS Batch fila, além de configurar as permissões do IAM relacionadas.

Para obter mais informações sobre como AWS Step Functions controlar outros AWS serviços, consulteUsando AWS Step Functions com outros serviços.

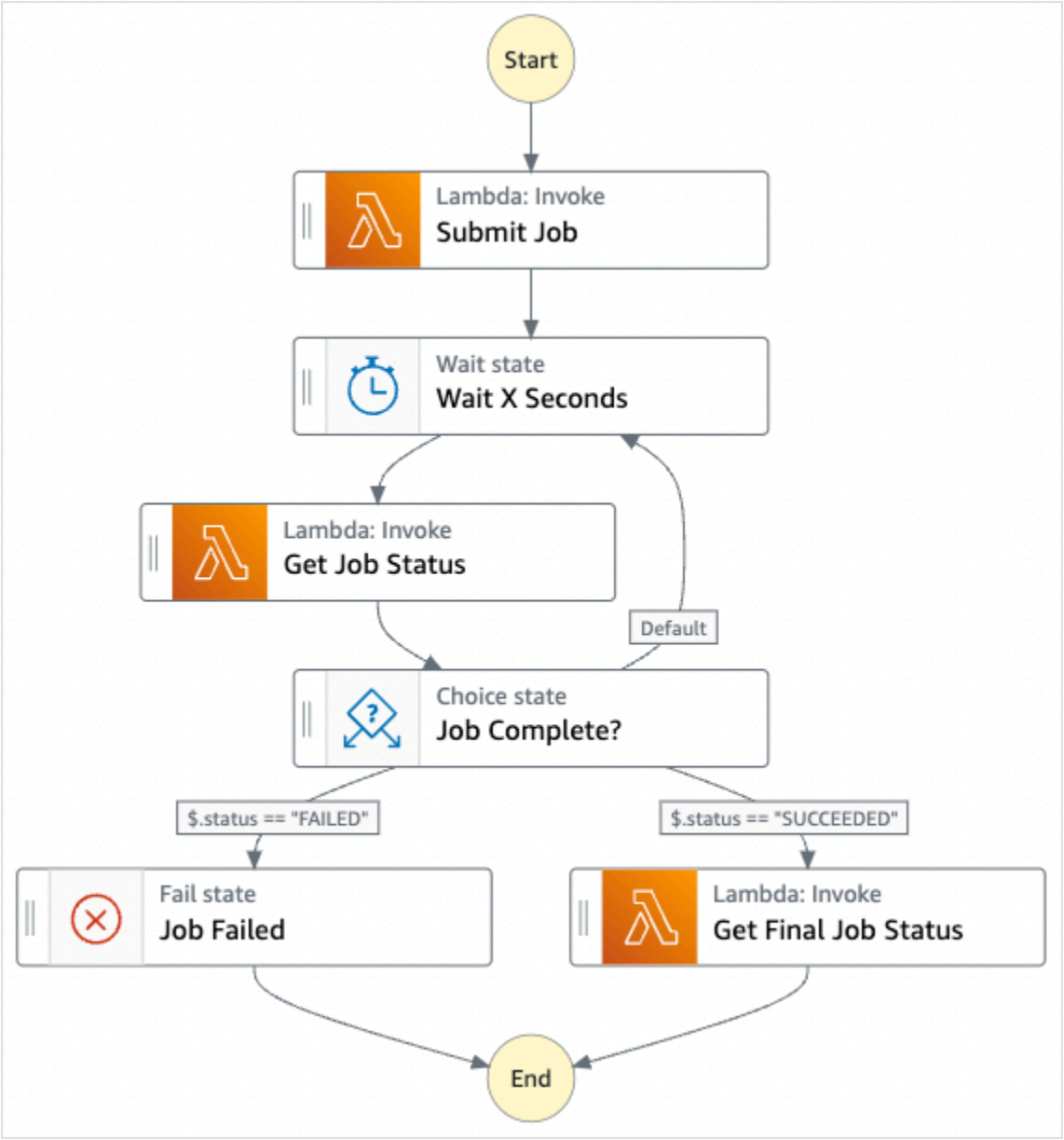
Etapa 1: Criar a máquina de estado e provisionar os recursos

1. Abra o console do Step Functions e clique em Criar máquina de estado.
2. Digite **`Job Poller`** na caixa de pesquisa e escolha Instrumento de sondagem de trabalho nos resultados da pesquisa que são retornados.
3. Escolha Próximo para continuar.
4. Step Functions lista o Serviços da AWS usado no projeto de amostra que você selecionou. Também mostra um gráfico de fluxo de trabalho para o projeto de amostra. Implante esse projeto no seu Conta da AWS ou use-o como ponto de partida para criar seus próprios projetos. Com base em como você deseja prosseguir, escolha Executar uma demonstração ou Criar com base nela.

   Este projeto de exemplo implementa os recursos a seguir.

   - Três funções do Lambda para enviar um AWS Batch trabalho, obter o status atual do AWS Batch trabalho enviado e o status final de conclusão do trabalho.
   - Um AWS Batch emprego
   - Uma máquina de AWS Step Functions estado
   - Funções relacionadas AWS Identity and Access Management (IAM)

   A imagem a seguir mostra o gráfico do fluxo de trabalho do exemplo de projeto do Instrumento de sondagem de trabalho:



5. Escolha Usar modelo para continuar com a seleção.
6. Execute um destes procedimentos:

   - Se você selecionou Criar com base nela, o Step Functions criará o protótipo do fluxo de trabalho para o projeto de amostra selecionado. O Step Functions não implanta os recursos listados na definição do fluxo de trabalho.

     No Modo de design do Workflow Studio, arraste e solte os estados do Navegador de estados para continuar criando seu protótipo de fluxo de trabalho. Ou mude para o Modo de código que fornece um editor de código integrado semelhante ao VS Code para atualizar a definição Amazon States Language (ASL) de sua máquina de estado no console Step Functions. Para

obter mais informações sobre o uso do Workflow Studio para criar suas máquinas de estado, consulte Como usar o Workflow Studio.

> **Important**
>
> Lembre-se de atualizar o espaço reservado do nome do recurso da Amazon (ARN) para os recursos usados no projeto de amostra antes de executar o fluxo de trabalho.

- Se você selecionou Executar uma demonstração, o Step Functions cria um projeto de amostra somente para leitura que usa um AWS CloudFormation modelo para implantar os AWS recursos listados nesse modelo no seu. Conta da AWS

  > **Tip**
  >
  > Para visualizar a definição da máquina de estado do projeto de amostra, escolha Código.

  Quando estiver pronto, escolha Implemente e execute para implantar o projeto de amostra e criar os recursos.

  Pode levar até 10 minutos para que esses recursos e as permissões relacionadas ao IAM sejam criados. Enquanto seus recursos estão sendo implantados, você pode abrir o link do CloudFormation Stack ID para ver quais recursos estão sendo provisionados.

  Depois que todos os recursos do projeto de exemplo forem criados, você poderá ver o novo projeto de exemplo listado na página Máquinas de estado.

  > **Important**
  >
  > Taxas padrão podem ser aplicadas a cada serviço usado no CloudFormation modelo.

Etapa 2: Executar a máquina de estado

Depois que todos os recursos forem provisionados e implantados, a caixa de diálogo Iniciar execução será exibida com um exemplo de entrada semelhante ao seguinte:

```
{
```

```
    "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` instrui o estado `Wait` a fazer loop a cada 60 segundos.

- Na caixa de diálogo Iniciar execução, faça o seguinte:

  1. (Opcional) Para identificar a execução, insira um nome para ela no campo Nome. Por padrão, o Step Functions cria automaticamente um nome de execução exclusivo.

     > **Note**
     >
     > Step Functions permite criar nomes para máquinas de estado, execuções e atividades, além de rótulos que contenham caracteres não ASCII. Esses nomes não ASCII não funcionam com a Amazon. CloudWatch Para garantir que você possa acompanhar CloudWatch as métricas, escolha um nome que use somente caracteres ASCII.

  2. (Opcional) No campo Entrada, insira os valores de entrada no formato JSON para executar o fluxo de trabalho.

     Se você optar por Executar uma demonstração, não precisará fornecer nenhuma entrada de execução.

     > **Note**
     >
     > Se o projeto de demonstração que você implementou contiver dados de entrada de execução pré-preenchidos, use essa entrada para executar a máquina de estado.

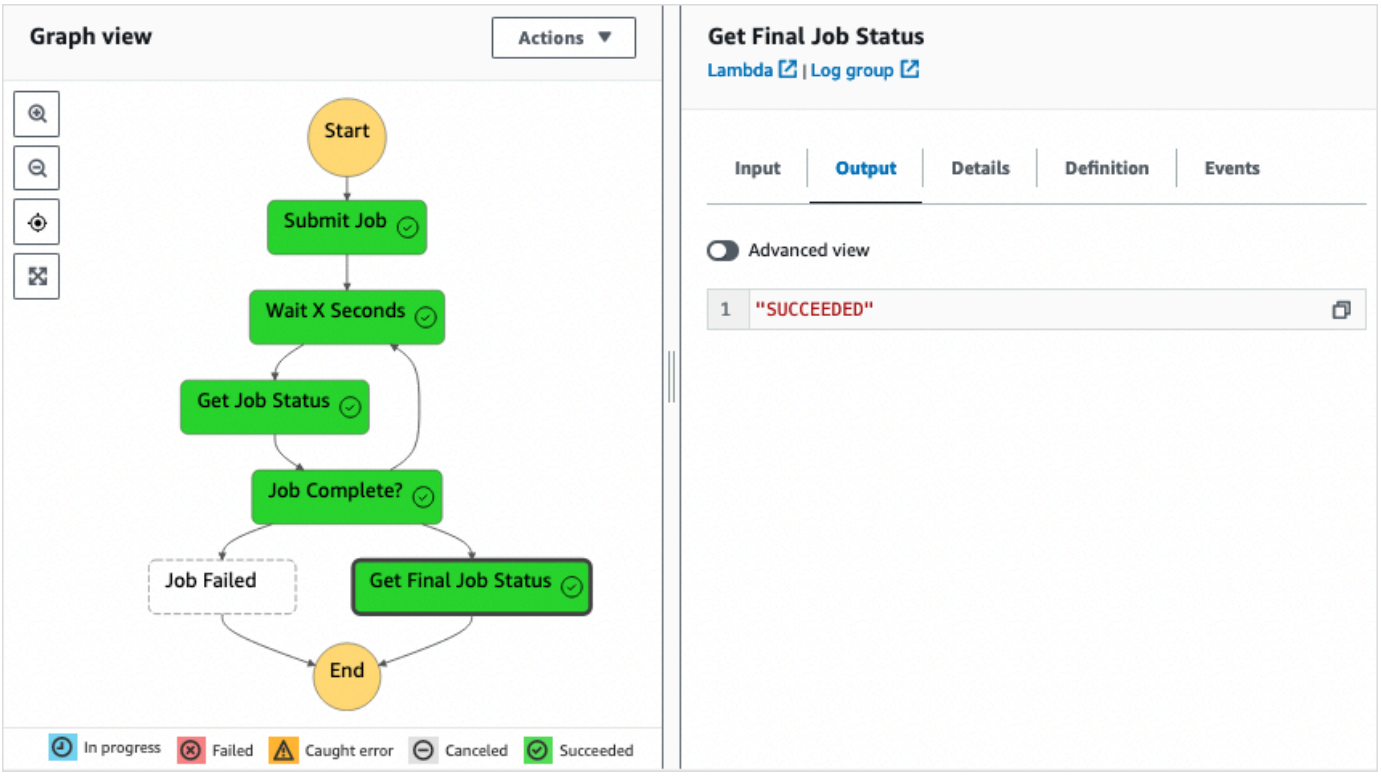
  3. Selecione Iniciar execução.

4. O console do Step Functions direciona você para uma página em que o título é o ID da execução. Essa página é conhecida como página de Detalhes da execução. Nesta página, você pode revisar os resultados da execução à medida que a execução avança ou após a conclusão.

   Para revisar os resultados da execução, escolha estados individuais na Exibição em gráfico e, em seguida, escolha as guias individuais no painel Detalhes da etapa para visualizar os detalhes de cada estado, incluindo entrada, saída e definição, respectivamente. Para obter detalhes sobre as informações de execução que você pode visualizar na página Detalhes da execução, consulte Página de Detalhes da execução — Visão geral da interface.

Por exemplo, para ver o status de mudança do seu AWS Batch trabalho e os resultados em loop da sua execução, escolha a guia Saída.

A imagem a seguir mostra o gráfico do estado de execução na Visualização de gráfico. Também mostra a saída de execução da etapa selecionada na guia Saída.



Exemplo de código da máquina de estado

A máquina de estado neste projeto de amostra se integra AWS Lambda ao envio de um AWS Batch trabalho. Navegue por este exemplo de máquina de estado para ver como o Step Functions controla o Lambda e. AWS Batch

Para obter mais informações sobre como AWS Step Functions controlar outros AWS serviços, consulteUsando AWS Step Functions com outros serviços.

```
{
    "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 tarefas (Lambda, Amazon SNS)

Esse projeto de amostra cria um temporizador de tarefas. Ele implementa uma máquina de AWS Step Functions estado que implementa um `Wait` estado e usa uma AWS Lambda função que envia uma notificação do Amazon Simple Notification Service (Amazon SNS). O estado Aguardar é um tipo de estado que aguarda um trigger para executar uma unidade de trabalho específica.

> **Note**
>
> Esse projeto de amostra implementa uma AWS Lambda função para enviar uma notificação do Amazon Simple Notification Service (Amazon SNS). Você também pode enviar uma notificação do Amazon SNS diretamente do Amazon States Language. Consulte Usando AWS Step Functions com outros serviços.

Esse projeto de amostra cria a máquina de estado, uma função Lambda e um tópico do Amazon SNS, além de configurar as permissões AWS Identity and Access Management relacionadas (IAM). Para obter mais informações sobre os recursos criados com o projeto de exemplo Task Timer (Temporizador de tarefas), consulte o seguinte:

Para obter mais informações sobre como AWS Step Functions controlar outros AWS serviços, consulteUsando AWS Step Functions com outros serviços.

- AWS CloudFormation Guia do usuário
- Guia do desenvolvedor do Amazon Simple Notification Service
- AWS Lambda Guia do desenvolvedor
- Guia de conceitos básicos do IAM

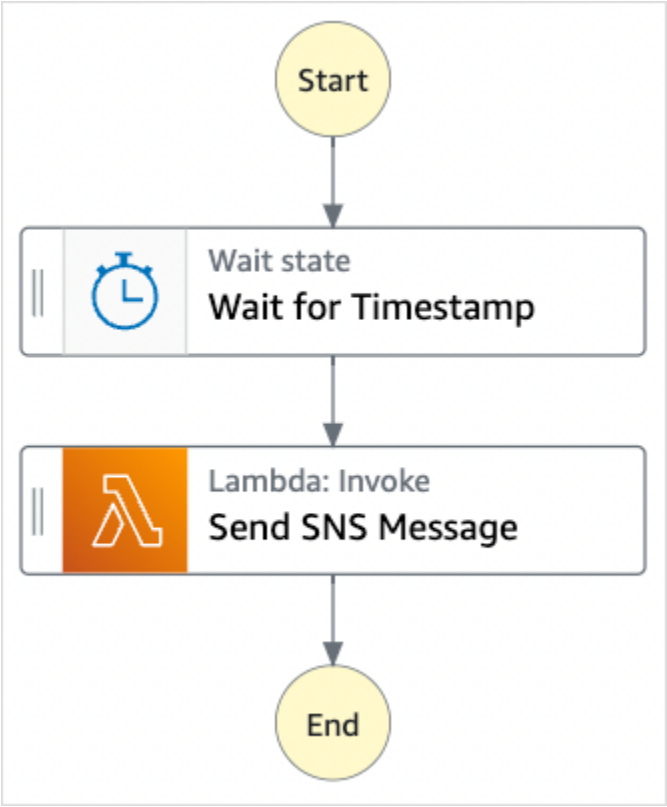
Etapa 1: Criar a máquina de estado e provisionar os recursos

1. Abra o console do Step Functions e clique em Criar máquina de estado.
2. Digite **`Task Timer`** na caixa de pesquisa e escolha Temporizador de tarefas nos resultados da pesquisa que são retornados.
3. Escolha Próximo para continuar.
4. Step Functions lista o Serviços da AWS usado no projeto de amostra que você selecionou. Também mostra um gráfico de fluxo de trabalho para o projeto de amostra. Implante esse projeto no seu Conta da AWS ou use-o como ponto de partida para criar seus próprios projetos. Com base em como você deseja prosseguir, escolha Executar uma demonstração ou Criar com base nela.

   Este projeto de exemplo implementa os recursos a seguir.

   - uma função do Lambda que envia uma notificação do Amazon SNS.
   - Uma máquina de AWS Step Functions estado
   - Funções relacionadas AWS Identity and Access Management (IAM)

   A imagem a seguir mostra o gráfico do fluxo de trabalho do exemplo de projeto do Temporizador de tarefas:



5. Escolha Usar modelo para continuar com a seleção.
6. Execute um destes procedimentos:

   - Se você selecionou Criar com base nela, o Step Functions criará o protótipo do fluxo de trabalho para o projeto de amostra selecionado. O Step Functions não implanta os recursos listados na definição do fluxo de trabalho.

     No Modo de design do Workflow Studio, arraste e solte os estados do Navegador de estados para continuar criando seu protótipo de fluxo de trabalho. Ou mude para o Modo de código que fornece um editor de código integrado semelhante ao VS Code para atualizar a definição Amazon States Language (ASL) de sua máquina de estado no console Step Functions. Para obter mais informações sobre o uso do Workflow Studio para criar suas máquinas de estado, consulte Como usar o Workflow Studio.

     > **Important**
     >
     > Lembre-se de atualizar o espaço reservado do nome do recurso da Amazon (ARN) para os recursos usados no projeto de amostra antes de executar o fluxo de trabalho.

- Se você selecionou Executar uma demonstração, o Step Functions cria um projeto de amostra somente para leitura que usa um AWS CloudFormation modelo para implantar os AWS recursos listados nesse modelo no seu. Conta da AWS

  > **Tip**
  >
  > Para visualizar a definição da máquina de estado do projeto de amostra, escolha Código.

  Quando estiver pronto, escolha Implemente e execute para implantar o projeto de amostra e criar os recursos.

  Pode levar até 10 minutos para que esses recursos e as permissões relacionadas ao IAM sejam criados. Enquanto seus recursos estão sendo implantados, você pode abrir o link do CloudFormation Stack ID para ver quais recursos estão sendo provisionados.

  Depois que todos os recursos do projeto de exemplo forem criados, você poderá ver o novo projeto de exemplo listado na página Máquinas de estado.

  > **Important**
  >
  > Taxas padrão podem ser aplicadas a cada serviço usado no CloudFormation modelo.

Etapa 2: Executar a máquina de estado

Depois que todos os recursos forem provisionados e implantados, a caixa de diálogo Iniciar execução será exibida com um exemplo de entrada semelhante ao seguinte:

```
{
    "jobName": "my-job",{
  "topic": "arn:aws:sns:us-east-2:123456789012:StepFunctionsSample-TaskTimercc68840e-
c3d3-42a8-911e-821b7ce248e5-SNSTopic-44UjcFxzhACT",
  "message": "HelloWorld",
  "timer_seconds": 10
}
```

- Na caixa de diálogo Iniciar execução, faça o seguinte:

1. (Opcional) Para identificar a execução, insira um nome para ela no campo Nome. Por padrão, o Step Functions cria automaticamente um nome de execução exclusivo.

   > **Note**
   >
   > Step Functions permite criar nomes para máquinas de estado, execuções e atividades, além de rótulos que contenham caracteres não ASCII. Esses nomes não ASCII não funcionam com a Amazon. CloudWatch Para garantir que você possa acompanhar CloudWatch as métricas, escolha um nome que use somente caracteres ASCII.

2. (Opcional) No campo Entrada, insira os valores de entrada no formato JSON para executar o fluxo de trabalho.

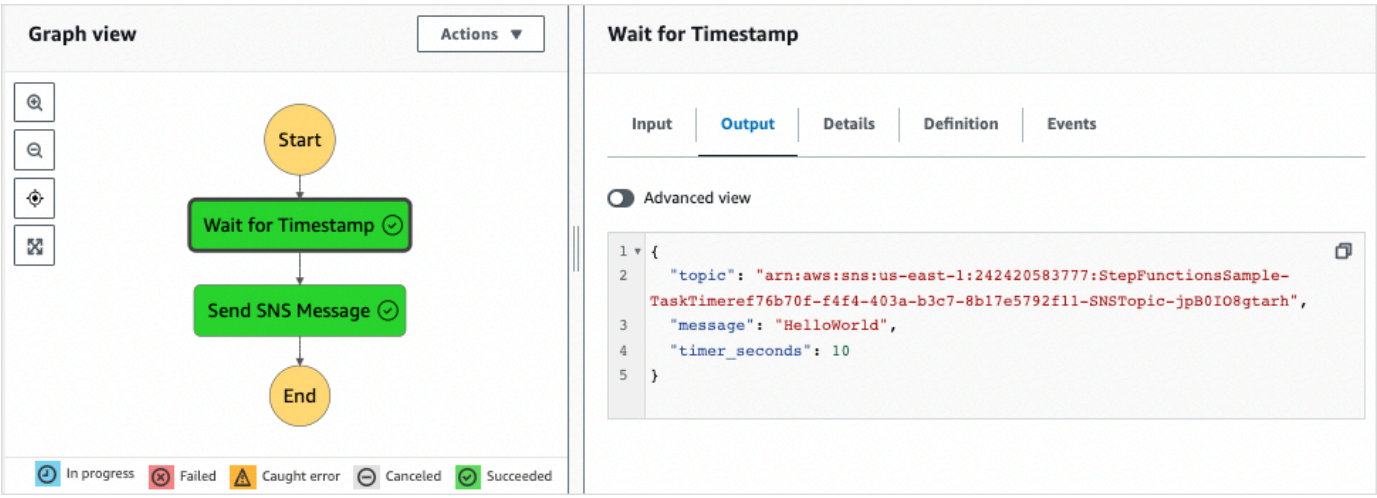
   Se você optar por Executar uma demonstração, não precisará fornecer nenhuma entrada de execução.

   > **Note**
   >
   > Se o projeto de demonstração que você implementou contiver dados de entrada de execução pré-preenchidos, use essa entrada para executar a máquina de estado.

3. Selecione Iniciar execução.
4. O console do Step Functions direciona você para uma página em que o título é o ID da execução. Essa página é conhecida como página de Detalhes da execução. Nesta página, você pode revisar os resultados da execução à medida que a execução avança ou após a conclusão.

   Para revisar os resultados da execução, escolha estados individuais na Exibição em gráfico e, em seguida, escolha as guias individuais no painel Detalhes da etapa para visualizar os detalhes de cada estado, incluindo entrada, saída e definição, respectivamente. Para obter detalhes sobre as informações de execução que você pode visualizar na página Detalhes da execução, consulte Página de Detalhes da execução — Visão geral da interface.

Por exemplo, a imagem a seguir mostra a saída da etapa selecionada Aguarde pelo Timestamp. A saída dessa etapa é passada como entrada para a etapa Enviar mensagem SNS.



Exemplo de Padrão de Retorno de Chamada (Amazon SQS, Amazon SNS, Lambda)

Este projeto de amostra demonstra como fazer uma AWS Step Functions pausa durante uma tarefa e esperar que um processo externo retorne um token de tarefa que foi gerado quando a tarefa foi iniciada.

Quando esse projeto de exemplo é implantado, e uma execução é iniciada, as seguintes etapas ocorrem:

1. O Step Functions transmite uma mensagem que inclui um token de tarefa para uma fila do Amazon Simple Queue Service (Amazon SQS).
2. O Step Functions, então, pausa, aguardando o retorno desse token.
3. A fila do Amazon SQS aciona uma AWS Lambda função que chama `SendTaskSuccess`com o mesmo token de tarefa.
4. Quando o token é recebido, o fluxo de trabalho continua.
5. A tarefa `"Notify Success"` publica uma mensagem do Amazon Simple Notification Service (Amazon SNS) informando que o retorno de chamada foi recebido.

Para saber como implementar o padrão de retorno de chamada no Step Functions, consulte Aguardar um retorno de chamada com um token de tarefa.

Para obter mais informações sobre como AWS Step Functions controlar outros AWS serviços, consulteUsando AWS Step Functions com outros serviços.

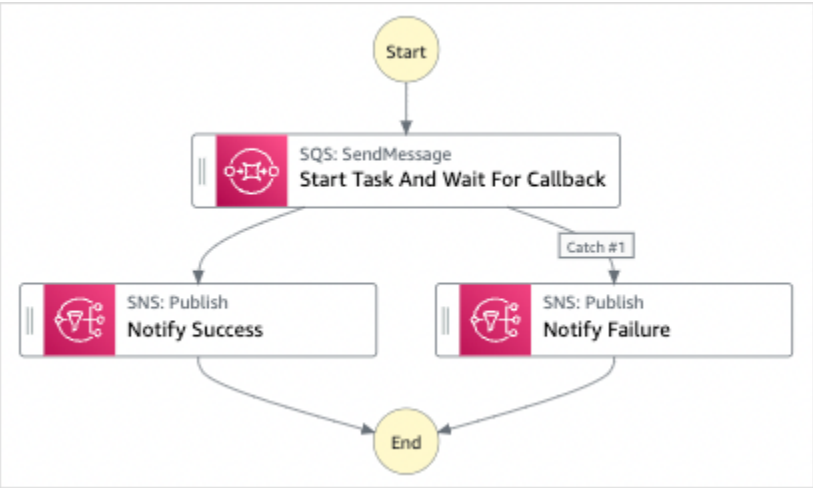
Etapa 1: Criar a máquina de estado e provisionar os recursos

1. Abra o console do Step Functions e clique em Criar máquina de estado.
2. Digite **`Callback pattern example`** no campo de pesquisa e escolha Exemplo de padrão de retorno de chamada nos resultados da pesquisa que são retornados.
3. Escolha Próximo para continuar.
4. Step Functions lista o Serviços da AWS usado no projeto de amostra que você selecionou. Também mostra um gráfico de fluxo de trabalho para o projeto de amostra. Implante esse projeto no seu Conta da AWS ou use-o como ponto de partida para criar seus próprios projetos. Com base em como você deseja prosseguir, escolha Executar uma demonstração ou Criar com base nela.

   Este projeto de exemplo implementa os recursos a seguir.

   - Uma fila de mensagens do Amazon SQS.
   - Uma função Lambda que chama a ação da API Step Functions. SendTaskSuccess
   - Um tópico do Amazon SNS para notificar o sucesso ou a falha de uma tarefa, indicando se o fluxo de trabalho pode ou não continuar.
   - Uma máquina de AWS Step Functions estado
   - Funções relacionadas AWS Identity and Access Management (IAM)

   A imagem a seguir mostra o gráfico do fluxo de trabalho para o projeto de exemplo de Exemplo do padrão de retorno de chamada:



5. Escolha Usar modelo para continuar com a seleção.

6. Execute um destes procedimentos:

   - Se você selecionou Criar com base nela, o Step Functions criará o protótipo do fluxo de trabalho para o projeto de amostra selecionado. O Step Functions não implanta os recursos listados na definição do fluxo de trabalho.

     No Modo de design do Workflow Studio, arraste e solte os estados do Navegador de estados para continuar criando seu protótipo de fluxo de trabalho. Ou mude para o Modo de código que fornece um editor de código integrado semelhante ao VS Code para atualizar a definição Amazon States Language (ASL) de sua máquina de estado no console Step Functions. Para obter mais informações sobre o uso do Workflow Studio para criar suas máquinas de estado, consulte Como usar o Workflow Studio.

     > **Important**
     >
     > Lembre-se de atualizar o espaço reservado do nome do recurso da Amazon (ARN) para os recursos usados no projeto de amostra antes de executar o fluxo de trabalho.

   - Se você selecionou Executar uma demonstração, o Step Functions cria um projeto de amostra somente para leitura que usa um AWS CloudFormation modelo para implantar os AWS recursos listados nesse modelo no seu. Conta da AWS

     > **Tip**
     >
     > Para visualizar a definição da máquina de estado do projeto de amostra, escolha Código.

     Quando estiver pronto, escolha Implemente e execute para implantar o projeto de amostra e criar os recursos.

     Pode levar até 10 minutos para que esses recursos e as permissões relacionadas ao IAM sejam criados. Enquanto seus recursos estão sendo implantados, você pode abrir o link do CloudFormation Stack ID para ver quais recursos estão sendo provisionados.

     Depois que todos os recursos do projeto de exemplo forem criados, você poderá ver o novo projeto de exemplo listado na página Máquinas de estado.

> **Important**
>
> Taxas padrão podem ser aplicadas a cada serviço usado no CloudFormation modelo.

Etapa 2: Executar a máquina de estado

1. Na página Máquinas de estado, escolha seu projeto de exemplo.
2. Na página do projeto de exemplo, escolha Iniciar execução.
3. Na caixa de diálogo Iniciar execução, faça o seguinte:

    1. (Opcional) Para identificar a execução, insira um nome para ela no campo Nome. Por padrão, o Step Functions cria automaticamente um nome de execução exclusivo.

        > **Note**
        >
        > Step Functions permite criar nomes para máquinas de estado, execuções e atividades, além de rótulos que contenham caracteres não ASCII. Esses nomes não ASCII não funcionam com a Amazon. CloudWatch Para garantir que você possa acompanhar CloudWatch as métricas, escolha um nome que use somente caracteres ASCII.

    2. (Opcional) No campo Entrada, insira os valores de entrada no formato JSON para executar o fluxo de trabalho.

        Se você optar por Executar uma demonstração, não precisará fornecer nenhuma entrada de execução.

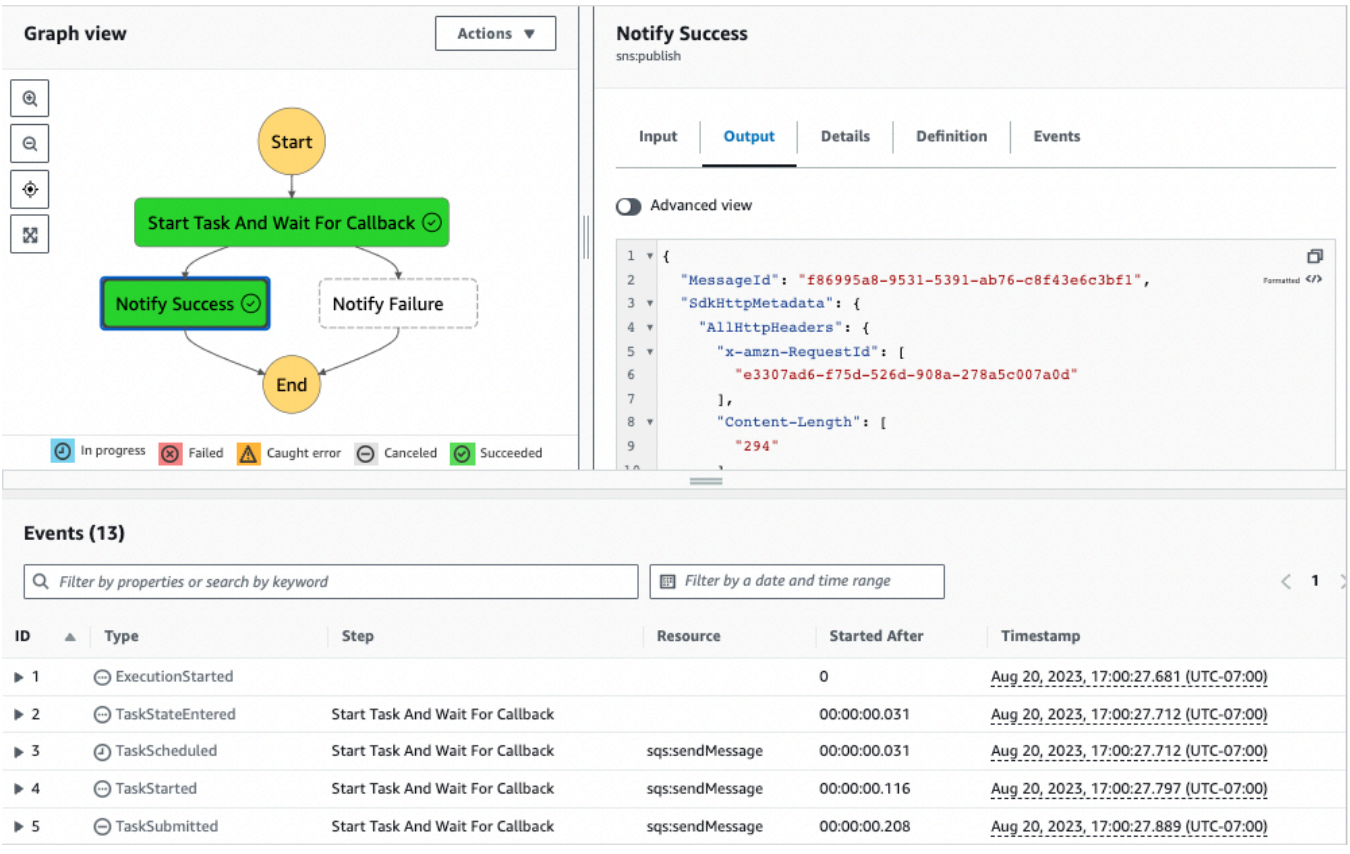
        > **Note**
        >
        > Se o projeto de demonstração que você implementou contiver dados de entrada de execução pré-preenchidos, use essa entrada para executar a máquina de estado.

    3. Selecione Iniciar execução.
    4. O console do Step Functions direciona você para uma página em que o título é o ID da execução. Essa página é conhecida como página de Detalhes da execução. Nesta página,

você pode revisar os resultados da execução à medida que a execução avança ou após a conclusão.

Para revisar os resultados da execução, escolha estados individuais na Exibição em gráfico e, em seguida, escolha as guias individuais no painel Detalhes da etapa para visualizar os detalhes de cada estado, incluindo entrada, saída e definição, respectivamente. Para obter detalhes sobre as informações de execução que você pode visualizar na página Detalhes da execução, consulte Página de Detalhes da execução — Visão geral da interface.

Por exemplo, para analisar como o Step Functions progrediu no fluxo de trabalho e recebeu um retorno de chamada do Amazon SQS, revise as entradas na tabela Eventos. A imagem a seguir mostra a saída de execução da etapa Notificar Sucesso. Também mostra os cinco primeiros eventos do histórico de eventos de execução. Expanda cada evento para ver mais detalhes sobre esse evento.



Exemplo de Retorno de Chamada do Lambda

Para ver como os componentes desse projeto de amostra funcionam juntos, veja os recursos que foram implantados em sua AWS conta. Por exemplo, veja a seguir a função do Lambda que chama o Step Functions com o token de tarefa.

```
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);
        });
    }
};
```

Gerenciamento de um Trabalho do Amazon EMR

Este exemplo de projeto demonstra o Amazon EMR AWS Step Functions e a integração.

Ele mostra como criar um cluster do Amazon EMR, adicionar e executar várias etapas e, por fim, encerrar o cluster.

> **Important**
>
> O Amazon EMR não tem uma camada de definição de preço gratuita. A execução do projeto de exemplo incorrerá em custos. Você pode encontrar informações sobre preços na página de Definição de preços do Amazon EMR. A disponibilidade da integração do serviço do Amazon EMR está sujeita à disponibilidade de APIs do Amazon EMR. Por esse motivo, esse projeto de amostra pode não funcionar corretamente em algumas AWS regiões. Consulte a documentação do Amazon EMR quanto a limitações em Regiões especiais.

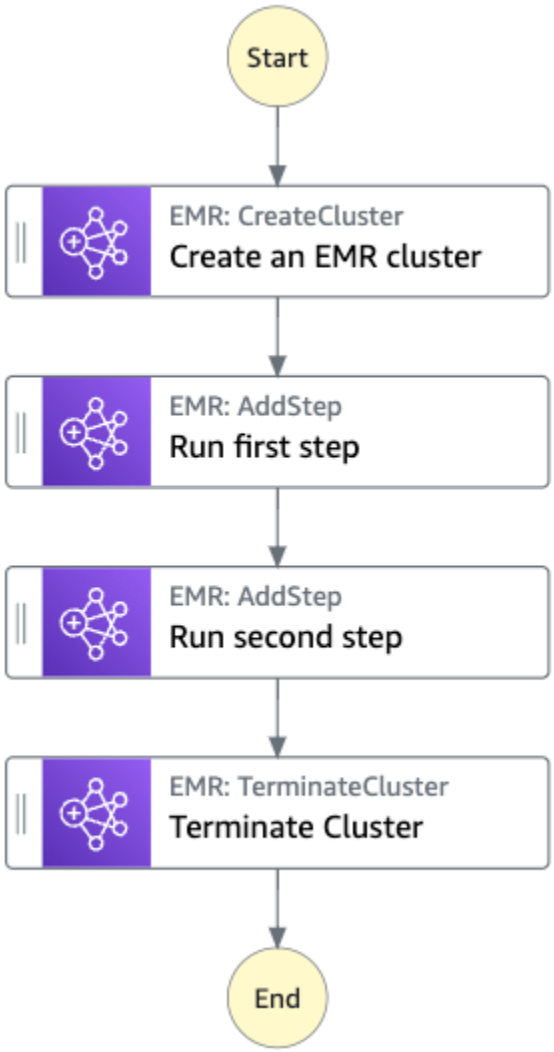
Etapa 1: Criar a máquina de estado e provisionar os recursos

1. Abra o console do Step Functions e clique em Criar máquina de estado.
2. Digite **`Manage an EMR job`** no campo de pesquisa e escolha Gerenciar um trabalho do EMR a partir dos resultados da pesquisa que são retornados.
3. Escolha Próximo para continuar.
4. Step Functions lista o Serviços da AWS usado no projeto de amostra que você selecionou. Também mostra um gráfico de fluxo de trabalho para o projeto de amostra. Implante esse projeto no seu Conta da AWS ou use-o como ponto de partida para criar seus próprios projetos. Com base em como você deseja prosseguir, escolha Executar uma demonstração ou Criar com base nela.

   Este projeto de exemplo implementa os recursos a seguir.

   - Um bucket do Amazon S3
   - Um cluster do Amazon EMR
   - Uma máquina de AWS Step Functions estado
   - Funções relacionadas AWS Identity and Access Management (IAM)

   A imagem a seguir mostra o gráfico do fluxo de trabalho para o projeto de exemplo Gerenciar um trabalho do EMR:



5. Escolha Usar modelo para continuar com a seleção.
6. Execute um destes procedimentos:

   - Se você selecionou Criar com base nela, o Step Functions criará o protótipo do fluxo de trabalho para o projeto de amostra selecionado. O Step Functions não implanta os recursos listados na definição do fluxo de trabalho.

     No Modo de design do Workflow Studio, arraste e solte os estados do Navegador de estados para continuar criando seu protótipo de fluxo de trabalho. Ou mude para o Modo de código que fornece um editor de código integrado semelhante ao VS Code para atualizar a definição Amazon States Language (ASL) de sua máquina de estado no console Step Functions. Para obter mais informações sobre o uso do Workflow Studio para criar suas máquinas de estado, consulte Como usar o Workflow Studio.

> **Important**
>
> Lembre-se de atualizar o espaço reservado do nome do recurso da Amazon (ARN) para os recursos usados no projeto de amostra antes de executar o fluxo de trabalho.

- Se você selecionou Executar uma demonstração, o Step Functions cria um projeto de amostra somente para leitura que usa um AWS CloudFormation modelo para implantar os AWS recursos listados nesse modelo no seu. Conta da AWS

  > **Tip**
  >
  > Para visualizar a definição da máquina de estado do projeto de amostra, escolha Código.

  Quando estiver pronto, escolha Implemente e execute para implantar o projeto de amostra e criar os recursos.

  Pode levar até 10 minutos para que esses recursos e as permissões relacionadas ao IAM sejam criados. Enquanto seus recursos estão sendo implantados, você pode abrir o link do CloudFormation Stack ID para ver quais recursos estão sendo provisionados.

  Depois que todos os recursos do projeto de exemplo forem criados, você poderá ver o novo projeto de exemplo listado na página Máquinas de estado.

  > **Important**
  >
  > Taxas padrão podem ser aplicadas a cada serviço usado no CloudFormation modelo.

Etapa 2: Executar a máquina de estado

1. Na página Máquinas de estado, escolha seu projeto de exemplo.
2. Na página do projeto de exemplo, escolha Iniciar execução.
3. Na caixa de diálogo Iniciar execução, faça o seguinte:

1. (Opcional) Para identificar a execução, insira um nome para ela no campo Nome. Por padrão, o Step Functions cria automaticamente um nome de execução exclusivo.

   > **Note**
   >
   > Step Functions permite criar nomes para máquinas de estado, execuções e atividades, além de rótulos que contenham caracteres não ASCII. Esses nomes não ASCII não funcionam com a Amazon. CloudWatch Para garantir que você possa acompanhar CloudWatch as métricas, escolha um nome que use somente caracteres ASCII.

2. (Opcional) No campo Entrada, insira os valores de entrada no formato JSON para executar o fluxo de trabalho.

   Se você optar por Executar uma demonstração, não precisará fornecer nenhuma entrada de execução.

   > **Note**
   >
   > Se o projeto de demonstração que você implementou contiver dados de entrada de execução pré-preenchidos, use essa entrada para executar a máquina de estado.

3. Selecione Iniciar execução.
4. O console do Step Functions direciona você para uma página em que o título é o ID da execução. Essa página é conhecida como página de Detalhes da execução. Nesta página, você pode revisar os resultados da execução à medida que a execução avança ou após a conclusão.

   Para revisar os resultados da execução, escolha estados individuais na Exibição em gráfico e, em seguida, escolha as guias individuais no painel Detalhes da etapa para visualizar os detalhes de cada estado, incluindo entrada, saída e definição, respectivamente. Para obter detalhes sobre as informações de execução que você pode visualizar na página Detalhes da execução, consulte Página de Detalhes da execução — Visão geral da interface.

Exemplo de código da máquina de estado

A máquina de estado deste projeto de exemplo se integra ao Amazon EMR aprovando os parâmetros diretamente para esses recursos. Navegue nesta máquina de estado de exemplo para ver como o Step Functions usa uma máquina de estado para chamar a tarefa do Amazon EMR de forma síncrona, aguarda o resultado de êxito ou falha e encerra o cluster.

Para obter mais informações sobre como AWS Step Functions controlar outros AWS serviços, consulteUsando AWS Step Functions com outros serviços.

```
{
  "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
    }
  }
}
```

Exemplo do IAM

Esse exemplo de política AWS Identity and Access Management (IAM) gerado pelo projeto de amostra inclui o menor privilégio necessário para executar a máquina de estado e os recursos relacionados. É uma prática recomendada para incluir somente as permissões necessárias nas suas políticas do 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"
            ]
        }
    ]
}
```

A política a seguir garante que o `addStep` tenha permissões 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 obter informações sobre como configurar o IAM ao usar Step Functions com outros AWS serviços, consulte Políticas do IAM para serviços integrados.

Executar uma tarefa do EMR Serverless

Este projeto de exemplo demonstra como você pode criar e iniciar um aplicativo do EMR Serverless. Este projeto também mostra como você pode executar vários trabalhos dentro desse aplicativo.

Esse projeto de amostra cria a máquina de estado, os AWS recursos de suporte e configura as permissões relacionadas do IAM. Explore esse projeto de exemplo para saber como executar EMR Serverless tarefas usando máquinas de estado do Step Functions ou usá-lo como ponto de partida para seus próprios projetos.

> **Important**
>
> O EMR Serverless não tem uma camada de definição de preço gratuita. A execução do projeto de exemplo incorrerá em custos. Você pode encontrar informações sobre preços na página de Definição de preços do Amazon EMR Serverless.
> Adicionalmente, a disponibilidade da integração do serviço do EMR Serverless está sujeita à disponibilidade de APIs do EMR Serverless. Por isso, este projeto de exemplo pode não funcionar corretamente ou não estar disponível em algumas Regiões da AWS. Consulte

o tópico Outras considerações para obter informações sobre a disponibilidade do EMR Serverless em Regiões da AWS.

Modelo do AWS CloudFormation e recursos adicionais

Você usa um modelo do CloudFormation para implementar esse projeto de exemplo. Esse modelo cria os seguintes recursos em seu Conta da AWS:

- Uma máquina de estado do Step Functions.
- Função de execução para a máquina de estado. Essa função concede as permissões que sua máquina de estado precisa para acessar outros recursos Serviços da AWS e recursos, como a EMR Serverless CreateApplicationação.
- Função de execução de função para o EMR Serverless. Essa função concede as permissões que a execução de uma tarefa do EMR Serverless pode assumir ao chamar outros serviços em seu nome.

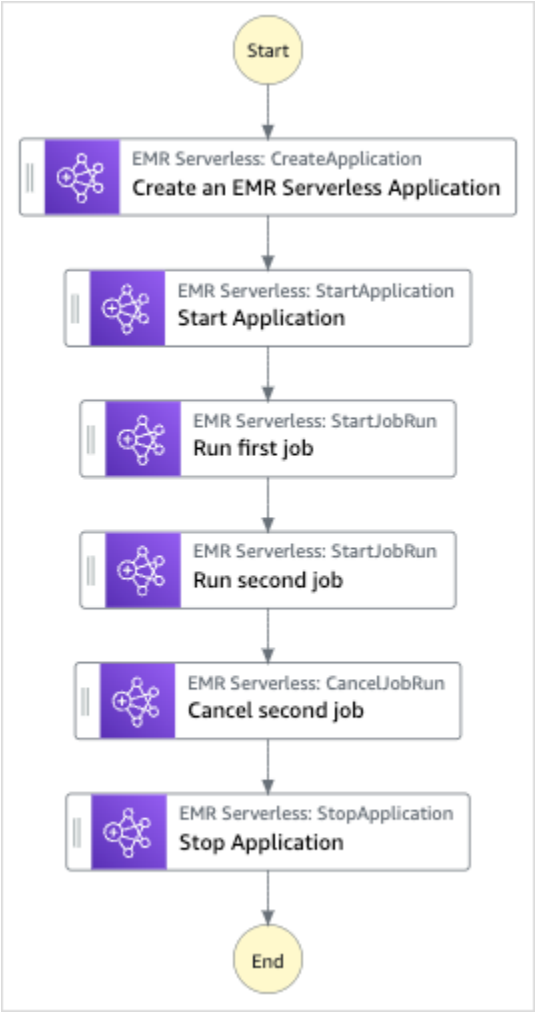
Etapa 1: Criar a máquina de estado e provisionar os recursos

1. Abra o console do Step Functions e clique em Criar máquina de estado.
2. Digite **`EMR Serverless`** no campo de pesquisa e escolha Executar uma tarefa do EMR Serverless a partir dos resultados de pesquisa que são retornados.
3. Escolha Próximo para continuar.
4. Step Functions lista o Serviços da AWS usado no projeto de amostra que você selecionou. Também mostra um gráfico de fluxo de trabalho para o projeto de amostra. Implante esse projeto no seu Conta da AWS ou use-o como ponto de partida para criar seus próprios projetos. Com base em como você deseja prosseguir, escolha Executar uma demonstração ou Criar com base nela.

   Este projeto de exemplo implementa os recursos a seguir.

   - Uma máquina de estado do Step Functions
   - Funções do AWS Identity and Access Management (IAM) relacionadas

   A imagem a seguir mostra o gráfico do fluxo de trabalho do projeto de exemplo para Executar uma tarefa do EMR Serverless:



5. Escolha Usar modelo para continuar com a seleção.
6. Execute um destes procedimentos:

   - Se você selecionou Criar com base nela, o Step Functions criará o protótipo do fluxo de trabalho para o projeto de amostra selecionado. O Step Functions não implanta os recursos listados na definição do fluxo de trabalho.

     No Modo de design do Workflow Studio, arraste e solte os estados do Navegador de estados para continuar criando seu protótipo de fluxo de trabalho. Ou mude para o Modo de código que fornece um editor de código integrado semelhante ao VS Code para atualizar a definição Amazon States Language (ASL) de sua máquina de estado no console Step Functions. Para obter mais informações sobre o uso do Workflow Studio para criar suas máquinas de estado, consulte Como usar o Workflow Studio.

> **Important**
>
> Lembre-se de atualizar o espaço reservado do nome do recurso da Amazon (ARN) para os recursos usados no projeto de amostra antes de executar o fluxo de trabalho.

- Se você selecionou Executar uma demonstração, o Step Functions cria um projeto de amostra somente para leitura que usa um AWS CloudFormation modelo para implantar os AWS recursos listados nesse modelo no seu. Conta da AWS

  > **Tip**
  >
  > Para visualizar a definição da máquina de estado do projeto de amostra, escolha Código.

  Quando estiver pronto, escolha Implemente e execute para implantar o projeto de amostra e criar os recursos.

  Pode levar até 10 minutos para que esses recursos e as permissões relacionadas ao IAM sejam criados. Enquanto seus recursos estão sendo implantados, você pode abrir o link do CloudFormation Stack ID para ver quais recursos estão sendo provisionados.

  Depois que todos os recursos do projeto de exemplo forem criados, você poderá ver o novo projeto de exemplo listado na página Máquinas de estado.

  > **Important**
  >
  > Taxas padrão podem ser aplicadas a cada serviço usado no CloudFormation modelo.

Etapa 2: Executar a máquina de estado

1. Na página Máquinas de estado, escolha seu projeto de exemplo.
2. Na página do projeto de exemplo, escolha Iniciar execução.
3. Na caixa de diálogo Iniciar execução, faça o seguinte:

1. (Opcional) Para identificar a execução, insira um nome para ela no campo Nome. Por padrão, o Step Functions gera automaticamente um nome de execução exclusivo.

   > **Note**
   >
   > O Step Functions permite criar nomes de máquina de estado, execuções, atividades e labels que contenham caracteres não ASCII. Esses nomes não ASCII não funcionam com a Amazon. CloudWatch Para garantir que você possa acompanhar CloudWatch as métricas, escolha um nome que use somente caracteres ASCII.

2. (Opcional) No campo Entrada, insira os valores de entrada no formato JSON para executar o fluxo de trabalho.

   Se você optar por Executar uma demonstração, não precisará fornecer nenhuma entrada de execução.

3. Selecione Iniciar execução.

4. O console do Step Functions direciona você para uma página em que o título é o ID da execução. Essa página é conhecida como página de Detalhes da execução. Nesta página, você pode revisar os resultados da execução à medida que a execução avança ou após a conclusão.

   Para revisar os resultados da execução, escolha estados individuais na Exibição em gráfico e, em seguida, escolha as guias individuais no painel Detalhes da etapa para visualizar os detalhes de cada estado, incluindo entrada, saída e definição, respectivamente. Para obter detalhes sobre as informações de execução que você pode visualizar na página Detalhes da execução, consulte Página de Detalhes da execução — Visão geral da interface.

Iniciar um fluxo de trabalho dentro de um fluxo de trabalho (Step Functions, Lambda)

Este exemplo de projeto demonstra como usar uma máquina de AWS Step Functions estado para iniciar outras execuções de máquina de estado. Para obter informações sobre iniciar execuções de máquinas de estado desde outra máquina de estado, consulte Iniciar execuções de fluxo de trabalho usando um estado Tarefa.

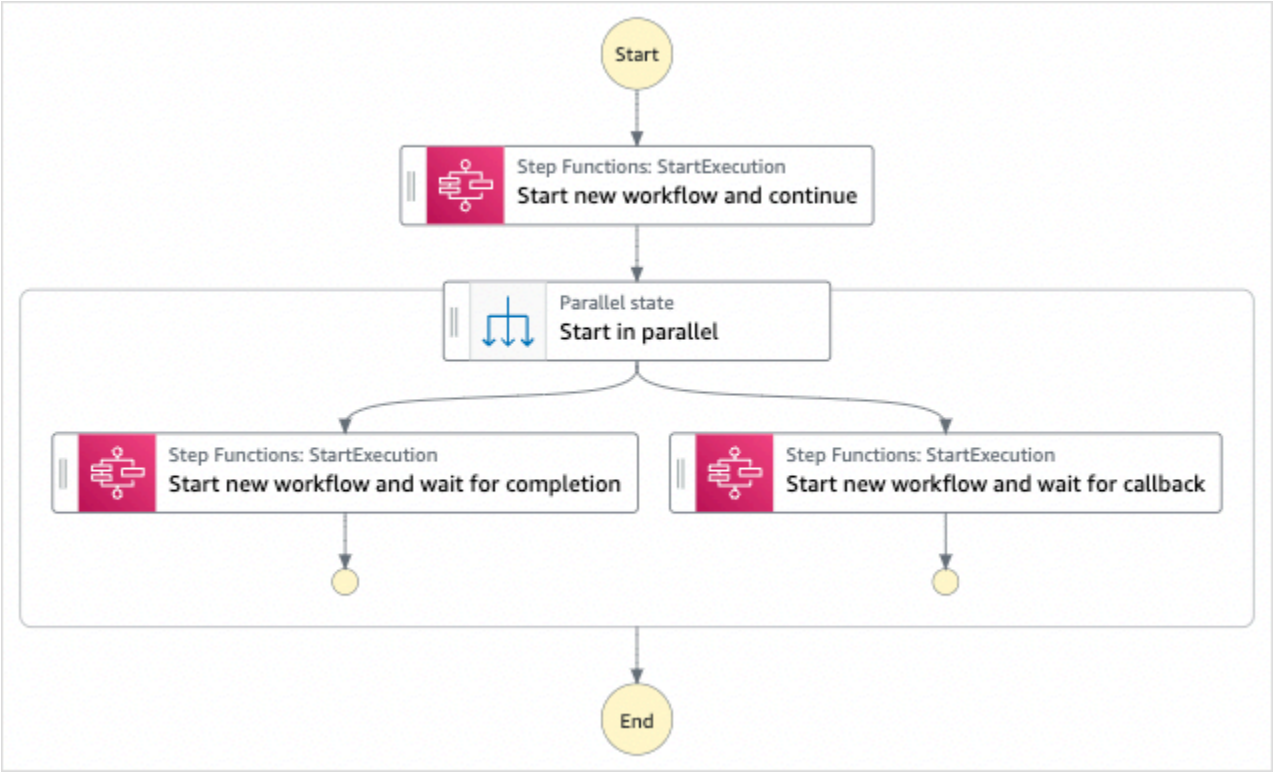
Etapa 1: Criar a máquina de estado e provisionar os recursos

1. Abra o console do Step Functions e clique em Criar máquina de estado.
2. Digite **`Start a workflow within a workflow`** na caixa de pesquisa e escolha Iniciar um fluxo de trabalho dentro de um fluxo de trabalho nos resultados da pesquisa que são retornados.
3. Escolha Próximo para continuar.
4. Step Functions lista o Serviços da AWS usado no projeto de amostra que você selecionou. Também mostra um gráfico de fluxo de trabalho para o projeto de amostra. Implante esse projeto no seu Conta da AWS ou use-o como ponto de partida para criar seus próprios projetos. Com base em como você deseja prosseguir, escolha Executar uma demonstração ou Criar com base nela.

   Este projeto de exemplo implementa os recursos a seguir.

   - Uma máquina de estado adicional. A execução dessa máquina de estado é iniciada por aquela que você executa.
   - Uma função do Lambda de chamada de retorno. Essa função é usada na máquina de estado adicional para implementar o mecanismo de retorno de chamada.
   - Uma máquina de AWS Step Functions estado
   - Funções relacionadas AWS Identity and Access Management (IAM)

   A imagem a seguir mostra o gráfico do fluxo de trabalho do projeto Iniciar um fluxo de trabalho em um exemplo de fluxo de trabalho:



5. Escolha Usar modelo para continuar com a seleção.
6. Execute um destes procedimentos:

   - Se você selecionou Criar com base nela, o Step Functions criará o protótipo do fluxo de trabalho para o projeto de amostra selecionado. O Step Functions não implanta os recursos listados na definição do fluxo de trabalho.

     No Modo de design do Workflow Studio, arraste e solte os estados do Navegador de estados para continuar criando seu protótipo de fluxo de trabalho. Ou mude para o Modo de código que fornece um editor de código integrado semelhante ao VS Code para atualizar a definição Amazon States Language (ASL) de sua máquina de estado no console Step Functions. Para obter mais informações sobre o uso do Workflow Studio para criar suas máquinas de estado, consulte Como usar o Workflow Studio.

     > **Important**
     >
     > Lembre-se de atualizar o espaço reservado do nome do recurso da Amazon (ARN) para os recursos usados no projeto de amostra antes de executar o fluxo de trabalho.

- Se você selecionou Executar uma demonstração, o Step Functions cria um projeto de amostra somente para leitura que usa um AWS CloudFormation modelo para implantar os AWS recursos listados nesse modelo no seu. Conta da AWS

  > **Tip**
  >
  > Para visualizar a definição da máquina de estado do projeto de amostra, escolha Código.

  Quando estiver pronto, escolha Implemente e execute para implantar o projeto de amostra e criar os recursos.

  Pode levar até 10 minutos para que esses recursos e as permissões relacionadas ao IAM sejam criados. Enquanto seus recursos estão sendo implantados, você pode abrir o link do CloudFormation Stack ID para ver quais recursos estão sendo provisionados.

  Depois que todos os recursos do projeto de exemplo forem criados, você poderá ver o novo projeto de exemplo listado na página Máquinas de estado.

  > **Important**
  >
  > Taxas padrão podem ser aplicadas a cada serviço usado no CloudFormation modelo.

Etapa 2: Executar a máquina de estado

1. Na página Máquinas de estado, escolha seu projeto de exemplo.
2. Na página do projeto de exemplo, escolha Iniciar execução.
3. Na caixa de diálogo Iniciar execução, faça o seguinte:

   1. (Opcional) Para identificar a execução, insira um nome para ela no campo Nome. Por padrão, o Step Functions cria automaticamente um nome de execução exclusivo.

      > **Note**
      >
      > Step Functions permite criar nomes para máquinas de estado, execuções e atividades, além de rótulos que contenham caracteres não ASCII. Esses nomes não

ASCII não funcionam com a Amazon. CloudWatch Para garantir que você possa acompanhar CloudWatch as métricas, escolha um nome que use somente caracteres ASCII.

2. (Opcional) No campo Entrada, insira os valores de entrada no formato JSON para executar o fluxo de trabalho.

   Se você optar por Executar uma demonstração, não precisará fornecer nenhuma entrada de execução.

   > **Note**
   >
   > Se o projeto de demonstração que você implementou contiver dados de entrada de execução pré-preenchidos, use essa entrada para executar a máquina de estado.

3. Selecione Iniciar execução.
4. O console do Step Functions direciona você para uma página em que o título é o ID da execução. Essa página é conhecida como página de Detalhes da execução. Nesta página, você pode revisar os resultados da execução à medida que a execução avança ou após a conclusão.

   Para revisar os resultados da execução, escolha estados individuais na Exibição em gráfico e, em seguida, escolha as guias individuais no painel Detalhes da etapa para visualizar os detalhes de cada estado, incluindo entrada, saída e definição, respectivamente. Para obter detalhes sobre as informações de execução que você pode visualizar na página Detalhes da execução, consulte Página de Detalhes da execução — Visão geral da interface.

Exemplo de código da máquina de estado

A máquina de estado neste projeto de amostra integra outra máquina de estado e AWS Lambda passa parâmetros diretamente para esses recursos.

Navegue por este exemplo de máquina de estado para ver como o Step Functions chama a ação da API `StartExecution` para a outra máquina de estado. Ele executa duas instâncias da outra máquina de estado em paralelo: uma usando o padrão Executar um trabalho (.sync) e outra usando o padrão Aguardar um retorno de chamada com um token de tarefa.

Para obter mais informações sobre como AWS Step Functions controlar outros AWS serviços, consulteUsando AWS Step Functions com outros serviços.

```
{
  "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 obter informações sobre como configurar o IAM ao usar Step Functions com outros AWS serviços, consulte[Políticas do IAM para serviços integrados](#).

Processar dados dinamicamente com um estado Mapa

Este projeto de exemplo demonstra o paralelismo dinâmico usando um estado [Mapa](#).

Neste projeto, Step Functions usa uma AWS Lambda função para extrair mensagens de uma fila do Amazon SQS e passar uma matriz JSON dessas mensagens para um estado. `Map` Para cada mensagem na fila, a máquina de estado grava a mensagem no DynamoDB, invoca a outra função do

Lambda para remover a mensagem do Amazon SQS e publica a mensagem no tópico do Amazon SNS.

Para obter mais informações sobre estados `Map` e integrações de serviços do Step Functions, consulte os tópicos a seguir.

- Mapa
- Usando AWS Step Functions com outros serviços

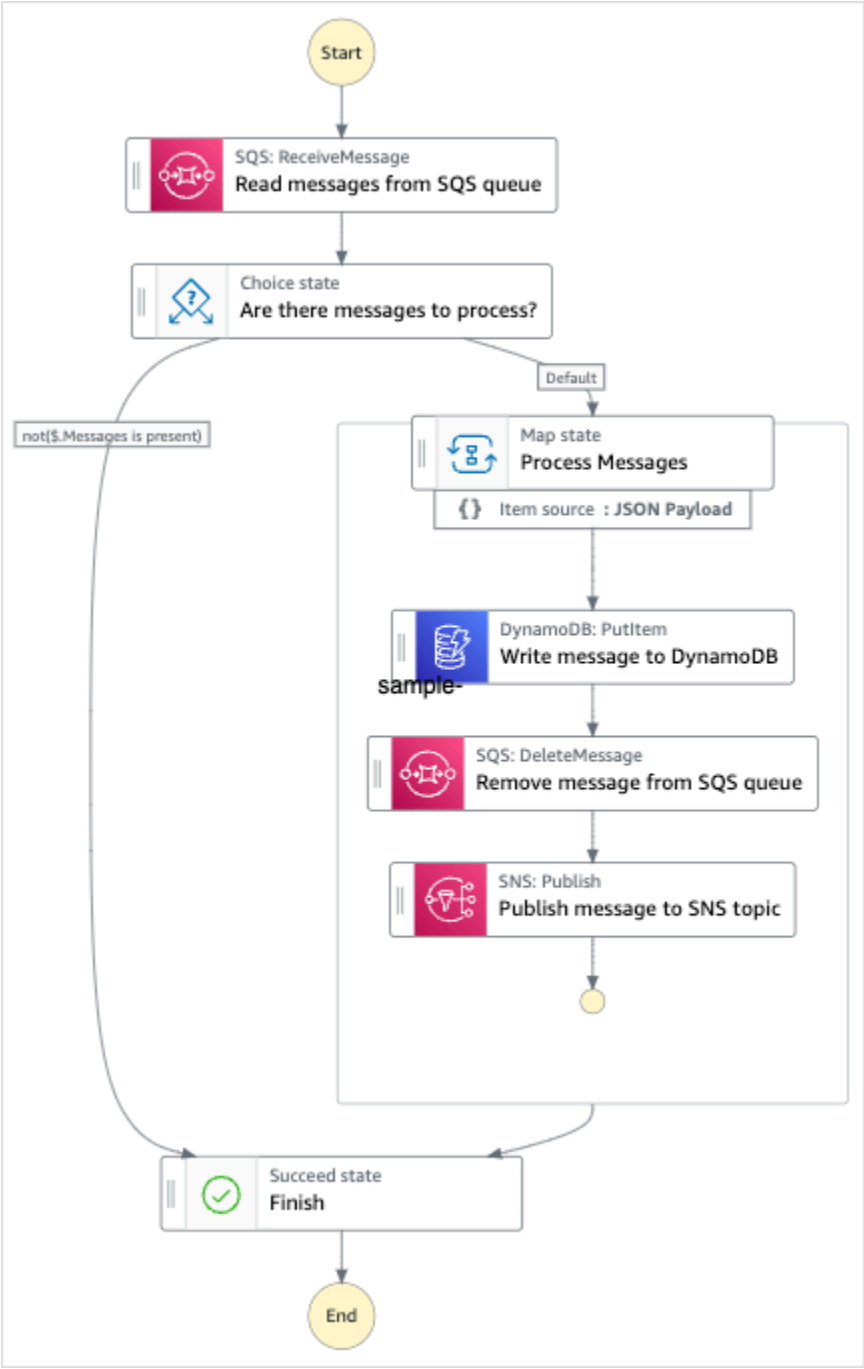
Etapa 1: Criar a máquina de estado e provisionar os recursos

1. Abra o console do Step Functions e clique em Criar máquina de estado.
2. Digite **`Dynamically process data with a Map state`** no campo de pesquisa e escolha Processar dados dinamicamente com um estado Mapa a partir dos resultados de pesquisa que são retornados.
3. Escolha Próximo para continuar.
4. Step Functions lista o Serviços da AWS usado no projeto de amostra que você selecionou. Também mostra um gráfico de fluxo de trabalho para o projeto de amostra. Implante esse projeto no seu Conta da AWS ou use-o como ponto de partida para criar seus próprios projetos. Com base em como você deseja prosseguir, escolha Executar uma demonstração ou Criar com base nela.

   Este projeto de exemplo implementa os recursos a seguir.

   - Uma fila do Amazon SQS onde o estado Mapa lê e remove mensagens de forma iterativa.
   - Uma tabela do DynamoDB onde o estado Mapa grava mensagens de forma iterativa.
   - Um tópico do Amazon SNS onde o Step Functions publica as mensagens que lê da fila do Amazon SQS.
   - Duas AWS Lambda funções
   - Uma máquina de AWS Step Functions estado
   - Funções relacionadas AWS Identity and Access Management (IAM)

   A imagem a seguir mostra o gráfico do fluxo de trabalho para o exemplo de projeto Processar dados dinamicamente com um estado Mapa:



5. Escolha Usar modelo para continuar com a seleção.
6. Execute um destes procedimentos:

   - Se você selecionou Criar com base nela, o Step Functions criará o protótipo do fluxo de trabalho para o projeto de amostra selecionado. O Step Functions não implanta os recursos listados na definição do fluxo de trabalho.

No Modo de design do Workflow Studio, arraste e solte os estados do Navegador de estados para continuar criando seu protótipo de fluxo de trabalho. Ou mude para o Modo de código que fornece um editor de código integrado semelhante ao VS Code para atualizar a definição Amazon States Language (ASL) de sua máquina de estado no console Step Functions. Para obter mais informações sobre o uso do Workflow Studio para criar suas máquinas de estado, consulte Como usar o Workflow Studio.

> **Important**
>
> Lembre-se de atualizar o espaço reservado do nome do recurso da Amazon (ARN) para os recursos usados no projeto de amostra antes de executar o fluxo de trabalho.

- Se você selecionou Executar uma demonstração, o Step Functions cria um projeto de amostra somente para leitura que usa um AWS CloudFormation modelo para implantar os AWS recursos listados nesse modelo no seu. Conta da AWS

  > **Tip**
  >
  > Para visualizar a definição da máquina de estado do projeto de amostra, escolha Código.

  Quando estiver pronto, escolha Implemente e execute para implantar o projeto de amostra e criar os recursos.

  Pode levar até 10 minutos para que esses recursos e as permissões relacionadas ao IAM sejam criados. Enquanto seus recursos estão sendo implantados, você pode abrir o link do CloudFormation Stack ID para ver quais recursos estão sendo provisionados.

  Depois que todos os recursos do projeto de exemplo forem criados, você poderá ver o novo projeto de exemplo listado na página Máquinas de estado.

  > **Important**
  >
  > Taxas padrão podem ser aplicadas a cada serviço usado no CloudFormation modelo.

Depois que os recursos do projeto de exemplo forem implantados, será necessário adicionar itens à fila do Amazon SQS e inscrever-se no tópico do Amazon SNS antes de iniciar uma execução da máquina de estado.

Etapa 2: Assinar o tópico do Amazon SNS

1. Abra o console do Amazon SNS.
2. Escolha Topics (Tópicos) e escolha o tópico que foi criado pelo projeto de exemplo de estado `Map`.

   O nome será semelhante a MapSampleProj-SNSTopic-1CQO4HQ3iR1KN.
3. Selecione Criar assinatura.

   A página Create subscription (Criar assinatura) é exibida, listando o Topic ARN (ARN do tópico) para o tópico.
4. Em Protocol (Protocolo), escolha Email (E-mail).
5. Em Endpoint, insira um endereço de e-mail para assinar o tópico.
6. Selecione Criar assinatura.

   > **Note**
   >
   > Você deve confirmar a assinatura no seu e-mail antes da ativação.

7. Abra o e-mail de Subscription Confirmation (Confirmação de assinatura) na conta relacionada e abra a URL Confirm subscription (Confirmar assinatura).

   A página Subscription confirmed! (Assinatura confirmada!) é exibida.

Etapa 3: Adicionar mensagens à fila do Amazon SQS

1. Abra o console do Amazon SQS.
2. Escolha a fila criada pelo projeto de exemplo de estado `Map`.

   O nome será semelhante a MapSampleProj-sqsQueue-1udic9vzdOrn7.
3. Escolha Send and receive messages (Enviar e receber mensagens).
4. Na janela Enviar e receber mensagens, insira uma mensagem e escolha Enviar mensagem.

5. Digitando outra mensagem e escolha Enviar mensagem. Continue inserindo mais mensagens até ter várias na fila do Amazon SQS.

Etapa 4: Executar a máquina de estado

> **Note**
>
> As filas no Amazon SNS são eventualmente consistentes. Para obter melhores resultados, aguarde alguns minutos entre o preenchimento da sua fila e a execução da sua máquina de estado.

1. Na página Máquinas de estado, escolha seu projeto de exemplo.
2. Na página do projeto de exemplo, escolha Iniciar execução.
3. Na caixa de diálogo Iniciar execução, faça o seguinte:

    1. (Opcional) Para identificar a execução, insira um nome para ela no campo Nome. Por padrão, o Step Functions cria automaticamente um nome de execução exclusivo.

        > **Note**
        >
        > Step Functions permite criar nomes para máquinas de estado, execuções e atividades, além de rótulos que contenham caracteres não ASCII. Esses nomes não ASCII não funcionam com a Amazon. CloudWatch Para garantir que você possa acompanhar CloudWatch as métricas, escolha um nome que use somente caracteres ASCII.

    2. (Opcional) No campo Entrada, insira os valores de entrada no formato JSON para executar o fluxo de trabalho.

        Se você optar por Executar uma demonstração, não precisará fornecer nenhuma entrada de execução.

        > **Note**
        >
        > Se o projeto de demonstração que você implementou contiver dados de entrada de execução pré-preenchidos, use essa entrada para executar a máquina de estado.

3. Selecione Iniciar execução.
4. O console do Step Functions direciona você para uma página em que o título é o ID da execução. Essa página é conhecida como página de Detalhes da execução. Nesta página, você pode revisar os resultados da execução à medida que a execução avança ou após a conclusão.

   Para revisar os resultados da execução, escolha estados individuais na Exibição em gráfico e, em seguida, escolha as guias individuais no painel Detalhes da etapa para visualizar os detalhes de cada estado, incluindo entrada, saída e definição, respectivamente. Para obter detalhes sobre as informações de execução que você pode visualizar na página Detalhes da execução, consulte Página de Detalhes da execução — Visão geral da interface.

Exemplo de código da máquina de estado

A máquina de estado deste exemplo de projeto se integra ao Amazon SQS, ao Amazon SNS e ao Lambda passando parâmetros diretamente para esses recursos.

Navegue por este exemplo de máquina de estado para ver como o Step Functions controla o Lambda, o DynamoDB e o Amazon SNS conectando-se ao nome do recurso da Amazon (ARN) no campo `Resource` e passando `Parameters` para a API de serviço.

Para obter mais informações sobre como AWS Step Functions controlar outros AWS serviços, consulteUsando AWS Step Functions com outros serviços.

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

Exemplo do IAM

Esse exemplo de política AWS Identity and Access Management (IAM) gerado pelo projeto de amostra inclui o menor privilégio necessário para executar a máquina de estado e os recursos relacionados. Recomendamos que você inclua apenas as permissões que forem necessárias em suas políticas do 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 obter informações sobre como configurar o IAM ao usar Step Functions com outros AWS serviços, consulte Políticas do IAM para serviços integrados.

Processar um arquivo CSV com o Mapa distribuído

Este projeto de exemplo demonstra como você pode usar o estado Mapa distribuído para iterar mais de 10.000 linhas de um arquivo CSV que é gerado usando uma função do Lambda. O arquivo CSV contém informações de envio dos pedidos do cliente e é armazenado em um bucket do Amazon S3. O Mapa distribuído itera em um lote de 10 linhas no arquivo CSV para análise de dados.

O Mapa distribuído contém uma função do Lambda para detectar qualquer pedido atrasado. O Mapa distribuído também contém um Mapa Inline para processar os pedidos atrasados em um lote e

retornar esses pedidos atrasados em uma matriz. Para cada pedido atrasado, o Mapa Inline envia uma mensagem para uma fila do Amazon SQS. Por fim, esse projeto de exemplo armazena os resultados do Execução de mapa em outro bucket do Amazon S3 em seu Conta da AWS.

Com o Mapa distribuído, você pode realizar até 10.000 execuções paralelas de fluxo de trabalho secundário por vez. Neste projeto de exemplo, a simultaneidade máxima do Mapa distribuído é definida em 1000, o que a limita a 1000 execuções paralelas de fluxo de trabalho secundário.

Esse projeto de amostra cria a máquina de estado, os AWS recursos de suporte e configura as permissões relacionadas do IAM. Explore este projeto de exemplo para saber como usar o Mapa distribuído para orquestrar workloads paralelas em grande escala ou usá-lo como ponto de partida para seus próprios projetos.

AWS CloudFormation modelo e recursos adicionais

Você usa um CloudFormation modelo para implantar esse projeto de amostra. Esse modelo cria os seguintes recursos em seu Conta da AWS:

- Máquina de estado do Step Functions.
- Função de execução para a máquina de estado. Essa função concede as permissões que sua máquina de estado precisa para acessar outros Serviços da AWS recursos, como a ação Invoke da função Lambda.
- Uma função do Lambda chamada `CSVGeneratorFunction` gera um arquivo CSV que contém os detalhes do pedido do cliente.
- Função de execução para a função do Lambda do gerador CSV. Essa função concede à função permissão para acessar outras Serviços da AWS.
- Um bucket de entrada do Amazon S3 para armazenar o arquivo CSV gerado.
- Uma função do Lambda de detecção de pedidos atrasados que analisa os dados do arquivo CSV e detecta qualquer pedido atrasado.
- Função de execução para a função do Lambda do pedido atrasado. Essa função concede à função permissão para acessar outras Serviços da AWS.
- Um bucket de saída do Amazon S3 para armazenar os resultados da análise dos pedidos do cliente.
- Uma fila do Amazon SQS para a qual o Step Functions envia mensagens para cada pedido atrasado. Essas mensagens contêm os IDs dos clientes e de seus pedidos.
- Um grupo de CloudWatch registros que armazena informações relacionadas ao histórico de execução da máquina de estado.

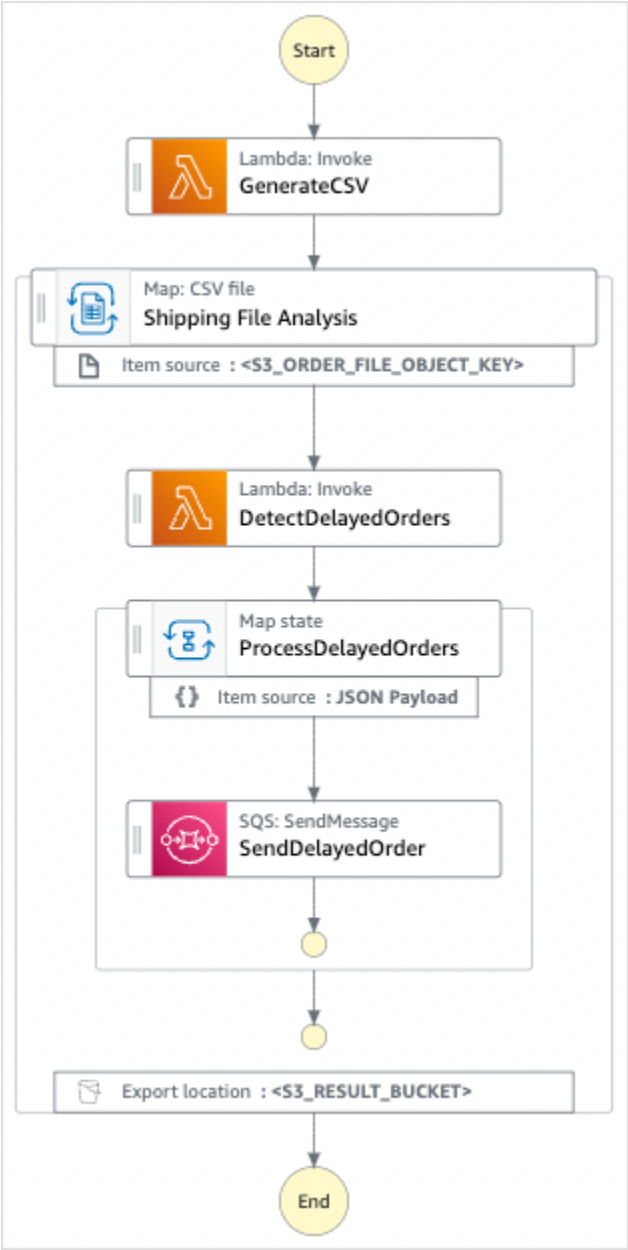
> **Important**
>
> Aplicam-se taxas padrão para cada serviço.

Etapa 1: Criar a máquina de estado e provisionar os recursos

1. Abra o console do Step Functions e clique em Criar máquina de estado.
2. Digite **`Distributed Map to process a CSV file in S3`** no campo de pesquisa e escolha Mapa distribuído para processar um arquivo CSV no S3 a partir dos resultados da pesquisa que são retornados.
3. Escolha Próximo para continuar.
4. Step Functions lista o Serviços da AWS usado no projeto de amostra que você selecionou. Também mostra um gráfico de fluxo de trabalho para o projeto de amostra. Implante esse projeto no seu Conta da AWS ou use-o como ponto de partida para criar seus próprios projetos. Com base em como você deseja prosseguir, escolha Executar uma demonstração ou Criar com base nela.

   Para obter informações sobre os recursos que serão criados para esse projeto de exemplo, consulte AWS CloudFormation modelo e recursos adicionais.

   A imagem a seguir mostra o gráfico do fluxo de trabalho para o Mapa distribuído para processar um arquivo CSV no projeto de exemplo do S3:



5. Escolha Usar modelo para continuar com a seleção.
6. Execute um destes procedimentos:

   - Se você selecionou Criar com base nela, o Step Functions criará o protótipo do fluxo de trabalho para o projeto de amostra selecionado. O Step Functions não implanta os recursos listados na definição do fluxo de trabalho.

     No Modo de design do Workflow Studio, arraste e solte os estados do Navegador de estados para continuar criando seu protótipo de fluxo de trabalho. Ou mude para o Modo de código que fornece um editor de código integrado semelhante ao VS Code para atualizar a definição

Amazon States Language (ASL) de sua máquina de estado no console Step Functions. Para obter mais informações sobre o uso do Workflow Studio para criar suas máquinas de estado, consulte Como usar o Workflow Studio.

> **Important**
>
> Lembre-se de atualizar o espaço reservado do nome do recurso da Amazon (ARN) para os recursos usados no projeto de amostra antes de executar o fluxo de trabalho.

- Se você selecionou Executar uma demonstração, o Step Functions cria um projeto de amostra somente para leitura que usa um AWS CloudFormation modelo para implantar os AWS recursos listados nesse modelo no seu. Conta da AWS

  > **Tip**
  >
  > Para visualizar a definição da máquina de estado do projeto de amostra, escolha Código.

  Quando estiver pronto, escolha Implemente e execute para implantar o projeto de amostra e criar os recursos.

  Pode levar até 10 minutos para que esses recursos e as permissões relacionadas ao IAM sejam criados. Enquanto seus recursos estão sendo implantados, você pode abrir o link do CloudFormation Stack ID para ver quais recursos estão sendo provisionados.

  Depois que todos os recursos do projeto de exemplo forem criados, você poderá ver o novo projeto de exemplo listado na página Máquinas de estado.

  > **Important**
  >
  > Taxas padrão podem ser aplicadas a cada serviço usado no CloudFormation modelo.

Etapa 2: Executar a máquina de estado

Depois que todos os recursos forem aprovisionados e implementados, você poderá executar a máquina de estado.

1. Na página Máquinas de estado, escolha seu projeto de exemplo.
2. Na página do projeto de exemplo, escolha Iniciar execução.
3. Na caixa de diálogo Iniciar execução, faça o seguinte:

   a. (Opcional) Insira os valores de entrada no formato JSON para executar seu projeto de exemplo.

      Se você optar por Executar uma demonstração, não precisará fornecer nenhuma entrada de execução.

      > **Note**
      >
      > Se o projeto de demonstração que você implementou contiver dados de entrada de execução pré-preenchidos, use essa entrada para executar a máquina de estado.

   b. Selecione Iniciar execução.

   c. (Opcional) O console Step Functions direciona você para uma página intitulada com seu ID de execução. Essa página é conhecida como página de Detalhes da execução. Nesta página, você pode revisar os resultados da execução à medida que a execução avança ou após a conclusão.

      Depois que a execução for concluída, escolha estados individuais na Exibição em gráfico e, em seguida, escolha as guias individuais no painel de Detalhes da etapa para visualizar os detalhes de cada estado, incluindo entrada, saída e definição, respectivamente.

      - Para obter detalhes sobre as informações de execução que você pode visualizar na página Detalhes da execução, consulte Página de Detalhes da execução — Visão geral da interface.
      - Para obter mais informações sobre a visualização da execução do estado Mapa distribuído no console, consulte Examinando o Map Run.

   d. (Opcional) Revise os resultados da execução exportados para o bucket do Amazon S3. Esses resultados incluem dados, como entrada e saída de execução, ARN e status de execução. Para ter mais informações, consulte ResultWriter.

Processe dados em um bucket do Amazon S3 com o Mapa distribuído

Este projeto de exemplo demonstra como você pode usar o estado Mapa distribuído para processar dados em grande escala, por exemplo, analisar dados meteorológicos históricos e identificar a estação meteorológica que tem a temperatura média mais alta do planeta a cada mês. Os dados meteorológicos são registrados em mais de 12.000 arquivos CSV, que, por sua vez, são armazenados em um bucket do Amazon S3.

Esse projeto de exemplo inclui dois estados de Mapa distribuído, chamados de NOA Data da cópia distribuída do S3 e ProcessNOAAData. A cópia distribuída do S3 NOA Data itera sobre os arquivos CSV em um bucket público do Amazon S3 chamado noaa-gsod-pdse os copia para um bucket do Amazon S3 em seu. Conta da AWS ProcessNOAAData itera sobre os arquivos copiados e inclui uma função do Lambda que executa a análise de temperatura.

O projeto de amostra primeiro verifica o conteúdo do bucket do Amazon S3 com uma chamada para a ação da API ListObjectsV2. Com base no número de chaves retornadas em resposta a essa chamada, o projeto de exemplo toma uma das seguintes decisões:

- Se a contagem de chaves for maior ou igual a 1, o projeto fará a transição para o estado ProcessNOAAData. Esse estado de Mapa Distribuído inclui uma Lambda função chamada TemperatureFunctionque encontra a estação meteorológica que teve a temperatura média mais alta a cada mês. Essa função retorna um dicionário com o `year-month` como chave e um dicionário que contém informações sobre a estação meteorológica como valor.
- Se a contagem de chaves retornadas não exceder 1, o estado de dados NOA da cópia distribuída do S3 listará todos os objetos do bucket público noaa-gsod-pdse copiará iterativamente os objetos individuais para outro bucket em sua conta em lotes de 100. Um Mapa inline executa a cópia iterativa dos objetos.

  Depois que todos os objetos são copiados, o projeto passa para o estado ProcessNOAAData para processar os dados meteorológicos.

O projeto de amostra finalmente faz a transição para uma Lambda função redutora que realiza uma agregação final dos resultados retornados pela TemperatureFunctionfunção e grava os resultados em uma tabela. Amazon DynamoDB

Com o Mapa distribuído, você pode realizar até 10.000 execuções paralelas de fluxo de trabalho secundário por vez. Neste projeto de exemplo, a simultaneidade máxima do Mapa distribuído do ProcessNOAAData é definida em 3.000, o que o limita a 3.000 execuções paralelas de fluxo de trabalho secundário.

Esse projeto de amostra cria a máquina de estado, os AWS recursos de suporte e configura as permissões relacionadas do IAM. Explore este projeto de exemplo para saber como usar o Mapa distribuído para orquestrar workloads paralelas em grande escala ou usá-lo como ponto de partida para seus próprios projetos.

> **Important**
>
> Esse projeto de exemplo está disponível somente na região Leste dos EUA (Norte da Virgínia).

AWS CloudFormation modelo e recursos adicionais

Você usa um CloudFormation modelo para implantar esse projeto de amostra. Esse modelo cria os seguintes recursos em seu Conta da AWS:

- Máquina de estado do Step Functions.
- Função de execução para a máquina de estado. Essa função concede as permissões que sua máquina de estado precisa para acessar outros Serviços da AWS recursos, como a ação Invoke da função Lambda.
- Um bucket do Amazon S3 nomeado `NOAADataBucket`. Esse bucket contém os arquivos CSV com dados meteorológicos.
- Uma função do Lambda chamada `ReducerFunction` que executa uma agregação final dos dados meteorológicos e grava os resultados em uma tabela do Amazon DynamoDB.
- Função de execução para a função redutora do Lambda. Essa função concede à função permissão para acessar outras Serviços da AWS.
- Um bucket de saída do Amazon S3 nomeado `ResultsBucket` para armazenar os resultados da análise meteorológica.
- Uma tabela do DynamoDB chamada `ResultsDynamoDBTable` que contém os resultados retornados pelo `ReducerFunction`.
- Uma função do Lambda chamada `TemperatureFunction` que encontra a temperatura média mensal mais alta.

- Função de execução para a função do Lambda. Essa função concede à função permissão para acessar outras Serviços da AWS.
- Um grupo de CloudWatch registros que armazena informações relacionadas ao histórico de execução da máquina de estado.

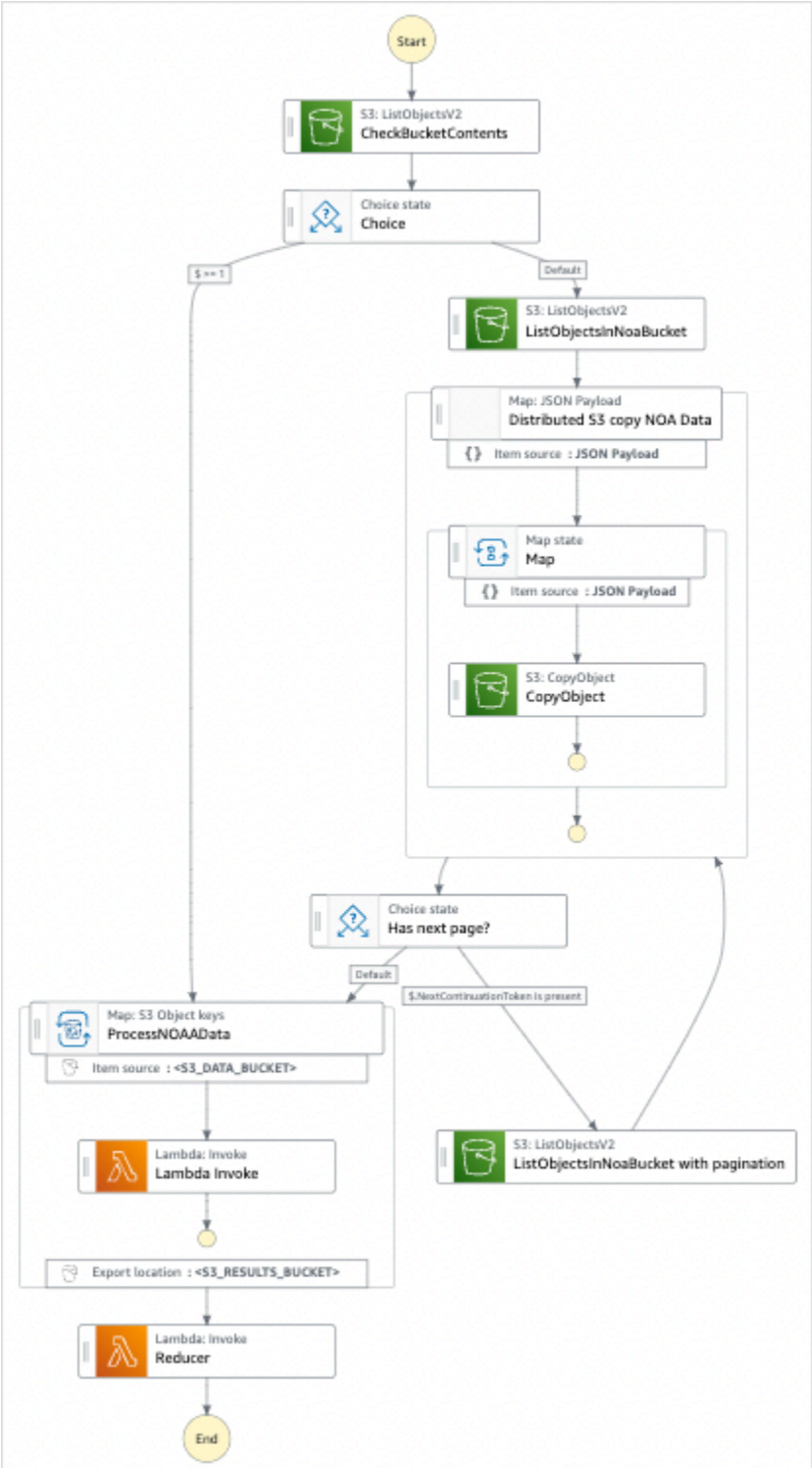
> **Important**
>
> Aplicam-se taxas padrão para cada serviço.

Etapa 1: Criar a máquina de estado e provisionar os recursos

1. Abra o console do Step Functions e clique em Criar máquina de estado.
2. Digite **`Distributed Map to process files in S3`** no campo de pesquisa e escolha Mapa distribuído para processar arquivos no S3 a partir dos resultados da pesquisa que são retornados.
3. Escolha Próximo para continuar.
4. Step Functions lista o Serviços da AWS usado no projeto de amostra que você selecionou. Também mostra um gráfico de fluxo de trabalho para o projeto de amostra. Implante esse projeto no seu Conta da AWS ou use-o como ponto de partida para criar seus próprios projetos. Com base em como você deseja prosseguir, escolha Executar uma demonstração ou Criar com base nela.

   Para obter informações sobre os recursos que serão criados para esse projeto de exemplo, consulte AWS CloudFormation modelo e recursos adicionais.

   A imagem a seguir mostra o gráfico do fluxo de trabalho do Mapa distribuído para processar arquivos no projeto de exemplo do S3:



5. Escolha Usar modelo para continuar com a seleção.
6. Execute um destes procedimentos:

- Se você selecionou Criar com base nela, o Step Functions criará o protótipo do fluxo de trabalho para o projeto de amostra selecionado. O Step Functions não implanta os recursos listados na definição do fluxo de trabalho.

  No Modo de design do Workflow Studio, arraste e solte os estados do Navegador de estados para continuar criando seu protótipo de fluxo de trabalho. Ou mude para o Modo de código que fornece um editor de código integrado semelhante ao VS Code para atualizar a definição Amazon States Language (ASL) de sua máquina de estado no console Step Functions. Para obter mais informações sobre o uso do Workflow Studio para criar suas máquinas de estado, consulte Como usar o Workflow Studio.

  > **Important**
  >
  > Lembre-se de atualizar o espaço reservado do nome do recurso da Amazon (ARN) para os recursos usados no projeto de amostra antes de executar o fluxo de trabalho.

- Se você selecionou Executar uma demonstração, o Step Functions cria um projeto de amostra somente para leitura que usa um AWS CloudFormation modelo para implantar os AWS recursos listados nesse modelo no seu. Conta da AWS

  > **Tip**
  >
  > Para visualizar a definição da máquina de estado do projeto de amostra, escolha Código.

  Quando estiver pronto, escolha Implemente e execute para implantar o projeto de amostra e criar os recursos.

  Pode levar até 10 minutos para que esses recursos e as permissões relacionadas ao IAM sejam criados. Enquanto seus recursos estão sendo implantados, você pode abrir o link do CloudFormation Stack ID para ver quais recursos estão sendo provisionados.

  Depois que todos os recursos do projeto de exemplo forem criados, você poderá ver o novo projeto de exemplo listado na página Máquinas de estado.

> **Important**
>
> Taxas padrão podem ser aplicadas a cada serviço usado no CloudFormation modelo.

Etapa 2: Executar a máquina de estado

Depois que todos os recursos forem aprovisionados e implementados, você poderá executar a máquina de estado.

1. Na página Máquinas de estado, escolha seu projeto de exemplo.
2. Na página do projeto de exemplo, escolha Iniciar execução.
3. Na caixa de diálogo Iniciar execução, faça o seguinte:

   a. (Opcional) Insira os valores de entrada no formato JSON para executar seu projeto de exemplo.

      Se você optar por Executar uma demonstração, não precisará fornecer nenhuma entrada de execução.

      > **Note**
      >
      > Se o projeto de demonstração que você implementou contiver dados de entrada de execução pré-preenchidos, use essa entrada para executar a máquina de estado.

   b. Selecione Iniciar execução.

   c. (Opcional) O console Step Functions direciona você para uma página intitulada com seu ID de execução. Essa página é conhecida como página de Detalhes da execução. Nesta página, você pode revisar os resultados da execução à medida que a execução avança ou após a conclusão.

      Depois que a execução for concluída, escolha estados individuais na Exibição em gráfico e, em seguida, escolha as guias individuais no painel de Detalhes da etapa para visualizar os detalhes de cada estado, incluindo entrada, saída e definição, respectivamente.

- Para obter detalhes sobre as informações de execução que você pode visualizar na página Detalhes da execução, consulte Página de Detalhes da execução — Visão geral da interface.
- Para obter mais informações sobre a visualização da execução do estado Mapa distribuído no console, consulte Examinando o Map Run.

d. (Opcional) Revise os resultados da execução exportados para o bucket do Amazon S3. Esses resultados incluem dados, como entrada e saída de execução, ARN e status de execução. Para ter mais informações, consulte ResultWriter.

Treinar um modelo de machine learning.

Este exemplo de projeto demonstra como usar SageMaker e AWS Step Functions treinar um modelo de aprendizado de máquina e como transformar em lote um conjunto de dados de teste.

Nesse projeto, o Step Functions usa uma função do Lambda para propagar um bucket do Amazon S3 com um conjunto de dados de teste. Em seguida, ele treina um modelo de aprendizado de máquina e executa uma transformação em lote, usando a integração SageMaker de serviços.

Para obter mais informações sobre integrações de serviços SageMaker e Step Functions, consulte o seguinte:

- Usando AWS Step Functions com outros serviços
- Gerencie SageMaker com Step Functions

> **Note**
>
> Este projeto de exemplo pode incorrer em cobranças.
> Para novos AWS usuários, um nível de uso gratuito está disponível. Neste nível, os serviços são gratuitos abaixo de um determinado nível de uso. Para obter mais informações sobre AWS custos e o nível gratuito, consulte SageMaker Preços.

Etapa 1: Criar a máquina de estado e provisionar os recursos

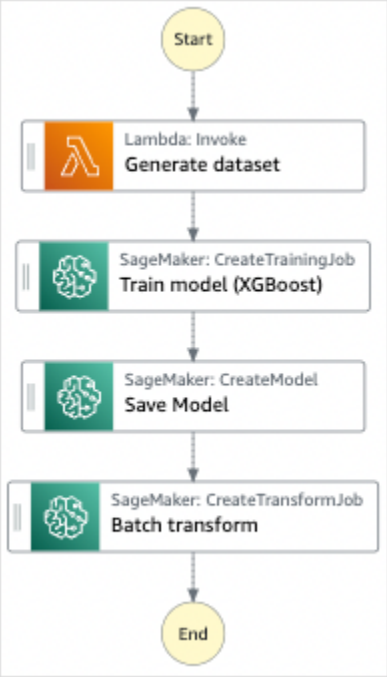
1. Abra o console do Step Functions e clique em Criar máquina de estado.

2. Digite **`Train a machine learning model`** na caixa de pesquisa e escolha Treinar um modelo de machine learning nos resultados da pesquisa que são retornados.
3. Escolha Próximo para continuar.
4. Step Functions lista o Serviços da AWS usado no projeto de amostra que você selecionou. Também mostra um gráfico de fluxo de trabalho para o projeto de amostra. Implante esse projeto no seu Conta da AWS ou use-o como ponto de partida para criar seus próprios projetos. Com base em como você deseja prosseguir, escolha Executar uma demonstração ou Criar com base nela.

   Este projeto de exemplo implementa os recursos a seguir.

   - Uma AWS Lambda função
   - Um bucket do Amazon Simple Storage Service (Amazon S3)
   - Uma máquina de AWS Step Functions estado
   - Funções relacionadas AWS Identity and Access Management (IAM)

   A imagem a seguir mostra o gráfico do fluxo de trabalho para o exemplo de projeto Treinar um modelo de machine learning:



5. Escolha Usar modelo para continuar com a seleção.
6. Execute um destes procedimentos:

- Se você selecionou Criar com base nela, o Step Functions criará o protótipo do fluxo de trabalho para o projeto de amostra selecionado. O Step Functions não implanta os recursos listados na definição do fluxo de trabalho.

  No Modo de design do Workflow Studio, arraste e solte os estados do Navegador de estados para continuar criando seu protótipo de fluxo de trabalho. Ou mude para o Modo de código que fornece um editor de código integrado semelhante ao VS Code para atualizar a definição Amazon States Language (ASL) de sua máquina de estado no console Step Functions. Para obter mais informações sobre o uso do Workflow Studio para criar suas máquinas de estado, consulte Como usar o Workflow Studio.

  > **Important**
  >
  > Lembre-se de atualizar o espaço reservado do nome do recurso da Amazon (ARN) para os recursos usados no projeto de amostra antes de executar o fluxo de trabalho.

- Se você selecionou Executar uma demonstração, o Step Functions cria um projeto de amostra somente para leitura que usa um AWS CloudFormation modelo para implantar os AWS recursos listados nesse modelo no seu. Conta da AWS

  > **Tip**
  >
  > Para visualizar a definição da máquina de estado do projeto de amostra, escolha Código.

  Quando estiver pronto, escolha Implemente e execute para implantar o projeto de amostra e criar os recursos.

  Pode levar até 10 minutos para que esses recursos e as permissões relacionadas ao IAM sejam criados. Enquanto seus recursos estão sendo implantados, você pode abrir o link do CloudFormation Stack ID para ver quais recursos estão sendo provisionados.

  Depois que todos os recursos do projeto de exemplo forem criados, você poderá ver o novo projeto de exemplo listado na página Máquinas de estado.

> **Important**
>
> Taxas padrão podem ser aplicadas a cada serviço usado no CloudFormation modelo.

Etapa 2: Executar a máquina de estado

1. Na página Máquinas de estado, escolha seu projeto de exemplo.
2. Na página do projeto de exemplo, escolha Iniciar execução.
3. Na caixa de diálogo Iniciar execução, faça o seguinte:

   1. (Opcional) Para identificar a execução, insira um nome para ela no campo Nome. Por padrão, o Step Functions cria automaticamente um nome de execução exclusivo.

      > **Note**
      >
      > Step Functions permite criar nomes para máquinas de estado, execuções e atividades, além de rótulos que contenham caracteres não ASCII. Esses nomes não ASCII não funcionam com a Amazon. CloudWatch Para garantir que você possa acompanhar CloudWatch as métricas, escolha um nome que use somente caracteres ASCII.

   2. (Opcional) No campo Entrada, insira os valores de entrada no formato JSON para executar o fluxo de trabalho.

      Se você optar por Executar uma demonstração, não precisará fornecer nenhuma entrada de execução.

      > **Note**
      >
      > Se o projeto de demonstração que você implementou contiver dados de entrada de execução pré-preenchidos, use essa entrada para executar a máquina de estado.

   3. Selecione Iniciar execução.
   4. O console do Step Functions direciona você para uma página em que o título é o ID da execução. Essa página é conhecida como página de Detalhes da execução. Nesta página,

você pode revisar os resultados da execução à medida que a execução avança ou após a conclusão.

Para revisar os resultados da execução, escolha estados individuais na Exibição em gráfico e, em seguida, escolha as guias individuais no painel Detalhes da etapa para visualizar os detalhes de cada estado, incluindo entrada, saída e definição, respectivamente. Para obter detalhes sobre as informações de execução que você pode visualizar na página Detalhes da execução, consulte Página de Detalhes da execução — Visão geral da interface.

Exemplo de código da máquina de estado

A máquina de estado neste projeto de amostra se integra com SageMaker e AWS Lambda passando parâmetros diretamente para esses recursos e usa um bucket do Amazon S3 para a fonte e a saída dos dados de treinamento.

Navegue por este exemplo de máquina de estado para ver como o Step Functions controla o Lambda e. SageMaker

Para obter mais informações sobre como AWS Step Functions controlar outros AWS serviços, consulteUsando AWS Step Functions com outros serviços.

```
{
  "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 obter informações sobre como configurar o IAM ao usar Step Functions com outros AWS serviços, consulte[Políticas do IAM para serviços integrados](#).

Exemplo do IAM

Essas políticas de exemplo AWS Identity and Access Management (IAM) geradas pelo projeto de amostra incluem o menor privilégio necessário para executar a máquina de estado e os recursos

relacionados. Recomendamos que você inclua apenas as permissões que forem necessárias em suas políticas do 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"
        }
    ]
}
```

A política a seguir permite que a função do Lambda propague o bucket do Amazon S3 com dados de exemplo.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::trainandbatchtransform-s3bucket-1jn1le6gadwfz/*",
            "Effect": "Allow"
        }
    ]
}
```

Para obter informações sobre como configurar o IAM ao usar Step Functions com outros AWS serviços, consulte Políticas do IAM para serviços integrados.

Ajustar um modelo de machine learning

Este exemplo de projeto demonstra como SageMaker ajustar os hiperparâmetros de um modelo de aprendizado de máquina e transformar em lote um conjunto de dados de teste.

Nesse projeto, o Step Functions usa uma função do Lambda para propagar um bucket do Amazon S3 com um conjunto de dados de teste. Em seguida, ele cria um trabalho de ajuste de hiperparâmetros usando a integração SageMaker de serviços. Em seguida, ele usa uma função Lambda para extrair o caminho dos dados, salva o modelo de ajuste, extrai o nome do modelo e, em seguida, executa um trabalho de transformação em lote para realizar a inferência. SageMaker

Para obter mais informações sobre integrações de serviços SageMaker e Step Functions, consulte o seguinte:

- Usando AWS Step Functions com outros serviços
- Gerencie SageMaker com Step Functions

> **Note**
>
> Este projeto de exemplo pode incorrer em cobranças.
> Para novos AWS usuários, um nível de uso gratuito está disponível. Neste nível, os serviços são gratuitos abaixo de um determinado nível de uso. Para obter mais informações sobre AWS custos e o nível gratuito, consulte SageMakerPreços.

Etapa 1: Criar a máquina de estado e provisionar os recursos

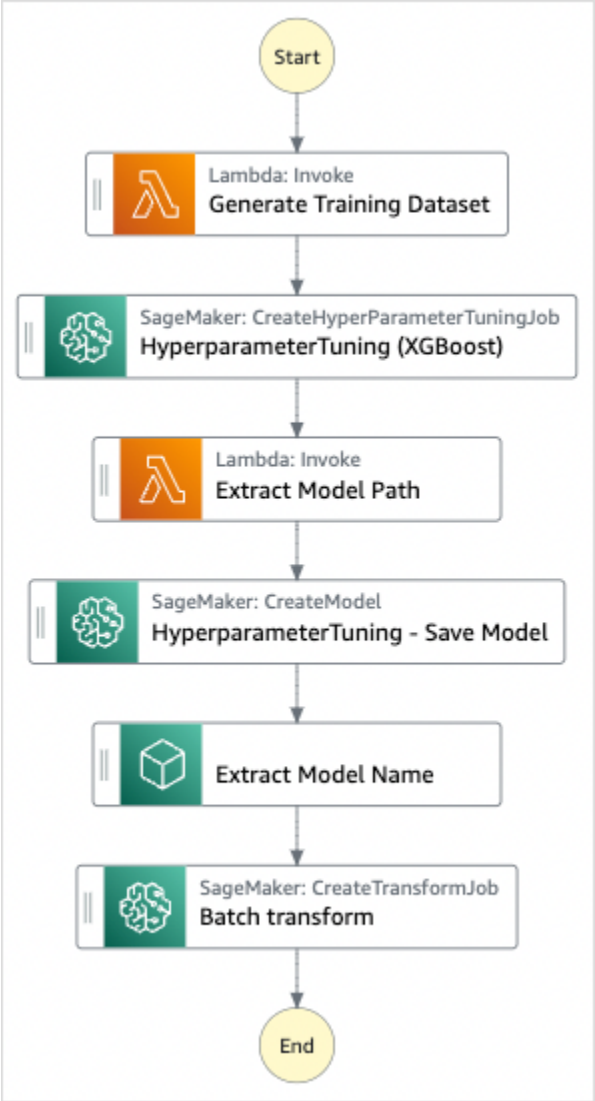
1. Abra o console do Step Functions e clique em Criar máquina de estado.
2. Digite **`Tune a machine learning model`** na caixa de pesquisa e escolha Ajustar um modelo de machine learning nos resultados da pesquisa que são retornados.
3. Escolha Próximo para continuar.
4. Step Functions lista o Serviços da AWS usado no projeto de amostra que você selecionou. Também mostra um gráfico de fluxo de trabalho para o projeto de amostra. Implante esse projeto no seu Conta da AWS ou use-o como ponto de partida para criar seus próprios projetos.

Com base em como você deseja prosseguir, escolha Executar uma demonstração ou Criar com base nela.

Este projeto de exemplo implementa os recursos a seguir.

- Três AWS Lambda funções
- Um bucket do Amazon Simple Storage Service (Amazon S3)
- Uma máquina de AWS Step Functions estado
- Funções relacionadas AWS Identity and Access Management (IAM)

A imagem a seguir mostra o gráfico do fluxo de trabalho para o exemplo de projeto Ajustar um modelo de machine learning:



5. Escolha Usar modelo para continuar com a seleção.

6. Execute um destes procedimentos:

   - Se você selecionou Criar com base nela, o Step Functions criará o protótipo do fluxo de trabalho para o projeto de amostra selecionado. O Step Functions não implanta os recursos listados na definição do fluxo de trabalho.

     No Modo de design do Workflow Studio, arraste e solte os estados do Navegador de estados para continuar criando seu protótipo de fluxo de trabalho. Ou mude para o Modo de código que fornece um editor de código integrado semelhante ao VS Code para atualizar a definição Amazon States Language (ASL) de sua máquina de estado no console Step Functions. Para obter mais informações sobre o uso do Workflow Studio para criar suas máquinas de estado, consulte Como usar o Workflow Studio.

     > **Important**
     >
     > Lembre-se de atualizar o espaço reservado do nome do recurso da Amazon (ARN) para os recursos usados no projeto de amostra antes de executar o fluxo de trabalho.

   - Se você selecionou Executar uma demonstração, o Step Functions cria um projeto de amostra somente para leitura que usa um AWS CloudFormation modelo para implantar os AWS recursos listados nesse modelo no seu. Conta da AWS

     > **Tip**
     >
     > Para visualizar a definição da máquina de estado do projeto de amostra, escolha Código.

     Quando estiver pronto, escolha Implemente e execute para implantar o projeto de amostra e criar os recursos.

     Pode levar até 10 minutos para que esses recursos e as permissões relacionadas ao IAM sejam criados. Enquanto seus recursos estão sendo implantados, você pode abrir o link do CloudFormation Stack ID para ver quais recursos estão sendo provisionados.

     Depois que todos os recursos do projeto de exemplo forem criados, você poderá ver o novo projeto de exemplo listado na página Máquinas de estado.

> **Important**
>
> Taxas padrão podem ser aplicadas a cada serviço usado no CloudFormation modelo.

Etapa 2: Executar a máquina de estado

1. Na página Máquinas de estado, escolha seu projeto de exemplo.
2. Na página do projeto de exemplo, escolha Iniciar execução.
3. Na caixa de diálogo Iniciar execução, faça o seguinte:

   1. (Opcional) Para identificar a execução, insira um nome para ela no campo Nome. Por padrão, o Step Functions cria automaticamente um nome de execução exclusivo.

      > **Note**
      >
      > Step Functions permite criar nomes para máquinas de estado, execuções e atividades, além de rótulos que contenham caracteres não ASCII. Esses nomes não ASCII não funcionam com a Amazon. CloudWatch Para garantir que você possa acompanhar CloudWatch as métricas, escolha um nome que use somente caracteres ASCII.

   2. (Opcional) No campo Entrada, insira os valores de entrada no formato JSON para executar o fluxo de trabalho.

      Se você optar por Executar uma demonstração, não precisará fornecer nenhuma entrada de execução.

      > **Note**
      >
      > Se o projeto de demonstração que você implementou contiver dados de entrada de execução pré-preenchidos, use essa entrada para executar a máquina de estado.

   3. Selecione Iniciar execução.
   4. O console do Step Functions direciona você para uma página em que o título é o ID da execução. Essa página é conhecida como página de Detalhes da execução. Nesta página,

você pode revisar os resultados da execução à medida que a execução avança ou após a conclusão.

Para revisar os resultados da execução, escolha estados individuais na Exibição em gráfico e, em seguida, escolha as guias individuais no painel Detalhes da etapa para visualizar os detalhes de cada estado, incluindo entrada, saída e definição, respectivamente. Para obter detalhes sobre as informações de execução que você pode visualizar na página Detalhes da execução, consulte Página de Detalhes da execução — Visão geral da interface.

Exemplo de código da máquina de estado

A máquina de estado neste projeto de amostra se integra com SageMaker e AWS Lambda passando parâmetros diretamente para esses recursos e usa um bucket do Amazon S3 para a fonte e a saída dos dados de treinamento.

Navegue por este exemplo de máquina de estado para ver como o Step Functions controla o Lambda e. SageMaker

Para obter mais informações sobre como AWS Step Functions controlar outros AWS serviços, consulteUsando AWS Step Functions com outros serviços.

```
{
    "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 obter informações sobre como configurar o IAM ao usar o Step Functions com outros serviços da AWS , consulte Políticas do IAM para serviços integrados.

Exemplos do IAM

Essas políticas de exemplo AWS Identity and Access Management (IAM) geradas pelo projeto de amostra incluem o menor privilégio necessário para executar a máquina de estado e os recursos relacionados. Recomendamos que você inclua apenas as permissões que forem necessárias em suas políticas do IAM.

A política do IAM a seguir é anexada à máquina de estado e permite que a execução da máquina de estado acesse os recursos necessários SageMaker, do Lambda e do 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"
        }
    ]
}
```

A política do IAM a seguir é referenciada nos campos `TrainingJobDefinition` e `HyperparameterTuning` do 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"
        }
    ]
}
```

A política do IAM a seguir permite que a função do Lambda propague o bucket do Amazon S3 com dados de exemplo.

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

```
        {
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::stepfunctionssample-sagemak-
bucketformodelanddata-80fblmdlcs9f/*",
            "Effect": "Allow"
        }
    ]
}
```

Para obter informações sobre como configurar o IAM ao usar o Step Functions com outros serviços da AWS , consulte Políticas do IAM para serviços integrados.

Processar mensagens de alto volume do Amazon SQS (fluxos de trabalho expressos)

Este exemplo de projeto demonstra como usar um fluxo de trabalho AWS Step Functions expresso para processar mensagens ou dados de uma fonte de eventos de alto volume, como o Amazon Simple Queue Service (Amazon SQS). Como os fluxos de trabalho expressos podem ser iniciados em uma taxa muito alta, eles são ideais para processamento de eventos de alto volume ou cargas de trabalho de dados em streaming.

Aqui estão dois métodos comumente usados para executar sua máquina de estado de uma fonte de evento:

- Configure uma regra do Amazon CloudWatch Events para iniciar a execução de uma máquina de estado sempre que a fonte do evento emitir um evento. Para obter mais informações, consulte Criação de uma regra de CloudWatch eventos que é acionada em um evento.
- Mapeie a origem do evento para uma função do Lambda e escreva o código da função para executar sua máquina de estado. A AWS Lambda função é invocada sempre que sua fonte de eventos emite um evento, iniciando, por sua vez, a execução de uma máquina de estado. Para obter mais informações, consulte Usar o AWS Lambda com o Amazon SQS.

O exemplo de projeto usa o segundo método para iniciar uma execução cada vez que a fila do Amazon SQS enviar uma mensagem. Você pode usar uma configuração semelhante para acionar a execução do fluxos de trabalho expressos a partir de outras fontes de eventos, como o Amazon Simple Storage Service (Amazon S3), o Amazon DynamoDB e o Amazon Kinesis.

Para obter mais informações sobre o fluxos de trabalho expressos e as integrações de serviço do Step Functions, consulte o seguinte:

- Comparação entre os fluxos de trabalho padrão e expresso
- Usando AWS Step Functions com outros serviços
- Cotas

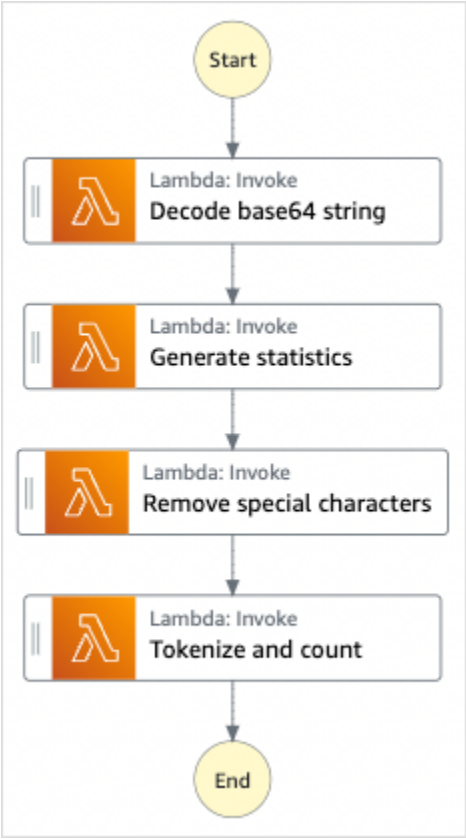
Etapa 1: Criar a máquina de estado e provisionar os recursos

1. Abra o console do Step Functions e clique em Criar máquina de estado.
2. Digite **`Process high-volume messages from SQS`** na caixa de pesquisa e escolha Processar mensagens de alto volume do SQS nos resultados da pesquisa que são retornados.
3. Escolha Próximo para continuar.
4. Step Functions lista o Serviços da AWS usado no projeto de amostra que você selecionou. Também mostra um gráfico de fluxo de trabalho para o projeto de amostra. Implante esse projeto no seu Conta da AWS ou use-o como ponto de partida para criar seus próprios projetos. Com base em como você deseja prosseguir, escolha Executar uma demonstração ou Criar com base nela.

   Este projeto de exemplo implementa os recursos a seguir:

   - Quatro funções do Lambda
   - Uma fila do Amazon SQS
   - Uma máquina de AWS Step Functions estado
   - Funções relacionadas AWS Identity and Access Management (IAM)

   A imagem a seguir mostra o gráfico do fluxo de trabalho para o exemplo de projeto Processar mensagens de alto volume do SQS:



5. Escolha Usar modelo para continuar com a seleção.
6. Execute um destes procedimentos:

   - Se você selecionou Criar com base nela, o Step Functions criará o protótipo do fluxo de trabalho para o projeto de amostra selecionado. O Step Functions não implanta os recursos listados na definição do fluxo de trabalho.

     No Modo de design do Workflow Studio, arraste e solte os estados do Navegador de estados para continuar criando seu protótipo de fluxo de trabalho. Ou mude para o Modo de código que fornece um editor de código integrado semelhante ao VS Code para atualizar a definição Amazon States Language (ASL) de sua máquina de estado no console Step Functions. Para obter mais informações sobre o uso do Workflow Studio para criar suas máquinas de estado, consulte Como usar o Workflow Studio.

     > **Important**
     >
     > Lembre-se de atualizar o espaço reservado do nome do recurso da Amazon (ARN) para os recursos usados no projeto de amostra antes de executar o fluxo de trabalho.

- Se você selecionou Executar uma demonstração, o Step Functions cria um projeto de amostra somente para leitura que usa um AWS CloudFormation modelo para implantar os AWS recursos listados nesse modelo no seu. Conta da AWS

  > **Tip**
  >
  > Para visualizar a definição da máquina de estado do projeto de amostra, escolha Código.

  Quando estiver pronto, escolha Implemente e execute para implantar o projeto de amostra e criar os recursos.

  Pode levar até 10 minutos para que esses recursos e as permissões relacionadas ao IAM sejam criados. Enquanto seus recursos estão sendo implantados, você pode abrir o link do CloudFormation Stack ID para ver quais recursos estão sendo provisionados.

  Depois que todos os recursos do projeto de exemplo forem criados, você poderá ver o novo projeto de exemplo listado na página Máquinas de estado.

  > **Important**
  >
  > Taxas padrão podem ser aplicadas a cada serviço usado no CloudFormation modelo.

Etapa 2: Acionar a execução da máquina de estado

1. Abra o console do Amazon SQS.
2. Selecione a fila que foi criada pelo projeto de exemplo.

   O nome será semelhante a Example-SQSQueue-wJalrXUtnFEMI.
3. Na lista Queue Actions (Ações da fila), selecione Send a Message (Enviar uma mensagem).
4. Use o botão de cópia para copiar a seguinte mensagem e, na janela Send a Message (Enviar uma mensagem), insira-a e escolha Send Message (Enviar mensagem) .

> **Note**
>
> Nesta mensagem de exemplo, a linha `input:` foi formatada com quebras de linha para caber na página. Use o botão Copiar ou certifique-se de que ele seja inserido como uma única linha sem quebras.

```
{
      "input":
 "QW5kIGxpa2UgdGhlIGJhc2VsZXNzIGZhYnJpYyBvZiB0aGlzIHZpc2lvbiwgVGhlIGNsb3VkLWNhcHBlZCB0b3dlc

  91cyBwYWxhY2VzLCBUaGUgc29sZW1uIHRlbXBsZXMsIHRoZSBncmVhdCBnbG9iZSBpdHNlbGbigJQgWWVhLCBhbGw

  ZXJpdOKAlHNoYWxsIGRpc3NvbHZlLCBBbmQgbGlrZSB0aGlzIGluc3Vic3RhbnRpYWwgcGFnZWFudCBmYWRlZCwgT

  FjayBiZWhpbmQuIFdlIGFyZSBzdWNoIHN0dWZmIEFzIGRyZWFtcyBhcmUgbWFkZSBvbiwgYW5kIG91ciBsaXR0bGU

  ZGVkIHdpdGggYSBzbGVlcC4gU2lyLCBJIGFtIHZleGVkLiBCZWFyIHdpdGggbXkgd2Vha25lc3MuIE15IG9sZCBic

  xlZC4gQmUgbm90IGRpc3R1cmJlZCB3aXRoIG15IGluZmlybWl0eS4gSWYgeW91IGJlIHBsZWFzZWQsIHJldGlyZSB

  QW5kIHRoZXJlIHJlcG9zZS4gQSB0dXJuIG9yIHR3byBJ4oCZbGwgd2FsayBUbyBzdGlsbCBteSBzdWF0aW5nIG1pb
}
```

5. Escolha Fechar.
6. Abra o console do Step Functions.
7. Acesse seu grupo de CloudWatch registros do Amazon Logs e inspecione os registros. O nome do grupo de registros será semelhante a example- ExpressLogGroup -WjAlrxUtnFemi.

Exemplo de código da função do Lambda

Veja a seguir um código da função Lambda que mostra como a função Lambda inicial inicia a execução de uma máquina de estado usando o 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
    )
```

Exemplo de código da máquina de estado

O fluxo de trabalho expresso neste projeto de exemplo consiste em um conjunto de funções do Lambda para processamento de texto.

Para obter mais informações sobre como AWS Step Functions controlar outros AWS serviços, consulteUsando AWS Step Functions com outros serviços.

```
{
  "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
    }
  }
}
```

Exemplo do IAM

Esse exemplo de política AWS Identity and Access Management (IAM) gerado pelo projeto de amostra inclui o menor privilégio necessário para executar a máquina de estado e os recursos relacionados. Recomendamos que você inclua apenas as permissões que forem necessárias em suas políticas do 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"
```

```
        }
    ]
}
```

A política a seguir garante que haja permissões suficientes para CloudWatch os 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 obter informações sobre como configurar o IAM ao usar Step Functions com outros AWS serviços, consulte[Políticas do IAM para serviços integrados](#).

Exemplo de verificação seletiva (Fluxos de trabalho expressos)

Esse projeto de exemplo demonstra como combinar fluxos de trabalho padrão e expressos executando uma simulação de fluxo de trabalho de comércio eletrônico que faz verificação seletiva. A implantação desse exemplo de projeto cria uma máquina de estado de fluxos de trabalho padrão, uma máquina de estado do fluxos de trabalho expressos aninhada, uma função do AWS Lambda , uma fila do Amazon Simple Queue Service (Amazon SQS) e um tópico do Amazon Simple Notification Service (Amazon SNS).

Para obter mais informações sobre fluxos de trabalho expressos, fluxos de trabalho aninhados e integrações de serviço do Step Functions, consulte o seguinte:

- Comparação entre os fluxos de trabalho padrão e expresso
- Iniciar execuções de fluxo de trabalho usando um estado Tarefa
- Usando AWS Step Functions com outros serviços

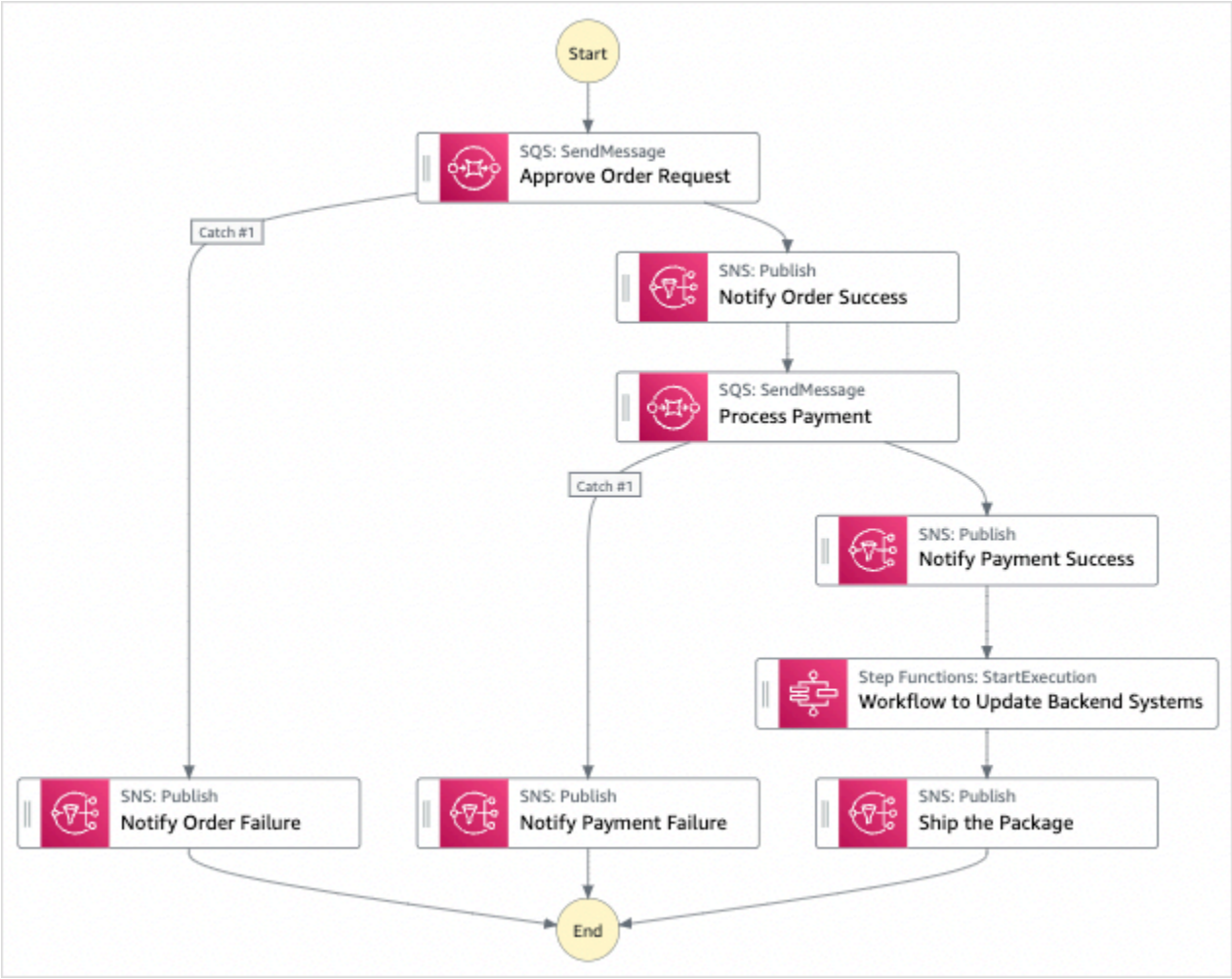
Etapa 1: Criar a máquina de estado e provisionar os recursos

1. Abra o console do Step Functions e clique em Criar máquina de estado.
2. Digite **`Selective checkpointing example`** no campo de pesquisa e escolha Exemplo de verificação seletiva nos resultados da pesquisa que são retornados.
3. Escolha Próximo para continuar.
4. Step Functions lista o Serviços da AWS usado no projeto de amostra que você selecionou. Também mostra um gráfico de fluxo de trabalho para o projeto de amostra. Implante esse projeto no seu Conta da AWS ou use-o como ponto de partida para criar seus próprios projetos. Com base em como você deseja prosseguir, escolha Executar uma demonstração ou Criar com base nela.

   Este projeto de exemplo implementa os recursos a seguir.

   - Uma AWS Lambda função
   - Uma fila do Amazon SQS
   - Um tópico do Amazon SNS
   - Uma máquina de AWS Step Functions estado do tipo Standard
   - Uma máquina de estado do Step Functions do tipo Express
   - Funções relacionadas AWS Identity and Access Management (IAM)

   A imagem a seguir mostra o gráfico do fluxo de trabalho para o projeto de exemplo de Exemplo de verificação seletiva:



5. Escolha Usar modelo para continuar com a seleção.
6. Execute um destes procedimentos:

   - Se você selecionou Criar com base nela, o Step Functions criará o protótipo do fluxo de trabalho para o projeto de amostra selecionado. O Step Functions não implanta os recursos listados na definição do fluxo de trabalho.

     No Modo de design do Workflow Studio, arraste e solte os estados do Navegador de estados para continuar criando seu protótipo de fluxo de trabalho. Ou mude para o Modo de código que fornece um editor de código integrado semelhante ao VS Code para atualizar a definição Amazon States Language (ASL) de sua máquina de estado no console Step Functions. Para obter mais informações sobre o uso do Workflow Studio para criar suas máquinas de estado, consulte Como usar o Workflow Studio.

> **Important**
>
> Lembre-se de atualizar o espaço reservado do nome do recurso da Amazon (ARN) para os recursos usados no projeto de amostra antes de executar o fluxo de trabalho.

- Se você selecionou Executar uma demonstração, o Step Functions cria um projeto de amostra somente para leitura que usa um AWS CloudFormation modelo para implantar os AWS recursos listados nesse modelo no seu. Conta da AWS

  > **Tip**
  >
  > Para visualizar a definição da máquina de estado do projeto de amostra, escolha Código.

  Quando estiver pronto, escolha Implemente e execute para implantar o projeto de amostra e criar os recursos.

  Pode levar até 10 minutos para que esses recursos e as permissões relacionadas ao IAM sejam criados. Enquanto seus recursos estão sendo implantados, você pode abrir o link do CloudFormation Stack ID para ver quais recursos estão sendo provisionados.

  Depois que todos os recursos do projeto de exemplo forem criados, você poderá ver o novo projeto de exemplo listado na página Máquinas de estado.

  > **Important**
  >
  > Taxas padrão podem ser aplicadas a cada serviço usado no CloudFormation modelo.

Depois que os recursos do projeto de exemplo forem implantados, faça o seguinte.

Etapa 2: Executar a máquina de estado

1. Na página Máquinas de estado, escolha seu projeto de exemplo.
2. Na página do projeto de exemplo, escolha Iniciar execução.
3. Na caixa de diálogo Iniciar execução, faça o seguinte:

1. (Opcional) Para identificar a execução, insira um nome para ela no campo Nome. Por padrão, o Step Functions cria automaticamente um nome de execução exclusivo.

   > **Note**
   >
   > Step Functions permite criar nomes para máquinas de estado, execuções e atividades, além de rótulos que contenham caracteres não ASCII. Esses nomes não ASCII não funcionam com a Amazon. CloudWatch Para garantir que você possa acompanhar CloudWatch as métricas, escolha um nome que use somente caracteres ASCII.

2. (Opcional) No campo Entrada, insira os valores de entrada no formato JSON para executar o fluxo de trabalho.

   Se você optar por Executar uma demonstração, não precisará fornecer nenhuma entrada de execução.

   > **Note**
   >
   > Se o projeto de demonstração que você implementou contiver dados de entrada de execução pré-preenchidos, use essa entrada para executar a máquina de estado.

3. Selecione Iniciar execução.
4. O console do Step Functions direciona você para uma página em que o título é o ID da execução. Essa página é conhecida como página de Detalhes da execução. Nesta página, você pode revisar os resultados da execução à medida que a execução avança ou após a conclusão.

   Para revisar os resultados da execução, escolha estados individuais na Exibição em gráfico e, em seguida, escolha as guias individuais no painel Detalhes da etapa para visualizar os detalhes de cada estado, incluindo entrada, saída e definição, respectivamente. Para obter detalhes sobre as informações de execução que você pode visualizar na página Detalhes da execução, consulte Página de Detalhes da execução — Visão geral da interface.

4. Acesse seu grupo de CloudWatch registros de registros e inspecione os registros. O nome do grupo de registros será semelhante a example- ExpressLogGroup -WjAlrxUtnFemi.

Exemplo de código da máquina de estado principal (fluxos de trabalho padrão)

A máquina de estado neste exemplo de projeto integra-se com o Amazon SQS, o Amazon SNS e o fluxos de trabalho expressos do Step Functions.

Navegue por esta máquina de estado de exemplo para ver como o Step Functions processa a entrada do Amazon SQS e do Amazon SNS e, em seguida, usa uma máquina de estado do fluxos de trabalho expressos aninhada para atualizar sistemas de back-end.

Para obter mais informações sobre como AWS Step Functions controlar outros AWS serviços, consulteUsando AWS Step Functions com outros serviços.

```
{
    "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
        }
    }
}
```

Exemplo de perfil do IAM para a máquina de estado principal

Essas políticas de exemplo AWS Identity and Access Management (IAM) geradas pelo projeto de amostra incluem o menor privilégio necessário para executar a máquina de estado e os recursos relacionados. Recomendamos que você inclua apenas as permissões que forem necessárias em suas políticas do IAM.

Política do Amazon SNS:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "sns:Publish"
            ],
            "Resource": "arn:aws:sns:us-east-1:123456789012:Checkpoint-SNSTopic-
wJalrXUtnFEMI",
            "Effect": "Allow"
        }
    ]
}
```

Política do 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 execução dos 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"
        }
    ]
}
```

Exemplo de código de máquina de estado para a máquina de estado aninhado (fluxos de trabalho expressos)

A máquina de estado neste projeto de exemplo atualiza informações de back-end quando chamada pela máquina de estado principal.

Navegue por esta máquina de estado de exemplo para ver como o Step Functions atualiza os diferentes componentes dos simulados de sistemas de back-end de comércio eletrônico.

Para obter mais informações sobre como AWS Step Functions controlar outros AWS serviços, consulteUsando AWS Step Functions com outros serviços.


```
{
  "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
    }
  }
}
```

Exemplo de perfil do IAM para máquina de estado secundária

Esse exemplo de política AWS Identity and Access Management (IAM) gerado pelo projeto de amostra inclui o menor privilégio necessário para executar a máquina de estado e os recursos relacionados. Recomendamos que você inclua apenas as permissões que forem necessárias em suas políticas do IAM.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": [
```

```
                "arn:aws:lambda:us-east-1:123456789012:function:Example-
UpdateDatabaseLambdaFunction-wJalrXUtnFEMI"
            ],
            "Effect": "Allow"
        }
    ]
}
```

A política a seguir garante que haja permissões suficientes para CloudWatch os 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 obter informações sobre como configurar o IAM ao usar o Step Functions com outros serviços da AWS , consulte Políticas do IAM para serviços integrados.

Crie um AWS CodeBuild projeto (CodeBuild, Amazon SNS)

Este exemplo de projeto demonstra como usá-lo AWS Step Functions para criar um AWS CodeBuild projeto, executar testes e, em seguida, enviar uma notificação ao Amazon SNS.

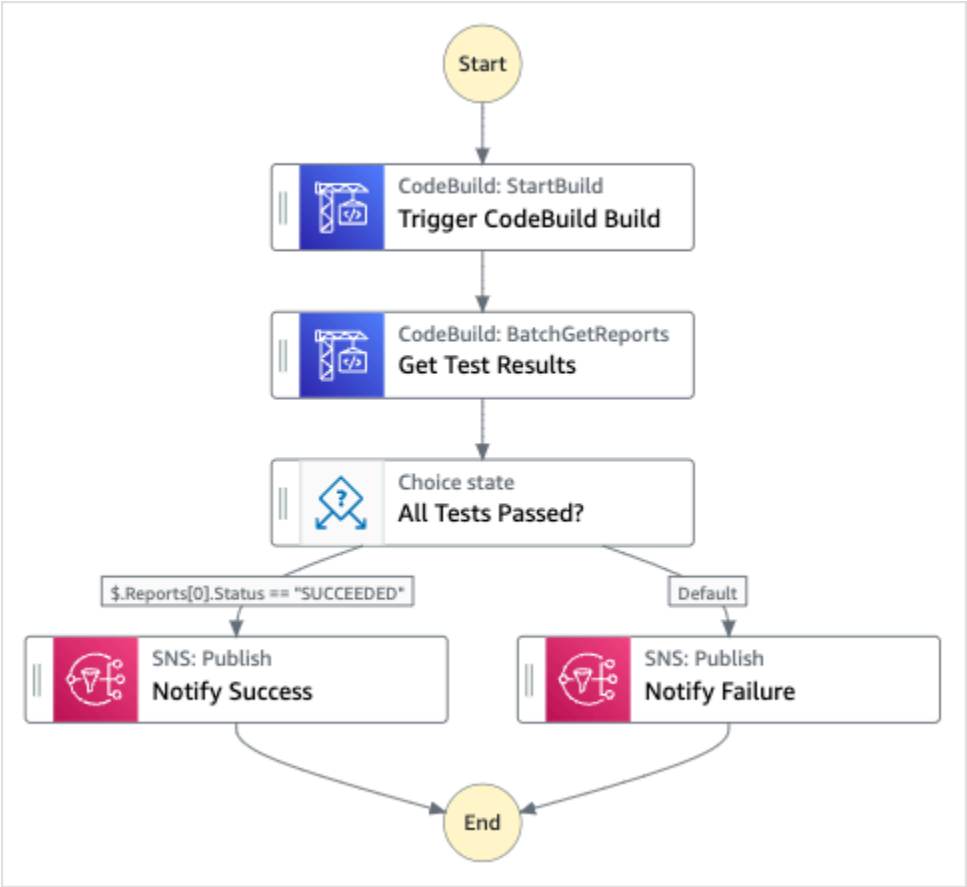
Etapa 1: Criar a máquina de estado e provisionar os recursos

1. Abra o console do Step Functions e clique em Criar máquina de estado.
2. Digite **`Start a CodeBuild build`** na caixa de pesquisa e escolha Iniciar uma CodeBuild compilação nos resultados da pesquisa que são retornados.
3. Escolha Próximo para continuar.
4. Step Functions lista o Serviços da AWS usado no projeto de amostra que você selecionou. Também mostra um gráfico de fluxo de trabalho para o projeto de amostra. Implante esse projeto no seu Conta da AWS ou use-o como ponto de partida para criar seus próprios projetos. Com base em como você deseja prosseguir, escolha Executar uma demonstração ou Criar com base nela.

   Este projeto de exemplo implementa os recursos a seguir.

   - Uma AWS CodeBuild construção
   - Um tópico do Amazon SNS
   - Uma máquina de AWS Step Functions estado
   - Funções relacionadas AWS Identity and Access Management (IAM)

   A imagem a seguir mostra o gráfico do fluxo de trabalho do projeto Start CodeBuild a build sample:



5. Escolha Usar modelo para continuar com a seleção.
6. Execute um destes procedimentos:

   - Se você selecionou Criar com base nela, o Step Functions criará o protótipo do fluxo de trabalho para o projeto de amostra selecionado. O Step Functions não implanta os recursos listados na definição do fluxo de trabalho.

     No Modo de design do Workflow Studio, arraste e solte os estados do Navegador de estados para continuar criando seu protótipo de fluxo de trabalho. Ou mude para o Modo de código que fornece um editor de código integrado semelhante ao VS Code para atualizar a definição Amazon States Language (ASL) de sua máquina de estado no console Step Functions. Para obter mais informações sobre o uso do Workflow Studio para criar suas máquinas de estado, consulte Como usar o Workflow Studio.

> **Important**
>
> Lembre-se de atualizar o espaço reservado do nome do recurso da Amazon (ARN) para os recursos usados no projeto de amostra antes de executar o fluxo de trabalho.

- Se você selecionou Executar uma demonstração, o Step Functions cria um projeto de amostra somente para leitura que usa um AWS CloudFormation modelo para implantar os AWS recursos listados nesse modelo no seu. Conta da AWS

  > **Tip**
  >
  > Para visualizar a definição da máquina de estado do projeto de amostra, escolha Código.

  Quando estiver pronto, escolha Implemente e execute para implantar o projeto de amostra e criar os recursos.

  Pode levar até 10 minutos para que esses recursos e as permissões relacionadas ao IAM sejam criados. Enquanto seus recursos estão sendo implantados, você pode abrir o link do CloudFormation Stack ID para ver quais recursos estão sendo provisionados.

  Depois que todos os recursos do projeto de exemplo forem criados, você poderá ver o novo projeto de exemplo listado na página Máquinas de estado.

  > **Important**
  >
  > Taxas padrão podem ser aplicadas a cada serviço usado no CloudFormation modelo.

Etapa 2: Executar a máquina de estado

1. Na página Máquinas de estado, escolha seu projeto de exemplo.
2. Na página do projeto de exemplo, escolha Iniciar execução.
3. Na caixa de diálogo Iniciar execução, faça o seguinte:

1. (Opcional) Para identificar a execução, insira um nome para ela no campo Nome. Por padrão, o Step Functions cria automaticamente um nome de execução exclusivo.

   > **Note**
   >
   > Step Functions permite criar nomes para máquinas de estado, execuções e atividades, além de rótulos que contenham caracteres não ASCII. Esses nomes não ASCII não funcionam com a Amazon. CloudWatch Para garantir que você possa acompanhar CloudWatch as métricas, escolha um nome que use somente caracteres ASCII.

2. (Opcional) No campo Entrada, insira os valores de entrada no formato JSON para executar o fluxo de trabalho.

   Se você optar por Executar uma demonstração, não precisará fornecer nenhuma entrada de execução.

   > **Note**
   >
   > Se o projeto de demonstração que você implementou contiver dados de entrada de execução pré-preenchidos, use essa entrada para executar a máquina de estado.

3. Selecione Iniciar execução.
4. O console do Step Functions direciona você para uma página em que o título é o ID da execução. Essa página é conhecida como página de Detalhes da execução. Nesta página, você pode revisar os resultados da execução à medida que a execução avança ou após a conclusão.

   Para revisar os resultados da execução, escolha estados individuais na Exibição em gráfico e, em seguida, escolha as guias individuais no painel Detalhes da etapa para visualizar os detalhes de cada estado, incluindo entrada, saída e definição, respectivamente. Para obter detalhes sobre as informações de execução que você pode visualizar na página Detalhes da execução, consulte Página de Detalhes da execução — Visão geral da interface.

Exemplo de código da máquina de estado

A máquina de estado neste projeto de amostra se integra ao Amazon CodeBuild SNS.

Navegue por este exemplo de máquina de estado para ver como o Step Functions usa uma máquina de estado para criar um CodeBuild projeto e, em seguida, envia um tópico do Amazon SNS com uma mensagem sobre se o trabalho foi bem-sucedido ou falhou.

Para obter mais informações sobre como o Step Functions pode controlar outros AWS serviços, consulteUsando AWS Step Functions com outros serviços.

```
{
  "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 obter informações sobre como configurar o IAM ao usar o Step Functions com outros serviços da AWS , consulte Políticas do IAM para serviços integrados.

Pré-processar dados e treinar um modelo de Machine Learning

Este exemplo de projeto demonstra como usar SageMaker e AWS Step Functions pré-processar dados e treinar um modelo de aprendizado de máquina.

Nesse projeto, o Step Functions usa uma função do Lambda para propagar um bucket do Amazon S3 com um conjunto de dados de teste e um script do Python para processamento de dados. Em seguida, ele treina um modelo de aprendizado de máquina e executa uma transformação em lote, usando a integração SageMaker de serviços.

Para obter mais informações sobre integrações de serviços SageMaker e Step Functions, consulte o seguinte:

- Usando AWS Step Functions com outros serviços
- Gerencie SageMaker com Step Functions

> **Note**
>
> Este projeto de exemplo pode incorrer em cobranças.
> Para novos AWS usuários, um nível de uso gratuito está disponível. Neste nível, os serviços são gratuitos abaixo de um determinado nível de uso. Para obter mais informações sobre AWS custos e o nível gratuito, consulte SageMaker Preços.

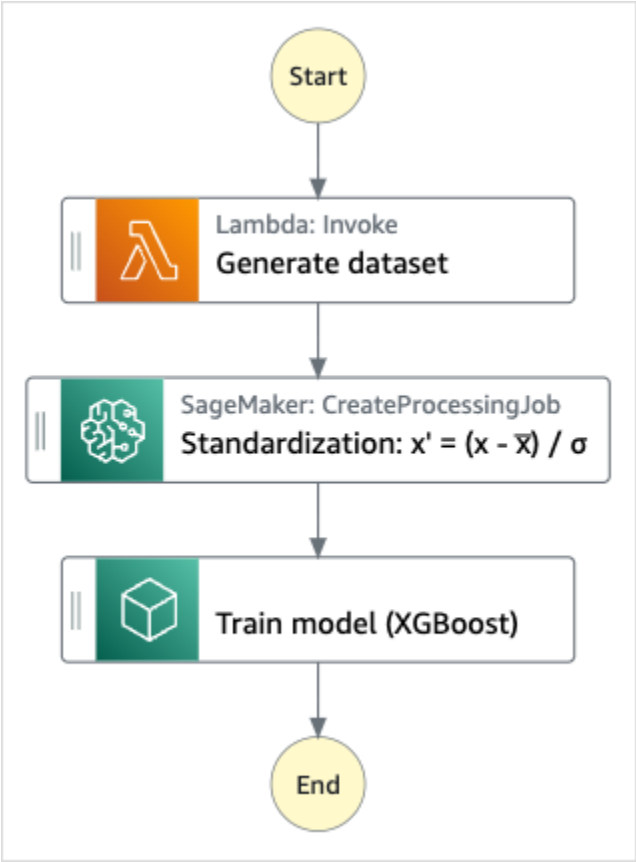
Etapa 1: Criar a máquina de estado e provisionar os recursos

1. Abra o console do Step Functions e clique em Criar máquina de estado.
2. Digite **`Preprocess data and train a machine learning model`** na caixa de pesquisa e escolha Pré-processar dados e treinar um modelo de machine learning nos resultados da pesquisa que são retornados.
3. Escolha Próximo para continuar.
4. Step Functions lista o Serviços da AWS usado no projeto de amostra que você selecionou. Também mostra um gráfico de fluxo de trabalho para o projeto de amostra. Implante esse projeto no seu Conta da AWS ou use-o como ponto de partida para criar seus próprios projetos. Com base em como você deseja prosseguir, escolha Executar uma demonstração ou Criar com base nela.

   Este projeto de exemplo implementa os recursos a seguir.

   - Uma AWS Lambda função
   - Um bucket do Amazon S3
   - Uma máquina de AWS Step Functions estado
   - Funções relacionadas AWS Identity and Access Management (IAM)

   A imagem a seguir mostra o gráfico do fluxo de trabalho para o exemplo de projeto Pré-processar dados e treinar um modelo de machine learning:



5. Escolha Usar modelo para continuar com a seleção.
6. Execute um destes procedimentos:

   - Se você selecionou Criar com base nela, o Step Functions criará o protótipo do fluxo de trabalho para o projeto de amostra selecionado. O Step Functions não implanta os recursos listados na definição do fluxo de trabalho.

     No Modo de design do Workflow Studio, arraste e solte os estados do Navegador de estados para continuar criando seu protótipo de fluxo de trabalho. Ou mude para o Modo de código que fornece um editor de código integrado semelhante ao VS Code para atualizar a definição Amazon States Language (ASL) de sua máquina de estado no console Step Functions. Para obter mais informações sobre o uso do Workflow Studio para criar suas máquinas de estado, consulte Como usar o Workflow Studio.

     > **Important**
     >
     > Lembre-se de atualizar o espaço reservado do nome do recurso da Amazon (ARN) para os recursos usados no projeto de amostra antes de executar o fluxo de trabalho.

- Se você selecionou Executar uma demonstração, o Step Functions cria um projeto de amostra somente para leitura que usa um AWS CloudFormation modelo para implantar os AWS recursos listados nesse modelo no seu. Conta da AWS

  > **Tip**
  >
  > Para visualizar a definição da máquina de estado do projeto de amostra, escolha Código.

  Quando estiver pronto, escolha Implemente e execute para implantar o projeto de amostra e criar os recursos.

  Pode levar até 10 minutos para que esses recursos e as permissões relacionadas ao IAM sejam criados. Enquanto seus recursos estão sendo implantados, você pode abrir o link do CloudFormation Stack ID para ver quais recursos estão sendo provisionados.

  Depois que todos os recursos do projeto de exemplo forem criados, você poderá ver o novo projeto de exemplo listado na página Máquinas de estado.

  > **Important**
  >
  > Taxas padrão podem ser aplicadas a cada serviço usado no CloudFormation modelo.

Etapa 2: Executar a máquina de estado

1. Na página Máquinas de estado, escolha seu projeto de exemplo.
2. Na página do projeto de exemplo, escolha Iniciar execução.
3. Na caixa de diálogo Iniciar execução, faça o seguinte:

   1. (Opcional) Para identificar a execução, insira um nome para ela no campo Nome. Por padrão, o Step Functions cria automaticamente um nome de execução exclusivo.

      > **Note**
      >
      > Step Functions permite criar nomes para máquinas de estado, execuções e atividades, além de rótulos que contenham caracteres não ASCII. Esses nomes não

ASCII não funcionam com a Amazon. CloudWatch Para garantir que você possa acompanhar CloudWatch as métricas, escolha um nome que use somente caracteres ASCII.

2. (Opcional) No campo Entrada, insira os valores de entrada no formato JSON para executar o fluxo de trabalho.

   Se você optar por Executar uma demonstração, não precisará fornecer nenhuma entrada de execução.

   > **Note**
   >
   > Se o projeto de demonstração que você implementou contiver dados de entrada de execução pré-preenchidos, use essa entrada para executar a máquina de estado.

3. Selecione Iniciar execução.
4. O console do Step Functions direciona você para uma página em que o título é o ID da execução. Essa página é conhecida como página de Detalhes da execução. Nesta página, você pode revisar os resultados da execução à medida que a execução avança ou após a conclusão.

   Para revisar os resultados da execução, escolha estados individuais na Exibição em gráfico e, em seguida, escolha as guias individuais no painel Detalhes da etapa para visualizar os detalhes de cada estado, incluindo entrada, saída e definição, respectivamente. Para obter detalhes sobre as informações de execução que você pode visualizar na página Detalhes da execução, consulte Página de Detalhes da execução — Visão geral da interface.

Exemplo de código da máquina de estado

A máquina de estado neste projeto de amostra se integra com SageMaker e AWS Lambda passando parâmetros diretamente para esses recursos e usa um bucket do Amazon S3 para a fonte e a saída dos dados de treinamento.

Navegue por este exemplo de máquina de estado para ver como o Step Functions controla o Lambda e. SageMaker

Para obter mais informações sobre como AWS Step Functions controlar outros AWS serviços, consulteUsando AWS Step Functions com outros serviços.

```
{
  "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 obter informações sobre como configurar o IAM ao usar Step Functions com outros AWS serviços, consulte[Políticas do IAM para serviços integrados](#).

Exemplo do IAM

Essas políticas de exemplo AWS Identity and Access Management (IAM) geradas pelo projeto de amostra incluem o menor privilégio necessário para executar a máquina de estado e os recursos relacionados. Recomendamos que você inclua apenas as permissões que forem necessárias em suas políticas do 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"
        }
    ]
}
```

A política a seguir permite que a função do Lambda propague o bucket do Amazon S3 com dados de exemplo.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:PutObject"
            ],
```

```
            "Resource": "arn:aws:s3:::featuretransform-
bucketforcodeanddata-1jn1le6gadwfz/*",
            "Effect": "Allow"
        }
    ]
}
```

Para obter informações sobre como configurar o IAM ao usar Step Functions com outros AWS serviços, consulte[Políticas do IAM para serviços integrados](#).

Exemplo de orquestração do Lambda

Este projeto de amostra demonstra como integrar AWS Lambda funções em máquinas de estado Step Functions.

Neste projeto, o Step Functions usa funções do Lambda para verificar o preço de uma ação e determinar uma recomendação de negociação de compra ou venda. O usuário então recebe essa recomendação e pode escolher entre comprar ou vender as ações. O resultado da negociação é retornado usando um tópico do SNS.

Para obter mais informações sobre integrações de serviços do Step Functions, consulte o seguinte:

- Usando AWS Step Functions com outros serviços
- Políticas do IAM para:
  - Políticas do IAM para AWS Lambda
  - Políticas do IAM para Amazon SQS
  - Políticas do IAM para Amazon SNS

> **Note**
>
> Este projeto de exemplo pode incorrer em cobranças.
> Para novos AWS usuários, um nível de uso gratuito está disponível. Neste nível, os serviços são gratuitos abaixo de um determinado nível de uso. Para obter mais informações sobre AWS custos e o nível gratuito, consulte Preços.

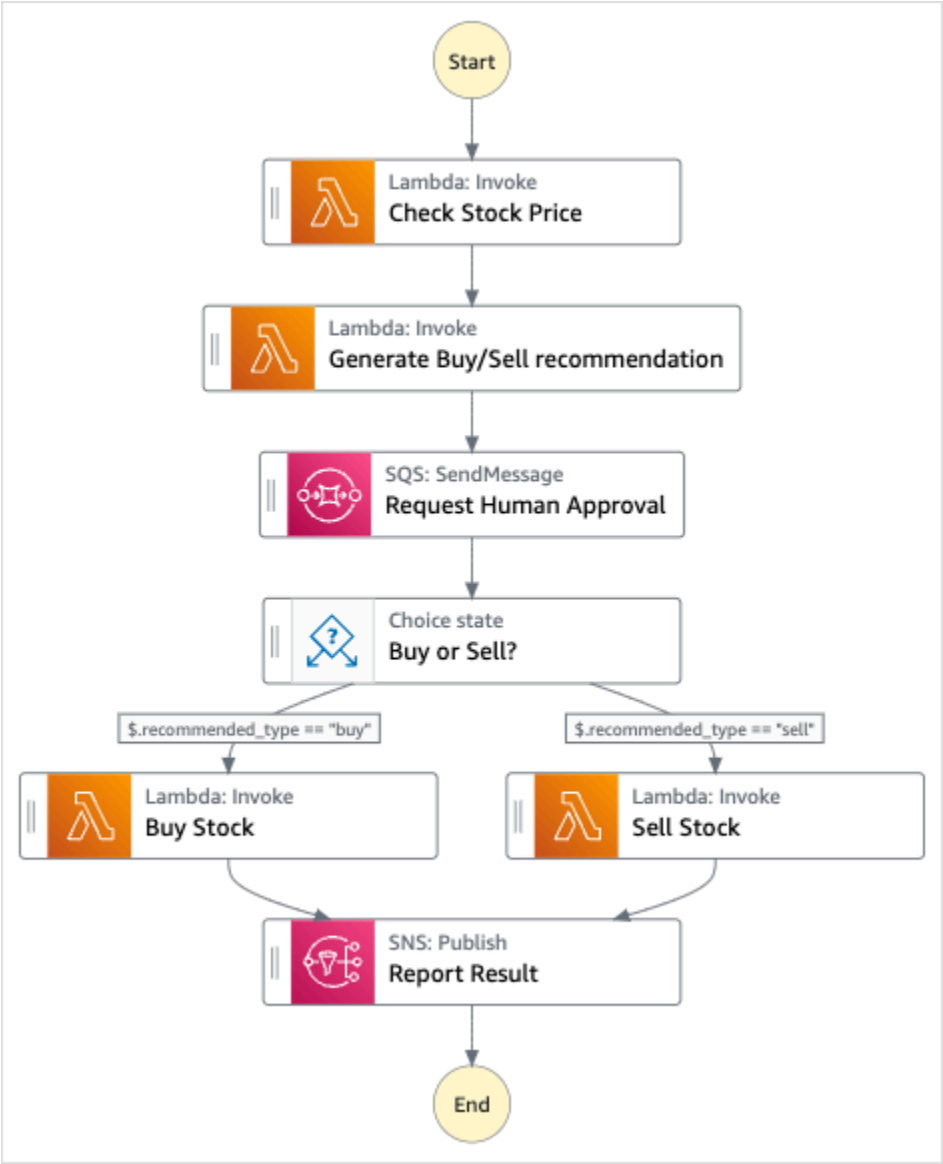
Etapa 1: Criar a máquina de estado e provisionar os recursos

1. Abra o console do Step Functions e clique em Criar máquina de estado.
2. Digite **`Orchestrate Lambda functions`** no campo de pesquisa e escolha Orquestrar funções do Lambda a partir dos resultados de pesquisa que são retornados.
3. Escolha Próximo para continuar.
4. Step Functions lista o Serviços da AWS usado no projeto de amostra que você selecionou. Também mostra um gráfico de fluxo de trabalho para o projeto de amostra. Implante esse projeto no seu Conta da AWS ou use-o como ponto de partida para criar seus próprios projetos. Com base em como você deseja prosseguir, escolha Executar uma demonstração ou Criar com base nela.

   Este projeto de exemplo implementa os recursos a seguir.

   - Cinco funções do Lambda
   - Uma fila do Amazon Simple Queue Service
   - Um tópico do Amazon Simple Notification Service
   - Uma máquina de estado do AWS Step Functions
   - Funções do AWS Identity and Access Management (IAM) relacionadas

   A imagem a seguir mostra o gráfico do fluxo de trabalho do exemplo de projeto para Orquestrar funções do Lambda:



5. Escolha Usar modelo para continuar com a seleção.
6. Execute um destes procedimentos:

   - Se você selecionou Criar com base nela, o Step Functions criará o protótipo do fluxo de trabalho para o projeto de amostra selecionado. O Step Functions não implanta os recursos listados na definição do fluxo de trabalho.

     No Modo de design do Workflow Studio, arraste e solte os estados do Navegador de estados para continuar criando seu protótipo de fluxo de trabalho. Ou mude para o Modo de código que fornece um editor de código integrado semelhante ao VS Code para atualizar a definição Amazon States Language (ASL) de sua máquina de estado no console Step Functions. Para

obter mais informações sobre o uso do Workflow Studio para criar suas máquinas de estado, consulte Como usar o Workflow Studio.

> **Important**
>
> Lembre-se de atualizar o espaço reservado do nome do recurso da Amazon (ARN) para os recursos usados no projeto de amostra antes de executar o fluxo de trabalho.

- Se você selecionou Executar uma demonstração, o Step Functions cria um projeto de amostra somente para leitura que usa um AWS CloudFormation modelo para implantar os AWS recursos listados nesse modelo no seu. Conta da AWS

  > **Tip**
  >
  > Para visualizar a definição da máquina de estado do projeto de amostra, escolha Código.

  Quando estiver pronto, escolha Implemente e execute para implantar o projeto de amostra e criar os recursos.

  Pode levar até 10 minutos para que esses recursos e as permissões relacionadas ao IAM sejam criados. Enquanto seus recursos estão sendo implantados, você pode abrir o link do CloudFormation Stack ID para ver quais recursos estão sendo provisionados.

  Depois que todos os recursos do projeto de exemplo forem criados, você poderá ver o novo projeto de exemplo listado na página Máquinas de estado.

  > **Important**
  >
  > Taxas padrão podem ser aplicadas a cada serviço usado no CloudFormation modelo.

Etapa 2: Executar a máquina de estado

Depois que todos os recursos forem provisionados e implantados, a caixa de diálogo Iniciar execução será exibida.

1. Na página Máquinas de estado, escolha seu projeto de exemplo.

2. Na página do projeto de exemplo, escolha Iniciar execução.
3. Na caixa de diálogo Iniciar execução, faça o seguinte:

    1. (Opcional) Para identificar a execução, insira um nome para ela no campo Nome. Por padrão, o Step Functions cria automaticamente um nome de execução exclusivo.

        > **Note**
        >
        > Step Functions permite criar nomes para máquinas de estado, execuções e atividades, além de rótulos que contenham caracteres não ASCII. Esses nomes não ASCII não funcionam com a Amazon. CloudWatch Para garantir que você possa acompanhar CloudWatch as métricas, escolha um nome que use somente caracteres ASCII.

    2. (Opcional) No campo Entrada, insira os valores de entrada no formato JSON para executar o fluxo de trabalho.

        Se você optar por Executar uma demonstração, não precisará fornecer nenhuma entrada de execução.

        > **Note**
        >
        > Se o projeto de demonstração que você implementou contiver dados de entrada de execução pré-preenchidos, use essa entrada para executar a máquina de estado.

    3. Selecione Iniciar execução.
    4. O console do Step Functions direciona você para uma página em que o título é o ID da execução. Essa página é conhecida como página de Detalhes da execução. Nesta página, você pode revisar os resultados da execução à medida que a execução avança ou após a conclusão.

        Para revisar os resultados da execução, escolha estados individuais na Exibição em gráfico e, em seguida, escolha as guias individuais no painel Detalhes da etapa para visualizar os detalhes de cada estado, incluindo entrada, saída e definição, respectivamente. Para obter detalhes sobre as informações de execução que você pode visualizar na página Detalhes da execução, consulte Página de Detalhes da execução — Visão geral da interface.

Sobre a máquina de estado e sua execução

A máquina de estado neste projeto de amostra se integra AWS Lambda passando parâmetros diretamente para esses recursos, usa uma fila do Amazon SQS para gerenciar a solicitação de aprovação humana e usa um tópico do Amazon SNS para retornar os resultados da consulta.

Uma execução do Step Functions recebe um texto JSON como entrada e passa essa entrada para o primeiro estado no fluxo de trabalho. Os estados individuais recebem dados do JSON como entrada e geralmente passam dados do JSON como saída para o próximo estado. Neste exemplo de projeto, a saída de cada etapa é passada como entrada para a próxima etapa no fluxo de trabalho. Por exemplo, a etapa Gerar recomendação de compra/venda recebe a saída da etapa Verificar preço da ação como entrada. Além disso, a saída da etapa Gerar recomendação de compra/venda é passada como entrada para a próxima etapa, Solicitar aprovação humana, que imita uma etapa de aprovação humana.

> **Note**
>
> Para visualizar a saída retornada por uma etapa e a entrada passada para outra etapa, abra a página Detalhes da execução do fluxo de trabalho. Na seção Detalhes da etapa, visualize a entrada e a saída de cada etapa selecionada no Modo de visualização.

Para implementar uma etapa de aprovação humana, você normalmente pausa a execução do fluxo de trabalho até que um token de tarefa seja retornado. Neste exemplo de projeto, uma mensagem é passada para uma fila do Amazon SQS, que é usada como um gatilho para a função do Lambda definida para lidar com a funcionalidade de retorno de chamada. A mensagem contém um token de tarefa e a saída retornada pela etapa anterior. A função do Lambda é invocada com a carga da mensagem. A execução do fluxo de trabalho é pausada até receber o token da tarefa de volta com uma chamada de API `SendTaskSuccess`. Para mais informações sobre tokens de tarefa, consulte Aguardar um retorno de chamada com um token de tarefa.

O código a seguir para a função `StepFunctionsSample-HelloLambda-ApproveSqsLambda` mostra como ela é definida para aprovar automaticamente qualquer tarefa enviada pela fila do Amazon SQS por meio da máquina de estado do Step Functions.

Exemplo de código da função do Lambda para lidar com a funcionalidade de retorno de chamada e retornar o token da tarefa

```
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);
      });
  }
};
```

Navegue por esta máquina de estado de exemplo para ver como o Step Functions controla o Lambda e o Amazon SQS.

Para obter mais informações sobre como AWS Step Functions controlar outros AWS serviços, consulteUsando AWS Step Functions com outros serviços.

```
{
    "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 obter informações sobre como configurar o IAM ao usar Step Functions com outros AWS serviços, consulte[Políticas do IAM para serviços integrados](#).

Exemplos do IAM

Essas políticas de exemplo AWS Identity and Access Management (IAM) geradas pelo projeto de amostra incluem o menor privilégio necessário para executar a máquina de estado e os recursos relacionados. Recomendamos que você inclua apenas as permissões que forem necessárias em suas políticas do 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 obter informações sobre como configurar o IAM ao usar Step Functions com outros AWS serviços, consulte Políticas do IAM para serviços integrados.

Iniciar uma consulta do Athena

Esse projeto de exemplo, baseado em fluxos de trabalho padrão, demonstra como usar o Step Functions e o Amazon Athena para iniciar uma consulta do Athena e enviar uma notificação com os resultados da consulta.

Neste projeto, o Step Functions usa funções Lambda e um AWS Glue rastreador para gerar um conjunto de dados de exemplo. Em seguida, ele executa uma consulta usando a integração do serviço Athena e retorna os resultados usando um tópico do SNS.

Para obter mais informações sobre integrações de serviços do Athena e do Step Functions, consulte os tópicos a seguir.

- Usando AWS Step Functions com outros serviços
- Chame o Athena com o Step Functions

> **Note**
>
> Este projeto de exemplo pode incorrer em cobranças.
> Para novos AWS usuários, um nível de uso gratuito está disponível. Neste nível, os serviços são gratuitos abaixo de um determinado nível de uso. Para obter mais informações sobre AWS custos e o nível gratuito, consulte Preços do Athena.

Etapa 1: Criar a máquina de estado e provisionar os recursos

1. Abra o console do Step Functions e clique em Criar máquina de estado.
2. Digite **`Start an Athena query`** na caixa de pesquisa e escolha Iniciar uma Athena consulta nos resultados da pesquisa que são retornados.
3. Escolha Próximo para continuar.
4. Step Functions lista o Serviços da AWS usado no projeto de amostra que você selecionou. Também mostra um gráfico de fluxo de trabalho para o projeto de amostra. Implante esse projeto no seu Conta da AWS ou use-o como ponto de partida para criar seus próprios projetos. Com base em como você deseja prosseguir, escolha Executar uma demonstração ou Criar com base nela.

   Este projeto de exemplo implementa os recursos a seguir.

   - Uma Amazon Athena consulta
   - Um Crawler do AWS Glue
   - Um tópico do Amazon SNS
   - Uma máquina de estado do AWS Step Functions

- Funções do AWS Identity and Access Management (IAM) relacionadas

A imagem a seguir mostra o gráfico do fluxo de trabalho do projeto de exemplo Iniciar uma consulta do Athena:

5. Escolha Usar modelo para continuar com a seleção.
6. Execute um destes procedimentos:

   - Se você selecionou Criar com base nela, o Step Functions criará o protótipo do fluxo de trabalho para o projeto de amostra selecionado. O Step Functions não implanta os recursos listados na definição do fluxo de trabalho.

     No Modo de design do Workflow Studio, arraste e solte os estados do Navegador de estados para continuar criando seu protótipo de fluxo de trabalho. Ou mude para o Modo de código que fornece um editor de código integrado semelhante ao VS Code para atualizar a definição Amazon States Language (ASL) de sua máquina de estado no console Step Functions. Para

obter mais informações sobre o uso do Workflow Studio para criar suas máquinas de estado, consulte Como usar o Workflow Studio.

> **Important**
>
> Lembre-se de atualizar o espaço reservado do nome do recurso da Amazon (ARN) para os recursos usados no projeto de amostra antes de executar o fluxo de trabalho.

- Se você selecionou Executar uma demonstração, o Step Functions cria um projeto de amostra somente para leitura que usa um AWS CloudFormation modelo para implantar os AWS recursos listados nesse modelo no seu. Conta da AWS

> **Tip**
>
> Para visualizar a definição da máquina de estado do projeto de amostra, escolha Código.

Quando estiver pronto, escolha Implemente e execute para implantar o projeto de amostra e criar os recursos.

Pode levar até 10 minutos para que esses recursos e as permissões relacionadas ao IAM sejam criados. Enquanto seus recursos estão sendo implantados, você pode abrir o link do CloudFormation Stack ID para ver quais recursos estão sendo provisionados.

Depois que todos os recursos do projeto de exemplo forem criados, você poderá ver o novo projeto de exemplo listado na página Máquinas de estado.

> **Important**
>
> Taxas padrão podem ser aplicadas a cada serviço usado no CloudFormation modelo.

Etapa 2: Executar a máquina de estado

1. Na página Máquinas de estado, escolha seu projeto de exemplo.
2. Na página do projeto de exemplo, escolha Iniciar execução.
3. Na caixa de diálogo Iniciar execução, faça o seguinte:

1. (Opcional) Para identificar a execução, insira um nome para ela no campo Nome. Por padrão, o Step Functions cria automaticamente um nome de execução exclusivo.

   > **Note**
   >
   > Step Functions permite criar nomes para máquinas de estado, execuções e atividades, além de rótulos que contenham caracteres não ASCII. Esses nomes não ASCII não funcionam com a Amazon. CloudWatch Para garantir que você possa acompanhar CloudWatch as métricas, escolha um nome que use somente caracteres ASCII.

2. (Opcional) No campo Entrada, insira os valores de entrada no formato JSON para executar o fluxo de trabalho.

   Se você optar por Executar uma demonstração, não precisará fornecer nenhuma entrada de execução.

   > **Note**
   >
   > Se o projeto de demonstração que você implementou contiver dados de entrada de execução pré-preenchidos, use essa entrada para executar a máquina de estado.

3. Selecione Iniciar execução.
4. O console do Step Functions direciona você para uma página em que o título é o ID da execução. Essa página é conhecida como página de Detalhes da execução. Nesta página, você pode revisar os resultados da execução à medida que a execução avança ou após a conclusão.

   Para revisar os resultados da execução, escolha estados individuais na Exibição em gráfico e, em seguida, escolha as guias individuais no painel Detalhes da etapa para visualizar os detalhes de cada estado, incluindo entrada, saída e definição, respectivamente. Para obter detalhes sobre as informações de execução que você pode visualizar na página Detalhes da execução, consulte Página de Detalhes da execução — Visão geral da interface.

Exemplo de código da máquina de estado

A máquina de estado neste projeto de amostra se integra ao Athena AWS Lambda e passa parâmetros diretamente para esses recursos, e usa um tópico do SNS para retornar os resultados da consulta.

Navegue por esta máquina de estado de exemplo para ver como o Step Functions controla o Lambda e o Athena.

Para obter mais informações sobre como AWS Step Functions controlar outros AWS serviços, consulteUsando AWS Step Functions com outros serviços.

```
{
  "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 obter informações sobre como configurar o IAM ao usar Step Functions com outros AWS serviços, consulte[Políticas do IAM para serviços integrados](#).

Exemplo do IAM

Essas políticas de exemplo AWS Identity and Access Management (IAM) geradas pelo projeto de amostra incluem o menor privilégio necessário para executar a máquina de estado e os recursos relacionados. Recomendamos que você inclua apenas as permissões que forem necessárias em suas políticas do 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 obter informações sobre como configurar o IAM ao usar Step Functions com outros AWS serviços, consulte[Políticas do IAM para serviços integrados](#).

Executar várias consultas (Amazon Athena, Amazon SNS)

Este exemplo de projeto demonstra como executar queries do Athena em sucessão e, em seguida, em paralelo, lidar com erros; depois, enviar uma notificação ao Amazon SNS com base no sucesso ou na falha das consultas.

Neste projeto, o Step Functions usa uma máquina de estado para executar queries do Athena de forma síncrona. Depois que os resultados da query forem retornados, entre no estado paralelo com duas queries do Athena em execução paralela. Em seguida, ele aguarda até o êxito ou a falha do trabalho e envia um tópico do Amazon SNS com uma mensagem sobre o êxito ou a falha do trabalho.

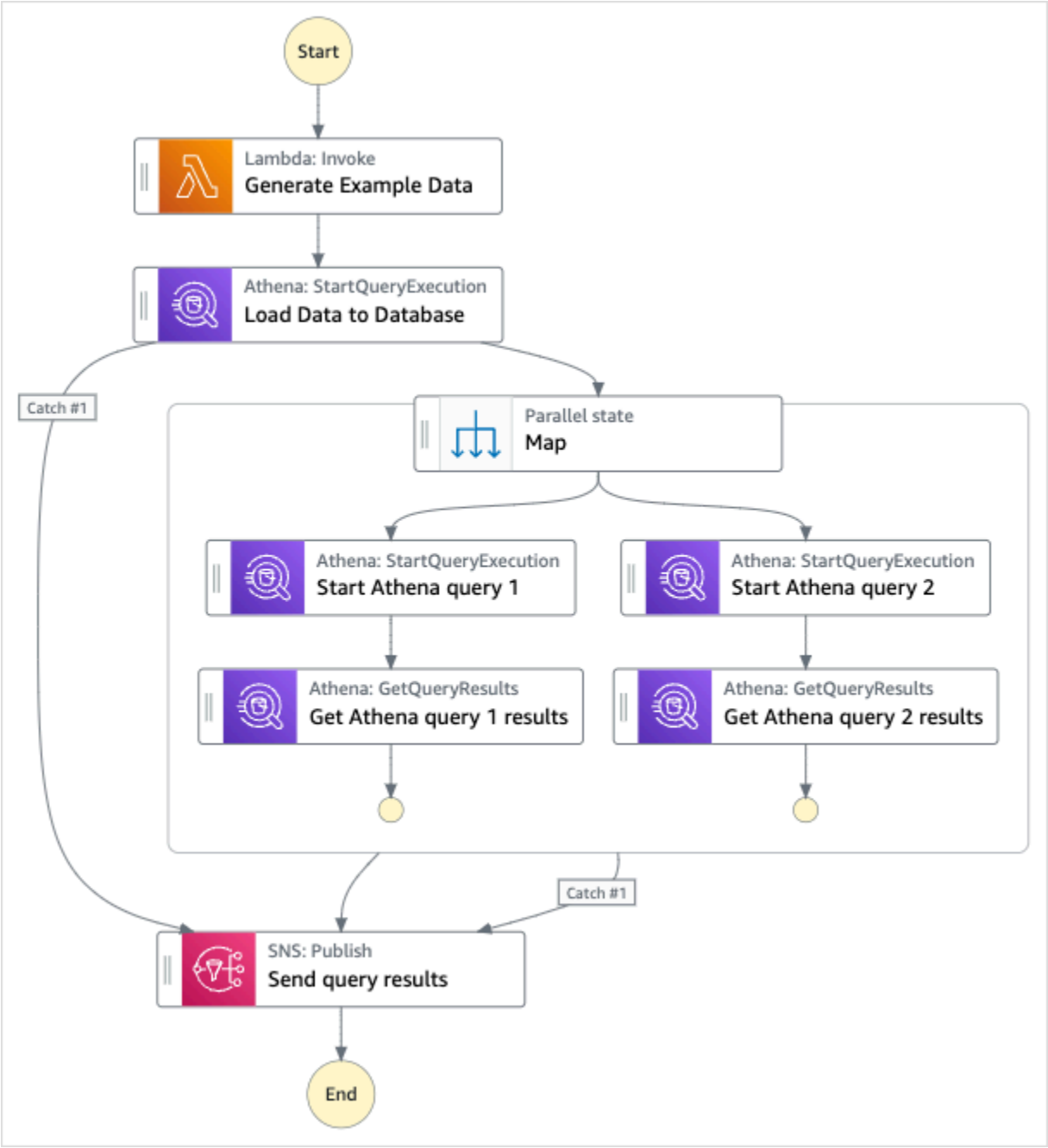
Etapa 1: Criar a máquina de estado e provisionar os recursos

1. Abra o console do Step Functions e clique em Criar máquina de estado.
2. Digite **`Execute multiple queries`** na caixa de pesquisa e escolha Executar várias consultas nos resultados da pesquisa que são retornados.
3. Escolha Próximo para continuar.
4. Step Functions lista o Serviços da AWS usado no projeto de amostra que você selecionou. Também mostra um gráfico de fluxo de trabalho para o projeto de amostra. Implante esse projeto no seu Conta da AWS ou use-o como ponto de partida para criar seus próprios projetos. Com base em como você deseja prosseguir, escolha Executar uma demonstração ou Criar com base nela.

   Este projeto de exemplo implementa os recursos a seguir.

   - Queries do Amazon Athena
   - Um tópico do Amazon SNS
   - Uma máquina de estado do AWS Step Functions
   - Funções do AWS Identity and Access Management (IAM) relacionadas

   A imagem a seguir mostra o gráfico do fluxo de trabalho do exemplo de projeto Executar várias consultas:



5. Escolha Usar modelo para continuar com a seleção.
6. Execute um destes procedimentos:

   - Se você selecionou Criar com base nela, o Step Functions criará o protótipo do fluxo de trabalho para o projeto de amostra selecionado. O Step Functions não implanta os recursos listados na definição do fluxo de trabalho.

     No Modo de design do Workflow Studio, arraste e solte os estados do Navegador de estados para continuar criando seu protótipo de fluxo de trabalho. Ou mude para o Modo de código

que fornece um editor de código integrado semelhante ao VS Code para atualizar a definição Amazon States Language (ASL) de sua máquina de estado no console Step Functions. Para obter mais informações sobre o uso do Workflow Studio para criar suas máquinas de estado, consulte Como usar o Workflow Studio.

> **Important**
>
> Lembre-se de atualizar o espaço reservado do nome do recurso da Amazon (ARN) para os recursos usados no projeto de amostra antes de executar o fluxo de trabalho.

- Se você selecionou Executar uma demonstração, o Step Functions cria um projeto de amostra somente para leitura que usa um AWS CloudFormation modelo para implantar os AWS recursos listados nesse modelo no seu. Conta da AWS

  > **Tip**
  >
  > Para visualizar a definição da máquina de estado do projeto de amostra, escolha Código.

  Quando estiver pronto, escolha Implemente e execute para implantar o projeto de amostra e criar os recursos.

  Pode levar até 10 minutos para que esses recursos e as permissões relacionadas ao IAM sejam criados. Enquanto seus recursos estão sendo implantados, você pode abrir o link do CloudFormation Stack ID para ver quais recursos estão sendo provisionados.

  Depois que todos os recursos do projeto de exemplo forem criados, você poderá ver o novo projeto de exemplo listado na página Máquinas de estado.

  > **Important**
  >
  > Taxas padrão podem ser aplicadas a cada serviço usado no CloudFormation modelo.

Etapa 2: Executar a máquina de estado

1. Na página Máquinas de estado, escolha seu projeto de exemplo.

2. Na página do projeto de exemplo, escolha Iniciar execução.
3. Na caixa de diálogo Iniciar execução, faça o seguinte:

    1. (Opcional) Para identificar a execução, insira um nome para ela no campo Nome. Por padrão, o Step Functions gera automaticamente um nome de execução exclusivo.

        > **Note**
        >
        > O Step Functions permite criar nomes de máquina de estado, execuções, atividades e labels que contenham caracteres não ASCII. Esses nomes não ASCII não funcionam com a Amazon. CloudWatch Para garantir que você possa acompanhar CloudWatch as métricas, escolha um nome que use somente caracteres ASCII.

    2. (Opcional) No campo Entrada, insira os valores de entrada no formato JSON para executar o fluxo de trabalho.

        Se você optar por Executar uma demonstração, não precisará fornecer nenhuma entrada de execução.
    3. Selecione Iniciar execução.
    4. O console do Step Functions direciona você para uma página em que o título é o ID da execução. Essa página é conhecida como página de Detalhes da execução. Nesta página, você pode revisar os resultados da execução à medida que a execução avança ou após a conclusão.

        Para revisar os resultados da execução, escolha estados individuais na Exibição em gráfico e, em seguida, escolha as guias individuais no painel Detalhes da etapa para visualizar os detalhes de cada estado, incluindo entrada, saída e definição, respectivamente. Para obter detalhes sobre as informações de execução que você pode visualizar na página Detalhes da execução, consulte Página de Detalhes da execução — Visão geral da interface.

Exemplo de código da máquina de estado

A máquina de estado deste exemplo de projeto se integra ao Amazon Athena e ao Amazon SNS passando parâmetros diretamente para esses recursos.

Navegue por esta máquina de estado de exemplo para ver como o Step Functions controla o Amazon Athena e o Amazon SNS conectando-se com o nome do recurso da Amazon (ARN) no campo `Resource` e passando o `Parameters` para a API de serviço.

Para obter mais informações sobre como AWS Step Functions controlar outros AWS serviços, consulteUsando AWS Step Functions com outros serviços.

```
{
    "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
      }
    }
  }
```

Exemplos do IAM

Esse exemplo de política AWS Identity and Access Management (IAM) gerado pelo projeto de amostra inclui o menor privilégio necessário para executar a máquina de estado e os recursos relacionados. Recomendamos que você inclua apenas as permissões que forem necessárias em suas políticas do 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 obter informações sobre como configurar o IAM ao usar Step Functions com outros AWS serviços, consulte[Políticas do IAM para serviços integrados](#).

Consulte grandes conjuntos de dados (Amazon Athena, Amazon S3, Amazon SNS AWS Glue)

Este exemplo de projeto demonstra como ingerir um grande conjunto de dados no Amazon S3 e particioná-lo AWS Glue por meio de rastreadores e, em seguida, executar consultas do Amazon Athena nessa partição.

Neste projeto, a máquina de estado Step Functions invoca um AWS Glue rastreador que particiona um grande conjunto de dados no Amazon S3. Depois que o AWS Glue rastreador retorna uma mensagem de sucesso, o fluxo de trabalho executa as consultas do Athena nessa partição. Quando a execução da query for concluída com sucesso, uma notificação do Amazon SNS será enviada para um tópico do Amazon SNS.

Etapa 1: Criar a máquina de estado e provisionar os recursos

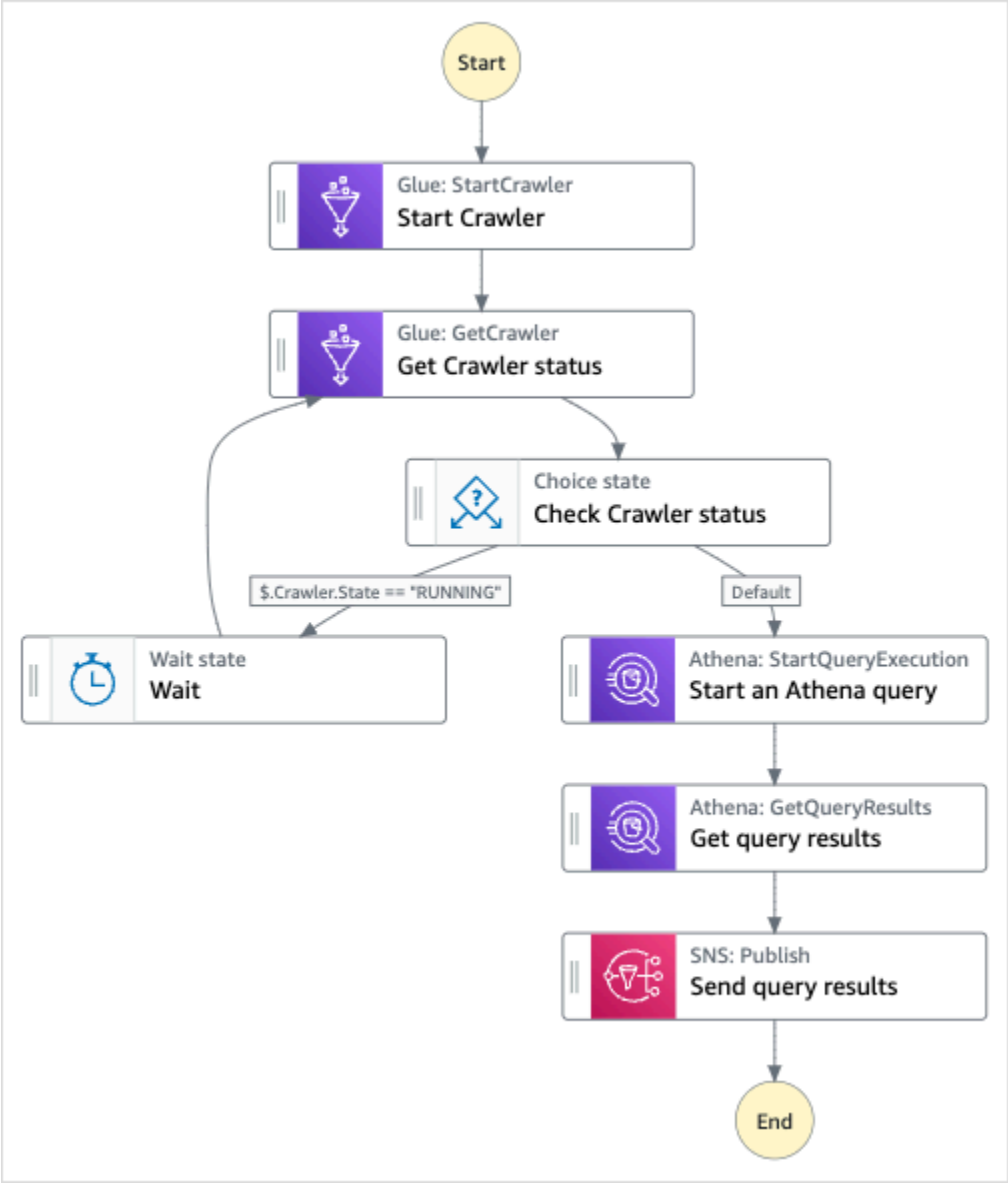
1. Abra o console do Step Functions e clique em Criar máquina de estado.
2. Digite **`Query large datasets`** na caixa de pesquisa e escolha Consultar grandes conjuntos de dados nos resultados da pesquisa que são retornados.
3. Escolha Próximo para continuar.
4. Step Functions lista o Serviços da AWS usado no projeto de amostra que você selecionou. Também mostra um gráfico de fluxo de trabalho para o projeto de amostra. Implante esse projeto no seu Conta da AWS ou use-o como ponto de partida para criar seus próprios projetos. Com base em como você deseja prosseguir, escolha Executar uma demonstração ou Criar com base nela.

   Este projeto de exemplo implementa os recursos a seguir.

   - Um bucket do Amazon S3

- Um Crawler do AWS Glue
- Um tópico do Amazon SNS
- Uma máquina de estado do AWS Step Functions
- Funções do AWS Identity and Access Management (IAM) relacionadas

A imagem a seguir mostra o gráfico do fluxo de trabalho do exemplo de projeto Consultar grandes conjuntos de dados:



5. Escolha Usar modelo para continuar com a seleção.
6. Execute um destes procedimentos:

- Se você selecionou Criar com base nela, o Step Functions criará o protótipo do fluxo de trabalho para o projeto de amostra selecionado. O Step Functions não implanta os recursos listados na definição do fluxo de trabalho.

  No Modo de design do Workflow Studio, arraste e solte os estados do Navegador de estados para continuar criando seu protótipo de fluxo de trabalho. Ou mude para o Modo de código que fornece um editor de código integrado semelhante ao VS Code para atualizar a definição Amazon States Language (ASL) de sua máquina de estado no console Step Functions. Para obter mais informações sobre o uso do Workflow Studio para criar suas máquinas de estado, consulte Como usar o Workflow Studio.

  > **Important**
  >
  > Lembre-se de atualizar o espaço reservado do nome do recurso da Amazon (ARN) para os recursos usados no projeto de amostra antes de executar o fluxo de trabalho.

- Se você selecionou Executar uma demonstração, o Step Functions cria um projeto de amostra somente para leitura que usa um AWS CloudFormation modelo para implantar os AWS recursos listados nesse modelo no seu. Conta da AWS

  > **Tip**
  >
  > Para visualizar a definição da máquina de estado do projeto de amostra, escolha Código.

  Quando estiver pronto, escolha Implemente e execute para implantar o projeto de amostra e criar os recursos.

  Pode levar até 10 minutos para que esses recursos e as permissões relacionadas ao IAM sejam criados. Enquanto seus recursos estão sendo implantados, você pode abrir o link do CloudFormation Stack ID para ver quais recursos estão sendo provisionados.

  Depois que todos os recursos do projeto de exemplo forem criados, você poderá ver o novo projeto de exemplo listado na página Máquinas de estado.

> **Important**
>
> Taxas padrão podem ser aplicadas a cada serviço usado no CloudFormation modelo.

Etapa 2: Executar a máquina de estado

1. Na página Máquinas de estado, escolha seu projeto de exemplo.
2. Na página do projeto de exemplo, escolha Iniciar execução.
3. Na caixa de diálogo Iniciar execução, faça o seguinte:

    1. (Opcional) Para identificar a execução, insira um nome para ela no campo Nome. Por padrão, o Step Functions gera automaticamente um nome de execução exclusivo.

        > **Note**
        >
        > O Step Functions permite criar nomes de máquina de estado, execuções, atividades e labels que contenham caracteres não ASCII. Esses nomes não ASCII não funcionam com a Amazon. CloudWatch Para garantir que você possa acompanhar CloudWatch as métricas, escolha um nome que use somente caracteres ASCII.

    2. (Opcional) No campo Entrada, insira os valores de entrada no formato JSON para executar o fluxo de trabalho.

        Se você optar por Executar uma demonstração, não precisará fornecer nenhuma entrada de execução.

    3. Selecione Iniciar execução.
    4. O console do Step Functions direciona você para uma página em que o título é o ID da execução. Essa página é conhecida como página de Detalhes da execução. Nesta página, você pode revisar os resultados da execução à medida que a execução avança ou após a conclusão.

        Para revisar os resultados da execução, escolha estados individuais na Exibição em gráfico e, em seguida, escolha as guias individuais no painel Detalhes da etapa para visualizar os detalhes de cada estado, incluindo entrada, saída e definição, respectivamente. Para obter

detalhes sobre as informações de execução que você pode visualizar na página Detalhes da execução, consulte Página de Detalhes da execução — Visão geral da interface.

Exemplo de código da máquina de estado

A máquina de estado neste projeto de amostra se integra ao Amazon S3 AWS Glue, Amazon Athena e Amazon SNS passando parâmetros diretamente para esses recursos.

Navegue por este exemplo de máquina de estado para ver como o Step Functions controla o Amazon S3 AWS Glue, o Amazon Athena e o Amazon SNS conectando-se ao Amazon Resource Name (ARN) no campo e passando para `Resource` a API do serviço. `Parameters`

Para obter mais informações sobre como AWS Step Functions controlar outros AWS serviços, consulteUsando AWS Step Functions com outros serviços.

```
{
    "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
      }
    }
  }
```

Exemplos do IAM

Essas políticas de exemplo AWS Identity and Access Management (IAM) geradas pelo projeto de amostra incluem o menor privilégio necessário para executar a máquina de estado e os recursos relacionados. Recomendamos que você inclua apenas as permissões que forem necessárias em suas políticas do 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 obter informações sobre como configurar o IAM ao usar Step Functions com outros AWS serviços, consulte Políticas do IAM para serviços integrados.

Mantenha os dados atualizados (Amazon Athena, Amazon S3,) AWS Glue

Este projeto de amostra demonstra como consultar uma tabela de destino para obter dados atuais com o AWS Glue Catalog e, em seguida, atualizá-los com novos dados de outras fontes usando o Amazon Athena.

Neste projeto, a máquina de estado do Step Functions chama o AWS Glue Catalog para verificar se existe uma tabela de destino em um bucket do Amazon S3. Se nenhuma tabela for encontrada, uma nova tabela será criada. Em seguida, Step Functions executa uma consulta do Athena para adicionar linhas à tabela de destino de uma fonte de dados diferente: primeiro consultando a tabela de destino para obter a data mais recente, depois consultando a tabela de origem em busca de dados mais recentes e inserindo-os na tabela de destino.

Etapa 1: Criar a máquina de estado e provisionar os recursos

1. Abra o console do Step Functions e clique em Criar máquina de estado.
2. Digite **`Keep data up to date`** na caixa de pesquisa e escolha Manter os dados atualizados nos resultados da pesquisa que são retornados.
3. Escolha Próximo para continuar.
4. Step Functions lista o Serviços da AWS usado no projeto de amostra que você selecionou. Também mostra um gráfico de fluxo de trabalho para o projeto de amostra. Implante esse projeto no seu Conta da AWS ou use-o como ponto de partida para criar seus próprios projetos. Com base em como você deseja prosseguir, escolha Executar uma demonstração ou Criar com base nela.

   Este projeto de exemplo implementa os recursos a seguir.

   - Um bucket do Amazon S3
   - Queries do Amazon Athena
   - Uma chamada AWS Glue Data Catalog

- Uma máquina de estado do AWS Step Functions
- Funções do AWS Identity and Access Management (IAM) relacionadas

A imagem a seguir mostra o gráfico do fluxo de trabalho do exemplo de projeto do Manter os dados atualizados:

5. Escolha Usar modelo para continuar com a seleção.
6. Execute um destes procedimentos:

   - Se você selecionou Criar com base nela, o Step Functions criará o protótipo do fluxo de trabalho para o projeto de amostra selecionado. O Step Functions não implanta os recursos listados na definição do fluxo de trabalho.

     No Modo de design do Workflow Studio, arraste e solte os estados do Navegador de estados para continuar criando seu protótipo de fluxo de trabalho. Ou mude para o Modo de código que fornece um editor de código integrado semelhante ao VS Code para atualizar a definição Amazon States Language (ASL) de sua máquina de estado no console Step Functions. Para obter mais informações sobre o uso do Workflow Studio para criar suas máquinas de estado, consulte Como usar o Workflow Studio.

> **Important**
>
> Lembre-se de atualizar o espaço reservado do nome do recurso da Amazon (ARN) para os recursos usados no projeto de amostra antes de executar o fluxo de trabalho.

- Se você selecionou Executar uma demonstração, o Step Functions cria um projeto de amostra somente para leitura que usa um AWS CloudFormation modelo para implantar os AWS recursos listados nesse modelo no seu. Conta da AWS

  > **Tip**
  >
  > Para visualizar a definição da máquina de estado do projeto de amostra, escolha Código.

  Quando estiver pronto, escolha Implemente e execute para implantar o projeto de amostra e criar os recursos.

  Pode levar até 10 minutos para que esses recursos e as permissões relacionadas ao IAM sejam criados. Enquanto seus recursos estão sendo implantados, você pode abrir o link do CloudFormation Stack ID para ver quais recursos estão sendo provisionados.

  Depois que todos os recursos do projeto de exemplo forem criados, você poderá ver o novo projeto de exemplo listado na página Máquinas de estado.

  > **Important**
  >
  > Taxas padrão podem ser aplicadas a cada serviço usado no CloudFormation modelo.

Etapa 2: Executar a máquina de estado

1. Na página Máquinas de estado, escolha seu projeto de exemplo.
2. Na página do projeto de exemplo, escolha Iniciar execução.
3. Na caixa de diálogo Iniciar execução, faça o seguinte:

1. (Opcional) Para identificar a execução, insira um nome para ela no campo Nome. Por padrão, o Step Functions gera automaticamente um nome de execução exclusivo.

   > **Note**
   >
   > O Step Functions permite criar nomes de máquina de estado, execuções, atividades e labels que contenham caracteres não ASCII. Esses nomes não ASCII não funcionam com a Amazon. CloudWatch Para garantir que você possa acompanhar CloudWatch as métricas, escolha um nome que use somente caracteres ASCII.

2. (Opcional) No campo Entrada, insira os valores de entrada no formato JSON para executar o fluxo de trabalho.

   Se você optar por Executar uma demonstração, não precisará fornecer nenhuma entrada de execução.

3. Selecione Iniciar execução.

4. O console do Step Functions direciona você para uma página em que o título é o ID da execução. Essa página é conhecida como página de Detalhes da execução. Nesta página, você pode revisar os resultados da execução à medida que a execução avança ou após a conclusão.

   Para revisar os resultados da execução, escolha estados individuais na Exibição em gráfico e, em seguida, escolha as guias individuais no painel Detalhes da etapa para visualizar os detalhes de cada estado, incluindo entrada, saída e definição, respectivamente. Para obter detalhes sobre as informações de execução que você pode visualizar na página Detalhes da execução, consulte Página de Detalhes da execução — Visão geral da interface.

Exemplo de código da máquina de estado

A máquina de estado neste projeto de amostra se integra ao Amazon S3 AWS Glue e ao Amazon Athena passando parâmetros diretamente para esses recursos.

Navegue por este exemplo de máquina de estado para ver como o Step Functions controla o Amazon S3 e o Amazon Athena conectando-se ao Amazon Resource Name (ARN) no campo e passando `Parameters` para `Resource` a API do serviço. AWS Glue

Para obter mais informações sobre como AWS Step Functions controlar outros AWS serviços, consulteUsando AWS Step Functions com outros serviços.

```
{
    "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
      }
```

```
    }
  }
```

Exemplo do IAM

Esse exemplo de política AWS Identity and Access Management (IAM) gerado pelo projeto de amostra inclui o menor privilégio necessário para executar a máquina de estado e os recursos relacionados. Recomendamos que você inclua apenas as permissões que forem necessárias em suas políticas do 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 obter informações sobre como configurar o IAM ao usar Step Functions com outros AWS serviços, consulte Políticas do IAM para serviços integrados.

Gerenciar um cluster do Amazon EKS

Este projeto de exemplo demonstra como usar o Step Functions e o Amazon Elastic Kubernetes Service para criar um cluster do Amazon EKS com um grupo de nós, executar uma tarefa no Amazon EKS e examinar a saída. Ao terminar, ele remove os grupos de nós e o cluster do Amazon EKS.

Para obter mais informações sobre o Step Functions e suas integrações de serviço, consulte o seguinte:

- Usando AWS Step Functions com outros serviços
- Chamar o Amazon EKS com o Step Functions

> **Note**
>
> Este projeto de exemplo pode incorrer em cobranças.
> Para novos AWS usuários, um nível de uso gratuito está disponível. Neste nível, os serviços são gratuitos abaixo de um determinado nível de uso. Para obter mais informações sobre AWS custos e o nível gratuito, consulte os preços do Amazon EKS.

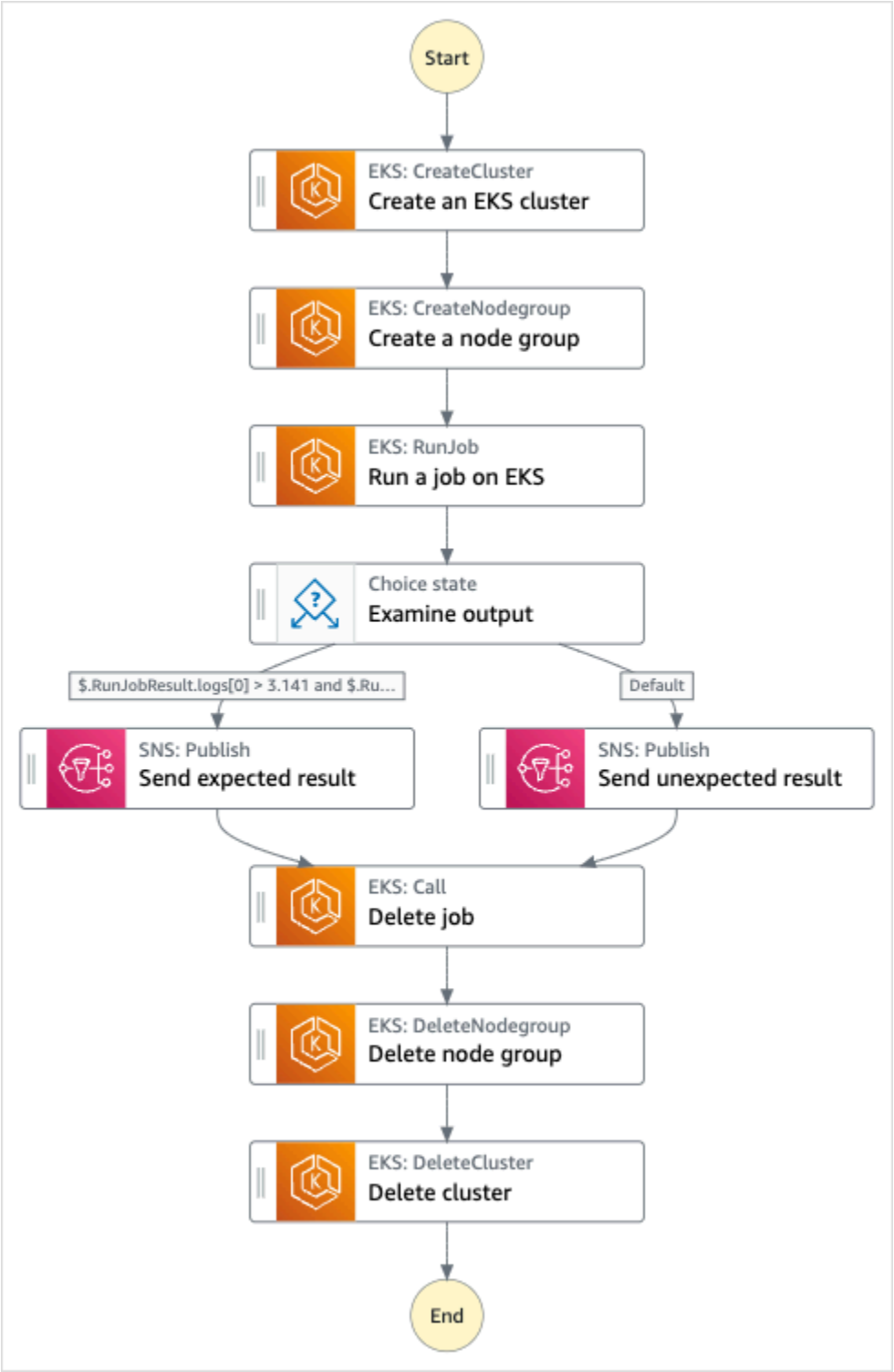
Etapa 1: Criar a máquina de estado e provisionar os recursos

1. Abra o console do Step Functions e clique em Criar máquina de estado.
2. Digite **`Manage an EKS cluster`** no campo de pesquisa e escolha Gerenciar um cluster EKS nos resultados da pesquisa que são retornados.
3. Escolha Próximo para continuar.
4. Step Functions lista o Serviços da AWS usado no projeto de amostra que você selecionou. Também mostra um gráfico de fluxo de trabalho para o projeto de amostra. Implante esse projeto no seu Conta da AWS ou use-o como ponto de partida para criar seus próprios projetos. Com base em como você deseja prosseguir, escolha Executar uma demonstração ou Criar com base nela.

Este projeto de exemplo implementa os recursos a seguir.

- Um cluster do Amazon Elastic Kubernetes Service
- Um tópico do Amazon SNS
- Uma máquina de estado do AWS Step Functions
- Funções do AWS Identity and Access Management (IAM) relacionadas

A imagem a seguir mostra o gráfico do fluxo de trabalho para o projeto de exemplo Gerenciar um cluster EKS:



5. Escolha Usar modelo para continuar com a seleção.
6. Execute um destes procedimentos:

- Se você selecionou Criar com base nela, o Step Functions criará o protótipo do fluxo de trabalho para o projeto de amostra selecionado. O Step Functions não implanta os recursos listados na definição do fluxo de trabalho.

  No Modo de design do Workflow Studio, arraste e solte os estados do Navegador de estados para continuar criando seu protótipo de fluxo de trabalho. Ou mude para o Modo de código que fornece um editor de código integrado semelhante ao VS Code para atualizar a definição Amazon States Language (ASL) de sua máquina de estado no console Step Functions. Para obter mais informações sobre o uso do Workflow Studio para criar suas máquinas de estado, consulte Como usar o Workflow Studio.

  > **Important**
  >
  > Lembre-se de atualizar o espaço reservado do nome do recurso da Amazon (ARN) para os recursos usados no projeto de amostra antes de executar o fluxo de trabalho.

- Se você selecionou Executar uma demonstração, o Step Functions cria um projeto de amostra somente para leitura que usa um AWS CloudFormation modelo para implantar os AWS recursos listados nesse modelo no seu. Conta da AWS

  > **Tip**
  >
  > Para visualizar a definição da máquina de estado do projeto de amostra, escolha Código.

  Quando estiver pronto, escolha Implemente e execute para implantar o projeto de amostra e criar os recursos.

  Pode levar até 10 minutos para que esses recursos e as permissões relacionadas ao IAM sejam criados. Enquanto seus recursos estão sendo implantados, você pode abrir o link do CloudFormation Stack ID para ver quais recursos estão sendo provisionados.

  Depois que todos os recursos do projeto de exemplo forem criados, você poderá ver o novo projeto de exemplo listado na página Máquinas de estado.

> **Important**
>
> Taxas padrão podem ser aplicadas a cada serviço usado no CloudFormation modelo.

Etapa 2: Executar a máquina de estado

1. Na página Máquinas de estado, escolha seu projeto de exemplo.
2. Na página do projeto de exemplo, escolha Iniciar execução.
3. Na caixa de diálogo Iniciar execução, faça o seguinte:

    1. (Opcional) Para identificar a execução, insira um nome para ela no campo Nome. Por padrão, o Step Functions cria automaticamente um nome de execução exclusivo.

        > **Note**
        >
        > Step Functions permite criar nomes para máquinas de estado, execuções e atividades, além de rótulos que contenham caracteres não ASCII. Esses nomes não ASCII não funcionam com a Amazon. CloudWatch Para garantir que você possa acompanhar CloudWatch as métricas, escolha um nome que use somente caracteres ASCII.

    2. (Opcional) No campo Entrada, insira os valores de entrada no formato JSON para executar o fluxo de trabalho.

        Se você optar por Executar uma demonstração, não precisará fornecer nenhuma entrada de execução.

        > **Note**
        >
        > Se o projeto de demonstração que você implementou contiver dados de entrada de execução pré-preenchidos, use essa entrada para executar a máquina de estado.

    3. Selecione Iniciar execução.
    4. O console do Step Functions direciona você para uma página em que o título é o ID da execução. Essa página é conhecida como página de Detalhes da execução. Nesta página,

você pode revisar os resultados da execução à medida que a execução avança ou após a conclusão.

Para revisar os resultados da execução, escolha estados individuais na Exibição em gráfico e, em seguida, escolha as guias individuais no painel Detalhes da etapa para visualizar os detalhes de cada estado, incluindo entrada, saída e definição, respectivamente. Para obter detalhes sobre as informações de execução que você pode visualizar na página Detalhes da execução, consulte Página de Detalhes da execução — Visão geral da interface.

Exemplo de código da máquina de estado

A máquina de estado neste projeto de exemplo se integra ao Amazon EKS criando um cluster e um grupo de nós do Amazon EKS e usa um tópico do SNS para retornar resultados.

Navegue por este exemplo de máquina de estado para ver como o Step Functions gerencia clusters e grupos de nós do Amazon EKS.

Para obter mais informações sobre como AWS Step Functions controlar outros AWS serviços, consulteUsando AWS Step Functions com outros serviços.

```
{
  "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 obter informações sobre como configurar o IAM ao usar Step Functions com outros AWS serviços, consulte[Políticas do IAM para serviços integrados](#).

Exemplo do IAM

Essas políticas de exemplo AWS Identity and Access Management (IAM) geradas pelo projeto de amostra incluem o menor privilégio necessário para executar a máquina de estado e os recursos relacionados. Recomendamos que você inclua apenas as permissões que forem necessárias em suas políticas do 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 obter informações sobre como configurar o IAM ao usar Step Functions com outros AWS serviços, consulte Políticas do IAM para serviços integrados.

Fazer uma chamada para o API Gateway

Este projeto de exemplo demonstra como usar o Step Functions para fazer uma chamada para o API Gateway e verificar se a chamada foi bem-sucedida.

Para obter mais informações sobre integrações de serviços do API Gateway e do Step Functions, consulte os tópicos a seguir.

- Usando AWS Step Functions com outros serviços
- Chame o API Gateway com o Step Functions

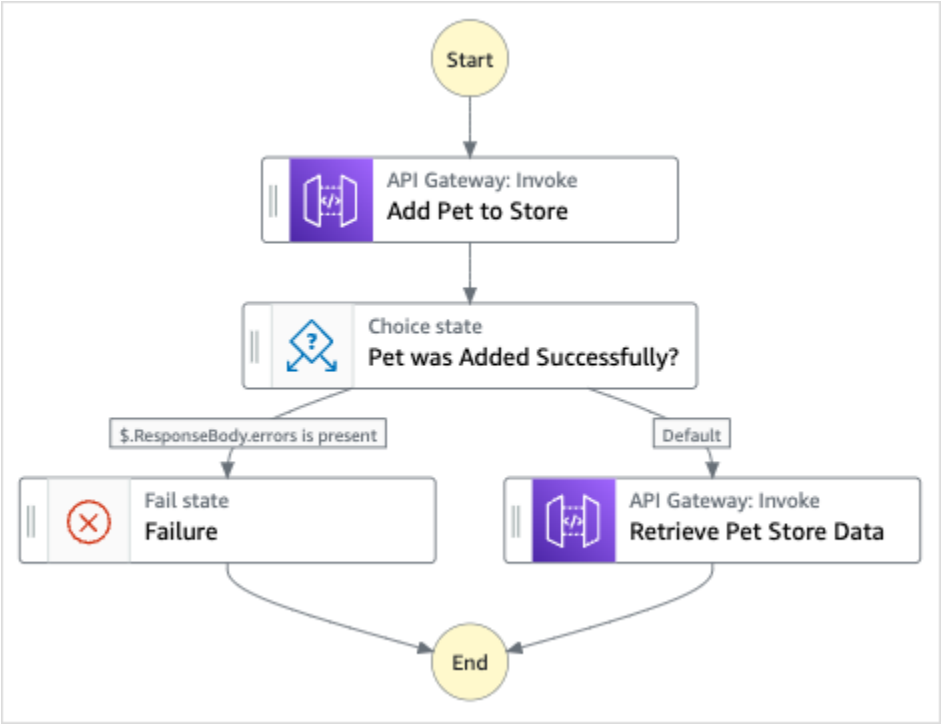
Etapa 1: Criar a máquina de estado e provisionar os recursos

1. Abra o console do Step Functions e clique em Criar máquina de estado.
2. Digite **`Make a call to API Gateway`** no campo de pesquisa e escolha Fazer uma chamada para API Gateway a partir dos resultados de pesquisa que são retornados.
3. Escolha Próximo para continuar.
4. Step Functions lista o Serviços da AWS usado no projeto de amostra que você selecionou. Também mostra um gráfico de fluxo de trabalho para o projeto de amostra. Implante esse projeto no seu Conta da AWS ou use-o como ponto de partida para criar seus próprios projetos. Com base em como você deseja prosseguir, escolha Executar uma demonstração ou Criar com base nela.

   Este projeto de exemplo implementa os recursos a seguir.

   - Uma API REST do Amazon API Gateway que é chamada pela máquina de estado.
   - Uma máquina de estado do AWS Step Functions
   - Funções do AWS Identity and Access Management (IAM) relacionadas

   A imagem a seguir mostra o gráfico do fluxo de trabalho do projeto de exemplo para Fazer uma chamada para o API Gateway:



5. Escolha Usar modelo para continuar com a seleção.
6. Execute um destes procedimentos:

   - Se você selecionou Criar com base nela, o Step Functions criará o protótipo do fluxo de trabalho para o projeto de amostra selecionado. O Step Functions não implanta os recursos listados na definição do fluxo de trabalho.

     No Modo de design do Workflow Studio, arraste e solte os estados do Navegador de estados para continuar criando seu protótipo de fluxo de trabalho. Ou mude para o Modo de código que fornece um editor de código integrado semelhante ao VS Code para atualizar a definição Amazon States Language (ASL) de sua máquina de estado no console Step Functions. Para obter mais informações sobre o uso do Workflow Studio para criar suas máquinas de estado, consulte Como usar o Workflow Studio.

     > **Important**
     >
     > Lembre-se de atualizar o espaço reservado do nome do recurso da Amazon (ARN) para os recursos usados no projeto de amostra antes de executar o fluxo de trabalho.

   - Se você selecionou Executar uma demonstração, o Step Functions cria um projeto de amostra somente para leitura que usa um AWS CloudFormation modelo para implantar os AWS recursos listados nesse modelo no seu. Conta da AWS

> **Tip**
>
> Para visualizar a definição da máquina de estado do projeto de amostra, escolha Código.

Quando estiver pronto, escolha Implemente e execute para implantar o projeto de amostra e criar os recursos.

Pode levar até 10 minutos para que esses recursos e as permissões relacionadas ao IAM sejam criados. Enquanto seus recursos estão sendo implantados, você pode abrir o link do CloudFormation Stack ID para ver quais recursos estão sendo provisionados.

Depois que todos os recursos do projeto de exemplo forem criados, você poderá ver o novo projeto de exemplo listado na página Máquinas de estado.

> **Important**
>
> Taxas padrão podem ser aplicadas a cada serviço usado no CloudFormation modelo.

Etapa 2: Executar a máquina de estado

1. Na página Máquinas de estado, escolha seu projeto de exemplo.
2. Na página do projeto de exemplo, escolha Iniciar execução.
3. Na caixa de diálogo Iniciar execução, faça o seguinte:

    1. (Opcional) Para identificar a execução, insira um nome para ela no campo Nome. Por padrão, o Step Functions cria automaticamente um nome de execução exclusivo.

        > **Note**
        >
        > Step Functions permite criar nomes para máquinas de estado, execuções e atividades, além de rótulos que contenham caracteres não ASCII. Esses nomes não ASCII não funcionam com a Amazon. CloudWatch Para garantir que você possa

acompanhar CloudWatch as métricas, escolha um nome que use somente caracteres ASCII.

2. (Opcional) No campo Entrada, insira os valores de entrada no formato JSON para executar o fluxo de trabalho.

   Se você optar por Executar uma demonstração, não precisará fornecer nenhuma entrada de execução.

   > **Note**
   >
   > Se o projeto de demonstração que você implementou contiver dados de entrada de execução pré-preenchidos, use essa entrada para executar a máquina de estado.

3. Selecione Iniciar execução.
4. O console do Step Functions direciona você para uma página em que o título é o ID da execução. Essa página é conhecida como página de Detalhes da execução. Nesta página, você pode revisar os resultados da execução à medida que a execução avança ou após a conclusão.

   Para revisar os resultados da execução, escolha estados individuais na Exibição em gráfico e, em seguida, escolha as guias individuais no painel Detalhes da etapa para visualizar os detalhes de cada estado, incluindo entrada, saída e definição, respectivamente. Para obter detalhes sobre as informações de execução que você pode visualizar na página Detalhes da execução, consulte Página de Detalhes da execução — Visão geral da interface.

Exemplo de código da máquina de estado

A máquina de estado neste projeto de exemplo se integra ao API Gateway chamando a API REST do API Gateway e transmitindo os parâmetros necessários.

Navegue por esse exemplo de máquina de estado para ver como o Step Functions interage com o Gateway de API.

Para obter mais informações sobre como AWS Step Functions controlar outros AWS serviços, consulteUsando AWS Step Functions com outros serviços.

```
{
  "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 obter informações sobre como configurar o IAM ao usar Step Functions com outros AWS serviços, consulte[Políticas do IAM para serviços integrados](#).

Exemplo do IAM

Essas políticas de exemplo AWS Identity and Access Management (IAM) geradas pelo projeto de amostra incluem o menor privilégio necessário para executar a máquina de estado e os recursos relacionados. Recomendamos que você inclua apenas as permissões que forem necessárias em suas políticas do 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 obter informações sobre como configurar o IAM ao usar Step Functions com outros AWS serviços, consulte[Políticas do IAM para serviços integrados](#).

Chamar um microsserviço em execução no Fargate usando a integração do API Gateway

Este projeto de amostra demonstra como usar o Step Functions para fazer uma chamada para o API Gateway a fim de interagir com um serviço e também para verificar se a chamada foi bem-sucedida. AWS Fargate

Para obter mais informações sobre integrações de serviços do API Gateway e do Step Functions, consulte os tópicos a seguir.

- Usando AWS Step Functions com outros serviços
- Chame o API Gateway com o Step Functions

> **Note**
>
> Este projeto de exemplo pode incorrer em cobranças.
> Para novos AWS usuários, um nível de uso gratuito está disponível. Neste nível, os serviços são gratuitos abaixo de um determinado nível de uso. Para obter mais informações sobre AWS custos e o nível gratuito, consulte Preços.

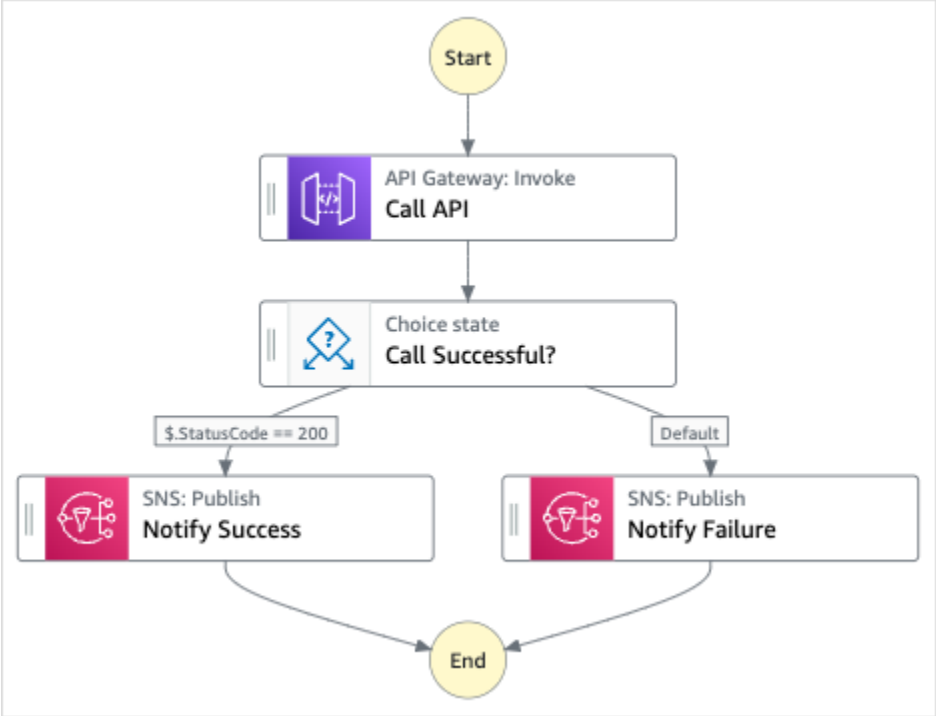
Etapa 1: Criar a máquina de estado e provisionar os recursos

1. Abra o console do Step Functions e clique em Criar máquina de estado.
2. Digite **`Call a microservice with API Gateway`** no campo de pesquisa e escolha Chamar um microsserviço com o API Gateway a partir dos resultados de pesquisa que são retornados.
3. Escolha Próximo para continuar.
4. Step Functions lista o Serviços da AWS usado no projeto de amostra que você selecionou. Também mostra um gráfico de fluxo de trabalho para o projeto de amostra. Implante esse projeto no seu Conta da AWS ou use-o como ponto de partida para criar seus próprios projetos. Com base em como você deseja prosseguir, escolha Executar uma demonstração ou Criar com base nela.

   Este projeto de exemplo implementa os recursos a seguir.

- Uma API HTTP do Amazon API Gateway que é chamada pela máquina de estado.
- Um Link Amazon VPC do Amazon API Gateway.
- Uma Amazon Virtual Private Cloud.
- Uma Application Load Balancer.
- Um cluster do Fargate.
- Um tópico do Amazon SNS
- Uma máquina de AWS Step Functions estado
- Funções relacionadas AWS Identity and Access Management (IAM)
- Vários serviços adicionais são necessários para permitir que esses recursos funcionem juntos.

A imagem a seguir mostra o gráfico do fluxo de trabalho para o projeto de exemplo de Chamar um microsserviço com o API Gateway:



5. Escolha Usar modelo para continuar com a seleção.
6. Execute um destes procedimentos:

   - Se você selecionou Criar com base nela, o Step Functions criará o protótipo do fluxo de trabalho para o projeto de amostra selecionado. O Step Functions não implanta os recursos listados na definição do fluxo de trabalho.

No Modo de design do Workflow Studio, arraste e solte os estados do Navegador de estados para continuar criando seu protótipo de fluxo de trabalho. Ou mude para o Modo de código que fornece um editor de código integrado semelhante ao VS Code para atualizar a definição Amazon States Language (ASL) de sua máquina de estado no console Step Functions. Para obter mais informações sobre o uso do Workflow Studio para criar suas máquinas de estado, consulte Como usar o Workflow Studio.

> **Important**
>
> Lembre-se de atualizar o espaço reservado do nome do recurso da Amazon (ARN) para os recursos usados no projeto de amostra antes de executar o fluxo de trabalho.

- Se você selecionou Executar uma demonstração, o Step Functions cria um projeto de amostra somente para leitura que usa um AWS CloudFormation modelo para implantar os AWS recursos listados nesse modelo no seu. Conta da AWS

  > **Tip**
  >
  > Para visualizar a definição da máquina de estado do projeto de amostra, escolha Código.

  Quando estiver pronto, escolha Implemente e execute para implantar o projeto de amostra e criar os recursos.

  Pode levar até 10 minutos para que esses recursos e as permissões relacionadas ao IAM sejam criados. Enquanto seus recursos estão sendo implantados, você pode abrir o link do CloudFormation Stack ID para ver quais recursos estão sendo provisionados.

  Depois que todos os recursos do projeto de exemplo forem criados, você poderá ver o novo projeto de exemplo listado na página Máquinas de estado.

  > **Important**
  >
  > Taxas padrão podem ser aplicadas a cada serviço usado no CloudFormation modelo.

Etapa 2: Executar a máquina de estado

1. Na página Máquinas de estado, escolha seu projeto de exemplo.
2. Na página do projeto de exemplo, escolha Iniciar execução.
3. Na caixa de diálogo Iniciar execução, faça o seguinte:

    1. (Opcional) Para identificar a execução, insira um nome para ela no campo Nome. Por padrão, o Step Functions cria automaticamente um nome de execução exclusivo.

        > **Note**
        >
        > Step Functions permite criar nomes para máquinas de estado, execuções e atividades, além de rótulos que contenham caracteres não ASCII. Esses nomes não ASCII não funcionam com a Amazon. CloudWatch Para garantir que você possa acompanhar CloudWatch as métricas, escolha um nome que use somente caracteres ASCII.

    2. (Opcional) No campo Entrada, insira os valores de entrada no formato JSON para executar o fluxo de trabalho.

        Se você optar por Executar uma demonstração, não precisará fornecer nenhuma entrada de execução.

        > **Note**
        >
        > Se o projeto de demonstração que você implementou contiver dados de entrada de execução pré-preenchidos, use essa entrada para executar a máquina de estado.

    3. Selecione Iniciar execução.
    4. O console do Step Functions direciona você para uma página em que o título é o ID da execução. Essa página é conhecida como página de Detalhes da execução. Nesta página, você pode revisar os resultados da execução à medida que a execução avança ou após a conclusão.

        Para revisar os resultados da execução, escolha estados individuais na Exibição em gráfico e, em seguida, escolha as guias individuais no painel Detalhes da etapa para visualizar os detalhes de cada estado, incluindo entrada, saída e definição, respectivamente. Para obter

detalhes sobre as informações de execução que você pode visualizar na página Detalhes da execução, consulte Página de Detalhes da execução — Visão geral da interface.

Exemplo de código da máquina de estado

A máquina de estado neste projeto de exemplo se integra ao Gateway de API chamando uma API HTTP do Gateway do API conectado a um serviço no Fargate. Ele é hospedado em uma sub-rede privada e acessado por meio de um application load balancer privado.

Navegue por esse exemplo de máquina de estado para ver como o Step Functions interage com o Gateway de API e retorna os resultados.

Para obter mais informações sobre como AWS Step Functions controlar outros AWS serviços, consulteUsando AWS Step Functions com outros serviços.

```
{
  "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 obter informações sobre como configurar o IAM ao usar Step Functions com outros AWS serviços, consulte[Políticas do IAM para serviços integrados](#).

Exemplo do IAM

Essas políticas de exemplo AWS Identity and Access Management (IAM) geradas pelo projeto de amostra incluem o menor privilégio necessário para executar a máquina de estado e os recursos relacionados. Recomendamos que você inclua apenas as permissões que forem necessárias em suas políticas do 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 obter informações sobre como configurar o IAM ao usar Step Functions com outros AWS serviços, consultePolíticas do IAM para serviços integrados.

Envie um evento personalizado para EventBridge

Este exemplo de projeto demonstra como usar o Step Functions para enviar um evento personalizado para um barramento de eventos que corresponda a uma regra com vários destinos (Amazon EventBridge AWS Lambda, Amazon Simple Notification Service, Amazon Simple Queue Service).

Para obter mais informações sobre o Step Functions e suas integrações de serviço, consulte o seguinte:

- Usando AWS Step Functions com outros serviços
- Chamada EventBridge com Step Functions

> **Note**
>
> Este projeto de exemplo pode incorrer em cobranças.
> Para novos AWS usuários, um nível de uso gratuito está disponível. Neste nível, os serviços são gratuitos abaixo de um determinado nível de uso. Para obter mais informações sobre AWS custos e o nível gratuito, consulte EventBridge Preços.

Etapa 1: Criar a máquina de estado e provisionar os recursos

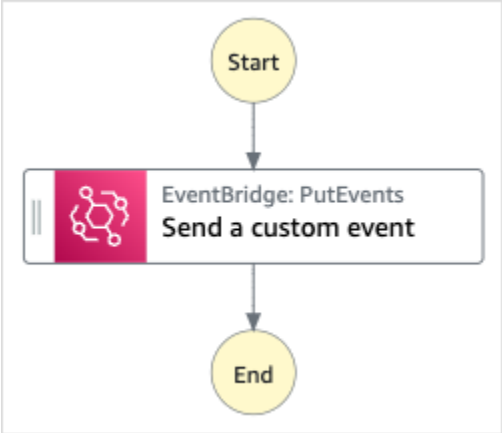
1. Abra o console do Step Functions e clique em Criar máquina de estado.
2. Digite **`Send a custom event to EventBridge`** na caixa de pesquisa e escolha Enviar um evento personalizado para o EventBridge nos resultados da pesquisa que são retornados.
3. Escolha Próximo para continuar.

4. Step Functions lista o Serviços da AWS usado no projeto de amostra que você selecionou. Também mostra um gráfico de fluxo de trabalho para o projeto de amostra. Implante esse projeto no seu Conta da AWS ou use-o como ponto de partida para criar seus próprios projetos. Com base em como você deseja prosseguir, escolha Executar uma demonstração ou Criar com base nela.

   Este projeto de exemplo implementa os recursos a seguir.

   - Um evento do Amazon EventBridge
   - Um tópico do Amazon SNS
   - Uma fila do Amazon SQS
   - Uma função do Lambda
   - Uma máquina de AWS Step Functions estado
   - Funções relacionadas AWS Identity and Access Management (IAM)

   A imagem a seguir mostra o gráfico do fluxo de trabalho do exemplo de projeto Enviar um evento personalizado para o EventBridge:

5. Escolha Usar modelo para continuar com a seleção.
6. Execute um destes procedimentos:

   - Se você selecionou Criar com base nela, o Step Functions criará o protótipo do fluxo de trabalho para o projeto de amostra selecionado. O Step Functions não implanta os recursos listados na definição do fluxo de trabalho.

     No Modo de design do Workflow Studio, arraste e solte os estados do Navegador de estados para continuar criando seu protótipo de fluxo de trabalho. Ou mude para o Modo de código que fornece um editor de código integrado semelhante ao VS Code para atualizar a definição

Amazon States Language (ASL) de sua máquina de estado no console Step Functions. Para obter mais informações sobre o uso do Workflow Studio para criar suas máquinas de estado, consulte Como usar o Workflow Studio.

> **Important**
>
> Lembre-se de atualizar o espaço reservado do nome do recurso da Amazon (ARN) para os recursos usados no projeto de amostra antes de executar o fluxo de trabalho.

- Se você selecionou Executar uma demonstração, o Step Functions cria um projeto de amostra somente para leitura que usa um AWS CloudFormation modelo para implantar os AWS recursos listados nesse modelo no seu. Conta da AWS

  > **Tip**
  >
  > Para visualizar a definição da máquina de estado do projeto de amostra, escolha Código.

  Quando estiver pronto, escolha Implemente e execute para implantar o projeto de amostra e criar os recursos.

  Pode levar até 10 minutos para que esses recursos e as permissões relacionadas ao IAM sejam criados. Enquanto seus recursos estão sendo implantados, você pode abrir o link do CloudFormation Stack ID para ver quais recursos estão sendo provisionados.

  Depois que todos os recursos do projeto de exemplo forem criados, você poderá ver o novo projeto de exemplo listado na página Máquinas de estado.

  > **Important**
  >
  > Taxas padrão podem ser aplicadas a cada serviço usado no CloudFormation modelo.

Etapa 2: Executar a máquina de estado

1. Na página Máquinas de estado, escolha seu projeto de exemplo.
2. Na página do projeto de exemplo, escolha Iniciar execução.

3. Na caixa de diálogo Iniciar execução, faça o seguinte:

   1. (Opcional) Para identificar a execução, insira um nome para ela no campo Nome. Por padrão, o Step Functions cria automaticamente um nome de execução exclusivo.

      > **Note**
      >
      > Step Functions permite criar nomes para máquinas de estado, execuções e atividades, além de rótulos que contenham caracteres não ASCII. Esses nomes não ASCII não funcionam com a Amazon. CloudWatch Para garantir que você possa acompanhar CloudWatch as métricas, escolha um nome que use somente caracteres ASCII.

   2. (Opcional) No campo Entrada, insira os valores de entrada no formato JSON para executar o fluxo de trabalho.

      Se você optar por Executar uma demonstração, não precisará fornecer nenhuma entrada de execução.

      > **Note**
      >
      > Se o projeto de demonstração que você implementou contiver dados de entrada de execução pré-preenchidos, use essa entrada para executar a máquina de estado.

   3. Selecione Iniciar execução.
   4. O console do Step Functions direciona você para uma página em que o título é o ID da execução. Essa página é conhecida como página de Detalhes da execução. Nesta página, você pode revisar os resultados da execução à medida que a execução avança ou após a conclusão.

      Para revisar os resultados da execução, escolha estados individuais na Exibição em gráfico e, em seguida, escolha as guias individuais no painel Detalhes da etapa para visualizar os detalhes de cada estado, incluindo entrada, saída e definição, respectivamente. Para obter detalhes sobre as informações de execução que você pode visualizar na página Detalhes da execução, consulte Página de Detalhes da execução — Visão geral da interface.

Exemplo de código da máquina de estado

A máquina de estado neste projeto de amostra se integra EventBridge ao enviar um evento personalizado para um barramento de EventBridge eventos. O evento enviado para o barramento de eventos corresponde a uma EventBridge regra que aciona uma função Lambda que envia mensagens para um tópico do Amazon SNS e uma fila do Amazon SQS.

Navegue por este exemplo de máquina de estado para ver como o Step Functions gerencia EventBridge.

Para obter mais informações sobre como AWS Step Functions controlar outros AWS serviços, consulteUsando AWS Step Functions com outros serviços.

```
{
  "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 obter informações sobre como configurar o IAM ao usar Step Functions com outros AWS serviços, consultePolíticas do IAM para serviços integrados.

Exemplo do IAM

Essas políticas de exemplo AWS Identity and Access Management (IAM) geradas pelo projeto de amostra incluem o menor privilégio necessário para executar a máquina de estado e os recursos relacionados. Recomendamos que você inclua apenas as permissões que forem necessárias em suas políticas do 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 obter informações sobre como configurar o IAM ao usar Step Functions com outros AWS serviços, consulte Políticas do IAM para serviços integrados.

Invocar fluxos de trabalho expresso síncronos

Este exemplo de projeto demonstra como invocar fluxos de trabalho expresso síncronos por meio do Amazon API Gateway para gerenciar um banco de dados de funcionários.

Neste projeto, o Step Functions usa endpoints do API Gateway para iniciar os fluxos de trabalho expresso síncronos do Step Functions. Em seguida, eles usam o DynamoDB para pesquisar, adicionar e remover funcionários em um banco de dados de funcionários.

Para obter informações sobre fluxos de trabalho expresso síncronos no Step Functions, consulte Fluxos de trabalho expresso síncronos e assíncronos.

> **Note**
>
> Este projeto de exemplo pode incorrer em cobranças.

Para novos AWS usuários, um nível de uso gratuito está disponível. Neste nível, os serviços são gratuitos abaixo de um determinado nível de uso. Para obter mais informações sobre AWS custos e o nível gratuito, consulte Step Functions Pricing.

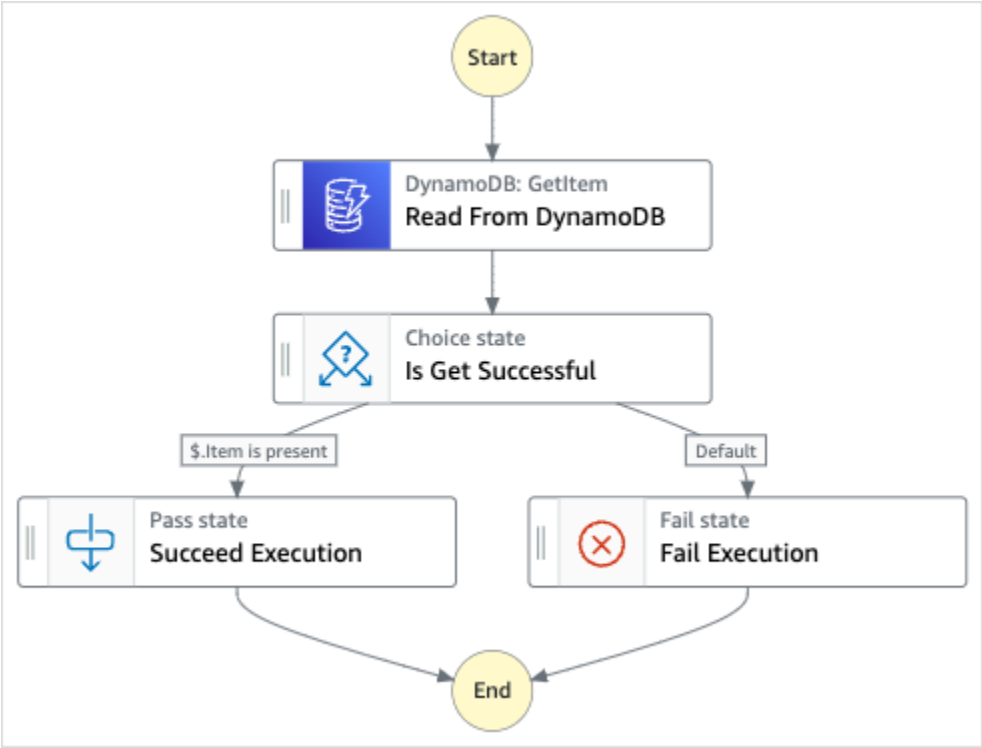
Etapa 1: Criar a máquina de estado e provisionar os recursos

1. Abra o console do Step Functions e clique em Criar máquina de estado.
2. Digite **`Invoke Synchronous Express Workflows through API Gateway`** na caixa de pesquisa e escolha Invocar fluxos de trabalho expresso síncronos por meio do API Gateway entre os resultados de pesquisa que são retornados.
3. Escolha Próximo para continuar.
4. Step Functions lista o Serviços da AWS usado no projeto de amostra que você selecionou. Também mostra um gráfico de fluxo de trabalho para o projeto de amostra. Implante esse projeto no seu Conta da AWS ou use-o como ponto de partida para criar seus próprios projetos. Com base em como você deseja prosseguir, escolha Executar uma demonstração ou Criar com base nela.

   Este projeto de exemplo implementa os recursos a seguir.

   - Uma API Amazon API Gateway HTTPS que é chamada por uma máquina de estado.
   - Uma tabela do Amazon DynamoDB.
   - Três máquinas de AWS Step Functions estado.
   - Funções relacionadas AWS Identity and Access Management (IAM).

   A imagem a seguir mostra o gráfico do fluxo de trabalho para o exemplo de projeto Invocar fluxos de trabalho expresso síncronos por meio do API Gateway:



5. Escolha Usar modelo para continuar com a seleção.
6. Execute um destes procedimentos:

   - Se você selecionou Criar com base nela, o Step Functions criará o protótipo do fluxo de trabalho para o projeto de amostra selecionado. O Step Functions não implanta os recursos listados na definição do fluxo de trabalho.

     No Modo de design do Workflow Studio, arraste e solte os estados do Navegador de estados para continuar criando seu protótipo de fluxo de trabalho. Ou mude para o Modo de código que fornece um editor de código integrado semelhante ao VS Code para atualizar a definição Amazon States Language (ASL) de sua máquina de estado no console Step Functions. Para obter mais informações sobre o uso do Workflow Studio para criar suas máquinas de estado, consulte Como usar o Workflow Studio.

     > **Important**
     >
     > Lembre-se de atualizar o espaço reservado do nome do recurso da Amazon (ARN) para os recursos usados no projeto de amostra antes de executar o fluxo de trabalho.

- Se você selecionou Executar uma demonstração, o Step Functions cria um projeto de amostra somente para leitura que usa um AWS CloudFormation modelo para implantar os AWS recursos listados nesse modelo no seu. Conta da AWS

  > **Tip**
  >
  > Para visualizar a definição da máquina de estado do projeto de amostra, escolha Código.

  Quando estiver pronto, escolha Implemente e execute para implantar o projeto de amostra e criar os recursos.

  Pode levar até 10 minutos para que esses recursos e as permissões relacionadas ao IAM sejam criados. Enquanto seus recursos estão sendo implantados, você pode abrir o link do CloudFormation Stack ID para ver quais recursos estão sendo provisionados.

  Depois que todos os recursos do projeto de exemplo forem criados, você poderá ver o novo projeto de exemplo listado na página Máquinas de estado.

  > **Important**
  >
  > Taxas padrão podem ser aplicadas a cada serviço usado no CloudFormation modelo.

Etapa 2: Executar a máquina de estado

1. Na página Máquinas de estado, escolha seu projeto de exemplo.
2. Na página do projeto de exemplo, escolha Iniciar execução.
3. Na caixa de diálogo Iniciar execução, faça o seguinte:

   1. (Opcional) Para identificar a execução, insira um nome para ela no campo Nome. Por padrão, o Step Functions cria automaticamente um nome de execução exclusivo.

      > **Note**
      >
      > Step Functions permite criar nomes para máquinas de estado, execuções e atividades, além de rótulos que contenham caracteres não ASCII. Esses nomes não

ASCII não funcionam com a Amazon. CloudWatch Para garantir que você possa acompanhar CloudWatch as métricas, escolha um nome que use somente caracteres ASCII.

2. (Opcional) No campo Entrada, insira os valores de entrada no formato JSON para executar o fluxo de trabalho.

   Se você optar por Executar uma demonstração, não precisará fornecer nenhuma entrada de execução.

   > **Note**
   >
   > Se o projeto de demonstração que você implementou contiver dados de entrada de execução pré-preenchidos, use essa entrada para executar a máquina de estado.

3. Selecione Iniciar execução.
4. O console do Step Functions direciona você para uma página em que o título é o ID da execução. Essa página é conhecida como página de Detalhes da execução. Nesta página, você pode revisar os resultados da execução à medida que a execução avança ou após a conclusão.

   Para revisar os resultados da execução, escolha estados individuais na Exibição em gráfico e, em seguida, escolha as guias individuais no painel Detalhes da etapa para visualizar os detalhes de cada estado, incluindo entrada, saída e definição, respectivamente. Para obter detalhes sobre as informações de execução que você pode visualizar na página Detalhes da execução, consulte Página de Detalhes da execução — Visão geral da interface.

Exemplo de código da máquina de estado

A máquina de estado neste exemplo de projeto se integra ao API Gateway e ao DynamoDB usando o API Gateway para invocar um fluxo de trabalho expresso síncrono, que então atualiza ou lê do banco de dados de funcionários usando o DynamoDB.

Navegue por essa máquina de estado de exemplo para ver como o Step Functions lê o DynamoDB para recuperar as informações dos funcionários.

Para entender mais sobre como invocar o Step Functions usando o API Gateway, consulte o seguinte:

- Chame o API Gateway com o Step Functions
- Como invocar um Gateway privado no Guia do desenvolvedor do API Gateway.

Para obter mais informações sobre como AWS Step Functions controlar outros AWS serviços, consulteUsando AWS Step Functions com outros serviços.

```
{
  "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 obter informações sobre como configurar o IAM ao usar Step Functions com outros AWS serviços, consulte[Políticas do IAM para serviços integrados](#).

Exemplos do IAM

Essas políticas de exemplo AWS Identity and Access Management (IAM) geradas pelo projeto de amostra incluem o menor privilégio necessário para executar a máquina de estado e os recursos relacionados. Recomendamos que você inclua apenas as permissões que forem necessárias em suas políticas do 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 obter informações sobre como configurar o IAM ao usar Step Functions com outros AWS serviços, consulte[Políticas do IAM para serviços integrados](#).

Executar fluxos de trabalho ETL/ELT usando o Amazon Redshift (Lambda, API de dados do Amazon Redshift)

Este projeto de exemplo demonstra como usar o Step Functions e a API de dados do Amazon Redshift para executar um fluxo de trabalho ETL/ELT que carrega dados no data warehouse do Amazon Redshift.

Neste projeto, o Step Functions usa uma AWS Lambda função e a API Amazon Redshift Data para criar os objetos de banco de dados necessários e gerar um conjunto de dados de exemplo e, em seguida, executa dois trabalhos em paralelo que executam o carregamento de tabelas de dimensões, seguidos por uma tabela de fatos. Quando as duas tarefas de carregamento de dimensões terminam com sucesso, o Step Functions executa o trabalho de carregamento da tabela de fatos, executa a tarefa de validação e, em seguida, pausa o cluster do Amazon Redshift.

> **Note**
>
> Você pode modificar a lógica do ETL para receber dados de outras fontes, como o Amazon S3, que pode usar o comando COPY para copiar dados do Amazon S3 para uma tabela do Amazon Redshift.

Para obter mais informações sobre integrações de serviços do Amazon Redshift e do Step Functions, consulte o seguinte:

- Usando AWS Step Functions com outros serviços
- Usar a API de dados do Amazon Redshift
- Serviço de API de dados do Amazon Redshift
- Como criar uma máquina de estado Step Functions que usa Lambda
- Políticas do IAM para:
  - Políticas do IAM para AWS Lambda
  - Autorizar acesso à API de dados do Amazon Redshift

> **Note**
>
> Este projeto de exemplo pode incorrer em cobranças.
> Para novos AWS usuários, um nível de uso gratuito está disponível. Neste nível, os serviços são gratuitos abaixo de um determinado nível de uso. Para obter mais informações sobre AWS custos e o nível gratuito, consulte AWS Step Functions preços.

Etapa 1: Criar a máquina de estado e provisionar os recursos

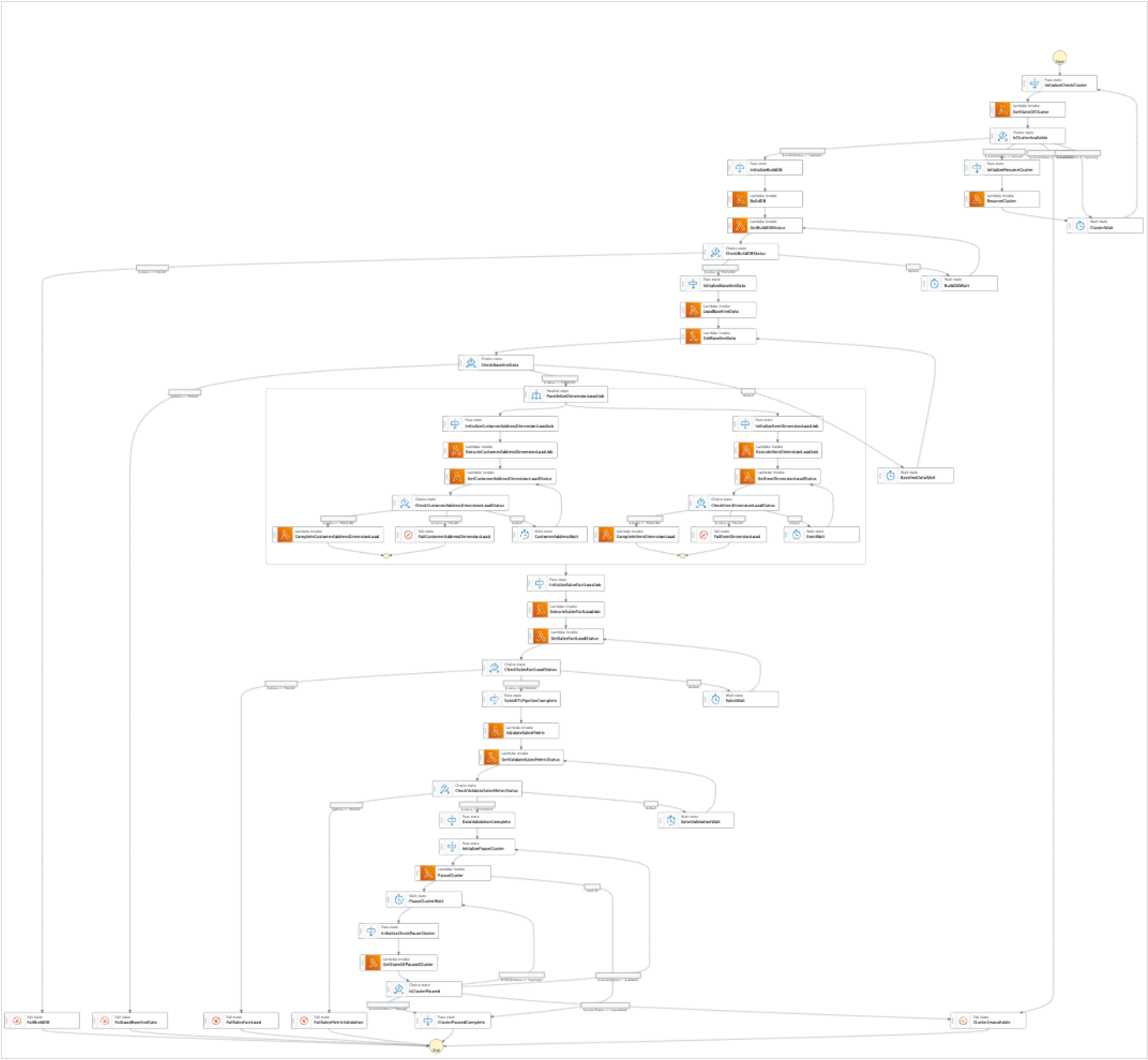
1. Abra o console do Step Functions e clique em Criar máquina de estado.
2. Digite **`ETL job in Amazon Redshift`** no campo de pesquisa e escolha Tarefa ETL no Amazon Redshift a partir dos resultados de pesquisa que são retornados.
3. Escolha Próximo para continuar.
4. Step Functions lista o Serviços da AWS usado no projeto de amostra que você selecionou. Também mostra um gráfico de fluxo de trabalho para o projeto de amostra. Implante esse projeto no seu Conta da AWS ou use-o como ponto de partida para criar seus próprios projetos.

Com base em como você deseja prosseguir, escolha Executar uma demonstração ou Criar com base nela.

Este projeto de exemplo implementa os recursos a seguir.

- Um cluster do Amazon Redshift
- Duas funções do Lambda
- Um esquema do Amazon Redshift
- Cinco tabelas do Amazon Redshift
- Uma máquina de AWS Step Functions estado
- Funções relacionadas AWS Identity and Access Management (IAM).

A imagem a seguir mostra o gráfico do fluxo de trabalho do projeto de exemplo para Tarefa ETL no Amazon Redshift:



5. Escolha Usar modelo para continuar com a seleção.
6. Execute um destes procedimentos:

   - Se você selecionou Criar com base nela, o Step Functions criará o protótipo do fluxo de trabalho para o projeto de amostra selecionado. O Step Functions não implanta os recursos listados na definição do fluxo de trabalho.

     No Modo de design do Workflow Studio, arraste e solte os estados do Navegador de estados para continuar criando seu protótipo de fluxo de trabalho. Ou mude para o Modo de código que fornece um editor de código integrado semelhante ao VS Code para atualizar a definição

Amazon States Language (ASL) de sua máquina de estado no console Step Functions. Para obter mais informações sobre o uso do Workflow Studio para criar suas máquinas de estado, consulte Como usar o Workflow Studio.

> **Important**
>
> Lembre-se de atualizar o espaço reservado do nome do recurso da Amazon (ARN) para os recursos usados no projeto de amostra antes de executar o fluxo de trabalho.

- Se você selecionou Executar uma demonstração, o Step Functions cria um projeto de amostra somente para leitura que usa um AWS CloudFormation modelo para implantar os AWS recursos listados nesse modelo no seu. Conta da AWS

  > **Tip**
  >
  > Para visualizar a definição da máquina de estado do projeto de amostra, escolha Código.

  Quando estiver pronto, escolha Implemente e execute para implantar o projeto de amostra e criar os recursos.

  Pode levar até 10 minutos para que esses recursos e as permissões relacionadas ao IAM sejam criados. Enquanto seus recursos estão sendo implantados, você pode abrir o link do CloudFormation Stack ID para ver quais recursos estão sendo provisionados.

  Depois que todos os recursos do projeto de exemplo forem criados, você poderá ver o novo projeto de exemplo listado na página Máquinas de estado.

  > **Important**
  >
  > Taxas padrão podem ser aplicadas a cada serviço usado no CloudFormation modelo.

Etapa 2: Executar a máquina de estado

1. Na página Máquinas de estado, escolha seu projeto de exemplo.
2. Na página do projeto de exemplo, escolha Iniciar execução.

3. Na caixa de diálogo Iniciar execução, faça o seguinte:

   1. (Opcional) Para identificar a execução, insira um nome para ela no campo Nome. Por padrão, o Step Functions cria automaticamente um nome de execução exclusivo.

      > **Note**
      >
      > Step Functions permite criar nomes para máquinas de estado, execuções e atividades, além de rótulos que contenham caracteres não ASCII. Esses nomes não ASCII não funcionam com a Amazon. CloudWatch Para garantir que você possa acompanhar CloudWatch as métricas, escolha um nome que use somente caracteres ASCII.

   2. (Opcional) No campo Entrada, insira os valores de entrada no formato JSON para executar o fluxo de trabalho.

      Se você optar por Executar uma demonstração, não precisará fornecer nenhuma entrada de execução.

      > **Note**
      >
      > Se o projeto de demonstração que você implementou contiver dados de entrada de execução pré-preenchidos, use essa entrada para executar a máquina de estado.

   3. Selecione Iniciar execução.
   4. O console do Step Functions direciona você para uma página em que o título é o ID da execução. Essa página é conhecida como página de Detalhes da execução. Nesta página, você pode revisar os resultados da execução à medida que a execução avança ou após a conclusão.

      Para revisar os resultados da execução, escolha estados individuais na Exibição em gráfico e, em seguida, escolha as guias individuais no painel Detalhes da etapa para visualizar os detalhes de cada estado, incluindo entrada, saída e definição, respectivamente. Para obter detalhes sobre as informações de execução que você pode visualizar na página Detalhes da execução, consulte Página de Detalhes da execução — Visão geral da interface.

Exemplo de código da máquina de estado

A máquina de estado neste projeto de amostra se integra AWS Lambda transmitindo a lógica ETL InputPath diretamente para esses recursos e sendo executada de forma assíncrona usando a API de dados do Amazon Redshift.

Navegue por esse exemplo de máquina de estado para ver como o Step Functions controla AWS Lambda e a API de dados do Amazon Redshift.

Para obter mais informações sobre como AWS Step Functions controlar outros AWS serviços, consulteUsando AWS Step Functions com outros serviços.

```
{
"Comment": "A simple ETL workflow for loading dimension and fact tables",
  "StartAt": "InitializeCheckCluster",
  "States": {
    "InitializeCheckCluster": {
      "Type": "Pass",
      "Next": "GetStateOfCluster",
      "Result": {
        "input": {
          "redshift_cluster_id": "cfn36-redshiftcluster-AKIAI44QH8DHBEXAMPLE",
          "operation": "status"
        }
      }
    },
    "GetStateOfCluster": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftOperations-AKIAIOSFODNN7EXAMPLE",
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
      "Next": "IsClusterAvailable",
      "InputPath": "$",
      "ResultPath": "$.clusterStatus"
    },
    "IsClusterAvailable": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.clusterStatus",
          "StringEquals": "available",
          "Next": "InitializeBuildDB"
```

```
        },
        {
          "Variable": "$.clusterStatus",
          "StringEquals": "paused",
          "Next": "InitializeResumeCluster"
        },
        {
          "Variable": "$.clusterStatus",
          "StringEquals": "unavailable",
          "Next": "ClusterUnavailable"
        },
        {
          "Variable": "$.clusterStatus",
          "StringEquals": "resuming",
          "Next": "ClusterWait"
        }
      ]
    },
    "ClusterWait": {
      "Type": "Wait",
      "Seconds": 720,
      "Next": "InitializeCheckCluster"
    },
    "InitializeResumeCluster": {
      "Type": "Pass",
      "Next": "ResumeCluster",
      "Result": {
        "input": {
          "redshift_cluster_id": "cfn36-redshiftcluster-AKIAI44QH8DHBEXAMPLE",
          "operation": "resume"
        }
      }
    },
    "ResumeCluster": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftOperations-AKIAIOSFODNN7EXAMPLE",
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
      "Next": "ClusterWait",
      "InputPath": "$",
      "ResultPath": "$"
    },
    "InitializeBuildDB": {
```

```
      "Type": "Pass",
      "Next": "BuildDB",
      "Result": {
        "input": {
          "redshift_cluster_id": "cfn36-redshiftcluster-AKIAI44QH8DHBEXAMPLE",
          "redshift_database": "dev",
          "redshift_user": "awsuser",
          "redshift_schema": "tpcds",
          "action": "build_database",
          "sql_statement": [
            "create schema if not exists {0} authorization {1};",
            "create table if not exists {0}.customer",
            "(c_customer_sk          int4     not null encode az64",
            ",c_customer_id          char(16) not null encode zstd",
            ",c_current_addr_sk      int4              encode az64",
            ",c_first_name           char(20)          encode zstd",
            ",c_last_name            char(30)          encode zstd",
            ",primary key (c_customer_sk)",
            ") distkey(c_customer_sk);",
            "--",
            "create table if not exists {0}.customer_address",
            "(ca_address_sk    int4     not null encode az64",
            ",ca_address_id    char(16) not null encode zstd",
            ",ca_state         char(2)           encode zstd",
            ",ca_zip           char(10)          encode zstd",
            ",ca_country       varchar(20)       encode zstd",
            ",primary key (ca_address_sk)",
            ") distkey(ca_address_sk);",
            "--",
            "create table if not exists {0}.date_dim",
            "(d_date_sk           integer  not null encode az64",
            ",d_date_id           char(16) not null encode zstd",
            ",d_date              date              encode az64",
            ",d_day_name          char(9)           encode zstd",
            ",primary key (d_date_sk)",
            ") diststyle all;",
            "--",
            "create table if not exists {0}.item",
            "(i_item_sk        int4     not null encode az64",
            ",i_item_id        char(16) not null encode zstd",
            ",i_rec_start_date date              encode az64",
            ",i_rec_end_date   date              encode az64",
            ",i_current_price  numeric(7,2)      encode az64",
            ",i_category       char(50)          encode zstd",
```

```
            ",i_product_name   char(50)          encode zstd",
            ",primary key (i_item_sk)",
            ") distkey(i_item_sk) sortkey(i_category);",
            "--",
            "create table if not exists {0}.store_sales",
            "(ss_sold_date_sk       int4",
            ",ss_item_sk            int4 not null encode az64",
            ",ss_customer_sk        int4          encode az64",
            ",ss_addr_sk            int4          encode az64",
            ",ss_store_sk           int4          encode az64",
            ",ss_ticket_number      int8 not null encode az64",
            ",ss_quantity           int4          encode az64",
            ",ss_net_paid           numeric(7,2)  encode az64",
            ",ss_net_profit         numeric(7,2)  encode az64",
            ",primary key (ss_item_sk, ss_ticket_number)",
            ") distkey(ss_item_sk) sortkey(ss_sold_date_sk);"
          ]
        }
      }
    },
    "BuildDB": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
      "Next": "GetBuildDBStatus",
      "InputPath": "$",
      "ResultPath": "$"
    },
    "GetBuildDBStatus": {
      "Type": "Task",
      "Next": "CheckBuildDBStatus",
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
      "InputPath": "$",
      "ResultPath": "$.status"
    },
    "CheckBuildDBStatus": {
      "Type": "Choice",
      "Choices": [
        {
```

```
          "Variable": "$.status",
          "StringEquals": "FAILED",
          "Next": "FailBuildDB"
        },
        {
          "Variable": "$.status",
          "StringEquals": "FINISHED",
          "Next": "InitializeBaselineData"
        }
      ],
      "Default": "BuildDBWait"
    },
    "BuildDBWait": {
      "Type": "Wait",
      "Seconds": 15,
      "Next": "GetBuildDBStatus"
    },
    "FailBuildDB": {
      "Type": "Fail",
      "Cause": "Database Build Failed",
      "Error": "Error"
    },
    "InitializeBaselineData": {
      "Type": "Pass",
      "Next": "LoadBaselineData",
      "Result": {
        "input": {
          "redshift_cluster_id": "cfn36-redshiftcluster-AKIAI44QH8DHBEXAMPLE",
          "redshift_database": "dev",
          "redshift_user": "awsuser",
          "redshift_schema": "tpcds",
          "action": "load_baseline_data",
          "sql_statement": [
            "begin transaction;",
            "truncate table {0}.customer;",
            "insert into {0}.customer
  (c_customer_sk,c_customer_id,c_current_addr_sk,c_first_name,c_last_name)",
            "values",
            "(7550,'AAAAAAAAOHNBAAAA',9264662,'Michelle','Deaton'),",
            "(37079,'AAAAAAAAHNAJAAAA',13971208,'Michael','Simms'),",
            "(40626,'AAAAAAAACLOJAAAA',1959255,'Susan','Ryder'),",
            "(2142876,'AAAAAAAAMJCLACAA',7644556,'Justin','Brown');",
            "analyze {0}.customer;",
            "--",
```

```
            "truncate table {0}.customer_address;",
            "insert into {0}.customer_address
(ca_address_sk,ca_address_id,ca_state,ca_zip,ca_country)",
            "values",
            "(13971208,'AAAAAAAAIAPCFNAA','NE','63451','United States'),",
            "(7644556,'AAAAAAAAMIFKEHAA','SD','58883','United States'),",
            "(9264662,'AAAAAAAAGBOFNIAA','CA','99310','United States');",
            "analyze {0}.customer_address;",
            "--",
            "truncate table {0}.item;",
            "insert into {0}.item
(i_item_sk,i_item_id,i_rec_start_date,i_rec_end_date,i_current_price,i_category,i_product_name
            "values",

"(3417,'AAAAAAAAIFNAAAAA','1997-10-27',NULL,14.29,'Electronics','ationoughtesepri
'),",
            "(9615,'AAAAAAAAOIFCAAAA','1997-10-27',NULL,9.68,'Home','antioughtcallyn
st'),",
            "(3693,'AAAAAAAAMGOAAAAA','2001-03-12',NULL,2.10,'Men','prin
stcallypri'),",

"(3630,'AAAAAAAAMCOAAAAA','2001-10-27',NULL,2.95,'Electronics','barpricallypri'),",

"(16506,'AAAAAAAAIHAEAAAA','2001-10-27',NULL,3.85,'Home','callybaranticallyought'),",

"(7866,'AAAAAAAAILOBAAAA','2001-10-27',NULL,12.60,'Jewelry','callycallyeingation');",
            "--",
            "analyze {0}.item;",
            "truncate table {0}.date_dim;",
            "insert into {0}.date_dim (d_date_sk,d_date_id,d_date,d_day_name)",
            "values",
            "(2450521,'AAAAAAAAJFEGFCAA','1997-03-13','Thursday'),",
            "(2450749,'AAAAAAAANDFGFCAA','1997-10-27','Monday'),",
            "(2451251,'AAAAAAAADDHGFCAA','1999-03-13','Saturday'),",
            "(2451252,'AAAAAAAAEDHGFCAA','1999-03-14','Sunday'),",
            "(2451981,'AAAAAAAANAKGFCAA','2001-03-12','Monday'),",
            "(2451982,'AAAAAAAAOAKGFCAA','2001-03-13','Tuesday'),",
            "(2452210,'AAAAAAAACPKGFCAA','2001-10-27','Saturday'),",
            "(2452641,'AAAAAAAABKMGFCAA','2003-01-01','Wednesday'),",
            "(2452642,'AAAAAAAACKMGFCAA','2003-01-02','Thursday');",
            "--",
            "analyze {0}.date_dim;",
            "--  commit and End transaction",
            "commit;",
```

```
            "end transaction;"
          ]
        }
      }
    },
    "LoadBaselineData": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
      "Next": "GetBaselineData",
      "InputPath": "$",
      "ResultPath": "$"
    },
    "GetBaselineData": {
      "Type": "Task",
      "Next": "CheckBaselineData",
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
      "InputPath": "$",
      "ResultPath": "$.status"
    },
    "CheckBaselineData": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.status",
          "StringEquals": "FAILED",
          "Next": "FailLoadBaselineData"
        },
        {
          "Variable": "$.status",
          "StringEquals": "FINISHED",
          "Next": "ParallelizeDimensionLoadJob"
        }
      ],
      "Default": "BaselineDataWait"
    },
    "BaselineDataWait": {
      "Type": "Wait",
      "Seconds": 20,
```

```
      "Next": "GetBaselineData"
    },
    "FailLoadBaselineData": {
      "Type": "Fail",
      "Cause": "Load Baseline Data Failed",
      "Error": "Error"
    },
    "ParallelizeDimensionLoadJob": {
      "Type": "Parallel",
      "Next": "InitializeSalesFactLoadJob",
      "ResultPath": "$.status",
      "Branches": [
        {
          "StartAt": "InitializeCustomerAddressDimensionLoadJob",
          "States": {
            "InitializeCustomerAddressDimensionLoadJob": {
              "Type": "Pass",
              "Next": "ExecuteCustomerAddressDimensionLoadJob",
              "Result": {
                "input": {
                  "redshift_cluster_id": "cfn36-redshiftcluster-AKIAI44QH8DHBEXAMPLE",
                  "redshift_database": "dev",
                  "redshift_user": "awsuser",
                  "redshift_schema": "tpcds",
                  "action": "load_customer_address",
                  "sql_statement": [
                    "begin transaction;",
                    "/* Create a staging table to hold the input data. Staging table is
created with BACKUP NO option for faster inserts and also data temporary */",
                    "drop table if exists {0}.stg_customer_address;",
                    "create table if not exists {0}.stg_customer_address",
                    "(ca_address_id    varchar(16)  encode zstd",
                    ",ca_state         varchar(2)   encode zstd",
                    ",ca_zip           varchar(10)  encode zstd",
                    ",ca_country       varchar(20)  encode zstd",
                    ")",
                    "backup no",
                    "diststyle even;",
                    "/* Ingest data from source */",
                    "insert into {0}.stg_customer_address
(ca_address_id,ca_state,ca_zip,ca_country)",
                    "values",
                    "('AAAAAAAACFBBAAAA','NE','','United States'),",
                    "('AAAAAAAAGAEFAAAA','NE','61749','United States'),",
```

```
                    "('AAAAAAAAPJKKAAAA','OK','','United States'),",
                    "('AAAAAAAAMIHGAAAA','AL','','United States');",
                    "/* Perform UPDATE for existing data with refreshed attribute
 values */",
                    "update {0}.customer_address",
                    "   set ca_state = stg_customer_address.ca_state,",
                    "       ca_zip = stg_customer_address.ca_zip,",
                    "       ca_country = stg_customer_address.ca_country",
                    "  from {0}.stg_customer_address",
                    " where customer_address.ca_address_id =
 stg_customer_address.ca_address_id;",
                    "/* Perform insert for new rows  */",
                    "insert into {0}.customer_address",
                    "(ca_address_sk",
                    ",ca_address_id",
                    ",ca_state",
                    ",ca_zip",
                    ",ca_country",
                    ")",
                    "with max_customer_address_sk as",
                    "(select max(ca_address_sk) max_ca_address_sk",
                    "from {0}.customer_address)",
                    "select row_number() over (order by
 stg_customer_address.ca_address_id) + max_customer_address_sk.max_ca_address_sk as
 ca_address_sk",
                    ",stg_customer_address.ca_address_id",
                    ",stg_customer_address.ca_state",
                    ",stg_customer_address.ca_zip",
                    ",stg_customer_address.ca_country",
                    "from {0}.stg_customer_address,",
                    "max_customer_address_sk",
                    "where stg_customer_address.ca_address_id not in (select
 customer_address.ca_address_id from {0}.customer_address);",
                    "/* Commit and End transaction */",
                    "commit;",
                    "end transaction;"
                ]
              }
            }
          },
          "ExecuteCustomerAddressDimensionLoadJob": {
            "Type": "Task",
            "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
```

```
              "TimeoutSeconds": 180,
              "HeartbeatSeconds": 60,
              "Next": "GetCustomerAddressDimensionLoadStatus",
              "InputPath": "$",
              "ResultPath": "$"
            },
            "GetCustomerAddressDimensionLoadStatus": {
              "Type": "Task",
              "Next": "CheckCustomerAddressDimensionLoadStatus",
              "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
              "TimeoutSeconds": 180,
              "HeartbeatSeconds": 60,
              "InputPath": "$",
              "ResultPath": "$.status"
            },
            "CheckCustomerAddressDimensionLoadStatus": {
              "Type": "Choice",
              "Choices": [
                {
                  "Variable": "$.status",
                  "StringEquals": "FAILED",
                  "Next": "FailCustomerAddressDimensionLoad"
                },
                {
                  "Variable": "$.status",
                  "StringEquals": "FINISHED",
                  "Next": "CompleteCustomerAddressDimensionLoad"
                }
              ],
              "Default": "CustomerAddressWait"
            },
            "CustomerAddressWait": {
              "Type": "Wait",
              "Seconds": 5,
              "Next": "GetCustomerAddressDimensionLoadStatus"
            },
            "CompleteCustomerAddressDimensionLoad": {
              "Type": "Task",
              "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
              "TimeoutSeconds": 180,
              "HeartbeatSeconds": 60,
              "End": true
```

```
            },
            "FailCustomerAddressDimensionLoad": {
              "Type": "Fail",
              "Cause": "ETL Workflow Failed",
              "Error": "Error"
            }
          }
        },
        {
          "StartAt": "InitializeItemDimensionLoadJob",
          "States": {
            "InitializeItemDimensionLoadJob": {
              "Type": "Pass",
              "Next": "ExecuteItemDimensionLoadJob",
              "Result": {
                "input": {
                  "redshift_cluster_id": "cfn36-redshiftcluster-AKIAI44QH8DHBEXAMPLE",
                  "redshift_database": "dev",
                  "redshift_user": "awsuser",
                  "redshift_schema": "tpcds",
                  "action": "load_item",
                  "sql_statement": [
                    "begin transaction;",
                    "/* Create a staging table to hold the input data. Staging table is
created with BACKUP NO option for faster inserts and also data temporary */",
                    "drop table if exists {0}.stg_item;",
                    "create table if not exists {0}.stg_item",
                    "(i_item_id        varchar(16) encode zstd",
                    ",i_rec_start_date date encode zstd",
                    ",i_rec_end_date   date encode zstd",
                    ",i_current_price  numeric(7,2) encode zstd",
                    ",i_category       varchar(50) encode zstd",
                    ",i_product_name   varchar(50) encode zstd",
                    ")",
                    "backup no",
                    "diststyle even;",
                    "/* Ingest data from source */",
                    "insert into {0}.stg_item",

"(i_item_id,i_rec_start_date,i_rec_end_date,i_current_price,i_category,i_product_name)",
                    "values",

"('AAAAAAAAABJBAAAA','2000-10-27',NULL,4.10,'Books','ationoughtesecally'),",
```

```
                    "('AAAAAAAAOPKBAAAA','2001-10-27',NULL,4.22,'Books','ableoughtn
 stcally'),",
                    "('AAAAAAAAHGPAAAAA','1997-10-27',NULL,29.30,'Books','priesen
 stpri'),",

 "('AAAAAAAAICMAAAAA','2001-10-27',NULL,1.93,'Books','eseoughtoughtpri'),",

 "('AAAAAAAAGPGBAAAA','2001-10-27',NULL,9.96,'Books','bareingeinganti'),",
                    "('AAAAAAAAANBEBAAAA','1997-10-27',NULL,2.25,'Music','n
 steseoughtanti'),",

 "('AAAAAAAACLAAAAAA','2001-10-27',NULL,1.71,'Home','bareingought'),",

 "('AAAAAAAAOBBDAAAA','2001-10-27',NULL,5.55,'Books','callyationantiableought');",
                    "/
*****************************************************************************************
                    "** Type 2 is maintained for i_current_price column.",
                    "** Update all attributes for the item when the price is not
 changed",
                    "** Sunset existing active item record with current i_rec_end_date
 and insert a new record when the price does not match",

 "*****************************************************************************************
                    "update {0}.item",
                    "   set i_category = stg_item.i_category,",
                    "       i_product_name = stg_item.i_product_name",
                    "  from {0}.stg_item",
                    " where item.i_item_id = stg_item.i_item_id",
                    "   and item.i_rec_end_date is null",
                    "   and item.i_current_price = stg_item.i_current_price;",
                    "insert into {0}.item",
                    "(i_item_sk",
                    ",i_item_id",
                    ",i_rec_start_date",
                    ",i_rec_end_date",
                    ",i_current_price",
                    ",i_category",
                    ",i_product_name",
                    ")",
                    "with max_item_sk as",
                    "(select max(i_item_sk) max_item_sk",
                    "   from {0}.item)",
                    "select row_number() over (order by stg_item.i_item_id) +
 max_item_sk as i_item_sk",
```

```
                    "      ,stg_item.i_item_id",
                    "      ,trunc(sysdate) as i_rec_start_date",
                    "      ,null as i_rec_end_date",
                    "      ,stg_item.i_current_price",
                    "      ,stg_item.i_category",
                    "      ,stg_item.i_product_name",
                    "  from {0}.stg_item, {0}.item, max_item_sk",
                    " where item.i_item_id = stg_item.i_item_id",
                    "   and item.i_rec_end_date is null",
                    "   and item.i_current_price <> stg_item.i_current_price;",
                    "/* Sunset penultimate records that were inserted as type 2 */",
                    "update {0}.item",
                    "   set i_rec_end_date = trunc(sysdate)",
                    "  from {0}.stg_item",
                    " where item.i_item_id = stg_item.i_item_id",
                    "   and item.i_rec_end_date is null",
                    "   and item.i_current_price <> stg_item.i_current_price;",
                    "/* Commit and End transaction */",
                    "commit;",
                    "end transaction;"
                  ]
                }
              }
            },
            "ExecuteItemDimensionLoadJob": {
              "Type": "Task",
              "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
              "TimeoutSeconds": 180,
              "HeartbeatSeconds": 60,
              "Next": "GetItemDimensionLoadStatus",
              "InputPath": "$",
              "ResultPath": "$"
            },
            "GetItemDimensionLoadStatus": {
              "Type": "Task",
              "Next": "CheckItemDimensionLoadStatus",
              "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
              "TimeoutSeconds": 180,
              "HeartbeatSeconds": 60,
              "InputPath": "$",
              "ResultPath": "$.status"
            },
```

```
            "CheckItemDimensionLoadStatus": {
              "Type": "Choice",
              "Choices": [
                {
                  "Variable": "$.status",
                  "StringEquals": "FAILED",
                  "Next": "FailItemDimensionLoad"
                },
                {
                  "Variable": "$.status",
                  "StringEquals": "FINISHED",
                  "Next": "CompleteItemDimensionLoad"
                }
              ],
              "Default": "ItemWait"
            },
            "ItemWait": {
              "Type": "Wait",
              "Seconds": 5,
              "Next": "GetItemDimensionLoadStatus"
            },
            "CompleteItemDimensionLoad": {
              "Type": "Task",
              "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
              "TimeoutSeconds": 180,
              "HeartbeatSeconds": 60,
              "End": true
            },
            "FailItemDimensionLoad": {
              "Type": "Fail",
              "Cause": "ETL Workflow Failed",
              "Error": "Error"
            }
          }
        }
      ]
    },
    "InitializeSalesFactLoadJob": {
      "Type": "Pass",
      "Next": "ExecuteSalesFactLoadJob",
      "Result": {
        "input": {
          "redshift_cluster_id": "cfn36-redshiftcluster-AKIAI44QH8DHBEXAMPLE",
```

```
          "redshift_database": "dev",
          "redshift_user": "awsuser",
          "redshift_schema": "tpcds",
          "snapshot_date": "2003-01-02",
          "action": "load_sales_fact",
          "sql_statement": [
            "begin transaction;",
            "/* Create a stg_store_sales staging table */",
            "drop table if exists {0}.stg_store_sales;",
            "create table {0}.stg_store_sales",
            "(sold_date             date encode zstd",
            ",i_item_id             varchar(16) encode zstd",
            ",c_customer_id         varchar(16) encode zstd",
            ",ca_address_id         varchar(16) encode zstd",
            ",ss_ticket_number      integer encode zstd",
            ",ss_quantity           integer encode zstd",
            ",ss_net_paid           numeric(7,2) encode zstd",
            ",ss_net_profit         numeric(7,2) encode zstd",
            ")",
            "backup no",
            "diststyle even;",
            "/* Ingest data from source */",
            "insert into {0}.stg_store_sales",

"(sold_date,i_item_id,c_customer_id,ca_address_id,ss_ticket_number,ss_quantity,ss_net_paid,ss_
            "values",

"('2003-01-02','AAAAAAAAIFNAAAAA','AAAAAAAAOHNBAAAA','AAAAAAAAGBOFNIAA',1403191,13,5046.37,150

"('2003-01-02','AAAAAAAAIFNAAAAA','AAAAAAAAOHNBAAAA','AAAAAAAAGBOFNIAA',1403191,13,2103.72,-12

"('2003-01-02','AAAAAAAAILOBAAAA','AAAAAAAAOHNBAAAA','AAAAAAAAGBOFNIAA',1403191,13,959.10,-130

"('2003-01-02','AAAAAAAAILOBAAAA','AAAAAAAAHNAJAAAA','AAAAAAAAIAPCFNAA',1403191,13,962.65,-475

"('2003-01-02','AAAAAAAAMCOAAAAA','AAAAAAAAHNAJAAAA','AAAAAAAAIAPCFNAA',1201746,17,111.60,-241

"('2003-01-02','AAAAAAAAMCOAAAAA','AAAAAAAAHNAJAAAA','AAAAAAAAIAPCFNAA',1201746,17,4013.02,-11

"('2003-01-02','AAAAAAAAMCOAAAAA','AAAAAAAAMJCLACAA','AAAAAAAAMIFKEHAA',1201746,17,2689.12,-55

"('2003-01-02','AAAAAAAAMGOAAAAA','AAAAAAAAMJCLACAA','AAAAAAAAMIFKEHAA',193971,18,1876.89,-556
            "/* Delete any rows from target store_sales for the input date for
idempotency */",
```

```
            "delete from {0}.store_sales where ss_sold_date_sk in (select d_date_sk
 from {0}.date_dim where d_date='{1}');",
            "/* Insert data from staging table to the target table */",
            "insert into {0}.store_sales",
            "(ss_sold_date_sk",
            ",ss_item_sk",
            ",ss_customer_sk",
            ",ss_addr_sk",
            ",ss_ticket_number",
            ",ss_quantity",
            ",ss_net_paid",
            ",ss_net_profit",
            ")",
            "select date_dim.d_date_sk ss_sold_date_sk",
            "      ,item.i_item_sk ss_item_sk",
            "      ,customer.c_customer_sk ss_customer_sk",
            "      ,customer_address.ca_address_sk ss_addr_sk",
            "      ,ss_ticket_number",
            "      ,ss_quantity",
            "      ,ss_net_paid",
            "      ,ss_net_profit",
            "  from {0}.stg_store_sales as store_sales",
            "  inner join {0}.date_dim on store_sales.sold_date = date_dim.d_date",
            "  left join {0}.item on store_sales.i_item_id = item.i_item_id and
 item.i_rec_end_date is null",
            "  left join {0}.customer on store_sales.c_customer_id =
 customer.c_customer_id",
            "  left join {0}.customer_address on store_sales.ca_address_id =
 customer_address.ca_address_id;",
            "/* Drop staging table */",
            "drop table if exists {0}.stg_store_sales;",
            "/* Commit and End transaction */",
            "commit;",
            "end transaction;"
          ]
        }
      }
    },
    "ExecuteSalesFactLoadJob": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
```

```
      "Next": "GetSalesFactLoadStatus",
      "InputPath": "$",
      "ResultPath": "$"
    },
    "GetSalesFactLoadStatus": {
      "Type": "Task",
      "Next": "CheckSalesFactLoadStatus",
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
      "InputPath": "$",
      "ResultPath": "$.status"
    },
    "CheckSalesFactLoadStatus": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.status",
          "StringEquals": "FAILED",
          "Next": "FailSalesFactLoad"
        },
        {
          "Variable": "$.status",
          "StringEquals": "FINISHED",
          "Next": "SalesETLPipelineComplete"
        }
      ],
      "Default": "SalesWait"
    },
    "SalesWait": {
      "Type": "Wait",
      "Seconds": 5,
      "Next": "GetSalesFactLoadStatus"
    },
    "FailSalesFactLoad": {
      "Type": "Fail",
      "Cause": "ETL Workflow Failed",
      "Error": "Error"
    },
    "ClusterUnavailable": {
      "Type": "Fail",
      "Cause": "Redshift cluster is not available",
      "Error": "Error"
```

```
    },
    "SalesETLPipelineComplete": {
      "Type": "Pass",
      "Next": "ValidateSalesMetric",
      "Result": {
        "input": {
          "redshift_cluster_id": "cfn36-redshiftcluster-AKIAI44QH8DHBEXAMPLE",
          "redshift_database": "dev",
          "redshift_user": "awsuser",
          "redshift_schema": "tpcds",
          "snapshot_date": "2003-01-02",
          "action": "validate_sales_metric",
          "sql_statement": [
            "select 1/count(1) from {0}.store_sales where ss_sold_date_sk in (select
 d_date_sk from {0}.date_dim where d_date='{1}')"
          ]
        }
      }
    },
    "ValidateSalesMetric": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
      "Next": "GetValidateSalesMetricStatus",
      "InputPath": "$",
      "ResultPath": "$"
    },
    "GetValidateSalesMetricStatus": {
      "Type": "Task",
      "Next": "CheckValidateSalesMetricStatus",
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
      "InputPath": "$",
      "ResultPath": "$.status"
    },
    "CheckValidateSalesMetricStatus": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.status",
```

```
          "StringEquals": "FAILED",
          "Next": "FailSalesMetricValidation"
        },
        {
          "Variable": "$.status",
          "StringEquals": "FINISHED",
          "Next": "DataValidationComplete"
        }
      ],
      "Default": "SalesValidationWait"
    },
    "SalesValidationWait": {
      "Type": "Wait",
      "Seconds": 5,
      "Next": "GetValidateSalesMetricStatus"
    },
    "FailSalesMetricValidation": {
      "Type": "Fail",
      "Cause": "Data Validation Failed",
      "Error": "Error"
    },
    "DataValidationComplete": {
      "Type": "Pass",
      "Next": "InitializePauseCluster"
    },
    "InitializePauseCluster": {
      "Type": "Pass",
      "Next": "PauseCluster",
      "Result": {
        "input": {
          "redshift_cluster_id": "cfn36-redshiftcluster-AKIAI44QH8DHBEXAMPLE",
          "operation": "pause"
        }
      }
    },
    "PauseCluster": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftOperations-AKIAIOSFODNN7EXAMPLE",
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
      "Next": "PauseClusterWait",
      "InputPath": "$",
      "ResultPath": "$.clusterStatus",
```

```
      "Catch": [
        {
          "ErrorEquals": [
            "States.ALL"
          ],
          "Next": "ClusterPausedComplete"
        }
      ]
    },
    "InitializeCheckPauseCluster": {
      "Type": "Pass",
      "Next": "GetStateOfPausedCluster",
      "Result": {
        "input": {
          "redshift_cluster_id": "cfn36-redshiftcluster-AKIAI44QH8DHBEXAMPLE",
          "operation": "status"
        }
      }
    },
    "GetStateOfPausedCluster": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftOperations-AKIAIOSFODNN7EXAMPLE",
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
      "Next": "IsClusterPaused",
      "InputPath": "$",
      "ResultPath": "$.clusterStatus"
    },
    "IsClusterPaused": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.clusterStatus",
          "StringEquals": "available",
          "Next": "InitializePauseCluster"
        },
        {
          "Variable": "$.clusterStatus",
          "StringEquals": "paused",
          "Next": "ClusterPausedComplete"
        },
        {
          "Variable": "$.clusterStatus",
```

```
          "StringEquals": "unavailable",
          "Next": "ClusterUnavailable"
        },
        {
          "Variable": "$.clusterStatus",
          "StringEquals": "resuming",
          "Next": "PauseClusterWait"
        }
      ]
    },
    "PauseClusterWait": {
      "Type": "Wait",
      "Seconds": 720,
      "Next": "InitializeCheckPauseCluster"
    },
    "ClusterPausedComplete": {
      "Type": "Pass",
      "End": true
    }
  }
}
```

Para obter informações sobre como configurar o IAM ao usar Step Functions com outros AWS serviços, consulte[Políticas do IAM para serviços integrados](#).

Exemplo do IAM

Essas políticas de exemplo AWS Identity and Access Management (IAM) geradas pelo projeto de amostra incluem o menor privilégio necessário para executar a máquina de estado e os recursos relacionados. Recomendamos que você inclua apenas as permissões que forem necessárias em suas políticas do 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 obter informações sobre como configurar o IAM ao usar Step Functions com outros AWS serviços, consulte[Políticas do IAM para serviços integrados](#).

Uso Step Functions e AWS Batch com tratamento de erros

Este projeto de exemplo demonstra como usar Step Functions para executar uma tarefa AWS Batch usando uma máquina de estado com recursos de tratamento de erros.

Nesse projeto, o Step Functions usa uma máquina de estado para chamar o trabalho AWS Batch de maneira síncrona. Em seguida, ele até o êxito ou a falha da tarefa, tenta novamente e detecta erros quando uma tarefa falha e, em seguida, envia um tópico Amazon SNS com uma mensagem sobre o êxito ou a falha da tarefa.

Etapa 1: Criar a máquina de estado e provisionar os recursos

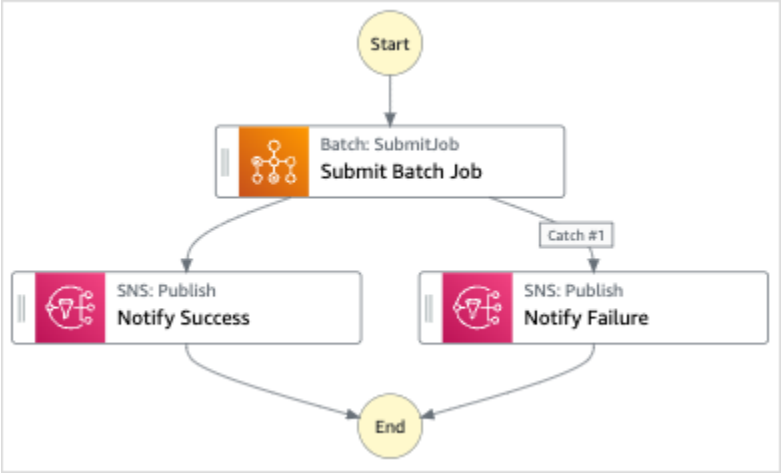
1. Abra o console do Step Functions e clique em Criar máquina de estado.
2. Digite **`Manage a batch job`** na caixa de pesquisa e escolha Gerenciar uma tarefa em lotes nos resultados da pesquisa que são retornados.
3. Escolha Próximo para continuar.
4. Step Functions lista o Serviços da AWS usado no projeto de amostra que você selecionou. Também mostra um gráfico de fluxo de trabalho para o projeto de amostra. Implante esse projeto no seu Conta da AWS ou use-o como ponto de partida para criar seus próprios projetos. Com base em como você deseja prosseguir, escolha Executar uma demonstração ou Criar com base nela.

   Este projeto de exemplo implementa os recursos a seguir.

   - Um AWS Batch emprego
   - Um tópico do Amazon SNS
   - Uma máquina de AWS Step Functions estado

- Funções relacionadas AWS Identity and Access Management (IAM)

A imagem a seguir mostra o gráfico do fluxo de trabalho para o projeto de exemplo Gerenciar uma tarefa em lote:



5. Escolha Usar modelo para continuar com a seleção.
6. Execute um destes procedimentos:

- Se você selecionou Criar com base nela, o Step Functions criará o protótipo do fluxo de trabalho para o projeto de amostra selecionado. O Step Functions não implanta os recursos listados na definição do fluxo de trabalho.

  No Modo de design do Workflow Studio, arraste e solte os estados do Navegador de estados para continuar criando seu protótipo de fluxo de trabalho. Ou mude para o Modo de código que fornece um editor de código integrado semelhante ao VS Code para atualizar a definição Amazon States Language (ASL) de sua máquina de estado no console Step Functions. Para obter mais informações sobre o uso do Workflow Studio para criar suas máquinas de estado, consulte Como usar o Workflow Studio.

  > **Important**
  >
  > Lembre-se de atualizar o espaço reservado do nome do recurso da Amazon (ARN) para os recursos usados no projeto de amostra antes de executar o fluxo de trabalho.

- Se você selecionou Executar uma demonstração, o Step Functions cria um projeto de amostra somente para leitura que usa um AWS CloudFormation modelo para implantar os AWS recursos listados nesse modelo no seu. Conta da AWS

> **Tip**
>
> Para visualizar a definição da máquina de estado do projeto de amostra, escolha Código.

Quando estiver pronto, escolha Implemente e execute para implantar o projeto de amostra e criar os recursos.

Pode levar até 10 minutos para que esses recursos e as permissões relacionadas ao IAM sejam criados. Enquanto seus recursos estão sendo implantados, você pode abrir o link do CloudFormation Stack ID para ver quais recursos estão sendo provisionados.

Depois que todos os recursos do projeto de exemplo forem criados, você poderá ver o novo projeto de exemplo listado na página Máquinas de estado.

> **Important**
>
> Taxas padrão podem ser aplicadas a cada serviço usado no CloudFormation modelo.

Etapa 2: Executar a máquina de estado

1. Na página Máquinas de estado, escolha seu projeto de exemplo.
2. Na página do projeto de exemplo, escolha Iniciar execução.
3. Na caixa de diálogo Iniciar execução, faça o seguinte:

    1. (Opcional) Para identificar a execução, insira um nome para ela no campo Nome. Por padrão, o Step Functions cria automaticamente um nome de execução exclusivo.

        > **Note**
        >
        > Step Functions permite criar nomes para máquinas de estado, execuções e atividades, além de rótulos que contenham caracteres não ASCII. Esses nomes não ASCII não funcionam com a Amazon. CloudWatch Para garantir que você possa

acompanhar CloudWatch as métricas, escolha um nome que use somente caracteres ASCII.

2. (Opcional) No campo Entrada, insira os valores de entrada no formato JSON para executar o fluxo de trabalho.

   Se você optar por Executar uma demonstração, não precisará fornecer nenhuma entrada de execução.

   > **Note**
   >
   > Se o projeto de demonstração que você implementou contiver dados de entrada de execução pré-preenchidos, use essa entrada para executar a máquina de estado.

3. Selecione Iniciar execução.
4. O console do Step Functions direciona você para uma página em que o título é o ID da execução. Essa página é conhecida como página de Detalhes da execução. Nesta página, você pode revisar os resultados da execução à medida que a execução avança ou após a conclusão.

   Para revisar os resultados da execução, escolha estados individuais na Exibição em gráfico e, em seguida, escolha as guias individuais no painel Detalhes da etapa para visualizar os detalhes de cada estado, incluindo entrada, saída e definição, respectivamente. Para obter detalhes sobre as informações de execução que você pode visualizar na página Detalhes da execução, consulte Página de Detalhes da execução — Visão geral da interface.

Exemplo de código da máquina de estado

A máquina de estado neste projeto de amostra se integra AWS Batch ao Amazon SNS passando parâmetros diretamente para esses recursos.

Navegue por este exemplo de máquina de estado para ver como o Step Functions controla AWS Batch o Amazon SNS conectando-se ao Amazon Resource Name (ARN) no `Resource` campo e passando `Parameters` para a API do serviço.

Para obter mais informações sobre como AWS Step Functions controlar outros AWS serviços, consulteUsando AWS Step Functions com outros serviços.

```
{
  "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
    }
  }
}
```

Exemplo do IAM

Esse exemplo de política AWS Identity and Access Management (IAM) gerado pelo projeto de amostra inclui o menor privilégio necessário para executar a máquina de estado e os recursos relacionados. Recomendamos que você inclua apenas as permissões que forem necessárias em suas políticas do 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 obter informações sobre como configurar o IAM ao usar Step Functions com outros AWS serviços, consulte Políticas do IAM para serviços integrados.

Distribuir um AWS Batch emprego

Este exemplo de projeto demonstra como usar o Mapa estado do Step Functions para distribuir AWS Batch trabalhos.

Neste projeto, o Step Functions usa uma máquina de estado para invocar uma função Lambda para fazer um pré-processamento simples e, em seguida, invoca AWS Batch vários trabalhos em paralelo usando o estado. Mapa

Etapa 1: Criar a máquina de estado e provisionar os recursos

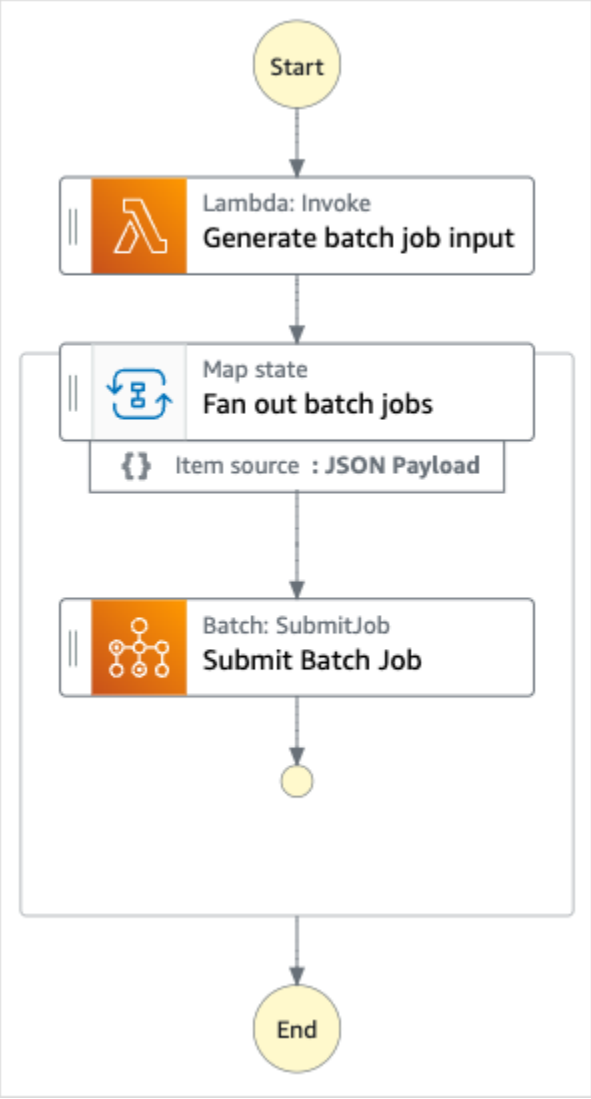
1. Abra o console do Step Functions e clique em Criar máquina de estado.
2. Digite **`Fan out a batch job`** no campo de pesquisa e escolha Disseminar uma tarefa em lotes nos resultados da pesquisa que são retornados.
3. Escolha Próximo para continuar.
4. Step Functions lista o Serviços da AWS usado no projeto de amostra que você selecionou. Também mostra um gráfico de fluxo de trabalho para o projeto de amostra. Implante esse projeto no seu Conta da AWS ou use-o como ponto de partida para criar seus próprios projetos.

Com base em como você deseja prosseguir, escolha Executar uma demonstração ou Criar com base nela.

Este projeto de exemplo implementa os recursos a seguir.

- Uma função do Lambda
- Fila de tarefas AWS Batch
- Uma máquina de AWS Step Functions estado
- Funções relacionadas AWS Identity and Access Management (IAM)

A imagem a seguir mostra o gráfico do fluxo de trabalho do projeto de exemplo Disseminar uma tarefa em lote:



5. Escolha Usar modelo para continuar com a seleção.

6. Execute um destes procedimentos:

   - Se você selecionou Criar com base nela, o Step Functions criará o protótipo do fluxo de trabalho para o projeto de amostra selecionado. O Step Functions não implanta os recursos listados na definição do fluxo de trabalho.

     No Modo de design do Workflow Studio, arraste e solte os estados do Navegador de estados para continuar criando seu protótipo de fluxo de trabalho. Ou mude para o Modo de código que fornece um editor de código integrado semelhante ao VS Code para atualizar a definição Amazon States Language (ASL) de sua máquina de estado no console Step Functions. Para obter mais informações sobre o uso do Workflow Studio para criar suas máquinas de estado, consulte Como usar o Workflow Studio.

     > **Important**
     >
     > Lembre-se de atualizar o espaço reservado do nome do recurso da Amazon (ARN) para os recursos usados no projeto de amostra antes de executar o fluxo de trabalho.

   - Se você selecionou Executar uma demonstração, o Step Functions cria um projeto de amostra somente para leitura que usa um AWS CloudFormation modelo para implantar os AWS recursos listados nesse modelo no seu. Conta da AWS

     > **Tip**
     >
     > Para visualizar a definição da máquina de estado do projeto de amostra, escolha Código.

     Quando estiver pronto, escolha Implemente e execute para implantar o projeto de amostra e criar os recursos.

     Pode levar até 10 minutos para que esses recursos e as permissões relacionadas ao IAM sejam criados. Enquanto seus recursos estão sendo implantados, você pode abrir o link do CloudFormation Stack ID para ver quais recursos estão sendo provisionados.

     Depois que todos os recursos do projeto de exemplo forem criados, você poderá ver o novo projeto de exemplo listado na página Máquinas de estado.

> **Important**
>
> Taxas padrão podem ser aplicadas a cada serviço usado no CloudFormation modelo.

Etapa 2: Executar a máquina de estado

1. Na página Máquinas de estado, escolha seu projeto de exemplo.
2. Na página do projeto de exemplo, escolha Iniciar execução.
3. Na caixa de diálogo Iniciar execução, faça o seguinte:

    1. (Opcional) Para identificar a execução, insira um nome para ela no campo Nome. Por padrão, o Step Functions cria automaticamente um nome de execução exclusivo.

        > **Note**
        >
        > Step Functions permite criar nomes para máquinas de estado, execuções e atividades, além de rótulos que contenham caracteres não ASCII. Esses nomes não ASCII não funcionam com a Amazon. CloudWatch Para garantir que você possa acompanhar CloudWatch as métricas, escolha um nome que use somente caracteres ASCII.

    2. (Opcional) No campo Entrada, insira os valores de entrada no formato JSON para executar o fluxo de trabalho.

        Se você optar por Executar uma demonstração, não precisará fornecer nenhuma entrada de execução.

        > **Note**
        >
        > Se o projeto de demonstração que você implementou contiver dados de entrada de execução pré-preenchidos, use essa entrada para executar a máquina de estado.

    3. Selecione Iniciar execução.
    4. O console do Step Functions direciona você para uma página em que o título é o ID da execução. Essa página é conhecida como página de Detalhes da execução. Nesta página,

você pode revisar os resultados da execução à medida que a execução avança ou após a conclusão.

Para revisar os resultados da execução, escolha estados individuais na Exibição em gráfico e, em seguida, escolha as guias individuais no painel Detalhes da etapa para visualizar os detalhes de cada estado, incluindo entrada, saída e definição, respectivamente. Para obter detalhes sobre as informações de execução que você pode visualizar na página Detalhes da execução, consulte Página de Detalhes da execução — Visão geral da interface.

Exemplo de código da máquina de estado

A máquina de estado neste projeto de amostra se integra AWS Batch ao Amazon SNS passando parâmetros diretamente para esses recursos.

Navegue por este exemplo de máquina de estado para ver como o Step Functions controla AWS Batch o Amazon SNS conectando-se ao Amazon Resource Name (ARN) no `Resource` campo e passando `Parameters` para a API do serviço.

Para obter mais informações sobre como AWS Step Functions controlar outros AWS serviços, consulteUsando AWS Step Functions com outros serviços.

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

Exemplo do IAM

Esses exemplos de políticas AWS Identity and Access Management (IAM) gerados pelo projeto de amostra incluem o menor privilégio necessário para executar a máquina de estado e os recursos relacionados. Recomendamos que você inclua apenas as permissões que forem necessárias em suas políticas do 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 obter informações sobre como configurar o IAM ao usar Step Functions com outros AWS serviços, consulte[Políticas do IAM para serviços integrados](#).

AWS Batch com Lambda

Este projeto de amostra demonstra como usar o Step Functions para pré-processar dados com AWS Lambda funções e, em seguida, AWS Batch orquestrar trabalhos.

Neste projeto, o Step Functions usa uma máquina de estado para invocar uma função do Lambda para fazer um pré-processamento simples antes do envio de uma tarefa de AWS Batch . Várias tarefas podem ser invocadas dependendo do resultado ou do sucesso do anterior.

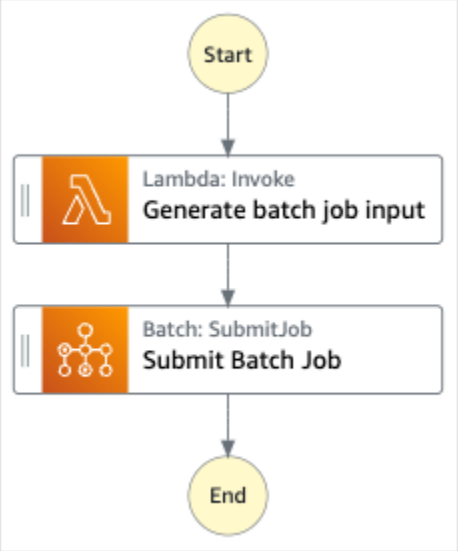
Etapa 1: Criar a máquina de estado e provisionar os recursos

1. Abra o console do Step Functions e clique em Criar máquina de estado.
2. Digite **`Batch job with Lambda`** no campo de pesquisa e escolha Trabalho em lote com Lambda a partir dos resultados de pesquisa que são retornados.
3. Escolha Próximo para continuar.
4. Step Functions lista o Serviços da AWS usado no projeto de amostra que você selecionou. Também mostra um gráfico de fluxo de trabalho para o projeto de amostra. Implante esse projeto no seu Conta da AWS ou use-o como ponto de partida para criar seus próprios projetos. Com base em como você deseja prosseguir, escolha Executar uma demonstração ou Criar com base nela.

   Este projeto de exemplo implementa os recursos a seguir.

   - Uma função do Lambda
   - Um trabalho do AWS Batch
   - Uma máquina de AWS Step Functions estado
   - Funções relacionadas AWS Identity and Access Management (IAM)

   A imagem a seguir mostra o gráfico do fluxo de trabalho para o projeto de exemplo de Trabalho em lote com Lambda:

5. Escolha Usar modelo para continuar com a seleção.
6. Execute um destes procedimentos:

- Se você selecionou Criar com base nela, o Step Functions criará o protótipo do fluxo de trabalho para o projeto de amostra selecionado. O Step Functions não implanta os recursos listados na definição do fluxo de trabalho.

  No Modo de design do Workflow Studio, arraste e solte os estados do Navegador de estados para continuar criando seu protótipo de fluxo de trabalho. Ou mude para o Modo de código que fornece um editor de código integrado semelhante ao VS Code para atualizar a definição Amazon States Language (ASL) de sua máquina de estado no console Step Functions. Para obter mais informações sobre o uso do Workflow Studio para criar suas máquinas de estado, consulte Como usar o Workflow Studio.

  > **Important**
  >
  > Lembre-se de atualizar o espaço reservado do nome do recurso da Amazon (ARN) para os recursos usados no projeto de amostra antes de executar o fluxo de trabalho.

- Se você selecionou Executar uma demonstração, o Step Functions cria um projeto de amostra somente para leitura que usa um AWS CloudFormation modelo para implantar os AWS recursos listados nesse modelo no seu. Conta da AWS

  > **Tip**
  >
  > Para visualizar a definição da máquina de estado do projeto de amostra, escolha Código.

  Quando estiver pronto, escolha Implemente e execute para implantar o projeto de amostra e criar os recursos.

  Pode levar até 10 minutos para que esses recursos e as permissões relacionadas ao IAM sejam criados. Enquanto seus recursos estão sendo implantados, você pode abrir o link do CloudFormation Stack ID para ver quais recursos estão sendo provisionados.

  Depois que todos os recursos do projeto de exemplo forem criados, você poderá ver o novo projeto de exemplo listado na página Máquinas de estado.

> **Important**
>
> Taxas padrão podem ser aplicadas a cada serviço usado no CloudFormation modelo.

Etapa 2: Executar a máquina de estado

1. Na página Máquinas de estado, escolha seu projeto de exemplo.
2. Na página do projeto de exemplo, escolha Iniciar execução.
3. Na caixa de diálogo Iniciar execução, faça o seguinte:

    1. (Opcional) Para identificar a execução, insira um nome para ela no campo Nome. Por padrão, o Step Functions cria automaticamente um nome de execução exclusivo.

        > **Note**
        >
        > Step Functions permite criar nomes para máquinas de estado, execuções e atividades, além de rótulos que contenham caracteres não ASCII. Esses nomes não ASCII não funcionam com a Amazon. CloudWatch Para garantir que você possa acompanhar CloudWatch as métricas, escolha um nome que use somente caracteres ASCII.

    2. (Opcional) No campo Entrada, insira os valores de entrada no formato JSON para executar o fluxo de trabalho.

        Se você optar por Executar uma demonstração, não precisará fornecer nenhuma entrada de execução.

        > **Note**
        >
        > Se o projeto de demonstração que você implementou contiver dados de entrada de execução pré-preenchidos, use essa entrada para executar a máquina de estado.

    3. Selecione Iniciar execução.
    4. O console do Step Functions direciona você para uma página em que o título é o ID da execução. Essa página é conhecida como página de Detalhes da execução. Nesta página,

você pode revisar os resultados da execução à medida que a execução avança ou após a conclusão.

Para revisar os resultados da execução, escolha estados individuais na Exibição em gráfico e, em seguida, escolha as guias individuais no painel Detalhes da etapa para visualizar os detalhes de cada estado, incluindo entrada, saída e definição, respectivamente. Para obter detalhes sobre as informações de execução que você pode visualizar na página Detalhes da execução, consulte Página de Detalhes da execução — Visão geral da interface.

Exemplo de código da máquina de estado

A máquina de estado neste projeto de amostra se integra AWS Batch ao Amazon SNS passando parâmetros diretamente para esses recursos.

Navegue por este exemplo de máquina de estado para ver como o Step Functions controla AWS Batch o Amazon SNS conectando-se ao Amazon Resource Name (ARN) no `Resource` campo e passando `Parameters` para a API do serviço.

Para obter mais informações sobre como AWS Step Functions controlar outros AWS serviços, consulteUsando AWS Step Functions com outros serviços.

```
{
  "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
    }
  }
}
```

Exemplo do IAM

Esses exemplos de políticas AWS Identity and Access Management (IAM) gerados pelo projeto de amostra incluem o menor privilégio necessário para executar a máquina de estado e os recursos relacionados. Recomendamos que você inclua apenas as permissões que forem necessárias em suas políticas do 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 obter informações sobre como configurar o IAM ao usar Step Functions com outros AWS serviços, consulte[Políticas do IAM para serviços integrados](#).

Realizar o encadeamento de prompts de IA com o Amazon Bedrock

Este exemplo de projeto demonstra como se integrar ao Amazon Bedrock para realizar o encadeamento de prompts de IA. Este exemplo de projeto mostra como criar chatbots de alta qualidade usando o Amazon Bedrock. O projeto encadeia alguns prompts e os resolve na sequência em que são fornecidos. O encadeamento desses prompts aumenta a capacidade do modelo de linguagem utilizado de fornecer uma resposta altamente organizada.

Esse projeto de amostra cria a máquina de estado, os AWS recursos de suporte e configura as permissões relacionadas do IAM. Explore esse exemplo de projeto para saber como usar a integração otimizada do Amazon Bedrock a máquinas de estado do Step Functions ou use-o como ponto de partida para os próprios projetos.

Tópicos

Modelo do AWS CloudFormation e recursos adicionais

Você usa um modelo do CloudFormation para implementar esse projeto de exemplo. Esse modelo cria os seguintes recursos em seu Conta da AWS:

- Uma máquina de estado do Step Functions.
- Função de execução para a máquina de estado. Essa função concede as permissões que sua máquina de estado precisa para acessar outros recursos Serviços da AWS e recursos, como a Amazon Bedrock InvokeModelação.

Pré-requisitos

Este exemplo de projeto usa o grande modelo de linguagem (LLM) Cohere Command. Para executar com êxito este exemplo de projeto, é necessário adicionar acesso a esse LLM pelo console do Amazon Bedrock. Para adicionar o acesso ao modelo, faça o seguinte:

1. Abra o console do Amazon Bedrock.
2. No painel de navegação, selecione Acesso ao modelo.
3. Selecione Gerenciar acesso ao modelo.
4. Marque a caixa de seleção ao lado de Cohere.
5. Escolha Solicitar acesso. O Status de acesso do modelo Cohere é exibido como Acesso concedido.

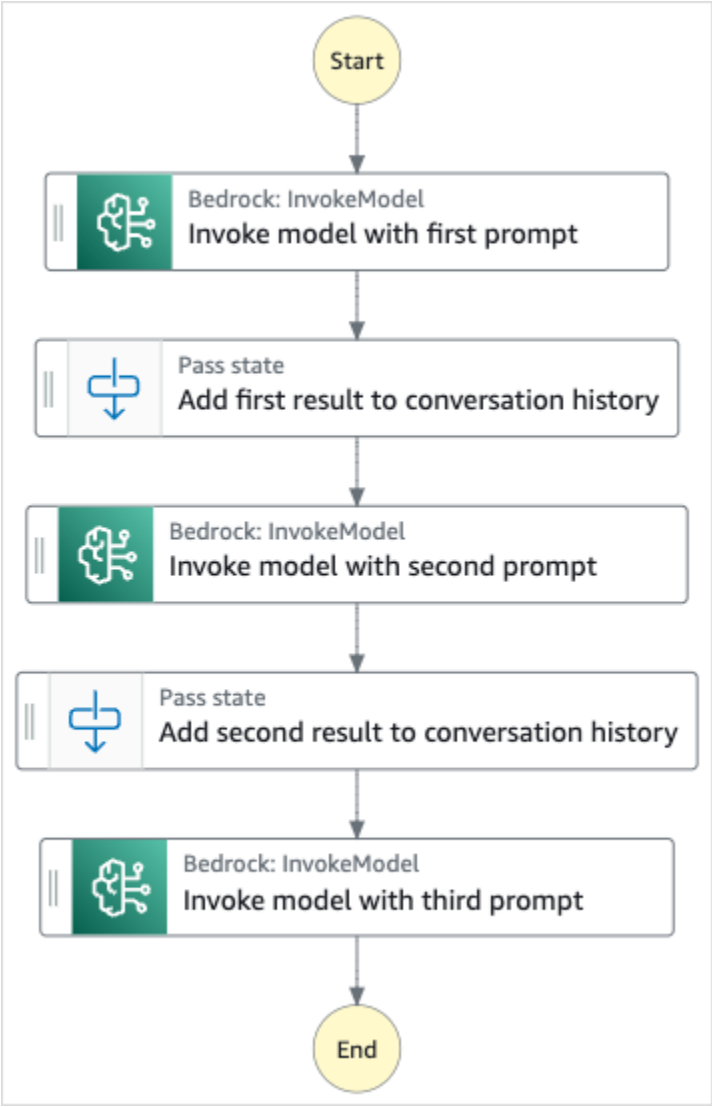
Etapa 1: Criar a máquina de estado e provisionar os recursos

1. Abra o console do Step Functions e clique em Criar máquina de estado.
2. Digite **`bedrock`** no campo de pesquisa e selecione Realizar encadeamento de prompts de IA com o Bedrock nos resultados de pesquisa exibidos.
3. Escolha Próximo para continuar.
4. Step Functions lista o Serviços da AWS usado no projeto de amostra que você selecionou. Também mostra um gráfico de fluxo de trabalho para o projeto de amostra. Implante esse projeto no seu Conta da AWS ou use-o como ponto de partida para criar seus próprios projetos. Com base em como você deseja prosseguir, escolha Executar uma demonstração ou Criar com base nela.

   Este projeto de exemplo implementa os recursos a seguir.

   - Uma máquina de AWS Step Functions estado
   - Funções relacionadas AWS Identity and Access Management (IAM)

   A imagem a seguir mostra o grafo de fluxo de trabalho do exemplo de projeto Realizar encadeamento de prompts de IA com o Bedrock:



5. Escolha Usar modelo para continuar com a seleção.
6. Execute um destes procedimentos:

   - Se você selecionou Criar com base nela, o Step Functions criará o protótipo do fluxo de trabalho para o projeto de amostra selecionado. O Step Functions não implanta os recursos listados na definição do fluxo de trabalho.

     No Modo de design do Workflow Studio, arraste e solte os estados do Navegador de estados para continuar criando seu protótipo de fluxo de trabalho. Ou mude para o Modo de código que fornece um editor de código integrado semelhante ao VS Code para atualizar a definição Amazon States Language (ASL) de sua máquina de estado no console Step Functions. Para obter mais informações sobre o uso do Workflow Studio para criar suas máquinas de estado, consulte Como usar o Workflow Studio.

> **Important**
>
> Lembre-se de atualizar o espaço reservado do nome do recurso da Amazon (ARN) para os recursos usados no projeto de amostra antes de executar o fluxo de trabalho.

- Se você selecionou Executar uma demonstração, o Step Functions cria um projeto de amostra somente para leitura que usa um AWS CloudFormation modelo para implantar os AWS recursos listados nesse modelo no seu. Conta da AWS

  > **Tip**
  >
  > Para visualizar a definição da máquina de estado do projeto de amostra, escolha Código.

  Quando estiver pronto, escolha Implemente e execute para implantar o projeto de amostra e criar os recursos.

  Pode levar até 10 minutos para que esses recursos e as permissões relacionadas ao IAM sejam criados. Enquanto seus recursos estão sendo implantados, você pode abrir o link do CloudFormation Stack ID para ver quais recursos estão sendo provisionados.

  Depois que todos os recursos do projeto de exemplo forem criados, você poderá ver o novo projeto de exemplo listado na página Máquinas de estado.

  > **Important**
  >
  > Taxas padrão podem ser aplicadas a cada serviço usado no CloudFormation modelo.

Etapa 2: Executar a máquina de estado

1. Na página Máquinas de estado, escolha seu projeto de exemplo.
2. Na página do projeto de exemplo, escolha Iniciar execução.
3. Na caixa de diálogo Iniciar execução, faça o seguinte:

1. (Opcional) Para identificar a execução, insira um nome para ela no campo Nome. Por padrão, o Step Functions gera automaticamente um nome de execução exclusivo.

   > **Note**
   >
   > O Step Functions permite criar nomes de máquina de estado, execuções, atividades e labels que contenham caracteres não ASCII. Esses nomes não ASCII não funcionam com a Amazon. CloudWatch Para garantir que você possa acompanhar CloudWatch as métricas, escolha um nome que use somente caracteres ASCII.

2. (Opcional) No campo Entrada, insira os valores de entrada no formato JSON para executar o fluxo de trabalho.

   Se você optar por Executar uma demonstração, não precisará fornecer nenhuma entrada de execução.

3. Selecione Iniciar execução.
4. O console do Step Functions direciona você para uma página em que o título é o ID da execução. Essa página é conhecida como página de Detalhes da execução. Nesta página, você pode revisar os resultados da execução à medida que a execução avança ou após a conclusão.

   Para revisar os resultados da execução, escolha estados individuais na Exibição em gráfico e, em seguida, escolha as guias individuais no painel Detalhes da etapa para visualizar os detalhes de cada estado, incluindo entrada, saída e definição, respectivamente. Para obter detalhes sobre as informações de execução que você pode visualizar na página Detalhes da execução, consulte Página de Detalhes da execução — Visão geral da interface.

Cotas

AWS Step Functions coloca cotas nos tamanhos de determinados parâmetros da máquina de estado, como o número de ações da API durante um determinado período de tempo ou o número de máquinas de estado que você pode definir. Embora o objetivo dessas cotas seja impedir que uma máquina de estado mal configurada consuma todos os recursos do sistema, muitas não são fixas.

Para solicitar um aumento da cota de serviço, você pode fazer um dos seguintes procedimentos:

- Abra o console do Service Quotas em https://console.aws.amazon.com/servicequotas/home. Para solicitar um aumento de cota usando o console do Service Quotas, consulte Solicitação de aumento de cota no Guia do Usuário do Service Quotas.
- Use a página Support Center no AWS Management Console para solicitar um aumento de cota para os recursos fornecidos por AWS Step Functions região. Para obter mais informações, consulte Service Quotas do AWS em Referência geral da AWS.

> **Note**
>
> Se determinada etapa da execução da máquina de estado ou da execução de uma atividade demorar muito, você pode configurar o tempo limite de uma máquina de estado para que ela provoque um evento de tempo limite.

Tópicos

Cotas gerais

Cota	Descrição
Nomes no Step Functions	Os nomes das máquinas de estado, execuções e tarefas de atividade não devem exceder oitenta caracteres. Esses nomes devem ser exclusivos para sua conta e AWS região e não devem conter nenhum dos seguintes itens: • Espaço em branco • Caracteres curinga (`? *`) • Caracteres de colchete (`< > { } [ ]`) • Caracteres especiais (`" # % \ ^ | ~ ` $ & , ; : /`) • caracteres de controle (`\\u0000` - `\\u001f` ou `\\u007f` - `\\u009f`). Se sua máquina de estado for do tipo Express, você poderá fornecer o mesmo nome para várias execuções da máquina de estado. O Step Functions gera um ARN de execução exclusivo para cada execução de máquina de estado Express, mesmo que várias execuções tenham o mesmo nome. Step Functions permite criar nomes para máquinas de estado, execuções e atividades, além de rótulos que contenham caracteres não ASCII. Esses nomes não ASCII não funcionam com a Amazon. CloudWatch Para garantir que você possa acompanhar CloudWatch as métricas, escolha um nome que use somente caracteres ASCII.

Cotas relacionadas a contas

Recurso	Cota padrão	Pode ser aumentado para
Número máximo de máquinas de estado registradas	10.000	25.000
Número máximo de atividades registradas	10.000	15.000
Tamanho máximo de solicitação	1 MB por solicitação. Esse é o tamanho total de dados por solicitação de API do Step Functions, incluindo o cabeçalho da solicitação e todos os outros dados associados à solicitação.	Cota fixa
Máximo de execuções abertas por conta	1 milhão de execuções para cada Conta da AWS em cada Região da AWS. Exceder isso causará um erro `ExecutionLimitExceeded` . Não se aplica aos fluxos de trabalho expressos.	Milhões
Número máximo de Execução de mapas abertos Uma Execução de mapa aberta é uma Execução de mapa que foi iniciada, mas ainda não foi concluída. As corridas de mapas programadas aguardam no MapRunStartedevento até que o número total de execuções de mapas	1000 Essa cota é aplicável ao estado Mapa distribuído.	Cota fixa

Recurso	Cota padrão	Pode ser aumentado para
abertas seja menor que a cota padrão de 1000.		
Máximo de redrives de uma Execução de mapa.	1000 Essa cota é aplicável ao estado Mapa distribuído.	Cota fixa
Número máximo de execuções secundárias paralelas do Map Run	10.000	Cota fixa

Cotas relacionadas à tarefa HTTP

Tarefas HTTP têm controle de utilização usando um esquema de bucket de token para manter a largura de banda do serviço Step Functions.

Recurso	Tamanho do bucket	Taxa de reabastecimento por segundo
Tarefa HTTP	300	300

A tabela a seguir lista a cota da duração de uma tarefa HTTP.

Recurso	Cota padrão
Duração da tarefa HTTP A duração de uma tarefa HTTP se refere ao tempo gasto por ela para enviar uma solicitação HTTP e receber uma resposta.	60 segundos Essa é uma cota rígida que não pode ser alterada.

Cotas relacionadas aos controles de utilização de estado

As transições de estados do Step Functions tem seu controle de utilização limitado por meio de um esquema de bucket de token, para manter a largura de banda do serviço. Os fluxos de trabalho padrão e os fluxos de trabalho expressos têm diferentes controles de utilização de transições de estado. As cotas de fluxo de trabalho padrão são cotas flexíveis e podem ser aumentadas.

> **Note**
>
> A limitação na métrica do `StateTransition` serviço é relatada como `ExecutionThrottled` na Amazon. CloudWatch Para obter mais informações, consulte a `ExecutionThrottled` CloudWatch métrica.

	Standard		Express	
Métrica de serviço	Tamanho do bucket	Taxa de reabastecimento por segundo	Tamanho do bucket	Taxa de reabastecimento por segundo
`StateTransition` — No Leste dos EUA (Norte da Virgínia), Oeste dos EUA (Oregon) e na Europa (Irlanda)	5.000	5.000	Ilimitado	Ilimitado
`StateTransition` — Todas as outras regiões	800	800	Ilimitado	Ilimitado

Cotas relacionadas ao controle de utilização das ações de API

Algumas ações de API do Step Functions têm controle de utilização por meio de um esquema de bucket do token para manter a largura de banda do serviço. Essas cotas são flexíveis e podem ser aumentadas.

> **Note**
>
> As cotas de limitação são por conta, por região. AWS
> AWS Step Functions pode aumentar o tamanho do balde e a taxa de recarga a qualquer momento.

	Standard		Express	
Nome da API	Tamanho do bucket	Taxa de reabastecimento por segundo	Tamanho do bucket	Taxa de reabastecimento por segundo
`StartExecution` — No Leste dos EUA (Norte da Virgínia), Oeste dos EUA (Oregon) e na Europa (Irlanda)	1.300	300	6.000	6.000
`StartExecution` — Todas as outras regiões	800	150	6.000	6.000

Cota relacionada à API TestState

Nome da API	Cota	Pode ser aumentado para
TestState	1 transação por segundo (TPS)	Cota fixa

Outras cotas

Essas cotas são flexíveis e podem ser aumentadas.

	In US East (N. Virginia), US West (Oregon), and Europe (Ireland)		All other regions	
Nome da API	Tamanho do bucket	Taxa de reabastecimento por segundo	Tamanho do bucket	Taxa de reabastecimento 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	
Nome da API	Tamanho do bucket	Taxa de reabastecimento por segundo	Tamanho do bucket	Taxa de reabastecimento por segundo
`DescribeStateMachine`	200	20	200	20
`DescribeStateMachineForExecution`	200	1	200	1
`GetActivityTask`	3.000	500	1.500	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`	3.000	500	1.500	300
`SendTaskHeartbeat`	3.000	500	1.500	300

	In US East (N. Virginia), US West (Oregon), and Europe (Ireland)		All other regions	
Nome da API	Tamanho do bucket	Taxa de reabastecimento por segundo	Tamanho do bucket	Taxa de reabastecimento por segundo
`SendTaskSuccess`	3.000	500	1.500	300
StartSync Execution	As chamadas de API da execução do expresso síncrono não contribuem para os limites existentes de capacidade da conta. O Step Functions fornece a capacidade sob demanda e escala automaticamente com workload sustentada. Os picos na workload podem ser reduzidos até que a capacidade esteja disponível. Se sofrer controle de utilização, tente novamente depois de algum tempo. Para obter informações sobre fluxos de trabalho expressos síncronos, consulte Fluxos de trabalho expresso síncronos e assíncronos.			
`StopExecution`	1.000	200	500	25
`TagResource`	200	1	200	1
`UntagResource`	200	1	200	1
`UpdateStateMachine`	100	1	100	1

Cotas relacionadas a execuções de máquina de estado

A tabela a seguir descreve as cotas relacionadas às execuções de máquinas de estado. As cotas de execução da máquina de estado são cotas fixas que não podem ser alteradas, exceto a cota de tempo de retenção do histórico de execução.

Cota	Padrão	Express
Tempo máximo de execução	1 ano. Se uma execução for executada por mais do que o máximo de 1 ano, ela falhará com um `States.Timeout` erro e emitirá uma `ExecutionsTimedOut` CloudWatch métrica.	5 minutos. Se uma execução for executada por mais de 5 minutos no máximo, ela falhará com um `States.Timeout` erro e emitirá uma `ExecutionsTimedOut` CloudWatch métrica.
Tamanho máximo do histórico de execução	25 mil eventos em um único histórico de execução de máquina de estado. Se o histórico de execução atingir essa cota, haverá falha na execução. Para evitar isso, consulte Evitar atingir a cota do histórico.	Ilimitada.
Tempo máximo de ociosidade da execução	1 ano (restrito pelo tempo máximo de execução).	5 minutos (restrito pelo tempo máximo de execução).
Tempo de retenção do histórico de execução	90 dias após o encerramento da execução. Após esse tempo, não será mais possível recuperar ou visualizar o histórico de execução. Não há cotas adicionais para o número de execuções encerradas que são retidas pelo Step Functions. Para atender aos requisitos de conformidade, organizacionais ou regulamentares, você pode reduzir o período de retenção do histórico de execução	Para ver o histórico de execução, o registro do Amazon CloudWatch Logs deve ser configurado. Para ter mais informações, consulte Como registrar usando o CloudWatch Logs.

Cota	Padrão	Express
	para 30 dias, enviando uma solicitação de cota. Para fazer isso, use o AWS Support Center Console e crie um novo caso. A alteração para reduzir o período de retenção para trinta dias é aplicável a cada conta em uma região.	
Período redrivable de execução O período Redrivable refere-se ao tempo durante o qual você pode redrive uma determinada execução do fluxo de trabalho padrão. Esse período começa no dia em que uma máquina de estado conclui sua execução.	14 dias. Essa cota fixa é aplicável ao estado Mapa distribuído.	No momento, o Redrive não é compatível com os fluxos de trabalho expressos.

Cotas relacionadas a execuções de tarefas

A tabela a seguir descreve as cotas relacionadas às execuções de tarefas. Todas essas são cotas rígidas que não podem ser alteradas.

Cota	Padrão	Express
Tempo máximo de execução de tarefa	1 ano (restrito pelo tempo máximo de execução)	5 minutos (restrito pelo tempo máximo de execução)

Cota	Padrão	Express
Tempo máximo que o Step Functions mantém uma tarefa na fila	1 ano (restrito pelo tempo máximo de execução)	5 minutos (restrito pelo tempo máximo de execução)
Número máximo de pesquisas de atividade por nome do recurso da Amazon (ARN)	1.000 agentes de sondagem chamando `GetActivityTask` por ARN. Exceder essa cota resultará neste erro: "The maximum number of workers concurrently polling for activity tasks has been reached." (O número máximo de operadores fazendo a sondagem de tarefas de atividade simultaneamente foi atingido.)	Não se aplica a fluxos de trabalho expressos.
Tamanho máximo de entrada ou saída para uma tarefa, estado ou execução	256 KB de dados como uma cadeia codificada em UTF-8. Essa cota afeta tarefas (atividade, função do Lambda ou serviço integrado), estados ou dados de resultado de execução e dados de entrada ao programar uma tarefa, inserir um estado ou iniciar uma execução.	256 KB de dados como uma cadeia codificada em UTF-8. Essa cota afeta tarefas (atividade, função do Lambda ou serviço integrado), estados ou dados de resultado de execução e dados de entrada ao programar uma tarefa, inserir um estado ou iniciar uma execução.

Cotas relacionadas a versões e aliases

Recurso	Cota padrão
Número máximo de versões publicadas da máquina de estado	Mil para cada máquina de estado.

Recurso	Cota padrão
	Para solicitar um aumento desse limite flexível, use a página Support Center no AWS Management Console.
Número máximo de aliases de máquinas de estado	Cem para cada máquina de estado. Para solicitar um aumento desse limite flexível, use a página Support Center no AWS Management Console.

Restrições relacionadas à marcação

Esteja ciente dessas restrições ao marcar recursos do Step Functions.

> **Note**
>
> As restrições de marcação não podem ser aumentadas como outras cotas.

Restrição	Descrição
Número máximo de tags por recurso	50
Tamanho máximo da chave	128 caracteres do Unicode em UTF-8
Tamanho máximo do valor	256 caracteres Unicode em UTF-8
Restrição de prefixo	Não use o `aws:` prefixo nos nomes ou valores das tags porque ele está reservado para AWS uso. Você não pode editar nem excluir nomes ou valores de tag com esse prefixo. As tags com esse prefixo não contam para as tags por cota de recurso.

Restrição	Descrição
Restrições de caracteres	As tags só podem conter letras Unicode, números, espaços em branco ou estes símbolos: `_ . : / = + - @`.

Registro e monitoramento em AWS Step Functions

O registro e o monitoramento são importantes para manter a confiabilidade, a disponibilidade e o desempenho do Step Functions e de suas AWS soluções. Existem várias ferramentas disponíveis para uso com o Step Functions:

> **Tip**
>
> Para implantar um exemplo de fluxo de trabalho em seu Conta da AWS e aprender a monitorar métricas, registros e traços da execução do fluxo de trabalho, consulte o Módulo 12 - Observabilidade do AWS Step Functions Workshop.

Tópicos

Monitorando Step Functions usando CloudWatch

O monitoramento é uma parte importante da manutenção da confiabilidade, disponibilidade e desempenho de AWS Step Functions suas AWS soluções. Você deve coletar o máximo de dados de monitoramento dos AWS serviços que você usa para poder depurar falhas de vários pontos. Antes de começar a monitorar o Step Functions, você deve criar um plano de monitoramento que responda às seguintes perguntas:

- Quais são seus objetivos de monitoramento?
- Quais recursos você vai monitorar?
- Com que frequência você vai monitorar esses recursos?
- Quais ferramentas de monitoramento você usará?
- Quem realizará o monitoramento das tarefas?

- Quem deve ser notificado quando algo der errado?

A próxima etapa é estabelecer um parâmetro de desempenho normal do em seu ambiente. Para isso, meça o desempenho em vários momentos e em diferentes condições de carga. Ao monitorar o Step Functions, avalie a possibilidade de armazenar os dados históricos de monitoramento. Esses dados podem servir de parâmetro para você comparar com os dados de desempenho atuais, identificar padrões normais de desempenho e anomalias de desempenho e criar meios para a solução de problemas.

Por exemplo, com o Step Functions, você pode monitorar quantas atividades ou AWS Lambda tarefas falham devido a um tempo limite de pulsação. Quando o desempenho ficar aquém do parâmetro que você estabeleceu, é provável que precise alterar seu intervalo de pulsação.

Para estabelecer um parâmetro, você deve monitorar no mínimo as métricas a seguir:

- `ActivitiesStarted`
- `ActivitiesTimedOut`
- `ExecutionsStarted`
- `ExecutionsTimedOut`
- `LambdaFunctionsStarted`
- `LambdaFunctionsTimedOut`

As seções a seguir descrevem as métricas que o Step Functions fornece à Amazon CloudWatch. Você pode usar essas métricas para rastrear suas máquinas de estado e atividades e definir alarmes sobre valores limite. Você pode visualizar as métricas usando AWS Management Console o.

Métricas que relatam um intervalo de tempo

Algumas das CloudWatch métricas do Step Functions são intervalos de tempo, sempre medidos em milissegundos. Essas métricas geralmente correspondem aos estágios da execução para os quais você pode definir limites de tempo para uma máquina de estado, atividade e função do Lambda, com nomes descritivos.

Por exemplo, a métrica `ActivityRunTime` mede quanto tempo uma atividade precisa para ser concluída depois que começa a ser executada. Você pode definir um valor de tempo limite para o mesmo espaço de tempo.

No CloudWatch console, você pode obter os melhores resultados se escolher a média como estatística de exibição para métricas de intervalo de tempo.

Métricas que relatam uma contagem

Algumas das CloudWatch métricas do Step Functions relatam os resultados como uma contagem. Por exemplo, o `ExecutionsFailed` registra o número de execuções de máquina de estado com falha.

O Step Functions emite duas `ExecutionsStarted` métricas para cada execução de máquina de estado. Isso faz com que a SampleCountestatística da `ExecutionsStarted` métrica mostre o valor de 2 para cada execução de máquina de estado. A SampleCount estatística mostra `ExecutionStarted=0` quando `ExecutionStarted=1` a execução é concluída.

> **Tip**
>
> Recomendamos selecionar Sum como estatística de exibição para métricas que relatam uma contagem no CloudWatch console.

Métricas de execução

O `AWS/States` namespace inclui as seguintes métricas para todas as execuções de Step Functions. Essas são métricas adimensionais que se aplicam à sua conta em uma região.

Métrica	Descrição
`OpenExecutionCount`	Número aproximado de execuções abertas no momento — fluxos de trabalho que estão em andamento na sua conta. A intenção é fornecer informações sobre quando seus fluxos de trabalho estão se aproximando do limite máximo de execução, para evitar ExecutionLimitExceedederros ao chamar `StartExecution` ou `RedriveExecution` para fluxos de trabalho padrão. A métrica depende das transições de estado do fluxo de trabalho ativo, portanto, em níveis baixos, a estimativa pode não

Métrica	Descrição
	estar alinhada com a contagem observada do fluxo de trabalho em execução.
`OpenExecutionLimit`	Número máximo de execuções abertas. Para ter mais informaçõ es, consulte Cotas relacionadas a contas. Esse limite não se aplica aos fluxos de trabalho expressos.

Métricas de execução para máquina de estado com versão ou alias

Quando você executa uma execução de máquina de estado com uma versão ou um alias, o Step Functions emite as seguintes métricas. A `ExecutionThrottled` métrica só será emitida no caso de execução limitada. Essas métricas incluirão `StateMachineArn` a identificação de uma máquina de estado específica.

Métrica	Descrição
`ExecutionTime`	Intervalo, em milissegundos, entre o momento em que a execução é iniciada e a hora em que ela é encerrada.
`ExecutionThrottled`	Número de `StateEntered` eventos e novas tentativas que foram limitados. Isso está relacionado à limitação `StateTran sition` . Para ter mais informações, consulte Cotas relaciona das aos controles de utilização de estado.
`ExecutionsAborted`	Número de execuções abortadas ou encerradas.
`ExecutionsFailed`	Número de execuções malsucedidas.
`ExecutionsStarted`	Número de execuções iniciadas.
`ExecutionsSucceeded`	Número de execuções concluídas com sucesso.
`ExecutionsTimedOut`	Número de execuções que expiram por qualquer motivo.

Métricas de execução para fluxos de trabalho expressos

O namespace de `AWS/States` inclui as seguintes métricas para as execuções dos fluxos de trabalho expressos do Step Functions.

Métrica	Descrição
`ExpressExecutionMemory`	A memória total consumida por um fluxo de trabalho expresso.
`ExpressExecutionBilledDuration`	A duração pela qual um fluxo de trabalho expresso é cobrado.
`ExpressExecutionBilledMemory`	A quantidade de memória consumida pela qual um fluxo de trabalho expresso é cobrado.

Métricas de execução de Redrive para fluxos de trabalho padrão

Quando você redrive uma execução de máquina de estado, o Step Functions emite as seguintes métricas.

Para todas as execuções redriven, a métrica `Executions*` é emitida. Por exemplo, digamos que uma execução redriven seja abortada. Essa execução emitirá pontos de dados diferentes de zero para `RedrivenExecutionsAborted` e `ExecutionsAborted`.

Métrica	Descrição
`ExecutionsRedriven`	Número de redriven execuções.
`RedrivenExecutionsAborted`	Número de redriven execuções canceladas ou encerradas.
`RedrivenExecutionsTimedOut`	Número de redriven execuções que expiram por qualquer motivo.
`RedrivenExecutionsSucceeded`	Número de redriven execuções concluídas com êxito.

Métrica	Descrição
`RedrivenExecutionsFailed`	Número de redriven execuções que falharam.

Dimensão das métricas de execução do Step Functions

Dimensão	Descrição
`StateMachineArn`	O Nome do Recurso da Amazon (ARN) da máquina de estado para a execução em questão.

Dimensões para execuções com versão

Dimensão	Descrição
`StateMachineArn`	O Nome do Recurso da Amazon (ARN) da máquina de estado cuja execução foi iniciada por uma versão.
`Version`	Versão da máquina de estado utilizada para iniciar a execução.

Dimensões para execuções com alias

Dimensão	Descrição
`StateMachineArn`	O Nome do Recurso da Amazon (ARN) da máquina de estado cuja execução foi iniciada por um alias.
`Alias`	Alias da máquina de estado utilizado para iniciar a execução.

Métricas de contagem de recursos para versões e aliases

O namespace `AWS/States` inclui as seguintes métricas para a contagem de versões e aliases de uma máquina de estado.

Métrica	Descrição
AliasCount	Número de aliases criados para a máquina de estado. Você pode criar até cem aliases para cada máquina de estado.
VersionCount	Número de versões publicadas para a máquina de estado. Você pode publicar até mil versões de uma máquina de estado.

Dimensão das métricas de contagem de recursos para versões e aliases

Dimensão	Descrição
`ResourceArn`	O Nome do Recurso da Amazon (ARN) da máquina de estado com uma versão ou alias.

Métricas de atividade

O namespace `AWS/States` inclui as métricas a seguir para atividades do Step Functions.

Métrica	Descrição
`ActivityRunTime`	Intervalo, em milissegundos, entre a hora em que a atividade começa e a hora em que ela é encerrada.
`ActivityScheduleTime`	Intervalo, em milissegundos, durante o qual a atividade permanece no estado de agendamento.
`ActivityTime`	Intervalo, em milissegundos, entre a hora em que a atividade é agendada e a hora em que ela é encerrada.
`ActivitiesFailed`	Número de atividades que falharam.
`ActivitiesHeartbeatTimedOut`	Número de atividades que atingem o tempo limite devido ao tempo limite do batimento cardíaco.

Métrica	Descrição
`ActivitiesScheduled`	Número de atividades programadas.
`ActivitiesStarted`	Número de atividades iniciadas.
`ActivitiesSucceeded`	Número de atividades concluídas com sucesso.
`ActivitiesTimedOut`	Número de atividades que expiram quando fecham.

Dimensão das métricas de atividade do Step Functions

Dimensão	Descrição
`ActivityArn`	O ARN da atividade.

Métricas de função do Lambda

O namespace `AWS/States` inclui as métricas a seguir para funções do Lambda no Step Functions.

Métrica	Descrição
`LambdaFunctionRunTime`	Intervalo, em milissegundos, entre o momento em que a função Lambda é iniciada e a hora em que ela é fechada.
`LambdaFunctionScheduleTime`	Intervalo, em milissegundos, durante o qual a função Lambda permanece no estado de agendamento.
`LambdaFunctionTime`	Intervalo, em milissegundos, entre a hora em que a função Lambda é programada e a hora em que ela é fechada.
`LambdaFunctionsFailed`	Número de funções Lambda com falha.
`LambdaFunctionsScheduled`	Número de funções Lambda programadas.

Métrica	Descrição
`LambdaFunctionsStarted`	Número de funções Lambda iniciadas.
`LambdaFunctionsSucceeded`	Número de funções Lambda concluídas com sucesso.
`LambdaFunctionsTimedOut`	Número de funções do Lambda que atingem o tempo limite de fechamento.

Dimensão para Métricas de Função do Lambda para o Step Functions

Dimensão	Descrição
`LambdaFunctionArn`	O ARN da função Lambda.

> **Note**
>
> As métricas da função do Lambda são emitidas para estados de tarefas que especificam o ARN da função Lambda no campo `Resource`. Em vez disso, estados de tarefas que usam `"Resource": "arn:aws:states:::lambda:invoke"` emitem métricas de integração de serviços. Para ter mais informações, consulte Invocar o Lambda com o Step Functions.

Métricas de integração de serviço

O namespace `AWS/States` inclui as seguintes métricas para integrações de serviços do Step Functions. Para ter mais informações, consulte Usando AWS Step Functions com outros serviços.

Métrica	Descrição
`ServiceIntegrationRunTime`	Intervalo, em milissegundos, entre o momento em que a tarefa de serviço é iniciada e a hora em que ela é fechada.

Métrica	Descrição
`ServiceIntegrationScheduleTime`	Intervalo, em milissegundos, durante o qual a tarefa de serviço permanece no estado agendado.
`ServiceIntegrationTime`	Intervalo, em milissegundos, entre a hora em que a Tarefa de Serviço é agendada e a hora em que ela é fechada.
`ServiceIntegrationsFailed`	Número de tarefas de serviço com falha.
`ServiceIntegrationsScheduled`	Número de tarefas de serviço agendadas.
`ServiceIntegrationsStarted`	Número de tarefas de serviço iniciadas.
`ServiceIntegrationsSucceeded`	Número de tarefas de serviço concluídas com êxito.
`ServiceIntegrationsTimedOut`	Número de tarefas de serviço que atingem o tempo limite de fechamento.

Dimensão para métricas de integração de serviço do Step Functions

Dimensão	Descrição
`ServiceIntegrationResourceArn`	O ARN do recurso do serviço integrado.

Métricas de serviço

O namespace `AWS/States` inclui as métricas a seguir para o serviço do Step Functions.

Métrica	Descrição
`ThrottledEvents`	Contagem de solicitações que foram limitadas.

Métrica	Descrição
`ProvisionedBucketSize`	Contagem de solicitações disponíveis por segundo.
`ProvisionedRefillRate`	Contagem de solicitações por segundo que são permitidas no bucket.
`ConsumedCapacity`	Contagem de solicitações por segundo.

Dimensão das métricas de serviço do Step Functions

Dimensão	Descrição
`ServiceMetric`	Filtra os dados para mostrar as métricas de transições de estado.

Métricas da API

O namespace `AWS/States` inclui as métricas a seguir para funções de API do Step Functions.

Métrica	Descrição
`ThrottledEvents`	Contagem de solicitações que foram limitadas.
`ProvisionedBucketSize`	Contagem de solicitações disponíveis por segundo.
`ProvisionedRefillRate`	Contagem de solicitações por segundo que são permitidas no bucket.
`ConsumedCapacity`	Contagem de solicitações por segundo.

Dimensão das métricas de API do Step Functions

Dimensão	Descrição
`APIName`	Filtra dados para uma API do nome de API especificado.

Entrega de CloudWatch métricas de melhor esforço

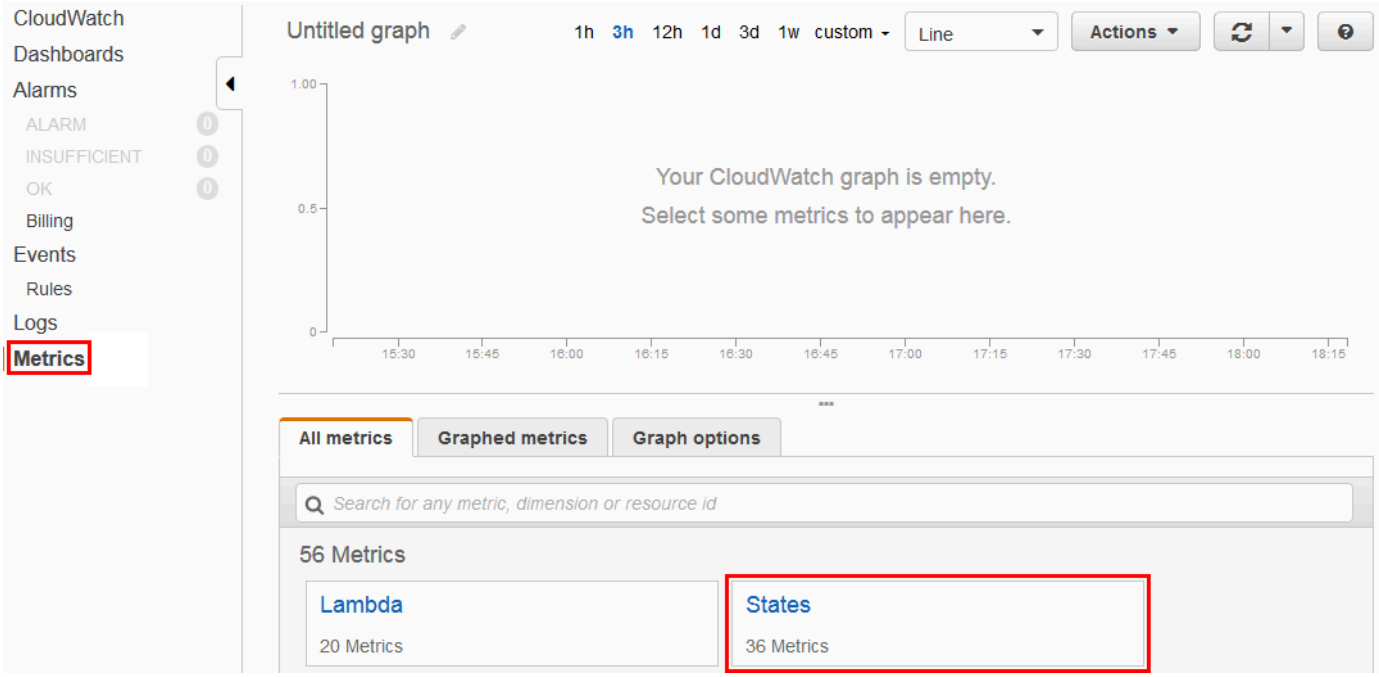
As métricas do CloudWatch são entregues com base em melhor esforço.

A integridade e pontualidade das métricas não são garantidas. O ponto de dados para uma solicitação específica pode ser retornado com um timestamp posterior à solicitação processada. O ponto de dados pode ser adiado por CloudWatch um minuto antes de ser disponibilizado ou pode nem mesmo ser entregue. CloudWatchas métricas de solicitação dão uma ideia das execuções da máquina de estado quase em tempo real. Não se trata de uma contabilidade completa de todas as métricas relacionadas à execução.

Devido à natureza de melhor esforço deste atributo, os relatórios disponíveis no Painel de gerenciamento de custos e faturamento podem incluir uma ou mais solicitações de acesso que não aparecem nas métricas de execução.

Visualizar as métricas do Step Functions

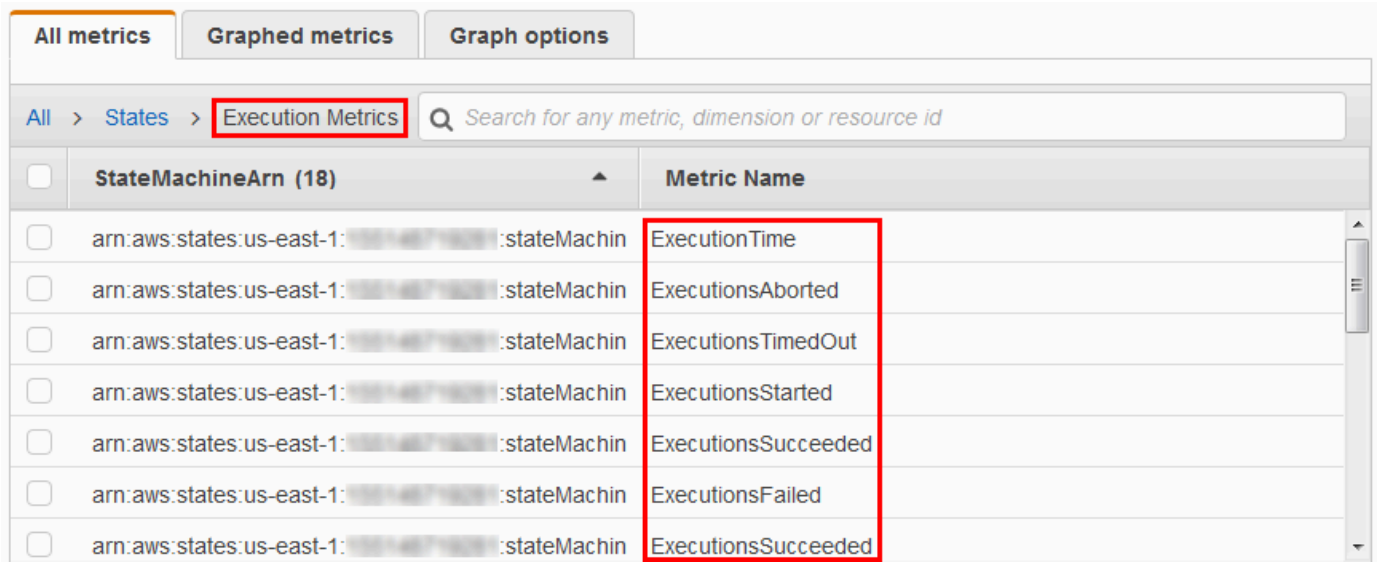
1. Faça login no AWS Management Console e abra o CloudWatch console.
2. Escolha Metrics (Métricas) e, na guia All Metrics (Todas as métricas), escolha States (Estados).



Se você tiver qualquer execução recente, verá até quatro tipos de métrica:

- Execution Metrics (Métricas de execução)
- Métricas de função de atividade
- Métricas de função do Lambda
- Métricas de integração de serviço

3. Escolha um tipo de métrica para ver uma lista de métricas.

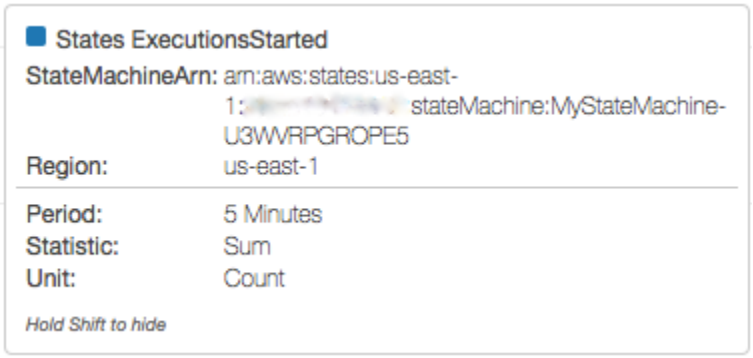


- Para classificar suas métricas por nome da métrica ou StateMachineArn, use os cabeçalhos das colunas.
- Para visualizar os gráficos correspondentes a uma métrica, marque a caixa de seleção ao lado da métrica na lista. Você pode alterar os parâmetros do gráfico usando os controles de período acima da visualização do gráfico.

  Você pode escolher períodos personalizados usando valores relativos ou absolutos (especifique dias e horas). Você também pode usar a lista suspensa para exibir valores como linhas, áreas empilhadas ou números (valores).
- Para visualizar detalhes de um gráfico, passe o mouse sobre o código de cor da métrica que é exibido abaixo do gráfico.

  Os detalhes da métrica são exibidos.

Para obter mais informações sobre como trabalhar com CloudWatch métricas, consulte Usando CloudWatch as métricas da Amazon no Guia CloudWatch do usuário da Amazon.

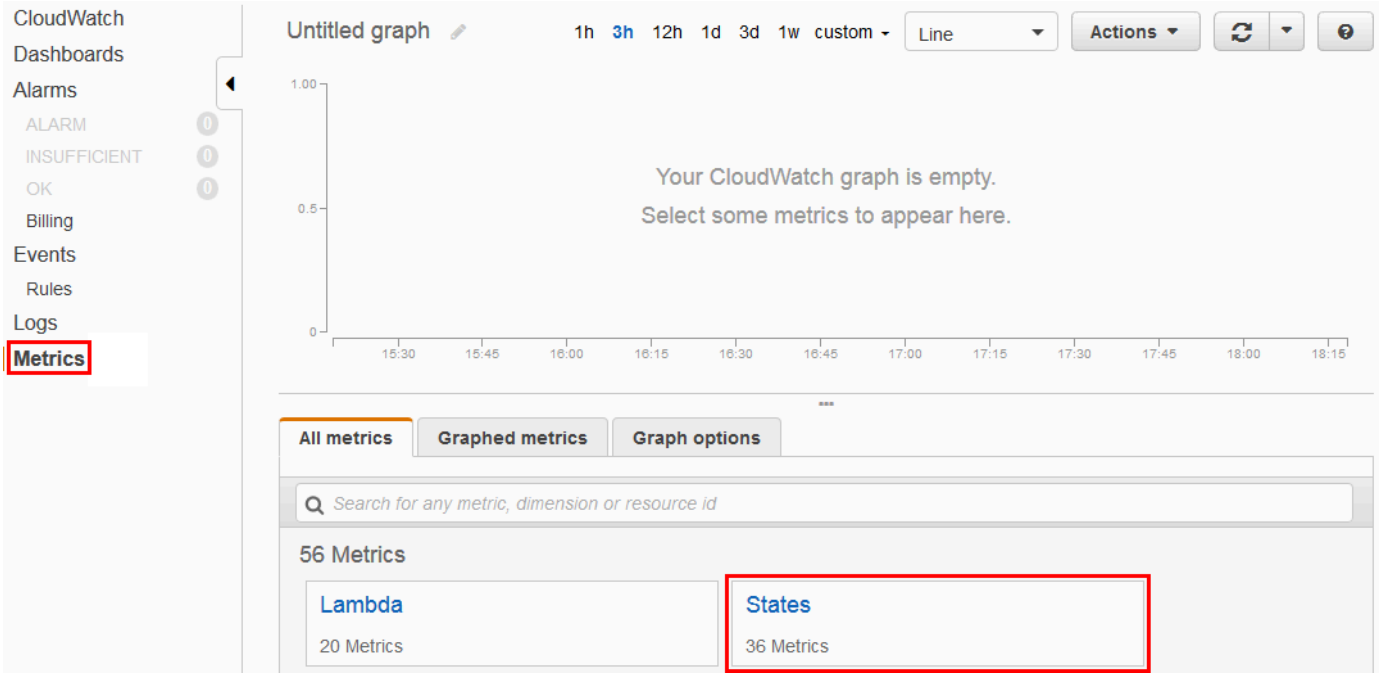
Como configurar alarmes para o Step Functions

Você pode usar os CloudWatch alarmes da Amazon para realizar ações. Por exemplo, se quiser saber quando um limite de alarme foi atingido, você pode definir um alarme para enviar uma notificação para um tópico do Amazon SNS ou para enviar um e-mail quando a métrica do `StateMachinesFailed` ultrapassar determinado limite.

Para definir um alarme em uma métrica

1. Faça login no AWS Management Console e abra o CloudWatch console.

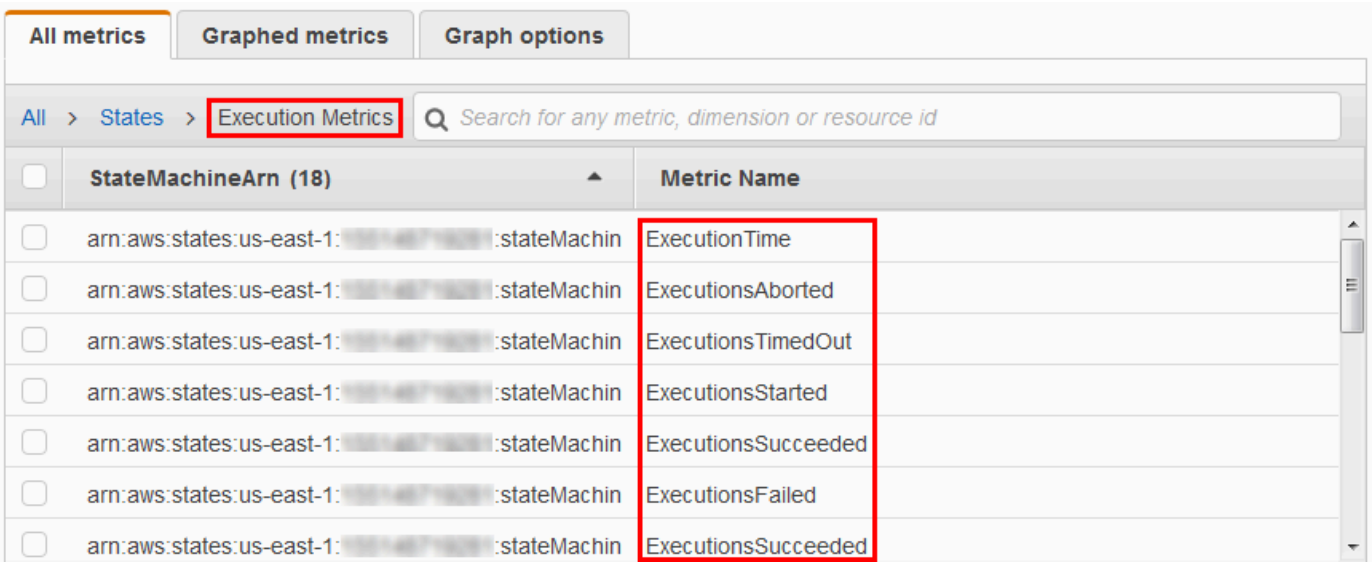
2. Escolha Metrics (Métricas) e, na guia All Metrics (Todas as métricas), escolha States (Estados).



Se você tiver qualquer execução recente, verá até quatro tipos de métrica:

- Execution Metrics (Métricas de execução)
- Métricas de função de atividade
- Métricas de função do Lambda
- Métricas de integração de serviço

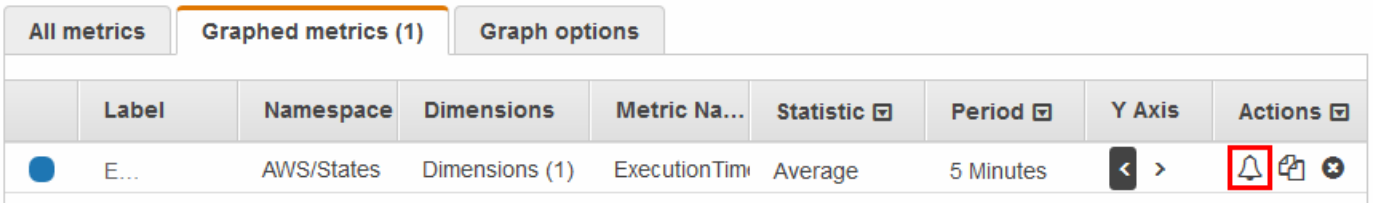
3. Escolha um tipo de métrica para ver uma lista de métricas.



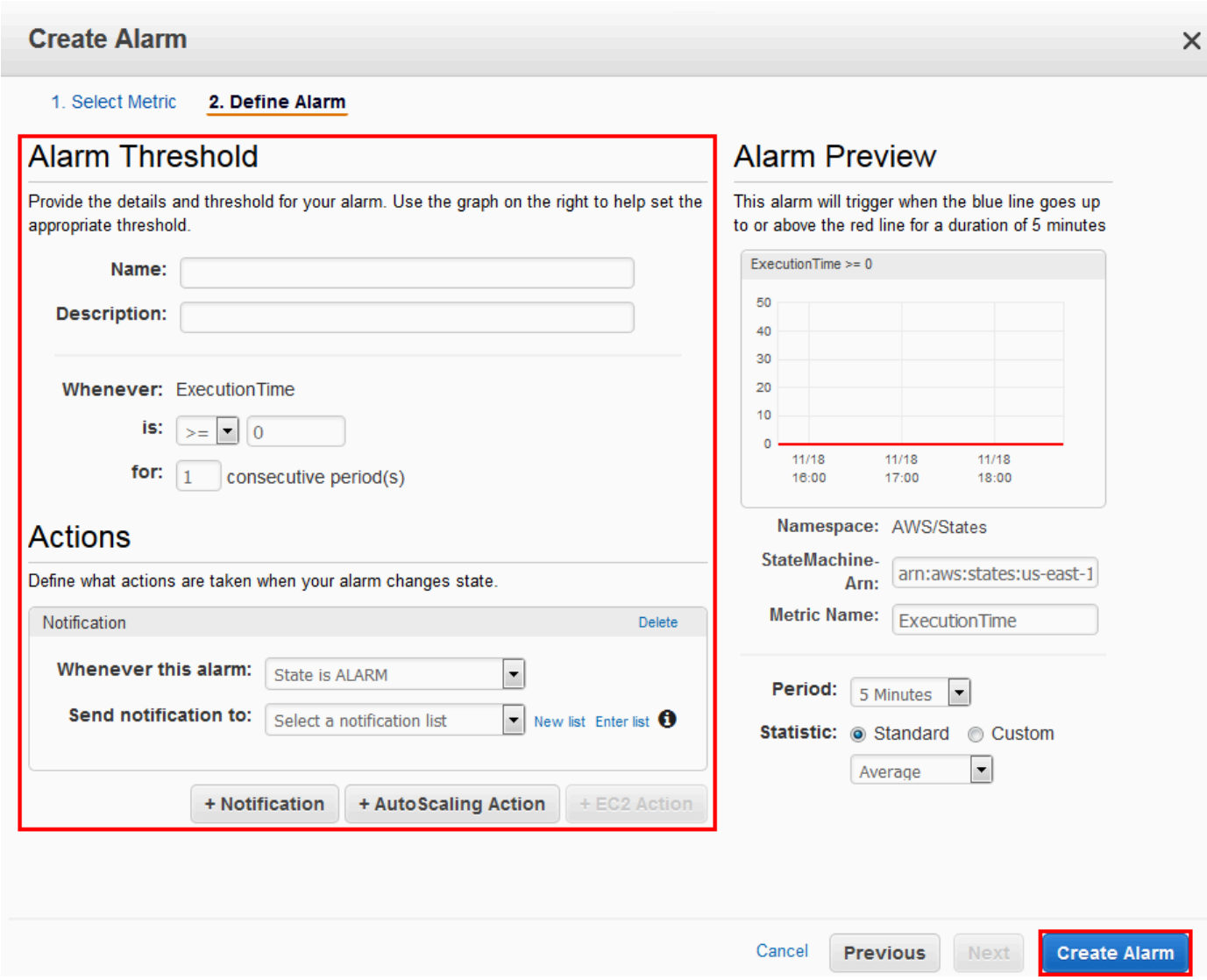
4. Escolha uma métrica e, depois, Graphed metrics (Métricas em gráfico).
5. Escolha

   ao lado de uma métrica na lista.



A página Create Alarm (Criar alarme) é exibida.



6. Insira os valores para Alarm threshold (Limite de alarme) e Actions (Ações) e escolha Create Alarm (Criar alarme).

Para obter mais informações sobre a configuração e o uso de CloudWatch alarmes, consulte Criação de CloudWatch alarmes da Amazon no Guia CloudWatch do usuário da Amazon.

EventBridge (CloudWatch Eventos) para mudanças no status de execução do Step Functions

A Amazon EventBridge é um AWS serviço que permite que você responda às mudanças de estado em um AWS recurso. Por exemplo, você pode responder às alterações do status de execução de um fluxo de trabalho padrão do Step Functions EventBridge usando as duas formas a seguir:

- Você pode configurar EventBridge regras para reagir aos eventos que são emitidos quando o status de execução de uma máquina de estado do Step Functions muda. Isso permite que você monitore seus fluxos de trabalho sem a necessidade de sondar constantemente usando a API do DescribeExecution. Com base nas mudanças nas execuções das máquinas de estado, você pode usar um EventBridge destino para iniciar novas execuções de máquinas de estado, chamar AWS Lambda funções, publicar mensagens nos tópicos do Amazon Simple Notification Service (Amazon SNS) e muito mais.
- Você também pode configurar uma máquina de estado Step Functions como destino em EventBridge. Isso permite que você acione uma execução de um fluxo de trabalho do Step Functions em resposta a um evento de outro serviço da AWS .

  Para obter mais informações, consulte o Guia EventBridge do usuário da Amazon.

No entanto, os fluxos de trabalho expressos não emitem eventos para o. EventBridge Para monitorar a execução de um fluxo de trabalho expresso, você pode usar o CloudWatch Logs. Para fazer isso, na página Detalhes da execução da máquina de estado, escolha as guias Monitoramento e Registro em log. Na guia Monitoramento, você pode visualizar as CloudWatch métricas de eventos, como duração da execução, erros de execução e memória faturada. Na guia Registro em log, você pode ver os logs recentes e a configuração do log.

> **Tip**
>
> Para implantar um exemplo de fluxo de trabalho expresso em seu Conta da AWS e aprender a monitorar fluxos de trabalho expressos, consulte o módulo Monitorando fluxos de trabalho expressos do The AWS Step Functions Workshop.

EventBridge cargas úteis

Um EventBridge evento pode conter uma propriedade de entrada em sua definição. Para alguns eventos, um EventBridge evento também pode conter uma propriedade de saída em sua definição.

- Se a entrada de escape combinada e a saída de escape enviadas EventBridge excederem 248 KB, a entrada será excluída. Da mesma forma, se a saída de escape exceder 248 KB, a saída será excluída. Isso é resultado das cotas de EventBridge eventos.
- Você pode determinar se uma carga foi truncada com as propriedades `inputDetails` e `outputDetails`. Para ver mais informações, consulte o Tipo de dados de `CloudWatchEventsExecutionDataDetails`.
- Para fluxos de trabalho padrão, você pode ver a entrada e a saída completas usando DescribeExecution.
- O `DescribeExecution` não está disponível para fluxos de trabalho expressos. Para ver a entrada/saída completa, você pode agrupar seu fluxo de trabalho expresso com um fluxo de trabalho padrão. Outra opção é usar ARNs do Amazon S3. Para ver mais informações sobre como utilizar ARNs, consulte the section called “Use ARNs do Amazon S3 em vez de transmitir grandes cargas”.

Tópicos

- Exemplos de eventos do Step Functions
- Roteamento de um evento Step Functions para EventBridge o console EventBridge

Exemplos de eventos do Step Functions

Veja a seguir exemplos de Step Functions enviando eventos para EventBridge:

Tópicos

- Execução iniciada
- Execução bem-sucedida
- Falha na execução
- A execução atingiu o tempo limite
- Execução interrompida

Em cada caso, a seção `detail` em dados de evento fornece as mesmas informações da API do `DescribeExecution`. O campo `status` indica o status da execução no momento em que o evento foi enviado (`RUNNING`, `SUCCEEDED`, `FAILED`, `TIMED_OUT` ou `ABORTED` dependendo do evento emitido).

Execução 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
    }
}
```

Execução bem-sucedida

```
{
    "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
        }
    }
}
```

Falha na execução

```
{
    "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
    }
}
```

A execução atingiu o tempo limite

```
{
    "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
    }
}
```

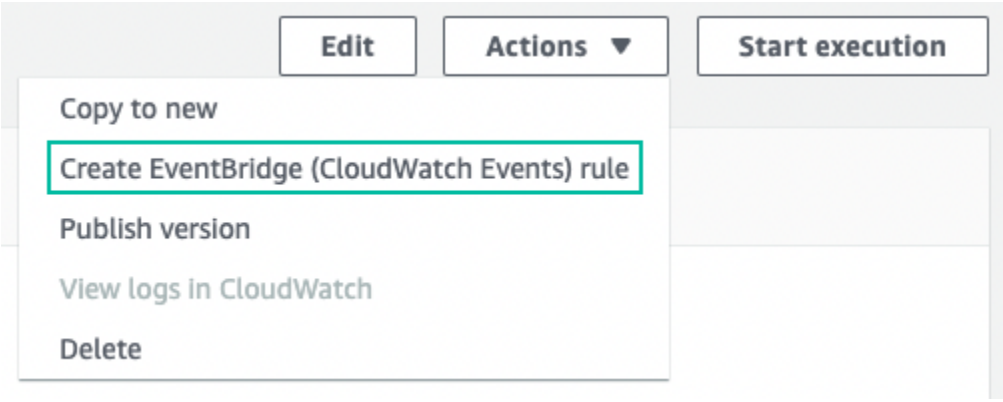
Execução interrompida

```
{
    "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
    }
}
```

Roteamento de um evento Step Functions para EventBridge o console EventBridge

Use as instruções a seguir para aprender como acionar a execução de uma máquina de estado do Step Functions sempre que uma máquina de estado específica do Step Functions for executada com êxito. Você usa o EventBridge console da Amazon para especificar a máquina de estado cuja execução você deseja acionar.

1. Na página Detalhes de uma máquina de estado, escolha Ações e, em seguida, escolha Criar regra EventBridge (CloudWatch Eventos).



   Como alternativa, abra o EventBridge console em https://console.aws.amazon.com/events/. No painel de navegação, selecione Regras em Barramentos.
2. Escolha a opção Criar regra. Isso abre a página Definir detalhes da regra.
3. Insira um Nome (por exemplo, *`StepFunctionsEventRule`*) e, opcionalmente, uma Descrição para a regra.
4. Para Barramento de eventos e Tipo de regra, mantenha as seleções padrão.
5. Escolha Próximo. Isso abre a página Criar padrão de evento.
6. Em Origem do evento, mantenha a seleção padrão de AWS eventos ou eventos de EventBridge parceiros.
7. Mantenha as seleções padrão para as seções Evento de amostra e Método de criação.
8. Em Padrão de evento, faça o seguinte:

   a. Na lista suspensa Origem do evento, mantenha a seleção padrão de serviços da AWS .
   b. Na lista suspensa Serviço da AWS , selecione Step Functions.
   c. Na lista suspensa Tipo de evento, selecione Alteração do status de execução do Step Functions.

d. (Opcional) Configure um status específico, o nome do recurso da Amazon (ARN) da máquina de estado ou o ARN da execução. Para esse procedimento, selecione Status específicos e, em seguida, BEM-SUCEDIDO na lista suspensa.

9. Escolha Próximo. Isso abre a página Selecionar destinos.
10. Em Tipos de destino, mantenha a seleção padrão do serviço da AWS .
11. Na lista suspensa Selecionar um destino, escolha um AWS serviço. Por exemplo, você pode iniciar uma função do Lambda ou executar uma máquina de estado do Step Functions. Para esse procedimento, selecione máquina de estado do Step Functions.
12. Na lista suspensa Máquina de estado, escolha uma máquina de estado.
13. Em Perfil de execução, mantenha a seleção padrão de Criar uma nova função para este recurso específico.
14. Escolha Próximo. A página Configurar tags é aberta.
15. Escolha Próximo novamente. A página Revisar e criar é aberta.
16. Analise os detalhes da regra e selecione Criar regra.

    A regra é criada e a página de regras é exibida, listando todas as suas EventBridge regras da Amazon.

Gravando chamadas de API com AWS CloudTrail

AWS Step Functions é integrado com AWS CloudTrail, um serviço que fornece um registro das ações realizadas por um usuário, função ou um AWS service (Serviço da AWS). CloudTrail captura todas as chamadas de API Step Functions como eventos. As chamadas capturadas incluem chamadas do Step Functions console e chamadas de código para as operações Step Functions da API. Usando as informações coletadas por CloudTrail, você pode determinar a solicitação que foi feita Step Functions, o endereço IP do qual a solicitação foi feita, quando foi feita e detalhes adicionais.

Cada entrada de log ou evento contém informações sobre quem gerou a solicitação. As informações de identidade ajudam a determinar:

- Se a solicitação foi feita com credenciais de usuário raiz ou credenciais de usuário.
- Se a solicitação foi feita em nome de um usuário do Centro de Identidade do IAM.
- Se a solicitação foi feita com credenciais de segurança temporárias de um perfil ou de um usuário federado.

- Se a solicitação foi feita por outro AWS service (Serviço da AWS).

CloudTrail está ativo Conta da AWS quando você cria a conta e você tem acesso automático ao histórico de CloudTrail eventos. O histórico de CloudTrail eventos fornece um registro visível, pesquisável, baixável e imutável dos últimos 90 dias de eventos de gerenciamento registrados em um. Região da AWS Para obter mais informações, consulte Trabalhando com o histórico de CloudTrail eventos no Guia AWS CloudTrail do usuário. Não há CloudTrail cobrança pela visualização do histórico de eventos.

Para um registro contínuo dos eventos dos Conta da AWS últimos 90 dias, crie uma trilha ou um armazenamento de dados de eventos do CloudTrailLake.

CloudTrail trilhas

Uma trilha permite CloudTrail entregar arquivos de log para um bucket do Amazon S3. Todas as trilhas criadas usando o AWS Management Console são multirregionais. Só é possível criar uma trilha de região única ou de várias regiões usando a AWS CLI. É recomendável criar uma trilha multirregional porque você captura todas as atividades Regiões da AWS em sua conta. Se você criar uma trilha de região única, poderá visualizar somente os eventos registrados na Região da AWS da trilha. Para obter mais informações sobre trilhas, consulte Criar uma trilha para a Conta da AWS e Criar uma trilha para uma organização no Guia do usuário do AWS CloudTrail .

Você pode entregar uma cópia dos seus eventos de gerenciamento contínuos para o bucket do Amazon S3 sem nenhum custo CloudTrail criando uma trilha. No entanto, há cobranças de armazenamento do Amazon S3. Para obter mais informações sobre CloudTrail preços, consulte AWS CloudTrail Preços. Para receber informações sobre a definição de preço do Amazon S3, consulte Definição de preço do Amazon S3.

CloudTrail Armazenamentos de dados de eventos em Lake

CloudTrail O Lake permite que você execute consultas baseadas em SQL em seus eventos. CloudTrail O Lake converte eventos existentes no formato JSON baseado em linhas para o formato Apache ORC. O ORC é um formato colunar de armazenamento otimizado para recuperação rápida de dados. Os eventos são agregados em armazenamentos de dados de eventos, que são coleções imutáveis de eventos baseados nos critérios selecionados com a aplicação de seletores de eventos avançados. Os seletores que você aplica a um armazenamento de dados de eventos controlam quais eventos persistem e estão disponíveis para você consultar. Para obter mais informações sobre o CloudTrail Lake, consulte Trabalhando com o AWS CloudTrail Lake no Guia AWS CloudTrail do Usuário.

CloudTrail Os armazenamentos e consultas de dados de eventos em Lake incorrem em custos. Ao criar um armazenamento de dados de eventos, você escolhe a opção de preço que deseja usar para ele. A opção de preço determina o custo para a ingestão e para o armazenamento de eventos, e o período de retenção padrão e máximo para o armazenamento de dados de eventos. Para obter mais informações sobre CloudTrail preços, consulte AWS CloudTrail Preços.

Eventos de dados em CloudTrail

Os eventos de dados fornecem informações sobre as operações de recursos realizadas em um recurso (por exemplo, leitura ou gravação em um objeto do Amazon S3). Elas também são conhecidas como operações de plano de dados. Eventos de dados geralmente são atividades de alto volume. Por padrão, CloudTrail não registra eventos de dados. O histórico de CloudTrail eventos não registra eventos de dados.

Há cobranças adicionais para eventos de dados. Para obter mais informações sobre CloudTrail preços, consulte AWS CloudTrail Preços.

Você pode registrar eventos de dados para os tipos de Step Functions recursos usando o CloudTrail console ou AWS CLI as operações CloudTrail da API. Para obter mais informações sobre como registrar eventos de dados em log, consulte Registrar eventos de dados com o AWS Management Console e Registrar eventos de dados com a AWS Command Line Interface no Guia do usuário do AWS CloudTrail .

A tabela a seguir lista os tipos de Step Functions recursos para os quais você pode registrar eventos de dados. A coluna Tipo de evento de dados mostra o valor a ser escolhido na lista de tipos de eventos de dados no CloudTrail console. A coluna de valor resources.type mostra o **`resources.type`** valor, que você especificaria ao configurar seletores de eventos avançados usando as APIs ou. AWS CLI CloudTrail A CloudTrail coluna Data APIs logged to mostra as chamadas de API registradas CloudTrail para o tipo de recurso.

É possível configurar seletores de eventos avançados para filtrar os campos `eventName`, `readOnly` e `resources.ARN` para registrar somente os eventos que são importantes para você. Para obter mais informações sobre esses campos, consulte AdvancedFieldSelector na Referência de API do AWS CloudTrail .

Tipo de evento de dados	valor resources.type	APIs de dados registradas em CloudTrail
Máquina de estado do Step Functions	`AWS::StepFunctions::StateMachine`	• InvokeHTTPEndpoint

Eventos de gestão em CloudTrail

Os eventos de gerenciamento fornecem informações sobre as operações de gerenciamento que são realizadas nos recursos do seu Conta da AWS. Elas também são conhecidas como operações de plano de controle. Por padrão, CloudTrail registra eventos de gerenciamento.

State Machine (Máquina de estado)

- CreateStateMachine
- ListStateMachines
- DescribeStateMachine
- UpdateStateMachine
- DeleteStateMachine
- ValidateStateMachineDefinition
- TestState

Apelido de máquina de estado

- CreateStateMachineAlias
- ListStateMachineAliases
- DescribeStateMachineAlias
- UpdateStateMachineAlias
- DeleteStateMachineAlias

Versão do State Machine

- ListStateMachineVersions

- PublishStateMachineVersion
- DeleteStateMachineVersion

Execuções

- StartExecution
- StartSyncExecution
- RedriveExecution
- ListExecutions
- DescribeExecution
- GetExecutionHistory
- DescribeStateMachineForExecution
- StopExecution

Atividades

- CreateActivity
- ListActivities
- DescribeActivity
- DeleteActivity
- GetActivityTask

Token de tarefa

- SendTaskSuccess
- SendTaskHeartbeat
- SendTaskFailure

MapRun

- ListMapRuns
- DescribeMapRun
- UpdateMapRun

Tags

- ListTagsForResource
- TagResource
- UntagResource

Exemplos de evento

Um evento representa uma única solicitação de qualquer fonte e inclui informações sobre a operação de API solicitada, a data e a hora da operação, os parâmetros da solicitação e assim por diante. CloudTrail os arquivos de log não são um rastreamento de pilha ordenado das chamadas públicas de API, portanto, os eventos não aparecem em nenhuma ordem específica.

O exemplo a seguir mostra um evento CloudTrail de dados que demonstra`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"
}
```

O exemplo a seguir mostra um evento CloudTrail de gerenciamento que demonstra a `CreateStateMachine` operação.

```
{
    "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 obter informações sobre o conteúdo do CloudTrail registro, consulte o conteúdo do CloudTrail registro no Guia AWS CloudTrail do usuário.

Como registrar usando o CloudWatch Logs

Os Fluxos de trabalho Padrão registram o histórico de execuções no AWS Step Functions, embora você possa opcionalmente configurar o registro em log no Amazon CloudWatch Logs.

Ao contrário dos fluxos de trabalho padrão, os fluxos de trabalho expressos não registram o histórico de execuções no AWS Step Functions. Para ver o histórico de execução e os resultados de um Fluxo de trabalho expressoo, é necessário configurar o registro em log para o Amazon CloudWatch Logs. A publicação de logs não bloqueia nem diminui as execuções.

> **Note**
>
> Quando você configura o registro em log, as cobranças do CloudWatch Logs serão aplicadas e você será cobrado de acordo com a taxa de logs vendidos. Para obter mais informações, consulte Logs vendidos na guia Logs na Definição de preço do CloudWatch.

Configurar registro em log da

Quando você criar um Fluxo de trabalho Padrão usando o console do Step Functions, ele não será configurado para ativar o registro em log no CloudWatch Logs. Um Fluxo de trabalho expressoo criado com o console do Step Functions será configurado por padrão para ativar o registro em log no CloudWatch Logs.

Para fluxos de trabalho expressoos, o Step Functions pode criar um perfil com a política do AWS Identity and Access Management (IAM) necessária para o CloudWatch Logs. Se você criar um Fluxo de trabalho Padrão ou um Fluxo de trabalho expressoo usando a API, a CLI ou o AWS CloudFormation, o Step Functions não ativará o registro em log por padrão, e você precisará garantir que o perfil tenha as permissões necessárias.

Para cada execução iniciada no console, o Step Functions fornece um link para o CloudWatch Logs, configurado com o filtro correto para buscar eventos de log específicos para essa execução.

Para configurar o log, você pode transmitir o parâmetro LoggingConfiguration ao usar CreateStateMachine ou UpdateStateMachine. É possível analisar ainda mais os dados no CloudWatch Logs usando o CloudWatch Logs Insights. Para obter mais informações, consulte Analisar os dados de log com o CloudWatch Logs Insights.

Payloads do CloudWatch Logs

Eventos do histórico de execução podem conter propriedades de entrada ou saída nas definições. Se a entrada ou saída de escape enviada ao CloudWatch Logs exceder 248 KB, ela será truncada como resultado das cotas do CloudWatch Logs.

- Você pode determinar se um payload foi truncado revisando as propriedades `inputDetails` e `outputDetails`. Para ver mais informações, consulte o Tipo de dados de `HistoryEventExecutionDataDetails`.
- Para fluxos de trabalho padrão, você pode ver o histórico completo de execução usando `GetExecutionHistory`.
- O `GetExecutionHistory` não está disponível para fluxos de trabalho expressos. Se quiser ver a entrada e a saída completas, você poderá usar ARNs do Amazon S3. Para obter mais informações, consulte the section called “Use ARNs do Amazon S3 em vez de transmitir grandes cargas”.

Políticas do IAM para registro em log no CloudWatch Logs

Você também precisará configurar o perfil do IAM de execução da máquina de estado para ter a permissão adequada para registrar no CloudWatch Logs, conforme mostrado no exemplo a seguir.

Exemplo de política do IAM

O exemplo a seguir é de uma política que você pode utilizar para configurar as permissões. Conforme mostrado no exemplo a seguir, você precisa especificar * no campo `Resource` porque

as ações da API do CloudWatch, como CreateLogDelivery e DescribeLogGroups, não oferecem suporte aos tipos de recursos definidos por Amazon CloudWatch Logs. Para obter mais informações, consulte Ações definidas pelo Amazon CloudWatch Logs.

- Para obter informações sobre recursos CloudWatch, consulte Recursos e operações do CloudWatch Logs no Guia do usuário do Amazon CloudWatch.
- Para obter informações sobre as permissões que você precisa para configurar o envio de logs para o CloudWatch Logs, consulte Permissões de usuário na seção intitulada Logs enviados para o 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": "*"
        }
    ]
}
```

Não é possível acessar os CloudWatch Logs

Se você não conseguir acessar os CloudWatch Logs, certifique-se de ter feito o seguinte:

1. Configurado o perfil do IAM de execução da máquina de estado para ter a permissão adequada para registrar no CloudWatch Logs.

Se você estiver usando as solicitações CreateStateMachine ou UpdateStateMachine, certifique-se de ter especificado o perfil do IAM no parâmetro `roleArn` que contém as permissões, conforme mostrado no exemplo anterior.

2. Verificado se a política de recursos do CloudWatch Logs não excede o limite de 5.120 caracteres para as políticas de recursos do CloudWatch Logs.

   Se você excedeu o limite de caracteres, remova permissões desnecessárias da política de recursos do CloudWatch Logs ou prefixe o nome do grupo de logs com `/aws/vendedlogs`, o que concederá permissões ao grupo de logs sem acrescentar mais caracteres à política de recursos. Os nomes dos grupos de logs criados no console do Step Functions têm o prefixo `/aws/vendedlogs/states`. Para obter mais informações, consulte Restrições de tamanho de políticas de recursos do Amazon CloudWatch Logs.

Níveis de log

Você pode escolher entre `OFF`, `ALL`, `ERROR` ou `FATAL`. Nenhum tipo de evento é registrado quando definido como `OFF` e todos os tipos de evento são registrados quando definidos como `ALL`. Para `ERROR` e `FATAL`, consulte a tabela a seguir.

Para obter mais informações sobre os dados de execução exibidos para as execuções do Fluxo de trabalho expressoo com base nesses Níveis de log, consulte Execuções de Fluxo de trabalho Padrão e Expresso no console.

Tipo de evento	**ALL**	**ERROR**	**FATAL**	**OFF**
ChoiceStateEntered	✓			
ChoiceStateExited	✓			
ExecutionAborted	✓	✓	✓	
ExecutionFailed	✓	✓	✓	
ExecutionStarted	✓			

Tipo de evento	ALL	ERROR	FATAL	OFF
Execution Succeeded	✓			
Execution TimedOut	✓	✓	✓	
FailStateEntered	✓	✓		
LambdaFunctionFailed	✓	✓		
LambdaFunctionScheduled	✓			
LambdaFunctionScheduleFailed	✓	✓		
LambdaFunctionStarted	✓			
LambdaFunctionStartFailed	✓	✓		
LambdaFunctionSucceeded	✓			
LambdaFunctionTimedOut	✓	✓		
MapIterationAborted	✓	✓		
MapIterationFailed	✓	✓		
MapIterationStarted	✓			

Tipo de evento	ALL	ERROR	FATAL	OFF
MapIterationSucceeded	✓			
MapRunAborted	✓	✓		
MapRunFailed	✓	✓		
MapStateAborted	✓	✓		
MapStateEntered	✓			
MapStateExited	✓			
MapStateFailed	✓	✓		
MapStateStarted	✓			
MapStateSucceeded	✓			
ParallelStateAborted	✓	✓		
ParallelStateEntered	✓			
ParallelStateExited	✓			
ParallelStateFailed	✓	✓		
ParallelStateStarted	✓			

Tipo de evento	ALL	ERROR	FATAL	OFF
ParallelStateSucceeded	✓			
PassStateEntered	✓			
PassStateExited	✓			
SucceedStateEntered	✓			
SucceedStateExited	✓			
TaskFailed	✓	✓		
TaskScheduled	✓			
TaskStarted	✓			
TaskStartFailed	✓	✓		
TaskStateAborted	✓	✓		
TaskStateEntered	✓			
TaskStateExited	✓			
TaskSubmitFailed	✓	✓		
TaskSubmitted	✓			
TaskSucceeded	✓			
TaskTimedOut	✓	✓		

Tipo de evento	ALL	ERROR	FATAL	OFF
WaitState Aborted	✓	✓		
WaitState Entered	✓			
WaitStateExited	✓			

AWS X-Ray e Step Functions

Você pode usar o AWS X-Ray para visualizar os componentes da máquina de estado, identificar gargalos de desempenho e solucionar problemas de solicitações que resultaram em erros. A máquina de estado envia dados de rastreamento ao X-Ray que, então, processa os dados para gerar um mapa de serviço e resumos de rastreamento pesquisáveis.

Com o X-Ray ativado para sua máquina de estado, você pode rastrear solicitações conforme elas são executadas no Step Functions, em todas as AWS regiões onde o X-Ray está disponível. Isso fornece uma visão geral detalhada de toda uma solicitação do Step Functions. O Step Functions enviará rastreamentos ao X-Ray para execuções de máquina de estado, mesmo quando um ID de rastreamento não é transmitido por um serviço upstream. Você pode usar um mapa do serviço X-Ray para visualizar a latência de uma solicitação, incluindo quaisquer AWS serviços integrados ao X-Ray. Também é possível configurar regras de amostragem para informar ao X-Ray quais solicitações registrar e com quais taxas de amostragem, de acordo com os critérios especificados.

Quando o X-Ray não está ativado para a máquina de estado, e um serviço upstream não passa um ID de rastreamento, o Step Functions não envia rastreamentos ao X-Ray para execuções de máquinas de estado. Entretanto, se o ID de rastreamento for transmitido por um serviço upstream, o Step Functions enviará rastreamentos ao X-Ray para execuções de máquina de estado.

Você pode usar AWS X-Ray com Step Functions em regiões onde ambos são suportados. Consulte as páginas de endpoints e cotas do Step Functions e do X-Ray para obter informações sobre o suporte regional para X-Ray e Step Functions.

Cotas combinadas de X-Ray e Step Functions

Você pode adicionar dados a um rastreamento por até sete dias e consultar dados de rastreamento que remontam a trinta dias, o período em que o X-Ray armazena os dados de rastreamento. Os rastreamentos estarão sujeitos às cotas de X-Ray. Além de outras cotas, o X-Ray fornece um tamanho mínimo de rastreamento garantido de 100 KB para máquinas de estado do Step Functions. Se mais de 100 KB de dados de rastreamento forem fornecidos ao X-Ray, isso poderá resultar em um rastreamento congelado. Consulte a seção de service quotas da página Endpoints e cotas do X-Ray para obter mais informações sobre outras cotas do X-Ray.

Important

O Step Functions não oferece suporte ao rastreamento do X-Ray para as execuções do fluxo de trabalho secundário iniciadas por um estado de Mapa distribuído porque é fácil exceder o Limite de tamanho do documento de rastreamento para essas execuções.

Tópicos

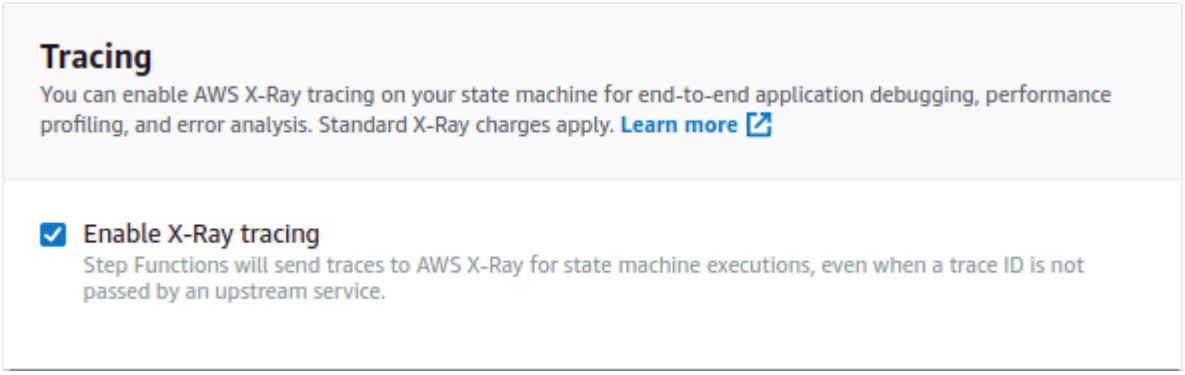
Definição e configuração

Ative o rastreamento do X-Ray ao criar uma máquina de estado

Você pode ativar o rastreamento do X-Ray ao criar uma nova máquina de estado selecionando Ativar rastreamento do X-Ray na página Especificar detalhes.

1. Abra o console do Step Functions e clique em Criar máquina de estado.
2. Na página Escolher método de criação, escolha uma opção apropriada para criar a máquina de estado. Se você escolher Executar um projeto de amostra, não poderá ativar o rastreamento do X-Ray durante a criação da máquina de estado e precisará ativar o rastreamento do X-Ray após a criação da máquina de estado. Para obter mais informações sobre como ativar o X-Ray em uma máquina de estado existente, consulte Ativar o X-Ray em uma máquina de estado existente.

   Escolha Próximo.
3. Na página Especificar detalhes, configure a máquina de estado.
4. Escolha Ativar rastreamento do X-Ray.



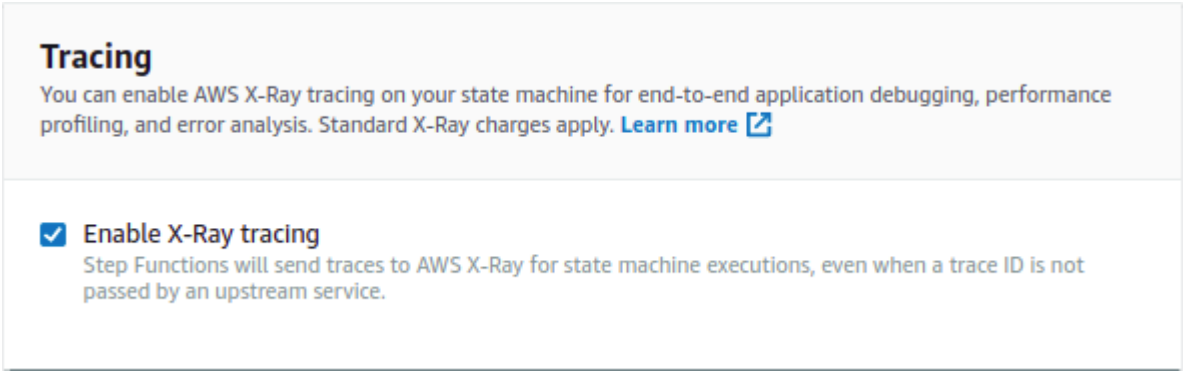
A máquina de estado do Step Functions agora enviará rastreamentos ao X-Ray para execuções de máquinas de estado.

> **Note**
>
> Se você optar por usar um perfil do IAM existente, você deverá garantir que as gravações no X-Ray sejam permitidas. Para obter mais informações sobre as permissões de que você precisa, consulte as Políticas do IAM para o X-Ray.

Ativar o X-Ray em uma máquina de estado existente

Para ativar o X-Ray em uma máquina de estado existente:

1. No console do Step Functions, selecione a máquina de estado para a qual você deseja ativar o rastreamento.
2. Selecione a opção Editar.
3. Escolha Ativar rastreamento do X-Ray.



Você verá uma notificação informando que talvez seja necessário fazer alterações adicionais.

> **Note**
>
> Ao ativar o X-Ray para uma máquina de estado existente, você terá que se certificar de que tem uma política do IAM com permissões suficientes para que o X-Ray execute rastreamentos. Você pode adicionar uma manualmente ou gerar uma. Para obter mais informações, consulte a seção Política do IAM para o Políticas do IAM para AWS X-Ray.

4. (Opcional) Gere automaticamente um novo perfil para a máquina de estado a fim de incluir permissões do X-Ray.
5. Escolha Salvar.

Configurar rastreamento do X-Ray para o Step Functions

Quando você executa pela primeira vez uma máquina de estado com o X-Ray Tracing ativado, ela usa os valores de configuração padrão para o X-Ray Tracing. AWS X-Ray não coleta dados para cada solicitação enviada a um aplicativo. Em vez disso, ele coleta dados para um número estatisticamente significativo de solicitações. O padrão é registrar a primeira solicitação a cada segundo e cinco por cento de todas as solicitações adicionais. Uma solicitação por segundo é

o reservatório. Isso garante que pelo menos um rastreamento seja registrado a cada segundo à medida que o serviço atende às solicitações. Cinco por cento é a taxa segundo a qual as solicitações adicionais, além do tamanho de reservatório, são amostradas.

Para evitar cobranças incorridas para o serviço quando você está começando a usá-lo, a taxa de amostragem padrão é conservadora. Você pode configurar o X-Ray para modificar a regra de amostragem padrão e configurar regras adicionais que aplicam a amostragem com base nas propriedades do serviço ou da solicitação.

Por exemplo, talvez você queira desativar a amostragem e rastrear todas as solicitações de chamadas que modificam o estado, o identificador Contas da AWS ou as transações. Para chamadas somente leitura de alto volume, como a sondagem de plano de fundo, as verificações de integridade ou a manutenção da conexão, você pode coletar amostras a uma taxa baixa e ainda obter dados suficientes para observar quaisquer problemas que ocorram.

Para configurar uma regra de amostragem para a máquina de estado:

1. Acesse o console do X-Ray.
2. Escolha Sampling (Amostragem).
3. Para criar uma regra, escolha Criar regra de amostragem.

   Para editar uma regra, escolha o nome de uma regra.

   Para excluir uma regra, escolha uma regra e use o menu Actions (Ações) para excluí-la.

Algumas partes das regras de amostragem existentes, como o nome e a prioridade, não podem ser alteradas. Em vez disso, adicione ou clone uma regra existente, faça as alterações desejadas e use a nova regra.

Para obter informações detalhadas sobre as regras de amostragem do X-Ray e como configurar os vários parâmetros, consulte Configurando regras de amostragem no console do X-Ray.

Integrar serviços upstream

Para integrar a execução dos fluxos de trabalho do Step Functions, como fluxos de trabalho padrão, Synchronous e padrão, com um serviço upstream, você precisa definir o `traceHeader`. Isso é feito automaticamente para você se você estiver usando uma API HTTP no API Gateway. No entanto, se você estiver usando uma função do Lambda e/ou um SDK, precisará configurar você mesmo o `traceHeader` nas chamadas de API `StartExecution` ou `StartSyncExecution`.

É necessário especificar o formato `traceHeader` como `\p{ASCII}#`. Além disso, para permitir que o Step Functions use o mesmo ID de rastreamento, você deve especificar o formato como `Root={TRACE_ID};Sampled={1 or 0}`. Se você estiver usando uma função do Lambda, substitua o `TRACE_ID`pelo ID de rastreamento no segmento atual e defina o campo Amostragem como 1 se o modo de amostragem for true e `0` se o modo de amostragem for false. Fornecer o ID de rastreamento nesse formato garante que você obtenha um rastreamento completo.

Veja a seguir um exemplo escrito em Python para mostrar como especificar o `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)
```

Conceitos

O console do X-Ray

O AWS X-Ray console permite que você visualize mapas e rastreamentos de serviços para solicitações atendidas por seus aplicativos. Você pode acessar o console para ver informações detalhadas coletadas pelo X-Ray quando ele está ativado para a máquina de estado.

Consulte Como visualizar o console do X-Ray para obter informações sobre como acessar o console do X-Ray para execuções de máquina de estado.

Para obter informações detalhadas sobre o console do X-Ray, consulte a documentação do console do X-Ray.

Segmentos, subsegmentos e rastreamentos

Um segmento registra informações sobre uma solicitação na máquina de estado. Ele contém informações como o trabalho que a máquina de estado executa e também pode conter subsegmentos com informações sobre chamadas downstream.

Um rastreamento coleta todos os segmentos gerados por uma única solicitação.

Amostragem

Para garantir um rastreamento eficiente e fornecer uma amostra representativa das solicitações atendidas pela aplicação, o X-Ray aplica um algoritmo de amostragem para determinar quais solicitações são rastreadas. Isso pode ser alterado editando as regras de amostragem.

Indicadores

Para a máquina de estado, o X-Ray medirá o tempo de invocação, o tempo de transição de estado, o tempo geral de execução do Step Functions e as variações desse tempo de execução. Essas informações podem ser acessadas por meio do console do X-Ray.

Análises

O console do AWS X-Ray Analytics é uma ferramenta interativa para interpretar dados de rastreamento. É possível refinar o conjunto de dados ativo com filtros cada vez mais granulares clicando nos gráficos e nos painéis de métricas e campos que estão associados ao conjunto de rastreamentos atual. Isso permite analisar o desempenho da máquina de estado, além de localizar e identificar rapidamente problemas de desempenho.

Para obter informações detalhadas sobre a análise de X-Ray, consulte Interação com o console do AWS X-Ray Analytics

Integrações de serviços do Step Functions e X-Ray

Alguns dos AWS serviços que se integram ao Step Functions oferecem integração com AWS X-Ray a adição de um cabeçalho de rastreamento às solicitações, a execução do daemon X-Ray ou a tomada de decisões de amostragem e o upload de dados de rastreamento para o X-Ray. Outros devem ser instrumentados usando o AWS X-Ray SDK. Alguns ainda não oferecem suporte à integração com o X-Ray. A integração do X-Ray é necessária para fornecer dados de rastreamento completos ao usar uma integração de serviços com o Step Functions

Suporte nativo ao X-Ray

As integrações de serviços com suporte nativo ao X-Ray incluem:

- Amazon Simple Notification Service
- Amazon Simple Queue Service
- AWS Lambda
- AWS Step Functions

Instrumentação necessária

Integrações de serviços que exigem instrumentação do X-Ray:

- Amazon Elastic Container Service
- AWS Batch
- AWS Fargate

Somente rastreamento do lado do cliente

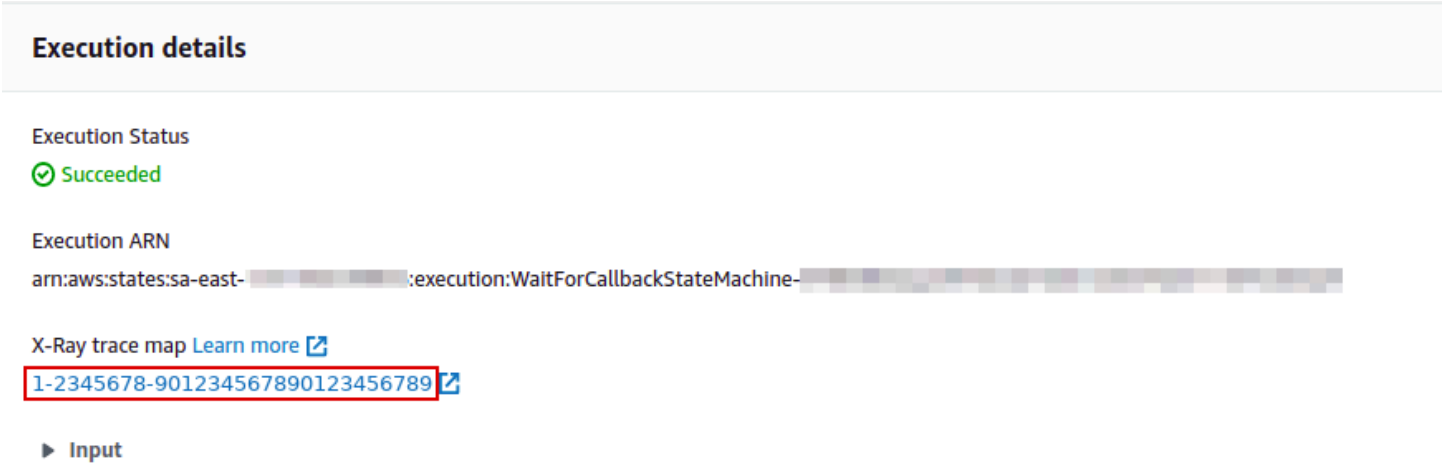
Outras integrações de serviço não são compatíveis com rastreamentos do X-Ray. No entanto, os rastreamentos do lado do cliente ainda podem ser coletados:

- Amazon DynamoDB
- Amazon EMR
- Amazon SageMaker
- AWS CodeBuild
- AWS Glue

Como visualizar o console do X-Ray

O X-Ray recebe dados de serviços como segmentos. Em seguida, o X-Ray agrupa segmentos que tenham uma solicitação em comum em rastreamentos. O X-Ray processa os rastreamentos para gerar um gráfico de serviço que apresenta uma representação visual da aplicação.

Depois de iniciar a execução da máquina de estado, você pode visualizar os rastreamentos do X-Ray escolhendo o link do mapa de rastreamento do X-Ray na seção Detalhes da execução.



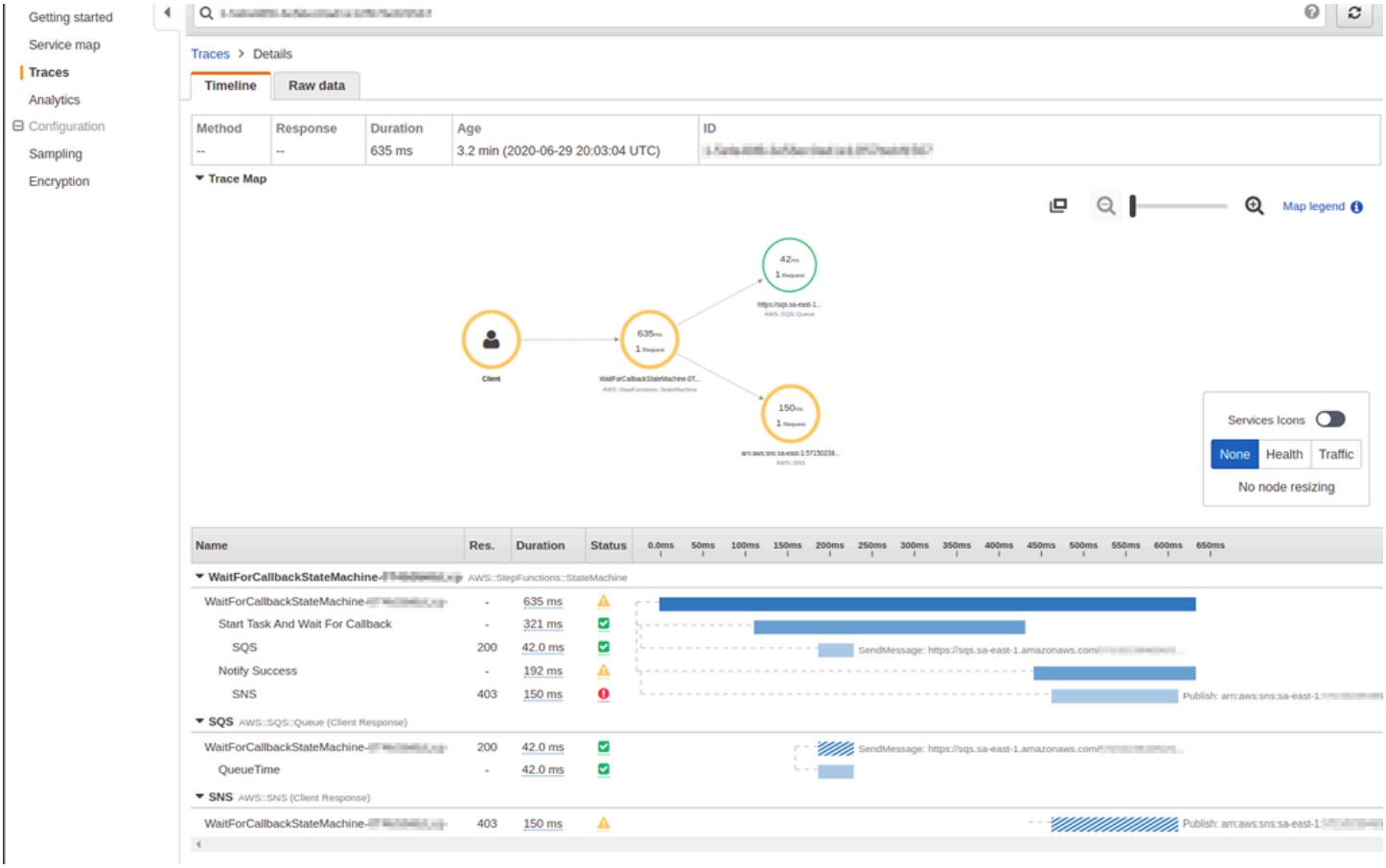
Depois de ativar o X-Ray para a máquina de estado, as informações de rastreamento das execuções podem ser visualizadas no console do X-Ray.

Como visualizar informações de rastreamento do X-Ray para o Step Functions

As seguintes etapas ilustram o tipo de informação que você pode ver no console depois de ativar o X-Ray e realizar uma execução. São mostrados rastreamentos do X-Ray para o projeto de amostra Exemplo de Padrão de Retorno de Chamada (Amazon SQS, Amazon SNS, Lambda) .

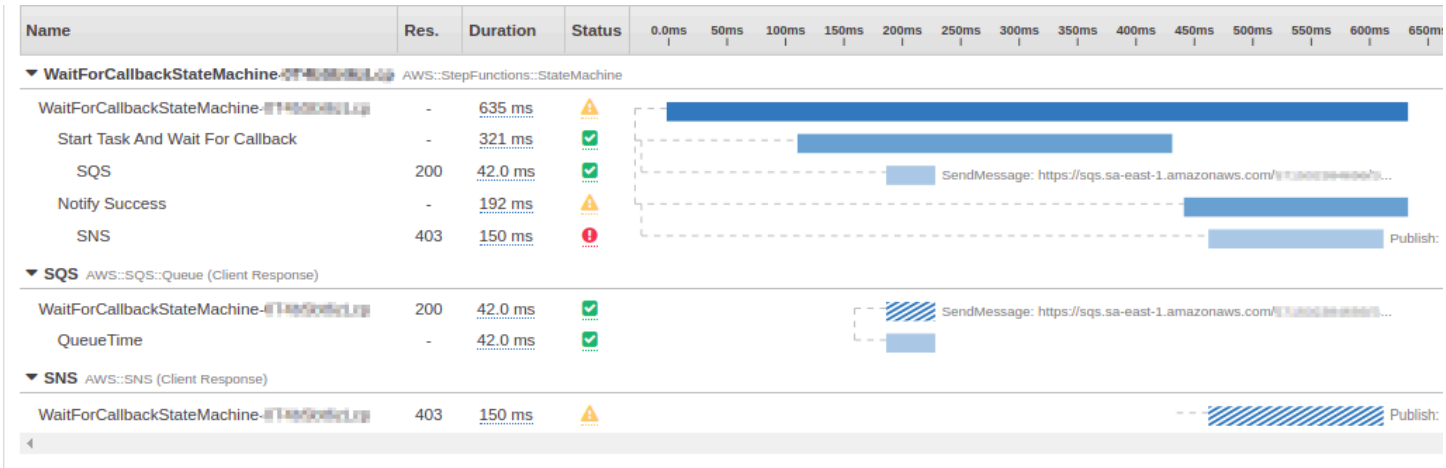
Rastreamentos

Depois que uma execução for concluída, você poderá navegar até o console do X-Ray, onde verá a página Rastreamentos do X-Ray. Isso exibe uma visão geral do mapa de serviços, bem como informações de rastreamento e segmento para a máquina de estado.

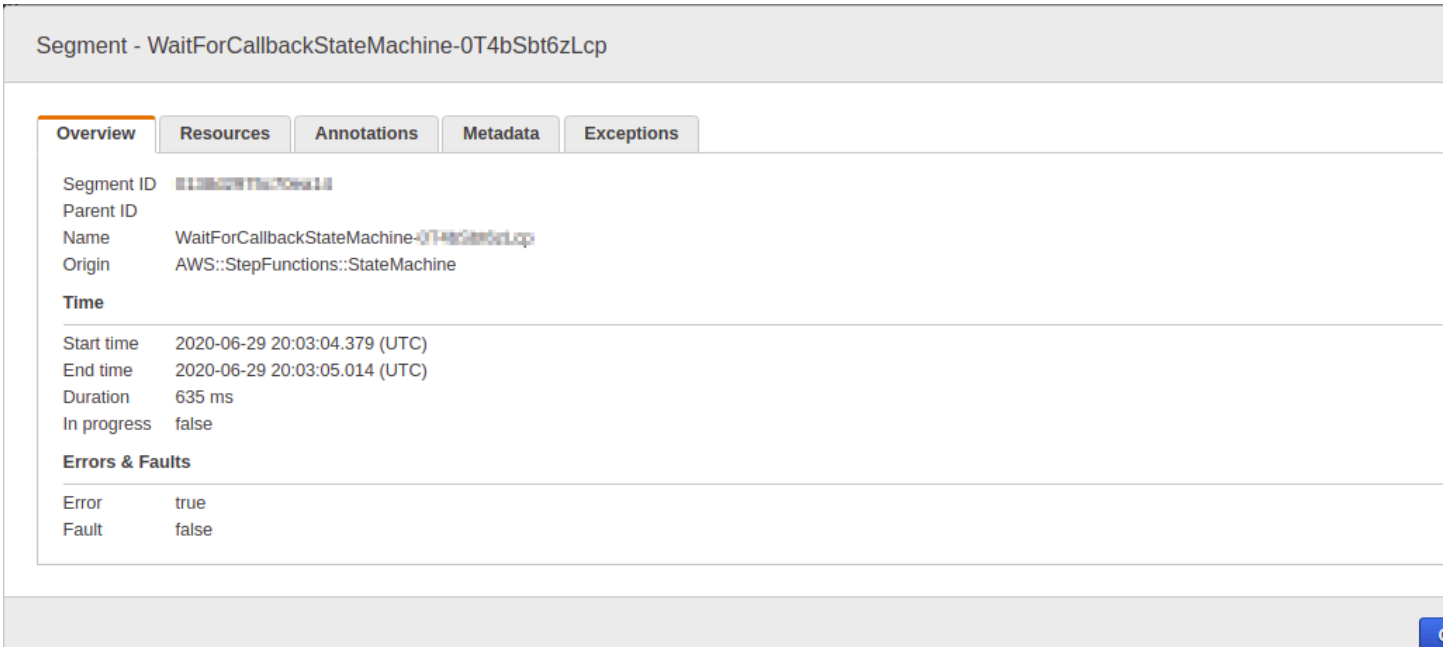


Mapa de serviço

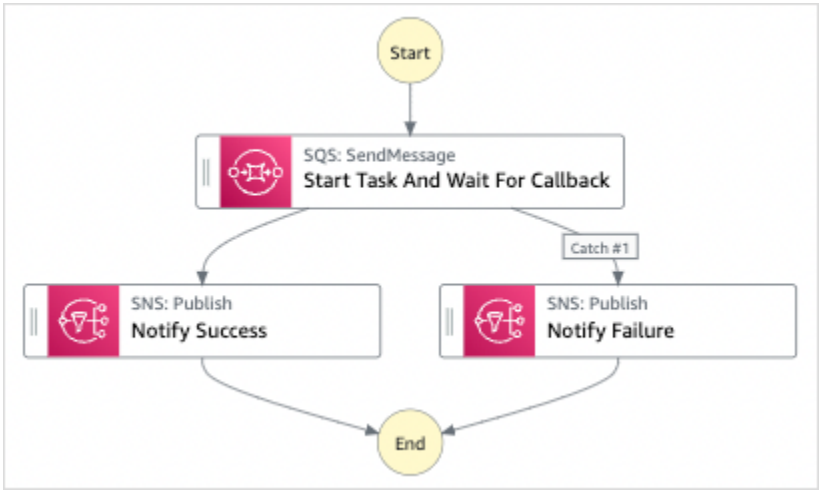
O mapa de serviço no console do X-Ray ajuda a identificar serviços nos quais estejam ocorrendo erros, nos quais haja conexões com alta latência ou ver rastreamentos para solicitações que foram malsucedidas.



No mapa de rastreamento, você pode escolher um nó de serviço para visualizar as solicitações desse nó ou uma borda entre dois nós para visualizar as solicitações que percorreram essa conexão. Aqui, o nó `WaitForCallBack` foi selecionado e você pode ver informações adicionais sobre o status de execução e resposta.



Você pode ver como o mapa de serviço do X-Ray se correlaciona com a máquina de estado. Há um nó de mapa de serviço para cada integração de serviço que é chamado pelo Step Functions, desde que ele seja compatível com o X-Ray.



Segmentos e subsegmentos

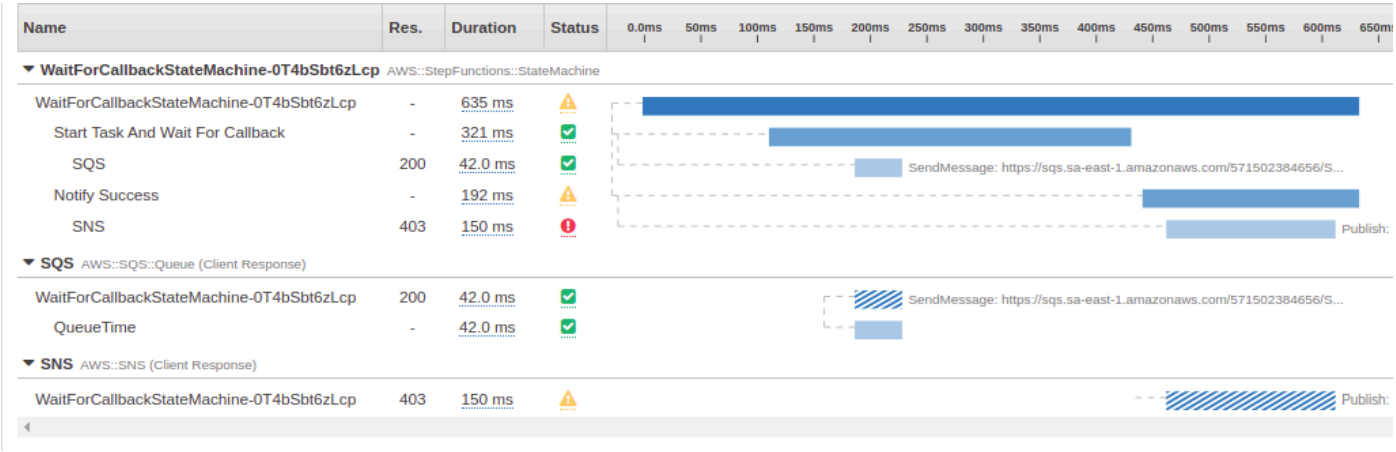
Um rastreamento coleta todos os segmentos gerados por uma única solicitação. Cada segmento fornece o nome do recurso, os detalhes sobre a solicitação e os detalhes sobre o trabalho realizado.

Na página Rastreamentos, você pode ver os segmentos e, se expandidos, os subsegmentos correspondentes. Você pode escolher um segmento ou subsegmento para ver informações detalhadas sobre ele.

Escolha cada uma das guias para ver como as informações de segmentos e subsegmentos são exibidas.

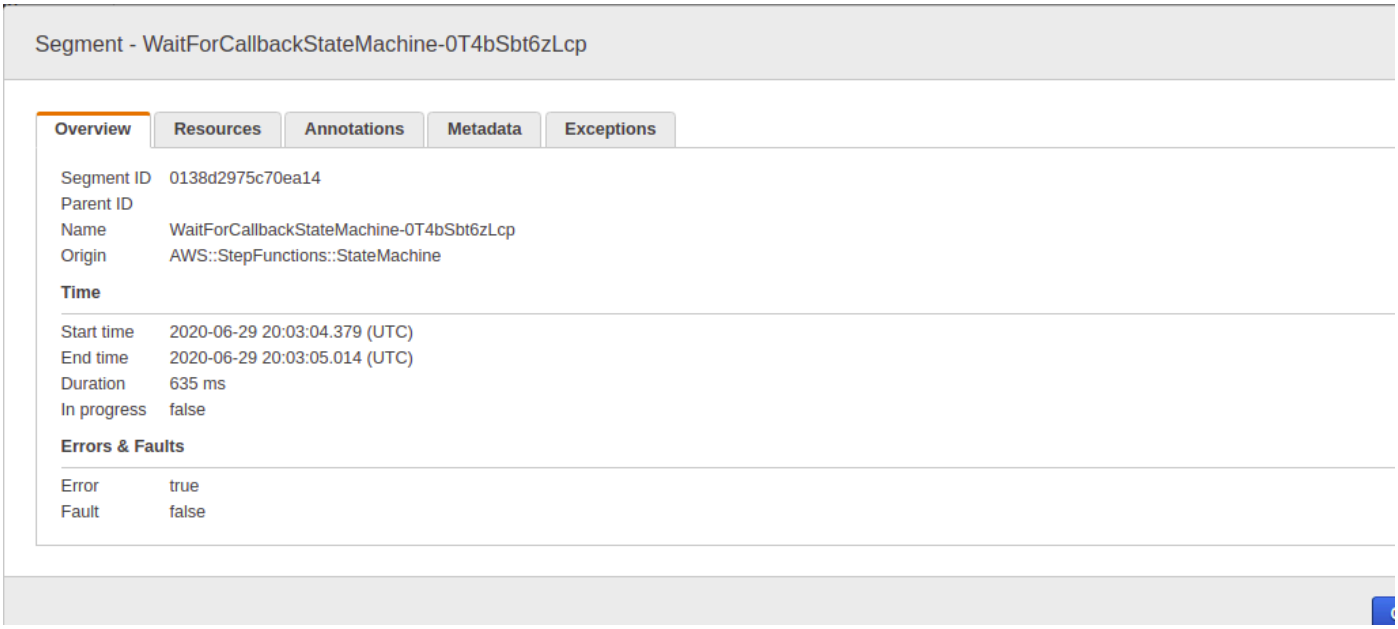
Overview of Segments

Uma visão geral dos segmentos e subsegmentos dessa máquina de estado. Há um segmento diferente para cada nó no mapa de serviço.



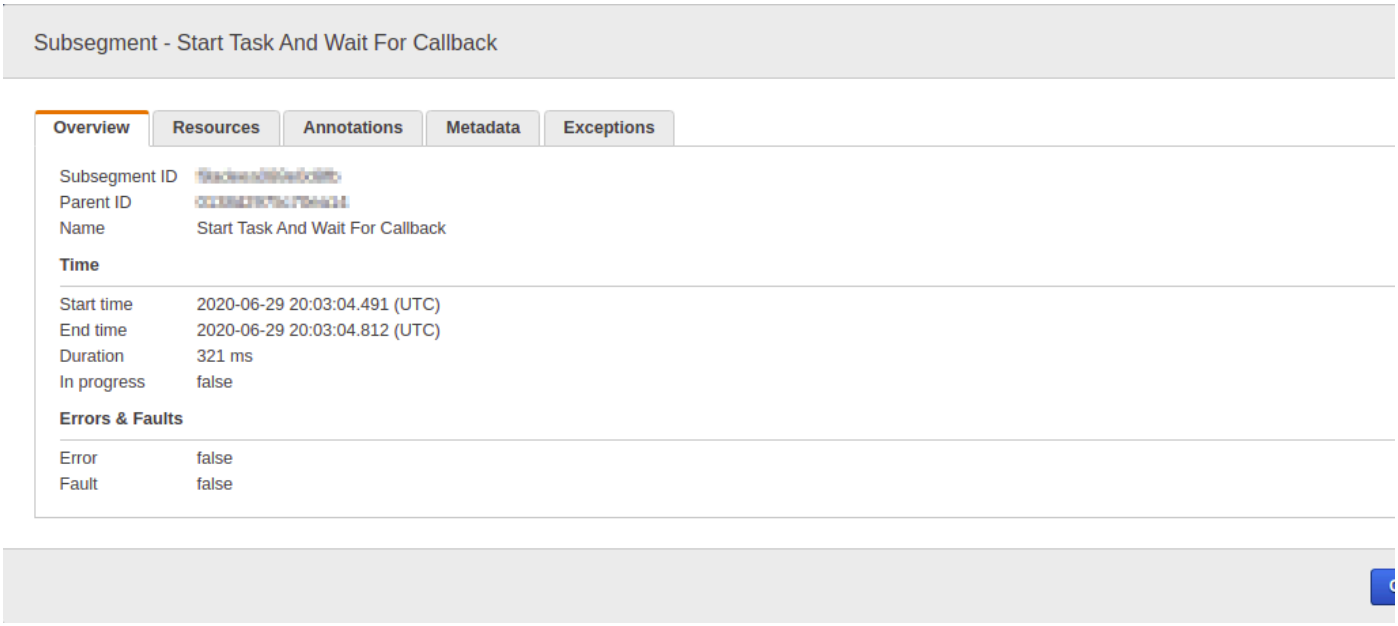
View segment detail

Escolher um segmento fornece o nome do recurso, os detalhes sobre a solicitação e os detalhes sobre o trabalho realizado.



View subsegment detail

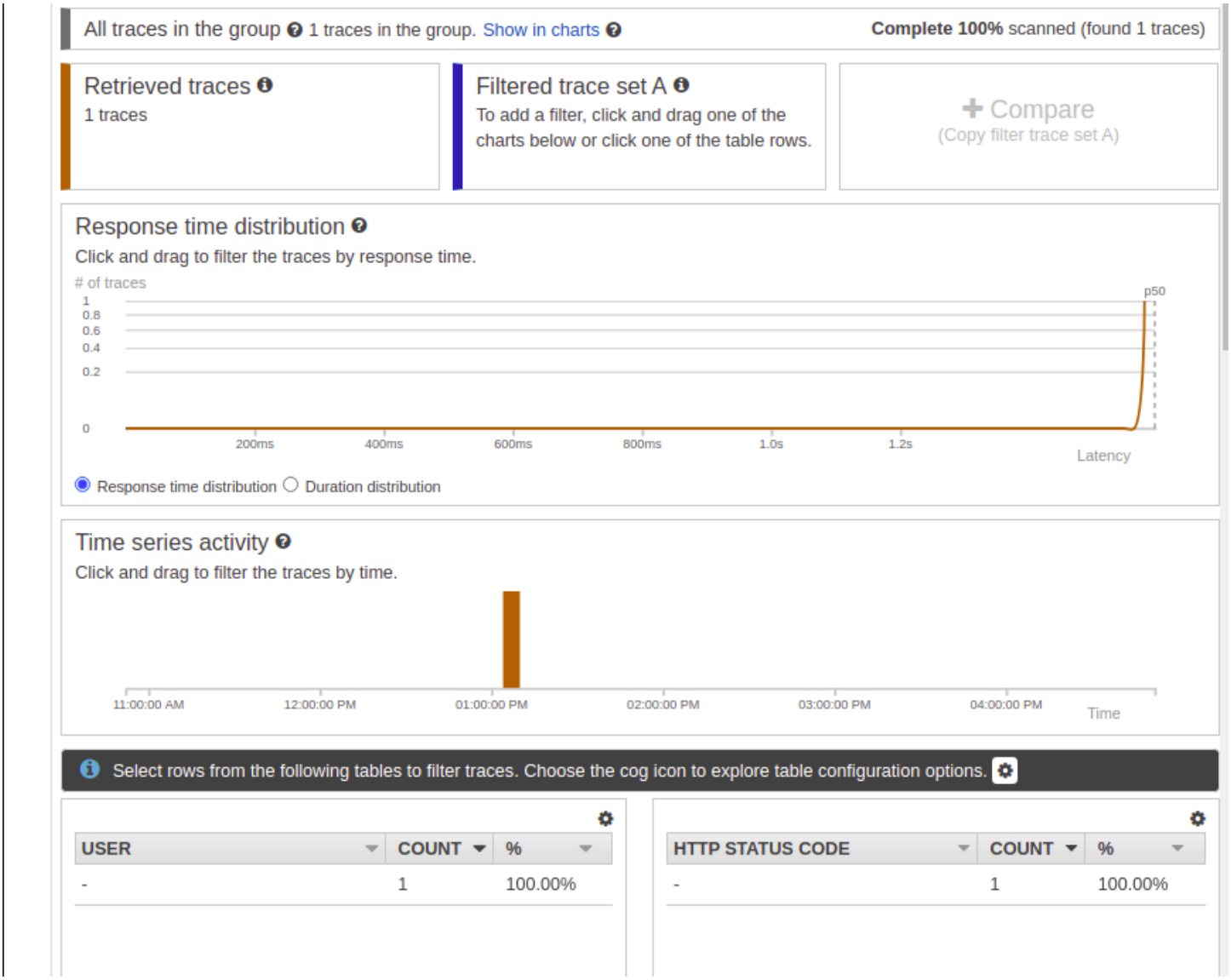
Um segmento pode dividir os dados sobre o trabalho feito em subsegmentos. A escolha de um subsegmento permite que você visualize informações e detalhes de temporização mais granulares. Um subsegmento pode conter detalhes adicionais sobre uma chamada para um AWS serviço, uma API HTTP externa ou um banco de dados SQL.



Análises

O console do AWS X-Ray Analytics é uma ferramenta interativa para interpretar dados de rastreamento. Você pode usar isso para entender mais facilmente o desempenho da máquina de estado. O console permite explorar, analisar e visualizar rastreamentos por meio de gráficos interativos de tempo de resposta e de séries temporais. Isso pode ajudar você a localizar rapidamente problemas de desempenho e latência.

É possível refinar o conjunto de dados ativo com filtros cada vez mais granulares clicando nos gráficos e nos painéis de métricas e campos que estão associados ao conjunto de rastreamentos atual.

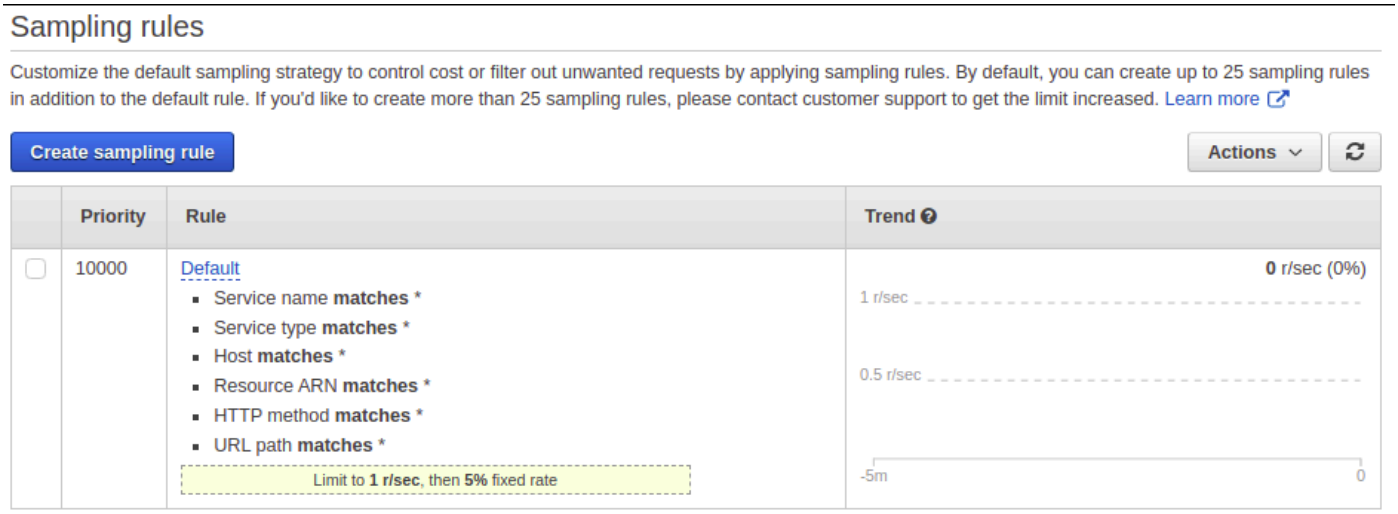


Configuração

Você pode configurar opções de amostragem e criptografia no console do X-Ray.

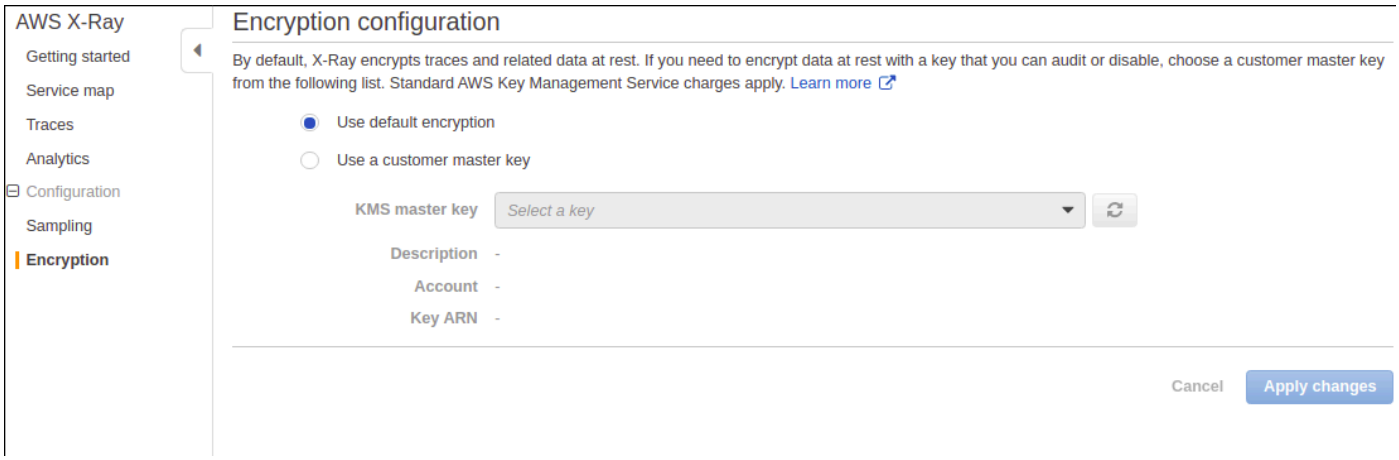
Sampling

Escolha Amostragem para ver detalhes sobre a taxa de amostragem e a configuração. Você pode alterar as regras de amostragem para controlar a quantidade de dados gravados e modificar o comportamento de amostragem para atender às suas necessidades específicas.



Encryption

Escolha Criptografia para modificar as configurações de criptografia. Você pode usar a configuração padrão, na qual o X-Ray criptografa os rastreamentos e a data em repouso ou, se necessário, pode escolher uma chave mestra do cliente. As taxas de AWS KMS padrão se aplicam no último caso.



E se não houver dados no mapa de rastreamento ou no mapa de serviço?

Se você tiver ativado o X-Ray, mas não conseguir ver nenhum dado no console do X-Ray, verifique se:

- Os perfis do IAM estão configurados corretamente para permitir a gravação no X-Ray.
- As regras de amostragem permitem a amostragem de dados.

- Como pode haver um pequeno atraso até que os perfis do IAM recém-criados ou modificados sejam aplicados, verifique o rastreamento ou os mapas de serviço novamente após alguns minutos.
- Se você ver Data Not Found no painel X-Ray Traces, verifique as configurações da sua conta do IAM e verifique se ela AWS Security Token Service está ativada para a região pretendida. Para obter mais informações, consulte Ativação e desativação AWS STSRegião da AWS em um Guia do usuário do IAM.

Usar o Notificações de Usuários da AWS com o AWS Step Functions

É possível usar o Notificações de Usuários da AWS para configurar canais de entrega para receber notificações sobre eventos do AWS Step Functions. Você recebe uma notificação quando um evento corresponde a uma regra especificada. É possível receber notificações de eventos por meio de vários canais, incluindo e-mail, notificações de chat do AWS Chatbot ou notificações por push do AWS Console Mobile Application. Você também pode visualizar notificações na Central de Notificações do Console. O Notificações de Usuários oferece é compatível com agregação, o que pode reduzir o número de notificações que você recebe durante eventos específicos.

Segurança em AWS Step Functions

Esta seção fornece informações sobre AWS Step Functions segurança e autenticação.

O Step Functions usa o IAM para controlar o acesso a outros AWS serviços e recursos. Para obter uma visão geral de como o IAM funciona, consulte Visão geral do gerenciamento de acesso no Guia do usuário do IAM. Para obter uma visão geral das credenciais de segurança, consulte Credenciais de segurança da AWS na Referência geral da Amazon Web Services.

Proteção de dados em AWS Step Functions

O modelo de responsabilidade AWS compartilhada modelo se aplica à proteção de dados em AWS Step Functions. Conforme descrito neste modelo, AWS é responsável por proteger a infraestrutura global que executa todos os Nuvem AWS. Você é responsável por manter o controle sobre seu conteúdo hospedado nessa infraestrutura. Você também é responsável pelas tarefas de configuração e gerenciamento de segurança dos Serviços da AWS que usa. Para obter mais informações sobre a privacidade de dados, consulte as Perguntas Frequentes sobre Privacidade de Dados.. Para obter mais informações sobre a proteção de dados na Europa, consulte a postagem do blog AWS LGPD e Modelo de Responsabilidade Compartilhada no AWS Blog de Segurança.

Para fins de proteção de dados, recomendamos que você proteja Conta da AWS as credenciais e configure usuários individuais com AWS IAM Identity Center ou AWS Identity and Access Management (IAM). Dessa maneira, cada usuário receberá apenas as permissões necessárias para cumprir suas obrigações de trabalho. Recomendamos também que você proteja seus dados das seguintes formas:

- Use uma autenticação multifator (MFA) com cada conta.
- Use SSL/TLS para se comunicar com os recursos. AWS Exigimos TLS 1.2 e recomendamos TLS 1.3.
- Configure a API e o registro de atividades do usuário com AWS CloudTrail.
- Use soluções de AWS criptografia, juntamente com todos os controles de segurança padrão Serviços da AWS.
- Use serviços gerenciados de segurança avançada, como o Amazon Macie, que ajuda a localizar e proteger dados sigilosos armazenados no Amazon S3.
- Se você precisar de módulos criptográficos validados pelo FIPS 140-2 ao acessar AWS por meio de uma interface de linha de comando ou de uma API, use um endpoint FIPS. Para ter

mais informações sobre endpoints do FIPS disponíveis, consulte Federal Information Processing Standard (FIPS) 140-2.

É altamente recomendável que nunca sejam colocadas informações de identificação confidenciais, como endereços de e-mail dos seus clientes, em marcações ou campos de formato livre, como um campo Nome. Isso inclui quando você trabalha com Step Functions ou outros Serviços da AWS usando o console, a API ou AWS os SDKs. AWS CLI Quaisquer dados inseridos em tags ou campos de texto de formato livre usados para nomes podem ser usados para logs de faturamento ou de diagnóstico. Se você fornecer um URL para um servidor externo, recomendemos fortemente que não sejam incluídas informações de credenciais no URL para validar a solicitação a esse servidor.

Criptografia em AWS Step Functions

Criptografia em repouso

O Step Functions sempre criptografa seus dados em repouso. Os dados inseridos AWS Step Functions são criptografados em repouso usando criptografia transparente do lado do servidor. Isso ajuda a reduzir a carga e a complexidade operacionais necessárias para proteger dados confidenciais. Com a criptografia em repouso, é possível criar aplicativos que priorizam a segurança e que atendem a requisitos de conformidade e regulamentação de criptografia.

Criptografia em trânsito

O Step Functions criptografa dados em trânsito entre o serviço e outros produtos integrados da AWS (consulte Usando AWS Step Functions com outros serviços). Todos os dados transmitidos entre o Step Functions e os serviços integrados são criptografados usando Transport Layer Security (TLS).

Gerenciamento de identidades e acesso no AWS Step Functions

O acesso a AWS Step Functions requer credenciais que AWS possam ser usadas para autenticar suas solicitações. Essas credenciais devem ter permissões para acessar AWS recursos, como recuperar dados de eventos de outros AWS recursos. As seções a seguir fornecem detalhes sobre como é possível usar o AWS Identity and Access Management (IAM) e o Step Functions para ajudar a proteger seus recursos controlando quem pode acessá-los.

AWS Identity and Access Management (IAM) é uma ferramenta AWS service (Serviço da AWS) que ajuda o administrador a controlar com segurança o acesso aos AWS recursos. Os administradores

do IAM controlam quem pode ser autenticado (conectado) e autorizado (tem permissões) a usar Step Functions os recursos. O IAM é um AWS service (Serviço da AWS) que você pode usar sem custo adicional.

Público

A forma como você usa AWS Identity and Access Management (IAM) difere, dependendo do trabalho que você faz Step Functions.

Usuário do serviço — Se você usar o Step Functions serviço para realizar seu trabalho, seu administrador fornecerá as credenciais e as permissões de que você precisa. À medida que você usa mais Step Functions recursos para fazer seu trabalho, talvez precise de permissões adicionais. Entender como o acesso é gerenciado pode ajudá-lo a solicitar as permissões corretas ao seu administrador. Se não for possível acessar um atributo no Step Functions, consulte Solução de problemas AWS Step Functions de identidade e acesso.

Administrador de serviços — Se você é responsável pelos Step Functions recursos da sua empresa, provavelmente tem acesso total Step Functions a. É seu trabalho determinar quais Step Functions recursos e recursos seus usuários do serviço devem acessar. Assim, você deve enviar solicitações ao administrador do IAM para alterar as permissões dos usuários de seu serviço. Revise as informações nesta página para entender os Introdução ao IAM. Para saber mais sobre como sua empresa pode usar o IAM com Step Functions, consulteComo AWS Step Functions funciona com o IAM.

Administrador do IAM: Se você for um administrador do IAM, talvez queira saber detalhes sobre como pode gravar políticas para gerenciar acesso ao Step Functions. Para ver exemplos de políticas Step Functions baseadas em identidade que você pode usar no IAM, consulte. Exemplos de políticas baseadas em identidade para AWS Step Functions

Autenticando com identidades

A autenticação é a forma como você faz login AWS usando suas credenciais de identidade. Você deve estar autenticado (conectado AWS) como o Usuário raiz da conta da AWS, como usuário do IAM ou assumindo uma função do IAM.

Você pode entrar AWS como uma identidade federada usando credenciais fornecidas por meio de uma fonte de identidade. AWS IAM Identity Center Usuários (IAM Identity Center), a autenticação de login único da sua empresa e suas credenciais do Google ou do Facebook são exemplos de identidades federadas. Quando você faz login como identidade federada, o administrador já

configurou anteriormente a federação de identidades usando perfis do IAM. Ao acessar AWS usando a federação, você está assumindo indiretamente uma função.

Dependendo do tipo de usuário que você é, você pode entrar no AWS Management Console ou no portal de AWS acesso. Para obter mais informações sobre como fazer login em AWS, consulte Como fazer login Conta da AWS no Guia do Início de Sessão da AWS usuário.

Se você acessar AWS programaticamente, AWS fornece um kit de desenvolvimento de software (SDK) e uma interface de linha de comando (CLI) para assinar criptograficamente suas solicitações usando suas credenciais. Se você não usa AWS ferramentas, você mesmo deve assinar as solicitações. Para obter mais informações sobre como usar o método recomendado para assinar solicitações por conta própria, consulte Assinatura de solicitações de AWS API no Guia do usuário do IAM.

Independente do método de autenticação usado, também pode ser exigido que você forneça informações adicionais de segurança. Por exemplo, AWS recomenda que você use a autenticação multifator (MFA) para aumentar a segurança da sua conta. Para saber mais, consulte Autenticação multifator no Guia AWS IAM Identity Center do usuário e Utilizar a autenticação multifator (MFA) na AWS no Guia do usuário do IAM.

Conta da AWS usuário root

Ao criar uma Conta da AWS, você começa com uma identidade de login que tem acesso completo a todos Serviços da AWS os recursos da conta. Essa identidade é chamada de usuário Conta da AWS raiz e é acessada fazendo login com o endereço de e-mail e a senha que você usou para criar a conta. É altamente recomendável não usar o usuário raiz para tarefas diárias. Proteja as credenciais do usuário raiz e use-as para executar as tarefas que somente ele puder executar. Para obter a lista completa das tarefas que exigem login como usuário raiz, consulte Tarefas que exigem credenciais de usuário raiz no Guia do Usuário do IAM.

Identidade federada

Como prática recomendada, exija que usuários humanos, incluindo usuários que precisam de acesso de administrador, usem a federação com um provedor de identidade para acessar Serviços da AWS usando credenciais temporárias.

Uma identidade federada é um usuário do seu diretório de usuários corporativo, de um provedor de identidade da web AWS Directory Service, do diretório do Identity Center ou de qualquer usuário que acesse usando credenciais fornecidas Serviços da AWS por meio de uma fonte de identidade.

Quando as identidades federadas são acessadas Contas da AWS, elas assumem funções, e as funções fornecem credenciais temporárias.

Para o gerenciamento de acesso centralizado, recomendamos usar o AWS IAM Identity Center. Você pode criar usuários e grupos no IAM Identity Center ou pode se conectar e sincronizar com um conjunto de usuários e grupos em sua própria fonte de identidade para uso em todos os seus Contas da AWS aplicativos. Para obter mais informações sobre o Centro de Identidade do IAM, consulte "O que é o Centro de Identidade do IAM?" no Guia do usuário AWS IAM Identity Center .

Usuários e grupos do IAM

Um usuário do IAM é uma identidade dentro da sua Conta da AWS que tem permissões específicas para uma única pessoa ou aplicativo. Sempre que possível, recomendamos contar com credenciais temporárias em vez de criar usuários do IAM com credenciais de longo prazo, como senhas e chaves de acesso. No entanto, se você tiver casos de uso específicos que exijam credenciais de longo prazo com usuários do IAM, recomendamos alternar as chaves de acesso. Para obter mais informações, consulte Alterne as chaves de acesso regularmente para casos de uso que exijam credenciais de longo prazo no Guia do Usuário do IAM.

Um grupo do IAM é uma identidade que especifica uma coleção de usuários do IAM. Não é possível fazer login como um grupo. É possível usar grupos para especificar permissões para vários usuários de uma vez. Os grupos facilitam o gerenciamento de permissões para grandes conjuntos de usuários. Por exemplo, você pode ter um grupo chamado IAMAdmins e conceder a esse grupo permissões para administrar recursos do IAM.

Usuários são diferentes de perfis. Um usuário é exclusivamente associado a uma pessoa ou a uma aplicação, mas um perfil pode ser assumido por qualquer pessoa que precisar dele. Os usuários têm credenciais permanentes de longo prazo, mas os perfis fornecem credenciais temporárias. Para saber mais, consulte Quando criar um usuário do IAM (em vez de um perfil) no Guia do usuário do IAM.

Perfis do IAM

Uma função do IAM é uma identidade dentro da sua Conta da AWS que tem permissões específicas. Ele é semelhante a um usuário do IAM, mas não está associado a uma pessoa específica. Você pode assumir temporariamente uma função do IAM no AWS Management Console trocando de funções. Você pode assumir uma função chamando uma operação de AWS API AWS CLI ou usando uma URL personalizada. Para obter mais informações sobre métodos para o uso de perfis, consulte Utilizar perfis do IAM no Guia do usuário do IAM.

Funções do IAM com credenciais temporárias são úteis nas seguintes situações:

- Acesso de usuário federado: para atribuir permissões a identidades federadas, você pode criar um perfil e definir permissões para ele. Quando uma identidade federada é autenticada, essa identidade é associada ao perfil e recebe as permissões definidas pelo mesmo. Para obter mais informações sobre perfis para federação, consulte Criar um perfil para um provedor de identidades de terceiros no Guia do Usuário do IAM. Se você usar o Centro de identidade do IAM, configure um conjunto de permissões. Para controlar o que suas identidades podem acessar após a autenticação, o Centro de identidade do IAM correlaciona o conjunto de permissões a um perfil no IAM. Para obter informações sobre conjuntos de permissões, consulte Conjuntos de permissões no Guia do usuário AWS IAM Identity Center .
- Permissões temporárias para usuários do IAM –- um usuário ou um perfil do IAM pode presumir um perfil do IAM para obter temporariamente permissões diferentes para uma tarefa específica.
- Acesso entre contas –- é possível usar um perfil do IAM para permitir que alguém (uma entidade principal confiável) em outra conta acesse recursos em sua conta. Os perfis são a principal forma de conceder acesso entre contas. No entanto, com alguns Serviços da AWS, você pode anexar uma política diretamente a um recurso (em vez de usar uma função como proxy). Para saber a diferença entre funções e políticas baseadas em recursos para acesso entre contas, consulte Acesso a recursos entre contas no IAM no Guia do usuário do IAM.
- Acesso entre serviços — Alguns Serviços da AWS usam recursos em outros Serviços da AWS. Por exemplo, quando você faz uma chamada em um serviço, é comum que esse serviço execute aplicativos no Amazon EC2 ou armazene objetos no Amazon S3. Um serviço pode fazer isso usando as permissões do principal de chamada, usando um perfil de serviço ou um perfil vinculado a serviço.
  - Sessões de acesso direto (FAS) — Quando você usa um usuário ou uma função do IAM para realizar ações AWS, você é considerado principal. Ao usar alguns serviços, você pode executar uma ação que inicia outra ação em um serviço diferente. O FAS usa as permissões do diretor chamando um AWS service (Serviço da AWS), combinadas com a solicitação AWS service (Serviço da AWS) para fazer solicitações aos serviços posteriores. As solicitações do FAS são feitas somente quando um serviço recebe uma solicitação que requer interações com outros Serviços da AWS ou com recursos para ser concluída. Nesse caso, você precisa ter permissões para executar ambas as ações. Para obter detalhes da política ao fazer solicitações de FAS, consulte Encaminhar sessões de acesso.
  - Função de serviço: um perfil de serviço é um perfil do IAM que um serviço assume para realizar ações em seu nome. Um administrador do IAM pode criar, modificar e excluir um perfil de

serviço do IAM. Para obter mais informações, consulte Criar um perfil para delegar permissões a um AWS service (Serviço da AWS) no Guia do Usuário do IAM.

- Função vinculada ao serviço — Uma função vinculada ao serviço é um tipo de função de serviço vinculada a um. AWS service (Serviço da AWS) O serviço pode presumir a função de executar uma ação em seu nome. As funções vinculadas ao serviço aparecem em você Conta da AWS e são de propriedade do serviço. Um administrador do IAM pode visualizar, mas não editar as permissões para funções vinculadas ao serviço.

- Aplicativos em execução no Amazon EC2 — Você pode usar uma função do IAM para gerenciar credenciais temporárias para aplicativos que estão sendo executados em uma instância do EC2 e fazendo AWS CLI solicitações de API. AWS É preferível fazer isso a armazenar chaves de acesso na instância do EC2. Para atribuir uma AWS função a uma instância do EC2 e disponibilizá-la para todos os seus aplicativos, você cria um perfil de instância anexado à instância. Um perfil de instância contém o perfil e permite que os programas em execução na instância do EC2 obtenham credenciais temporárias. Para mais informações, consulte Utilizar um perfil do IAM para conceder permissões a aplicações em execução nas instâncias do Amazon EC2 no Guia do usuário do IAM.

Para saber se deseja usar perfis do IAM, consulte Quando criar um perfil do IAM (em vez de um usuário) no Guia do usuário do IAM.

Gerenciando acesso usando políticas

Você controla o acesso AWS criando políticas e anexando-as a AWS identidades ou recursos. Uma política é um objeto AWS que, quando associada a uma identidade ou recurso, define suas permissões. AWS avalia essas políticas quando um principal (usuário, usuário raiz ou sessão de função) faz uma solicitação. As permissões nas políticas determinam se a solicitação será permitida ou negada. A maioria das políticas é armazenada AWS como documentos JSON. Para obter mais informações sobre a estrutura e o conteúdo de documentos de políticas JSON, consulte Visão geral das políticas JSON no Guia do Usuário do IAM.

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

Por padrão, usuários e funções não têm permissões. Para conceder aos usuários permissões para executar ações nos recursos que eles precisam, um administrador do IAM pode criar políticas do IAM. O administrador pode então adicionar as políticas do IAM aos perfis e os usuários podem presumir os perfis.

As políticas do IAM definem permissões para uma ação independente do método usado para executar a operação. Por exemplo, suponha que você tenha uma política que permite a ação `iam:GetRole`. Um usuário com essa política pode obter informações de função da AWS Management Console AWS CLI, da ou da AWS API.

Políticas baseadas em identidade

As políticas baseadas em identidade são documentos de políticas de permissões JSON que você pode anexar a uma identidade, como usuário do IAM, grupo de usuários ou perfil. Essas políticas controlam quais ações os usuários e perfis podem realizar, em quais recursos e em que condições. Para saber como criar uma política baseada em identidade, consulte Criando políticas do IAM no Guia do Usuário do IAM.

As políticas baseadas em identidade podem ser categorizadas ainda adicionalmente como políticas em linha ou políticas gerenciadas. As políticas em linha são anexadas diretamente a um único usuário, grupo ou perfil. As políticas gerenciadas são políticas autônomas que você pode associar a vários usuários, grupos e funções em seu Conta da AWS. As políticas AWS gerenciadas incluem políticas gerenciadas e políticas gerenciadas pelo cliente. Para saber como escolher entre uma política gerenciada ou uma política em linha, consulte Escolher entre políticas gerenciadas e políticas em linha no Guia do Usuário do IAM.

Políticas baseadas em recursos

Políticas baseadas em recursos são documentos de políticas JSON que você anexa a um recurso. São exemplos de políticas baseadas em recursos as políticas de confiança de perfil do IAM e as políticas de bucket do Amazon S3. Em serviços que suportem políticas baseadas em recursos, os administradores de serviço podem usá-las para controlar o acesso a um recurso específico. Para o recurso ao qual a política está anexada, a política define quais ações um principal especificado pode executar nesse recurso e em que condições. Você deve especificar uma entidade principal em uma política baseada em recursos. Os diretores podem incluir contas, usuários, funções, usuários federados ou. Serviços da AWS

Políticas baseadas em recursos são políticas em linha localizadas nesse serviço. Você não pode usar políticas AWS gerenciadas do IAM em uma política baseada em recursos.

Listas de controle de acesso (ACLs)

As listas de controle de acesso (ACLs) controlam quais entidades principais (membros, usuários ou perfis da conta) têm permissões para acessar um recurso. As ACLs são semelhantes as políticas baseadas em recursos, embora não usem o formato de documento de política JSON.

O Amazon S3 e o Amazon VPC são exemplos de serviços que oferecem suporte a ACLs. AWS WAF Para saber mais sobre ACLs, consulte Visão geral da lista de controle de acesso (ACL) no Guia do Desenvolvedor do Amazon Simple Storage Service.

Outros tipos de política

AWS oferece suporte a tipos de políticas adicionais menos comuns. Esses tipos de política podem definir o máximo de permissões concedidas a você pelos tipos de política mais comuns.

- Limites de permissões: um limite de permissões é um atributo avançado no qual você define o máximo de permissões que uma política baseada em identidade pode conceder a uma entidade do IAM (usuário ou perfil do IAM). É possível definir um limite de permissões para uma entidade. As permissões resultantes são a interseção das políticas baseadas em identidade de uma entidade com seus limites de permissões. As políticas baseadas em recurso que especificam o usuário ou o perfil no campo `Principal` não são limitadas pelo limite de permissões. Uma negação explícita em qualquer uma dessas políticas substitui a permissão. Para obter mais informações sobre limites de permissões, consulte Limites de permissões para identidades do IAM no Guia do Usuário do IAM.
- Políticas de controle de serviço (SCPs) — SCPs são políticas JSON que especificam as permissões máximas para uma organização ou unidade organizacional (OU) em. AWS Organizations AWS Organizations é um serviço para agrupar e gerenciar centralmente vários Contas da AWS que sua empresa possui. Se você habilitar todos os atributos em uma organização, poderá aplicar políticas de controle de serviço (SCPs) a qualquer uma ou a todas as contas. O SCP limita as permissões para entidades nas contas dos membros, incluindo cada uma Usuário raiz da conta da AWS. Para obter mais informações sobre o Organizações e SCPs, consulte How SCPs work (Como os SCPs funcionam) no Guia do usuário do AWS Organizations .
- Políticas de sessão: são políticas avançadas que você transmite como um parâmetro quando cria de forma programática uma sessão temporária para um perfil ou um usuário federado. As permissões da sessão resultante são a interseção das políticas baseadas em identidade do usuário ou do perfil e das políticas de sessão. As permissões também podem ser provenientes de uma política baseada em atributo. Uma negação explícita em qualquer uma dessas políticas substitui a permissão. Para obter mais informações, consulte Políticas de sessão no Guia do Usuário do IAM.

Vários tipos de política

Quando vários tipos de política são aplicáveis a uma solicitação, é mais complicado compreender as permissões resultantes. Para saber como AWS determinar se uma solicitação deve ser permitida quando vários tipos de políticas estão envolvidos, consulte Lógica de avaliação de políticas no Guia do usuário do IAM.

Controle de acesso

É possível ter credenciais válidas para autenticar suas solicitações. No entanto, a menos que tenha permissões, não é possível criar nem acessar os recursos do Step Functions. Por exemplo, você deve ter permissões para invocar AWS Lambda alvos do Amazon Simple Notification Service (Amazon SNS) e do Amazon Simple Queue Service (Amazon SQS) associados às suas regras do Step Functions.

As seções a seguir descrevem como gerenciar permissões para o Step Functions.

- Criar um perfil do IAM para sua máquina de estado
- Como criar permissões granulares do IAM para usuários que não são administradores
- Endpoints da Amazon VPC para o Step Functions
- Políticas do IAM para serviços integrados
- Políticas do IAM para usar o estado Mapa Distribuído

Ações políticas para Step Functions

Oferece compatibilidade com ações de políticas	Sim

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos, e em que condições.

O elemento `Action` de uma política JSON descreve as ações que você pode usar para permitir ou negar acesso em uma política. As ações de política geralmente têm o mesmo nome da operação de AWS API associada. Existem algumas exceções, como ações somente de permissão, que não têm uma operação de API correspondente. Algumas operações também exigem várias ações em uma política. Essas ações adicionais são chamadas de ações dependentes.

Incluem ações em uma política para conceder permissões para executar a operação associada.

Para ver uma lista de Step Functions ações, consulte Recursos definidos por AWS Step Functions na Referência de autorização de serviço.

As ações de política Step Functions usam o seguinte prefixo antes da ação:

```
states
```

Para especificar várias ações em uma única instrução, separe-as com vírgulas.

```
"Action": [
      "states:action1",
      "states:action2"
         ]
```

Para ver exemplos de políticas Step Functions baseadas em identidade, consulte. Exemplos de políticas baseadas em identidade para AWS Step Functions

Recursos políticos para Step Functions

Oferece compatibilidade com recursos de políticas	Sim

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos, e em que condições.

O elemento de política JSON `Resource` especifica o objeto ou os objetos aos quais a ação se aplica. As instruções devem incluir um elemento `Resource` ou `NotResource`. Como prática recomendada, especifique um recurso usando seu nome do recurso da Amazon (ARN). Isso pode ser feito para ações que oferecem compatibilidade com um tipo de recurso específico, conhecido como permissões em nível de recurso.

Para ações que não oferecem compatibilidade com permissões em nível de recurso, como operações de listagem, use um curinga (*) para indicar que a instrução se aplica a todos os recursos.

```
"Resource": "*"
```

Para ver uma lista dos tipos de Step Functions recursos e seus ARNs, consulte Ações definidas por AWS Step Functions na Referência de Autorização de Serviço. Para saber com quais ações você pode especificar o ARN de cada recurso, consulte Recursos definidos pelo AWS Step Functions.

Para ver exemplos de políticas Step Functions baseadas em identidade, consulte. Exemplos de políticas baseadas em identidade para AWS Step Functions

Chaves de condição de política para Step Functions

Suporta chaves de condição de política específicas de serviço	Sim

Os administradores podem usar políticas AWS JSON para especificar quem tem acesso ao quê. Ou seja, qual entidade principal pode executar ações em quais recursos, e em que condições.

O elemento `Condition` (ou bloco `Condition`) permite que você especifique condições nas quais uma instrução estiver em vigor. O elemento `Condition` é opcional. É possível criar expressões condicionais que usem agentes de condição, como “igual a” ou “menor que”, para fazer a condição da política corresponder aos valores na solicitação.

Se você especificar vários elementos `Condition` em uma instrução ou várias chaves em um único `Condition` elemento, a AWS os avaliará usando uma operação lógica `AND`. Se você especificar vários valores para uma única chave de condição, AWS avalia a condição usando uma `OR` operação lógica. Todas as condições devem ser atendidas antes que as permissões da instrução sejam concedidas.

Você também pode usar variáveis de espaço reservado ao especificar condições. Por exemplo, é possível conceder a um usuário do IAM permissão para acessar um atributo somente se ele estiver marcado com seu nome de usuário do IAM. Para obter mais informações, consulte Elementos da política do IAM: variáveis e tags no Guia do usuário do IAM.

AWS suporta chaves de condição globais e chaves de condição específicas do serviço. Para ver todas as chaves de condição AWS globais, consulte as chaves de contexto de condição AWS global no Guia do usuário do IAM.

Para ver uma lista de chaves de Step Functions condição, consulte Chaves de condição AWS Step Functions na Referência de autorização de serviço. Para saber com quais ações e recursos você pode usar uma chave de condição, consulte Recursos definidos por AWS Step Functions.

Para ver exemplos de políticas Step Functions baseadas em identidade, consulte. Exemplos de políticas baseadas em identidade para AWS Step Functions

ACLs em Step Functions

Oferece compatibilidade com ACLs	Não

As listas de controle de acesso (ACLs) controlam quais entidades principais (membros, usuários ou perfis da conta) têm permissões para acessar um recurso. As ACLs são semelhantes as políticas baseadas em recursos, embora não usem o formato de documento de política JSON.

ABAC com Step Functions

Oferece compatibilidade com ABAC (tags em políticas)	Parcial

O controle de acesso baseado em recurso (ABAC) é uma estratégia de autorização que define permissões com base em recursos. Em AWS, esses atributos são chamados de tags. Você pode anexar tags a entidades do IAM (usuários ou funções) e a vários AWS recursos. A marcação de entidades e atributos é a primeira etapa do ABAC. Em seguida, você cria políticas de ABAC para permitir operações quando a tag da entidade principal corresponder à tag do recurso que ela estiver tentando acessar.

O ABAC é útil em ambientes que estão crescendo rapidamente e ajuda em situações onde o gerenciamento de políticas se torna um problema.

Para controlar o acesso baseado em tags, forneça informações sobre as tags no elemento de condição de uma política usando as `aws:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`* ou chaves de condição `aws:TagKeys`.

Se um serviço oferecer suporte às três chaves de condição para cada tipo de recurso, o valor será Sim para o serviço. Se um serviço oferecer suporte às três chaves de condição somente para alguns tipos de recursos, o valor será Parcial

Para obter mais informações sobre o ABAC, consulte O que é ABAC? no Guia do Usuário do IAM. Para visualizar um tutorial com etapas para configurar o ABAC, consulte Utilizar controle de acesso baseado em atributos (ABAC) no Guia do usuário do IAM.

Usando credenciais temporárias com Step Functions

Oferece compatibilidade com credenciais temporárias	Sim

Alguns Serviços da AWS não funcionam quando você faz login usando credenciais temporárias. Para obter informações adicionais, incluindo quais Serviços da AWS funcionam com credenciais temporárias, consulte Serviços da AWS “Trabalhe com o IAM” no Guia do usuário do IAM.

Você está usando credenciais temporárias se fizer login AWS Management Console usando qualquer método, exceto um nome de usuário e senha. Por exemplo, quando você acessa AWS usando o link de login único (SSO) da sua empresa, esse processo cria automaticamente credenciais temporárias. Você também cria automaticamente credenciais temporárias quando faz login no console como usuário e, em seguida, alterna perfis. Para obter mais informações sobre como alternar funções, consulte Alternar para um perfil (console) no Guia do usuário do IAM.

Você pode criar manualmente credenciais temporárias usando a AWS API AWS CLI ou. Em seguida, você pode usar essas credenciais temporárias para acessar AWS. AWS recomenda que você gere credenciais temporárias dinamicamente em vez de usar chaves de acesso de longo prazo. Para mais informações, consulte Credenciais de segurança temporárias no IAM.

Permissões principais entre serviços para Step Functions

Suporte para o recurso Encaminhamento de sessões de acesso (FAS)	Sim

Quando você usa um usuário ou uma função do IAM para realizar ações AWS, você é considerado principal. Ao usar alguns serviços, você pode executar uma ação que inicia outra ação em um serviço diferente. O FAS usa as permissões do diretor chamando um AWS service (Serviço da AWS), combinadas com a solicitação AWS service (Serviço da AWS) para fazer solicitações aos serviços posteriores. As solicitações do FAS são feitas somente quando um serviço recebe uma solicitação que requer interações com outros Serviços da AWS ou com recursos para ser concluída. Nesse caso, você precisa ter permissões para executar ambas as ações. Para obter detalhes da política ao fazer solicitações de FAS, consulte Encaminhamento de sessões de acesso.

Funções de serviço para Step Functions

Oferece compatibilidade com funções de serviço	Sim

O perfil de serviço é um perfil do IAM que um serviço assume para realizar ações em seu nome. Um administrador do IAM pode criar, modificar e excluir um perfil de serviço do IAM. Para obter mais informações, consulte Criar um perfil para delegar permissões a um AWS service (Serviço da AWS) no Guia do Usuário do IAM.

> **Warning**
>
> Alterar as permissões de uma função de serviço pode interromper Step Functions a funcionalidade. Edite as funções de serviço somente quando Step Functions fornecer orientação para fazer isso.

Funções vinculadas a serviços para Step Functions

Oferece suporte a perfis vinculados ao serviço	Não

Uma função vinculada ao serviço é um tipo de função de serviço vinculada a um. AWS service (Serviço da AWS) O serviço pode presumir a função de executar uma ação em seu nome. As funções vinculadas ao serviço aparecem em você Conta da AWS e são de propriedade do serviço. Um administrador do IAM pode visualizar, mas não pode editar as permissões para perfis vinculados ao serviço.

Para obter detalhes sobre como criar ou gerenciar perfis vinculados a serviços, consulte Serviços da AWS que funcionam com o IAM. Encontre um serviço na tabela que inclua um `Yes` na coluna Função vinculada ao serviço. Escolha o link Sim para visualizar a documentação do perfil vinculado a serviço desse serviço.

Como AWS Step Functions funciona com o IAM

Antes de usar o IAM para gerenciar o acesso Step Functions, saiba com quais recursos do IAM estão disponíveis para uso Step Functions.

Recursos do IAM que você pode usar com AWS Step Functions

Atributo do IAM	Step Functions apoio
Políticas baseadas em identidade	Sim
Políticas baseadas em recursos	Não
Ações das políticas	Sim
Atributos de políticas	Sim
Chaves de condição de política (específicas do serviço)	Sim
ACLs	Não
ABAC (tags em políticas)	Parcial
Credenciais temporárias	Sim
Permissões de entidade principal	Sim
Perfis de serviço	Sim
Perfis vinculados ao serviço	Não

Para ter uma visão de alto nível de como Step Functions e outros AWS serviços funcionam com a maioria dos recursos do IAM, consulte AWS os serviços que funcionam com o IAM no Guia do usuário do IAM.

Exemplos de políticas baseadas em identidade para AWS Step Functions

Por padrão, usuários e funções não têm permissão para criar ou modificar recursos do Step Functions . Eles também não podem realizar tarefas usando a AWS API AWS Management Console, AWS Command Line Interface (AWS CLI) ou. Para conceder aos usuários permissão para executar ações nos recursos de que precisam, um administrador do IAM pode criar políticas do IAM. O administrador pode então adicionar as políticas do IAM aos perfis e os usuários podem presumir os perfis.

Para saber como criar uma política baseada em identidade do IAM usando esses exemplos de documento de política JSON, consulte Criação de políticas do IAM no Guia do Usuário do IAM.

Para obter detalhes sobre ações e tipos de recursos definidos por Step Functions, incluindo o formato dos ARNs para cada um dos tipos de recursos, consulte Ações, recursos e chaves de condição AWS Step Functions na Referência de autorização de serviço.

Tópicos

- Melhores práticas de política
- Usar o console do Step Functions
- Permitir que usuários visualizem suas próprias permissões

Melhores práticas de política

As políticas baseadas em identidade determinam se alguém pode criar, acessar ou excluir Step Functions recursos em sua conta. Essas ações podem incorrer em custos para seus Conta da AWS. Ao criar ou editar políticas baseadas em identidade, siga estas diretrizes e recomendações:

- Comece com as políticas AWS gerenciadas e avance para as permissões de privilégios mínimos — Para começar a conceder permissões aos seus usuários e cargas de trabalho, use as políticas AWS gerenciadas que concedem permissões para muitos casos de uso comuns. Eles estão disponíveis no seu Conta da AWS. Recomendamos que você reduza ainda mais as permissões definindo políticas gerenciadas pelo AWS cliente que sejam específicas para seus casos de uso. Para obter mais informações, consulte Políticas gerenciadas pela AWS ou Políticas gerenciadas pela AWS para funções de trabalho no Guia do Usuário do IAM.
- Aplique permissões de privilégio mínimo –- ao definir permissões com as políticas do IAM, conceda apenas as permissões necessárias para executar uma tarefa. Você faz isso definindo as ações que podem ser executadas em atributos específicos sob condições específicas, também conhecidas como permissões de privilégio mínimo. Para obter mais informações sobre como usar o IAM para aplicar permissões, consulte Políticas e permissões no IAM no Guia do Usuário do IAM.
- Use condições nas políticas do IAM para restringir ainda mais o acesso –- você pode adicionar uma condição às políticas para limitar o acesso a ações e recursos. Por exemplo, você pode gravar uma condição de política para especificar que todas as solicitações devem ser enviadas usando SSL. Você também pode usar condições para conceder acesso às ações de serviço se elas forem usadas por meio de uma ação específica AWS service (Serviço da AWS), como AWS

CloudFormation. Para obter mais informações, consulte Elementos da política JSON do IAM: Condição no Guia do usuário do IAM.

- Use o IAM Access Analyzer para validar suas políticas do IAM a fim de garantir permissões seguras e funcionais –- o IAM Access Analyzer valida as políticas novas e existentes para que elas sigam a linguagem de política do IAM (JSON) e as práticas recomendadas do IAM. O IAM Access Analyzer oferece mais de 100 verificações de política e recomendações acionáveis para ajudá-lo a criar políticas seguras e funcionais. Para obter mais informações, consulte Validação de políticas do IAM Access Analyzer no Guia do Usuário do IAM.
- Exigir autenticação multifator (MFA) — Se você tiver um cenário que exija usuários do IAM ou um usuário root, ative Conta da AWS a MFA para obter segurança adicional. Para exigir MFA quando as operações de API forem chamadas, adicione condições de MFA às suas políticas. Para obter mais informações, consulte Configuração de acesso à API protegido por MFA no Guia do Usuário do IAM.

Para obter mais informações sobre as práticas recomendadas do IAM, consulte Práticas Recomendadas de Segurança no IAM no Guia do Usuário do IAM.

Usar o console do Step Functions

Para acessar o AWS Step Functions console, você deve ter um conjunto mínimo de permissões. Essas permissões devem permitir que você liste e visualize detalhes sobre os Step Functions recursos em seu Conta da AWS. Se você criar uma política baseada em identidade que seja mais restritiva do que as permissões mínimas necessárias, o console não funcionará como pretendido para entidades (usuários ou perfis) com essa política.

Você não precisa permitir permissões mínimas do console para usuários que estão fazendo chamadas somente para a API AWS CLI ou para a AWS API. Em vez disso, permita o acesso somente a ações que correspondam a operação de API que estiverem tentando executar.

Para garantir que usuários e funções ainda possam usar o Step Functions console, anexe também a política Step Functions `ConsoleAccess` ou a política `ReadOnly` AWS gerenciada às entidades. Para obter mais informações, consulte Adicionando Permissões a um Usuário no Guia do Usuário do IAM.

Permitir que usuários visualizem suas próprias permissões

Este exemplo mostra como criar uma política que permita que os usuários do IAM visualizem as políticas gerenciadas e em linha anexadas a sua identidade de usuário. Essa política inclui

permissões para concluir essa ação no console ou programaticamente usando a API AWS CLI ou 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 baseadas em identidade para Step Functions

Suporta políticas baseadas em identidade	Sim

As políticas baseadas em identidade são documentos de políticas de permissões JSON que você pode anexar a uma identidade, como usuário, grupo de usuários ou perfil do IAM. Essas políticas controlam quais ações os usuários e perfis podem realizar, em quais recursos e em que condições. Para saber como criar uma política baseada em identidade, consulte Criando políticas do IAM no Guia do Usuário do IAM.

Com as políticas baseadas em identidade do IAM, é possível especificar ações ou recursos permitidos ou negados, assim como as condições sob as quais as ações são permitidas ou negadas. Você não pode especificar a entidade principal em uma política baseada em identidade porque ela se aplica ao usuário ou perfil ao qual ela está anexada. Para saber mais sobre todos os elementos que podem ser usados em uma política JSON, consulte Referência de elementos da política JSON do IAM no Guia do Usuário do IAM.

Exemplos de políticas baseadas em identidade para Step Functions

Para ver exemplos de políticas Step Functions baseadas em identidade, consulte. Exemplos de políticas baseadas em identidade para AWS Step Functions

Políticas baseadas em recursos dentro Step Functions

Oferece compatibilidade com políticas baseadas em recursos	Não

Políticas baseadas em recursos são documentos de políticas JSON que você anexa a um recurso. São exemplos de políticas baseadas em recursos as políticas de confiança de perfil do IAM e as políticas de bucket do Amazon S3. Em serviços que suportem políticas baseadas em recursos, os administradores de serviço podem usá-las para controlar o acesso a um recurso específico. Para o recurso ao qual a política está anexada, a política define quais ações um principal especificado pode executar nesse recurso e em que condições. Você deve especificar uma entidade principal em uma política baseada em recursos. Os diretores podem incluir contas, usuários, funções, usuários federados ou. Serviços da AWS

Para permitir o acesso entre contas, você pode especificar uma conta inteira ou as entidades do IAM em outra conta como a entidade principal em uma política baseada em atributo. Adicionar uma entidade principal entre contas à política baseada em recurso é apenas metade da tarefa de estabelecimento da relação de confiança. Quando o principal e o recurso são diferentes Contas da AWS, um administrador do IAM na conta confiável também deve conceder permissão à entidade

principal (usuário ou função) para acessar o recurso. Eles concedem permissão ao anexar uma política baseada em identidade para a entidade. No entanto, se uma política baseada em recurso conceder acesso a uma entidade principal na mesma conta, nenhuma política baseada em identidade adicional será necessária. Para obter mais informações, consulte Acesso a recursos entre contas no IAM no Guia do usuário do IAM.

AWS políticas gerenciadas para AWS Step Functions

Uma política AWS gerenciada é uma política autônoma criada e administrada por AWS. AWS as políticas gerenciadas são projetadas para fornecer permissões para muitos casos de uso comuns, para que você possa começar a atribuir permissões a usuários, grupos e funções.

Lembre-se de que as políticas AWS gerenciadas podem não conceder permissões de privilégio mínimo para seus casos de uso específicos porque elas estão disponíveis para uso de todos os AWS clientes. Recomendamos que você reduza ainda mais as permissões definindo políticas gerenciadas pelo cliente da específicas para seus casos de uso.

Você não pode alterar as permissões definidas nas políticas AWS gerenciadas. Se AWS atualizar as permissões definidas em uma política AWS gerenciada, a atualização afetará todas as identidades principais (usuários, grupos e funções) às quais a política está anexada. AWS é mais provável que atualize uma política AWS gerenciada quando uma nova AWS service (Serviço da AWS) é lançada ou novas operações de API são disponibilizadas para serviços existentes.

Para obter mais informações, consulte Políticas gerenciadas pela AWS no Guia do usuário do IAM.

AWS política gerenciada: AWSStepFunctionsConsoleFullAccess

É possível anexar a política `AWSStepFunctionsConsoleFullAccess` a suas identidades do IAM.

Essa política concede permissões de *`administrador`* que permitem que um usuário acesse o console Step Functions. Para uma experiência de console completa, um usuário também pode precisar da PassRole permissão iam: em outras funções do IAM que podem ser assumidas pelo serviço.

AWS política gerenciada: AWSStepFunctionsReadOnlyAccess

É possível anexar a política `AWSStepFunctionsReadOnlyAccess` a suas identidades do IAM.

Essa política concede permissões *somente para leitura* que permitem que um usuário ou função liste e descreva máquinas de estado, atividades, execuções, atividades MapRuns, tags e alias e versões de máquinas de estado. Essa política também concede permissão para verificar a sintaxe das definições de máquina de estado fornecidas por você.

AWS política gerenciada: AWSStepFunctionsFullAccess

É possível anexar a política `AWSStepFunctionsFullAccess` a suas identidades do IAM.

Essa política concede permissões *completas* a um usuário ou função para usar a API Step Functions. Para ter acesso total, um usuário deve ter PassRole permissão *iam:* em pelo menos uma função do IAM que possa ser assumida pelo serviço.

Step Functions atualizações nas políticas AWS gerenciadas

Veja detalhes sobre as atualizações das políticas AWS gerenciadas Step Functions desde que esse serviço começou a rastrear essas alterações. Para obter alertas automáticos sobre alterações feitas nesta página, inscreva-se no feed RSS na página Step Functions Histórico do documento.

Alteração	Descrição	Data
AWSStepFunctionsReadOnlyAccess: atualizar para uma política existente	Step Functions adicionou novas permissões para permitir a chamada `states:ValidateStateMachineDefinition` da ação da API para verificar a sintaxe das definições da máquina de estado que você fornece.	25 de abril de 2024
AWSStepFunctionsReadOnlyAccess: atualizar para uma política existente	Step Functions adicionou novas permissões para permitir listar e ler dados relacionados a: Tags (ListTagsForResource), Mapa distribuído (ListMapexecuções, DescribeMapRun), versões e	02 de abril de 2024

Alteração	Descrição	Data
	aliases (DescribeStateMachineAlias, ListStateMachineAliases, ListStateMachineVersions).	
Step Functions começou a rastrear as alterações	Step Functions começou a rastrear as mudanças em suas políticas AWS gerenciadas.	02 de abril de 2024

Criar um perfil do IAM para sua máquina de estado

AWS Step Functions pode executar código e acessar AWS recursos (como invocar uma AWS Lambda função). Para manter a segurança, você deve conceder acesso ao Step Functions para esses recursos usando uma função do IAM.

Tutoriais do Step FunctionsNeste guia, você pode aproveitar as funções do IAM geradas automaticamente que são válidas para a AWS região na qual você cria a máquina de estado. No entanto, você pode criar seu próprio perfil do IAM para uma máquina de estado.

Ao ser criada para suas máquinas de estado, uma política do IAM deve incluir as permissões que você gostaria que fossem assumidas pelas máquinas de estado. Você pode usar uma política AWS gerenciada existente como exemplo ou criar uma política personalizada do zero que atenda às suas necessidades específicas. Para ver mais informações, consulte Criar políticas do IAM no Guia do usuário do IAM.

Para criar um perfil do IAM para uma máquina de estado, siga as etapas desta seção.

Neste exemplo, você cria um perfil do IAM com permissão para invocar uma função do Lambda.

Criar um perfil para o Step Functions

1. Faça login no console do IAM e escolha Perfis, Criar perfil.
2. Na página Selecionar entidade confiável, em serviço da AWS , selecione Step Functions na lista e, em seguida, Próximo: permissões.
3. Na página Attached permissions policy, escolha Next: Review.

4. Na página Review (Revisão), insira `StepFunctionsLambdaRole` para Role Name (Nome da função) e escolha Create role (Criar função).

   O perfil do IAM é exibido na lista de perfis.

Para ver mais informações sobre permissões e políticas do IAM, consulte Gerenciamento de acesso no Guia do usuário do IAM.

Prevenção do problema do substituto confuso entre serviços

O problema de "confused deputy" é uma questão de segurança em que uma entidade que não tem permissão para executar uma ação pode coagir uma entidade mais privilegiada a executá-la. Em AWS, a falsificação de identidade entre serviços pode resultar no problema confuso do deputado. A personificação entre serviços pode ocorrer quando um serviço (o serviço de chamada) chama outro serviço (o serviço chamado). Esse tipo de falsificação de identidade pode ocorrer entre contas e serviços. O serviço de chamada pode ser manipulado de modo a usar suas permissões para atuar nos recursos de outro cliente de uma forma na qual ele não deveria ter permissão para acessar.

Para evitar que delegados confusos, AWS fornece ferramentas que ajudam você a proteger seus dados em todos os serviços com diretores de serviços que receberam acesso aos recursos em sua conta. Esta seção se concentra na prevenção de delegados confusos entre serviços, específica de AWS Step Functions; no entanto, você pode aprender mais sobre esse tópico na seção Confused Deputy Problem do Guia do usuário do IAM.

Recomendamos o uso das chaves de contexto de condição global `aws:SourceArn` e `aws:SourceAccount` em políticas de recursos para limitar as permissões que o Step Functions concede a outro serviço para acessar seus recursos. Use `aws:SourceArn` se quiser apenas um recurso associado a acessibilidade de serviço. Use `aws:SourceAccount` se quiser permitir que qualquer recurso nessa conta seja associado ao uso entre serviços.

A maneira mais eficaz de se proteger contra o problema do substituto confuso é usar a chave de contexto de condição global `aws:SourceArn` com o ARN completo do recurso. Se você não souber o ARN completo do recurso ou estiver especificando vários recursos, use a chave de condição de contexto global `aws:SourceArn` com caracteres curingas (*) para as partes desconhecidas do ARN. Por exemplo, `arn:aws:states:*:111122223333:*`.

Aqui está um exemplo de uma política confiável que mostra como você pode usar `aws:SourceArn` e `aws:SourceAccount` com o Step Functions para evitar o problema de substituto 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"
           }
        }
     }
  ]
}
```

Anexar uma política em linha

O Step Functions pode controlar outros serviços diretamente em no estado de uma `Task`. Anexe políticas em linha para permitir que o Step Functions acesse as ações de API dos serviços que você precisa controlar.

1. Abra o console do IAM, escolha Perfis, procure seu perfil do Step Functions e selecione-o.
2. Selecione Adicionar política em linha.
3. Use o Visual editor (Editor visual) ou a guia JSON a fim de criar políticas para a função.

Para obter mais informações sobre como AWS Step Functions controlar outros AWS serviços, consulteUsando AWS Step Functions com outros serviços.

> **Note**
>
> Para obter exemplos de políticas do IAM criadas pelo console do Step Functions, consulte Políticas do IAM para serviços integrados.

Como criar permissões granulares do IAM para usuários que não são administradores

As políticas gerenciadas padrão no IAM, como`ReadOnly`, não abrangem totalmente todos os tipos de AWS Step Functions permissões. Esta seção descreve esses tipos diferentes de permissões e apresenta alguns exemplos de configurações.

O Step Functions tem quatro categorias de permissões. Dependendo do acesso que você deseja fornecer a um usuário, pode controlar o acesso usando as permissões dessas categorias.

Permissões no nível do serviço

Aplique aos componentes da API que não atuam em um recurso específico.

Permissões no nível da máquina de estado

Aplicar a todos os componentes de API que atuam em uma máquina de estado específica.

Permissões no nível da execução

Aplicar a todos os componentes de API que atuam em uma execução específica.

Permissões no nível da atividade

Aplicar a todos os componentes de API que atuam em uma atividade específica ou em uma determinada instância de uma atividade.

Permissões no nível do serviço

Esse nível de permissão se aplica a todas as ações da API que não atuam em um recurso específico. Isso inclui `CreateStateMachineCreateActivity`, `ListStateMachinesListActivities`,, `ValidationStateMachineDefinition` e.

```
{
```

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

Permissões no nível da máquina de estado

Esse nível de permissão se aplica a todas as ações de API que atuam em uma máquina de estado específica. Essas operações de API requerem o nome do recurso da Amazon (ARN) da máquina de estado como parte da solicitação, como `DeleteStateMachine`, `DescribeStateMachine`, `StartExecution` e `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*"
      ]
    }
  ]
}
```

Permissões no nível da execução

Esse nível de permissão se aplica a todas as ações de API que atuam em uma execução específica. Essas operações de API exigem o ARN da execução como parte da solicitação, como `DescribeExecution`, `GetExecutionHistory` e `StopExecution`.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:DescribeExecution",
        "states:DescribeStateMachineForExecution",
        "states:GetExecutionHistory",
        "states:StopExecution"
      ],
      "Resource": [
        "arn:aws:states:*:*:execution:*:ExecutionPrefix*"
      ]
    }
  ]
}
```

Permissões no nível da atividade

Esse nível de permissão se aplica a todas as ações de API que atuam em uma atividade específica ou em uma determinada instância dela. Essas operações de API exigem o ARN da atividade ou o token da instância como parte da solicitação, como `DeleteActivity`, `DescribeActivity`, `GetActivityTask` e `SendTaskHeartbeat`.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:DescribeActivity",
        "states:DeleteActivity",
        "states:GetActivityTask",
        "states:SendTaskHeartbeat"
      ],
      "Resource": [
        "arn:aws:states:*:*:activity:ActivityPrefix*"
      ]
    }
  ]
}
```

Acessando recursos em outros Contas da AWS em seus fluxos de trabalho

O Step Functions fornece acesso entre contas a recursos configurados de forma diferente Contas da AWS em seus fluxos de trabalho. Usando as integrações de serviços Step Functions, você pode invocar qualquer AWS recurso entre contas, mesmo que isso AWS service (Serviço da AWS) não ofereça suporte a políticas baseadas em recursos ou chamadas entre contas.

Por exemplo, suponha que você possua dois Contas da AWS, chamados de Desenvolvimento e Teste, no mesmo Região da AWS. Usando o acesso entre contas, seu fluxo de trabalho na conta de Desenvolvimento pode acessar recursos, como buckets do Amazon S3, tabelas do Amazon DynamoDB e funções do Lambda que estão disponíveis na conta de Teste.

> **Important**
>
> As funções do IAM e as políticas baseadas em recurso delegam o acesso entre contas em uma única partição. Por exemplo, suponha que você tenha uma conta no Oeste dos EUA (Norte da Califórnia) na partição `aws` padrão. Além disso, você tem uma conta na China (Pequim) na partição `aws-cn`. Você não pode usar uma política baseada em recurso do Amazon S3 em sua conta na China (Pequim) para permitir o acesso de usuários em sua conta `aws` padrão.

Para ver mais informações sobre o acesso entre contas, consulte Lógica de avaliação de políticas entre contas no Guia do usuário do IAM.

Embora cada um Conta da AWS mantenha controle total sobre seus próprios recursos, com o Step Functions, você pode reorganizar, trocar, adicionar ou remover etapas em seus fluxos de trabalho sem a necessidade de personalizar nenhum código. Você pode fazer isso mesmo quando os processos mudam ou os aplicativos evoluem.

Você também pode invocar execuções de máquinas de estado aninhadas para que elas estejam disponíveis em diferentes contas. Fazer isso separa e isola seus fluxos de trabalho de forma eficiente. Ao usar o padrão de integração de serviços de `.sync` em seus fluxos de trabalho que acessam outro fluxo de trabalho do Step Functions em uma conta diferente, o Step Functions usa uma sondagem que consome sua cota atribuída. Para ter mais informações, consulte Executar um trabalho (.sync).

> **Note**
>
> Atualmente, a integração entre regiões do AWS SDK e o acesso a AWS recursos entre regiões não estão disponíveis no Step Functions.

Conteúdo

Principais conceitos neste tópico

Perfil de execução

Uma função do IAM que Step Functions usa para executar código e acessar AWS recursos, como a ação Invoke da AWS Lambda função.

Integração de serviços

As ações da API de integração do AWS SDK que podem ser chamadas de dentro de um `Task` estado em seus fluxos de trabalho.

Conta de origem

E Conta da AWS que é dona da máquina estatal e iniciou sua execução.

Conta de destino

E Conta da AWS para a qual você faz chamadas entre contas.

Perfil de destino

Um perfil do IAM na conta de destino que a máquina de estado assume para fazer chamadas para recursos que a conta de destino possui.

Executar um trabalho (.sync)

Um padrão de integração de serviços usado para chamar serviços, como AWS Batch. Também faz com que uma máquina de estado do Step Functions espere a conclusão de um trabalho antes de avançar para o próximo estado. Para indicar que o Step Functions deve aguardar, acrescente o sufixo `.sync` no campo `Resource` na definição do estado `Task`.

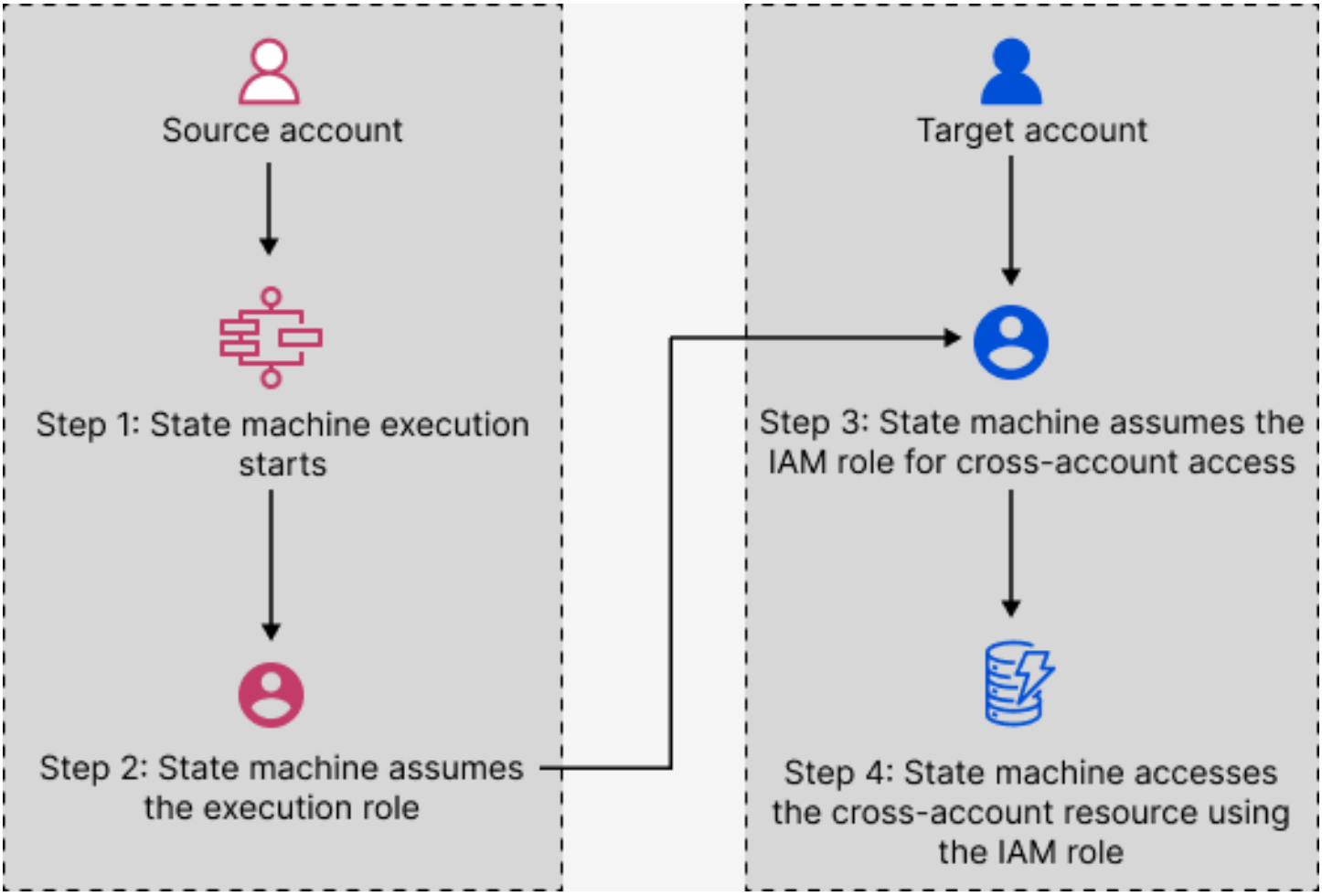
Invocar recursos entre contas

Para invocar um recurso entre contas em seus fluxos de trabalho, faça o seguinte:

1. Crie um perfil do IAM na conta de destino que contém o recurso. Esse perfil concede permissões à conta de origem que contém a máquina de estado para acessar os recursos da conta de destino.
2. Na definição do estado da `Task`, especifique o perfil do IAM de destino a ser assumido pela máquina de estado antes de invocar o recurso entre contas.
3. Modifique a política de confiança no perfil do IAM de destino para permitir que a conta de origem assuma esse perfil temporariamente. A política de confiança deve incluir o nome do recurso da Amazon (ARN) da máquina de estado definida na conta de origem. Além disso, defina as permissões apropriadas na função de destino do IAM para chamar o AWS recurso.
4. Atualize o perfil de execução da conta de origem para incluir a permissão necessária para assumir o perfil do IAM de destino.

Para ver um exemplo, consulte Tutorial: Acessando recursos entre contas AWS.

> **Note**
>
> Você pode configurar sua máquina de estado para assumir um perfil do IAM para acessar recursos de várias Contas da AWS. No entanto, uma máquina de estado pode assumir somente um perfil do IAM por vez.



Tutorial: Acessando recursos entre contas AWS

Com o suporte de acesso entre contas no Step Functions, você pode compartilhar recursos configurados em diferentes Contas da AWS. Neste tutorial, mostraremos o processo de acesso a uma função do Lambda entre contas definida em uma conta denominada Produção. Essa função é invocada de uma máquina de estado em uma conta denominada Desenvolvimento. Neste tutorial, a conta de Desenvolvimento é chamada de conta de origem, enquanto a conta de Produção é a conta de destino que contém o perfil do IAM de destino.

Para começar, na definição do estado `Task`, você especifica o perfil do IAM de destino que a máquina de estado deve assumir antes de invocar a função do Lambda entre contas. Em seguida, modifique a política de confiança no perfil do IAM de destino para permitir que a conta de origem assuma o perfil de destino temporariamente. Além disso, para chamar o AWS recurso, defina as permissões apropriadas na função de destino do IAM. Por fim, atualize o perfil de execução da conta de origem para especificar a permissão necessária para assumir o perfil de destino.

Você pode configurar sua máquina de estado para assumir um perfil do IAM para acessar recursos de várias Contas da AWS. No entanto, uma máquina de estado pode assumir somente um perfil do IAM por vez com base na definição do estado da `Task`.

> **Note**
>
> Atualmente, a integração entre regiões do AWS SDK e o acesso a AWS recursos entre regiões não estão disponíveis no Step Functions.

Conteúdos

Pré-requisitos

- Este tutorial usa o exemplo de uma função do Lambda para demonstrar como configurar o acesso entre contas. Você pode usar qualquer outro AWS recurso, mas certifique-se de ter configurado o recurso em uma conta diferente.

> **Important**
>
> As funções do IAM e as políticas baseadas em recurso delegam o acesso entre contas em uma única partição. Por exemplo, suponha que você tenha uma conta no Oeste dos EUA (Norte da Califórnia) na partição `aws` padrão. Além disso, você tem uma conta na China (Pequim) na partição `aws-cn`. Você não pode usar uma política baseada em recurso do

Amazon S3 em sua conta na China (Pequim) para permitir o acesso de usuários em sua conta `aws` padrão.

- Anote o nome do recurso da Amazon (ARN) do recurso entre contas em um arquivo de texto. Posteriormente neste tutorial, você fornecerá esse ARN na definição do estado `Task` da sua máquina de estado. Veja a seguir um exemplo de um ARN de função do Lambda:

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

- Crie o perfil do IAM de destino que a máquina de estado precisa assumir.

Etapa 1: atualizar a definição do estado Tarefa para especificar o perfil de destino

No estado `Task` do fluxo de trabalho, adicione um campo `Credentials` contendo a identidade que a máquina de estado deve assumir antes de invocar a função do Lambda entre contas.

O procedimento a seguir demonstra como acessar uma função do Lambda entre contas chamada `Echo`. Você pode chamar qualquer AWS recurso seguindo estas etapas.

1. Abra o console do Step Functions e clique em Criar máquina de estado.
2. Na página Escolher método de criação, escolha Projetar seu fluxo de trabalho visualmente e mantenha todas as seleções padrão.
3. Para abrir o Workflow Studio, escolha Próximo.
4. Na guia Ações, arraste e solte o estado `Task` na tela. Isso invoca a função do Lambda entre contas que está usando esse estado da `Task`.
5. Na guia Configuração, faça o seguinte:
   a. Renomeie o estado para **`Cross-account call`**.
   b. Em Nome da função, escolha Inserir nome da função e, em seguida, insira o ARN da função do Lambda na caixa. Por exemplo, `arn:aws:lambda:us-east-2:111122223333:function:Echo`.
   c. Em Fornecer ARN do perfil do IAM, especifique o ARN do perfil do IAM. Por exemplo, `arn:aws:iam::111122223333:role/LambdaRole`.

> **Tip**
>
> Como alternativa, você também pode especificar um caminho de referência para um par de valores-chave existente na entrada JSON do estado que contém o ARN do perfil do IAM. Para fazer isso, escolha Obter ARN do perfil do IAM em runtime na entrada de estado. Para obter um exemplo de especificação de um valor usando um caminho de referência, consulte Especificar o JSONPath como ARN do perfil do IAM.

6. Escolha Próximo.
7. Na página Revisar código gerado, escolha Próximo.
8. Na página Especificar configurações da máquina de estado, especifique detalhes da nova máquina de estado, como nome, permissões e nível de registro em log.
9. Escolha Criar uma máquina de estado.
10. Anote o ARN do perfil do IAM e o ARN da máquina de estado em um arquivo de texto. Você precisará apresentar esses ARNs na política de confiança da conta de destino.

A definição do estado da `Task` agora deve ser semelhante à definição a seguir.

```
{
  "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
    }
  }
}
```

Etapa 2: atualizar a política de confiança do perfil de destino

O perfil do IAM deve existir na conta de destino e você deve modificar sua política de confiança para permitir que a conta de origem assuma esse perfil temporariamente. Além disso, você pode controlar quem pode assumir o perfil do IAM de destino.

Depois de criar a relação de confiança, um usuário da conta de origem pode usar a operação da AssumeRoleAPI AWS Security Token Service (AWS STS). Essa operação fornece credenciais de segurança temporárias que permitem o acesso aos AWS recursos em uma conta de destino.

1. Abra o console do IAM em https://console.aws.amazon.com/iam/.
2. No painel de navegação do console, escolha Perfis e, em seguida, use a caixa Pesquisar para pesquisar o perfil do IAM de destino. Por exemplo, *LambdaRole*.
3. Selecione a guia Trust relationships (Relações de confiança).
4. Selecione Editar política de confiança e cole a seguinte política de confiança. Certifique-se de substituir o Conta da AWS número e o ARN da função do IAM. O campo `sts:ExternalId` controla ainda mais quem pode assumir o perfil. O nome da máquina de estado deve incluir somente caracteres compatíveis com a AWS Security Token Service `AssumeRole` API. Para obter mais informações, consulte AssumeRolea Referência AWS Security Token Service da 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. Mantenha essa janela aberta e prossiga para a próxima etapa para realizar outras ações.

Etapa 3: adicionar a permissão necessária no perfil de destino

As permissões nas políticas do IAM determinam se uma solicitação específica será permitida ou negada. O perfil do IAM de destino deve ter a permissão correta para invocar a função do Lambda.

1. Escolha a aba Permissões.
2. Escolha Adicionar permissões e depois Criar política em linha.
3. Escolha a guia JSON e substitua o conteúdo existente pela permissão a seguir. Substitua o ARN da função do 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. Escolha Revisar política.
5. Na página Revisar política, insira um nome da permissão e, em seguida, selecione Criar política.

Etapa 4: adicionar permissão no perfil de execução para assumir o perfil de destino

O Step Functions não gera automaticamente a AssumeRolepolítica para todas as integrações de serviços entre contas. Você deve adicionar a permissão necessária no perfil de execução da máquina de estado para permitir que ela assuma um perfil do IAM de destino em uma ou mais Contas da AWS.

1. Abra o perfil de execução da sua máquina de estado no console do IAM em https://console.aws.amazon.com/iam/. Para fazer isso:

   a. Abra a máquina de estado que você criou na Etapa 1 na conta de origem.

b. Na página Detalhes da máquina de estado, selecione ARN do perfil do IAM.

2. Na guia Permissões, selecione Adicionar permissões e Criar política em linha.
3. Escolha a guia JSON e substitua o conteúdo existente pela permissão a seguir. Substitua o ARN da função do 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. Escolha Revisar política.
5. Na página Revisar política, insira um nome da permissão e, em seguida, selecione Criar política.

Acesso entre contas para o padrão de integração .sync

Ao usar os padrões de integração de serviços `.sync` em seus fluxos de trabalho, o Step Functions pesquisa o recurso entre contas invocado para confirmar que a tarefa foi concluída. Isso causa um pequeno atraso entre o tempo real de conclusão da tarefa e o momento em que o Step Functions reconhece a tarefa como concluída. O perfil do IAM de destino precisa das permissões necessárias para que uma invocação `.sync` conclua esse ciclo de sondagem. Para fazer isso, o perfil do IAM de destino deve ter uma política de confiança que permita que a conta de origem a assuma. Além disso, o perfil do IAM de destino precisa das permissões necessárias para concluir o ciclo de sondagem.

> **Note**
>
> Para fluxos de trabalho expressos aninhados, `arn:aws:states:::states:startExecution.sync` não é compatível no momento. Use `arn:aws:states:::aws-sdk:sfn:startSyncExecution` em vez disso.

Atualização da política de confiança para chamadas .sync

Atualize a política de confiança do perfil do IAM de destino, conforme mostrado no exemplo a seguir. O campo `sts:ExternalId` controla ainda mais quem pode assumir o perfil. O nome da máquina de estado deve incluir somente caracteres compatíveis com a AWS Security Token Service `AssumeRole` API. Para obter mais informações, consulte AssumeRolea Referência AWS Security Token Service da 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"
        }
      }
    }
  ]
}
```

Permissões necessárias para chamadas .sync

Para conceder as permissões necessárias para sua máquina de estado, atualize-as para o perfil do IAM de destino. Para ter mais informações, consulte the section called “Políticas do IAM para serviços integrados”. As EventBridge permissões da Amazon das políticas de exemplo não são necessárias. Por exemplo, para iniciar uma máquina de estado, adicione as seguintes permissões.

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

Endpoints da Amazon VPC para o Step Functions

Se você usa a Amazon Virtual Private Cloud (Amazon VPC) para hospedar seus AWS recursos, você pode estabelecer uma conexão entre sua Amazon VPC e fluxos de trabalho. AWS Step Functions É possível usar essa conexão com seus fluxos de trabalho do Step Functions sem passar pela internet pública. Os endpoints da Amazon VPC são compatíveis com fluxos de trabalho padrão, fluxos de trabalho expressos e fluxos de trabalho expressos síncronos.

A Amazon VPC permite que você lance AWS recursos em uma rede virtual personalizada. Você pode usar uma VPC para controlar as configurações de rede, como o intervalo de endereços IP, sub-redes, tabelas de rotas e gateways de rede. Para obter informações sobre como criar suas próprias VPCs, consulte o Guia do usuário da Amazon VPC.

Para conectar sua Amazon VPC ao Step Functions, você deve primeiro definir uma interface VPC endpoint, que permite conectar sua VPC a outros serviços. AWS O endpoint fornece conectividade confiável e escalável sem a necessidade de um gateway da Internet, da instância de conversão de endereço de rede (NAT) ou de uma conexão VPN. Para obter mais informações, consulte Endpoints da VPC da interface (AWS PrivateLink) no Guia do usuário da Amazon VPC.

Criação do endpoint

Você pode criar um AWS Step Functions endpoint em sua VPC usando AWS Management Console o, AWS Command Line Interface the AWS CLI(), AWS um SDK, AWS Step Functions a API ou. AWS CloudFormation

Para obter informações sobre como criar e configurar um endpoint usando o console da Amazon VPC ou a AWS CLI, consulte Creating an Interface Endpoint (“Criar um endpoint da interface”) no Manual do usuário da Amazon VPC.

> **Note**
>
> Ao criar um endpoint, especifique o Step Functions como o serviço ao qual a VPC deve se conectar. No console da Amazon VPC, os nomes dos serviços variam de acordo com a AWS região. Por exemplo, se você escolher Leste dos EUA (Norte da Virgínia), o nome do serviço para fluxos de trabalho padrão e expressos será com.amazonaws.us-east-1.states, e o nome do serviço para fluxos de trabalho expressos síncronos será com.amazonaws.us-east-1.sync-states.

> **Note**
>
> É possível usar Endpoints da VPC sem substituir o endpoint no SDK por meio do DNS privado. No entanto, para substituir o endpoint no SDK para fluxos de trabalho expressos síncronos, será necessário definir a configuração `DisableHostPrefixInjection` como `true`. Exemplo (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 obter informações sobre como criar e configurar um endpoint usando AWS CloudFormation, consulte o recurso AWS: :EC2: :VPCendpoint no Guia do usuário.AWS CloudFormation

Políticas de endpoint da Amazon VPC

Para controlar o acesso de conectividade ao Step Functions, você pode anexar uma política de endpoint AWS Identity and Access Management (IAM) ao criar um endpoint da Amazon VPC. É

possível criar regras complexas do IAM anexando várias políticas de endpoint. Para obter mais informações, consulte:

- Políticas de endpoint da Amazon Virtual Private Cloud para o Step Functions
- Como criar permissões granulares do IAM para usuários que não são administradores
- Controle do acesso a serviços com VPC endpoints

Políticas de endpoint da Amazon Virtual Private Cloud para o Step Functions

É possível criar uma política de endpoint de Amazon VPC para o Step Functions na qual você especifica o seguinte:

- A entidade principal que pode executar ações.
- As ações que podem ser executadas.
- Os recursos nos quais as ações podem ser executadas.

O exemplo a seguir mostra uma política de endpoint da Amazon VPC que permite que um usuário crie máquinas de estado e nega a todos os outros usuários a permissão para excluir máquinas de estado. A política de exemplo também concede permissão de execução a todos os usuários do .

```
{
    "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 obter mais informações sobre como criar políticas de endpoint, consulte o seguinte:

- Como criar permissões granulares do IAM para usuários que não são administradores
- Controle do acesso a serviços com VPC endpoints

Políticas do IAM para serviços integrados

Quando você cria uma máquina de estado no AWS Step Functions console, o Step Functions produz uma política AWS Identity and Access Management (IAM) com base nos recursos usados na definição da máquina de estado da seguinte forma:

- Se sua máquina de estado usar uma das integrações otimizadas, o Step Functions criará uma política com as permissões e funções necessárias para sua máquina de estado. (Exceção: a MediaConvert integração exige que você configure manualmente as permissões — consulte Políticas do IAM para AWS Elemental MediaConvert.)
- Se sua máquina de estado usar uma das integrações do AWS SDK, uma função do IAM com permissões parciais será criada. Depois, você poderá usar o console do IAM para adicionar quaisquer políticas de perfil ausentes.

Os exemplos a seguir mostram como o Step Functions gera uma política do IAM com base na definição da máquina de estado. Os itens no exemplo de código, como *`[[resourceName]]`* são substituídos pelos recursos estáticos listados na definição da máquina de estado. Se você tiver vários recursos estáticos, haverá uma entrada para cada um no perfil do IAM.

Recursos dinâmicos e estáticos

Os recursos estáticos são definidos diretamente no estado da tarefa da máquina de estado. Ao incluir as informações sobre os recursos que você deseja chamar diretamente nos estados Tarefa, o Step Functions cria um perfil do IAM somente para esses recursos.

Os recursos dinâmicos são aqueles transmitidos para a entrada de estado e acessados usando um caminho (consulte Caminhos). Se você estiver transmitindo recursos dinâmicos para sua tarefa, o Step Functions criará uma política mais privilegiada que especifica: `"Resource": "*"`.

Permissões adicionais para tarefas usando o padrão Executar um trabalho

Para tarefas que usam o padrão Executar um trabalho (aquelas que terminam em `.sync`), são necessárias permissões adicionais para monitorar e receber uma resposta das ações de API dos serviços conectados. As políticas relacionadas incluem mais permissões do que para tarefas que usam os padrões Solicitar resposta ou Aguardar o retorno de chamada. Consulte Padrões de integração de serviço para obter informações sobre tarefas síncronas.

> **Note**
>
> Você precisa fornecer permissões adicionais para integrações de serviços que ofereçam suporte ao padrão Run a Job (.sync).

O Step Functions usa dois métodos para monitorar o status de um trabalho quando um trabalho é executado em um serviço conectado, sondagem e eventos.

A sondagem requer permissão para as ações de API `Describe` ou `Get`, como `ecs:DescribeTasks` ou `glue:GetJobRun`. Se essas permissões estiverem ausentes em seu perfil, talvez o Step Functions não consiga determinar o status do trabalho. Isso ocorre porque algumas integrações de serviços Run a Job (.sync) não oferecem suporte a EventBridge eventos, e alguns serviços só enviam eventos com base no melhor esforço.

Os eventos enviados de AWS serviços para a Amazon EventBridge são direcionados ao Step Functions usando uma regra gerenciada e exigem permissões para `events:PutTargetsevents:PutRule`, `events:DescribeRule` e. Se essas permissões estiverem ausentes em seu perfil, pode haver um atraso antes que o Step Functions tome conhecimento da conclusão de seu trabalho. Para obter mais informações sobre EventBridge eventos, consulte Eventos de AWS serviços.

> **Note**
>
> Para tarefas Executar um trabalho (.sync) compatíveis com pesquisas e eventos, sua tarefa ainda pode ser concluída corretamente usando eventos. Isso pode ocorrer mesmo que

seu perfil não tenha as permissões necessárias para a sondagem. Nesse caso, talvez você não perceba imediatamente que as permissões de sondagem estão incorretas ou ausentes. Na raras instâncias em que o evento não é entregue ou processado pelo Step Functions, sua execução pode ficar travada. Para verificar se suas permissões de enquete estão configuradas corretamente, você pode executar uma execução em um ambiente sem EventBridge eventos das seguintes formas:

- Exclua a regra gerenciada de EventBridge, que é responsável por encaminhar eventos para Step Functions. Essa regra gerenciada é compartilhada por todas as máquinas de estado em sua conta; portanto, você deve realizar essa ação somente em uma conta de teste ou desenvolvimento para evitar qualquer impacto não intencional em outras máquinas de estado. Você pode identificar a regra gerenciada específica a ser excluída inspecionando o campo `Resource` usado para `events:PutRule` no modelo de política do serviço de destino. A regra gerenciada será recriada na próxima vez que você criar ou atualizar uma máquina de estado que usa essa integração de serviços. Para obter mais informações sobre como excluir EventBridge regras, consulte Como desativar ou excluir uma regra.
- Teste com o Step Functions Local, que não é compatível com o uso de eventos para concluir tarefas Executar um trabalho (.sync). Para usar o Step Functions Local, assuma o perfil do IAM usado pela sua máquina de estado. Pode ser necessário editar o relacionamento de confiança. Defina as variáveis de ambiente `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY` e `AWS_SESSION_TOKEN` com os valores do perfil assumido e, em seguida, inicie o Step Functions Local usando java -jar StepFunctionsLocal.jar. Por fim, use o --endpoint-url parâmetro AWS CLI with the para criar uma máquina de estado, iniciar uma execução e obter o histórico de execução. Para ter mais informações, consulte Testando máquinas de estado localmente.

Se uma tarefa que usa o padrão Executar um trabalho (.sync) for interrompida, o Step Functions fará o possível para cancelar a tarefa. A sondagem requer permissão para as ações de API `Cancel`, `Stop`, `Terminate` ou `Delete`, como `batch:TerminateJob` ou `eks:DeleteCluster`. Se essas permissões estiverem ausentes em seu perfil, o Step Functions não poderá cancelar sua tarefa e você poderá acumular cobranças adicionais enquanto ela continuar em execução. Para ver mais informações sobre como interromper tarefas, consulte Executar um trabalho.

Modelos de política usados para criar perfis do IAM

Os tópicos a seguir incluem os modelos de política usados ao escolher que o Step Functions crie um novo perfil para você.

> **Note**
>
> Analise esses modelos para entender como o Step Functions cria suas políticas do IAM e como um exemplo de como criar manualmente políticas do IAM para o Step Functions ao trabalhar com outros AWS serviços. Para ver mais informações sobre integrações de serviços do Step Functions, consulte Usando AWS Step Functions com outros serviços.

Tópicos

- Políticas do IAM para AWS Step Functions
- Políticas do IAM para AWS X-Ray
- Tarefas de atividade ou nenhuma tarefa

Políticas do IAM para o Amazon API Gateway

Os modelos de exemplo a seguir mostram como AWS Step Functions gera políticas do IAM com base nos recursos na definição da sua máquina de estado. Para obter mais informações, consulte Políticas do IAM para serviços integrados e Padrões de integração de serviço.

Recursos:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "execute-api:Invoke"
            ],
            "Resource": [
                "arn:[[region]]:[[accountId]]:*"
            ]
        }
    ]
}
```

O exemplo de código a seguir mostra uma política de recursos para chamar o 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 do IAM para o Amazon Athena

Os modelos de exemplo a seguir mostram como AWS Step Functions gera políticas do IAM com base nos recursos na definição da sua máquina de estado. Para obter mais informações, consulte Políticas do IAM para serviços integrados e Padrões de integração de serviço.

`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 dinâmicos

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 do IAM para AWS Batch

Os modelos de exemplo a seguir mostram como AWS Step Functions gera políticas do IAM com base nos recursos na definição da sua máquina de estado. Para obter mais informações, consulte Políticas do IAM para serviços integrados e Padrões de integração de serviço.

Como AWS Batch fornece suporte parcial para controle de acesso em nível de recurso, você deve usar. `"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

Ao criar uma máquina de estado usando o console, o Step Functions cria automaticamente um perfil de execução para a máquina de estado com os privilégios mínimos necessários. Essas IAM funções geradas automaticamente são válidas para a máquina Região da AWS em que você cria a máquina de estado.

Os modelos de exemplo a seguir mostram como AWS Step Functions gera políticas do IAM com base nos recursos na definição da sua máquina de estado. Para obter mais informações, consulte Políticas do IAM para serviços integrados e Padrões de integração de serviço.

Recomendamos não incluir curingas ao criar políticas do IAM. Como prática recomendada de segurança, você deve definir o escopo de suas políticas o máximo possível. Use políticas dinâmicas somente quando determinados parâmetros de entrada não forem conhecidos durante o runtime.

Neste tópico

Exemplos de políticas do IAM para integração do Amazon Bedrock ao Step Functions

A seção a seguir descreve as permissões do IAM necessárias com base na API do Amazon Bedrock utilizada para uma base específica ou um modelo provisionado. Esta seção também contém exemplos de políticas que concedem acesso total.

Lembre-se de substituir o texto em *`itálico`* pelas informações específicas do recurso.

- IAMexemplo de política para acessar um modelo de fundação específico usando InvokeModel
- IAMexemplo de política para acessar um modelo provisionado específico usando InvokeModel

- Exemplo de IAM política de acesso completo para usar InvokeModel
- Exemplo de política do IAM que acessa um modelo de base específico como modelo de base
- Exemplo de política do IAM que acessa um modelo personalizado específico como modelo de base
- Exemplo de IAM política de acesso completo para usar CreateModelCustomizationJob .sync
- IAMexemplo de política para acessar um modelo de fundação específico usando CreateModelCustomizationJob .sync
- IAMexemplo de política para acessar um modelo personalizado usando CreateModelCustomizationJob .sync
- Exemplo de IAM política de acesso completo para usar CreateModelCustomizationJob .sync

IAMexemplo de política para acessar um modelo de fundação específico usando InvokeModel

Veja a seguir um exemplo IAM de política para uma máquina de estado que acessa um modelo básico específico chamado `amazon.titan-text-express-v1` usando a ação da 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"
            ]
        }
    ]
}
```

IAMexemplo de política para acessar um modelo provisionado específico usando InvokeModel

Veja a seguir um exemplo IAM de política para uma máquina de estado que acessa um modelo provisionado específico chamado `c2oi931ulksx` usando a ação da InvokeModelAPI.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Sid": "InvokeModel1",
      "Action": [
        "bedrock:InvokeModel"
      ],
      "Resource": [
        "arn:aws:bedrock:us-east-2:123456789012:provisioned-model/c2oi931ulksx"
      ]
    }
  ]
}
```

Exemplo de IAM política de acesso completo para usar InvokeModel

Veja a seguir um exemplo IAM de política para uma máquina de estado que fornece acesso total quando você usa a ação da 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/*"
            ]
        }
    ]
}
```

Exemplo de política do IAM que acessa um modelo de base específico como modelo de base

Veja a seguir um exemplo IAM de política para que uma máquina de estado acesse um modelo básico específico `amazon.titan-text-express-v1` chamado de modelo básico usando a ação da 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"
            ]
        }
    ]
}
```

Exemplo de política do IAM que acessa um modelo personalizado específico como modelo de base

Veja a seguir um exemplo IAM de política para que uma máquina de estado acesse um modelo personalizado específico como modelo base usando a ação da 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]]"
            ]
        }
    ]
}
```

Exemplo de IAM política de acesso completo para usar CreateModelCustomizationJob .sync

Veja a seguir um exemplo IAM de política para uma máquina de estado que fornece acesso total quando você usa a ação da 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"
            ]
        }
    ]
}
```

IAMexemplo de política para acessar um modelo de fundação específico usando CreateModelCustomizationJob .sync

Veja a seguir um exemplo IAM de política para que uma máquina de estado acesse um modelo básico específico chamado `amazon.titan-text-express-v1` usando a ação da 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"
            ]
        }
    ]
}
```

IAMexemplo de política para acessar um modelo personalizado usando CreateModelCustomizationJob .sync

Veja a seguir um exemplo IAM de política para uma máquina de estado acessar um modelo personalizado usando a ação da 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"
            ]
        }
    ]
}
```

Exemplo de IAM política de acesso completo para usar CreateModelCustomizationJob .sync

Veja a seguir um exemplo IAM de política para uma máquina de estado que fornece acesso total quando você usa a ação da 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 do IAM para AWS CodeBuild

Os modelos de exemplo a seguir mostram como AWS Step Functions gera políticas do IAM com base nos recursos na definição da sua máquina de estado. Para obter mais informações, consulte Políticas do IAM para serviços integrados e Padrões de integração de serviço.

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 dinâmicos

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 dinâmicos

```
{
  "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 dinâmicos

```
{
  "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 dinâmicos

```
{
  "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 dinâmicos

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 dinâmicos

```
{
    "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 dinâmicos

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 dinâmicos

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codebuild:DeleteBuildBatch"
            ],
            "Resource": [
                "arn:aws:codebuild:[[region]]:[[accountId]]:project/*"
            ]
        }
    ]
}
```

Políticas do IAM para o Amazon DynamoDB

Os modelos de exemplo a seguir mostram como AWS Step Functions gera políticas do IAM com base nos recursos na definição da sua máquina de estado. Para obter mais informações, consulte Políticas do IAM para serviços integrados e Padrões de integração de serviço.

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 dinâmicos

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "dynamodb:GetItem",
                "dynamodb:PutItem",
                "dynamodb:UpdateItem",
                "dynamodb:DeleteItem"
            ],
            "Resource": "*"
        }
    ]
}
```

Para ver mais informações sobre as políticas do IAM para todas as ações de API do DynamoDB, consulte políticas do IAM com o DynamoDB no Guia do desenvolvedor do Amazon DynamoDB. Para ver mais informações sobre as políticas do IAM para o PartiQL for DynamoDB, consulte políticas do IAM com o PartiQL for DynamoDB no Guia do desenvolvedor do Amazon DynamoDB.

Políticas do IAM para Amazon ECS/AWS Fargate

Os modelos de exemplo a seguir mostram como AWS Step Functions gera políticas do IAM com base nos recursos na definição da sua máquina de estado. Para obter mais informações, consulte Políticas do IAM para serviços integrados e Padrões de integração de serviço.

Uma vez que o valor de `TaskId` não é conhecido até que a tarefa seja enviada, o Step Functions cria uma política de `"Resource": "*"` com mais privilégios.

> **Note**
>
> Você só pode interromper tarefas do Amazon Elastic Container Service (Amazon ECS) que foram iniciadas pelo Step Functions, apesar da política do 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 dinâmicos

```
{
    "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 dinâmicos

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:RunTask"
            ],
            "Resource": "*"
        }
    ]
}
```

Se suas tarefas agendadas do Amazon ECS exigirem o uso de uma função de execução de tarefa, uma função de tarefa ou uma substituição de função de tarefa, você deverá adicionar `iam:PassRole` permissões para cada função de execução de tarefa, função de tarefa ou substituição de função de tarefa à função CloudWatch Events IAM da entidade chamadora, que neste caso é Step Functions.

Políticas do IAM para o Amazon EKS

Os modelos de exemplo a seguir mostram como AWS Step Functions gera políticas do IAM com base nos recursos na definição da sua máquina de estado. Para obter mais informações, consulte Políticas do IAM para serviços integrados e Padrões de integração de serviço.

`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 ver mais informações sobre o uso do Amazon EKS com o Step Functions, consulte Chamar o Amazon EKS com o Step Functions.

Políticas do IAM para o Amazon EMR

Os modelos de exemplo a seguir mostram como AWS Step Functions gera políticas do IAM com base nos recursos na definição da sua máquina de estado. Para obter mais informações, consulte Políticas do IAM para serviços integrados e Padrões de integração de serviço.

`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 dinâmicos

```
{
  "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 dinâmicos

```
{
    "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 dinâmicos

```
{
    "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 dinâmicos

```
{
    "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 dinâmicos

```
{
    "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 dinâmicos

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "elasticmapreduce:TerminateJobFlows",
        "elasticmapreduce:DescribeCluster"
      ],
      "Resource": "arn:aws:elasticmapreduce:*:*:cluster/*"
    }
  ]
}
```

Políticas do IAM para o Amazon EMR no EKS

Os modelos de exemplo a seguir mostram como AWS Step Functions gera políticas do IAM com base nos recursos na definição da sua máquina de estado. Para obter mais informações, consulte Políticas do IAM para serviços integrados e Padrões de integração de serviço.

`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 dinâmicos

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 dinâmicos

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 do IAM para o Amazon EMR Serverless

Ao criar uma máquina de estado usando o console, o Step Functions cria automaticamente um perfil de execução para a máquina de estado com os privilégios mínimos necessários. Essas IAM funções geradas automaticamente são válidas para a máquina Região da AWS em que você cria a máquina de estado.

Os modelos de exemplo a seguir mostram como AWS Step Functions gera políticas do IAM com base nos recursos na definição da sua máquina de estado. Para obter mais informações, consulte Políticas do IAM para serviços integrados e Padrões de integração de serviço.

Recomendamos não incluir curingas ao criar políticas do IAM. Como prática recomendada de segurança, você deve definir o escopo de suas políticas o máximo possível. Use políticas dinâmicas somente quando determinados parâmetros de entrada não forem conhecidos durante o runtime.

Além disso, os usuários administradores devem ter cuidado ao conceder perfis de execução a usuários não administradores para executar as máquinas de estado. Recomendamos que você inclua políticas passRole nos perfis de execução se estiver criando políticas por conta própria. Também recomendamos que você adicione as chaves de contexto `aws:SourceARN` e `aws:SourceAccount` e nos perfis de execução.

Neste tópico

Exemplos de políticas do IAM para integração do EMR sem servidor com o Step Functions

Exemplo de política do IAM para CreateApplication

Veja a seguir um exemplo de política do IAM para uma máquina de estado com um CreateApplication Estado da tarefa estado.

> **Note**
>
> Você precisa especificar as CreateServiceLinkedRole permissões em suas políticas do IAM durante a criação do primeiro aplicativo em sua conta. Depois disso, não será necessário adicionar essa permissão. Para obter informações sobre isso CreateServiceLinkedRole, consulte CreateServiceLinkedRoleem https://docs.aws.amazon.com/IAM/latest/APIReference/.

Os recursos estáticos e dinâmicos para as políticas a seguir são os mesmos.

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"
                }
            }
        }
    ]
}
```

Exemplo de política do IAM para StartApplication

Recursos estáticos

Veja a seguir exemplos de políticas do IAM para recursos estáticos quando você usa uma máquina de estado com um StartApplication Estado da tarefa 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 dinâmicos

Veja a seguir exemplos de políticas do IAM para recursos dinâmicos quando você usa uma máquina de estado com um StartApplication Estado da tarefa 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/*"
            ]
        }
    ]
}
```

Exemplo de política do IAM para StopApplication

Recursos estáticos

Veja a seguir exemplos de políticas do IAM para recursos estáticos quando você usa uma máquina de estado com um StopApplication Estado da tarefa 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 dinâmicos

Veja a seguir exemplos de políticas do IAM para recursos dinâmicos quando você usa uma máquina de estado com um StopApplication Estado da tarefa 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/*"
            ]
        }
    ]
}
```

Exemplo de política do IAM para DeleteApplication

Recursos estáticos

Veja a seguir exemplos de políticas do IAM para recursos estáticos quando você usa uma máquina de estado com um DeleteApplication Estado da tarefa 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 dinâmicos

Veja a seguir exemplos de políticas do IAM para recursos dinâmicos quando você usa uma máquina de estado com um DeleteApplication Estado da tarefa 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/*"
             ]
        }
    ]
}
```

Exemplo de política do IAM para StartJobRun

Recursos estáticos

Veja a seguir exemplos de políticas do IAM para recursos estáticos quando você usa uma máquina de estado com um StartJobRun Estado da tarefa 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 dinâmicos

Veja a seguir exemplos de políticas do IAM para recursos dinâmicos quando você usa uma máquina de estado com um StartJobRun Estado da tarefa 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"
                }
            }
        }
    ]
}
```

Exemplo de política do IAM para CancelJobRun

Recursos estáticos

Veja a seguir exemplos de políticas do IAM para recursos estáticos quando você usa uma máquina de estado com um CancelJobRun Estado da tarefa 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 dinâmicos

Veja a seguir exemplos de políticas do IAM para recursos dinâmicos quando você usa uma máquina de estado com um CancelJobRun Estado da tarefa 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 do IAM para a Amazon EventBridge

Os modelos de exemplo a seguir mostram como AWS Step Functions gera políticas do IAM com base nos recursos na definição da sua máquina de estado. Para obter mais informações, consulte Políticas do IAM para serviços integrados e Padrões de integração de serviço.

`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 dinâmicos

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

Para obter mais informações sobre como usar EventBridge com Step Functions, consulteChamada EventBridge com Step Functions.

Políticas do IAM para AWS Lambda

Os modelos de exemplo a seguir mostram como AWS Step Functions gera políticas do IAM com base nos recursos na definição da sua máquina de estado. Para obter mais informações, consulte Políticas do IAM para serviços integrados e Padrões de integração de serviço.

AWS Step Functions gera uma política do IAM com base na definição da sua máquina de estado. Para uma máquina de estado com dois estados de AWS Lambda tarefa que chamam `function1` e `function2`, uma política com `lambda:Invoke` permissões para as duas funções deve ser usada.

Isso é mostrado no exemplo a seguir.

```
{
    "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 do IAM para AWS Elemental MediaConvert

Os modelos de exemplo a seguir mostram como AWS Step Functions exige que você configure suas políticas do IAM com base nos recursos em sua definição de máquina de estado. Você pode usar o console do IAM para adicionar quaisquer políticas de função ausentes. Para obter mais informações, consulte Políticas do IAM para serviços integrados e Padrões de integração de serviço.

Como MediaConvert fornece suporte parcial para controle de acesso em nível de recurso, você deve usar. `"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 do IAM para AWS Glue

Os modelos de exemplo a seguir mostram como AWS Step Functions gera políticas do IAM com base nos recursos na definição da sua máquina de estado. Para obter mais informações, consulte Políticas do IAM para serviços integrados e Padrões de integração de serviço.

AWS Glue não tem controle baseado em 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 do IAM para AWS Glue DataBrew

Os modelos de exemplo a seguir mostram como AWS Step Functions gera políticas do IAM com base nos recursos na definição da sua máquina de estado. Para obter mais informações, consulte Políticas do IAM para serviços integrados e Padrões de integração de serviço.

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 do IAM para a Amazon SageMaker

Os modelos de exemplo a seguir mostram como AWS Step Functions gera políticas do IAM com base nos recursos na definição da sua máquina de estado. Para obter mais informações, consulte Políticas do IAM para serviços integrados e Padrões de integração de serviço.

> **Note**
>
> Para esses exemplos, *[[roleArn]]* consulte o Amazon Resource Name (ARN) da função do IAM SageMaker usada para acessar artefatos de modelo e imagens do docker para

implantação em instâncias de computação de ML ou para trabalhos de transformação em lote. Para obter mais informações, 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 dinâmicos

.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 não criará automaticamente uma política para `CreateTransformJob` quando você criar uma máquina de estado que se integre a. SageMaker Você deve anexar uma política em linha ao perfil criado com base em um dos exemplos do IAM a seguir.

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 dinâmicos

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 do IAM para Amazon SNS

Os modelos de exemplo a seguir mostram como AWS Step Functions gera políticas do IAM com base nos recursos na definição da sua máquina de estado. Para obter mais informações, consulte Políticas do IAM para serviços integrados e Padrões de integração de serviço.

Recursos estáticos

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sns:Publish"
            ],
            "Resource": [
                "arn:aws:sns:[[region]]:[[accountId]]:[[topicName]]"
```

```
            ]
        }
    ]
}
```

Recursos com base em um caminho ou que publicam em *`TargetArn`* ou *`PhoneNumber`*

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sns:Publish"
            ],
            "Resource": "*"
        }
    ]
}
```

Políticas do IAM para Amazon SQS

Os modelos de exemplo a seguir mostram como AWS Step Functions gera políticas do IAM com base nos recursos na definição da sua máquina de estado. Para obter mais informações, consulte Políticas do IAM para serviços integrados e Padrões de integração de serviço.

Recursos estáticos

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sqs:SendMessage"
            ],
            "Resource": [
                "arn:aws:sqs:[[region]]:[[accountId]]:[[queueName]]"
            ]
        }
    ]
```

```
}
```

Recursos dinâmicos

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sqs:SendMessage"
            ],
            "Resource": "*"
        }
    ]
}
```

Políticas do IAM para AWS Step Functions

Para uma máquina de estado que chama `StartExecution` para uma única execução de fluxo de trabalho aninhado, use uma política do IAM que limita as permissões a essa máquina de estado.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "states:StartExecution"
            ],
            "Resource": [

 "arn:aws:states:[[region]]:[[accountId]]:stateMachine:[[stateMachineName]]"
            ]
        }
    ]
}
```

Para obter mais informações, consulte as informações a seguir.

- Usando AWS Step Functions com outros serviços

- Transmitir parâmetros para uma API de serviço
- Gerencie AWS Step Functions execuções como um serviço 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 obter mais informações sobre execuções de fluxos de trabalho aninhados, consulte: Iniciar execuções de fluxo de trabalho usando um estado Tarefa.

Políticas do IAM para AWS X-Ray

Os modelos de exemplo a seguir mostram como AWS Step Functions gera políticas do IAM com base nos recursos na definição da sua máquina de estado. Para obter mais informações, consulte Políticas do IAM para serviços integrados e Padrões de integração de serviço.

Para habilitar o rastreamento do X-Ray, você precisará de uma política do IAM com permissões adequadas. Se a máquina de estado usa outros serviços integrados, podem ser necessárias políticas adicionais do IAM. Veja as políticas do IAM para suas integrações de serviços específicas.

Ao criar uma máquina de estado com o rastreamento do X-Ray ativado, uma política do IAM é criada automaticamente.

> **Note**
>
> Para habilitar o rastreamento do X-Ray para uma máquina de estado existente, será necessário garantir a adição de uma política com permissões suficientes.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "xray:PutTraceSegments",
                "xray:PutTelemetryRecords",
                "xray:GetSamplingRules",
                "xray:GetSamplingTargets"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Para ver mais informações sobre o uso do X-Ray com o Step Functions, consulte AWS X-Ray e Step Functions.

Tarefas de atividade ou nenhuma tarefa

Para obter uma máquina de estado que tem apenas tarefas de `Activity` ou nenhuma tarefa, use uma política do IAM que nega o acesso a todos os recursos e ações.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "*",
            "Resource": "*"
        }
    ]
```

```
}
```

Para obter mais informações sobre como usar tarefas `Activity`, consulte Atividades.

Políticas do IAM para usar o estado Mapa Distribuído

Ao criar fluxos de trabalho com o console do Step Functions, o Step Functions pode gerar automaticamente políticas do IAM com base nos recursos na definição de fluxo de trabalho. Essas políticas incluem os privilégios mínimos necessários para permitir que o perfil da máquina de estado invoque a ação da API `StartExecution` para o estado Mapa Distribuído. Essas políticas também incluem os privilégios mínimos necessários: Step Functions para acessar AWS recursos, como buckets e objetos do Amazon S3 e funções Lambda. É altamente recomendável que você inclua apenas as permissões que forem necessárias em suas políticas do IAM. Por exemplo, se o fluxo de trabalho incluir um estado `Map` no modo distribuído, defina o escopo de suas políticas até o bucket e a pasta específicos do Amazon S3 que contêm o conjunto de dados.

> **Important**
>
> Se você especificar um bucket e um objeto do Amazon S3, ou prefixo, com um caminho de referência para um par de valores-chave existente na entrada do estado Mapa Distribuído, certifique-se de atualizar as políticas de IAM do fluxo de trabalho. Defina o escopo das políticas até o bucket e os nomes de objetos para os quais o caminho é resolvido em runtime.

Neste tópico:

Exemplo de política do IAM para executar um estado Mapa Distribuído

Ao incluir um estado Mapa Distribuído nos fluxos de trabalho, o Step Functions precisa de permissões apropriadas para permitir que o perfil de máquina de estado invoque a ação da API `StartExecution` para o estado Mapa Distribuído.

O exemplo de política do IAM a seguir concede os privilégios mínimos necessários ao perfil da máquina de estado para executar o estado Mapa Distribuído.

> **Note**
>
> Substitua *stateMachineName* pelo nome da máquina de estado na qual você está usando o estado Mapa Distribuído. Por exemplo, `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:*"
    }
  ]
}
```

Exemplo de política do IAM para redriving um estado Mapa Distribuído

Você pode reiniciar execuções malsucedidas do fluxo de trabalho secundário em uma Execução de mapa redriving o fluxo de trabalho principal. Um fluxo de trabalho principal redriven redrives todos os estados malsucedidos, incluindo o Mapa distribuído. Certifique-se de que o perfil de execução tenha os privilégios mínimos necessários para permitir que ele invoque a ação da API `RedriveExecution` no fluxo de trabalho principal.

O exemplo de política do IAM a seguir concede os privilégios mínimos necessários ao perfil da máquina de estado para redriving um estado Mapa Distribuído.

> **Note**
>
> Substitua *stateMachineName* pelo nome da máquina de estado na qual você está usando o estado Mapa Distribuído. Por exemplo, `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:*"
    }
  ]
}
```

Exemplos de políticas do IAM para ler dados de conjuntos de dados do Amazon S3

Os exemplos de políticas do IAM a seguir concedem os privilégios mínimos necessários para acessar seus conjuntos de dados do Amazon S3 usando ListObjectsas ações de GetObjectV2 e API.

Example Política do IAM para objeto do Amazon S3 como conjunto de dados

O exemplo a seguir mostra uma política do IAM que concede os privilégios mínimos para acessar os objetos organizados em *processImages* em um bucket do Amazon S3 chamado *myBucket*.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
```

```
            ],
            "Resource": [
                "arn:aws:s3:::myBucket"
            ],
            "Condition": {
                "StringLike": {
                    "s3:prefix": [
                        "processImages"
                    ]
                }
            }
        }
    ]
}
```

Example Política do IAM para um arquivo CSV como conjunto de dados

O exemplo a seguir mostra uma política do IAM que concede os privilégios mínimos para acessar um arquivo CSV chamado *ratings.csv*.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::myBucket/csvDataset/ratings.csv"
            ]
        }
    ]
}
```

Example Política do IAM para um inventário Amazon S3 como conjunto de dados

O exemplo a seguir mostra uma política do IAM que concede os privilégios mínimos para acessar um relatório de inventário 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/*"
            ]
        }
    ]
}
```

Exemplo de política do IAM para gravar dados em um bucket do Amazon S3

O exemplo de política do IAM a seguir concede os privilégios mínimos necessários para gravar os resultados da execução do fluxo de trabalho secundário em uma pasta chamada *csvJobs* em um bucket do Amazon S3 usando a ação de API PutObject.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListMultipartUploadParts",
                "s3:AbortMultipartUpload"
            ],
            "Resource": [
                "arn:aws:s3:::resultBucket/csvJobs/*"
            ]
        }
    ]
}
```

Permissões do IAM para AWS KMS key bucket criptografado do Amazon S3

O estado Mapa Distribuído usa uploads de várias partes para gravar os resultados da execução do fluxo de trabalho secundário em um bucket do Amazon S3. Se o bucket for criptografado

usando uma chave AWS Key Management Service (AWS KMS), será necessário também incluir permissões em sua política do IAM para realizar as ações `kms:Decrypt`, `kms:Encrypt` e `kms:GenerateDataKey` na chave. Essas permissões são necessárias porque o Amazon S3 precisa descriptografar e ler os dados de partes de arquivos criptografados antes de concluir o multipart upload.

O exemplo de política do IAM a seguir concede permissão para as ações `kms:Decrypt`, `kms:Encrypt` e `kms:GenerateDataKey` na chave usada para criptografar o bucket do 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 obter mais informações, consulte Carregando um arquivo grande para o Amazon S3 com criptografia usando uma AWS KMS key na Central de Conhecimento da AWS .

Se seu usuário ou função do IAM for Conta da AWS igual aoKMS key, você deverá ter essas permissões na política de chaves. Se o seu usuário ou perfil do IAM pertencer a uma conta diferente da KMS key, será necessário ter as permissões na política de chave e no usuário ou perfil do IAM.

Políticas baseadas em tag

O Step Functions oferece suporte a políticas baseadas em tags. Por exemplo, é possível restringir o acesso a todos os recursos do Step Functions que incluam uma tag com a chave `environment` e o 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 aplica `Deny` à capacidade de excluir máquinas de estado ou atividades, interromper execuções e adicionar ou excluir novas tags para todos os recursos que foram marcados como `environment/production`.

Para autorização baseada em tags, os recursos de execução da máquina de estado, conforme mostrado no exemplo a seguir, herdam as tags associadas a uma máquina de estado.

```
arn:<partition>:states:<Region>:<account-id>:execution:<StateMachineName>:<ExecutionId>
```

Quando você chama DescribeExecutionou outras APIs nas quais você especifica o ARN do recurso de execução, o Step Functions usa tags associadas à máquina de estado para aceitar ou negar a solicitação enquanto realiza a autorização baseada em tags. Isso ajuda você a permitir ou negar acesso às execuções no nível da máquina de estado.

Para obter mais informações sobre a marcação, consulte o seguinte:

- Marcação no Step Functions
- Controlar o acesso com tags do IAM

Solução de problemas AWS Step Functions de identidade e acesso

Use as seguintes informações para ajudar a diagnosticar e corrigir problemas comuns que podem ser encontrados ao trabalhar com o Step Functions e o IAM.

Tópicos

Não tenho autorização para executar uma ação no Step Functions

Se você receber uma mensagem de erro informando que não tem autorização para executar uma ação, é preciso atualizar suas políticas para permitir que você realize a ação.

O erro do exemplo a seguir ocorre quando o usuário `mateojackson` tenta usar o console para visualizar detalhes sobre um recurso do *`my-example-widget`* fictício, mas não tem as permissões fictícias do `states:`*`GetWidget`*.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 states:GetWidget on resource: my-example-widget
```

Nesse caso, a política de Mateo deve ser atualizada para permitir que ele tenha acesso ao recurso *`my-example-widget`* usando a ação `states:`*`GetWidget`*.

Se precisar de ajuda, entre em contato com seu AWS administrador. Seu administrador é a pessoa que forneceu suas credenciais de login.

Não estou autorizado a realizar iam: PassRole

Se você receber uma mensagem de erro informando que não tem autorização para executar a ação `iam:PassRole`, as suas políticas deverão ser atualizadas para permitir a passagem de um perfil para o Step Functions.

Alguns Serviços da AWS permitem que você passe uma função existente para esse serviço em vez de criar uma nova função de serviço ou uma função vinculada ao serviço. Para fazer isso, é preciso ter permissões para passar o perfil para o serviço.

O exemplo de erro a seguir ocorre quando um usuário do IAM chamado `marymajor` tenta usar o console para executar uma ação no Step Functions. No entanto, a ação exige que o serviço tenha permissões concedidas por um perfil de serviço. Mary não tem permissões para passar o perfil para o serviço.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Nesse caso, as políticas de Mary devem ser atualizadas para permitir que ela realize a ação `iam:PassRole`.

Se precisar de ajuda, entre em contato com seu AWS administrador. Seu administrador é a pessoa que forneceu suas credenciais de login.

Quero permitir que pessoas fora da minha acessem meus Conta da AWS recursos do Step Functions

Você pode criar um perfil que os usuários de outras contas ou pessoas fora da sua organização podem usar para acessar seus recursos. Você pode especificar quem é confiável para assumir o perfil. Para serviços que oferecem compatibilidade com políticas baseadas em recursos ou listas de controle de acesso (ACLs), você pode usar essas políticas para conceder às pessoas acesso aos seus recursos.

Para saber mais, consulte:

- Para saber se o Step Functions é compatível com esses recursos, consulte Como AWS Step Functions funciona com o IAM.
- Para saber como fornecer acesso aos seus recursos em todos os Contas da AWS que você possui, consulte Como fornecer acesso a um usuário do IAM em outro Conta da AWS que você possui no Guia do usuário do IAM.
- Para saber como fornecer acesso aos seus recursos a terceiros Contas da AWS, consulte Como fornecer acesso Contas da AWS a terceiros no Guia do usuário do IAM.
- Para saber como conceder acesso por meio da federação de identidades, consulte Conceder acesso a usuários autenticados externamente (federação de identidades) no Guia do usuário do IAM.
- Para saber a diferença entre usar funções e políticas baseadas em recursos para acesso entre contas, consulte Acesso a recursos entre contas no IAM no Guia do usuário do IAM.

Registro e Monitoramento

Para obter informações sobre login e monitoramento AWS Step Functions, consulte a Registrar e monitorar seção.

Validação de conformidade para AWS Step Functions

Auditores terceirizados avaliam a segurança e a conformidade AWS Step Functions como parte de vários programas de AWS conformidade. Isso inclui SOC, PCI, FedRAMP, HIPAA e outros.

Para obter uma lista de AWS serviços no escopo de programas de conformidade específicos, consulte AWS Serviços no escopo do programa de conformidade AWS . Para obter informações gerais, consulte Programas de AWS conformidade Programas AWS de .

Você pode baixar relatórios de auditoria de terceiros usando AWS Artifact. Para obter mais informações, consulte Baixar relatórios em AWS Artifact .

Sua responsabilidade com relação à compatibilidade ao usar o Step Functions é determinada pela confidencialidade dos dados, pelos objetivos de conformidade da empresa e pelos regulamentos e leis aplicáveis. A AWS oferece os seguintes recursos para ajudar na conformidade:

- Guias de início rápido sobre sobre segurança e conformidade — Esses guias de implantação discutem considerações arquitetônicas e fornecem etapas para a implantação de ambientes básicos com foco em segurança e conformidade em. AWS
- Arquitetura para segurança e conformidade com a HIPAA na Amazon Web Services — Este whitepaper descreve como as empresas podem usar AWS para criar aplicativos compatíveis com a HIPAA.
- AWS Recursos de https://aws.amazon.com/compliance/resources/ de conformidade — Essa coleção de pastas de trabalho e guias pode ser aplicada ao seu setor e local.
- Avaliação de recursos com regras no Guia do AWS Config desenvolvedor — O AWS Config serviço avalia o quão bem suas configurações de recursos estão em conformidade com as práticas internas, as diretrizes e os regulamentos do setor.
- AWS Security Hub— Esse AWS serviço fornece uma visão abrangente do seu estado de segurança interno, AWS que ajuda você a verificar sua conformidade com os padrões e as melhores práticas do setor de segurança.

Resiliência em AWS Step Functions

A infraestrutura AWS global é construída em torno de AWS regiões e zonas de disponibilidade. AWS As regiões fornecem várias zonas de disponibilidade fisicamente separadas e isoladas, conectadas a redes de baixa latência, alta taxa de transferência e alta redundância. Com as zonas

de disponibilidade, é possível projetar e operar aplicativos e bancos de dados que automaticamente executam o failover entre as zonas sem interrupção. As zonas de disponibilidade são altamente disponíveis, tolerantes a falhas e escaláveis que uma ou várias infraestruturas de datacenters tradicionais.

Para obter mais informações sobre AWS regiões e zonas de disponibilidade, consulte Infraestrutura AWS global.

Além da infraestrutura AWS global, o Step Functions oferece vários recursos para ajudar a suportar suas necessidades de resiliência e backup de dados.

Segurança de infraestrutura em AWS Step Functions

Como serviço gerenciado, AWS Step Functions é protegido pela segurança de rede AWS global. Para obter informações sobre serviços AWS de segurança e como AWS proteger a infraestrutura, consulte AWS Cloud Security. Para projetar seu AWS ambiente usando as melhores práticas de segurança de infraestrutura, consulte Proteção de infraestrutura no Security Pillar AWS Well-Architected Framework.

Você usa chamadas de API AWS publicadas para acessar Step Functions pela rede. Os clientes devem oferecer suporte para:

- Transport Layer Security (TLS). Exigimos TLS 1.2 e recomendamos TLS 1.3.
- Conjuntos de criptografia com perfect forward secrecy (PFS) como DHE (Ephemeral Diffie-Hellman) ou ECDHE (Ephemeral Elliptic Curve Diffie-Hellman). A maioria dos sistemas modernos, como Java 7 e versões posteriores, comporta esses modos.

Além disso, as solicitações devem ser assinadas usando um ID da chave de acesso e uma chave de acesso secreta associada a uma entidade principal do IAM. Ou você pode usar o AWS Security Token Service (AWS STS) para gerar credenciais de segurança temporárias para assinar solicitações.

Você pode chamar as operações da AWS API de qualquer local da rede, mas Step Functions não oferece suporte a políticas de acesso baseadas em recursos, que podem incluir restrições com base no endereço IP de origem. Também é possível usar políticas do Step Functions para controlar o acesso de endpoints de Amazon Virtual Private Cloud (Amazon VPC) ou VPCs específicas. Efetivamente, isso isola o acesso à rede a um determinado Step Functions recurso somente da VPC específica dentro da AWS rede.

Análise de configuração e vulnerabilidade em AWS Step Functions

A configuração e os controles de TI são uma responsabilidade compartilhada entre você AWS e você, nosso cliente. Para obter mais informações, consulte o modelo de responsabilidade AWS compartilhada.

Migração de cargas de trabalho do Step Functions AWS Data Pipeline para o Step Functions

AWS lançou o AWS Data Pipeline serviço em 2012. Naquela época, os clientes procuravam um serviço que os permitisse mover dados entre diferentes fontes de dados usando uma variedade de opções de computação. À medida que as necessidades de transferência de dados mudaram com o tempo, as soluções para essas necessidades também mudaram. Agora você tem a opção de escolher a solução que melhor atenda às suas necessidades comerciais. Por exemplo, você pode fazer o seguinte:

- Use Step Functions para orquestrar fluxos de trabalho entre vários Serviços da AWS.
- Use o Amazon Managed Workflows for Apache Airflow (Amazon MWAA) para gerenciar a orquestração do fluxo de trabalho do Apache Airflow.
- Use AWS Glue para executar e orquestrar aplicativos Apache Spark.

Você pode migrar casos de uso típicos AWS Data Pipeline para Step Functions ou Amazon MWAA. AWS Glue A opção escolhida depende de sua workload atual em AWS Data Pipeline. Este tópico explica como migrar do Step Functions AWS Data Pipeline para o Step Functions.

Tópicos

Migração de workloads do AWS Data Pipeline

O Step Functions é um serviço de orquestração sem servidor que permite criar fluxos de trabalho para suas aplicações essenciais aos negócios. Com o Workflow Studio do Step Functions, você pode criar fluxos de trabalho e integrá-los com mais de 11.000 ações de API de mais de 250 Serviços da AWS. Isso inclui Serviços da AWS Amazon EMR e Amazon DynamoDB. AWS Lambda Você pode usar o Step Functions para orquestrar pipelines de processamento de dados, lidar com erros e trabalhar com os limites de controle de utilização nos serviços subjacentes dos Serviços da AWS. Você pode criar fluxos de trabalho que processam e publicam modelos de machine

learning, orquestram microsserviços e controlam fluxos de trabalho de extração, transformação e carregamento (ETL) com o AWS Glue. Você também pode criar fluxos de trabalho automatizados e de longa duração para aplicativos que exigem interação humana.

O Step Functions é um serviço totalmente gerenciado fornecido pela AWS. Isso significa que a AWS gerencia tarefas como manter a infraestrutura, aplicar patches aos funcionários e gerenciar atualizações da versão do sistema operacional para você.

Quando seu caso de uso corresponder às seguintes condições, recomendamos que você migre do AWS Data Pipeline Step Functions:

- Você prefere um serviço de orquestração de fluxo de trabalho sem servidor e altamente disponível.
- Você precisa de uma solução que carregue com a granularidade da execução de uma única tarefa.
- Suas cargas de trabalho envolvem a orquestração de tarefas para várias outras Serviços da AWS, como Amazon EMR, Lambda ou DynamoDB. AWS Glue
- Você precisa de uma solução low-code com um designer drag-and-drop visual para criar fluxos de trabalho. Essa solução não deve exigir o aprendizado de conceitos de programação complexos e desconhecidos.
- Você precisa de um serviço que se integre com mais de 250 Serviços da AWS que cubram mais de 11.000 ações de API. Esse serviço também deve se integrar a serviços e atividades personalizados fora do AWS.

Mapeamento de conceitos entre o Step Functions e o AWS Data Pipeline

AWS Data Pipeline e Step Functions compartilham alguns conceitos comuns. Por exemplo, para definir seus fluxos de trabalho, você usa o formato JSON em ambos AWS Data Pipeline e em Step Functions. No Step Functions, você usa Amazon States Language, que é uma linguagem estruturada baseada em JSON. Você usa a Amazon States Language (ASL) para definir seus fluxos de trabalho e alternar entre as representações textuais e visuais do seu fluxo de trabalho. Esse formato baseado em JSON ajuda a simplificar o armazenamento de seus fluxos de trabalho em uma ferramenta de controle de origem. Também ajuda você a gerenciar várias versões de seus fluxos de trabalho, controlar seu acesso ou automatizar sua orquestração com métodos de CI/CD.

A tabela a seguir descreve o mapeamento entre os principais conceitos usados nos dois serviços. A coluna de conceitos do pipeline de dados à esquerda lista os conceitos em AWS Data Pipeline,

enquanto a coluna de conceitos do Step Functions à direita lista os conceitos equivalentes em Step Functions.

Conceitos do pipeline de dados	Conceitos do Step Functions
Pipelines	Fluxos de trabalho
Definição de pipeline	Amazon States Language (ASL)
Atividades	States e Estado da tarefa
Instâncias	Execuções
Attempts	Catchers e retriers
Cronograma do pipeline	• Execuções com o Amazon Scheduler EventBridge • Eventos acionados por meio de EventBridge Pipes
Expressões e funções de pipeline	• Funções intrínsecas • Funções do Lambda usando integração de serviços

Projetos de exemplo do Step Functions

Para obter uma introdução ao Step Functions, consulte o vídeo a seguir:

Introdução à AWS Step Functions orquestração de serviços

A lista a seguir contém alguns projetos de exemplos de implementações para os casos de uso mais comuns do AWS Data Pipeline com o Step Functions. Você pode usar esses projetos de amostra como referência para AWS Data Pipeline migrar do Step Functions. Você também pode usá-los como um padrão para criar seus próprios fluxos de trabalho e integrá-los aos Serviços da AWS compatíveiscom base em seu caso de uso.

- Gerenciamento de um Trabalho do Amazon EMR
- Executar uma tarefa de processamento de dados no Amazon EMR Sem Servidor

- Executar trabalhos do Hie/Pig/Hadoop
- Consulte grandes conjuntos de dados (Amazon Athena, Amazon S3, Amazon SNS AWS Glue)
- Executar fluxos de trabalho ETL/ELT usando o Amazon Redshift
- AWS Glue Orquestrando rastreadores
- Execute um script shell com o Step Functions

Para saber mais sobre o Step Functions, consulte os seguintes tópicos e recursos:

- Tutoriais do Step Functions
- Projetos de amostra para Step Functions
- O Workshop de AWS Step Functions

Comparação de preços

AWS Data Pipeline é precificado pelo número de tubulações e seu nível de uso. As atividades realizadas mais de uma vez por dia (alta frequência) custam $1 por mês por atividade. As atividades realizadas uma vez por dia ou menos (baixa frequência) custam $0,60 por mês por atividade. Os Pipelines Inativos custam $1 por pipeline. Para obter mais informações sobre a definição de preço, consulte a página de Definição de preço do AWS Data Pipeline.

O Step Functions tem dois tipos de fluxos de trabalho: padrão e expressos. Cada tipo de fluxo de trabalho tem um modelo de definição de preços diferente. Essa comparação é baseada no fluxo de trabalho padrão, pois é a que melhor corresponde aos casos de uso comuns de AWS Data Pipeline. Os fluxos de trabalho padrão custam US$ 0,025 por 1.000 transições de estado. Não há custo para máquinas de estado inativas. Você paga somente pelo que usar. Para obter mais informações sobre a definição de preço, consulte a página de Definição de preço do AWS Step Functions.

Solução de problemas

Se você encontrar dificuldades ao trabalhar com o Step Functions, use os seguintes recursos para solução de problemas.

Tópicos

Solução de problemas gerais

Não consigo criar uma máquina de estado.

O perfil do IAM associado à máquina de estado pode não ter permissões suficientes. Verifique as permissões do perfil do IAM, inclusive para tarefas de integração de serviços do AWS, X-Ray e registro do CloudWatch. São necessárias permissões adicionais para os estados da tarefa `.sync`.

Não consigo usar um JsonPath para referenciar a saída da tarefa anterior.

Para um JsonPath, uma chave JSON deve terminar com `.$`. Isso significa que um JsonPath só pode ser usado em um par de chave-valor. Se quiser usar um JsonPath em outros lugares, como uma matriz, você pode usar funções intrínsecas. Por exemplo, você pode usar algo semelhante a:

Saída da tarefa A:

```
{
    "sample": "test"
}
```

Tarefa B:

```
{
```

```
    "JsonPathSample.$": "$.sample"
}
```

> **Tip**
>
> Use o simulador de fluxo de dados no console do Step Functions para testar a sintaxe do caminho JSON, entender melhor como os dados são manipulados em um estado e ver como os dados são transmitidos entre os estados.

Houve um atraso nas transições de estado.

Para fluxos de trabalho padrão, há um limite no número de transições de estado. Quando você excede o limite de transição de estado, o Step Functions atrasa as transições de estado até que o bucket da cota seja preenchido. O controle de utilização do limite de transição de estado pode ser monitorado analisando a métrica `ExecutionThrottled` na seção Métricas de execução da página CloudWatch Metrics.

Quando eu inicio novas execuções do Fluxo de trabalho padrão, elas falham com o erro **`ExecutionLimitExceeded`**.

O Step Functions tem um limite de 1.000.000 de execuções abertas para cada Conta da AWS em cada Região da AWS. Se você exceder esse limite, o Step Functions gerará um erro `ExecutionLimitExceeded`. Não se aplica ao fluxos de trabalho expressos. Você pode usar a seguinte matemática do CloudWatch Metrics no Guia do usuário do Amazon CloudWatch para aproximar o número de execuções abertas: `ExecutionsStarted - (ExecutionsSucceeded + ExecutionsTimedOut + ExecutionsFailed + ExecutionsAborted)`.

Uma falha em uma ramificação em um estado paralelo faz com que toda a execução falhe.

Esse é um comportamento esperado. Para evitar falhas ao usar um estado paralelo, configure o Step Functions para capturar erros gerados por cada ramificação.

Solução de problemas de integrações de serviço

Meu trabalho foi concluído no serviço downstream, mas, no Step Functions, o estado da tarefa permanece “Em andamento” ou sua conclusão está atrasada.

Para padrões de integração de serviços `.sync`, o Step Functions usa regras do EventBridge, APIs downstream ou uma combinação de ambos para detectar o status do trabalho downstream. Para alguns serviços, o Step Functions não cria regras do EventBridge para monitorar. Por exemplo, para a integração de serviço AWS Glue, em vez de usar as regras do EventBridge, o Step Functions faz uma chamada `glue:GetJobRun`. Dada a frequência das chamadas de API, há uma diferença entre a conclusão da tarefa downstream e o tempo de conclusão da tarefa do Step Functions. O Step Functions exige permissões do IAM para gerenciar as regras do EventBridge e fazer chamadas para o serviço downstream. Para obter mais detalhes sobre como permissões insuficientes em sua função de execução podem afetar a conclusão de tarefas, consulte Permissões adicionais para tarefas usando o padrão Executar um trabalho.

Quero retornar uma saída JSON de uma execução de máquina de estado aninhada.

Há duas integrações de serviços síncronos do Step Functions para o Step Functions: `startExecution.sync` e `startExecution.sync:2`. Ambas aguardam a conclusão da máquina de estado aninhado, mas retornam formatos diferentes de `Output`. Você pode usar `startExecution.sync:2` para retornar uma saída JSON em `Output`.

Não consigo invocar uma função do Lambda de outra conta.

Como acessar a função do Lambda com suporte entre contas

Se o acesso entre contas dos recursos AWS estiver disponível em sua região, use o método a seguir para invocar uma função do Lambda de outra conta.

Para invocar um recurso entre contas em seus fluxos de trabalho, faça o seguinte:

1. Crie um perfil do IAM na conta de destino que contém o recurso. Esse perfil concede permissões à conta de origem que contém a máquina de estado para acessar os recursos da conta de destino.

2. Na definição do estado da `Task`, especifique o perfil do IAM de destino a ser assumido pela máquina de estado antes de invocar o recurso entre contas.
3. Modifique a política de confiança no perfil do IAM de destino para permitir que a conta de origem assuma esse perfil temporariamente. A política de confiança deve incluir o nome do recurso da Amazon (ARN) da máquina de estado definida na conta de origem. Além disso, defina as permissões apropriadas no perfil do IAM de destino para chamar o recurso da AWS.
4. Atualize o perfil de execução da conta de origem para incluir a permissão necessária para assumir o perfil do IAM de destino.

Para ver um exemplo, consulte Tutorial: Acessando recursos entre contas AWS.

> **Note**
>
> Você pode configurar sua máquina de estado para assumir um perfil do IAM para acessar recursos de várias Contas da AWS. No entanto, uma máquina de estado pode assumir somente um perfil do IAM por vez.

Para obter um exemplo de uma definição de estado `Task` que especifica um recurso entre contas, consulte Exemplos do campo Credenciais do estado Tarefa.

Como acessar a função do Lambda sem suporte entre contas

Se o acesso entre contas dos recursos AWS não estiver disponível em sua região, use o método a seguir para invocar uma função do Lambda de outra conta.

No campo `Resource` do estado `Task`, use `arn:aws:states:::lambda:invoke` e passe `FunctionArn` nos parâmetros. O perfil do IAM associado à máquina de estado deve ter as permissões corretas para invocar funções do Lambda entre contas: `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
      }
   }
}
```

Não consigo ver os tokens de tarefas transmitidos dos estados `.waitForTaskToken`.

No `Parameters` campo do `Task` estado, você deve passar um token de tarefa. Por exemplo, você pode usar algo semelhante ao código a seguir.

```
{
   "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**
>
> Você pode tentar usar `.waitForTaskToken` com qualquer ação de API. Porém, algumas APIs não têm parâmetros adequados.

Atividades de solução de problemas

A execução da minha máquina de estado está emperrada em um estado de atividade.

O estado de uma tarefa de atividade não começa até você sondar um token de tarefa usando a ação da API GetActivityTask. Como prática recomendada, adicione um tempo limite no nível da tarefa para evitar uma execução emperrada. Para obter mais informações, consulte Usar tempos limite para evitar travar execuções.

Se sua máquina de estado estiver emperrada no evento ActivityScheduled, isso indica que sua frota de trabalhadores ativos tem problemas ou está subdimensionada. Você deve monitorar a métrica da métrica `ActivityScheduleTime` CloudWatch e definir um alarme quando esse tempo aumentar. No entanto, para expirar qualquer execução de máquina de estado emperrada em que o estado `Activity` não faça a transição para o estado `ActivityStarted`, defina um tempo limite no nível da máquina de estado. Para fazer isso, especifique um campo `TimeoutSeconds` no início da definição da máquina de estado, fora do campo `States`.

Meu trabalhador da atividade atinge o tempo limite enquanto aguarda por um token de tarefa.

Os trabalhadores usam a ação de API GetActivityTask para recuperar uma tarefa com o ARN de atividade especificada que está programada para execução por uma máquina de estado em execução. `GetActivityTask` inicia uma longa sondagem para que o serviço mantenha a conexão HTTP aberta e responda assim que uma tarefa estiver disponível. O tempo máximo em que o serviço retém a solicitação antes de responder é de 60 segundos. Se nenhuma tarefa estiver disponível em 60 segundos, a pesquisa retornará um `taskToken` com uma string nula. Para evitar esse tempo limite, configure um soquete do lado do cliente com um tempo limite de pelo menos 65 segundos no AWS SDK ou no cliente que você está usando para fazer a chamada da API.

Solucionar problemas nos fluxos de trabalho expresso

Meu aplicativo expira antes de receber uma resposta de uma chamada de API **`StartSyncExecution`**.

Configure o tempo limite do soquete do lado do cliente no AWS SDK ou no cliente que você usa para fazer a chamada de API. Para receber uma resposta, o tempo limite deve ter um valor maior do que a duração das execuções do fluxos de trabalho expresso.

Não consigo ver o histórico de execução para solucionar problemas do fluxos de trabalho expresso.

O fluxos de trabalho expressos não registra o histórico de execução no AWS Step Functions. Em vez disso, você deve ativar o registro do CloudWatch. Depois que o registro estiver ativado, você poderá usar as consultas do CloudWatch Logs Insights para revisar suas execuções do fluxos de trabalho expresso. Você também pode visualizar o histórico de execução das execuções do fluxos de trabalho expresso no console Step Functions se escolher o botão Ativar na guia Execuções. Para obter mais informações, consulte Visualizar e depurar execuções no console do Step Functions.

Para listar as execuções com base na duração:

```
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 listar as execuções que falharam e foram canceladas:

```
fields ispresent(execution_arn) as isRes | filter type in ["ExecutionFailed",
 "ExecutionAborted", "ExecutionTimedOut"]
```

Informações relacionadas

A tabela a seguir lista os recursos relacionados que serão úteis à medida que você utilizar este serviço.

Recurso	Descrição
AWS Step Functions Referência da API	Descrições de ações de API, parâmetros e tipos de dados, além de uma lista de erros que o serviço retorna.
Referência da linha de comando do AWS Step Functions	Descrições dos comandos da AWS CLI que você pode usar para trabalhar com o AWS Step Functions.
Informações sobre o produto para o Step Functions	A principal página da Web para obter informaçõ es sobre o Step Functions.
Fóruns de discussão	Um fórum baseado na comunidade para os desenvolvedores debaterem questões técnicas relacionadas ao Step Functions e outros serviços da AWS.
Informações do AWS Support	A principal página da Web para obter informaçõ es sobre o AWS Support, um canal de suporte de resposta rápida e com atendimento individual, para ajudar você a desenvolver e executar aplicações nos serviços de infraestr utura da AWS.

Lançamentos de atributos recentes

A tabela a seguir lista as Regiões nas quais os novos atributos do Step Functions estão disponíveis.

Data de lançamento	Nome do atributo	Regiões disponíveis
26 de novembro de 2023	Invocar endpoints HTTPS públicos e testar estados individuais	• Leste dos EUA (Norte da Virgínia), us-east-1 • Oeste dos EUA (Oregon), us-west-2 • Leste dos EUA (Ohio), us-east-2 • Europa (Irlanda), eu-west-1 • Europa (Frankfurt), eu-central-1 • Europa (Estocolmo), eu-north-1 • Ásia-Pacífico (Sydney), ap-southeast-2 • Ásia-Pacífico (Tóquio), ap-northeast-1 • Ásia-Pacífico (Singapura), ap-southeast-1
15 de novembro de 2023	Execuções do Redrive	Para obter uma lista completa das Regiões da AWS em que esse atributo está disponíve l, consulte as opções na lista suspensa Região na página intitulada Serviços da AWS por Região.
12 de outubro de 2023	Integrações otimizadas para Amazon EMR Serverless	Para obter uma lista completa das Regiões da AWS em que esse atributo está disponíve

Data de lançamento	Nome do atributo	Regiões disponíveis
		l, consulte as opções na lista suspensa Região na página intitulada Serviços da AWS por Região.
7 de setembro de 2023	Tratamento aprimorado de erros	Para obter uma lista completa das Regiões da AWS em que esse atributo está disponíve l, consulte as opções na lista suspensa Região na página intitulada Serviços da AWS por Região.
31 de agosto de 2023	Aprimoramentos do Workflow Studio para uma experiência de criação simplificada	Para obter uma lista completa das Regiões da AWS em que esse atributo está disponíve l, consulte as opções na lista suspensa Região na página intitulada Serviços da AWS por Região.
22 de junho de 2023	Versões e aliases	Para obter uma lista completa das Regiões da AWS em que esse atributo está disponíve l, consulte as opções na lista suspensa Região na página intitulada Serviços da AWS por Região.
16 de junho de 2023	Novas integrações SDK do AWS	Para obter uma lista completa das Regiões da AWS em que esse atributo está disponíve l, consulte as opções na lista suspensa Região na página intitulada Serviços da AWS por Região.

Data de lançamento	Nome do atributo	Regiões disponíveis
1.º de dezembro de 2022	Orquestre fluxos de trabalho paralelos em grande escala para o processamento de dados com o estado Mapa distribuído	Para obter uma lista completa das Regiões da AWS em que esse atributo está disponível, consulte as opções na lista suspensa Região na página intitulada Serviços da AWS por Região.

Histórico do documento

Esta seção lista as principais alterações no Guia do desenvolvedor do AWS Step Functions .

Alteração	Descrição	Alterado em
Atualizações	AWS atualizações de políticas gerenciadas - nova permissão: `states:ValidateStateMachineDefinition` Foram adicionadas informações sobre a nova permissão para verificar a sintaxe de uma máquina de estado fornecida por você. Para saber mais, consulte AWS políticas gerenciadas para AWS Step Functions.	29 de abril de 2024
Novo atributo	Step Functions adiciona integração otimizada para AWS Elemental MediaConvert AWS Elemental MediaConvert fornece transcodificação de arquivos de áudio e vídeo em nível de transmissão, que os clientes podem automatizar com código de acordo com seus fluxos de trabalho de mídia. Com a integração otimizada do AWS Step Functions in MediaConvert, agora é possível orquestrar usando a ferramenta visual de baixo código Workflow Studio. Para saber mais, consulte a documentação de Manage AWS Elemental MediaConvert with Step Functions.	12 de abril de 2024
Atualizações	AWS atualizações de políticas gerenciadas - Atualização de uma política existente: `AWSStepFunctionsReadOnlyAccess` Foram adicionadas informações sobre novas permissões somente de leitura para tags, mapas distribuídos, versões e aliases. Para saber mais, consulte AWS políticas gerenciadas para AWS Step Functions.	02 de abril de 2024

Alteração	Descrição	Alterado em
Atualizações	Step Functions adiciona suporte para métricas do Open Workflow Com as métricas de fluxo de trabalho abertas, agora você tem visibilidade em nível de conta do número de fluxos de trabalho padrão em andamento, bem como do seu limite de fluxo de trabalho aberto. Você pode gerenciar cargas de trabalho em todos os fluxos de trabalho, independentemente de como eles foram iniciados, para garantir operações de fluxo de trabalho sem problemas. Você pode definir CloudWatch alarmes para monitorar seus fluxos de trabalho e receber alertas proativamente à medida que se aproxima de seus limites. Depois de alertado, você pode gerenciar com eficácia seus fluxos de trabalho tomando medidas como interromper fluxos de trabalho específicos ou solicitar um aumento de limite. Métricas de fluxo de trabalho abertas estão disponíveis CloudWatch para uso em fluxos de trabalho padrão sem necessidade de configuração adicional. Para saber mais, consulte Métricas de execução.	29 de fevereiro de 2024
Atualizações	Adições e atualizações de integração de serviços. Para ver a lista de integrações de AWS SDK novas e atualizadas, consulte. Registro de alterações para integrações de AWS SDK compatíveis Para ver a lista completa de serviços, consulteIntegrações de serviços AWS SDK compatíveis.	18 de janeiro de 2024
Novo atributo	Use o Workflow Studio no Application Composer para criar fluxos de trabalho sem servidor usando modelos do AWS CloudFormation. Para ter mais informações, consulte Usar o Workflow Studio no Application Composer.	27 de novembro de 2023

Alteração	Descrição	Alterado em
Novo atributo	O Step Functions agora permite invocar diretamente endpoints HTTPS públicos e testar estados individua is usando uma nova API Test State. Para obter mais informações, consulte: • Chamar APIs de terceiros • Usando a TestState API para testar um estado	26 de novembro de 2023
Novo recurso	O Step Functions agora se integra ao Amazon Bedrock. Para obter mais informações, consulte os tópicos a seguir. • Chamar o Amazon Bedrock com o Step Functions • Permissões do IAM para o Amazon Bedrock • Realizar o encadeamento de prompts de IA com o Amazon Bedrock • Usando AWS Step Functions com outros serviços	26 de novembro de 2023
Novo recurso	O Step Functions agora permite redrive execuções do fluxo de trabalho do tipo Padrão a partir do ponto de falha. Para obter mais informações, consulte Redriving execuções e Redriving execuções de mapa.	15 de novembro de 2023
Atualização somente da documentação	Publicou um novo tópico que explica como executar máquinas de estado em um cronograma usando o Amazon EventBridge Scheduler. Para ter mais informações, consulte Como usar o Agendador do Amazon EventBridge com o Amazon AWS Step Functions.	16 de outubro de 2023

Alteração	Descrição	Alterado em
Novo atributo	O Step Functions agora se integra ao Amazon EMR Serverless. Para obter mais informações, consulte os tópicos a seguir. • Chamar o Amazon EMR Serverless com o Step Functions • Executar uma tarefa do EMR Serverless • Integrações otimizadas para o Step Functions • Usando AWS Step Functions com outros serviços	12 de outubro de 2023
Atualização somente da documentação	Foram adicionadas informações sobre a execução de máquinas de estado em uma programação usando Amazon EventBridge Scheduler. Para ter mais informaçõ es, consulte Como usar o Agendador do EventBridge.	5 de outubro de 2023
Atualizar	Os tópicos de estado Mapa Distribuído foram reorganiz ados e atualizados para maior clareza, brevidade e estabelecimento de um mapa de jornada claro para novos usuários. Para ter mais informações, consulte Usar o estado Mapa no modo distribuído para orquestrar workloads paralelas em grande escala.	6 de outubro de 2023
Correções	Foram corrigidos exemplos de código em um tutorial para usar o AWS CDK v2. Para ter mais informações, consulte Criar uma máquina de estado do Lambda para o Step Functions usando o AWS CDK.	19 de setembro de 2023
Atualizar	Foram adicionadas informações sobre os recursos aprimorados de tratamento de erros que o Step Functions introduziu para identificar erros com clareza e implement ar novas tentativas com maior controle. Para obter mais informações, consulte Fail e Nova tentativa após um erro.	7 de setembro de 2023

Alteração	Descrição	Alterado em
Atualizar	O Step Functions adicionou aprimoramentos ao Workflow Studio para simplificar a experiência de criação do fluxo de trabalho. Para ter mais informações, consulte AWS Step Functions Studio de fluxo de trabalho.	31 de agosto de 2023
Atualização somente da documentação	Foram adicionadas informações sobre o dobro da contagem real de métricas relatada para a métrica `ExecutionsStarted` . Para ter mais informações, consulte Métricas que relatam uma contagem.	25 de julho de 2023
Atualização somente da documentação	O Step Functions adicionou dois novos projetos de amostra que demonstram os seguintes casos de uso comuns para o estado de Mapa Distribuído: • Como processar um arquivo CSV • Como processar dados em um bucket do Amazon S3	17 de julho de 2023
Atualização somente da documentação	Foi publicado um novo tópico sobre a implantação de máquinas de estado usando o Terraform. Para ter mais informações, consulte Implantação de máquinas de estado usando o Terraform.	5 de julho de 2023
Atualização somente da documentação	Os procedimentos a seguir foram atualizados para corresponder às alterações na EventBridge interface da Amazon. • Encaminhando um evento Step Functions para EventBridge • Iniciar a execução de uma máquina de estado em resposta a eventos do Amazon S3	26 de junho de 2023

Alteração	Descrição	Alterado em
Novo atributo	O Step Functions agora fornece a capacidade de criar várias versões e aliases de máquinas de estado para melhorar a resiliência ao implantar fluxos de trabalho de tecnologia sem servidor. Para ter mais informações, consulte Gerencie implantações contínuas de com versões e aliases.	22 de junho de 2023
Atualização somente da documentação	Foi melhorada a descrição dos campos `TimeoutSeconds` e `HeartbeatSeconds` para descrever como eles são diferentes uns dos outros. Para ter mais informações, consulte Campos do estado Tarefa.	22 de junho de 2023
Atualização somente da documentação	Foi publicada uma nova seção que descreve como nivelar uma matriz de matrizes normalmente retornadas como resultado dos estados Paralelo e Mapa. Para ter mais informações, consulte Nivelamento de uma matriz de matrizes.	20 de junho de 2023
Atualizar	O Step Functions expandiu o suporte para integrações de AWS SDK adicionando sete Serviços da AWS e 468 novas ações de API. Para obter mais informações, consulte Integrações de serviços AWS SDK compatíveis e Registro de alterações para integrações de AWS SDK compatíveis.	16 de junho de 2023
Atualização somente da documentação	Publicou um novo tópico que lista os Regiões da AWS recursos recém-lançados do Step Functions que estão disponíveis. Para ter mais informações, consulte Lançamentos de atributos recentes.	16 de junho de 2023
Atualização somente da documentação	Step Functions agora inclui uma seção sobre Notificações de Usuários da AWS, e AWS service (Serviço da AWS) que atua como um local central para suas AWS notificações no AWS Management Console. Para ter mais informações, consulte Usar o Notificações de Usuários da AWS com o Step Functions.	4 de maio de 2023

Alteração	Descrição	Alterado em
Atualização somente da documentação	Foi adicionada uma nova seção que explica sobre as permissões necessárias para gravar os resultados da execução do fluxo de trabalho filho em um bucket do Amazon S3 criptografado com uma chave AWS Key Management Service (AWS KMS). Para ter mais informaçõ es, consulte Permissões do IAM para AWS KMS key bucket criptografado do Amazon S3.	29 de abril de 2023
Atualização somente da documentação	Foi adicionado um novo tópico que explica sobre o recurso do Simulador de fluxo de dados. Para ter mais informaçõ es, consulte Simulador de fluxo de dados.	14 de abril de 2023
Atualização da cota	Foram adicionadas informações sobre a cota padrão de mil para execuções de mapas abertas em cada conta. Para ter mais informações, consulte Cotas relacionadas a contas.	5 de abril de 2023
Atualização somente da documentação	Foi adicionado um tópico que descreve quando migrar AWS Data Pipeline cargas de trabalho para o Step Functions. Este tópico também fornece uma lista de exemplos que explicam como realizar a migração. Para ter mais informações, consulte Migração de cargas de trabalho do Step Functions AWS Data Pipeline para o Step Functions.	30 de março de 2023
Atualização somente da documentação	Foi adicionada uma observação sobre a indisponibilidade do rastreamento do X-Ray para o estado Mapa Distribuí do. Para ter mais informações, consulte AWS X-Ray e Step Functions.	21 de março de 2023
Atualização somente da documentação	Foram adicionadas informações sobre como o Step Functions lida com autorização baseada em tags. Para obter mais informações, consulte Marcação no Step Functions e Políticas baseadas em tag.	15 de março de 2023

Alteração	Descrição	Alterado em
Atualização somente da documentação	Foram adicionadas informações sobre como o Step Functions analisa arquivos CSV usados como entrada no estado de Mapa Distribuído. Para ter mais informações, consulte Arquivo CSV em um bucket do Amazon S3.	14 de março de 2023
Atualização somente da documentação	Foram adicionadas informações sobre como o Step Functions lida com invocações entre contas para o padrão Executar um trabalho (.sync). Para obter mais informações, consulte Executar um trabalho (.sync).	1.º de março de 2023
Atualização somente da documentação	Reduza o período de retenção do histórico das execuções de fluxo de trabalho concluídas de 90 dias para 30 dias. Para obter mais informações sobre como ajustar o período de retenção, consulte Garantias de execução e Cotas relacionadas a execuções de máquina de estado.	21 de fevereiro de 2023
Atualizar	O Step Functions expandiu o suporte para integrações de AWS SDK adicionando 35 AWS serviços e 1100 novas ações de API. Para obter mais informações, consulte Integrações de serviços AWS SDK compatíveis e Registro de alterações para integrações de AWS SDK compatíveis.	17 de fevereiro de 2023
Atualização somente da documentação	Foi publicada uma série de tutoriais Conceitos básicos que explica o processo de criação de um fluxo de trabalho para uma aplicação de cartão de crédito usando o Step Functions. Para ter mais informações, consulte Começando com AWS Step Functions.	30 de dezembro de 2022
Novo atributo	O Step Functions adiciona suporte para orquestrar fluxos de trabalho paralelos em grande escala para processam ento de dados usando um novo Modo distribuído para o estado `Map`. Para ter mais informações, consulte Usar o estado Mapa no modo distribuído para orquestrar workloads paralelas em grande escala.	1.º de dezembro de 2022

Alteração	Descrição	Alterado em
Novo atributo	O Step Functions agora oferece suporte ao acesso a AWS recursos de várias contas configurados em outras contas. Para obter mais informações, consulte • Acessando recursos em outros Contas da AWS em seus fluxos de trabalho • Tutorial: Acessando recursos entre contas AWS • Task state	18 de novembro de 2022
Atualizar	O Step Functions agora fornece uma nova experiência de console para visualizar e depurar execuções de fluxo de trabalho expressoo. Para obter mais informações, consulte: • Execuções de Fluxo de trabalho Padrão e Expresso no console • Visualizar e depurar execuções no console do Step Functions	18 de outubro de 2022
Atualizar	Foi adicionado suporte para especificar opcionalmente o parâmetro `ExecutionRoleArn` ao usar as APIs `addStep` e `addStep.sync` para a integração de serviços otimizados do Amazon EMR. Para ter mais informações, consulte Chamar o Amazon EMR com o Step Functions.	20 de setembro de 2022
Atualização somente da documentação	Foi adicionado um novo tópico que fornece recomenda ções sobre como otimizar custos ao criar fluxos de trabalho de tecnologia sem servidor usando o Step Functions. Para ter mais informações, consulte Otimização de custos usando o fluxos de trabalho expressos.	15 de setembro de 2022

Alteração	Descrição	Alterado em
Atualizar	O Step Functions adiciona suporte a 14 novas funções intrínsecas para realizar tarefas de processamento de dados, como manipulações de matrizes, codificação e decodificação de dados, cálculos de hash, manipulação de dados JSON, operações de funções matemáticas e geração de identificadores exclusivos. Atualização somente da documentação: Foram agrupadas todas as funções intrínsecas existentes e recém-introduzidas nas seguintes categorias com base no tipo de tarefa de processamento de dados que elas ajudam você a realizar: • Funções intrínsecas para matrizes • Funções intrínsecas para codificação e decodificação de dados • função intrínseca para cálculo de hash • função intrínseca para manipulação de dados JSON • Funções intrínsecas para operações matemáticas • Função intrínseca para operação de strings • Função intrínseca para geração de identificadores exclusivos • Função intrínseca para operação genérica Para ter mais informações, consulte Funções intrínsecas.	31 de agosto de 2022
Atualizar	O Step Functions expandiu o suporte para integrações de AWS SDK adicionando mais três AWS serviços — AWS Billing Conductor,Amazon GameSparks, e. Amazon Pinpoint SMS and Voice V2 Para ter mais informações, consulte Registro de alterações para integrações de AWS SDK compatíveis.	26 de julho de 2022

Alteração	Descrição	Alterado em
Atualização somente da documentação	Foi adicionado um novo tópico para incluir um resumo de todas as atualizações feitas nas integrações do AWS SDK suportadas pelo Step Functions. Para mais informações, consulte Registro de alterações para integrações de AWS SDK compatíveis.	26 de julho de 2022
Atualização somente da documentação	AWS Step Functions O Guia do Desenvolvedor agora inclui detalhes sobre as métricas de execução emitidas especific amente para fluxos de trabalho expressos. Para ter mais informações, consulte Métricas de execução para fluxos de trabalho expressos.	9 de junho de 2022

Alteração	Descrição	Alterado em
Atualizar	Aprimoramentos do console do Step Functions O novo console agora apresenta uma página de Detalhes de execução reprojetada que inclui os seguintes aprimoramentos: • Capacidade de identificar rapidamente o motivo de uma falha na execução. • Dois novos modos de visualização para a máquina de estado: Visualização em tabela e Visualização de eventos. Essas visualizações também oferecem a capacidade de aplicar filtros para visualizar apenas as informações de interesse. Além disso, você pode classificar o conteúdo da Visualização do evento com base nos registros de data e hora do evento. • Alterne entre as diferentes iterações do estado `Map` no modo de Exibição em gráfico usando uma lista suspensa ou na visualização em árvore do modo de Visualização em tabela para estados `Map`. • Visualize informações detalhadas sobre cada estado no fluxo de trabalho, incluindo o caminho completo de transferência de dados de entrada e saída e novas tentativas para estados `Task` ou `Parallel`. • Aprimoramentos diversos, incluindo a opção de copiar o nome do recurso da Amazon de execução da máquina de estado, visualizar a contagem do total de transições da máquina de estado e exportar os detalhes da execução no formato JSON. Atualizações somente da documentação Foi adicionado um novo tópico para explicar os vários tipos de informações exibidas na página Detalhes da execução.	9 de maio de 2022

Alteração	Descrição	Alterado em
	Além disso, foi adicionado um tutorial para mostrar como examinar essas informações. Para obter mais informações, consulte: • Visualizar e depurar execuções no console do Step Functions • Tutorial: como examinar execuções de máquinas de estado usando o console do Step Functions	
Atualizar	O Step Functions agora fornece uma solução alternativa para impedir o confuso problema de segurança adjunta, que surge quando uma entidade (um serviço ou uma conta) é coagida por outra entidade a realizar uma ação. Para obter mais informações, consulte: • Prevenção do problema do substituto confuso entre serviços	2 de maio de 2022

Alteração	Descrição	Alterado em

Atualizar	• O Step Functions expandiu o suporte para integraçõ es de AWS SDK adicionando mais AWS 21 serviços. Para obter mais informações, consulte: Integrações de serviços AWS SDK compatíveis.
	• Atualizações somente da documentação:
	  • Foi adicionada uma lista de todos os prefixos de exceção presentes nas exceções que são geradas quando você executa erroneamente uma AWS integração de serviços SDK com Step Functions. Para obter mais informações, consulte: Integrações de serviços AWS SDK compatíveis.
	  • Foi adicionada uma lista de todas as ações de API não suportadas para integrações de AWS SDK compatíve is. Para obter mais informações, consulte: Ações de API não compatíveis para serviços compatíveis.
	  • Foi adicionada uma lista de todas as integrações de AWS SDK compatíveis que agora estão obsoletas. Para obter mais informações, consulte: Integrações de serviços SDK obsoletas AWS.
		19 de abril de 2022

Novo atributo	O Step Functions Local agora suporta integração de AWS SDK e simulação de integrações de serviços. Para obter mais informações, consulte: • Usando integrações de serviços simulados	28 de janeiro de 2022

Alteração	Descrição	Alterado em
Novo atributo	AWS Step Functions agora suporta a criação de uma API REST do Amazon API Gateway com máquina de estado expresso síncrona como integração de back-end usando o. AWS Cloud Development Kit (AWS CDK) Para obter mais informações, consulte: • Criando uma API REST do API Gateway com o Synchronous Express State Machine usando o AWS CDK	10 de dezembro de 2021
Atualizar	O Step Functions adicionou três novos projetos de amostra que demonstram a integração do Step Functions com o console atualizado do Amazon Athena. Para obter mais informações, consulte: • Executar várias consultas (Amazon Athena, Amazon SNS) • Consulte grandes conjuntos de dados (Amazon Athena, Amazon S3, Amazon SNS AWS Glue) • Mantenha os dados atualizados (Amazon Athena, Amazon S3,) AWS Glue	22 de novembro de 2021
Novo atributo	O Step Functions adicionou suporte a endpoints da Amazon VPC para Synchronous fluxos de trabalho expressos. Para obter mais informações, consulte: • Endpoints da Amazon VPC para o Step Functions	15 de novembro de 2021

Alteração	Descrição	Alterado em
Atualizar	AWS Step Functions adicionou três novos projetos de amostra que demonstram como usar a AWS Batch integração Step Functions. Para obter mais informações, consulte: • Distribuir um AWS Batch emprego • AWS Batch com Lambda • Uso Step Functions e AWS Batch com tratamento de erros	14 de outubro de 2021
Novo atributo	AWS Step Functions adicionou integrações de AWS SDK, permitindo que você use as ações da API para todos os mais de duzentos AWS serviços. Para obter mais informações, consulte: • AWS Integrações de serviços SDK • Reúna informações do bucket do Amazon S3 usando integrações de serviços AWS SDK	30 de setembro de 2021
Novo atributo	AWS Step Functions adicionou um designer visual de fluxo de trabalho, o AWS Step Functions Workflow Studio. Para obter mais informações, consulte: • AWS Step Functions Studio de fluxo de trabalho • Aprenda a usar o AWS Step Functions Workflow Studio	17 de junho de 2021
Atualizar	AWS Step Functions adicionou quatro novas APIs,`StartBuildBatch` ,`StopBuildBatch` , `RetryBuildBatch` `eDeleteBuildBatch` , à CodeBuild integração. Para obter mais informações, consulte: • Chamada AWS CodeBuild com Step Functions	04 de junho de 2021

Alteração	Descrição	Alterado em
Novo atributo	AWS Step Functions agora se integra com a Amazon EventBridge. Para obter mais informações, consulte: • Chamada EventBridge com Step Functions • Políticas do IAM para Step Functions e Políticas do IAM para a Amazon EventBridge • Um exemplo de projeto que mostra como Envie um evento personalizado para EventBridge	14 de maio de 2021
Atualizar	AWS Step Functions adicionou um novo projeto de amostra que mostra como usar o Step Functions e a API Amazon Redshift Data para executar um fluxo de trabalho ETL/ELT. Para obter mais informações, consulte: • Executar fluxos de trabalho ETL/ELT usando o Amazon Redshift (Lambda, API de dados do Amazon Redshift)	16 de abril de 2021
Novo atributo	AWS Step Functions tem um novo simulador de fluxo de dados no console. Para obter mais informações, consulte: • Console do Step Functions	8 de abril de 2021
Novo atributo	AWS Step Functions agora se integra ao Amazon EMR no EKS. Para obter mais informações, consulte: • Ligue para o Amazon EMR no EKS com AWS Step Functions	29 de março de 2021
Atualizar	O suporte a YAML para definições de máquina de estado foi adicionado a AWS Toolkit for Visual Studio Code e AWS CloudFormation. Para obter mais informações, consulte: • Suporte ao formato de definição • AWS Toolkit for Visual Studio Code	4 de março de 2021

Alteração	Descrição	Alterado em
Novo atributo	AWS Step Functions agora se integra com AWS Glue DataBrew. Para obter mais informações, consulte: • Gerencie AWS Glue DataBrew trabalhos com Step Functions • O que AWS Glue DataBrewé no guia do DataBrew desenvolvedor.	6 de janeiro de 2021
Novo atributo	AWS Step Functions Os fluxos de trabalho expressos síncronos agora estão disponíveis, oferecendo uma maneira fácil de orquestrar microsserviços. Para obter mais informações, consulte: • Fluxos de trabalho expresso síncronos e assíncronos • Um exemplo de projeto que mostra como Invocar fluxos de trabalho expresso síncronos • A documentação StartSyncExecutionda API.	24 de novembro de 2020
Novo atributo	AWS Step Functions agora se integra ao Amazon API Gateway. Para obter mais informações, consulte: • Chame o API Gateway com o Step Functions • Políticas do IAM para Step Functions e Políticas do IAM para o Amazon API Gateway • Um exemplo de projeto que mostra como Fazer uma chamada para o API Gateway	17 de novembro de 2020

Alteração	Descrição	Alterado em
Novo atributo	AWS Step Functions agora se integra ao Amazon Elastic Kubernetes Service. Para obter mais informações, consulte: • Chamar o Amazon EKS com o Step Functions • Políticas do IAM para Step Functions e Políticas do IAM para o Amazon EKS • Um exemplo de projeto que mostra como Gerenciar um cluster do Amazon EKS	16 de novembro de 2020
Novo atributo	AWS Step Functions agora se integra ao Amazon Athena. Para obter mais informações, consulte: • Chame o Athena com o Step Functions • Políticas do IAM para Step Functions e Políticas do IAM para o Amazon Athena • Um exemplo de projeto que mostra como Iniciar uma consulta do Athena	22 de outubro de 2020
Novo atributo	AWS Step Functions agora oferece suporte ao rastreamento de end-to-end fluxos de trabalho com AWS X-Ray, oferecendo visibilidade total das execuções de máquinas de estado e facilitando a análise e a depuração de seus aplicativos distribuídos. Para obter mais informações, consulte: • AWS X-Ray e Step Functions • Políticas do IAM para Step Functions e Políticas do IAM para AWS X-Ray • AWS Step Functions API Reference • TracingConfiguration	14 de setembro de 2020

Alteração	Descrição	Alterado em

Atualizar	AWS Step Functions agora suporta tamanhos de carga útil de até 256 KB de dados como uma string codificada em UTF-8. Isso permite processar payloads maiores nos fluxos de trabalho Padrão e Expresso.
	
	As máquinas de estado existentes não precisam ser alteradas para usar payloads maiores. No entanto, você precisará atualizar para as versões mais recentes do SDK do Step Functions e do Local Runner para usar as APIs atualizadas. Para obter mais informações, consulte:
	
	• Cotas
	• the section called “Use ARNs do Amazon S3 em vez de transmitir grandes cargas”
	• `States.DataLimitExceeded`
	• the section called “Payloads do CloudWatch Logs”
	• the section called “EventBridge cargas úteis”
	• AWS Step Functions API Reference
	
	  • CloudWatchEventsExecutionDataDetails
	  • HistoryEventExecutionDataDetails
	  • GetExecutionHistory
	  • ActivityScheduledEventDetails
	  • ActivitySucceededEventDetails
	  • CloudWatchEventsExecutionDataDetails
	  • ExecutionSucceededEventDetails
	  • LambdaFunctionScheduledEventDetails
	  • ExecutionSucceededEventDetails
	  • StateEnteredEventDetails
	  • StateExitedEventDetails
	  • TaskSubmittedEventDetails
		3 de setembro de 2020

Alteração	Descrição	Alterado em
	• TaskSucceededEventDetails	

Alteração	Descrição	Alterado em

Atualizar	A Amazon States Language foi atualizada da seguinte forma:
	• Choice Rules adicionaram
	  • Um operador de comparação nulo, `IsNull`. `IsNull` testa o valor nulo do JSON e pode ser usado para detectar se a saída de um estado anterior é nula ou não.
	  • Quatro outros novos operadores foram adicionados IsBoolean, IsNumeric, IsString IsTimestamp e.
	  • Um teste para a existência ou não existência de um campo usando o operador `IsPresent` . `IsPresent` pode ser usado para evitar erros `States.Runtime` quando há uma tentativa de acessar uma chave inexistente.
	  • Correspondência de padrões curinga para dar suporte à comparação de strings com padrões com um ou mais curingas.
	  • Comparação entre duas variáveis para operadores de comparação compatíveis.
	• Os valores de tempo limite e heartbeat em um estado `Task` agora podem ser fornecidos dinamicamente a partir da entrada de estado, em vez de um valor fixo usando os campos `TimeoutSecondsPath` e `HeartbeatSecondsPath` . Consulte o estado Estado da tarefa para obter mais informações.
	• O novo campo ResultSelector fornece uma forma de manipular o resultado de um estado antes de `ResultPath` ser aplicado. O campo `ResultSelector` é um campo opcional nos estados Mapa, Paralelo e Estado da tarefa.
		13 de agosto de 2020

Alteração	Descrição	Alterado em
	• Funções intrínsecas foram adicionadas para permitir operações básicas sem estados `Task`. As funções intrínsecas podem ser usadas nos campos `Parameters` e `ResultSelector`.	
Atualizar	AWS Step Functions agora suporta a chamada de SageMaker `CreateProcessingJob` API da Amazon. Para obter mais informações, consulte: • Gerencie SageMaker com Step Functions • Pré-processar dados e treinar um modelo de Machine Learning, um exemplo de projeto que demonstra `CreateProcessingJob`.	4 de agosto de 2020
Novo atributo	AWS Step Functions agora é suportado pelo AWS Serverless Application Model, facilitando a integração da orquestração do fluxo de trabalho em seus aplicativos sem servidor. Para obter mais informações, consulte: • AWS Step Functions e AWS SAM • AWS::Serverless::StateMachine • AWS SAM Modelos de política	27 de maio de 2020
Novo atributo	AWS Step Functions introduziu uma nova invocação síncrona para aninhar execuções de Step Functions. A nova invocação, `arn:aws:states:::states:startExecution.sync:2`, retorna um objeto JSON. A invocação original, `arn:aws:states:::states:startExecution.sync`, continua a ter suporte e retorna uma string JSON com escape. Para obter mais informações, consulte: • Gerencie AWS Step Functions execuções como um serviço integrado	19 de maio de 2020

Alteração	Descrição	Alterado em
Novo atributo	AWS Step Functions agora se integra com AWS CodeBuild . Para obter mais informações, consulte: • Usando AWS Step Functions com outros serviços • Chamada AWS CodeBuild com Step Functions • Integrações otimizadas para o Step Functions	5 de maio de 2020
Novo atributo	O Step Functions agora é compatível com o AWS Toolkit for Visual Studio Code, facilitando a criação e a visualização de fluxos de trabalho baseados em máquinas de estado sem sair do editor de código.	31 de março de 2020
Atualizar	Agora você pode configurar o registro no Amazon CloudWatch Logs para fluxos de trabalho padrão. Para obter mais informações, consulte: • Como registrar usando o CloudWatch Logs	25 de fevereiro de 2020
Novo atributo	AWS Step Functions agora podem ser acessados sem a necessidade de um endereço IP público, diretamente da Amazon Virtual Private Cloud (VPC). Para obter mais informações, consulte: • Endpoints da Amazon VPC para o Step Functions	23 de dezembro de 2019

Alteração	Descrição	Alterado em

Novo atributo	Os fluxos de trabalho expressos são um novo tipo de fluxo de trabalho, adequados para cargas de trabalho de processamento de eventos de alto volume, como ingestão de dados de IoT, processamento e transformação de dados de streaming e back-ends de aplicativos móveis.
	
	Para obter mais informações, leia os seguintes tópicos novos e atualizados.
	
	• Comparação entre os fluxos de trabalho padrão e expresso
	  • Garantias de execução
	• Usando AWS Step Functions com outros serviços
	  • Integrações otimizadas para o Step Functions
	• Processar mensagens de alto volume do Amazon SQS (fluxos de trabalho expressos)
	• Exemplo de verificação seletiva (Fluxos de trabalho expressos)
	• Cotas
	  • Cotas
	• Como registrar usando o CloudWatch Logs
	• AWS Step Functions API Reference
	  • CreateStateMachine
	  • UpdateStateMachine
	  • DescribeStateMachine
	  • DescribeStateMachineForExecution
	  • StopExecution
	  • DescribeExecution
	  • GetExecutionHistory
	  • ListExecutions
		3 de dezembro de 2019

Alteração	Descrição	Alterado em
	• ListStateMachines • StartExecution • CloudWatchLogsLogGroup • LogDestination • LoggingConfiguration	
Novo atributo	AWS Step Functions agora se integra ao Amazon EMR. Para obter mais informações, consulte: • Usando AWS Step Functions com outros serviços • Chamar o Amazon EMR com o Step Functions • Integrações otimizadas para o Step Functions	19 de novembro de 2019

Atualizar	AWS Step Functions lançou o AWS Step Functions Data Science SDK. Para obter mais informações, consulte.
	• Projeto no Github
	• Documentação do SDK
	• Os seguintes exemplos de notebooks, que estão disponíveis no SageMakerconsole e no GitHub projeto relacionado.
	  • `hello_world_workflow.ipynb`
	  • `machine_learning_workflow_abalone.ipynb`
	  • `training_pipeline_pytorch_mnist.ipynb`
		7 de novembro de 2019

Alteração	Descrição	Alterado em
Atualizar	O Step Functions agora suporta mais ações de API para a Amazon SageMaker e inclui dois novos projetos de amostra para demonstrar a funcionalidade. Para obter mais informações, consulte. • Gerencie SageMaker com Step Functions • Usando AWS Step Functions com outros serviços • Treinar um modelo de machine learning. • Ajustar um modelo de machine learning	3 de outubro de 2019
Novo atributo	O Step Functions oferece suporte ao início de novas execuções de fluxo de trabalho chamando `StartExecution` como uma API de serviço integrada. Consulte: • Iniciar execuções de fluxo de trabalho usando um estado Tarefa • Gerencie AWS Step Functions execuções como um serviço integrado • Usando AWS Step Functions com outros serviços • Políticas do IAM para iniciar execuções de fluxo de trabalho do Step Functions	12 de agosto de 2019

Alteração	Descrição	Alterado em
Novo atributo	O Step Functions inclui a capacidade de transmitir um token da tarefa a serviços integrados e pausar a execução até que esse token da tarefa retorne com `SendTaskSuccess` ou `SendTaskFailure` . Consulte: • Padrões de integração de serviço • Aguardar um retorno de chamada com um token de tarefa • Exemplo de Padrão de Retorno de Chamada (Amazon SQS, Amazon SNS, Lambda) • Integrações otimizadas para o Step Functions • Implantar um projeto de aprovação humana de exemplo • Métricas de integração de serviço O Step Functions agora fornece uma maneira de acessar informações dinâmicas sobre a execução atual diretamente no campo `"Parameters"` de uma definição de estado. Consulte: • Objeto de contexto • Passar nós do objeto de contexto como parâmetros	23 de maio de 2019
Novo atributo	O Step Functions suporta CloudWatch Eventos para alterações de status de execução, consulte: • EventBridge (CloudWatch Eventos) para mudanças no status de execução do Step Functions • Guia do usuário do Amazon CloudWatch Events	8 de maio de 2019

Alteração	Descrição	Alterado em
Novo atributo	O Step Functions oferece suporte a permissões do IAM usando tags. Para obter mais informações, consulte: • Marcação no Step Functions • Políticas baseadas em tag	5 de março de 2019
Novo atributo	O Step Functions Local agora está disponível. Você pode executar o Step Functions na máquina local para testes e desenvolvimento. O Step Functions Local está disponível para download como uma aplicação Java ou como uma imagem do Docker. Consulte Testando máquinas de estado localmente.	4 de fevereiro de 2019
Novo atributo	AWS Step Functions agora está disponível nas regiões de Pequim e Ningxia. Consulte Regiões com suporte.	15 de janeiro de 2018
Novo atributo	O Step Functions oferece suporte à marcação de recursos para ajudar a rastrear a alocação de custos. É possível marcar máquinas de estado na página Details (Detalhes) ou por ações de API. Consulte Marcação no Step Functions.	7 de janeiro de 2019
Novo atributo	AWS Step Functions agora está disponível nas regiões da Europa (Paris) e América do Sul (São Paulo). Consulte Regiões com suporte.	13 de dezembro de 2018
Novo atributo	AWS Step Functions agora está disponível na região Europa (Estocolmo). Consulte Regiões com suporte para obter uma lista de regiões compatíveis.	12 de dezembro de 2018

Alteração	Descrição	Alterado em
Novo atributo	O Step Functions agora se integra a alguns AWS serviços. Agora você pode chamar e transmitir parâmetros diretamente para a API desses serviços integrados a partir de um estado de tarefa na Amazon States Language. Para obter mais informações, consulte: • Usando AWS Step Functions com outros serviços • Transmitir parâmetros para uma API de serviço • Integrações otimizadas para o Step Functions	29 de novembro de 2018
Atualizar	Descrição melhorada de `TimeoutSeconds` e `HeartbeatSeconds` na documentação de estados de tarefa. Consulte Estado da tarefa.	24 de outubro de 2018
Atualizar	Melhorada a descrição para o limite Tamanho máximo do histórico de execução e fornecido um link para o tópico relacionado com as melhores práticas. • Cotas relacionadas a execuções de máquina de estado • Evitar atingir a cota do histórico	17 de outubro de 2018
Atualizar	Foi adicionado um novo tutorial à AWS Step Functions documentação: VejaIniciar a execução de uma máquina de estado em resposta a eventos do Amazon S3.	25 de setembro de 2018
Atualizar	Removida a entrada Máximo de execuções exibidas no console do Step Functions da documentação sobre limites. Consulte Cotas.	13 de setembro de 2018
Atualizar	Foi adicionado um tópico de melhores práticas à AWS Step Functions documentação sobre como melhorar a latência ao pesquisar tarefas de atividade. Consulte Evitar latência ao fazer uma sondagem de tarefas de atividade.	30 de agosto de 2018

Alteração	Descrição	Alterado em
Atualizar	Melhorou o AWS Step Functions tópico sobre atividades e trabalhadores de atividades. Consulte Atividades.	29 de agosto de 2018
Atualizar	Melhorou o AWS Step Functions tópico sobre CloudTrail integração. Consulte Gravando chamadas de API com AWS CloudTrail.	7 de agosto de 2018
Atualizar	Exemplos de JSON adicionados ao AWS CloudFormation tutorial. Consulte Como criar uma máquina de estado Lambda para o Step Functions usando o AWS CloudFormation.	23 de junho de 2018
Atualizar	Foi adicionado um novo tópico sobre o tratamento de erros de serviço do Lambda. Consulte Lidar com exceções do serviço Lambda.	20 de junho de 2018
Novo atributo	AWS Step Functions agora está disponível na região Ásia-Pacífico (Mumbai). Consulte Regiões com suporte para obter uma lista de regiões compatíveis.	28 de junho de 2018
Novo atributo	AWS Step Functions agora está disponível na região AWS GovCloud (Oeste dos EUA). Consulte Regiões com suporte para obter uma lista de regiões compatíveis. Para obter informações sobre o uso de Step Functions na região AWS GovCloud (Oeste dos EUA), consulte AWS GovCloud (US).	28 de junho de 2018
Atualizar	Documentação melhorada sobre lidar com erro para estados de `Parallel`. Consulte Como tratar erros.	20 de junho de 2018

Alteração	Descrição	Alterado em
Atualizar	Foi melhorada a documentação sobre o processamento de entrada e saída no Step Functions. Saiba como usar `InputPath` , `ResultPath` e `OutputPath` para controlar o fluxo de JSON por meio de seus fluxos de trabalho, estados e tarefas. Consulte: • Processamento de entrada e saída no Step Functions • ResultPath	7 de junho de 2018
Atualizar	Exemplos de código melhorado para estados paralelos. Consulte Paralelo.	4 de junho de 2018
Novo atributo	Agora você pode monitorar as métricas de API e serviço em CloudWatch. Consulte Monitorando Step Functions usando CloudWatch.	25 de maio de 2018
Atualizar	`StartExecution` , `StopExecution` e `StateTransition` agora têm limites de controle de utilização maiores nas seguintes regiões: • Leste dos EUA (Norte da Virgínia) • Oeste dos EUA (Oregon) • Europa (Irlanda) Para obter mais informações, consulte Cotas.	16 de maio de 2018
Novo atributo	AWS Step Functions agora está disponível nas regiões Oeste dos EUA (Norte da Califórnia) e Ásia-Pacífico (Seul). Consulte Regiões com suporte para obter uma lista de regiões compatíveis.	5 de maio de 2018

Alteração	Descrição	Alterado em
Atualizar	Procedimentos atualizados e imagens para corresponder alterações na interface.	25 de abril de 2018
Atualizar	Adicionou um novo tutorial que mostra como iniciar uma nova execução para continuar a trabalhar. Consulte Execuções contínuas de fluxo de trabalho de longa duração como uma nova execução. Este tutorial descreve um padrão de projeto que pode ajudar a evitar algumas limitações de serviço. Consulte Evitar atingir a cota do histórico.	19 de abril de 2018
Atualizar	Introdução aprimorada à documentação dos estados adicionando informações conceituais sobre máquinas de estado. Consulte States.	9 de março de 2018
Atualizar	Além de HTML, PDF e Kindle, o Guia do AWS Step Functions Desenvolvedor está disponível em GitHub. Para deixar um feedback, escolha o GitHub ícone no canto superior direito. 	2 de março de 2018
Atualizar	Foi adicionado um tópico que descreve outros recursos relacionados ao Step Functions. Consulte Informações relacionadas.	20 de fevereiro de 2018

Alteração	Descrição	Alterado em

Novo atributo	• Quando você cria uma nova máquina de estado, deve confirmar que o AWS Step Functions criará um perfil do IAM que permite o acesso às funções do Lambda.
	• Atualizados os seguintes tutoriais para refletir as pequenas alterações no fluxo de trabalho de criação de uma máquina de estado:
	  • Como criar uma máquina de estado Step Functions que usa Lambda
	  • Como criar uma máquina de estado de Atividade usando o Step Functions
	  • Tratar condições de erro usando uma máquina de estado Step Functions
	  • Repita um loop com o Lambda
		19 de fevereiro de 2018

Atualizar	Adicionado tópico que descreve um exemplo de operador de atividade escrito em Ruby. Essa implementação pode ser usada para criar um operador de atividade em Ruby diretamente ou como um padrão de design para a criação de um operador de atividade em outra linguagem. Consulte Exemplo de operador de atividade em Ruby.	6 de fevereiro de 2018
Atualizar	Foi adicionado um novo tutorial que descreve um padrão de design que usa uma função do Lambda para iterar uma contagem. Consulte Como criar uma máquina de estado Step Functions que usa Lambda.	31 de janeiro de 2018

Alteração	Descrição	Alterado em
Atualizar	Foi atualizado o conteúdo sobre permissões do IAM para incluir APIs de `DescribeStateMachineForExecution` e `UpdateStateMachine`. Consulte Como criar permissões granulares do IAM para usuários que não são administradores.	26 de janeiro de 2018
Atualizar	Foram adicionadas regiões recém-disponibilizadas: Canadá (Central), Ásia-Pacífico (Singapura). Consulte Regiões com suporte.	25 de janeiro de 2018
Atualizar	Foram atualizados tutoriais e procedimentos para mostrar que o IAM permite selecionar o Step Functions como um perfil.	24 de janeiro de 2018
Atualizar	Adicionado novo tópico de Melhores práticas que sugere não transmitir cargas grandes entre estados. Consulte Use ARNs do Amazon S3 em vez de transmitir grandes cargas.	23 de janeiro de 2018
Atualizar	Corrigidos procedimentos para corresponder à interface atualizada para a criação de uma máquina de estado: • Como criar uma máquina de estado Step Functions que usa Lambda • Como criar uma máquina de estado de Atividade usando o Step Functions • Tratar condições de erro usando uma máquina de estado Step Functions	17 de janeiro de 2018

Alteração	Descrição	Alterado em
Novo recurso	Você pode usar projetos de amostra para provisionar rapidamente máquinas de estado e todos os recursos da AWS relacionados. Consulte Projetos de amostra para Step Functions. Entre os projetos de amostra disponíveis estão: • Pesquisa de status de trabalho (AWS Batch Lambda,) • Temporizador de tarefas (Lambda, Amazon SNS) **Note** Esses projetos de amostra e a documentação relacionada substituem tutoriais que descreviam a implementação da mesma funcionalidade.	11 de janeiro de 2018
Atualizar	Foi adicionada uma seção de Melhores práticas que inclui informações sobre como evitar execuções travadas. Consulte Práticas recomendadas do Step Functions.	5 de janeiro de 2018
Atualizar	Foi adicionada uma observação sobre como novas tentativas podem afetar a definição de preço: **Note** As novas tentativas são tratadas como transições de estado. Para ver informações sobre como as transições de estado afetam o faturamento, consulte Preços do Step Functions.	8 de dezembro de 2017

Alteração	Descrição	Alterado em
Atualizar	Informações relacionadas adicionadas aos nomes de recurso: **Note** Step Functions permite criar nomes para máquinas de estado, execuções e atividades, além de rótulos que contenham caracteres não ASCII. Esses nomes não ASCII não funcionam com a Amazon. CloudWatch Para garantir que você possa acompanhar CloudWatch as métricas, escolha um nome que use somente caracteres ASCII.	6 de dezembro de 2017
Atualizar	As informações de visão geral da segurança foram aprimoradas e um tópico sobre permissões granulares do IAM foi adicionado. Consulte Segurança em AWS Step Functions e Como criar permissões granulares do IAM para usuários que não são administradores.	27 de novembro de 2017
Novo recurso	Você pode atualizar uma máquina de estado existente. Consulte Atualizar a máquina de estado.	15 de novembro de 2017

Alteração	Descrição	Alterado em
Atualizar	Foi adicionada uma nota para esclarecer erros `Lambda.Unknown`, e ela foi vinculada à documentação do Lambda nas seções a seguir: • Nomes de erro • Etapa 3: Criar uma máquina de estado com um campo Capturar **Note** Os erros não tratados no Lambda são relatados como `Lambda.Unknown` na saída do erro. Isso inclui out-of-memory erros e tempos limite de função. Você pode combinar com `Lambda.Unknown`, `States.ALL` ou `States.TaskFailed` para lidar com esses erros. Quando o Lambda atinge o número máximo de invocações, o erro é `Lambda.TooManyRequestsException`. Para obter mais informações sobre erros da função do Lambda, consulte Tratamento de erros e novas tentativas automáticas no Guia do desenvolvedor do AWS Lambda .	17 de outubro de 2017
Atualizar	As instruções do IAM foram corrigidas e esclarecidas, e as capturas de tela foram atualizadas em todos os tutoriais.	11 de outubro de 2017

Alteração	Descrição	Alterado em

Atualizar	• Foram adicionadas novas capturas de tela para que os resultados da execução da máquina de estado reflitam as alterações no console do Step Functions. As instruções do Lambda foram reescritas nos seguintes tutoriais para refletir as alterações no console do Lambda:
	  • Como criar uma máquina de estado Step Functions que usa Lambda
	  • Como criar um instrumento de sondagem de status de trabalho
	  • Criar um temporizador de tarefas
	  • Tratar condições de erro usando uma máquina de estado Step Functions
	• Foram corrigidas e esclarecidas informações sobre a criação de máquinas de estado nas seções a seguir:
	  • Como criar uma máquina de estado de Atividade usando o Step Functions
		6 de outubro de 2017

Atualizar	As instruções do Lambda foram reescritas nas seguintes seções para refletir as alterações no console do IAM: • Criar um perfil do IAM para sua máquina de estado • Como criar uma máquina de estado Step Functions que usa Lambda • Como criar um instrumento de sondagem de status de trabalho • Criar um temporizador de tarefas • Tratar condições de erro usando uma máquina de estado Step Functions • Criar uma API do Step Functions usando o API Gateway	5 de outubro de 2017

Alteração	Descrição	Alterado em
Atualizar	A seção Dados da máquina de estado foi reescrita.	28 de setembro de 2017
Novo atributo	Os limites relacionados ao controle de utilização de ações de API são aumentados para todas as regiões em que o Step Functions está disponível.	18 de setembro de 2017
Atualizar	• Informações corrigidas e elucidadas sobre como iniciar novas execuções em todos os tutoriais. • Informações corrigidas e elucidadas na seção Cotas relacionadas a contas.	14 de setembro de 2017
Atualizar	Os seguintes tutoriais foram reescritos para refletir as alterações no console do Lambda: • Como criar uma máquina de estado Step Functions que usa Lambda • Tratar condições de erro usando uma máquina de estado Step Functions • Como criar um instrumento de sondagem de status de trabalho	28 de agosto de 2017
Novo atributo	O Step Functions está disponível na Europa (Londres).	23 de agosto de 2017
Novo atributo	Os fluxos de trabalho visuais das máquinas de estado permitem que você amplie, reduza e centralize o gráfico.	21 de agosto de 2017

Alteração	Descrição	Alterado em
Novo atributo	**Important** Uma execução não pode usar o nome de outra execução por 90 dias. Quando você faz várias `StartExecution` chamadas com o mesmo nome, a nova execução não é executada. Para obter mais informações, consulte o parâmetro de solicitação `name` da ação da API `StartExecution` na Referência de APIs do AWS Step Functions .	18 de agosto de 2017
Atualizar	Informações adicionadas ao tutorial Criar uma API do Step Functions usando o API Gateway sobre uma alternativa para passar o ARN da máquina de estado.	17 de agosto de 2017
Atualizar	O novo tutorial Como criar um instrumento de sondagem de status de trabalho foi adicionado.	10 de agosto de 2017
Novo atributo	• Step Functions emite a `ExecutionThrottled` CloudWatch métrica. Para ter mais informações, consulte Monitorando Step Functions usando CloudWatch. • Adicionada a seção Cotas relacionadas aos controles de utilização de estado.	3 de agosto de 2017
Atualizar	Atualização das instruções na seção Etapa 1: Criar um perfil do IAM para o API Gateway.	18 de julho de 2017
Atualizar	Informações corrigidas e elucidadas na seção Choice.	23 de junho de 2017

Alteração	Descrição	Alterado em
Atualizar	Foram adicionadas informações sobre o uso de recursos em outras AWS contas aos seguintes tutoriais: • Como criar uma máquina de estado Step Functions que usa Lambda • Como criar uma máquina de estado Lambda para o Step Functions usando o AWS CloudFormation • Como criar uma máquina de estado de Atividade usando o Step Functions • Tratar condições de erro usando uma máquina de estado Step Functions	22 de junho de 2017
Atualizar	Informações corrigidas e elucidadas nas seções a seguir: • Tratar condições de erro usando uma máquina de estado Step Functions • States • Tratamento de erros no Step Functions	21 de junho de 2017
Atualizar	Todos os tutoriais foram reescritos para corresponder à atualização do console do Step Functions.	12 de junho de 2017
Novo atributo	O Step Functions está disponível na região Ásia-Pacífico (Sydney).	8 de junho de 2017
Atualizar	A seção Amazon States Language foi reestruturada.	7 de junho de 2017
Atualizar	Informações corrigidas e elucidadas na seção Como criar uma máquina de estado de Atividade usando o Step Functions.	6 de junho de 2017

Alteração	Descrição	Alterado em
Atualizar	Corrigidos os exemplos de código na seção Exemplos de máquina de estado que usam Nova tentativa e Detecção.	5 de junho de 2017
Atualizar	Reestruturou este guia usando padrões de AWS documentação.	31 de maio de 2017
Atualizar	Informações corrigidas e elucidadas na seção Paralelo.	25 de maio de 2017
Atualizar	As seções sobre caminhos e filtros foram fundidas na seção Processamento de entrada e saída no Step Functions.	24 de maio de 2017
Atualizar	Informações corrigidas e elucidadas na seção Monitorando Step Functions usando CloudWatch.	15 de maio de 2017
Atualizar	Atualização do `GreeterActivities.java` código de operador no tutorial Como criar uma máquina de estado de Atividade usando o Step Functions.	9 de maio de 2017
Atualizar	Um vídeo introdutório foi adicionado à seção O que AWS Step Functionsé.	19 de abril de 2017

Alteração	Descrição	Alterado em
Atualizar	Informações corrigidas e elucidadas nos tutoriais a seguir: • Como criar uma máquina de estado Step Functions que usa Lambda • Como criar uma máquina de estado de Atividade usando o Step Functions • Tratar condições de erro usando uma máquina de estado Step Functions	19 de abril de 2017
Atualizar	Informações sobre os modelos do Lambda foram adicionadas aos tutoriais Como criar uma máquina de estado Step Functions que usa Lambda e Tratar condições de erro usando uma máquina de estado Step Functions.	6 de abril de 2017
Atualizar	Alteração do limite "Tamanho máximo dos dados de entrada ou resultado" para "Tamanho máximo dos dados de entrada ou resultado para uma tarefa, estado ou execução" (32.768 caracteres). Para ter mais informações, consulte Cotas relacionadas a execuções de tarefas.	31 de março de 2017
Novo atributo	• O Step Functions suporta a execução de máquinas de estado definindo Step Functions como alvos do Amazon CloudWatch Events.	21 de março de 2017
Novo atributo	• O Step Functions permite que o tratamento de erros da função do Lambda seja o método de tratamento de erros preferido. • Atualização do tutorial Tratar condições de erro usando uma máquina de estado Step Functions e da seção Tratamento de erros no Step Functions.	16 de março de 2017
Novo atributo	O Step Functions está disponível na Europa (Frankfurt).	7 de março de 2017

Alteração	Descrição	Alterado em
Atualizar	Os tópicos do sumário foram reorganizados e atualizados nos seguintes tutoriais: • Como criar uma máquina de estado Step Functions que usa Lambda • Como criar uma máquina de estado de Atividade usando o Step Functions • Tratar condições de erro usando uma máquina de estado Step Functions	23 de fevereiro de 2017
Novo atributo	• A página Máquinas de estado do console do Step Functions inclui os botões Copiar para novo e Excluir. • As capturas de tela foram atualizadas para corresponder às alterações no console.	23 de fevereiro de 2017
Novo atributo	• O Step Functions oferece suporte à criação de APIs usando o API Gateway. • O tutorial Criar uma API do Step Functions usando o API Gateway foi adicionado.	14 de fevereiro de 2017
Novo atributo	• O Step Functions suporta integração com AWS CloudFormation. • O tutorial Como criar uma máquina de estado Lambda para o Step Functions usando o AWS CloudFormation foi adicionado.	10 de fevereiro de 2017
Atualizar	Elucidação sobre o comportamento atual dos campos `ResultPath` e `OutputPath` em relação aos estados `Parallel`.	6 de fevereiro de 2017
Atualizar	• Elucidação sobre restrições de denominação de máquinas de estado nos tutoriais. • Correção de alguns exemplos de código.	5 de janeiro de 2017

Alteração	Descrição	Alterado em
Atualizar	Os exemplos da função do Lambda foram atualizadas para usar o modelo de programação mais recente.	9 de dezembro de 2016
Lançamento inicial	Lançamento inicial do AWS Step Functions.	1° de dezembro de 2016

As traduções são geradas por tradução automática. Em caso de conflito entre o conteúdo da tradução e da versão original em inglês, a versão em inglês prevalecerá.