Entwicklerhandbuch

AWS Step Functions

AWS Step Functions: Entwicklerhandbuch

Table of Contents

Was ist AWS Step Functions?

AWS Step Functions ist ein visueller Workflow-Dienst, mit dem Sie verteilte Anwendungen erstellen, Prozesse automatisieren, Microservices orchestrieren und Daten- und ML-Pipelines (Machine Learning) erstellen können.

In der grafischen Konsole von Step Functions können Sie den Workflow Ihrer Anwendung als eine Reihe von ereignisgesteuerten Schritten sehen.

Step Functions basiert auf Zustandsmaschinen und Aufgaben. In Step Functions werden Zustandsmaschinen als Workflows bezeichnet. Dabei handelt es sich um eine Reihe von ereignisgesteuerten Schritten. Jeder Schritt in einem Workflow wird als Status bezeichnet. Ein Aufgabenstatus steht beispielsweise für eine Arbeitseinheit, die ein anderer AWS Dienst ausführt, z. B. das Aufrufen eines anderen Dienstes AWS-Service oder einer API.

Mit den integrierten Steuerelementen von Step Functions können Sie den Status jedes Schritts in Ihrem Workflow überprüfen, um sicherzustellen, dass Ihre Anwendung ordnungsgemäß und wie erwartet ausgeführt wird. Je nach Anwendungsfall können Sie Step Functions AWS Dienste wie Lambda aufrufen, um Aufgaben auszuführen. Sie können Workflows erstellen, die Modelle für maschinelles Lernen verarbeiten und veröffentlichen. Sie können über Step Functions AWS Dienste steuern lassen, z. B. AWS Glue um Workflows zum Extrahieren, Transformieren und Laden (ETL) zu erstellen. Sie können auch lang andauernde, automatisierte Workflows für Anwendungen erstellen, die menschliche Interaktion erfordern.

> **Tip**
>
> Um zu erfahren, wie Sie Step Functions verwenden, folgen Sie den interaktiven Modulen im AWS Step Functions Workshop oder lesen Sie den Abschnitt Erste Schritte in diesem Handbuch, um einen Workflow für Kreditkartenanträge zu erstellen.

Themen

- Unterstützte -Regionen
- Verwenden Sie Step Functions zum ersten Mal?

AWS SDK und optimierte Integrationen

Um andere AWS Dienste aufzurufen, können Sie die AWS SDK-Integrationen von Step Functions oder eine der optimierten Integrationen von Step Functions verwenden.

- Mit den AWS SDK-Integrationen können Sie jeden der über zweihundert AWS Dienste direkt von Ihrer Zustandsmaschine aus aufrufen, sodass Sie auf über neuntausend API-Aktionen zugreifen können.
- Die optimierten Integrationen von Step Functions wurden angepasst, um die Verwendung in Ihren Zustandsmaschinen zu vereinfachen.

Standard- und Express-Workflows

Step Functions hat zwei Workflowtypen. Standard-Workflows werden exakt einmal ausgeführt und können bis zu einem Jahr lang ausgeführt werden. Das bedeutet, dass jeder Schritt in einem Standard-Workflow genau einmal ausgeführt wird. Express-Workflows verfügen jedoch über eine at-least-once Workflow-Ausführung und können bis zu fünf Minuten lang ausgeführt werden. Das bedeutet, dass ein oder mehrere Schritte in einem Express-Workflow potenziell mehrmals ausgeführt werden können, während jeder Schritt im Workflow mindestens einmal ausgeführt wird.

Ausführungen sind Instanzen, in denen Sie Ihren Workflow ausführen, um Aufgaben auszuführen. Standard-Workflows eignen sich ideal für lange laufende, überprüfbare Workflows, da sie den Ausführungsverlauf und das visuelle Debugging anzeigen. Express-Workflows eignen sich ideal für high-event-rate Workloads wie Streaming-Datenverarbeitung und IoT-Datenaufnahme.

Spezifikationen für Standard-Workflows

- Ausführungsrate von 2.000 pro Sekunde
- Zustandsübergangsrate von 4.000 pro Sekunde
- Die Preise hängen vom jeweiligen Übergangsstaat ab
- Zeigt den Ausführungsverlauf und das visuelle Debugging an
- Support aller Serviceintegrationen und -muster

Spezifikationen für Express-Workflows

- Ausführungsrate von 100.000 pro Sekunde
- Nahezu unbegrenzte Zustandsübergangsrate
- Die Preise richten sich nach Anzahl und Dauer der Hinrichtungen
- Ausführungshistorie an Amazon senden CloudWatch
- Zeigt den Ausführungsverlauf und das visuelle Debugging auf der Grundlage der aktivierten Protokollebene an
- Support alle Serviceintegrationen und die meisten Muster

Weitere Informationen zu Standard- und Express-Workflows, einschließlich der Preise für Step Functions, finden Sie im Folgenden:

- Standard- und Express-Workflows
- AWS Step Functions -Preise

Anwendungsfälle

Step Functions verwaltet die Komponenten und die Logik Ihrer Anwendung, sodass Sie weniger Code schreiben und sich darauf konzentrieren können, Ihre Anwendung schnell zu erstellen und zu aktualisieren. In diesem Abschnitt werden typische Anwendungsfälle für die Arbeit mit Step Functions beschrieben.

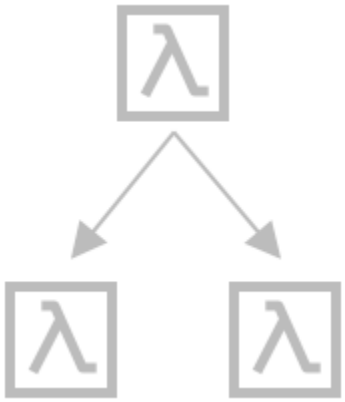
Anwendungsfall #1: Funktionsorchestrierung

Sie erstellen einen Workflow, der eine Gruppe von Lambda-Funktionen (Schritten) in einer bestimmten Reihenfolge ausführt. Die Ausgabe einer Lambda-Funktion geht an die Eingabe der nächsten Lambda-Funktion über. Der letzte Schritt in Ihrem Workflow liefert ein Ergebnis. Mit Step Functions können Sie sehen, wie jeder Schritt in Ihrem Workflow miteinander interagiert, sodass Sie sicherstellen können, dass jeder Schritt seine beabsichtigte Funktion erfüllt.

Ein Tutorial, das Ihnen zeigt, wie Sie eine Zustandsmaschine mit einer Gruppe von Funktionen erstellen, finden Sie im Folgenden:

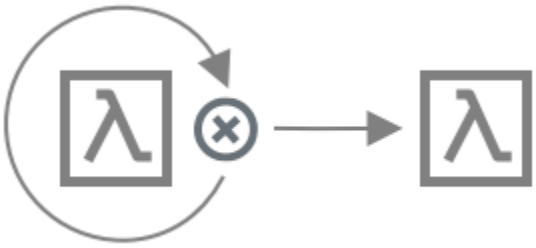
- Erste Schritte mit AWS Step Functions

Anwendungsfall #2: Verzweigung



Ein Kunde beantragt eine Erhöhung des Kreditlimits. Wenn Sie einen `Choice` Status verwenden, können Sie Step Functions veranlassen, Entscheidungen auf der Grundlage der Eingaben des `Choice` Status zu treffen. Wenn die Anfrage das vorab genehmigte Kreditlimit Ihres Kunden überschreitet, können Sie Step Functions die Anfrage Ihres Kunden zur Genehmigung an einen Manager senden lassen. Wenn die Anfrage das vorab genehmigte Kreditlimit Ihres Kunden unterschreitet, können Sie Step Functions die Anfrage automatisch genehmigen lassen.

Anwendungsfall #3: Fehlerbehandlung



`Retry`

In diesem Anwendungsfall fordert ein Kunde einen Benutzernamen an. Beim ersten Mal ist die Anfrage Ihres Kunden erfolglos. Mithilfe einer `Retry` Anweisung können Sie Step Functions veranlassen, die Anfrage Ihres Kunden erneut zu bearbeiten. Beim zweiten Mal ist die Anfrage Ihres Kunden erfolgreich.

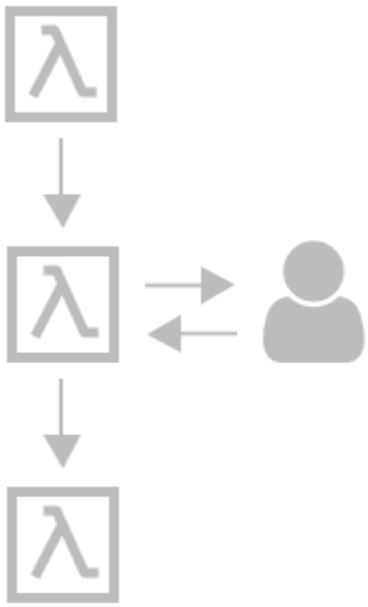
`Catch`

In einem ähnlichen Anwendungsfall fordert ein Kunde einen nicht verfügbaren Benutzernamen an. Mithilfe einer `Catch` Anweisung lassen Sie Step Functions einen verfügbaren Benutzernamen vorschlagen. Wenn Ihr Kunde den verfügbaren Benutzernamen verwendet, können Sie Step Functions zum nächsten Schritt in Ihrem Workflow übergehen lassen, dem Senden einer Bestätigungs-E-Mail. Wenn Ihr Kunde den verfügbaren Benutzernamen nicht verwendet, lassen Sie Step Functions zu einem anderen Schritt in Ihrem Workflow übergehen, nämlich den Anmeldevorgang erneut zu starten.

Detailliertere Beispiele `Retry` und `Catch` Aussagen finden Sie im Folgenden:

- Fehlerbehandlung in Step Functions

Anwendungsfall #4: Der Mensch ist auf dem Laufenden

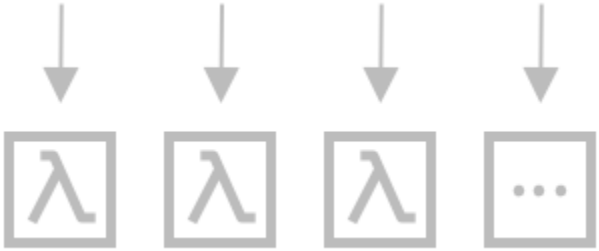


Mit einer Banking-App sendet einer Ihrer Kunden Geld an einen Freund. Ihr Kunde wartet auf eine Bestätigungs-E-Mail. Mit einem Callback und einem Task-Token weisen Step Functions Lambda an, das Geld Ihres Kunden zu senden, und melden sich, wenn der Freund Ihres Kunden es erhält. Nachdem Lambda gemeldet hat, dass der Freund Ihres Kunden das Geld erhalten hat, können Sie Step Functions veranlassen, mit dem nächsten Schritt in Ihrem Workflow fortzufahren, dem Kunden eine Bestätigungs-E-Mail zu senden.

Ein Beispielprojekt, das einen Rückruf mit einem Task-Token zeigt, finden Sie im Folgenden:

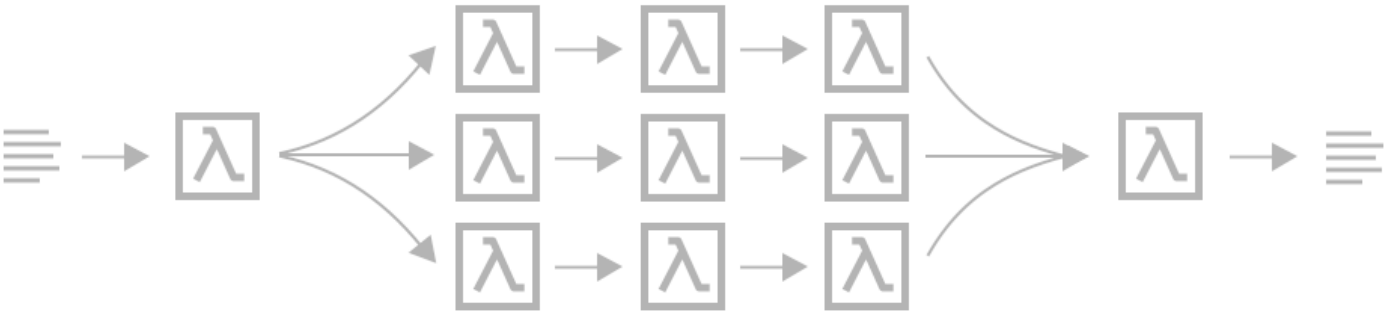
- Beispiel für ein Rückrufmuster (Amazon SQS, Amazon SNS, Lambda)

Anwendungsfall #5: Parallele Verarbeitung



Ein Kunde konvertiert eine Videodatei in fünf verschiedene Bildschirmauflösungen, sodass Zuschauer das Video auf mehreren Geräten ansehen können. Mithilfe eines `Parallel` Zustands gibt Step Functions die Videodatei ein, sodass Lambda sie gleichzeitig in die fünf Bildschirmauflösungen verarbeiten kann.

Anwendungsfall #6: Dynamische Parallelität



Ein Kunde bestellt drei Artikel, und Sie müssen jeden Artikel für die Lieferung vorbereiten. Sie überprüfen die Verfügbarkeit jedes Artikels, holen jeden Artikel ab und verpacken dann jeden Artikel für die Lieferung. Mithilfe eines `Map` Zustands lässt Step Functions Lambda jeden Artikel Ihres Kunden parallel verarbeiten. Sobald alle Artikel Ihres Kunden für den Versand verpackt sind, geht Step Functions zum nächsten Schritt in Ihrem Workflow über, der darin besteht, Ihrem Kunden eine Bestätigungs-E-Mail mit Informationen zur Sendungsverfolgung zu senden.

Ein Beispielprojekt, das dynamische Parallelität anhand eines `Map` Zustands zeigt, finden Sie im Folgenden:

- Dynamisches Verarbeiten von Daten mit einem Map-Status

Service-Integrationen

Step Functions lässt sich in mehrere AWS Dienste integrieren. Verwenden Sie die folgenden Dienstintegrationsmuster, um Step Functions mit diesen Diensten zu kombinieren:

Antwort anfordern (Standard)

- Rufen Sie einen Dienst auf und lassen Sie Step Functions zum nächsten Status übergehen, nachdem es eine HTTP-Antwort erhalten hat.

Führen Sie einen Job aus (.sync)

- Rufen Sie einen Service auf und lassen Sie Step Functions warten, bis ein Job abgeschlossen ist.

Warten Sie auf einen Rückruf mit einem Task-Token (. waitForTaskToken)

- Rufen Sie einen Dienst mit einem Task-Token auf und lassen Sie Step Functions warten, bis das Task-Token mit einem Callback zurückkehrt.

Die folgende Tabelle zeigt die verfügbaren Serviceintegrationen und Serviceintegrationsmuster für Step Functions.

Standard-Workflows und Express-Workflows unterstützen dieselben Integrationen, aber nicht dieselben Integrationsmuster.

- Die Unterstützung optimierter Integrationsmuster ist für jede Integration unterschiedlich.
- Express-Workflows unterstützen Run a Job (.sync) oder Wait for Callback () nicht. waitForTaskToken).
- Weitere Informationen finden Sie unter Standard- und Express-Workflows.

Standard Workflows

Unterstützte Serviceintegrationen

	Service	Request Response (Antwort anfordern)	Run a Job (Auftrag ausführen) (`.sync`)	Wait for Callback (Auf Rückruf warten) (`.waitForTaskToken`)
Optimierte Integrationen	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	✓		✓

	Service	Request Response (Antwort anfordern)	Run a Job (Auftrag ausführen) (`.sync`)	Wait for Callback (Auf Rückruf warten) (`.waitForTaskToken`)
	AWS Elemental MediaConvert	✓	✓	
	Amazon SageMaker	✓	✓	
	Amazon SNS	✓		✓
	Amazon SQS	✓		✓
	AWS Step Functions	✓	✓	✓
AWS SDK-Integrationen	Über zweihundert	✓		✓

Express Workflows

Unterstützte Serviceintegrationen

	Service	Request Response (Antwort anfordern)	Run a Job (Auftrag ausführen) (`.sync`)	Wait for Callback (Auf Rückruf warten) (`.waitForTaskToken`)
Optimiert	Amazon API Gateway	✓		
	Amazon Athena	✓		

	Service	Request Response (Antwort anfordern)	Run a Job (Auftrag ausführen) (`.sync`)	Wait for Callback (Auf Rückruf warten) (`.waitForTaskToken`)
Integrationen	AWS Batch	✓		
	Amazon Bedrock	✓		
	AWS CodeBuild	✓		
	Amazon DynamoDB	✓		
	Amazon ECS/Fargate	✓		
	Amazon EKS	✓		
	Amazon EMR	✓		
	Amazon EMR on EKS	✓		
	Amazon EMR Serverless	✓		
	Amazon EventBridge	✓		
	AWS Glue	✓		
	AWS Glue DataBrew	✓		
	AWS Lambda	✓		
	AWS Elemental MediaConvert	✓		
	Amazon SageMaker	✓		
	Amazon SNS	✓		
	Amazon SQS	✓		

	Service	Request Response (Antwort anfordern)	Run a Job (Auftrag ausführen) (`.sync`)	Wait for Callback (Auf Rückruf warten) (`.waitForTaskToken`)
	AWS Step Functions	✓		
AWS SDK-Integrationen	Über zweihundert	✓		

Unterstützte -Regionen

Die meisten AWS Regionen unterstützen Step Functions. Eine vollständige Liste der AWS Regionen, in denen Step Functions verfügbar ist, finden Sie AWS in der Regionentabelle.

Verwenden Sie Step Functions zum ersten Mal?

Wenn Sie Step Functions zum ersten Mal verwenden, helfen Ihnen die folgenden Themen dabei, die verschiedenen Aspekte der Arbeit mit Step Functions zu verstehen, einschließlich der Kombination von Step Functions mit anderen AWS Diensten:

- Tutorials für Step Functions
- Beispielprojekte für Step Functions
- AWS Step Functions Datenwissenschaft-SDK für Python

Erste Schritte mit AWS Step Functions

Step Functions ist ein serverloser Orchestrierungsdienst, mit dem Sie einen Anwendungsworkflow als eine Reihe von ereignisgesteuerten Schritten definieren können. Jeder Schritt im Workflow wird als Status bezeichnet. Am häufigsten verwenden Sie Status wie, Status der AufgabeChoice, und ParallelZuordnung, um Ihre Workflows zu definieren. Innerhalb von `Task` Bundesstaaten können Sie die AWS SDK-Integrationen verwenden, die Step Functions unterstützt, und mehrere AWS-Services in Ihren Workflows orchestrieren.

Themen

Die wichtigsten Konzepte

Bevor Sie mit den Tutorials beginnen, sollten Sie sich den Kontext der folgenden wichtigen Step Functions Functions-Begriffe ansehen.

Begriff	Beschreibung
Workflow	Eine Abfolge von Schritten, die häufig einen Geschäftsprozess widerspiegeln.
Zustände	Einzelne Schritte in Ihrer Zustandsmaschine, die Entscheidungen auf der Grundlage ihrer Eingaben treffen, anhand dieser Eingaben Aktionen ausführen und Ausgaben an andere Staaten weiterleiten können.

Begriff	Beschreibung
	Weitere Informationen finden Sie unter Zustände.
Workflow Studio	Ein visueller Workflow-Designer, der Ihnen hilft, Workflows schneller zu prototypisieren und zu erstellen. Weitere Informationen finden Sie unter AWS Step Functions Workflow-Studio.
State Machine (Zustandsautomat)	Ein anhand von JSON-Text definierter Workflow, der die einzelnen Status oder Schritte im Workflow zusammen mit Feldern wie `StartAtTimeoutSeconds` , und darstelltVersion. Weitere Informationen finden Sie unter Struktur der Zustandsmaschine.
Amazon States Language	Eine JSON-basierte, strukturierte Sprache, die zur Definition Ihrer Zustandsmaschinen verwendet wird. Mit ASL definieren Sie eine Sammlung von Status, die funktionieren können (`Task`Status), bestimmen, in welchen Status der nächste Status übergehen soll (`Choice`Status), und eine Ausführung mit einem Fehler beenden (`Fail`Status). Weitere Informationen finden Sie unter Amazon States Language.
Eingabe- und Ausgabekonfiguration	Staaten in einem Workflow erhalten JSON-Daten als Eingabe und geben JSON-Daten normalerweise als Ausgabe an den nächsten Status weiter. Step Functions bietet Filter zur Steuerung des Datenflusses zwischen Staaten. Weitere Informationen finden Sie unter Eingabe- und Ausgabeverarbeitung in Step Functions.
Service-Integration	Sie können AWS Service-API-Aktionen von Ihrem Workflow aus aufrufen. Weitere Informationen finden Sie unter Verwendung AWS Step Functions mit anderen Diensten.

Begriff	Beschreibung
Art der Servicein tegration	• AWS SDK-Integrationen — Standardmethode zum Aufrufen von über zweihundert AWS-Services und über neuntausend API-Aktionen direkt von Ihrer Zustandsmaschine aus. • Optimierte Integrationen — Maßgeschneiderte Integrationen, die das Aufrufen und den Datenaustausch mit bestimmten Diensten optimieren. Lambda Invoke konvertiert beispielsweise automatisch das `Payload` Feld der Antwort aus einer maskierten JSON-Zeichenfolge in ein JSON-Objekt.
Muster der Servicein tegration	Wenn Sie einen aufrufen AWS-Service, verwenden Sie eines der folgenden Serviceintegrationsmuster: • Antwort anfordern (Standard) — Rufen Sie einen Dienst auf und wechseln Sie sofort nach Erhalt einer HTTP-Antwort zum nächsten Status. • Job ausführen (.sync) — Rufen Sie einen Dienst auf und lassen Sie Step Functions warten, bis ein Job abgeschlossen ist. • Auf einen Callback mit einem Task-Token (.wait ForTask Token) warten — Rufen Sie einen Dienst mit einem Task-Token auf und lassen Sie Step Functions warten, bis das Task-Token mit einem Callback zurückkehrt.
Ausführung	State-Machine-Ausführungen sind Instanzen, in denen Sie Ihren Workflow ausführen, um Aufgaben auszuführen. Weitere Informationen finden Sie unter Ausführungen in Step Functions.

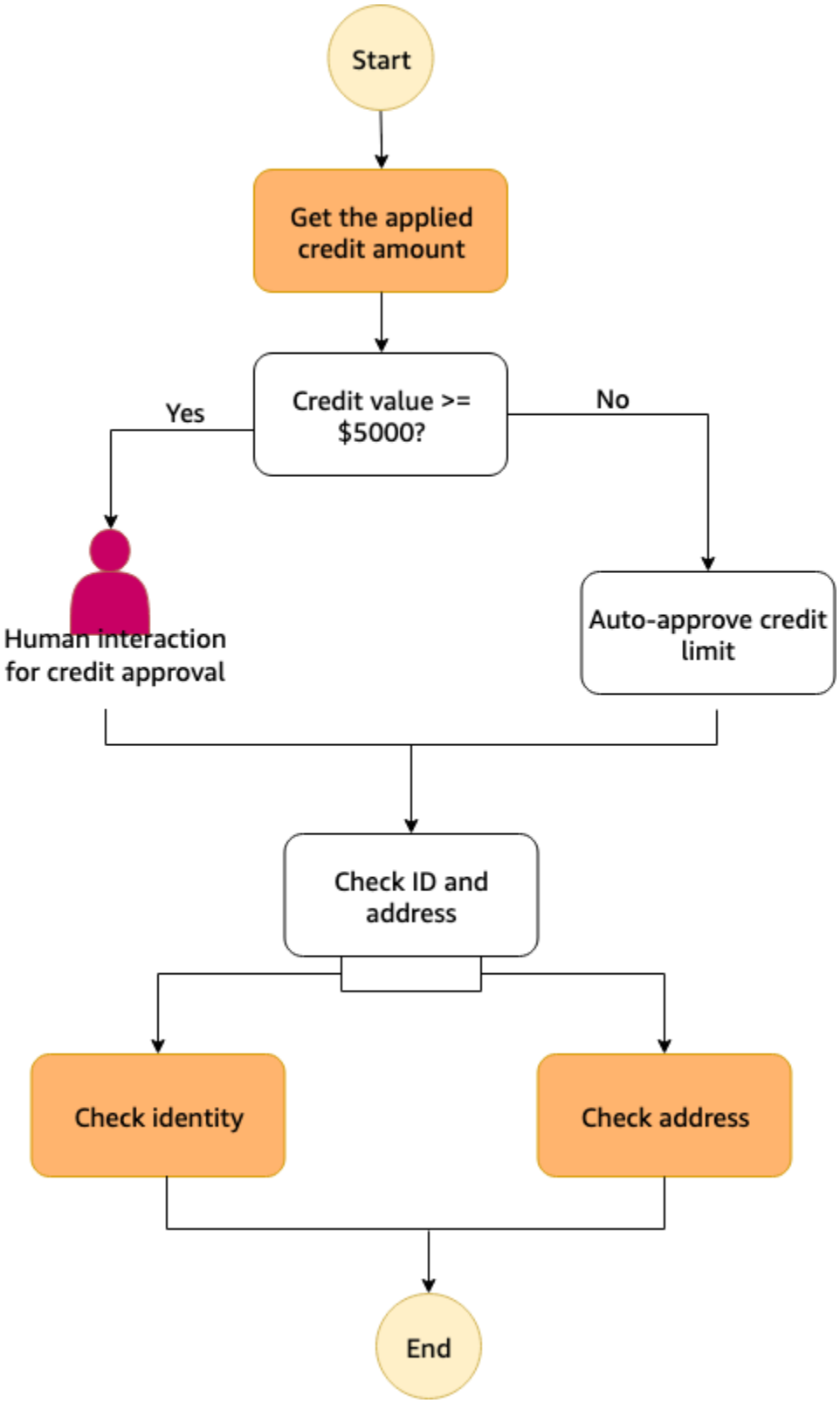
Tutorials in dieser Serie

Nach Abschluss dieser Tutorials verfügen Sie über einen Arbeitsablauf, der die Bearbeitung eines Kreditkartenantrags simuliert. Sie lernen, allgemeine Status zu verwenden und Ihren Arbeitsablauf in andere zu integrieren AWS-Services

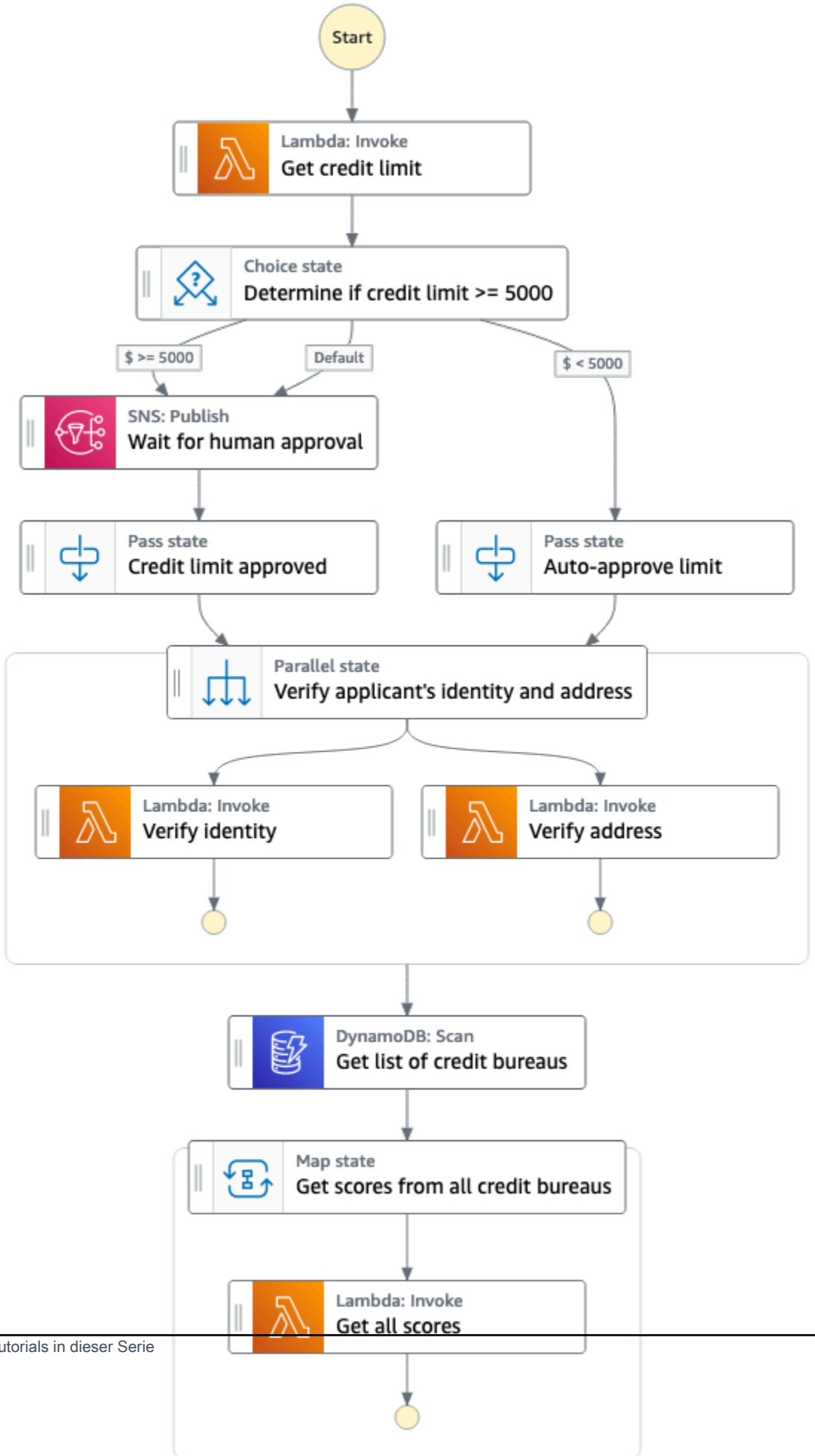
Mit Step Functions können viele Arten von Workflows erstellt werden, z. B. Datenverarbeitung, IT-Automatisierung, maschinelles Lernen und Medienkodierung.

Das folgende Flussdiagramm zeigt die Schritte, die ein Unternehmen zur Bearbeitung eines Kreditkartenantrags benötigt. Wenn der angeforderte Kreditbetrag unter 5000$ liegt, wird das

Kreditlimit automatisch genehmigt. Wenn die Anfrage das Limit überschreitet, fügt der Workflow einen Mitarbeiter hinzu, der die Identität des Antragstellers überprüft und die Kreditwürdigkeit überprüft.

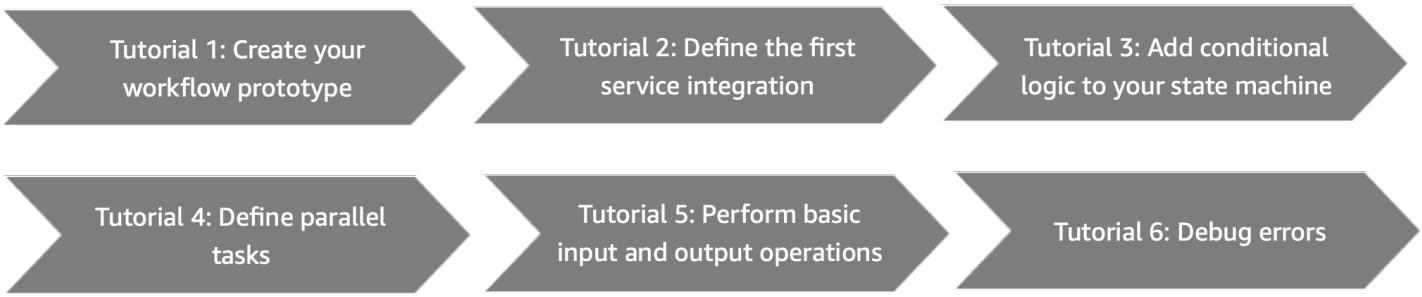


Das folgende Diagramm zeigt, wie die Geschäftsprozessschritte für Kreditanträge in einem Step Functions Functions-Workflow nach Bundesstaaten dargestellt werden.


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

In der folgenden Reihe von Tutorials werden Sie den Workflow für die Kreditkartenverarbeitung erstellen.

Wir empfehlen, diese Tutorials zu absolvieren, um die wichtigsten Funktionen von Step Functions kennenzulernen.



Bevor Sie beginnen, stellen Sie sicher, dass Sie die Voraussetzungen erfüllen.

Voraussetzungen für den Einstieg in AWS Step Functions

Führen Sie AWS Step Functions vor der ersten Verwendung die folgenden Aufgaben aus.

Melden Sie sich an für ein AWS-Konto

Wenn Sie noch keine haben AWS-Konto, führen Sie die folgenden Schritte aus, um eine zu erstellen.

Um sich für eine anzumelden AWS-Konto

1. Öffnen Sie https://portal.aws.amazon.com/billing/signup.
2. Folgen Sie den Online-Anweisungen.

   Bei der Anmeldung müssen Sie auch einen Telefonanruf entgegennehmen und einen Verifizierungscode über die Telefontasten eingeben.

   Wenn Sie sich für eine anmelden AWS-Konto, Root-Benutzer des AWS-Kontoswird eine erstellt. Der Root-Benutzer hat Zugriff auf alle AWS-Services und Ressourcen des Kontos. Aus Sicherheitsgründen sollten Sie einem Benutzer Administratorzugriff zuweisen und nur den Root-Benutzer verwenden, um Aufgaben auszuführen, für die Root-Benutzerzugriff erforderlich ist.

AWS sendet Ihnen nach Abschluss des Anmeldevorgangs eine Bestätigungs-E-Mail. Sie können jederzeit Ihre aktuelle Kontoaktivität anzeigen und Ihr Konto verwalten. Rufen Sie dazu https://aws.amazon.com/ auf und klicken Sie auf Mein Konto.

Erstellen Sie einen Benutzer mit Administratorzugriff

Nachdem Sie sich für einen angemeldet haben AWS-Konto, sichern Sie Ihren Root-Benutzer des AWS-Kontos AWS IAM Identity Center, aktivieren und erstellen Sie einen Administratorbenutzer, sodass Sie den Root-Benutzer nicht für alltägliche Aufgaben verwenden.

Sichern Sie Ihre Root-Benutzer des AWS-Kontos

1. Melden Sie sich AWS Management Consoleals Kontoinhaber an, indem Sie Root-Benutzer auswählen und Ihre AWS-Konto E-Mail-Adresse eingeben. Geben Sie auf der nächsten Seite Ihr Passwort ein.

   Hilfe bei der Anmeldung mit dem Root-Benutzer finden Sie unter Anmelden als Root-Benutzer im AWS-Anmeldung Benutzerhandbuch zu.
2. Aktivieren Sie die Multi-Faktor-Authentifizierung (MFA) für den Root-Benutzer.

   Anweisungen finden Sie unter Aktivieren eines virtuellen MFA-Geräts für Ihren AWS-Konto Root-Benutzer (Konsole) im IAM-Benutzerhandbuch.

Erstellen Sie einen Benutzer mit Administratorzugriff

1. Aktivieren Sie das IAM Identity Center.

   Anweisungen finden Sie unter Aktivieren AWS IAM Identity Center im AWS IAM Identity Center Benutzerhandbuch.
2. Gewähren Sie einem Benutzer in IAM Identity Center Administratorzugriff.

   Ein Tutorial zur Verwendung von IAM-Identity-Center-Verzeichnis als Identitätsquelle finden Sie unter Benutzerzugriff mit der Standardeinstellung konfigurieren IAM-Identity-Center-Verzeichnis im AWS IAM Identity Center Benutzerhandbuch.

Melden Sie sich als Benutzer mit Administratorzugriff an

- Um sich mit Ihrem IAM-Identity-Center-Benutzer anzumelden, verwenden Sie die Anmelde-URL, die an Ihre E-Mail-Adresse gesendet wurde, als Sie den IAM-Identity-Center-Benutzer erstellt haben.

  Hilfe bei der Anmeldung mit einem IAM Identity Center-Benutzer finden Sie im AWS-Anmeldung Benutzerhandbuch unter Anmeldung beim AWS Zugriffsportal.

Weisen Sie weiteren Benutzern Zugriff zu

1. Erstellen Sie in IAM Identity Center einen Berechtigungssatz, der der bewährten Methode zur Anwendung von Berechtigungen mit den geringsten Rechten folgt.

   Anweisungen finden Sie im Benutzerhandbuch unter Einen Berechtigungssatz erstellen.AWS IAM Identity Center
2. Weisen Sie Benutzer einer Gruppe zu und weisen Sie der Gruppe dann Single Sign-On-Zugriff zu.

   Anweisungen finden Sie im AWS IAM Identity Center Benutzerhandbuch unter Gruppen hinzufügen.

Tutorial 1: Erstellen Sie den Prototyp für Ihre Zustandsmaschine

In diesem Tutorial erstellen Sie den Prototyp für Ihren Workflow zur Kreditkartenverarbeitung mitDas Workflow-Studio von Step Functions. Sie wählen die erforderlichen API-Aktionen und -Status aus derAktionenundFließenTabs jeweils und verwenden Sie die Drag-and-Drop-Funktion von Workflow Studio, um den Workflow-Prototyp zu erstellen. In den nachfolgenden Tutorials erfahren Sie, wie Sie das konfigurierenAWS-Servicesund die Status der Step Functions, die Sie in diesem Workflow verwenden werden.

Um den Prototyp der Zustandsmaschine zu erstellen

1. Sie können auchStep Functions Functions-Konsoleund wähleZustandsmaschine erstellen.
2. In derWählen Sie eine VorlageDialogfeld, auswählenLeer.
3. Wählen Sie Select (Auswählen). Dadurch wird Workflow Studio geöffnet inEntwurfsmodus.
4. In Workflow Studio, von derAktionenTabulatortaste, ziehe undAWS LambdaAufrufenAPI-Aktion und setzen Sie sie in den leeren Zustand mit der BezeichnungZiehen Sie den ersten Status hierher. Sie können auch wie folgt konfiguriert werden:

   - In derKonfigurationRegisterkarte, fürName des Bundesstaates, eingeben**`Get credit limit`**.
5. Aus demFlussTabulatortaste, Drag-and-DropWahlStaat unter demSie können auch ein Kreditlimit festlegenBundesstaat. Sie können auch denWahlBundesstaat zu**`Credit applied >= 5000?`**.

6. Ziehen Sie die folgenden Staaten per Drag-and-Drop als Zweige derBeantragter Kredit **>=** 5000?Bundesstaat.

   a. Amazon SNS: Veröffentlichen— Aus demAktionenTabulatortaste, Drag-and-DropAmazon SNS: VeröffentlichenAPI-Aktion. Sie können auch den Status umbenennen**`Wait for human approval`**.

   b. PassBundesstaat — Aus demFlussTabulatortaste, Drag-and-DropPassierenBundesstaat. Sie können auch den Zweig umbenennen**`Auto-approve limit`**.

7. Ziehen Sie einPassStaat unter demWarte auf die Zustimmung des MenschenBundesstaat. Sie können auch umbenennenPassBundesstaat zu**`Credit limit approved`**.

8. Ziehen Sie einParallelStaat nach demWahlgeben Sie wie folgt an:

   a. Sie können auch denParallelStaat nach demKreditlimit genehmigtBundesstaat.

   b. Sie können auch denParallelStaat zu**`Verify applicant's identity and address`**.

   c. Unter den beiden Zweigen derParallelStatus, zwei per Drag-and-DropAWS LambdaAufrufenAPI-Aktionen.

   d. Benennen Sie diese Staaten um als**`Verify identity`**und**`Verify address`**jeweils.

   e. Wählen Sie dieLimit für automatische GenehmigungBundesstaat und fürNächstes Bundesland, wählenÜberprüfen Sie die Identität und Adresse des Antragstellers.

9. Ziehe eineDynamoDB-Scangeben Sie es an und legen Sie es unter dasÜberprüfen Sie die Identität und Adresse des AntragstellersBundesstaat. Sie können auch denDynamoDB-ScanBundesstaat zu**`Get list of credit bureaus`**.

10. Ziehen Sie einKarteBundesstaat nach demHolen Sie sich eine Liste der KreditauskunfteienBundesstaat. Konfigurieren Sie denKartegeben Sie wie folgt an:

    a. Sie können auch umbenennen**`Get scores from all credit bureaus`**.

    b. FürVerarbeitungsmodusSie können auch die Standardauswahl beibehaltenInline.

    c. Ziehen Sie und legen Sie einAWS LambdaAufrufenAPI-Aktion für den leeren Zustand mit der BezeichnungGeben Sie den Status hier ein.

    d. Sie können auch denAWS LambdaRufen Sie aufBundesstaat zu**`Get all scores`**.

11. Lassen Sie dieses Fenster geöffnet und fahren Sie mit dem nächsten Tutorial fort, um weitere Aktionen durchzuführen.

Nächste Schritte

Im nächsten Tutorial erfahren Sie, wie Sie die Lambda-Funktion integrieren, die vonSie können auch ein Kreditlimit festlegenBundesstaat.

Tutorial 2: Definieren Sie die erste Serviceintegration mithilfe einer Lambda-Funktion

In diesem Tutorial erfahren Sie, wie Sie die erste Serviceintegration für Ihren Workflow definieren. Sie verwenden den `Task` Status Get Credit Limit, um eine Lambda-Funktion aufzurufen. Innerhalb von `Task` Bundesstaaten können Sie die AWS SDK-Integrationen verwenden, die Step Functions unterstützt.

Um die erste Serviceintegration für Ihren Workflow zu definieren, erstellen Sie zunächst eine Lambda-Funktion. Aktualisieren Sie dann Ihren Workflow, um die Serviceintegration mit der Lambda-Funktion zu spezifizieren. Die in diesem Tutorial verwendete Lambda-Funktion gibt eine zufällig generierte Ganzzahl zurück, die das Kreditlimit darstellt, das ein Antragsteller beantragt hat.

Themen

Schritt 1: Lambda-Funktion erstellen und testen

Sie können Code für die Funktion im AWS Management Console oder in Ihrem bevorzugten Editor schreiben. In den folgenden Schritten erstellen Sie eine Node.js Lambda-Funktion mit dem Titel`RandomNumberforCredit`.

> **Important**
>
> Stellen Sie sicher, dass der Workflow-Prototyp, den Sie in Tutorial 1 erstellt haben, mit der Lambda-Funktion AWS-Region identisch ist, die Sie in diesem Tutorial erstellen werden.

1. Öffnen Sie in einem neuen Tab oder Fenster die Lambda-Konsole und erstellen Sie eine Node.js 16.x Lambda-Funktion mit dem Titel. **`RandomNumberforCredit`** Informationen zum Erstellen

einer Lambda-Funktion mithilfe der Konsole finden Sie unter Erstellen einer Lambda-Funktion in der Konsole im AWS Lambda Entwicklerhandbuch.

2. Wählen Sie auf der RandomNumberforCreditSeite index.mjs aus und ersetzen Sie den vorhandenen Code im Bereich Codequelle durch den folgenden Code.

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

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

};
```

3. Kopieren Sie im Abschnitt Funktionsübersicht den Amazon-Ressourcennamen der Lambda-Funktion und speichern Sie ihn in einer Textdatei. Sie benötigen die Funktion ARN, um die Serviceintegration für den Status Get Credit Limit anzugeben. Im Folgenden finden Sie ein Beispiel für einen ARN:

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

4. Wählen Sie Deploy und dann Test, um die Änderungen bereitzustellen und die Ausgabe der Lambda-Funktion zu sehen.

Schritt 2: Aktualisieren Sie den Workflow — konfigurieren Sie den Status „Kreditlimit abrufen“

In der Step Functions Functions-Konsole aktualisieren Sie Ihren Workflow, um die Serviceintegration mit der `RandomNumberforCredit` Lambda-Funktion zu spezifizieren, die Sie in Schritt 1 erstellt haben.

1. Öffnen Sie das Step Functions Functions-Konsolenfenster, das den Workflow-Prototyp enthält, den Sie in Tutorial 1 erstellt haben.
2. Wählen Sie den Status „Kreditlimit abrufen“ und gehen Sie auf der Registerkarte „Konfiguration“ wie folgt vor:

    a. Behalten Sie als Integrationstyp die Standardauswahl Optimiert bei.

Mithilfe von Step Functions können Sie andere Funktionen integrieren AWS-Services und diese in Ihren Workflows orchestrieren. Weitere Informationen zu Serviceintegrationen und ihren Typen finden Sie unter. Verwendung AWS Step Functions mit anderen Diensten

b. Wählen Sie als Funktionsname die RandomNumberforCreditLambda-Funktion aus der Dropdownliste aus.

c. Behalten Sie die Standardauswahl für die restlichen Elemente bei.

3. Lassen Sie dieses Fenster geöffnet und fahren Sie mit dem nächsten Tutorial fort, um weitere Aktionen durchzuführen.

> **Note**
>
> In diesem Tutorial haben Sie gelernt, wie Sie eine Lambda-Funktion innerhalb eines `Task` Zustands in Ihre Workflows integrieren können. Sie können auch andere unterstützte AWS SDK-Integrationen im `Task` Bundesstaat verwenden, indem Sie den Dienstnamen und den API-Aufruf angeben, wie in der folgenden Syntax dargestellt:
>
> ```
> arn:aws:states:::aws-sdk:serviceName:apiAction
> ```
>
> Weitere Informationen finden Sie unter Verwendung AWS Step Functions mit anderen Diensten.

Nächste Schritte

Im nächsten Tutorial implementieren Sie bedingte Logik in Ihrem Workflow. Bedingte Logik in Step Functions Functions-Zustandsmaschinen verhält sich ähnlich wie eine if-else-Anweisung in den meisten gängigen Programmiersprachen. Sie verwenden in Ihrem Workflow bedingte Logik, um den Ausführungspfad auf der Grundlage bedingter Informationen zu bestimmen.

Tutorial 3: Implementieren Sie eine If-else-Bedingung in Ihrem Workflow

Sie können if-else-Bedingungen in Ihren Workflows implementieren, indem Sie den `Choice` Status verwenden. Es bestimmt den Workflow-Ausführungspfad basierend darauf, ob eine angegebene Bedingung als wahr oder falsch bewertet wird.

In diesem Tutorial fügen Sie eine Bedingungslogik hinzu, um festzustellen, ob der von der in Tutorial 2 verwendeten `RandomNumberforCredit` Lambda-Funktion zurückgegebene Kreditbetrag einen bestimmten Schwellenwert überschreitet. Wenn der Betrag den Schwellenwert überschreitet, ist für die Genehmigung des Antrags eine menschliche Interaktion erforderlich. Andernfalls wird der Antrag automatisch genehmigt und es wird mit dem nächsten Schritt fortgefahren.

Sie ahmen den Schritt der menschlichen Interaktion nach, indem Sie die Workflow-Ausführung unterbrechen, bis ein Task-Token zurückgegeben wird. Dazu übergeben Sie ein Task-Token an die AWS SDK-Integration, die Sie in diesem Tutorial verwenden werden, nämlich Amazon Simple Notification Service. Die Workflow-Ausführung wird unterbrochen, bis das Task-Token mit einem `SendTaskSuccess`API-Aufruf zurückgesendet wird. Weitere Hinweise zur Verwendung von Task-Token finden Sie unterWarten auf einen Callback mit dem Aufgabentoken.

Da Sie die Schritte für die Genehmigung durch einen Menschen und die automatische Genehmigung bereits in Ihrem Workflow-Prototyp definiert haben, erstellen Sie in diesem Tutorial zunächst ein Amazon SNS-Thema, das das Callback-Token erhält. Anschließend erstellen Sie eine Lambda-Funktion, um die Callback-Funktionalität zu implementieren. Schließlich aktualisieren Sie Ihren Workflow-Prototyp, indem Sie die Details dieser AWS-Service Integrationen hinzufügen.

Themen

Schritt 1: Erstellen Sie ein Amazon SNS-Thema, das das Callback-Token empfängt

Um den Schritt der menschlichen Interaktion zu implementieren, veröffentlichen Sie in einem Amazon Simple Notification Service-Thema und übergeben das Callback-Task-Token an dieses Thema. Die Callback-Aufgabe unterbricht die Workflow-Ausführung, bis das Task-Token mit einer Payload zurückgegeben wird.

1. Öffnen Sie die Amazon SNS-Konsole und erstellen Sie einen Standard-Thementyp. Informationen zum Erstellen eines Themas finden Sie unter Erstellen eines Amazon SNS-Themas im Amazon Simple Notification Service Developer Guide.

2. Geben Sie den Themennamen als an**`TaskTokenTopic`**.
3. Achten Sie darauf, das Thema ARN zu kopieren und in einer Textdatei zu speichern. Sie benötigen das Thema ARN, wenn Sie die Dienstintegration für den Status Wait for human approval angeben. Im Folgenden finden Sie ein Beispielthema ARN:

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

4. Erstellen Sie ein E-Mail-Abonnement für das Thema und bestätigen Sie dann Ihr Abonnement. Informationen zum Abonnieren eines Themas finden Sie unter Erstellen eines Abonnements für das Thema im Amazon Simple Notification Service Developer Guide.

Schritt 2: Erstellen Sie eine Lambda-Funktion, um den Callback zu verarbeiten

Um die Callback-Funktionalität zu handhaben, definieren Sie eine Lambda-Funktion und fügen das Amazon SNS-Thema, das Sie in Schritt 1 erstellt haben, als Trigger für diese Funktion hinzu. Wenn Sie mit einem Task-Token im Amazon SNS-Thema veröffentlichen, wird die Lambda-Funktion mit der Payload der veröffentlichten Nachricht aufgerufen.

- Schritt 2.1: Erstellen Sie die Lambda-Funktion für den Callback
- Schritt 2.2: Hinzufügen des Amazon SNS-Themas als Trigger für die Lambda-Funktion
- Schritt 2.3: Bereitstellung der erforderlichen Berechtigungen für die IAM-Rolle der Lambda-Funktion

Schritt 2.1: Erstellen Sie die Lambda-Funktion für den Callback

In dieser Funktion bearbeiten Sie die Anfrage zur Genehmigung des Kreditlimits und geben das Ergebnis der Anfrage mit dem `SendTaskSuccess`API-Aufruf als erfolgreich zurück. Diese Lambda-Funktion gibt auch das Task-Token zurück, das sie vom Amazon SNS-Thema erhalten hat.

Der Einfachheit halber genehmigt die Lambda-Funktion, die für den Schritt der menschlichen Interaktion verwendet wird, automatisch jede Aufgabe und gibt das Task-Token mit einem `SendTaskSuccess` API-Aufruf zurück. Sie können die Lambda-Funktion benennen als**`callback-human-approval`**.

1. Öffnen Sie in einem neuen Tab oder Fenster die Lambda-Konsole und erstellen Sie eine Node.js 16.x Lambda-Funktion mit dem Titel. **`callback-human-approval`**

Informationen zum Erstellen einer Lambda-Funktion mithilfe der Konsole finden Sie im AWS LambdaEntwicklerhandbuch unter Erstellen einer Lambda-Funktion in der Konsole.

2. Ersetzen Sie auf der callback-human-approvalSeite den vorhandenen Code im Bereich Codequelle durch den folgenden Code.

```
// 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. Lassen Sie dieses Fenster geöffnet und führen Sie die Schritte im nächsten Abschnitt aus, um weitere Aktionen durchzuführen.

Schritt 2.2: Hinzufügen des Amazon SNS-Themas als Trigger für die Lambda-Funktion

Wenn Sie das Amazon SNS-Thema, das Sie in Schritt 1 dieses Tutorials erstellt haben, als Trigger für die Lambda-Funktion hinzufügen, die Sie in Schritt 2.1 dieses Tutorials erstellt haben, wird die Lambda-Funktion bei jeder Veröffentlichung zum Amazon SNS-Thema ausgelöst. Wenn Sie mit einem Task-Token im Amazon SNS-Thema veröffentlichen, wird die Lambda-Funktion mit der Payload der veröffentlichten Nachricht aufgerufen. Weitere Informationen zur Konfiguration von Triggern für Lambda-Funktionen finden Sie unter Konfiguration von Triggern im AWS LambdaDeveloper Guide.

1. Wählen Sie im Abschnitt Funktionsübersicht der `callback-human-approval` Lambda-Funktion die Option Trigger hinzufügen.
2. Wählen Sie aus der Dropdownliste der Trigger SNS als Trigger aus.
3. Geben Sie für das SNS-Thema den Namen des Amazon SNS-Themas ein, das Sie in Schritt 1 dieses Tutorials erstellt haben, und wählen Sie es aus der angezeigten Dropdown-Liste aus.
4. Wählen Sie Add (Hinzufügen) aus.
5. Lassen Sie dieses Fenster geöffnet und führen Sie die Schritte im nächsten Abschnitt aus, um weitere Aktionen durchzuführen.

Schritt 2.3: Bereitstellung der erforderlichen Berechtigungen für die IAM-Rolle der Lambda-Funktion

Sie müssen der `callback-human-approval` Lambda-Funktion die Berechtigungen für den Zugriff auf Step Functions gewähren, um das Task-Token zusammen mit dem `SendTaskSucess` API-Aufruf zurückzugeben.

1. Wählen Sie auf der callback-human-approvalSeite die Registerkarte Konfiguration und dann Berechtigungen aus.
2. Wählen Sie unter Ausführungsrolle den Rollennamen aus, um zur Rollenseite der AWS Identity and Access Management Konsole zu gelangen.
3. Um die erforderliche Berechtigung hinzuzufügen, wählen Sie Berechtigungen hinzufügen und dann Richtlinien anhängen.
4. Geben Sie in das Suchfeld den Text ein **`AWSStepFunctions`** und drücken Sie dann die Eingabetaste.

5. Wählen Sie AWSStepFunctionsFullAccessund scrollen Sie dann nach unten, um Richtlinien anhängen auszuwählen. Dadurch wird die Richtlinie hinzugefügt, die die erforderliche Berechtigung für die `callback-human-approval` Lambda-Funktionsrolle enthält.

Schritt 3: Aktualisieren Sie den Workflow — fügen Sie im Status Auswahl die Bedingungslogik if-else hinzu

Definieren Sie in der Step Functions-Konsole die Bedingungslogik für Ihren Workflow mithilfe des `Choice` Status. Wenn die von der `RandomNumberforCredit` Lambda-Funktion zurückgegebene Ausgabe weniger als 5000 beträgt, wird die angeforderte Gutschrift automatisch genehmigt. Wenn die zurückgegebene Leistung größer oder gleich 5000 ist, geht die Workflow-Ausführung zum Schritt der menschlichen Interaktion über, um das Kreditlimit zu genehmigen.

Im `Choice` Bundesstaat verwenden Sie einen Vergleichsoperator, um eine Eingabevariable mit einem bestimmten Wert zu vergleichen. Sie können die Eingabevariable als Ausführungseingabe angeben, während Sie eine State-Machine-Ausführung starten, oder die Ausgabe eines vorherigen Schritts als Eingabe für den aktuellen Schritt verwenden. Standardmäßig wird die Ausgabe eines Schritts in einer Variablen gespeichert, die aufgerufen wird`Payload`. Um den Wert der `Payload` Variablen für den Vergleich im `Choice` Status zu verwenden, verwenden Sie die $ Syntax, wie im folgenden Verfahren gezeigt.

Informationen darüber, wie Informationen von einem Zustand in einen anderen fließen und wie Eingabe und Ausgabe in Ihren Workflows spezifiziert werden, finden Sie unter Tutorial 7: Eingabe und Ausgabe konfigurieren undEingabe- und Ausgabeverarbeitung in Step Functions.

> **Note**
>
> Wenn der `Choice` Status eine in der Eingabe der State-Machine-Ausführung angegebene Eingabevariable für den Vergleich verwendet, verwenden Sie die `$.variable_name` Syntax, um den Vergleich durchzuführen. Verwenden Sie zum Beispiel die Syntax, um eine Variable zu vergleichen`$.myAge`. `myAge`

Da der `Choice` Bundesstaat in diesem Schritt Eingaben aus dem Status Get credit limit erhält, verwenden Sie die $ Syntax für die `Choice` Status-Konfiguration. Im Fehler beim Debuggen des ungültigen Pfades; Auswahlstatusfehler Abschnitt in Tutorial 8 erfahren Sie, wie sich das Ergebnis

der State-Machine-Ausführung unterscheidet, wenn Sie die `$.variable_name` `Choice` Syntax in der State-Konfiguration verwenden, um auf die Ausgabe aus einem vorherigen Schritt zu verweisen.

Um die If-else-Bedingungslogik mithilfe des Zustands hinzuzufügen **`Choice`**

1. Öffnen Sie das Step Functions-Konsolenfenster, das den Workflow-Prototyp enthält, in dem Sie erstellt habenTutorial 1: Erstellen Sie den Prototyp für Ihre Zustandsmaschine.
2. Wählen Sie den beantragten Kredit **>=** 5000? Status und geben Sie auf der Registerkarte Konfiguration die Bedingungslogik wie folgt an:
   a. Wählen Sie unter Auswahlregeln das Symbol Bearbeiten in der Kachel Regel #1 aus, um die First-Choice-Regel zu definieren.
   b. Wählen Sie Bedingungen hinzufügen aus.
   c. Geben Sie im Dialogfeld Bedingungen für Regel #1 für Variable den folgenden Text ein**$**.
   d. Für Operator ist die Auswahl kleiner als.
   e. Wählen Sie für Wert die Option Zahlenkonstante aus, und geben Sie dann **`5000`** in das Feld neben der Dropdownliste Wert ein.
   f. Wählen Sie Bedingungen speichern.
   g. Wählen Sie in der Dropdown-Liste Dann nächster Status ist: die Option Limit automatisch genehmigen aus.
   h. Wählen Sie Neue Auswahlregel hinzufügen aus, und definieren Sie dann die Regel der zweiten Wahl, wenn der Kreditbetrag größer oder gleich 5000 ist, indem Sie die Teilschritte 2.b bis 2.f wiederholen. Wählen Sie für Operator ist größer als oder gleich.
   i. Wählen Sie in der Dropdown-Liste Dann nächster Status ist: die Option Auf menschliche Genehmigung warten aus.
   j. Wählen Sie im Feld Standardregel das Symbol Bearbeiten aus, um die Standardauswahlregel zu definieren, und wählen Sie dann in der Dropdownliste Standardstatus die Option Auf Genehmigung durch einen Mitarbeiter warten aus. Sie definieren die Standardregel, um den nächsten Status anzugeben, zu dem der Übergang erfolgen soll, falls keine der Auswahlstatusbedingungen als wahr oder falsch bewertet wird.
3. Konfigurieren Sie den Status „Auf menschliche Genehmigung warten“ wie folgt:
   a. Geben Sie auf der Registerkarte Konfiguration unter Thema den Namen des Amazon SNS-Themas ein und wählen Sie den Namen aus TaskTokenTopic, der in der Dropdownliste angezeigt wird.

b. Wählen Sie für Nachricht die Option Nachricht eingeben aus der Dropdownliste aus. Im Feld Nachricht geben Sie die Nachricht an, die Sie zum Amazon SNS-Thema veröffentlichen möchten. In diesem Tutorial veröffentlichen Sie ein Task-Token als Nachricht.

   Mit einem Task-Token können Sie einen Step Functions-Workflow vom Typ Standard unterbrechen, bis ein externer Prozess abgeschlossen ist und das Task-Token zurückgegeben wird. Wenn Sie einen Task-Status als Callback-Aufgabe angeben, indem Sie das `.waitForTaskToken`Service-Integrationsmuster angeben, wird ein Task-Token generiert und in das Kontextobjekt eingefügt, wenn die Aufgabe gestartet wird. Das Kontextobjekt ist eine interne JSON-Struktur, die während einer Ausführung verfügbar ist und Informationen über Ihre State-Maschine und deren Ausführung enthält. Weitere Hinweise zu Kontextobjekten finden Sie unterContext-Objekt.

c. Geben Sie in das angezeigte Feld Folgendes als Nachricht ein:

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

d. Markieren Sie das Kontrollkästchen Auf Rückruf warten.

e. Wählen Sie im daraufhin angezeigten Dialogfeld die Option Fertig aus.

4. Lassen Sie dieses Fenster geöffnet und fahren Sie mit dem nächsten Tutorial fort, um weitere Aktionen zu erhalten.

Nächste Schritte

Im nächsten Tutorial erfahren Sie, wie Sie mehrere Aufgaben parallel ausführen.

Tutorial 4: Definieren Sie mehrere Aufgaben, die parallel ausgeführt werden sollen

Bisher haben Sie gelernt, Workflows sequentiell auszuführen. Sie können jedoch zwei oder mehr Schritte parallel ausführen, indem Sie den `Parallel` Status verwenden. Ein `Parallel` Zustand veranlasst den Interpreter, jeden Zweig gleichzeitig auszuführen.

Beide Zweige in einem `Parallel` Bundesstaat erhalten dieselbe Eingabe, aber jeder Zweig verarbeitet die für ihn spezifischen Inputteile. Step Functions wartet, bis jeder Zweig die Ausführung abgeschlossen hat, bevor er mit dem nächsten Schritt fortfährt.

In diesem Tutorial verwenden Sie den Status Parallel, um gleichzeitig die Identität und Adresse des Bewerbers zu überprüfen.

Themen

Schritt 1: Erstellen Sie die Lambda-Funktionen, um die erforderlichen Prüfungen durchzuführen

Dieser Arbeitsablauf für Kreditkartenanträge ruft zwei Lambda-Funktionen im Status Parallel auf, um die Identität und Adresse des Antragstellers zu überprüfen. Diese Prüfungen werden gleichzeitig im Status Parallel durchgeführt. Die State Machine schließt die Ausführung erst ab, nachdem beide parallelen Zweige die Ausführung abgeschlossen haben.

Um die Lambda-Funktionen Check-Identity und Check-Adresse zu erstellen

1. Öffnen Sie in einem neuen Tab oder Fenster die Lambda-Konsole und erstellen Sie zwei Node.js 16.x Lambda-Funktionen mit dem Titel und. `check-identity` `check-address` Informationen zum Erstellen einer Lambda-Funktion mithilfe der Konsole finden Sie im AWS LambdaEntwicklerhandbuch unter Erstellen einer Lambda-Funktion in der Konsole.
2. Öffnen Sie die Check-Identity-Funktionsseite und ersetzen Sie den vorhandenen Code im Quellbereich durch den folgenden Code:

```
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. Öffnen Sie die Funktionsseite „Adresse überprüfen“ und ersetzen Sie den vorhandenen Code im Quellbereich durch den folgenden Code:

```
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. Kopieren Sie für beide Lambda-Funktionen im Abschnitt Funktionsübersicht die jeweiligen Amazon Resource Names (ARN) und speichern Sie sie in einer Textdatei. Sie benötigen die Funktion ARNs, wenn Sie die Serviceintegration für die Identität und den Adressstatus des Bewerbers verifizieren angeben. Das Folgende ist ein Beispiel für einen ARN:

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

Schritt 2: Aktualisieren Sie den Workflow — Fügen Sie parallele Aufgaben hinzu, die ausgeführt werden sollen

In der Step Functions-Konsole aktualisieren Sie Ihren Workflow, um die Dienstintegration mit den Lambda-Funktionen Check-Identity und Checkaddress zu spezifizieren, die Sie in Schritt 1 erstellt haben.

Um parallele Aufgaben zum Workflow hinzuzufügen

1. Öffnen Sie das Step Functions-Konsolenfenster, das den Workflow-Prototyp enthält, in dem Sie erstellt habenTutorial 1: Erstellen Sie den Prototyp für Ihre Zustandsmaschine.
2. Wählen Sie den Status Identität überprüfen und gehen Sie auf der Registerkarte Konfiguration wie folgt vor:

   a. Behalten Sie für den Integrationstyp die Standardauswahl Optimiert bei.

   > **Note**
   >
   > Mithilfe von Step Functions können Sie andere Funktionen integrieren AWS-Services und diese in Ihren Workflows orchestrieren. Weitere Informationen zu Serviceintegrationen und ihren Typen finden Sie unter Verwendung AWS Step Functions mit anderen Diensten

b. Wählen Sie als Funktionsname die Lambda-Funktion check-identity aus der Dropdown-Liste aus.

c. Wählen Sie für Payload die Option Payload eingeben aus und ersetzen Sie dann die Beispiel-Payload durch die folgende Payload:

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

3. Wählen Sie den Status Adresse überprüfen und gehen Sie auf der Registerkarte Konfiguration wie folgt vor:

a. Behalten Sie für den Integrationstyp die Standardauswahl Optimiert bei.

b. Wählen Sie als Funktionsname die Lambda-Funktion mit der Prüfadresse aus der Dropdown-Liste aus.

c. Wählen Sie für Payload die Option Payload eingeben aus und ersetzen Sie dann die Beispiel-Payload durch die folgende Payload:

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

4. Wählen Sie Weiter aus.

Tutorial 5: Gleichzeitiges Iterieren über eine Sammlung von Elementen

Im vorherigen Tutorial haben Sie gelernt, wie Sie mithilfe des `Parallel` Status separate Schrittzweige parallel ausführen können. Mithilfe des `Map` Status können Sie für jedes Element in einem Datensatz eine Reihe von Workflow-Schritten ausführen. Die Iterationen des `Map` Staates laufen parallel, was eine schnelle Verarbeitung eines Datensatzes ermöglicht.

Indem Sie den `Map` Status in Ihre Workflows einbeziehen, können Sie Aufgaben wie die Datenverarbeitung mithilfe einer der beiden Methoden ausführenZustandsverarbeitungsmodi

zuordnen: Inline-Modus und Verteilter Modus. Um einen `Map` Status zu konfigurieren, definieren Sie einen`ItemProcessor`, der JSON-Objekte enthält, die den `Map` Zustandsverarbeitungsmodus und seine Definition angeben. In diesem Tutorial führen Sie den `Map` Status im Standard-Inline-Modus aus, der bis zu 40 gleichzeitige Iterationen unterstützt. Wenn Sie den `Map` Status im verteilten Modus ausführen, unterstützt er bis zu 10.000 parallel Ausführungen untergeordneter Workflows.

Wenn Ihre Workflow-Ausführung in den `Map` Status übergeht, iteriert sie über ein JSON-Array, das in der Statuseingabe angegeben ist. Für jedes Array-Element wird die entsprechende Iteration im Kontext des Workflows ausgeführt, der den Status enthält. `Map` Wenn alle Iterationen abgeschlossen sind, gibt der `Map` Status ein Array zurück, das die Ausgabe für jedes Element enthält, das von der verarbeitet wurde. `ItemProcessor`

In diesem Tutorial erfahren Sie, wie Sie den `Map` Status im Inline-Modus verwenden, um die Kreditwürdigkeit eines Bewerbers abzurufen, indem Sie eine Reihe von Auskunfteien durchlaufen. Dazu rufen Sie zunächst die Namen aller Auskunfteien ab, die in einer Amazon DynamoDB-Tabelle gespeichert sind, und verwenden dann den `Map` Status, um die Auskunfteiliste zu durchsuchen, um die Kreditwürdigkeit des Antragstellers abzurufen, die von jeder dieser Büros gemeldet wurden.

Themen

- Schritt 1: Erstellen Sie eine DynamoDB-Tabelle, um die Namen aller Auskunfteien zu speichern
- Schritt 2: State-Machine aktualisieren — Ergebnisse aus der DynamoDB-Tabelle abrufen
- Schritt 3: Erstellen Sie eine Lambda-Funktion, die die Kreditwürdigkeit aller Auskunfteien zurückgibt
- Schritt 4: Aktualisieren Sie die Zustandsmaschine — fügen Sie einen Kartenstatus hinzu, um iterativ Kredit-Scores abzurufen

Schritt 1: Erstellen Sie eine DynamoDB-Tabelle, um die Namen aller Auskunfteien zu speichern

In diesem Schritt erstellen Sie **`GetCreditBureau`** mithilfe der DynamoDB-Konsole eine benannte Tabelle. Die Tabelle verwendet das Zeichenkettenattribut Name als Partitionsschlüssel. In dieser Tabelle speichern Sie die Namen aller Auskunfteien, von denen Sie die Kreditwürdigkeit des Bewerbers abrufen möchten.

1. Melden Sie sich bei der DynamoDB-Konsole an AWS Management Console und öffnen Sie sie unter https://console.aws.amazon.com/dynamodb/.
2. Wählen Sie im Navigationsbereich der Konsole Tabellen und dann Tabelle erstellen aus.

3. Geben Sie die Tabellendetails wie folgt ein:

   a. Geben Sie für Table name (Tabellenname) den Namen **`GetCreditBureau`** ein.

   b. Geben Sie für den Partitionsschlüssel **`Name`** ein.

   c. Behalten Sie die Standardauswahl bei und wählen Sie Tabelle erstellen aus.

4. Nachdem Ihre Tabelle erstellt wurde, wählen Sie in der Tabellenliste die GetCreditBureauTabelle aus.
5. Wählen Sie Aktionen und anschließend Element erstellen aus.
6. Geben Sie unter Wert den Namen eines Kreditbüros ein. Zum Beispiel **`CredTrack`**.
7. Wählen Sie Create item (Element erstellen) aus.
8. Wiederholen Sie diesen Vorgang und erstellen Sie Artikel für Namen anderer Auskunfteien. Beispiel: **`KapFinn`** und **`CapTrust`**.

Schritt 2: State-Machine aktualisieren — Ergebnisse aus der DynamoDB-Tabelle abrufen

In der Step Functions Functions-Konsole fügen Sie einen `Task` Status hinzu und verwenden die AWSSDK-Integration, um die Namen von Auskunfteien aus der DynamoDB-Tabelle abzurufen, die Sie in Schritt 1 erstellt haben. Sie verwenden die Ausgabe dieses Schritts als Eingabe für den `Map` Status, den Sie später in Ihrem Workflow in diesem Tutorial hinzufügen werden.

1. Öffnen Sie die CreditCardWorkflowZustandsmaschine, um sie zu aktualisieren.
2. Wählen Sie die Option Liste des Bundesstaates der Kreditauskunfteien abrufen aus.
3. Geben Sie für API-Parameter den Wert für den Tabellennamen als **`GetCreditBureau`** an.

Schritt 3: Erstellen Sie eine Lambda-Funktion, die die Kreditwürdigkeit aller Auskunfteien zurückgibt

In diesem Schritt erstellen Sie eine Lambda-Funktion, die die Namen aller Auskunfteien als Eingabe empfängt und die Kreditwürdigkeit des Antragstellers für jede dieser Auskunfteien zurückgibt. Diese Lambda-Funktion wird von dem `Map` Status aus aufgerufen, den Sie in Schritt 4 dieses Tutorials zu Ihrem Workflow hinzufügen.

1. Erstellen Sie eine Node.js 16.x Lambda-Funktion und benennen Sie sie. **`get-credit-score`**

2. Fügen Sie auf der Seite mit dem Titel get-credit-scoreden folgenden Code in den Bereich Codequelle ein.

```
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. Stellen Sie die Lambda-Funktion bereit.

Schritt 4: Aktualisieren Sie die Zustandsmaschine — fügen Sie einen Kartenstatus hinzu, um iterativ Kredit-Scores abzurufen

In der Step Functions Functions-Konsole fügen Sie einen `Map` Bundesstaat hinzu, der die get-credit-scoreLambda-Funktion aufruft, um die Kreditwürdigkeit des Antragstellers für alle Auskunfteien zu überprüfen, die vom Status „Liste der Kreditbüros abrufen“ zurückgegeben wurden.

1. Öffnen Sie die CreditCardWorkflowZustandsmaschine, um sie zu aktualisieren.
2. Wählen Sie den Status Punkte aus allen Auskunfteien abrufen.
3. Wählen Sie auf der Registerkarte Konfiguration die Option Geben Sie einen Pfad zum Artikelarray an und geben Sie dann die Eingabetaste ein**`$.Items`**.
4. Wählen Sie Alle Punktzahlen abrufen, Schritt innerhalb des `Map` Bundesstaats aus.
5. Vergewissern Sie sich, dass auf der Registerkarte Konfiguration als Integrationstyp die Option Optimiert ausgewählt ist.
6. Geben Sie als Funktionsname den Namen der get-credit-scoreLambda-Funktion ein und wählen Sie ihn aus der angezeigten Dropdownliste aus.
7. Wählen Sie für Payload die Option No payload aus.

Tutorial 6: Speichern Sie den Workflow und führen Sie die Zustandsmaschine aus

Nachdem Sie nun die Ressourcen aller, die AWS-Services Sie im Workflow-Prototyp verwenden, konfiguriert haben, können Sie ihn als Step Functions Functions-Zustandsmaschine speichern und mit der Ausführung beginnen.

Themen

Schritt 1: Überprüfen Sie die automatisch generierte Zustandsmaschine und speichern Sie die Zustandsmaschine

Wenn Sie Status von der Registerkarte Flow auf die Arbeitsfläche in Workflow Studio ziehen und dort ablegen, um den Workflow-Prototyp zu erstellen, erstellt Step Functions automatisch die Amazon States Language (ASL-) Definition Ihres Workflows in Echtzeit. Sie können diese Definition nach Bedarf in der bearbeiten. Code-Editor

Um die ASL-Definition zu überprüfen und den Zustandsmaschine zu speichern

1. (Optional) Wählen Sie Definition aufInspector, um die Definition der Zustandsmaschine Amazon States Language (ASL) anzuzeigen, die automatisch auf der Grundlage Ihrer Auswahl auf den Registerkarten Aktionen und Flow sowie im Inspektorfenster generiert wird.

   > **Tip**
   >
   > Um die Definition zu bearbeiten, können Sie den Code-Editor öffnen, indem Sie oben auf der Seite „Code“ wählen. Fahren Sie für dieses Tutorial mit der automatisch generierten Definition fort.

2. Geben Sie einen Namen für Ihre Zustandsmaschine an. Wählen Sie dazu das Bearbeitungssymbol neben dem Standardnamen der Zustandsmaschine von MyStateMachine.

Geben Sie dann unter State-Machine-Konfiguration einen Namen in das Feld State-Machine-Name ein.

Geben Sie für dieses Tutorial den Namen **`CreditCardWorkflow`** ein.

3. (Optional) Geben Sie unter State-Machine-Konfiguration weitere Workflow-Einstellungen an, z. B. den Zustandsmaschinentyp und seine Ausführungsrolle.

   Behalten Sie für dieses Tutorial alle Standardauswahlen in den State-Machine-Einstellungen bei.

   > **Note**
   >
   > (Optional) Step Functions erstellt automatisch eine Ausführungsrolle für die Zustandsmaschine mit den geringsten Rechten, die zum Aufrufen der `RandomNumberforCredit` Lambda-Funktion und zum Veröffentlichen im Amazon SNS SNS-Thema erforderlich sind.

   Wenn Sie zuvor eine IAM-Rolle mit den richtigen Berechtigungen für Ihre Zustandsmaschine erstellt haben und diese verwenden möchten, wählen Sie unter Berechtigungen die Option Vorhandene Rolle auswählen und dann eine Rolle aus der Liste aus. Oder wählen Sie Einen Rollen-ARN eingeben aus und geben Sie dann einen ARN für diese IAM-Rolle ein.

4. Wählen Sie im Dialogfeld „Rollenerstellung bestätigen“ die Option Bestätigen aus, um fortzufahren.

   Sie können auch Rolleneinstellungen anzeigen wählen, um zur State-Machine-Konfiguration zurückzukehren.

   > **Note**
   >
   > Wenn Sie die von Step Functions erstellte IAM-Rolle löschen, kann Step Functions sie später nicht mehr neu erstellen. Ebenso kann Step Functions ihre ursprünglichen Einstellungen später nicht wiederherstellen, wenn Sie die Rolle ändern (z. B. indem Sie Step Functions aus den Principals in der IAM-Richtlinie entfernen).

Schritt 2: Fügen Sie die verbleibenden IAM-Richtlinien hinzu

Da Step Functions die Berechtigungen zum Aufrufen der im `Parallel` Status verwendeten Lambda-Funktionen nicht automatisch generiert, müssen Sie die erforderliche Richtlinie hinzufügen.

Um die verbleibende Richtlinie hinzuzufügen

1. Wählen Sie auf der CreditCardWorkflowSeite die IAM-Rolle für Ihren State Machine aus, um zur IAM-Konsole zu navigieren. Auf dieser Seite fügen Sie die erforderlichen Berechtigungen für die übrigen Lambda-Funktionen hinzu.
2. Wählen Sie Berechtigungen hinzufügen aus und wählen Sie dann Richtlinien direkt anhängen aus.
3. Geben Sie in das Suchfeld ein **`AWSLambdaRole`** und drücken Sie dann die Eingabetaste.
4. Wählen Sie Richtlinien anhängen AWSLambdaRoleund anschließend aus. Diese Richtlinie wurde jetzt der Ausführungsrolle Ihres State Machine hinzugefügt. Mit dieser Richtlinie können Sie jede Lambda-Funktion in Ihrer Zustandsmaschine aufrufen.

Schritt 3: Führen Sie die Zustandsmaschine aus

State-Machine-Ausführungen sind Instanzen, in denen Sie Ihren Workflow ausführen, um Aufgaben auszuführen.

Um die Zustandsmaschine auszuführen

1. Wählen Sie auf der CreditCardWorkflowSeite Ausführung starten aus.

   Das Dialogfeld Ausführung starten wird angezeigt.
2. Gehen Sie im Dialogfeld Ausführung starten wie folgt vor:

   a. (Optional) Um Ihre Ausführung zu identifizieren, können Sie im Feld Name einen Namen dafür angeben. Standardmäßig generiert Step Functions automatisch einen eindeutigen Ausführungsnamen.

   > **Note**
   >
   > Mit Step Functions können Sie Namen für Zustandsmaschinen, Ausführungen, Aktivitäten und Labels erstellen, die Nicht-ASCII-Zeichen enthalten. Diese Nicht-ASCII-Namen funktionieren nicht mit Amazon. CloudWatch Um sicherzustellen, dass

Sie CloudWatch Messwerte verfolgen können, wählen Sie einen Namen, der nur ASCII-Zeichen verwendet.

> **Note**
>
> Sie müssen keine Eingaben machen, um diese Zustandsmaschine auszuführen. Sie können jedoch bei Bedarf im Eingabebereich des Dialogfelds Ausführung starten für andere Zustandsmaschinen eine Ausführungseingabe angeben. Ein Beispiel für die Bereitstellung von Ausführungseingaben für eine Zustandsmaschine finden Sie unter Schritt 4: Starten einer neuen Ausführung des Tutorials Learn to use the AWS Step Functions Workflow Studio.

b. Wählen Sie Start execution (Ausführung starten) aus.

3. Die Step Functions Functions-Konsole leitet Sie zu einer Seite weiter, die mit Ihrer Ausführungs-ID betitelt ist. Diese Seite wird als Seite mit den Ausführungsdetails bezeichnet. Auf dieser Seite können Sie die Ausführungsergebnisse im Verlauf der Ausführung oder nach deren Abschluss überprüfen.

   Um die Ausführungsergebnisse zu überprüfen, wählen Sie in der Diagrammansicht einzelne Status aus und wählen Sie dann die einzelnen Registerkarten im Schrittdetails Bereich, um die Details der einzelnen Status, einschließlich Eingabe, Ausgabe und Definition, anzuzeigen. Einzelheiten zu den Ausführungsinformationen, die Sie auf der Seite mit den Ausführungsdetails einsehen können, finden Sie unterSeite mit Ausführungsdetails – Schnittstellenübersicht.

Tutorial 7: Eingabe und Ausgabe konfigurieren

Eine Step Functions-Ausführung erhält einen JSON-Text als Eingabe und übergibt diese Eingabe an den ersten Status im Workflow. Einzelne Status in einem Workflow erhalten JSON-Daten als Eingabe und geben normalerweise JSON-Daten als Ausgabe an den nächsten Status weiter. Standardmäßig werden Daten im Workflow von einem Status zum nächsten Status übertragen, es sei denn, Sie haben die Eingabe und/oder Ausgabe für einen oder mehrere Status im Workflow konfiguriert. Zu verstehen, wie die Informationen von einem Staat zum anderen fließen, und zu lernen, wie diese Daten gefiltert und bearbeitet werden können, ist der Schlüssel zur effektiven Gestaltung und Implementierung von Workflows in Step Functions.

Step Functions bietet mehrere Filter zur Steuerung des Eingabe- und Ausgabedatenflusses zwischen den Zuständen. Die folgenden Filter stehen für die Verwendung in Ihren Workflows zur Verfügung:

> **Note**
>
> Je nach Anwendungsfall müssen Sie möglicherweise nicht alle diese Filter in Ihren Workflows anwenden.

`InputPath`

Wählt aus, WELCHER Teil der gesamten Eingabe-Payload als Eingabe für eine Aufgabe verwendet werden soll. Wenn Sie dieses Feld angeben, wendet Step Functions dieses Feld zuerst an.

`Parameter`

Gibt an, WIE die Eingabe aussehen soll, bevor die Aufgabe aufgerufen wird. Mit dem `Parameters` Feld können Sie eine Sammlung von Schlüssel-Wert-Paaren erstellen, die als Eingabe an eine AWS-ServiceIntegration, z. B. eine AWS Lambda Funktion, übergeben werden. Diese Werte können statisch sein oder dynamisch aus der Statuseingabe oder dem Workflow-Kontextobjekt ausgewählt werden.

`ResultSelector`

Legt fest, WAS aus der Ausgabe einer Aufgabe ausgewählt werden soll. Mit dem `ResultSelector` Feld können Sie eine Sammlung von Schlüssel-Wert-Paaren erstellen, die das Ergebnis eines Zustands ersetzen, und diese Sammlung an diese übergeben. `ResultPath`

`ResultPath`

Legt fest, WO die Ausgabe einer Aufgabe abgelegt werden soll. Verwenden Sie den`ResultPath`, um festzustellen, ob die Ausgabe eines Zustands eine Kopie seiner Eingabe, des Ergebnisses, das er erzeugt, oder eine Kombination aus beidem ist.

`OutputPath`

Legt fest, WAS an den nächsten Status gesendet werden soll. Mit `OutputPath` können Sie unerwünschte Informationen herausfiltern und nur den Teil der JSON-Daten weitergeben, der Ihnen wichtig ist.

> **Tip**
>
> Die `Parameters` `ResultSelector` AND-Filter konstruieren JSON, wohingegen die `InputPath` und `OutputPath` -Filter bestimmte Knoten innerhalb eines JSON-Datenobjekts filtern und der `ResultPath` Filter erstellt ein Feld, unter dem die Ausgabe hinzugefügt werden kann.

In diesem Tutorial lernen Sie, wie Sie die folgenden Aufgaben ausführen:

- Wählen Sie mithilfe des InputPath Filters bestimmte Teile der Roheingabe aus
- Manipulieren Sie die ausgewählte Eingabe mithilfe des Parameter-Filters
- Konfiguration der Ausgabe mithilfe der OutputPath Filter ResultSelectorResultPath,, und

Weitere Informationen zur Konfiguration von Eingabe und Ausgabe in Ihren Workflows finden Sie unterEingabe- und Ausgabeverarbeitung in Step Functions.

Wählen Sie mithilfe des InputPath Filters bestimmte Teile der Roheingabe aus

Verwenden Sie den `InputPath` Filter, um einen bestimmten Teil der Eingabe-Payload auszuwählen.

Wenn Sie keinen Wert angeben`InputPath`, wird der Standardwert auf eingestellt$, was dazu führt, dass sich die Aufgabe des Status auf die gesamte Roheingabe bezieht und nicht auf einen bestimmten Teil.

Gehen Sie wie folgt vor, um zu erfahren, wie Sie den InputPath Filter verwenden:

- Schritt 1: Erstellen Sie eine Zustandsmaschine
- Schritt 2: Führen Sie die Zustandsmaschine aus
- Schritt 3: Verwenden Sie den InputPath Filter, um bestimmte Teile einer Ausführungseingabe auszuwählen

Schritt 1: Erstellen Sie eine Zustandsmaschine

> **Important**
>
> Stellen Sie sicher, dass sich Ihr State Machine unter demselben AWS Konto und derselben Region befindet wie die zuvor erstellte Lambda-Funktion.

1. Verwenden Sie das `Parallel` State-Beispiel, das Sie in Tutorial 4 kennengelernt haben, um eine neue Zustandsmaschine zu erstellen. Stellen Sie sicher, dass Ihr Workflow-Prototyp dem folgenden Prototyp ähnelt.
2. Konfigurieren Sie die Integrationen für die Funktionen `check-identity` und `check-address` Lambda. Hinweise zum Erstellen der Lambda-Funktionen und deren Verwendung in Ihrer Zustandsmaschine finden Sie unter Schritt 1: Erstellen Sie die Lambda-Funktionen, um die erforderlichen Prüfungen durchzuführen undSchritt 2: Aktualisieren Sie den Workflow — Fügen Sie parallele Aufgaben hinzu, die ausgeführt werden sollen.
3. Stellen Sie sicher, dass Sie für Payload die Standardauswahl „Statuseingabe als Nutzlast verwenden“ beibehalten.
4. Wählen Sie Weiter und führen Sie dann die Schritte 1 bis 3 in Tutorial 5 Schritt 1: Speichern Sie die Zustandsmaschine aus, um eine neue Zustandsmaschine zu erstellen. Geben Sie für dieses Tutorial Ihrem State Machine einen Namen**`WorkflowInputOutput`**.

Schritt 2: Führen Sie die Zustandsmaschine aus

1. Wählen Sie auf der WorkflowInputOutputSeite Ausführung starten aus.
2. (Optional) Um Ihre Ausführung zu identifizieren, können Sie im Feld Name einen Namen dafür angeben. Standardmäßig generiert Step Functions automatisch einen eindeutigen Ausführungsnamen.

   > **Note**
   >
   > Mit Step Functions können Sie Namen für Zustandsmaschinen, Ausführungen, Aktivitäten und Labels erstellen, die Nicht-ASCII-Zeichen enthalten. Diese Nicht-ASCII-Namen funktionieren nicht mit Amazon. CloudWatch Um sicherzustellen, dass Sie CloudWatch Messwerte verfolgen können, wählen Sie einen Namen, der nur ASCII-Zeichen verwendet.

3. Fügen Sie im Eingabebereich die folgenden JSON-Daten als Ausführungseingabe hinzu.

```
{
  "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. Wählen Sie Start execution (Ausführung starten) aus.
5. Die Ausführung der Zustandsmaschine führt zu einem Fehler, da Sie nicht angegeben haben, welche Teile der Ausführungseingabe die Funktionen `check-identity` und `check-address` Lambda-Funktionen verwenden müssen, um die erforderliche Identitäts- und Adressprüfung durchzuführen.
6. Fahren Sie mit Schritt 3 dieses Tutorials fort, um den Fehler zu beheben.

Schritt 3: Verwenden Sie den **`InputPath`** Filter, um bestimmte Teile einer Ausführungseingabe auszuwählen

1. Wählen Sie auf der Seite mit den Ausführungsdetails die Option Zustandsmaschine bearbeiten aus.
2. Um die Identität des Antragstellers zu überprüfen, wie in der Ausführungseingabe unter angegebenSchritt 2: Führen Sie die Zustandsmaschine aus, bearbeiten Sie die Aufgabendefinition „Identität verifizieren“ wie folgt:

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

Folglich stehen die folgenden JSON-Daten als Eingabe für die `check-identity` Funktion zur Verfügung.

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

3. Um die in der Ausführungseingabe angegebene Adresse des Bewerbers zu überprüfen, bearbeiten Sie die `Verify address` Aufgabendefinition wie folgt:

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

```
  }
  ...
```

Folglich stehen die folgenden JSON-Daten als Eingabe für die `check-address` Funktion zur Verfügung.

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

4. Wählen Sie Start execution (Ausführung starten) aus. Die Ausführung der Zustandsmaschine wird jetzt erfolgreich abgeschlossen.

Manipulieren Sie die ausgewählte Eingabe mithilfe des Parameter-Filters

Der `InputPath` Filter hilft Ihnen zwar dabei, die von Ihnen bereitgestellte rohe JSON-Eingabe zu begrenzen, aber mithilfe des `Parameters` Filters können Sie eine Sammlung von Schlüssel-Wert-Paaren als Eingabe übergeben. Bei diesen Schlüssel-Wert-Paaren kann es sich entweder um statische Werte handeln, die Sie in Ihrer State-Machine-Definition definieren, oder um Werte, die aus der Roheingabe mithilfe von ausgewählt wurden. `InputPath`

In Ihren Workflows `Parameters` werden sie danach angewendet`InputPath`. `Parameters`helfen Ihnen bei der Angabe, wie die zugrunde liegende Aufgabe ihre eingegebene Payload akzeptiert. Wenn die `check-address` Lambda-Funktion beispielsweise anstelle der JSON-Daten einen Zeichenfolgenparameter als Eingabe akzeptiert, können Sie den `Parameters` Filter verwenden, um die Eingabe zu transformieren.

Im folgenden Beispiel empfängt der `Parameters` Filter die Eingabe, die Sie mithilfe von `InputPath` in ausgewählt haben, Schritt 3: Verwenden Sie den `InputPath` Filter, um bestimmte Teile einer Ausführungseingabe auszuwählen und wendet die intrinsische Funktion `States.Format` auf die Eingabeelemente an, um eine Zeichenfolge mit dem Namen zu erstellen. `addressString` Intrinsische Funktionen helfen Ihnen dabei, grundlegende Datenverarbeitungsvorgänge für eine bestimmte Eingabe durchzuführen. Weitere Informationen finden Sie unter Intrinsische Funktionen.

```
"Parameters": {
```

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

Folglich wird die folgende Zeichenfolge erstellt und der `check-address` Lambda-Funktion als Eingabe zur Verfügung gestellt.

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

Konfiguration der Ausgabe mithilfe der OutputPath Filter ResultSelectorResultPath,, und

Wenn die `check-address` Lambda-Funktion in der WorkflowInputOutputState Machine aufgerufen wird, gibt die Funktion nach Durchführung der Adressüberprüfung eine Ausgabe-Payload zurück. Wählen Sie auf der Seite „Ausführungsdetails“ den Schritt „Adresse überprüfen“ und sehen Sie sich die Ausgabe-Payload innerhalb des Aufgabenergebnisses Schrittdetails im Bereich an.

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

Benutzen ResultSelector

Wenn Sie nun das Ergebnis der Identitäts- und Adressüberprüfungen für die folgenden Zustände in Ihrem Workflow bereitstellen müssen, können Sie den Knoten payload.body im Ausgabe-JSON

auswählen und die `StringToJson` intrinsische Funktion im `ResultSelector` Filter verwenden, um die Daten nach Bedarf zu formatieren.

`ResultSelector`wählt aus der Aufgabenausgabe aus, was benötigt wird. Im folgenden Beispiel wird die Zeichenfolge in $.payload.body verwendet und die **`States.StringToJson`** intrinsische Funktion angewendet, um die Zeichenfolge in JSON zu konvertieren, und fügt das resultierende JSON in den Identitätsknoten ein. **`ResultSelector`**

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

Folglich werden die folgenden JSON-Daten erstellt.

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

Bei der Arbeit mit diesen Eingabe- und Ausgabefiltern können auch Laufzeitfehler auftreten, die durch die Angabe ungültiger JSON-Pfadausdrücke entstehen. Weitere Informationen finden Sie unter .

Benutzen ResultPath

Mithilfe des `ResultPath` Felds können Sie in der ursprünglichen Eingabe-Payload einen Speicherort angeben, um das Ergebnis der Aufgabenverarbeitung eines Bundesstaates zu speichern.
Wenn Sie nichts angeben`ResultPath`, wird standardmäßig der Wert verwendet$, wodurch die anfängliche Eingabe-Payload durch das rohe Task-Ergebnis ersetzt wird. Wenn Sie `ResultPath` als angeben`null`, wird das Rohergebnis verworfen und die anfängliche Eingabe-Payload wird zur effektiven Ausgabe.

Wenn Sie das `ResultPath` Feld auf die mithilfe des Felds erstellten JSON-Daten anwenden, wird das Aufgabenergebnis innerhalb des Ergebnisknotens in der Eingabe-Payload hinzugefügt, wie im folgenden Beispiel gezeigt: `ResultSelector`

```
{
  "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
    }
  }
}
```

Benutzen OutputPath

Sie können einen Teil der Statusausgabe nach der Anwendung von auswählen`ResultPath`, um zum nächsten Status überzugehen. Auf diese Weise können Sie unerwünschte Informationen herausfiltern und nur den gewünschten Teil von JSON übergeben.

Im folgenden Beispiel speichert das `OutputPath` Feld die Statusausgabe im Ergebnisknoten:`"OutputPath": "$.results"`. Folglich sieht die endgültige Ausgabe des Zustands, die Sie an den nächsten Status übergeben können, wie folgt aus:

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

Verwendung von Konsolenfunktionen zur Visualisierung der Eingabe- und Ausgabedatenflüsse

Sie können den Eingabe- und Ausgabedatenfluss zwischen den Zuständen in Ihren Workflows mithilfe des Datenflusssimulators der Step Functions-Konsole oder der Option Erweiterte Ansicht auf der Seite Ausführungsdetails visualisieren.

Tutorial 8: Fehler in der Konsole debuggen

Bei der Arbeit mit Step Functions können Laufzeitfehler auftreten, die aus folgenden Gründen auftreten:

- Ein ungültiger JSON-Pfad für das Variablenfeld im `Choice` Bundesstaat.
- Problem mit der Definition von Zustandsmaschinen, z. B. keine passende Regel, die für einen `Choice` Status definiert wurde.
- Ungültige JSON-Pfadausdrücke beim Anwenden von Filtern zur Manipulation von Eingabe und Ausgabe.
- Die Aufgabe schlägt aufgrund einer Lambda-Funktionsausnahme fehl.
- IAM-Berechtigungsfehler.

In diesem Tutorial erfahren Sie, wie Sie einige dieser Fehler mithilfe der Step Functions-Konsole debuggen. Weitere Informationen finden Sie unter Fehlerbehandlung in Step Functions.

Themen

- Fehler beim Debuggen des ungültigen Pfades; Auswahlstatusfehler
- Debuggen von JSON-Pfadausdrucksfehlern beim Anwenden von Eingabe- und Ausgabefiltern

Fehler beim Debuggen des ungültigen Pfades; Auswahlstatusfehler

Wenn Sie im Variablenfeld des `Choice` Bundesstaates einen falschen oder nicht auflösbaren JSON-Pfad angeben oder keine passende Regel für den `Choice` Status definieren, erhalten Sie beim Ausführen Ihres Workflows eine Fehlermeldung.

Um den Fehler „Ungültiger Pfad“ zu veranschaulichen, führt dieses Tutorial einen `Choice` Statusfehler in Ihrem Workflow ein. Sie verwenden die CreditCardWorkflowState Machine und bearbeiten ihre Definition, um den Fehler einzufügen.

1. Öffnen Sie die Step Functions-Konsole und wählen Sie dann die CreditCardWorkflowState-Maschine aus.
2. Wählen Sie Bearbeiten, um die State-Machine-Definition zu bearbeiten. Nehmen Sie die im folgenden Code hervorgehobene Änderung an Ihrer State-Machine-Definition vor.

```
{
  "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. Wählen Sie Speichern und dann trotzdem speichern.
4. Starte die State Machine.
5. Führen Sie auf der Seite „Ausführungsdetails“ Ihrer State-Machine-Ausführung einen der folgenden Schritte aus:

   a. Wählen Sie in der Fehlermeldung Ursache aus, um den Grund für den Fehler bei der Ausführung einzusehen.

b. Wählen Sie in der Fehlermeldung Schrittdetails anzeigen, um den Schritt anzuzeigen, der den Fehler verursacht hat.

6. Wählen Sie im Abschnitt Schrittdetails auf der Registerkarte Eingabe und Ausgabe die Umschalttaste Erweiterte Ansicht, um den Pfad zur Eingabe- und Ausgabedatenübertragung für einen ausgewählten Status anzuzeigen.
7. Stellen Sie in der Diagrammansicht sicher, dass der zugewiesene Kredit **>=** 5000? ist ausgewählt und gehen Sie wie folgt vor:

   a. Sehen Sie sich den Eingabewert des Bundesstaates im Eingabefeld an.

   b. Wählen Sie die Registerkarte Definition und beachten Sie den JSON-Pfad, der für das Variablenfeld angegeben ist.

      Der Eingabewert für den angewandten Kredit **>=** 5000? state ist ein numerischer Wert, während Sie den JSON-Pfad für den Eingabewert als `$.Payload` angegeben haben. Während der Ausführung der State-Maschine kann der `Choice` Staat diesen JSON-Pfad nicht auflösen, da er nicht existiert.

8. Bearbeiten Sie die State Machine, um den Variablenfeldwert als anzugeben**$**.

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

Debuggen von JSON-Pfadausdrucksfehlern beim Anwenden von Eingabe- und Ausgabefiltern

Bei der Arbeit mit den Eingabe- und Ausgabefiltern können Laufzeitfehler auftreten, die durch die Angabe ungültiger JSON-Pfadausdrücke entstehen.

Das folgende Beispiel verwendet die WorkflowInputOutputin Tutorial 5 erstellte State-Maschine und zeigt ein Szenario, in dem Sie den `ResultSelector` Filter verwenden, um Teile der Aufgabenausgabe auszuwählen.

1. Wenden Sie den `ResultSelector` Filter an, um einen Teil der Aufgabenausgabe für den Schritt Identität überprüfen auszuwählen. Bearbeiten Sie dazu Ihre State Machine-Definition wie folgt:

```
{
  "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. Starte die State Machine.
3. Gehen Sie auf der Seite mit den Ausführungsdetails Ihrer State-Machine-Ausführung wie folgt vor:

   a. Wählen Sie in der Fehlermeldung Ursache aus, um den Grund für den Fehler bei der Ausführung einzusehen.

   b. Wählen Sie in der Fehlermeldung Schrittdetails anzeigen, um den Schritt anzuzeigen, der den Fehler verursacht hat.
4. Beachten Sie in der Fehlermeldung, dass der Inhalt des $.payload.Body-Knotens eine maskierte JSON-Zeichenfolge ist. Der Fehler ist aufgetreten, weil Sie mit der JSON-Pfadnotation nicht auf eine Zeichenfolge verweisen können.
5. Gehen Sie wie folgt vor, um auf den Knoten $.payload.body.Message zu verweisen:

   a. Verwenden Sie die `States.StringToJSON` intrinsische Funktion, um die Zeichenfolge zuerst in ein JSON-Format zu konvertieren.

   b. Geben Sie den JSON-Pfad für den Knoten $.payload.body.Message innerhalb der intrinsischen Funktion an.

   ```
   "ResultSelector": {
    "identity.$":"States.StringToJson($.Payload.body.message)"
   }
   ```
6. Führen Sie die State Machine erneut aus.

Anwendungsfälle

AWS Step Functionsermöglicht es Ihnen, visuelle Workflows zu erstellen, mit denen Sie Geschäftsanforderungen schnell in Anwendungen umsetzen können. Step Functions verwaltet Status, Checkpoints und Neustarts für Sie und bietet integrierte Funktionen zur automatischen Behandlung von Fehlern und Ausnahmen. Lesen Sie sich die folgenden Anwendungsfälle durch, um die Funktionen, die Step Functions Ihnen bieten kann, besser zu verstehen:

Themen

Datenverarbeitung

Angesichts des wachsenden Datenvolumens, das aus immer vielfältigeren Quellen stammt, müssen Unternehmen schnell handeln, um diese Daten zu verarbeiten, um sicherzustellen, dass sie schnellere, fundiertere Geschäftsentscheidungen treffen können. Um Daten in großem Maßstab verarbeiten zu können, müssen Unternehmen Ressourcen bereitstellen, um die Informationen zu verwalten, die sie von Mobilgeräten, Anwendungen, Satelliten, Marketing und Vertrieb, Betriebsdatenspeichern, Infrastruktur und mehr erhalten.

Step Functions bietet die Skalierbarkeit, Zuverlässigkeit und Verfügbarkeit, die für die erfolgreiche Verwaltung Ihrer Datenverarbeitungsworkflows erforderlich sind. Mit Step Functions können Sie Millionen von gleichzeitigen Ausführungen verwalten, da es horizontal skaliert und fehlertolerante Workflows bietet. Verarbeiten Sie Daten schneller, indem Sie parallele Ausführungen wie den `Parallel` Zustandstyp von Step Functions oder dynamische Parallelität mithilfe seines Zustandstyps verwenden. `Zuordnung` Als Teil Ihres Workflows können Sie den `Zuordnung` Status verwenden, um über Objekte in einem statischen Datenspeicher wie einem Amazon S3-Bucket zu iterieren. Mit Step Functions können Sie fehlgeschlagene Ausführungen auch einfach wiederholen oder eine bestimmte Methode zur Fehlerbehandlung wählen, ohne einen komplexen Prozess verwalten zu müssen.

Abhängig von Ihren Datenverarbeitungsanforderungen lässt sich Step Functions direkt in andere Datenverarbeitungsdienste integrieren, die von AWS beispielsweise AWS Batchzur

Stapelverarbeitung, Amazon EMR für die Verarbeitung großer Datenmengen, AWS Gluezur Datenvorbereitung, Athena zur Datenanalyse und AWS Lambdazur Datenverarbeitung bereitgestellt werden.

Zu den Arten von Datenverarbeitungsworkflows, für die Kunden Step Functions verwenden, gehören beispielsweise:

Datei-, Video- und Bildverarbeitung

- Nehmen Sie eine Sammlung von Videodateien und konvertieren Sie sie in andere Größen oder Auflösungen, die für das Gerät, auf dem sie angezeigt werden, ideal sind, z. B. Mobiltelefone, Laptops oder Fernseher.
- Nehmen Sie eine große Sammlung von Fotos auf, die von Benutzern hochgeladen wurden, und konvertieren Sie sie in Miniaturansichten oder Bilder mit unterschiedlicher Auflösung, die dann auf den Websites der Benutzer angezeigt werden können.
- Nehmen Sie halbstrukturierte Daten, wie z. B. eine CSV-Datei, und kombinieren Sie sie mit unstrukturierten Daten wie einer Rechnung, um einen Geschäftsbericht zu erstellen, der monatlich an die Geschäftsbeteiligten gesendet wird.
- Nehmen Sie Erdbeobachtungsdaten, die von Satelliten gesammelt wurden, konvertieren Sie sie in Formate, die aufeinander abgestimmt sind, und fügen Sie dann weitere auf der Erde gesammelte Datenquellen hinzu, um zusätzliche Erkenntnisse zu gewinnen.
- Nehmen Sie die Transportprotokolle verschiedener Transportarten für Produkte und suchen Sie mithilfe von Monte-Carlo-Simulationen nach Optimierungen. Senden Sie dann Berichte an die Organisationen und Personen, die sich beim Versand ihrer Waren auf Sie verlassen.

Aufgaben zum Extrahieren, Transformieren und Laden (ETL) koordinieren:

- Kombinieren Sie Datensätze über Verkaufschancen mit Datensätzen zu Marketingkennzahlen, indem Sie eine Reihe von Schritten zur Datenaufbereitung verwendenAWS Glue, und erstellen Sie Business Intelligence-Berichte, die im gesamten Unternehmen verwendet werden können.
- Erstellen, starten und beenden Sie einen Amazon EMR-Cluster für die Verarbeitung großer Datenmengen.

Stapelverarbeitung und HPC-Workloads (High Performance Computing):

- Erstellen Sie eine Pipeline für die sekundäre Genomanalyse, die rohe gesamte Genomsequenzen zu Variantenaufrufen verarbeitet. Richten Sie Rohdateien an eine Referenzsequenz aus und rufen Sie mithilfe dynamischer Parallelität Varianten auf einer bestimmten Liste von Chromosomen auf.
- Finden Sie Effizienzsteigerungen bei der Herstellung Ihres nächsten Mobilgeräts oder anderer Elektronik, indem Sie verschiedene Layouts mit verschiedenen elektrischen und chemischen Verbindungen simulieren. Führen Sie umfangreiche Batchverarbeitung Ihrer Workloads durch verschiedene Simulationen durch, um das optimale Design zu erhalten.

Machine Learning

Maschinelles Lernen ermöglicht es Unternehmen, gesammelte Daten schnell zu analysieren, um Muster zu erkennen, und dann Entscheidungen mit minimalem menschlichem Eingreifen zu treffen. Maschinelles Lernen beginnt mit einem ersten Datensatz, den sogenannten Trainingsdaten. Diese Trainingsdaten tragen dazu bei, die Vorhersagegenauigkeit eines Modells für maschinelles Lernen zu erhöhen, und dienen als Grundlage für das Lernen dieses Modells. Sobald das Modell als genau genug angesehen wird, um die Geschäftsanforderungen zu erfüllen, wird es in der Produktion eingesetzt. Das AWS Step FunctionsData Science Software Development Kit (SDK) ist eine Open-Source-Bibliothek, mit der Sie auf einfache Weise Workflows erstellen können, die Daten vorverarbeiten, Ihre Modelle mithilfe von Amazon und Step Functions trainieren SageMaker und dann veröffentlichen.

Durch die Vorverarbeitung vorhandener Datensätze erstellt ein Unternehmen häufig Trainingsdaten. Diese Methode fügt Informationen hinzu, z. B. indem Objekte in einem Bild beschriftet, Text mit Anmerkungen versehen oder Audio verarbeitet werden. Um Daten vorzuverarbeiten, können Sie eine Notebook-Instanz verwendenAWS Glue, oder Sie können eine SageMaker Notebook-Instanz erstellen, auf der die Jupyter Notebook-App ausgeführt wird. Sobald Ihre Daten bereit sind, können sie für einen einfachen Zugriff auf Amazon S3 hochgeladen werden. Während die Modelle für maschinelles Lernen trainiert werden, können Sie die Parameter jedes Modells anpassen, um die Genauigkeit zu verbessern, bis es einsatzbereit ist.

Mit Step Functions können Sie durchgängige Workflows für maschinelles Lernen orchestrieren. SageMaker Diese Workflows können Datenvorverarbeitung, Nachverarbeitung, Feature-Engineering, Datenvalidierung und Modellbewertung umfassen. Sobald das Modell in der Produktion eingesetzt wurde, können Sie neue Ansätze verfeinern und testen, um die Geschäftsergebnisse kontinuierlich zu verbessern. Sie können produktionsbereite Workflows direkt in Python erstellen oder das Step Functions Data Science SDK verwenden, um diesen Workflow zu kopieren, mit neuen Optionen zu experimentieren und den verfeinerten Workflow in der Produktion zu platzieren.

Einige Arten von Workflows für maschinelles Lernen, für die Kunden Step Functions verwenden, umfassen:

Betrugserkennung

- Identifizieren und verhindern Sie betrügerische Transaktionen wie Kreditbetrug.
- Erkennen und verhindern Sie Kontoübernahmen mithilfe von trainierten Modellen für maschinelles Lernen.
- Identifizieren Sie Werbemissbrauch, einschließlich der Erstellung gefälschter Konten, damit Sie schnell Maßnahmen ergreifen können.

Personalisierung und Empfehlungen

- Empfehlen Sie bestimmten Kunden Produkte auf der Grundlage dessen, was voraussichtlich ihr Interesse wecken wird.
- Prognostizieren Sie, ob ein Kunde sein Konto von einem kostenlosen Kontingent auf ein kostenpflichtiges Abonnement hochstufen wird.

Datenanreicherung

- Verwenden Sie die Datenanreicherung als Teil der Vorverarbeitung, um bessere Trainingsdaten für genauere Modelle des maschinellen Lernens bereitzustellen.
- Kommentieren Sie Text- und Audioauszüge, um syntaktische Informationen wie Sarkasmus und Slang hinzuzufügen.
- Kennzeichnen Sie zusätzliche Objekte in Bildern, um wichtige Informationen bereitzustellen, aus denen das Modell lernen kann, z. B. ob es sich bei einem Objekt um einen Apfel, einen Basketball, einen Stein oder ein Tier handelt.

Microservice-Orchestrierung

Die Microservice-Architektur unterteilt Anwendungen in lose miteinander verbundene Dienste. Zu den Vorteilen gehören eine verbesserte Skalierbarkeit, eine höhere Stabilität und eine schnellere Markteinführung. Jeder Microservice ist unabhängig, sodass es einfach ist, einen einzelnen Dienst oder eine einzelne Funktion zu skalieren, ohne die gesamte Anwendung skalieren zu müssen. Einzelne Dienste sind lose miteinander verknüpft, sodass sich unabhängige Teams auf einen einzigen Geschäftsprozess konzentrieren können, ohne dass sie die gesamte Anwendung

verstehen müssen. Mit Microservices können Sie auch wählen, welche einzelnen Komponenten Ihren Geschäftsanforderungen entsprechen, sodass Sie Ihre Auswahl flexibel ändern können, ohne Ihren gesamten Workflow neu schreiben zu müssen. Verschiedene Teams können die Programmiersprachen und Frameworks ihrer Wahl verwenden, um mit ihrem Microservice zu arbeiten, und dieser Microservice kann weiterhin über Anwendungsprogrammierschnittstellen (APIs) mit jedem anderen in der Anwendung kommunizieren.

Step Functions bietet Ihnen verschiedene Möglichkeiten, Ihre Microservice-Workflows zu verwalten. Für Workflows mit langer Laufzeit können Sie Standard-Workflows mit der AWS Fargate Integration verwenden, um Anwendungen zu orchestrieren, die in Containern ausgeführt werden. Für Workflows mit hohem Volumen von kurzer Dauer, die eine sofortige Reaktion erfordern, sind Synchronous Express-Workflows ideal. Diese können für webbasierte oder mobile Anwendungen verwendet werden, für die häufig Arbeitsabläufe von kurzer Dauer sind und die Durchführung einer Reihe von Schritten erfordern, bevor eine Antwort zurückgegeben wird. Sie können synchrone Express-Workflows direkt von Amazon API Gateway aus auslösen, und die Verbindung wird so lange geöffnet, bis der Workflow abgeschlossen ist oder es zu Timeouts kommt. Für Workflows mit kurzer Dauer, die keine sofortige Reaktion erfordern, bietet Step Functions asynchrone Express-Workflows.

Zu den Beispielen einiger API-Orchestrierungen, die Step Functions verwenden, gehören:

Synchrone Workflows oder Echtzeit-Workflows

- Ändern Sie einen Wert in einem Datensatz, z. B. die Aktualisierung des Nachnamens eines Mitarbeiters, und lassen Sie die Änderung sofort auf dem Bildschirm sichtbar werden.
- Aktualisieren Sie eine Bestellung während des Bezahlvorgangs, indem Sie beispielsweise einen Artikel hinzufügen, entfernen oder die Menge ändern, und teilen Sie dem Kunden die Aktualisierung dann sofort mit.
- Führen Sie einen Schnellverarbeitungsauftrag aus und senden Sie das Ergebnis sofort an den Anforderer zurück.

Container-Orchestrierung

- Führen Sie Jobs auf Kubernetes mit Amazon Elastic Kubernetes Service oder auf Amazon Elastic Container Service (ECS) mit Fargate aus und integrieren Sie sie als Teil desselben Workflows in andere AWS Dienste, z. B. das Senden von Benachrichtigungen mit Amazon SNS.

IT- und Sicherheitsautomatisierung

IT-Automatisierung kann dabei helfen, immer komplexere und zeitaufwändigere Abläufe zu bewältigen, z. B. das Aktualisieren und Patchen von Software, die Bereitstellung von Sicherheitsupdates zur Behebung von Sicherheitslücken, die Auswahl der Infrastruktur, das Synchronisieren von Daten, das Weiterleiten von Supporttickets und mehr. Die Automatisierung sich wiederholender und zeitaufwändiger Aufgaben kann es Ihrem Unternehmen ermöglichen, Routineoperationen schnell und konsistent in großem Umfang abzuschließen. So können Sie sich auf strategische Aufgaben wie die Entwicklung von Funktionen, komplexe Supportanfragen und Innovationen konzentrieren und gleichzeitig diesen wachsenden Anforderungen gerecht werden.

Mit Step Functions können Sie Workflows erstellen, die automatisch an die Anforderungen Ihres Unternehmens angepasst werden, ohne dass manuelles Eingreifen erforderlich ist. In Fällen, in denen in Ihrem Arbeitsablauf ein Fehler auftritt, ist häufig kein manuelles Eingreifen erforderlich. Mit Step Functions können Sie fehlgeschlagene Aufgaben automatisch wiederholen und einen exponentiellen Rückstand aufbauen, der Fehler in Ihrem Arbeitsablauf beheben kann.

Es kann Situationen geben, in denen menschliches Eingreifen erforderlich ist, bevor der Arbeitsablauf voranschreiten kann. Für die Genehmigung einer erheblichen Krediterhöhung kann beispielsweise die Zustimmung eines Menschen erforderlich sein. Um dies zu verwalten, können Sie in Step Functions eine Verzweigungslogik definieren, sodass nur Anfragen, die einen bestimmten Betrag überschreiten, eine menschliche Genehmigung erfordern, während alle anderen Anfragen automatisch abgeschlossen werden. In Fällen, in denen eine menschliche Genehmigung erforderlich ist, können Sie mit Step Functions den Workflow an einem bestimmten Schritt unterbrechen, auf eine Antwort warten und den Workflow dann fortsetzen, sobald die Antwort eingegangen ist.

Einige Beispiele für die Arten von Automatisierungsworkflows, für die Kunden Step Functions verwenden, sind:

IT-Automatisierung

- Beheben Sie Vorfälle wie das Öffnen eines SSH-Ports, wenig Speicherplatz oder öffentliche Zugriffe auf einen Amazon S3-Bucket automatisch.
- Automatisieren Sie den Einsatz von AWS CloudFormation StackSets

Automatisierung der Sicherheit

- Automatisieren Sie die Reaktion auf ein Szenario, in dem ein Benutzer und ein Benutzerzugriffsschlüssel offengelegt wurden.
- Beheben Sie Reaktionen auf Sicherheitsvorfälle automatisch anhand definierter Richtlinienmaßnahmen, z. B. der Beschränkung von Aktionen auf bestimmte ARNs oder der Anwendung anderer Aktionen.
- Warnen Sie Ihre Mitarbeiter innerhalb von Sekunden nach Erhalt vor Phishing-E-Mails.

Zustimmung durch den Menschen

- Automatisieren Sie das Training des Modells für maschinelles Lernen und fordern Sie dann die manuelle Genehmigung des Modells durch einen Datenwissenschaftler an, bevor Sie das Modell auf der Grundlage der eingegangenen Antwort automatisch bereitstellen oder ablehnen.
- Automatisieren Sie die Weiterleitung von Kundenfeedback auf der Grundlage einer Stimmungsanalyse, sodass Personen mit einer negativen Stimmung sofort zur manuellen Überprüfung weitergeleitet werden.

So funktioniert Step Functions

Dieser Abschnitt beschreibt wichtige Konzepte, die Ihnen dabei helfen, sich mit AWS Step Functions und der Funktionsweise dieses Service vertraut zu machen.

Themen

Standard- und Express-Workflows

Wenn Sie eine Zustandsmaschine erstellen, wählen Sie entweder den Typ Standard oder Express aus. Der Standardtyp für Zustandsmaschinen ist Standard. Eine Zustandsmaschine, deren Typ Standard ist, wird als Standard-Workflow bezeichnet, und eine Zustandsmaschine, deren Typ Express ist, wird als Express-Workflow bezeichnet.

Sowohl für Standard- als auch für Express-Workflows definieren Sie Ihre Zustandsmaschine mithilfe vonAmazon States Language. Ihre State-Machine-Ausführungen verhalten sich je nach Typ, den Sie auswählen, unterschiedlich.

> **Important**
>
> Der von Ihnen gewählte Typ kann nicht mehr geändert werden, nachdem Sie den Zustandsmaschine erstellt haben.

> **Note**
>
> Wenn Sie Ihre Zustandsmaschinen außerhalb der Step Functions Functions-Konsole definieren, z. B. in einem Editor Ihrer Wahl, müssen Sie Ihre Zustandsmaschinendefinitionen mit der Erweiterung .asl.json speichern.

Standard-Workflows eignen sich ideal für langwierige (bis zu einem Jahr), dauerhafte und überprüfbare Workflows. Sie können den vollständigen Ausführungsverlauf mit der Step Functions Functions-API bis zu 90 Tage nach Abschluss Ihrer Ausführung abrufen. Standard-Workflows folgen einem Exactly-Once-Modell, bei dem Ihre Aufgaben und Status nie mehr als einmal ausgeführt werden, es sei denn, Sie haben in ASL ein bestimmtes `Retry` Verhalten festgelegt. Dadurch eignen sich Standard-Workflows für die Orchestrierung nicht idempotenter Aktionen, wie z. B. das Starten eines Amazon EMR-Clusters oder die Verarbeitung von Zahlungen. Standard-Workflow-Ausführungen werden nach der Anzahl der verarbeiteten Statusübergänge abgerechnet.

Express-Workflows sind ideal für hochvolumige Ereignisverarbeitungs-Workloads wie IoT-Datenaufnahme, Streaming-Datenverarbeitung und -transformation sowie mobile Anwendungs-Backends. Sie können bis zu fünf Minuten lang ausgeführt werden. Express-Workflows verwenden ein at-least-onceModell, bei dem eine Ausführung möglicherweise mehr als einmal ausgeführt werden kann. Dies macht Express Workflows ideal für die Orchestrierung idempotenter Aktionen wie das Transformieren von Eingabedaten und das Speichern über eine PUT-Aktion in Amazon DynamoDB. Express-Workflow-Ausführungen werden nach der Anzahl der Ausführungen, der Ausführungsdauer und dem während der Ausführung verbrauchten Speicher abgerechnet.

Standard- und Express-Workflows können automatisch als Reaktion auf Ereignisse wie HTTP-Anfragen von Amazon API Gateway (vollständig verwaltete APIs im großen Maßstab), IoT-Regeln und über 140 anderen Ereignisquellen in Amazon EventBridge gestartet werden.

> **Tip**
>
> Ein Beispiel für einen Express-Workflow finden Sie AWS-Konto in Modul 7 — API Gateway, Parallel State, Express-Workflows von The AWS Step Functions Workshop.

Informationen zur Konsolenoberfläche für Standard- und Express-Workflow-Ausführungen finden Sie unterStandard- und Express-Workflow-Ausführungen in der Konsole.

Standard-Workflows im Vergleich zu Express-Workflows

	Standard-Workflows	Express-Workflows: Synchron und asynchron
Maximale Dauer	Ein Jahr	Fünf Minuten
Unterstützte Ausführungsstartrate	Informationen zu Kontingenten im Zusammenhang mit der unterstützten Ausführungsstartrate finden Sie unterKontingente im Zusammenhang mit der Drosselung von API-Aktionen.	Informationen zu Kontingenten im Zusammenhang mit der unterstützten Ausführungsstartrate finden Sie unterKontingente im Zusammenhang mit der Drosselung von API-Aktionen.
Unterstützte Zustandsübergangsrate	Informationen zu Kontingenten im Zusammenhang mit der Rate unterstützter Statusübergänge finden Sie unterKontingente im Zusammenhang mit staatlicher Drosselung.	Kein Limit
Preise	Die Preise richten sich nach der Anzahl der Zustandsübergänge. Ein Zustandsübergang wird jedes Mal gezählt, wenn ein Schritt in	Die Kosten richten sich nach der Anzahl der von Ihnen ausgeführten Ausführungen sowie deren Dauer und Speicherverbrauch.

	Standard-Workflows	Express-Workflows: Synchron und asynchron
	Ihrer Ausführung abgeschlossen ist.	
Ausführungsverlauf	Ausführungen können mit Step Functions Functions-APIs aufgelistet und beschrieben werden. Ausführungen können visuell über die Konsole debuggt werden. Sie können auch in CloudWatch Logs eingesehen werden, indem Sie die Protokollierung auf Ihrem State-Computer aktivieren. Weitere Informationen zum Debuggen von Standard-Workflow-Ausführungen in der Konsole finden Sie unter Standard- und Express-Workflow-Ausführungen in der Konsole und. Anzeigen und Debuggen von Ausführungen	Unbegrenzter Ausführungsverlauf, d. h. es werden so viele Ausführungshistorieeinträge verwaltet, wie Sie innerhalb eines Zeitraums von 5 Minuten generieren können. Ausführungen können in CloudWatch Logs oder in der Step Functions Functions-Konsole überprüft werden, indem Sie die Protokollierung auf Ihrer Zustandsmaschine aktivieren. Weitere Informationen zum Debuggen von Express Workflow-Ausführungen in der Konsole finden Sie unter und. Standard- und Express-Workflow-Ausführungen in der Konsole Anzeigen und Debuggen von Ausführungen
Ausführungssemantik	Workflow-Ausführung exakt einmal.	Asynchrone Express-Workflows: Eine Workflow-Ausführung. t-least-once Synchrone Express-Workflows: Eine t-most-once Workflow-Ausführung.

	Standard-Workflows	Express-Workflows: Synchron und asynchron
Service-Integrationen	Unterstützt alle Service-Integrationen und -Muster.	Unterstützt alle Service-Integrationen. **Note** Express-Workflows unterstützen keine Job-Run (`.sync`) - oder Callback (`.waitForTaskToken`) -Dienstintegrationsmuster.
Step Functions Functions-Aktivitäten	Unterstützt Step Functions Functions-Aktivitäten.	Unterstützt keine Step Functions Functions-Aktivitäten.

Synchrone und asynchrone Express-Workflows

Sie können zwischen zwei Arten von Express-Workflows wählen: Asynchrone Express-Workflows und synchrone Express-Workflows.

- Asynchrone Express-Workflows geben eine Bestätigung zurück, dass der Workflow gestartet wurde, warten aber nicht, bis der Workflow abgeschlossen ist. Um das Ergebnis zu erhalten, müssen Sie die CloudWatch Protokolle des Dienstes abfragen. Sie können Asynchrone Express-Workflows verwenden, wenn Sie keine sofortige Antwortausgabe benötigen, z. B. bei Messaging-Diensten oder Datenverarbeitung, von der andere Dienste nicht abhängig sind. Sie können asynchrone Express-Workflows als Reaktion auf ein Ereignis, durch einen verschachtelten Workflow in Step Functions oder mithilfe des `StartExecution` API-Aufrufs starten.
- Synchrone Express-Workflows starten einen Workflow, warten, bis er abgeschlossen ist, und geben dann das Ergebnis zurück. Synchrone Express-Workflows können zur Orchestrierung von Microservices verwendet werden. Mit Synchronous Express Workflows können Sie Anwendungen entwickeln, ohne zusätzlichen Code entwickeln zu müssen, um Fehler zu

behandeln, Wiederholungsversuche durchzuführen oder parallel Aufgaben auszuführen. Sie können Synchronous Express-Workflows ausführen, AWS Lambda die über Amazon API Gateway oder mithilfe des `StartSyncExecution` API-Aufrufs aufgerufen werden.

> **Note**
>
> Wenn Sie Step Functions Express Workflows synchron von der Konsole aus ausführen, läuft die `StartSyncExecution` Anforderung nach 60 Sekunden ab. Um die Express-Workflows synchron für eine Dauer von bis zu fünf Minuten auszuführen, stellen Sie die `StartSyncExecution` Anfrage mithilfe des AWS SDK oder AWS Command Line Interface (AWS CLI) anstelle der Step Functions Functions-Konsole.

Synchrone API-Aufrufe zur Express-Ausführung tragen nicht zu den bestehenden Kapazitätsgrenzen für Konten bei. Step Functions stellt Kapazität nach Bedarf bereit und skaliert automatisch bei anhaltender Arbeitslast. Ein Anstieg der Arbeitslast kann gedrosselt werden, bis Kapazität verfügbar ist.

Die Ausführung garantiert

Standard-Workflows	Asynchrone Express-Workflows	Synchrone Express-Workflows	
Workflow-Ausführung exakt einmal	Eine Workflow-Ausführung t-least-once	Eine t-most-once Workflow-Ausführung	
Der Ausführungsstatus bleibt intern zwischen Statusübergängen bestehen.	Der Ausführungsstatus bleibt zwischen Statusübergängen nicht bestehen.	Der Ausführungsstatus bleibt zwischen Statusübergängen nicht bestehen.	
Gibt automatisch eine idempotente Antwort zurück, wenn eine Ausführun	Idempotenz wird nicht automatisch behandelt. Das Starten mehrerer	Idempotenz wird nicht automatisch verwaltet. Step Functions wartet, sobald eine Ausführun	

Standard-Workflows	Asynchrone Express-Workflows	Synchrone Express-Workflows	
g mit demselben Namen wie ein aktuell ausgeführter Workflow gestartet wird. Der neue Workflow startet nicht und es wird eine Ausnahme ausgelöst, sobald der aktuell ausgeführte Workflow abgeschlossen ist.	Workflows mit demselben Namen führt zu gleichzeitigen Ausführungen. Kann zum Verlust des internen Workflow-Status führen, wenn die Zustandsmaschinenlogik nicht idempotent ist.	g gestartet wird, und gibt nach Abschluss das Ergebnis der Zustandsmaschine zurück. Workflows werden nicht neu gestartet, wenn eine Ausnahme auftritt.	

Standard-Workflows	Asynchrone Express-Workflows	Synchrone Express-Workflows	
Die Daten zum Ausführungsverlauf wurden nach 90 Tagen entfernt. Workflow-Namen können nach dem Entfernen von out-of-date Ausführungsdaten wiederverwendet werden. Um die Einhaltung gesetzlicher, organisatorischer oder behördlicher Anforderungen zu erfüllen, können Sie die Aufbewahrungsfrist für den Ausführungsverlauf auf 30 Tage reduzieren, indem Sie eine Kontingentanfrage senden. Verwenden Sie dazu den AWS Support Center Console und erstellen Sie einen neuen Fall.	Der Ausführungsverlauf wird nicht von Step Functions erfasst. Die Protokollierung muss über Amazon CloudWatch Logs aktiviert werden.	Der Ausführungsverlauf wird nicht von Step Functions erfasst. Die Protokollierung muss über Amazon CloudWatch Logs aktiviert werden.	

Kostenoptimierung mit Express Workflows

Step Functions bestimmt die Preise für Standard- und Express-Workflows auf der Grundlage des Workflowtyps, den Sie zum Erstellen Ihrer Zustandsmaschinen verwenden. Um die Kosten Ihrer

serverlosen Workflows zu optimieren, können Sie eine oder beide der folgenden Empfehlungen befolgen:

Themen

Informationen dazu, wie sich die Wahl eines Standard- oder Express-Workflow-Typs auf die Abrechnung auswirkt, finden Sie unter AWS Step FunctionsPreisgestaltung.

Tipp #1: Express-Workflows innerhalb von Standard-Workflows verschachteln

Step Functions führt Workflows aus, die eine begrenzte Dauer und Anzahl von Schritten haben. Einige Workflows können die Ausführung innerhalb eines kurzen Zeitraums abschließen. Andere erfordern möglicherweise eine Kombination aus langwierigen high-event-rate Workflows und Workflows. Mit Step Functions können Sie große, komplexe Workflows aus mehreren kleineren, einfacheren Workflows erstellen.

Um beispielsweise einen Workflow für die Auftragsabwicklung zu erstellen, können Sie alle Aktionen, die nicht idempotent sind, in einen Standard-Workflow aufnehmen. Dies könnte Aktionen wie die Genehmigung von Bestellungen durch menschliche Interaktion und die Bearbeitung von Zahlungen beinhalten. Anschließend können Sie eine Reihe idempotenter Aktionen, wie das Senden von Zahlungsbenachrichtigungen und die Aktualisierung des Produktbestands, in einem Express-Workflow kombinieren. Sie können diesen Express-Workflow innerhalb des Standard-Workflows verschachteln. In diesem Beispiel wird der Standard-Workflow als Parent State Machine bezeichnet. Der verschachtelte Express-Workflow wird als Child State Machine bezeichnet.

Tipp #2: Konvertieren Sie Standard-Workflows in Express-Workflows

Sie können Ihre vorhandenen Standard-Workflows in Express-Workflows konvertieren, wenn sie die folgenden Anforderungen erfüllen:

- Der Workflow muss seine Ausführung innerhalb von fünf Minuten abschließen.
- Der Workflow entspricht einem at-least-onceAusführungsmodell. Das bedeutet, dass jeder Schritt im Workflow mehr als genau einmal ausgeführt werden kann.
- Der Workflow verwendet nicht die `.waitForTaskToken` oder `.sync` Service-Integrationsmuster.

> **Important**
>
> Express-Workflows verwenden Amazon CloudWatch Logs, um Ausführungsverläufe aufzuzeichnen. Bei der Verwendung CloudWatch von Logs fallen zusätzliche Kosten an.

Um einen Standard-Workflow mithilfe der Konsole in einen Express-Workflow zu konvertieren

1. Öffnen Sie die Step Functions Functions-Konsole.
2. Wählen Sie auf der Seite State Machines einen State-Machine vom Typ Standard aus, um ihn zu öffnen.

   > **Tip**
   >
   > Wählen Sie in der Dropdownliste Beliebiger Typ die Option Standard aus, um die Liste der Zustandsmaschinen zu filtern und nur Standard-Workflows anzuzeigen.

3. Wählen Sie „In neu kopieren“.

   Workflow Studio wird geöffnet und Entwurfsmodus zeigt den Workflow der ausgewählten Zustandsmaschine an.
4. (Optional) Aktualisieren Sie den Workflow-Entwurf.
5. Geben Sie einen Namen für Ihre Zustandsmaschine an. Wählen Sie dazu das Bearbeitungssymbol neben dem Standardnamen der Zustandsmaschine von MyStateMachine. Geben Sie dann unter State-Machine-Konfiguration einen Namen in das Feld State-Machine-Name ein.
6. (Optional) Geben Sie unter State-Machine-Konfiguration weitere Workflow-Einstellungen an, z. B. den Zustandsmaschinentyp und seine Ausführungsrolle.

   Stellen Sie sicher, dass Sie als Typ die Option Express wählen. Behalten Sie alle anderen Standardauswahlen in den State-Machine-Einstellungen bei.

   > **Note**
   >
   > Wenn Sie einen Standard-Workflow konvertieren, der zuvor in AWS CDKoder definiert wurdeAWS SAM, müssen Sie den Wert `Type` und den `Resource` Namen ändern.

7. Wählen Sie im Dialogfeld zur Bestätigung der Rollenerstellung die Option Bestätigen aus, um fortzufahren.

   Sie können auch Rolleneinstellungen anzeigen wählen, um zur State-Machine-Konfiguration zurückzukehren.

   > **Note**
   >
   > Wenn Sie die von Step Functions erstellte IAM-Rolle löschen, kann Step Functions sie später nicht mehr neu erstellen. Ebenso kann Step Functions ihre ursprünglichen Einstellungen später nicht wiederherstellen, wenn Sie die Rolle ändern (z. B. indem Sie Step Functions aus den Principals in der IAM-Richtlinie entfernen).

Weitere Informationen zu bewährten Methoden und Richtlinien für die Verwaltung der Kostenoptimierung für Ihre Workflows finden Sie unter Erstellen kostengünstiger Workflows. AWS Step Functions

Zustände

Einzelne Staaten können auf der Grundlage ihrer Eingaben Entscheidungen treffen, anhand dieser Eingaben Aktionen ausführen und die Ergebnisse an andere Staaten weitergeben. In AWS Step Functions definieren Sie Ihre Workflows in der Amazon States Language (ASL). Die Step Functions-Konsole bietet eine grafische Darstellung Ihrer Zustandsmaschine, um die Logik Ihrer Anwendung zu visualisieren.

> **Note**
>
> Wenn Sie Ihre State-Machines außerhalb der Step Functions-Konsole definieren, z. B. in einem Editor Ihrer Wahl, müssen Sie Ihre State-Machine-Definitionen mit der Erweiterung .asl.json speichern.

Staaten sind Elemente in Ihrer Staatsmaschinerie. Ein Zustand wird durch seinen Namen bezeichnet, der eine beliebige Zeichenfolge sein kann, jedoch im Bereich des gesamten Zustandsautomaten eindeutig sein muss.

Zustände können eine Reihe von Funktionen in Ihrem Zustandsautomaten ausführen:

- Arbeiten Sie in Ihrem Zustandsautomaten (ein Aufgaben-Zustand).
- Eine Entscheidung zwischen Ausführungsverzweigungen treffen (ein Choice-Zustand)
- Eine Ausführung mit einem Fehler oder mit Erfolg beenden (ein Fail- oder Succeed-Zustand)
- Übergibt die Eingabe an die Ausgabe oder fügt einige feste Daten in den Workflow ein (ein Pass-Status)
- Geben Sie eine Verzögerung für einen bestimmten Zeitraum oder bis zu einem bestimmten Datum und einer bestimmten Uhrzeit an (Wartestatus)
- Parallele Ausführungsverzweigungen beginnen (ein Parallel-Zustand)
- Dynamische Iteration von Schritten (ein Zuordnungsstatus)

Im Folgenden finden Sie ein Beispiel für einen Zustand mit den Namen `HelloWorld`, der eine AWS Lambda-Funktion ausführt.

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

Zustände weisen viele gemeinsame Merkmale auf:

- Ein `Type` Feld, das angibt, um welche Art von Staat es sich handelt.
- Ein optionales `Comment` Feld für einen menschenlesbaren Kommentar oder eine Beschreibung des Bundesstaates.
- Jeder Zustand (außer einem `Succeed`- oder `Fail`-Zustand) erfordert ein `Next`-Feld oder kann alternativ ein Beendigungszustand werden, indem ein `End`-Feld angegeben wird.

> **Note**
>
> Ein `Choice`-Zustand kann mehr als ein `Next`-Feld haben, aber nur eines innerhalb jeder Auswahlregel. Ein `Choice` Staat kann es nicht gebrauchen`End`.

Bestimmte Zustandstypen erfordern zusätzliche Felder oder können die Nutzung häufiger Felder umdefinieren.

Nachdem Sie Standard-Workflows erstellt und ausgeführt haben, können Sie auf der Seite Ausführungsdetails in der Step Functions-Konsole auf Informationen über jeden Status, seine Eingabe und Ausgabe, wann er aktiv war und wie lange er aktiv war, zugreifen. Weitere Informationen finden Sie unter Anzeigen und Debuggen von Ausführungen in der Step Functions-Konsole.

Nachdem Sie Express Workflow-Ausführungen erstellt und ausgeführt haben und die Protokollierung für Ihren Express Workflow aktiviert ist, können Sie in Amazon CloudWatch Logs oder der Step Functions-Konsole auf Informationen zur Ausführung zugreifen. Weitere Informationen finden Sie unter Anzeigen und Debuggen von Ausführungen in der Step Functions-Konsole.

Themen

Amazon States Language

Die Amazon States Language ist eine JSON-basierte, strukturierte Sprache, die verwendet wird, um Ihre State Machine zu definieren, eine Sammlung von Zuständen, die funktionieren können (`Task`Zustände), zu bestimmen, zu welchen Zuständen als nächstes übergegangen werden soll (`Choice`Zustände), eine Ausführung mit einem Fehler zu beenden (`Fail`Zustände) usw.

Weitere Informationen finden Sie in der Amazon States Language Specification und in Statelint, einem Tool, das Amazon States Language-Code validiert.

Informationen zum Erstellen einer State Machine auf der Step Functions-Konsole mithilfe von Amazon States Language finden Sie unter Erste Schritte.

> **Note**
>
> Wenn Sie Ihre State-Machines außerhalb der Step Functions-Konsole definieren, z. B. in einem Editor Ihrer Wahl, müssen Sie Ihre State-Machine-Definitionen mit der Erweiterung .asl.json speichern.

Beispiel für eine Sprachspezifikation von Amazon States

```
{
  "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
    }
  }
}
```

Themen

- Struktur der Zustandsmaschine
- Intrinsische Funktionen
- Häufige Zustandsfelder

Struktur der Zustandsmaschine

Zustandsautomaten werden mit Hilfe von JSON-Text definiert, der eine Struktur repräsentiert, die folgende Felder enthält.

`Comment` (Optional)

Eine für Menschen lesbare Beschreibung des Zustandsautomaten.

`StartAt` (Erforderlich)

Eine Zeichenfolge, die exakt (auch in Groß-/Kleinschreibung) mit dem Namen eines der Zustandsautomaten übereinstimmen muss.

`TimeoutSeconds`(Fakultativ)

Die maximale Anzahl von Sekunden, die ein Zustandsautomat ausgeführt werden kann. Wenn sie länger als die angegebene Zeit ausgeführt wird, schlägt die Ausführung mit einem `States.Timeout`--Fehlernamen fehl.

`Version` (Optional)

Die Version der Amazon States-Sprache, die in der Zustandsmaschine verwendet wird (Standard ist „1.0").

`States` (Erforderlich)

Ein Objekt, das eine durch Kommata getrennte Gruppe von Zuständen umfasst.

Das Feld `States` enthält Zustände.

```
{
    "State1" : {
    },

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

Ein Zustandsautomat wird definiert durch die Zustände, die er umfasst, und die Beziehungen zwischen ihnen.

Im Folgenden wird ein Beispiel gezeigt.

```
{
  "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
    }
  }
}
```

Wenn eine Ausführung dieses Zustandsautomaten gestartet wird, beginnt das System mit dem Zustand, auf den das Feld `StartAt` verweist (`"HelloWorld"`). Wenn dieser Zustand ein Feld `"End": true` hat, stoppt die Ausführung und ein Ergebnis wird zurückgegeben. Andernfalls sucht das System nach einem Feld `"Next":` und fährt mit diesem Zustand fort. Dieser Vorgang wird wiederholt, bis das System einen Beendigungszustand erreicht (einen Zustand mit `"Type": "Succeed"`, `"Type": "Fail"` oder `"End": true`) oder ein Laufzeitfehler auftritt.

Die folgenden Regeln gelten für Zustände in einem Zustandsautomaten:

- Zustände können in beliebiger Reihenfolge innerhalb des umschließenden Blocks auftreten. Die Reihenfolge, in der sie aufgelistet werden, wirkt sich jedoch nicht auf die Reihenfolge aus, in der sie ausgeführt werden. Die Reihenfolge wird durch die Inhalte der Zustände bestimmt.
- Innerhalb eines Zustandsautomaten kann nur ein Zustand als `start`-Zustand bezeichnet werden, und zwar durch den Wert des Felds `StartAt` in der Top-Level-Struktur. Dieser Zustand ist derjenige, der als erster ausgeführt wird, wenn die Ausführung startet.
- Jeder Zustand, für den das `End`-Feld `true` ist, gilt als ein `end`- (oder `terminal`-)Zustand. Abhängig von Ihrer Zustandsmaschinen-Logik — wenn Ihre Zustandsmaschine beispielsweise mehrere Ausführungszweige hat — haben Sie möglicherweise mehr als einen Status. `end`
- Wenn Ihr Zustandsautomat nur aus einem Zustand besteht, kann das sowohl der `start`- als auch der `end`-Zustand sein.

Intrinsische Funktionen

Die Sprache von Amazon States bietet mehrere systeminterne Funktionen, auch als systeminterne Funktionen bezeichnet, mit denen Sie grundlegende Datenverarbeitungsvorgänge durchführen können, ohne einen Status zu verwenden. `Task` Intrinsics sind Konstrukte, die Funktionen in Programmiersprachen ähneln. Sie können verwendet werden, um Payload Builder bei der Verarbeitung der Daten zu unterstützen, die zum und vom `Resource` Feld eines Bundesstaates gesendet werden. `Task`

In Amazon States Language werden systeminterne Funktionen je nach Art der Datenverarbeitungsaufgabe, die Sie ausführen möchten, in die folgenden Kategorien eingeteilt:

- Intrinsische Eigenschaften für Arrays
- Intrinsische Funktionen für die Datenkodierung und -dekodierung
- Intrinsisch für die Hash-Berechnung

- Intrinsik für die Manipulation von JSON-Daten
- Intrinsik für mathematische Operationen
- Systemimmanent für die String-Operation
- Intrinsisch für die Generierung eindeutiger Identifikatoren
- Intrinsisch für den generischen Betrieb

> **Note**
>
> - Um systeminterne Funktionen verwenden zu können, müssen Sie den Schlüsselwert `.$` in Ihren State-Machine-Definitionen angeben, wie im folgenden Beispiel gezeigt:
>
> ```
> "KeyId.$": "States.Array($.Id)"
> ```
>
> - Sie können in Ihren Workflows bis zu 10 systeminterne Funktionen innerhalb eines Felds verschachteln. Das folgende Beispiel zeigt ein Feld mit dem Namen*myArn*, das neun verschachtelte systeminterne Funktionen enthält:
>
> ```
> "myArn.$": "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**
>
> Wenn Sie Step Functions in einer lokalen Entwicklungsumgebung verwenden, stellen Sie sicher, dass Sie Version 1.12.0 oder höher verwenden, um alle intrinsischen Funktionen in Ihre Workflows aufnehmen zu können.

Felder, die systeminterne Funktionen unterstützen

Die folgende Tabelle zeigt, welche Felder systemeigene Funktionen für jeden Status unterstützen.

Felder, die systeminterne Funktionen unterstützen

State	Passieren	Aufgabe	Wahl	Wait	Erfolgreich sein	Scheitern	Parallel	Zuordnung
InputPath								
Parameter	✓	✓					✓	✓
ResultSelector		✓					✓	✓
ResultPath								
OutputPath								
Variable								
<ComparisonOperator>								
Pfad								
TimeoutSecondsPath								
HeartbeatSecondsPath								
Anmeldeinformationen		✓						

Intrinsische Eigenschaften für Arrays

Verwenden Sie die folgenden systeminternen Merkmale, um Array-Manipulationen durchzuführen.

`States.Array`

Die `States.Array` systeminterne Funktion benötigt null oder mehr Argumente. Der Interpreter gibt ein JSON-Array zurück, das die Werte der Argumente in der angegebenen Reihenfolge enthält. Beispielsweise angesichts der folgenden Eingabe:

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

Du könntest verwenden

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

Was das folgende Ergebnis zurückgeben würde:

```
“BuildId”: [123456]
```

`States.ArrayPartition`

Verwenden Sie die `States.ArrayPartition` systemeigene Funktion, um ein großes Array zu partitionieren. Sie können diese systeminterne Funktion auch verwenden, um die Daten zu segmentieren und dann die Nutzdaten in kleinere Teile zu senden.

Diese systeminterne Funktion benötigt zwei Argumente. Das erste Argument ist ein Array, während das zweite Argument die Chunk-Größe definiert. Der Interpreter teilt das Eingabearray in mehrere Arrays der durch die Chunk-Größe angegebenen Größe auf. Die Länge des letzten Array-Chunks kann geringer sein als die Länge der vorherigen Array-Chunks, wenn die Anzahl der verbleibenden Elemente im Array kleiner als die Chunk-Größe ist.

Überprüfung der Eingabe

- Sie müssen ein Array als Eingabewert für das erste Argument der Funktion angeben.
- Sie müssen für das zweite Argument, das den Chunk-Größenwert darstellt, eine positive Ganzzahl ungleich Null angeben.

Wenn Sie für das zweite Argument einen Wert angeben, der keine Ganzzahl ist, rundet Step Functions ihn auf die nächste Ganzzahl ab.

- Das Eingabearray darf die Payload-Größenbeschränkung von Step Functions von 256 KB nicht überschreiten.

Nehmen wir zum Beispiel das folgende Eingabe-Array an:

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

Sie könnten die `States.ArrayPartition` Funktion verwenden, um das Array in Blöcke mit vier Werten zu unterteilen:

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

Was die folgenden Array-Blöcke zurückgeben würde:

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

Im vorherigen Beispiel gibt die `States.ArrayPartition` Funktion drei Arrays aus. Die ersten beiden Arrays enthalten jeweils vier Werte, die durch die Chunk-Größe definiert sind. Ein drittes Array enthält den verbleibenden Wert und ist kleiner als die definierte Chunk-Größe.

`States.ArrayContains`

Verwenden Sie die `States.ArrayContains` systeminterne Funktion, um festzustellen, ob ein bestimmter Wert in einem Array vorhanden ist. Sie können diese Funktion beispielsweise verwenden, um festzustellen, ob bei einer `Map` Status-Iteration ein Fehler aufgetreten ist.

Diese systeminterne Funktion benötigt zwei Argumente. Das erste Argument ist ein Array, während das zweite Argument der Wert ist, nach dem innerhalb des Arrays gesucht werden soll.

Überprüfung der Eingabe

- Sie müssen ein Array als Eingabewert für das erste Argument der Funktion angeben.
- Sie müssen ein gültiges JSON-Objekt als zweites Argument angeben.
- Das Eingabearray darf die Payload-Größenbeschränkung von Step Functions von 256 KB nicht überschreiten.

Nehmen wir zum Beispiel das folgende Eingabe-Array an:

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

Sie könnten die `States.ArrayContains` Funktion verwenden, um den `lookingFor` Wert innerhalb von zu findeninputArray:

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

Da der in gespeicherte Wert in enthalten `lookingFor` istinputArray, wird das folgende Ergebnis `States.ArrayContains` zurückgegeben:

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

States.ArrayRange

Verwenden Sie die `States.ArrayRange` systeminterne Funktion, um ein neues Array zu erstellen, das einen bestimmten Bereich von Elementen enthält. Das neue Array kann bis zu 1000 Elemente enthalten.

Diese Funktion benötigt drei Argumente. Das erste Argument ist das erste Element des neuen Arrays, das zweite Argument ist das letzte Element des neuen Arrays und das dritte Argument ist der Inkrementwert zwischen den Elementen im neuen Array.

Überprüfung der Eingabe

- Sie müssen für alle Argumente Ganzzahlwerte angeben.

  Wenn Sie für eines der Argumente einen Wert angeben, der keine Ganzzahl ist, rundet Step Functions ihn auf die nächste Ganzzahl ab.
- Sie müssen für das dritte Argument einen Wert ungleich Null angeben.
- Das neu generierte Array kann nicht mehr als 1000 Elemente enthalten.

Bei der folgenden Verwendung der `States.ArrayRange` Funktion wird beispielsweise ein Array mit einem ersten Wert von 1 und einem Endwert von 9 erstellt, und die Werte zwischen dem ersten und dem letzten Wert werden für jedes Element um zwei erhöht:

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

Was das folgende Array zurückgeben würde:

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

States.ArrayGetItem

Diese intrinsische Funktion gibt den Wert eines angegebenen Indexes zurück. Diese Funktion benötigt zwei Argumente. Das erste Argument ist ein Array von Werten und das zweite Argument ist der Array-Index des zurückzugebenden Werts.

Verwenden Sie beispielsweise die folgenden `index` Werte `inputArray` und:

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

Aus diesen Werten können Sie die `States.ArrayGetItem` Funktion verwenden, um den Wert an der `index` Position 5 innerhalb des Arrays zurückzugeben:

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

In diesem Beispiel `States.ArrayGetItem` würde das folgende Ergebnis zurückgegeben werden:

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

States.ArrayLength

Die `States.ArrayLength` systeminterne Funktion gibt die Länge eines Arrays zurück. Sie hat ein Argument, das Array, dessen Länge zurückgegeben werden soll.

Zum Beispiel bei dem folgenden Eingabe-Array:

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

Sie können verwenden`States.ArrayLength`, um die Länge zurückzugeben von`inputArray`:

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

In diesem Beispiel `States.ArrayLength` würde das folgende JSON-Objekt zurückgegeben, das die Array-Länge darstellt:

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

`States.ArrayUnique`

Die `States.ArrayUnique` systeminterne Funktion entfernt doppelte Werte aus einem Array und gibt ein Array zurück, das nur eindeutige Elemente enthält. Diese Funktion verwendet ein Array, das unsortiert sein kann, als einziges Argument.

Der folgende Code `inputArray` enthält beispielsweise eine Reihe von doppelten Werten:

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

Sie könnten die `States.ArrayUnique` Funktion als verwenden und das Array angeben, aus dem Sie doppelte Werte entfernen möchten:

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

Die `States.ArrayUnique` Funktion würde das folgende Array zurückgeben, das nur eindeutige Elemente enthält, und alle doppelten Werte entfernen:

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

Intrinsische Funktionen für die Datenkodierung und -dekodierung

Verwenden Sie die folgenden systemeigenen Funktionen, um Daten auf der Grundlage des Base64-Kodierungsschemas zu codieren oder zu dekodieren.

`States.Base64Encode`

Verwenden Sie die `States.Base64Encode` systeminterne Funktion, um Daten auf der Grundlage des MIME Base64-Kodierungsschemas zu codieren. Sie können diese Funktion

verwenden, um Daten an andere AWS Dienste zu übergeben, ohne eine Funktion zu verwenden. AWS Lambda

Diese Funktion benötigt als einziges Argument eine Datenzeichenfolge mit bis zu 10.000 Zeichen zum Kodieren.

Stellen Sie sich zum Beispiel die folgende `input` Zeichenfolge vor:

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

Sie können die `States.Base64Encode` Funktion verwenden, um die `input` Zeichenfolge als MIME-Base64-Zeichenfolge zu codieren:

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

Die `States.Base64Encode` Funktion gibt als Antwort die folgenden codierten Daten zurück:

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

`States.Base64Decode`

Verwenden Sie die `States.Base64Decode` systeminterne Funktion, um Daten auf der Grundlage des MIME Base64-Dekodierungsschemas zu dekodieren. Sie können diese Funktion verwenden, um Daten an andere AWS Dienste zu übergeben, ohne eine Lambda-Funktion zu verwenden.

Diese Funktion verwendet als einziges Argument eine Base64-kodierte Datenzeichenfolge mit bis zu 10.000 Zeichen für die Dekodierung.

Beispielsweise angesichts der folgenden Eingabe:

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

Sie können die `States.Base64Decode` Funktion verwenden, um die Base64-Zeichenfolge in eine für Menschen lesbare Zeichenfolge zu dekodieren:

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

Das `States.Base64Decode function` würde als Antwort die folgenden dekodierten Daten zurückgeben:

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

Intrinsisch für die Hash-Berechnung

`States.Hash`

Verwenden Sie die `States.Hash` systeminterne Funktion, um den Hashwert einer bestimmten Eingabe zu berechnen. Sie können diese Funktion verwenden, um Daten an andere AWS Dienste zu übergeben, ohne eine Lambda-Funktion zu verwenden.

Diese Funktion benötigt zwei Argumente. Das erste Argument sind die Daten, deren Hashwert Sie berechnen möchten. Das zweite Argument ist der Hash-Algorithmus, der zur Durchführung der Hash-Berechnung verwendet werden soll. Bei den von Ihnen angegebenen Daten muss es sich um eine Objektzeichenfolge mit 10.000 Zeichen oder weniger handeln.

Bei dem von Ihnen angegebenen Hash-Algorithmus kann es sich um einen der folgenden Algorithmen handeln:

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

Sie können diese Funktion beispielsweise verwenden, um den Hashwert der `Data` Zeichenfolge anhand der angegebenen `Algorithm` Werte zu berechnen:

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

Sie können die `States.Hash` Funktion verwenden, um den Hashwert zu berechnen:

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

Die `States.Hash` Funktion gibt als Antwort den folgenden Hashwert zurück:

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

Intrinsik für die Manipulation von JSON-Daten

Verwenden Sie diese Funktionen, um grundlegende Datenverarbeitungsvorgänge an JSON-Objekten durchzuführen.

`States.JsonMerge`

Verwenden Sie die `States.JsonMerge` systeminterne Funktion, um zwei JSON-Objekte zu einem einzigen Objekt zusammenzuführen. Diese Funktion benötigt drei Argumente. Die ersten beiden Argumente sind die JSON-Objekte, die Sie zusammenführen möchten. Das dritte Argument ist ein boolescher Wert von. `false` Dieser boolesche Wert bestimmt, ob der Deep Merging-Modus aktiviert ist.

Derzeit unterstützt Step Functions nur den Shallow-Merging-Modus. Daher müssen Sie den booleschen Wert als angeben. `false` Wenn im Shallow-Modus derselbe Schlüssel in beiden JSON-Objekten vorhanden ist, überschreibt der Schlüssel des letzteren Objekts denselben Schlüssel im ersten Objekt. Außerdem werden Objekte, die in einem JSON-Objekt verschachtelt sind, nicht zusammengeführt, wenn Sie Shallow Merging verwenden.

Sie können die `States.JsonMerge` Funktion beispielsweise verwenden, um die folgenden JSON-Objekte zusammenzuführen, die den Schlüssel gemeinsam haben. a

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

Sie können die Objekte json1 und json2 als Eingaben in der `States.JsonMerge` Funktion angeben, um sie zusammenzuführen:

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

Das `States.JsonMerge` gibt das folgende zusammengeführte JSON-Objekt als Ergebnis zurück. Im zusammengeführten JSON-Objekt `output` a ersetzt der `json2` Schlüssel des `json1` Objekts den Schlüssel des Objektsa. Außerdem a wird das verschachtelte Objekt im `json1` Objektschlüssel verworfen, da der flache Modus das Zusammenführen verschachtelter Objekte nicht unterstützt.

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

States.StringToJson

Die `States.StringToJson` Funktion verwendet als einziges Argument einen Referenzpfad zu einer maskierten JSON-Zeichenfolge.

Der Interpreter wendet einen JSON-Parser an und gibt das geparste JSON-Formular der Eingabe zurück. Sie können diese Funktion beispielsweise verwenden, um die folgende Eingabezeichenfolge zu maskieren:

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

Verwenden Sie die `States.StringToJson` Funktion und geben Sie das `escapedJsonString` als Eingabeargument an:

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

Die `States.StringToJson` Funktion gibt das folgende Ergebnis zurück:

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

States.JsonToString

Die `States.JsonToString` Funktion benötigt nur ein Argument, nämlich den Pfad, der die JSON-Daten enthält, die als Zeichenfolge ohne Escape-Zeichen zurückgegeben werden sollen. Der Interpreter gibt eine Zeichenfolge zurück, die JSON-Text enthält, der die durch den Pfad angegebenen Daten darstellt. Sie können beispielsweise den folgenden JSON-Pfad angeben, der einen Escape-Wert enthält:

```
{
```

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

Stellen Sie der `States.JsonToString` Funktion die darin enthaltenen Daten zur Verfügung`unescapedJson`:

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

Die `States.JsonToString` Funktion gibt die folgende Antwort zurück:

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

Intrinsik für mathematische Operationen

Verwenden Sie diese Funktionen, um mathematische Operationen auszuführen.

`States.MathRandom`

Verwenden Sie die `States.MathRandom` systeminterne Funktion, um eine Zufallszahl zwischen der angegebenen Startzahl (einschließlich) und der Endzahl (ausschließlich) zurückzugeben.

Sie können diese Funktion verwenden, um eine bestimmte Aufgabe auf zwei oder mehr Ressourcen zu verteilen.

Diese Funktion benötigt drei Argumente. Das erste Argument ist die Startnummer, das zweite Argument ist die Endnummer und das letzte Argument steuert den Startwert. Das Startwert-Argument ist optional. Wenn Sie diese Funktion mit demselben Ausgangswert verwenden, gibt sie eine identische Zahl zurück.

> **Important**
>
> Da die `States.MathRandom` Funktion keine kryptografisch sicheren Zufallszahlen zurückgibt, empfehlen wir, sie nicht für sicherheitsrelevante Anwendungen zu verwenden.

Überprüfung der Eingabe

- Sie müssen Ganzzahlwerte für die Argumente Startnummer und Endnummer angeben.

  Wenn Sie für das Argument Startnummer oder Endnummer einen Wert angeben, der keine Ganzzahl ist, rundet Step Functions diesen Wert auf die nächste Ganzzahl ab.

Um beispielsweise eine Zufallszahl zwischen eins und 999 zu generieren, können Sie die folgenden Eingabewerte verwenden:

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

Um die Zufallszahl zu generieren, geben Sie die end Werte `start` und für die `States.MathRandom` Funktion ein:

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

Die `States.MathRandom` Funktion gibt die folgende Zufallszahl als Antwort zurück:

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

`States.MathAdd`

Verwenden Sie die `States.MathAdd` systeminterne Funktion, um die Summe zweier Zahlen zurückzugeben. Sie können diese Funktion beispielsweise verwenden, um Werte innerhalb einer Schleife zu erhöhen, ohne eine Lambda-Funktion aufzurufen.

Überprüfung der Eingabe

- Sie müssen Ganzzahlwerte für alle Argumente angeben.

  Wenn Sie für eines oder beide Argumente einen Wert angeben, der keine Ganzzahl ist, rundet Step Functions ihn auf die nächste Ganzzahl ab.
- Sie müssen Ganzzahlwerte im Bereich von -2147483648 und 2147483647 angeben.

Sie können beispielsweise die folgenden Werte verwenden, um eins von 111 zu subtrahieren:

```
{
```

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

Verwenden Sie dann die `States.MathAdd` Funktion definierend `value1` als Startwert und `step` als Wert, um den Sie erhöht werden sollen: `value1`

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

Die `States.MathAdd` Funktion würde als Antwort die folgende Zahl zurückgeben:

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

Systemimmanent für die String-Operation

`States.StringSplit`

Verwenden Sie die `States.StringSplit` systeminterne Funktion, um eine Zeichenfolge in ein Array von Werten aufzuteilen. Diese Funktion benötigt zwei Argumente. Das erste Argument ist eine Zeichenfolge und das zweite Argument ist das Trennzeichen, mit dem die Funktion die Zeichenfolge teilt.

Example - Teilt eine Eingabezeichenfolge mit einem einzigen Trennzeichen

Verwenden Sie in diesem Beispiel, `States.StringSplit` um Folgendes zu teilen`inputString`, das eine Reihe von durch Kommas getrennten Werten enthält:

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

Verwenden Sie die `States.StringSplit` Funktion und definieren Sie `inputString` als erstes Argument und das Trennzeichen `splitter` als zweites Argument:

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

Die `States.StringSplit` Funktion gibt als Ergebnis das folgende String-Array zurück:

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

Example - Teilt eine Eingabezeichenfolge mit mehreren Trennzeichen auf

Verwenden Sie in diesem Beispiel, `States.StringSplit` um Folgendes zu unterteilen`inputString`, das mehrere Trennzeichen enthält:

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

Verwenden Sie die `States.StringSplit` Funktion wie folgt:

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

Die `States.StringSplit` Funktion gibt als Ergebnis das folgende String-Array zurück:

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

Intrinsisch für die Generierung eindeutiger Identifikatoren

`States.UUID`

Verwenden Sie die `States.UUID` systeminterne Funktion, um einen mit Zufallszahlen generierten Universally Unique Identifier (v4-UUID) zurückzugeben. Sie können diese Funktion beispielsweise verwenden, um andere AWS Dienste oder Ressourcen aufzurufen, die einen UUID-Parameter benötigen, oder Elemente in eine DynamoDB-Tabelle einzufügen.

Die `States.UUID` Funktion wird ohne Angabe von Argumenten aufgerufen:

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

Die Funktion gibt eine zufällig generierte UUID zurück, wie im folgenden Beispiel:

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

Intrinsisch für den generischen Betrieb

`States.Format`

Verwenden Sie die `States.Format` systeminterne Funktion, um eine Zeichenfolge sowohl aus literalen als auch aus interpolierten Werten zu erstellen. Diese Funktion benötigt ein oder mehrere Argumente. Der Wert des ersten Arguments muss eine Zeichenfolge sein und kann null oder mehr Instanzen der Zeichenfolge enthalten`{}`. Der Aufruf des intrinsischen Elements muss so viele verbleibende Argumente enthalten, wie es vorkommen kann. `{}` Der Interpreter gibt die im ersten Argument definierte Zeichenfolge zurück, wobei jedes Argument im systeminternen Aufruf durch den Wert des der Position entsprechenden Arguments `{}` ersetzt wird.

Sie können beispielsweise die folgenden Eingaben einer Person und einen `template` Satz verwendenname, in den ihr Name eingefügt werden soll:

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

Verwenden Sie die `States.Format` Funktion und geben Sie die `template` Zeichenfolge und die Zeichenfolge an, die anstelle der `{}` Zeichen eingefügt werden sollen:

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

oder

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

Bei einer der vorherigen Eingaben gibt die `States.Format` Funktion als Antwort die vollständige Zeichenfolge zurück:

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

Reservierte Zeichen in systemeigenen Funktionen

Die folgenden Zeichen sind für systeminterne Funktionen reserviert und müssen mit einem umgekehrten Schrägstrich ('\') maskiert werden, wenn sie im Wert vorkommen sollen:, und. ' { } \

Wenn das Zeichen als Teil des Werts erscheinen \ soll, ohne als Escape-Zeichen zu dienen, müssen Sie es mit einem umgekehrten Schrägstrich maskieren. Die folgenden Escape-Zeichenfolgen werden mit systemeigenen Funktionen verwendet:

- Die Literalzeichenfolge steht für. \' '
- Die Literalzeichenfolge \{ steht für. {
- Die Literalzeichenfolge \} steht für. }
- Die Literalzeichenfolge \\ steht für. \

In JSON müssen Backslashes, die in einem Zeichenfolgenliteralwert enthalten sind, mit einem anderen Backslash maskiert werden. Die entsprechende Liste für JSON lautet:

- Die maskierte Zeichenfolge \\\' steht für\'.
- Die maskierte Zeichenfolge \\\{ steht für\{.
- Die maskierte Zeichenfolge \\\} steht für\}.
- Die maskierte Zeichenfolge \\\\ steht für\\.

> **Note**
>
> Wenn in der systemeigenen Aufrufzeichenfolge ein offener Escape-Backslash gefunden \ wird, gibt der Interpreter einen Laufzeitfehler zurück.

Häufige Zustandsfelder

`Type` (Erforderlich)

Der Zustandstyp.

`Next`

Der Name des nächsten Zustands, der ausgeführt wird, wenn der aktuelle Zustand beendet ist. Einige Zustandstypen, z. B. `Choice`, erlauben mehrere Übergangszustände.

Wenn der aktuelle Status der letzte Status in Ihrem Workflow oder ein Endstatus wie Succeed oder istFehler, müssen Sie das `Next` Feld nicht angeben.

`End`

Gibt an, ob dieser Zustand ein Beendigungszustand ist (er beendet die Ausführung), wenn auf `true` gesetzt. Es kann eine beliebige Anzahl von Beendigungszuständen pro Zustandsautomaten geben. Von `Next` oder `End` kann nur einer in einem Zustand verwendet werden. Einige ZustandstypenChoice, wie z. B. oder Endzustände wie Succeed undFehler, unterstützen das `End` Feld nicht und verwenden es nicht.

`Comment` (Optional)

Enthält eine für Menschen lesbare Beschreibung des Zustands.

`InputPath` (Optional)

Ein Pfad der einen Teil der Eingabe des Zustands auswählt, der an die Aufgabe des Zustands zur Verarbeitung übergeben werden soll. Wenn dieses Attribut nicht angegeben wird, hat es den Wert $, der die gesamte Eingabe bezeichnet. Weitere Informationen finden Sie unter Verarbeitung von Eingabe und Ausgabe.

`OutputPath` (Optional)

Ein Pfad, der einen Teil der Ausgabe des Status auswählt, der an den nächsten Status übergeben werden soll. Wenn es weggelassen wird, hat es den Wert$, der die gesamte Ausgabe bezeichnet. Weitere Informationen finden Sie unter Verarbeitung von Eingabe und Ausgabe.

Pass

Ein `Pass`-Zustand (`"Type": "Pass"`) gibt die Eingabe an die Ausgabe weiter und führt keine Arbeit aus. `Pass`-Zustände sind nützlich beim Erstellen und Debuggen von Zustandsautomaten.

Sie können auch einen `Pass` Status verwenden, um die JSON-Statuseingabe mithilfe von Filtern zu transformieren und die transformierten Daten dann an den nächsten Status in Ihren Workflows zu übergeben. Hinweise zur Eingabetransformation finden Sie unterInputPath, Parameter und ResultSelector.

Zusätzlich zu den allgemeinen Zustandsfeldern erlauben `Pass`-Zustände die folgenden Felder.

`Result` (Optional)

Bezieht sich auf die Ausgabe einer virtuellen Aufgabe, die an den nächsten Status weitergegeben wird. Wenn Sie das `ResultPath` Feld in Ihren Status aufnehmen, `Result` wird die Maschinendefinition so platziert, wie von angegeben, `ResultPath` und an den nächsten Status weitergeleitet.

`ResultPath` (Optional)

Gibt an, wo die Ausgabe (relativ zur Eingabe) der angegebenen virtuellen Aufgabe platziert werden soll`Result`. Die Eingabe wird wie vom Feld `OutputPath` festgelegt gefiltert (falls vorhanden), bevor sie als Ausgabe des Zustands verwendet wird. Weitere Informationen finden Sie unter Verarbeitung von Eingabe und Ausgabe.

`Parameters` (Optional)

Erzeugt eine Sammlung von Schlüssel-Wert-Paaren, die als Eingabe übergeben werden. Sie können ihn `Parameters` als statischen Wert angeben oder mithilfe eines Pfads aus der Eingabe auswählen. Weitere Informationen finden Sie unter InputPath, Parameter und ResultSelector.

Beispiel für den Pass-Zustand

Hier finden Sie ein Beispiel eines `Pass`-Zustands, der einige feste Daten in den Zustandsautomaten einfügt, wahrscheinlich zu Testzwecken.

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

Angenommen, die Eingabe in diesen Zustand ist die folgende.

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

```
}
```

Dann wäre die Ausgabe wie folgt.

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

Status der Aufgabe

Ein `Task`-Zustand (`"Type": "Task"`) steht für eine einzelne Arbeitseinheit, die von einem Zustandsautomaten durchgeführt wird. Eine Aufgabe erledigt Aufgaben mithilfe einer Aktivität oder AWS Lambda Funktion, durch Integration mit anderen unterstützten AWS-Services APIs oder durch Aufrufen einer Drittanbieter-API wie Stripe.

Die Sprache Amazon States stellt Aufgaben dar, indem sie den Typ eines Zustands auf setzt `Task` und der Aufgabe den Amazon-Ressourcennamen (ARN) der Aktivität, die Lambda-Funktion oder den API-Endpunkt eines Drittanbieters bereitstellt. Die folgende Task-State-Definition ruft eine Lambda-Funktion mit dem Namen auf. *`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
}
```

In diesem Thema

- Aufgabentypen
- Felder für den Aufgabenstatus
- Beispiele für die Definition von Aufgabenstatus

- Aktivitäten

Aufgabentypen

Step Functions unterstützt die folgenden Aufgabentypen, die Sie in einer Aufgabenstatusdefinition angeben können:

- Aktivität
- Lambda-Funktionen
- A unterstützt AWS-Service
- Eine HTTP-Aufgabe

Sie geben einen Aufgabentyp an, indem Sie seinen ARN im `Resource` Feld einer Aufgabenstatusdefinition angeben. Das folgende Beispiel zeigt die Syntax des `Resource` Felds. Alle Task-Typen außer dem, der eine Drittanbieter-API aufruft, verwenden die folgende Syntax. Hinweise zur Syntax des HTTP-Tasks finden Sie unterRufen Sie APIs von Drittanbietern auf.

Ersetzen Sie in Ihrer Aufgabenstatusdefinition den kursiv geschriebenen Text in der folgenden Syntax durch die AWS ressourcenspezifischen Informationen.

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

In der folgenden Liste werden die einzelnen Komponenten dieser Syntax erklärt:

- `partition`ist die am häufigsten zu verwendende AWS Step Functions Partition`aws`.
- `service`gibt den Wert an, der zur Ausführung der Aufgabe AWS-Service verwendet wurde, und kann einer der folgenden Werte sein:
  - `states` für eine Aktivität.
  - `lambda`für eine Lambda-Funktion. Wenn Sie eine Integration mit anderen AWS-Services, z. B. Amazon SNS oder Amazon DynamoDB, durchführen, verwenden Sie oder. `sns` `dynamodb`
- `region`ist der AWS Regionalcode, in dem die Step Functions Functions-Aktivität oder der Zustandsmaschinentyp, die Lambda-Funktion oder eine andere AWS Ressource erstellt wurde.
- `account`ist die AWS-Konto ID, in der Sie die Ressource definiert haben.
- `task_type` ist der Typ der auszuführenden Aufgabe. Dabei kann es sich um einen der folgenden Werte handeln:

  - `activity`— Eine Aktivität.
  - `function`— Eine Lambda-Funktion.
  - *`servicename`*— Der Name eines unterstützten verbundenen Dienstes (siehe Optimierte Integrationen für Step Functions).
- `name`ist der registrierte Ressourcenname (Aktivitätsname, Lambda-Funktionsname oder Service-API-Aktion).

> **Note**
>
> Step Functions unterstützt das Verweisen auf ARNs über Partitionen oder Regionen hinweg nicht. `aws-cn`Kann beispielsweise keine Aufgaben in der `aws` Partition aufrufen und umgekehrt.

In den folgenden Abschnitten erhalten Sie weitere Informationen zu den einzelnen Aufgabentypen.

Aktivität

Aktivitäten repräsentieren Worker (Prozesse oder Threads), die von Ihnen implementiert und gehostet werden, die eine bestimmte Aufgabe ausführen. Sie werden nur von Standard-Workflows und nicht von Express-Workflows unterstützt.

`Resource`-ARNs von Aktivitäten verwenden die folgende Syntax.

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

> **Note**
>
> Sie müssen Aktivitäten mit Step Functions (mithilfe einer CreateActivityAPI-Aktion oder der Step Functions Functions-Konsole) erstellen, bevor sie zum ersten Mal verwendet werden.

Weitere Informationen zum Erstellen einer Aktivität und zum Implementieren von Workern finden Sie unter Activities.

Lambda-Funktionen

Lambda-Aufgaben führen eine Funktion aus mit AWS Lambda. Um eine Lambda-Funktion anzugeben, verwenden Sie den ARN der Lambda-Funktion im `Resource` Feld.

Abhängig von der Art der Integration (Optimierte Integration oder AWS SDK-Integration), die Sie für die Angabe einer Lambda-Funktion verwenden, variiert die Syntax des `Resource` Felds Ihrer Lambda-Funktion.

Die folgende `Resource` Feldsyntax ist ein Beispiel für eine optimierte Integration mit einer Lambda-Funktion.

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

Die folgende `Resource` Feldsyntax ist ein Beispiel für eine AWS SDK-Integration mit einer Lambda-Funktion.

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

Die folgende `Task` Zustandsdefinition zeigt ein Beispiel für eine optimierte Integration mit einer Lambda-Funktion namens*`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"
}
```

Nachdem die im `Resource` Feld angegebene Lambda-Funktion abgeschlossen ist, wird ihre Ausgabe in den im `Next` Feld angegebenen Status (“ NextState „) gesendet.

Ein unterstützter AWS-Service

Wenn Sie auf eine verbundene Ressource verweisen, ruft Step Functions direkt die API-Aktionen eines unterstützten Dienstes auf. Geben Sie im `Resource`-Feld den Service und die Aktion an.

Die `Resource`-ARNs von verbundenen Services verwenden die folgende Syntax.

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

> **Note**
>
> Um eine synchrone Verbindung zu einer verbundenen Ressource herzustellen, fügen Sie `.sync` zum *APIname*-Eintrag im ARN an. Weitere Informationen finden Sie unter Arbeiten mit anderen -Services.

Beispielsweise:

```
{
 "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
    }
  }
}
```

Felder für den Aufgabenstatus

Zusätzlich zu den allgemeinen Zustandsfeldern haben `Task`-Zustände die folgenden Felder.

`Resource` (Erforderlich)

Ein URI, vor allem ein ARN, der bestimmte auszuführende Aufgaben identifiziert.

`Parameters` (Optional)

Wird für die Weitergabe von Informationen zu den API-Aktionen von verbundenen Ressourcen verwendet. Die Parameter können eine Mischung aus statischem JSON und verwenden JsonPath. Weitere Informationen finden Sie unter Parameter an eine Service-API übergeben.

`Credentials` (Optional)

Gibt eine Zielrolle an, die die Ausführungsrolle der Zustandsmaschine übernehmen muss, bevor die angegebene `Resource` Rolle aufgerufen wird. Alternativ können Sie auch einen JSONPath-Wert oder eine systeminterne Funktion angeben, die zur Laufzeit auf der Grundlage der Ausführungseingabe in einen IAM-Rollen-ARN aufgelöst wird. Wenn Sie einen JSONPath-Wert angeben, müssen Sie ihm die Notation voranstellen. `$.`

Beispiele für die Verwendung dieses Felds im `Task` Bundesstaat finden Sie unter. Beispiele für das Feld Anmeldeinformationen für den Aufgabenstatus Ein Beispiel für die Verwendung dieses Felds für den Zugriff auf eine kontoübergreifende AWS Ressource von Ihrem Zustandsmaschine aus finden Sie unterTutorial: Zugriff auf kontoübergreifende Ressourcen AWS.

> **Note**
>
> Dieses Feld wird von denAufgabentypen, die Lambda-Funktionen verwenden, und von einem unterstützten AWS Dienst unterstützt.

`ResultPath` (Optional)

Gibt an, wohin (in der Eingabe) die Ergebnisse der Ausführung der in `Resource` angegebenen Aufgabe zu platzieren sind. Die Eingabe wird dann wie vom Feld `OutputPath` festgelegt gefiltert (falls vorhanden), bevor sie als Ausgabe des Zustands verwendet wird. Weitere Informationen finden Sie unter Verarbeitung von Eingabe und Ausgabe.

`ResultSelector` (Optional)

Übergeben Sie eine Sammlung von Schlüsselwertpaaren, wobei die Werte statisch sind oder aus dem Ergebnis ausgewählt werden. Weitere Informationen finden Sie unter ResultSelector.

`Retry` (Optional)

Ein Array von Objekten namens Retrier, die eine Wiederholungsrichtlinie für den Fall definieren, dass der Zustand auf Laufzeitfehler trifft. Weitere Informationen finden Sie unter Beispiele für Zustandsmaschinen mit Retry und Catch.

`Catch` (Optional)

Ein Array von Objekten namens Catcher, die einen Fallback-Zustand definieren. Dieser Zustand wird ausgeführt, falls der Zustand auf Laufzeitfehler trifft und seine Wiederholungsrichtlinie ausgeschöpft wurde oder nicht definiert ist. Weitere Informationen finden Sie unter Fallback-Zustände.

`TimeoutSeconds` (Optional)

Gibt die maximale Zeit an, für die eine Aktivität oder eine Aufgabe ausgeführt werden kann, bevor das Timeout mit dem States.Timeout Fehler auftritt und fehlschlägt. Der Timeout-Wert muss eine positive Ganzzahl ungleich Null sein. Der Standardwert ist 99999999.

Die Anzahl der Timeouts beginnt, nachdem eine Aufgabe gestartet wurde, z. B. wenn `ActivityStarted` `LambdaFunctionStarted` Ereignisse im Verlauf der Ausführungsereignisse protokolliert werden. Bei Aktivitäten beginnt die Zählung mit dem `GetActivityTask` Empfang eines Tokens und `ActivityStarted` wird im Verlauf der Ausführungsereignisse protokolliert.

Wenn eine Aufgabe gestartet wird, wartet Step Functions innerhalb der angegebenen `TimeoutSeconds` Dauer auf eine Erfolgs- oder Fehlschlagsantwort des Aufgaben- oder Aktivitätsarbeiters. Wenn der Aufgaben- oder Aktivitätsarbeiter innerhalb dieser Zeit nicht reagiert, markiert Step Functions die Workflow-Ausführung als fehlgeschlagen.

`TimeoutSecondsPath` (Optional)

Wenn Sie einen Timeout-Wert dynamisch aus der Statuseingabe mithilfe eines Referenzpfads bereitstellen möchten, verwenden Sie`TimeoutSecondsPath`. Wenn das Problem gelöst ist, muss der Referenzpfad Felder auswählen, deren Werte positive ganze Zahlen sind.

> **Note**
>
> Ein `Task` Status kann nicht sowohl als auch `TimeoutSeconds` enthalten. `TimeoutSecondsPath`

`HeartbeatSeconds` (Optional)

Bestimmt die Frequenz der Heartbeat-Signale, die ein Activity Worker während der Ausführung einer Aufgabe sendet. Heartbeats zeigen an, dass eine Aufgabe noch ausgeführt wird und mehr

Zeit benötigt, bis sie abgeschlossen ist. Heartbeats verhindern, dass bei einer Aktivität oder Aufgabe innerhalb der Dauer ein Timeout auftritt. `TimeoutSeconds`

`HeartbeatSeconds`muss ein positiver Ganzzahlwert ungleich Null sein, der unter dem `TimeoutSeconds` Feldwert liegt. Der Standardwert ist 99999999. Wenn zwischen den Heartbeats der Aufgabe mehr Zeit als die angegebenen Sekunden vergeht, schlägt der Task-Status mit einem Fehler fehl. States.Timeout

Bei Aktivitäten beginnt die Zählung mit dem `GetActivityTask` Empfang eines Tokens und `ActivityStarted` wird im Verlauf der Ausführungsereignisse protokolliert.

`HeartbeatSecondsPath` (Optional)

Wenn Sie einen Heartbeat-Wert dynamisch aus der Statuseingabe mithilfe eines Referenzpfads bereitstellen möchten, verwenden Sie`HeartbeatSecondsPath`. Wenn das Problem gelöst ist, muss der Referenzpfad Felder auswählen, deren Werte positive ganze Zahlen sind.

> **Note**
>
> Ein `Task` Status kann nicht sowohl als auch `HeartbeatSeconds` enthalten. `HeartbeatSecondsPath`

Ein `Task`-Zustand muss entweder das Feld `End` auf `true` setzen, wenn der Zustand die Ausführung beendet, oder einen Zustand im Feld `Next` bereitstellen, der nach Abschluss des `Task`-Zustands ausgeführt wird.

Beispiele für die Definition von Aufgabenstatus

Die folgenden Beispiele zeigen, wie Sie die Definition des Aufgabenstatus entsprechend Ihren Anforderungen angeben können.

- Angabe von Timeouts und Heartbeat-Intervallen für den Task-Status
  - Beispiel für ein statisches Timeout und eine Heartbeat-Benachrichtigung
  - Beispiel für ein dynamisches Task-Timeout und eine Heartbeat-Benachrichtigung
- Das Feld „Anmeldeinformationen“ wird verwendet
  - Festcodierte IAM-Rollen-ARN angeben
  - JsonPath als IAM-Rollen-ARN angeben

- Spezifizierung einer systemeigenen Funktion als IAM-Rollen-ARN

Timeouts und Heartbeat-Intervalle für den Aufgabenstatus

Es ist ein bewährtes Verfahren, für langlebige Aktivitäten einen Timeout-Wert und ein Heartbeat-Intervall festzulegen. Dies kann durch Angabe der Timeout- und Heartbeat-Werte oder durch dynamische Einstellung erreicht werden.

Beispiel für ein statisches Timeout und eine Heartbeat-Benachrichtigung

Wenn `HelloWorld` abgeschlossen ist, wird der nächste Zustand (hier `NextState` genannt) ausgeführt.

Wenn diese Aufgabe nicht innerhalb von 300 Sekunden abgeschlossen ist oder keine Heartbeat-Benachrichtigungen in Intervallen von 60 Sekunden sendet, wird die Aufgabe als `failed` gekennzeichnet.

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

Beispiel für ein dynamisches Task-Timeout und eine Heartbeat-Benachrichtigung

In diesem Beispiel wird nach Abschluss des AWS Glue Auftrags der nächste Status ausgeführt.

Wenn diese Aufgabe nicht innerhalb des vom AWS Glue Job dynamisch festgelegten Intervalls abgeschlossen werden kann, wird die Aufgabe als markiert`failed`.

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

```
  "Next": "NextState"
}
```

Beispiele für das Feld Anmeldeinformationen für den Aufgabenstatus

Festcodierte IAM-Rollen-ARN angeben

Das folgende Beispiel spezifiziert eine Ziel-IAM-Rolle, die die Ausführungsrolle einer Zustandsmaschine annehmen muss, um auf eine kontoübergreifende Lambda-Funktion mit dem Namen zuzugreifen. `Echo` In diesem Beispiel wird der ARN der Zielrolle als hartcodierter Wert angegeben.

```
{
  "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
    }
  }
}
```

JsonPath als IAM-Rollen-ARN angeben

Im folgenden Beispiel wird ein JsonPath-Wert angegeben, der zur Laufzeit in einen IAM-Rollen-ARN aufgelöst wird.

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

```
      },
      ...
    }
  }
}
```

Spezifizierung einer systemeigenen Funktion als IAM-Rollen-ARN

Im folgenden Beispiel wird die `States.Format`systeminterne Funktion verwendet, die zur Laufzeit in einen IAM-Rollen-ARN aufgelöst wird.

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

Aktivitäten

Aktivitäten sind eine AWS Step Functions Funktion, mit der Sie eine Aufgabe auf Ihrem State-Computer einrichten können, auf dem die Arbeit von einem Worker ausgeführt wird. Diese Aufgabe kann auf Amazon Elastic Compute Cloud (Amazon EC2), Amazon Elastic Container Service (Amazon ECS) und Mobilgeräten gehostet werden — praktisch überall.

Übersicht

In AWS Step Functions sind Aktivitäten eine Möglichkeit, Code, der irgendwo ausgeführt wird (der sogenannteAktivitäts-Worker) einer bestimmten Aufgabe in einem Zustandsautomaten zuzuordnen. Sie können eine Aktivität mithilfe der Step Functions-Konsole oder durch einen Anruf erstellen`CreateActivity`. Dies stellt einen Amazon-Ressourcennamen (ARN) für Ihren Aufgabenstatus bereit. Verwenden Sie diesen ARN, um den Aufgabenstatus für die Arbeit in Ihrem Aktivitäts-Worker abzurufen.

> **Note**
>
> Aktivitäten werden nicht versioniert und es wird erwartet, dass sie immer abwärtskompatibel sind. Wenn Sie eine abwärtsinkompatible Änderung an einer Aktivität vornehmen müssen, erstellen Sie in Step Functions eine neue Aktivität mit einem eindeutigen Namen.

Ein Activity Worker kann eine Anwendung sein, die auf einer Amazon EC2-Instance ausgeführt wird, eine AWS Lambda Funktion, ein Mobilgerät: jede Anwendung, die eine HTTP-Verbindung herstellen kann und überall gehostet wird. Wenn Step Functions den Status einer Aktivitätsaufgabe erreicht, wartet der Workflow darauf, dass ein Aktivitätsmitarbeiter nach einer Aufgabe fragt. Ein Activity Worker fragt Step Functions ab`GetActivityTask`, indem er den ARN für die entsprechende Aktivität verwendet und sendet. `GetActivityTask`gibt eine Antwort zurück, die `input` (eine Zeichenfolge mit JSON-Eingaben für die Aufgabe) und eine `taskToken`(eine eindeutige Kennung für die Aufgabe) enthält. Nachdem der Aktivitäts-Worker seine Arbeit abgeschlossen hat, kann er einen Bericht über seinen Erfolg oder Misserfolg mithilfe von `SendTaskSuccess` oder `SendTaskFailure` bereitstellen. Diese beiden Aufrufe verwenden das`taskToken`, das von `GetActivityTask` bereitgestellt wurde, um das Ergebnis dieser Aufgabe zuzuordnen.

APIs im Zusammenhang mit Aktivitätsaufgaben

Step Functions bietet APIs zum Erstellen und Auflisten von Aktivitäten, zum Anfordern einer Aufgabe und zum Verwalten des Ablaufs Ihrer State Machine auf der Grundlage der Ergebnisse Ihres Workers.

Im Folgenden sind die Step Functions APIs aufgeführt, die sich auf Aktivitäten beziehen:

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

> **Note**
>
> Das Abrufen von Aktivitätsaufgaben mit `GetActivityTask` kann für einige Implementierungen eine gewisse Latenz verursachen. Siehe Vermeiden Sie Latenz bei der Abfrage von Aktivitätsaufgaben.

Auf den Abschluss einer Aktivitätsaufgabe warten

Konfigurieren Sie, wie lange ein Zustand wartet, indem Sie `TimeoutSeconds` in der Aufgabendefinition festlegen. Wenn die Aufgabe aktiv bleiben und warten soll, senden Sie mit `SendTaskHeartbeat` regelmäßig einen Heartbeat von Ihrem Aktivitäts-Worker innerhalb der in `TimeoutSeconds` konfigurierten Zeit. Durch die Konfiguration einer langen Timeout-Dauer und das aktive Senden eines Herzschlags kann eine Aktivität in Step Functions bis zu einem Jahr warten, bis eine Ausführung abgeschlossen ist.

Wenn Sie beispielsweise einen Workflow benötigen, der auf das Ergebnis eines langen Prozesses wartet, gehen Sie wie folgt vor:

1. Erstellen Sie eine Aktivität mithilfe der Konsole oder unter Verwendung von `CreateActivity`. Notieren Sie sich den ARN der Aktivität.
2. Verweisen Sie auf diesen ARN in einem Aktivitätsaufgabenstatus in der Definition Ihres Zustandsautomaten und setzen Sie `TimeoutSeconds`.
3. Implementieren Sie einen Aktivitäts-Worker, der Arbeit mit `GetActivityTask` abruft und dabei auf diesen Aktivitäts-ARN verweist.
4. Verwenden Sie regelmäßig `SendTaskHeartbeat` innerhalb der in `HeartbeatSeconds` in Ihrer Aufgabendefinition des Zustandsautomaten festgelegten Zeit, um zu verhindern, dass ein Timeout für die Aufgabe entsteht.
5. Starten Sie die Ausführung Ihres Zustandsautomaten.
6. Starten Sie Ihren Aktivitäts-Worker-Prozess.

Die Ausführung wird an dem Aktivitätsaufgabenstatus unterbrochen und wartet darauf, dass Ihr Aktivitäts-Worker eine Aufgabe abruft. Nachdem Ihrem Aktivitäts-Worker ein `taskToken` übergeben wurde, wartet Ihr Workflow, bis `SendTaskSuccess` oder `SendTaskFailure`einen Status bereitstellt. Erhält die Ausführung keinen dieser Status oder einen `SendTaskHeartbeat`-

Aufruf, bevor die in `TimeoutSeconds` konfigurierte Zeit abgelaufen ist, schlägt die Ausführung fehl und der Ausführungsverlauf enthält ein `ExecutionTimedOut`-Ereignis.

Nächste Schritte

Ausführlichere Informationen über das Erstellen eines Zustandsautomaten, der Aktivitäts-Arbeitskräfte verwendet, finden Sie unter:

- Einen Activity State Machine mithilfe von Step Functions erstellen
- Beispiel für einen Aktivitäts-Worker in Ruby

Beispiel für einen Aktivitäts-Worker in Ruby

Im Folgenden finden Sie ein Beispiel für einen Aktivitäts-Worker, der AWS SDK for Ruby verwendet. Im Beispiel werden bewährte Methoden für die Implementierung eigener Aktivitäts-Worker gezeigt.

Der Code implementiert ein Verbraucher/Erzeuger-Muster mit einer konfigurierbaren Anzahl an Threads für Poller und Aktivitäts-Worker. Die Poller-Threads führen durchgängig Langabfragen für die Aktivitätsaufgabe aus. Wurde die Aktivitätsabfrage abgerufen, wird sie über eine gebundene Sperrwarteschlange weitergegeben, wo der Aktivitäts-Thread sie abgreifen kann.

- Weitere Informationen über die AWS SDK for Ruby finden Sie in der AWS SDK for RubyAPI-Referenz.
- Um diesen Code und die zugehörigen Ressourcen herunterzuladen, besuchen Sie das step-functions-ruby-activity-worker-Repository unterGitHub.

Der folgende Ruby-Code ist der Haupteintrittspunkt für diesen beispielhaften Aktivitäts-Worker in Ruby.

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

activity = StepFunctions::Activity.new(
  credentials: credentials,
  region: region,
  activity_arn: activity_arn,
```

```
  workers_count: 1,
  pollers_count: 1,
  heartbeat_delay: 30
)

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

Der Code enthält Vorgabewerte, die Sie ändern können, um auf Ihre Aktivität zu verweisen und sie Ihrer spezifischen Implementierung anzupassen. Dieser Code nimmt als Eingabe die eigentliche Implementierungslogik entgegen. Er gestattet Ihnen dabei, auf Ihre spezifische Aktivität zu verweisen und Ihre Anmeldeinformationen zu verwenden, und Sie können die Anzahl der Threads sowie die Heartbeat-Verzögerung konfigurieren. Weitere Informationen und zum Herunterladen des Codes finden Sie unter Step Functions Ruby Activity Worker.

Item	Beschreibung
`require_relative`	Relativer Pfad zum folgenden Beispielcode für einen Aktivitäts-Worker.
`region`	AWS-Region Ihrer Aktivität.
`workers_count`	Die Anzahl der Threads für Ihren Aktivitäts-Worker. Für die meisten Implementierungen sollten 10 bis 20 Threads ausreichend sein. Je länger die Verarbeitung der Aktivität dauert, desto mehr Threads benötigt sie gegebenen falls. Als Schätzwert in Sekunden multipliz ieren Sie die Anzahl der Prozessaktivitäten pro Sekunde mit der Aktivitätsverarbeitungslatenz des 99. Perzentils.
`pollers_count`	Die Anzahl der Threads für Ihre Poller. Für die meisten Implementierungen sollten 10 bis 20 Threads ausreichend sein.

Item	Beschreibung
`heartbeat_delay`	Die Verzögerung zwischen den Heartbeats in Sekunden.
`input`	Implementierungslogik Ihrer Aktivität

Nachfolgend finden Sie den in Ruby geschriebenen Aktivitäts-Worker, auf den in Ihrem Code mit `../lib/step_functions/activity` verwiesen wird.

```
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

Ein `Choice` state (`"Type": "Choice"`) fügt einer Zustandsmaschine Bedingungslogik hinzu.

Zusätzlich zu den meisten allgemeinen Statusfeldern enthält `Choice` States die folgenden zusätzlichen Felder.

`Choices` (Erforderlich)

Ein Array von Auswahlregeln, die bestimmen, in welchen Zustand der Zustandsautomat als nächstes übergeht. Sie verwenden einen Vergleichsoperator in einer Auswahlregel, um eine Eingabevariable mit einem bestimmten Wert zu vergleichen. Mithilfe von Auswahlregeln können Sie beispielsweise vergleichen, ob eine Eingabevariable größer oder kleiner als 100 ist.

Wenn ein `Choice` Status ausgeführt wird, bewertet er jede Auswahlregel als wahr oder falsch. Basierend auf dem Ergebnis dieser Bewertung geht Step Functions zum nächsten Status im Workflow über.

Sie müssen mindestens eine Regel im `Choice` Bundesstaat definieren.

`Default` (optional, empfohlen)

Der Name des nächsten Zustands, wenn keiner der Übergänge in `Choices` genommen wird.

> **Important**
>
> `Choice`-Zustände unterstützen nicht das Feld `End`. Außerdem verwenden sie `Next` nur innerhalb ihres `Choices`-Felds.

> **Tip**
>
> Ein Beispiel für einen Workflow, der einen `Choice` Status für Ihren verwendetAWS-Konto, finden Sie in Modul 5 — Status auswählen und Status des AWS Step Functions Workshops zuordnen.

Auswahlregeln

Ein `Choice` Staat muss ein `Choices` Feld haben, dessen Wert ein nicht leeres Array ist. Jedes Element in diesem Array ist ein Objekt namens Choice Rule, das Folgendes enthält:

- Ein Vergleich — Zwei Felder, die eine zu vergleichende Eingabevariable, die Art des Vergleichs und den Wert angeben, mit dem die Variable verglichen werden soll. Auswahlregeln unterstützen den Vergleich zwischen zwei Variablen. Innerhalb einer Auswahlregel kann der Wert einer Variablen mit einem anderen Wert aus der Statuseingabe verglichen werden, indem er `Path` an die Namen der unterstützten Vergleichsoperatoren angehängt wird. Die Werte der Felder `Variable` und Pfad in einem Vergleich müssen gültige Referenzpfade sein.
- Ein **`Next`**Feld — Der Wert dieses Felds muss mit einem Statusnamen in der State-Maschine übereinstimmen.

Im folgenden Beispiel wird geprüft, ob der numerische Wert gleich 1 ist.

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

Im folgenden Beispiel wird geprüft, ob die Zeichenfolge gleich `MyString` ist.

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

Im folgenden Beispiel wird geprüft, ob die Zeichenfolge größer als `MyStringABC` ist.

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

Das folgende Beispiel überprüft, ob die Zeichenfolge Null ist.

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

Das folgende Beispiel zeigt, wie die StringEquals Regel aufgrund der vorherigen `IsPresent` Auswahlregel nur ausgewertet wird, wenn sie `$.keyThatMightNotExist` existiert.

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

```
]
```

Im folgenden Beispiel wird geprüft, ob ein Muster mit einem Platzhalter übereinstimmt.

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

Im folgenden Beispiel wird geprüft, ob der Zeitstempel gleich `2001-01-01T12:00:00Z` ist.

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

Im folgenden Beispiel wird eine Variable mit einem anderen Wert aus der Statuseingabe verglichen.

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

Step Functions untersucht jede der Auswahlregeln in der im `Choices` Feld angegebenen Reihenfolge. Anschließend wechselt es in den Zustand, der im Feld `Next` der ersten Auswahlregel angegeben ist, in der die Variable mit dem Wert entsprechend dem Vergleichsoperator übereinstimmt.

Die folgenden Vergleichsoperatoren werden unterstützt:

- `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`

Für jeden dieser Operatoren muss der entsprechende Wert vom entsprechenden Typ sein: Zeichenfolge, Zahl, Boolean oder Zeitstempel. Step Functions versucht nicht, ein numerisches Feld einem Zeichenfolgenwert zuzuordnen. Da Zeitstempelfelder jedoch logisch gesehen Zeichenfolgen sind, ist es möglich, dass ein Feld, das als Zeitstempel gilt, mit einem Vergleichsoperator `StringEquals` übereinstimmt.

> **Note**
>
> Gehen Sie aus Gründen der Interoperabilität nicht davon aus, dass numerische Vergleiche mit Werten außerhalb der Größenordnung oder Präzision funktionieren, die der IEEE 754-2008 `binary64`-Datentyp repräsentiert. Insbesondere Ganzzahlen außerhalb des Bereichs $[-2^{53}+1, 2^{53}-1]$ können möglicherweise nicht in der erwarteten Weise verglichen werden.

Zeitstempel (z. B. `2016-08-18T17:33:00Z`) müssen RFC3339-Profil ISO 8601 entsprechen, mit weiteren Beschränkungen:

- Ein großes `T` muss die Teile Datum und Uhrzeit trennen.
- Ein großes `Z` muss kennzeichnen, dass keine numerische Zeitzonenabweichung besteht.

Informationen zum Verständnis des Verhaltens bei Vergleichen von Zeichenfolgen finden Sie in der Java `compareTo`-Dokumentation.
Die Werte der Operatoren `And` und `Or` müssen nicht leere Arrays von Auswahlregeln sein, die selbst keine `Next`-Felder enthalten dürfen. Ebenso muss der Wert eines `Not`-Operators eine einzelne Auswahlregel sein, die kein `Next`-Feld enthalten darf.
Sie können komplexe, verschachtelte Auswahlregeln mithilfe von `And`, `Not` und `Or` erstellen. Das Feld `Next` darf jedoch nur in einer Auswahlregel des obersten Levels erscheinen.
Ein Zeichenfolgenvergleich mit Mustern mit einem oder mehreren Platzhaltern („*“) kann mit dem StringMatches Vergleichsoperator durchgeführt werden. Das Platzhalterzeichen wird maskiert, indem der Standard `\\ (Ex: “\\*”)` verwendet wird. Keine anderen Zeichen als „*“ haben beim Abgleich eine besondere Bedeutung.

Beispiel für den Auswahlzustand

Es folgt ein Beispiel für einen `Choice`-Zustand und andere Zustände, in die er übergeht.

> **Note**
>
> Sie müssen das Feld `$.type` angeben. Wenn die Eingabe des Zustands das Feld `$.type` nicht enthält, schlägt die Ausführung fehl und es wird ein Fehler im Ausführungsverlauf angezeigt. Sie können in dem `StringEquals` Feld nur eine Zeichenfolge angeben, die einem Literalwert entspricht. Zum Beispiel `"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!"
}
```

In diesem Beispiel beginnt der Zustandsautomat mit folgendem Eingabewert.

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

Step Functions wechselt basierend auf dem `value` Feld in den `ValueInTwenties` Status.

Wenn der Zustand `Choice` keine Übereinstimmungen für seine `Choices` hat, wird stattdessen der im Feld `Default` bereitgestellte Zustand ausgeführt. Wenn der Zustand `Default` nicht angegeben ist, schlägt die Ausführung fehl.

Wait

Ein `Wait`-Zustand (`"Type": "Wait"`) verzögert die Fortsetzung des Zustandsautomaten für eine bestimmte Zeit. Sie können entweder eine relative Zeit auswählen, angegeben in Sekunden ab Beginn des Zustands, oder eine absolute Endzeit, angegeben als Zeitstempel.

Zusätzlich zu den allgemeinen Zustandsfeldern haben `Wait`-Zustände eines der folgenden Felder.

`Seconds`

Eine Wartezeit, in Sekunden, bevor der im Feld `Next` angegebene Zustand beginnt. Sie müssen die Zeit als positiven, ganzzahligen Wert zwischen 0 und 99999999 angeben.

`Timestamp`

Eine absolute Wartezeit, in Sekunden, bis der im Feld `Next` angegebene Zustand beginnt.

Zeitstempel müssen dem RFC3339-Profil von ISO 8601 entsprechen, mit den weiteren Einschränkungen, dass ein `T` (Großbuchstabe) die Bestandteile Datum und Uhrzeit trennen muss und ein `Z` (Großbuchstabe) anzeigen muss, dass keine numerische Zeitzonenabweichung besteht, z. B. `2024-08-18T17:33:00Z`.

> **Note**
>
> Wenn Sie derzeit die Wartezeit als Zeitstempel angeben, berücksichtigt Step Functions den Zeitwert bis zu Sekunden und kürzt Millisekunden.

`SecondsPath`

Eine Wartezeit, in Sekunden, bevor der im Feld `Next` angegebene Zustand beginnt, angegeben mittels eines Pfads aus den Eingabedaten des Zustands.

Sie müssen einen Ganzzahlwert für dieses Feld angeben.

`TimestampPath`

Eine absolute Wartezeit, in Sekunden, bis der im Feld `Next` angegebene Zustand beginnt, angegeben mittels eines Pfads aus den Eingabedaten des Zustands.

> **Note**
>
> Sie müssen entweder `Seconds`, `Timestamp`, `SecondsPath` oder `TimestampPath` genau angeben. Darüber hinaus beträgt die maximale Wartezeit, die Sie für Standard-Workflows und Express-Workflows angeben können, ein Jahr bzw. fünf Minuten.

Beispiele für den Wartezustand

Der folgende `Wait`-Zustand führt eine Verzögerung von 10 Sekunden in einen Zustandsautomaten ein.

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

Im nächsten Beispiel wartet der `Wait` Bundesstaat bis zu einer absoluten Uhrzeit, nämlich dem 14. März 2024, um 1:59 Uhr UTC.

```
"wait_until" : {
```

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

Die Wartezeit muss nicht hartcodiert sein. Beispielsweise angesichts der folgenden Eingabedaten:

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

Sie können den Wert "expirydate" aus der Eingabe mithilfe eines Referenzpfads auswählen, um ihn aus den Eingabedaten auszuwählen.

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

Succeed

Ein `Succeed`-Zustand (`"Type": "Succeed"`) stoppt eine Ausführung erfolgreich. Der `Succeed`-Zustand ist ein nützliches Ziel für `Choice`-Zustandsverzweigungen, die nichts anderes tun als die Ausführung zu stoppen.

Da `Succeed`-Zustände Terminalzustände sind, haben sie kein `Next`-Feld und benötigen kein `End`-Feld, wie im folgenden Beispiel gezeigt.

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

Fehler

Ein`Fail`Bundesstaat (`"Type": "Fail"`) stoppt die Ausführung der Zustandsmaschine und markiert sie als Fehler, es sei denn, sie wird von einem`Catch`blockieren.

Der `Fail`-Zustand erlaubt nur die Verwendung der Felder `Type` und `Comment` aus der Menge der allgemeinen Zustandsfelder. Darüber hinaus erlaubt der `Fail`-Zustand die folgenden Felder.

`Cause` (Optional)

Eine benutzerdefinierte Zeichenfolge, die die Ursache des Fehlers beschreibt. Sie können dieses Feld für Betriebs- oder Diagnosezwecke angeben.

`CausePath` (Optional)

Wenn Sie eine detaillierte Beschreibung der Fehlerursache dynamisch anhand der Statuseingabe bereitstellen möchten, verwenden Sie einenReferenzpfad, verwenden`CausePath`. Wenn das Problem gelöst ist, muss der Referenzpfad ein Feld auswählen, das einen Zeichenkettenwert enthält.

Sie können auch angeben`CausePath`hunter Verwendung einesintrinsische Funktiondas gibt eine Zeichenfolge zurück. Diese intrinsischen Eigenschaften sind:Bundesstaaten.Format,States.JsonToString,States.ArrayGetItem,States.Base64Encode,States.BaseHash, undStates.UUID.

> **Important**
>
> - Sie können entweder`Cause`oder`CausePath`, aber nicht beide in Ihrer Fail-State-Definition.
> - Im Sinne bewährter Sicherheitsmethoden wird empfohlen, dass Sie sensible Daten oder interne Systemdaten aus Sicherheitsgründen empfehlen wir, dass Sie vertrauliche Daten oder interne Systemdaten nicht mehr nutzen.

`Error` (Optional)

Ein Fehlername, den Sie für die Fehlerbehandlung angeben können, indem SieVersuchen Sie es erneutoderFangen-Felder. Sie können auch einen Fehlernamen für Betriebs- oder Diagnosezwecke angeben.

`ErrorPath` (Optional)

Wenn Sie einen Namen für den Fehler dynamisch aus der Statuseingabe angeben möchten, verwenden Sie einenReferenzpfad, verwenden`ErrorPath`. Wenn das Problem gelöst ist, muss der Referenzpfad ein Feld auswählen, das einen Zeichenkettenwert enthält.

Sie können auch angeben`ErrorPath`hunter Verwendung einesintrinsische Funktiondas gibt eine Zeichenfolge zurück. Diese intrinsischen Eigenschaften

sind:Bundesstaaten.Format,States.JsonToString,States.ArrayGetItem,States.Base64Encode,States.Base
Hash, undStates.UUID.

> **Important**
>
> - Sie können entwederErroroderErrorPath, aber nicht beide in Ihrer Fail-State-Definition.
> - Im Sinne bewährter Sicherheitsmethoden wird empfohlen, dass Sie vertrauliche Daten oder interne Systemdaten aus Sicherheitsgründen empfehlen wir, dass Sie vertrauliche Daten oder interne Systemdaten nicht mehr nutzen.

Da die Fail-Zustände den Zustandsautomaten immer beenden, haben sie kein Next-Feld und erfordern kein End-Feld.

Beispiele für Fail-State-Definitionen

Das folgende Beispiel für eine Definition von Fail State gibt Static anErrorundCauseFeldwerte.

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

Im folgenden Beispiel für die Definition eines Fehlerzustands werden Referenzpfade dynamisch verwendet, um das Problem zu lösenErrorundCauseFeldwerte.

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

Das folgende Beispiel für die Definition des Status Fail verwendetStates.Formatintrinsische Funktion zur Spezifizierung derErrorundCauseFeldwerte dynamisch.

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

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

Parallel

Der `Parallel` Status (`"Type": "Parallel"`) kann verwendet werden, um Ihrem Zustandsautomaten separate Ausführungszweige hinzuzufügen.

Zusätzlich zu den allgemeinen Zustandsfeldern führen `Parallel`-Zustände diese zusätzlichen Felder ein.

`Branches` (Erforderlich)

Ein Array von Objekten, das Zustandsautomaten bestimmt, die parallel ausgeführt werden. Jedes dieser Zustandsautomatenobjekte muss über Felder namens `States` und `StartAt` verfügen, die dasselbe bedeuten wie diejenigen im oberen Level eines Zustandsautomaten.

`ResultPath` (Optional)

Gibt an, wo (in der Eingabe) die Ausgabe der Verzweigungen platziert werden soll. Die Eingabe wird dann wie vom Feld `OutputPath` festgelegt gefiltert (falls vorhanden), bevor sie als Ausgabe des Zustands verwendet wird. Weitere Informationen finden Sie unter Verarbeitung von Eingabe und Ausgabe.

`ResultSelector` (Optional)

Übergeben Sie eine Sammlung von Schlüssel-Wert-Paaren, wobei die Werte statisch sind oder aus dem Ergebnis ausgewählt werden. Weitere Informationen finden Sie unter ResultSelector.

`Retry` (Optional)

Ein Array von Objekten namens Retrier, die eine Wiederholungsrichtlinie für den Fall definieren, dass der Zustand auf Laufzeitfehler trifft. Weitere Informationen finden Sie unter Beispiele für Zustandsmaschinen mit Retry und Catch.

`Catch` (Optional)

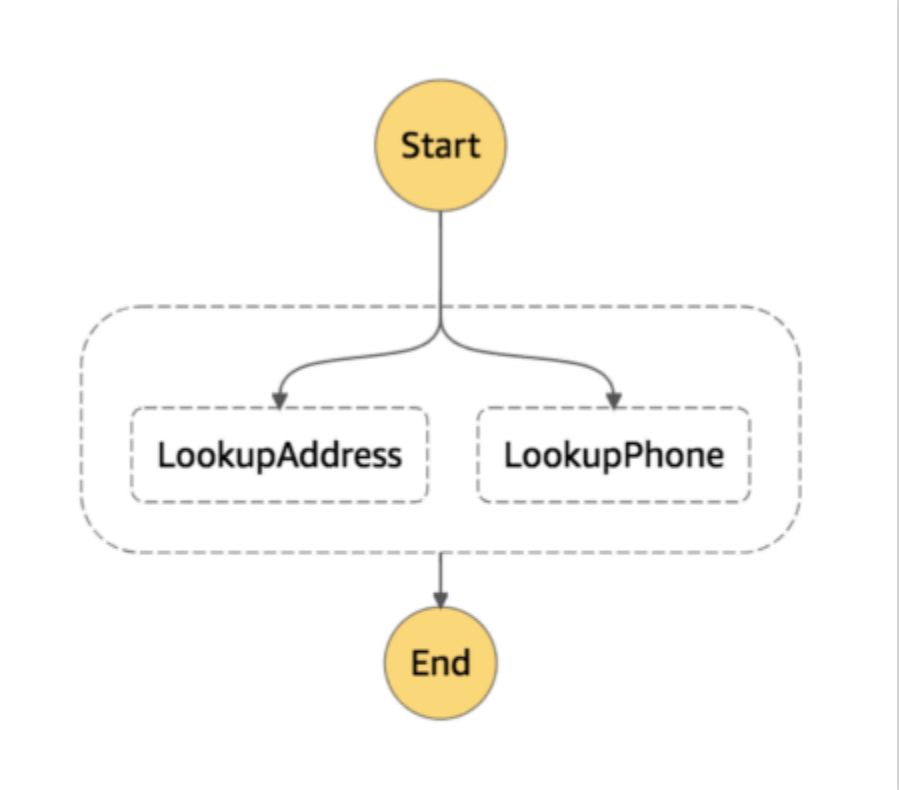
Ein Array von Objekten namens Catcher, die einen Fallback-Zustand definieren, der ausgeführt wird, wenn der Zustand auf Laufzeitfehler trifft und seine Wiederholungsrichtlinie ausgeschöpft wurde oder nicht definiert ist. Weitere Informationen finden Sie unter Fallback-Zustände.

Ein -`Parallel`Zustand bewirkt, dass jede Verzweigung AWS Step Functions so gleichzeitig wie möglich ausführt, beginnend mit dem Status im `StartAt` Feld dieser Verzweigung, und wartet, bis alle Verzweigungen beendet werden (einen Beendigungsstatus erreichen), bevor das `Next` Feld der `Parallel` Verzweigung verarbeitet wird.

Beispiel für den Parallelzustand

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

In diesem Beispiel werden die Verzweigungen `LookupAddress` und `LookupPhone` parallel ausgeführt. So sieht der visuelle Workflow in der Step-Functions-Konsole aus.



Jede Verzweigung muss unabhängig sein. Ein Zustand in einer Verzweigung eines `Parallel`-Zustands darf kein `Next`-Feld haben, das auf ein Feld außerhalb dieser Verzweigung abzielt, noch darf ein anderer Zustand außerhalb der Verzweigung in diese Verzweigung übergehen.

Ein- und Ausgabeverarbeitung des Parallelzustands

Ein `Parallel`-Zustand stellt jeder Verzweigung eine Kopie seiner eigenen Eingabedaten bereit (Änderungen durch das Feld `InputPath` unterliegend). Er generiert eine Ausgabe, die ein Array mit einem Element für jede Verzweigung ist, das die Ausgabe von dieser Verzweigung enthält. Es ist nicht erforderlich, dass alle Elemente vom selben Typ sind. Das Ausgabe-Array kann in die Eingabedaten eingefügt werden (und das Ganze als Ausgabe des `Parallel`-Zustands gesendet

werden), indem ein `ResultPath`-Feld auf die übliche Weise verwendet wird (siehe Verarbeitung von Eingabe und Ausgabe).

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

Wenn dem Zustand `FunWithMath` das Array `[3, 2]` als Eingabe übergeben wird, dann erhalten die Zustände `Add` und `Subtract` dieses Array als Eingabe. Die Ausgabe der `Subtract` Aufgaben `Add` und wäre die Summe der Array-Elemente 3 und 2, d. h. 5 und 1, während die Ausgabe des `Parallel` Zustands ein Array wäre.

```
[ 5, 1 ]
```

> **Tip**
>
> Wenn der Parallel- oder Map-Status, den Sie in Ihren Zustandsautomaten verwenden, ein Array von Arrays zurückgibt, können Sie sie mit dem ResultSelector Feld in ein flaches Array umwandeln. Weitere Informationen finden Sie unter Reduzieren eines Arrays von Arrays.

Fehlerbehandlung

Wenn eine der Verzweigungen aufgrund eines unbehandelten Fehlers oder durch den Übergang zu einem `Fail`-Zustand fehlschlägt, wird der gesamte `Parallel`-Zustand als fehlgeschlagen betrachtet und alle seine Verzweigungen werden gestoppt. Wenn der Fehler nicht vom `Parallel` Status selbst behandelt wird, stoppt Step Functions die Ausführung mit einem Fehler.

> **Note**
>
> Wenn ein paralleler Status fehlschlägt, werden aufgerufene Lambda-Funktionen weiterhin ausgeführt und Aktivitäts-Worker, die ein Aufgaben-Token verarbeiten, werden nicht gestoppt.
>
> - Um Aktivitäten mit langer Laufzeit zu beenden, verwenden Sie Heartbeats, um zu erkennen, ob sein Branch von Step Functions gestoppt wurde, und halten Sie Auftragnehmer an, die Aufgaben verarbeiten. Durch das Aufrufen von `SendTaskHeartbeat`, `SendTaskSuccess` oder `SendTaskFailure` wird ein Fehler gemeldet, wenn der Status einen Fehler aufweist. Weitere Informationen finden Sie unter Heartbeat-Fehler.
> - Das Ausführen von Lambda-Funktionen kann nicht gestoppt werden. Wenn Sie einen Fallback implementiert haben, verwenden Sie einen -`Wait`Zustand, damit die Bereinigungsarbeit stattfindet, nachdem die Lambda-Funktion abgeschlossen ist.

Zuordnung

Verwenden Sie den `Map` Status, um eine Reihe von Workflow-Schritten für jedes Element in einem Datensatz auszuführen. Die Iterationen des `Map` Staates laufen parallel, was eine schnelle Verarbeitung eines Datensatzes ermöglicht. `Map`Staaten können eine Vielzahl von Eingabetypen verwenden, darunter ein JSON-Array, eine Liste von Amazon S3 S3-Objekten oder eine CSV-Datei.

Step Functions bietet zwei Arten von Verarbeitungsmodi, um den `Map` Status in Ihren Workflows zu verwenden: den Inline-Modus und den verteilten Modus.

Informationen zu diesen Modi und zur Verwendung des `Map` Status in beiden Modi finden Sie in den folgenden Themen:

- Zustandsverarbeitungsmodi zuordnen
  - Verwenden des Kartenstatus im Inline-Modus
  - Verwenden von Map State im verteilten Modus

> **Tip**
>
> Ein Beispiel für einen Workflow, der einen `Map` Status für Ihren verwendetAWS-Konto, finden Sie in Modul 5 — Choice State and Map State of The AWS Step Functions Workshop.

Zustandsverarbeitungsmodi zuordnen

Step Functions bietet die folgenden Verarbeitungsmodi für den `Map` Status, je nachdem, wie Sie die Elemente in einem Datensatz verarbeiten möchten.

- Inline — Modus mit begrenzter Parallelität. In diesem Modus wird jede Iteration des `Map` Status im Kontext des Workflows ausgeführt, der den Status enthält. `Map` Step Functions fügt die Ausführungshistorie dieser Iterationen zur Ausführungshistorie des übergeordneten Workflows hinzu. Standardmäßig werden `Map` Staaten im Inline-Modus ausgeführt.

  In diesem Modus akzeptiert der `Map` Status nur ein JSON-Array als Eingabe. Außerdem unterstützt dieser Modus bis zu 40 gleichzeitige Iterationen.

  Weitere Informationen finden Sie unter Verwenden des Kartenstatus im Inline-Modus.
- Verteilt — Modus mit hoher Parallelität. In diesem Modus führt der `Map` Status jede Iteration als untergeordnete Workflow-Ausführung aus, wodurch eine hohe Parallelität von bis zu 10.000 parallel untergeordneten Workflow-Ausführungen ermöglicht wird. Jede untergeordnete Workflow-Ausführung hat ihren eigenen Ausführungsverlauf, der von dem des übergeordneten Workflows getrennt ist.

  In diesem Modus kann der `Map` Staat entweder ein JSON-Array oder eine Amazon S3 S3-Datenquelle, z. B. eine CSV-Datei, als Eingabe akzeptieren.

Weitere Informationen finden Sie unter Verwenden von Map State im verteilten Modus.

Welchen Modus Sie verwenden sollten, hängt davon ab, wie Sie die Elemente in einem Datensatz verarbeiten möchten. Verwenden Sie den `Map` Status im Inline-Modus, wenn der Ausführungsverlauf Ihres Workflows 25.000 Einträge nicht überschreitet oder wenn Sie nicht mehr als 40 gleichzeitige Iterationen benötigen.

Verwenden Sie den `Map` Status im Modus Verteilt, wenn Sie umfangreiche parallel Workloads orchestrieren müssen, die eine beliebige Kombination der folgenden Bedingungen erfüllen:

- Die Größe Ihres Datensatzes übersteigt 256 KB.
- Der Verlauf der Ausführungsereignisse des Workflows umfasst mehr als 25.000 Einträge.
- Sie benötigen eine Parallelität von mehr als 40 parallel Iterationen.

Themen

Unterschiede im Inline-Modus und im verteilten Modus

In der folgenden Tabelle werden die Unterschiede zwischen den Modi Inline und Distributed hervorgehoben.

Inline-Modus	Verteilter Modus
Supported data sources	
Akzeptiert ein JSON-Array, das aus einem vorherigen Schritt im Workflow als Eingabe übergeben wurde.	Akzeptiert die folgenden Datenquellen als Eingabe: • JSON-Array, das aus einem vorherigen Schritt im Workflow übergeben wurde

Inline-Modus	Verteilter Modus
	• JSON-Datei in einem Amazon S3 S3-Bucket, der ein Array enthält • CSV-Datei in einem Amazon S3 S3-Bucket • Amazon S3 S3-Objektliste • Amazon S3 Inventory
Map iterations	
In diesem Modus wird jede Iteration des `Map` Status im Kontext des Workflows ausgeführt, der den `Map` Status enthält. Step Functions fügt die Ausführungshistorie dieser Iterationen zur Ausführungshistorie des übergeordneten Workflows hinzu.	In diesem Modus führt der `Map` Status jede Iteration als untergeordnete Workflow-Ausführung aus, wodurch eine hohe Parallelität von bis zu 10.000 parallel untergeordneten Workflow-Ausführungen ermöglicht wird. Jede untergeordnete Workflow-Ausführung hat ihren eigenen Ausführungsverlauf, der von dem des übergeordneten Workflows getrennt ist.
Maximum concurrency for parallel iterations	
Ermöglicht es Ihnen, bis zu 40 Iterationen so gleichzeitig wie möglich auszuführen.	Ermöglicht die Ausführung von bis zu 10.000 parallel untergeordneten Workflow-Ausführungen, um Millionen von Datenelementen gleichzeitig zu verarbeiten.
Input payload and event history sizes	

Inline-Modus	Verteilter Modus
Erzwingt ein Limit von 256 KB für die Größe der Eingabe-Payload und 25.000 Einträge im Verlauf der Ausführungsereignisse.	Ermöglicht es Ihnen, die Beschränkung der Nutzlastgröße zu umgehen, da der `Map` Status Eingaben direkt aus Amazon S3 S3-Datenq uellen lesen kann. In diesem Modus können Sie auch Einschrän kungen im Ausführungsverlauf umgehen, da die vom `Map` Status gestarteten untergeor dneten Workflow-Ausführungen ihre eigenen, vom Ausführungsverlauf des übergeordneten Workflows getrennten Ausführungshistorien beibehalten.
Monitoring and observability	
Sie können den Ausführungsverlauf des Workflows von der Konsole aus oder durch Aufrufen der `GetExecutionHistory` API-Aktion überprüfen. Sie können den Ausführungsverlauf auch über CloudWatch X-Ray einsehen.	Wenn Sie einen `Map` Status im verteilten Modus ausführen, erstellt Step Functions eine Map Run-Ressource. Ein Map Run bezieht sich auf eine Reihe von untergeordneten Workflow-Ausführungen, die im Status „Distributed Map“ gestartet werden. Sie können einen Map Run in der Step Functions-Konsole anzeigen. Sie können die `DescribeMapRun` API-Aktio n auch aufrufen. Ein Map Run sendet auch Metriken an. CloudWatch Weitere Informationen finden Sie unter Untersuchen der Map-Ausführung einer Ausführung des verteilten Map-Zustands.

Verwenden des Kartenstatus im Inline-Modus

Standardmäßig wird states `Map` im Inline-Modus ausgeführt. Im Inline-Modus akzeptiert der Map-Status nur ein JSON-Array als Eingabe. Es erhält dieses Array aus einem früheren Schritt im Workflow. In diesem Modus wird jede Iteration des `Map` Status im Kontext des Workflows ausgeführt,

der den Map Status enthält. Step Functions fügt die Ausführungshistorie dieser Iterationen zur Ausführungshistorie des übergeordneten Workflows hinzu.

In diesem Modus unterstützt der Map Status bis zu 40 gleichzeitige Iterationen.

Ein Map Status, der auf Inline gesetzt ist, wird als Inline Map-Status bezeichnet. Verwenden Sie den Map Status im Inline-Modus, wenn der Ausführungsverlauf Ihres Workflows 25.000 Einträge nicht überschreitet oder wenn Sie nicht mehr als 40 gleichzeitige Iterationen benötigen.

Eine Einführung in die Verwendung des Inline-Map-Status finden Sie im Tutorial. Wiederholen Sie eine Aktion mit dem Status „Inline Map“

Inhalt

Die wichtigsten Konzepte in diesem Thema

Inline-Modus

Ein Modus des Bundesstaates mit begrenzter Parallelität. Map In diesem Modus wird jede Iteration des Map Status im Kontext des Workflows ausgeführt, der den Status enthält. Map Step Functions fügt die Ausführungshistorie dieser Iterationen zur Ausführungshistorie des übergeordneten Workflows hinzu. MapStatus werden standardmäßig im Inline-Modus ausgeführt.

Dieser Modus akzeptiert nur ein JSON-Array als Eingabe und unterstützt bis zu 40 gleichzeitige Iterationen.

Status der Inline-Karte

Ein Map Status, der auf den Inline-Modus eingestellt ist.

Arbeitsablauf zuordnen

Die Gruppe von Schritten, die der Map Staat für jede Iteration ausführt.

Iteration des Zustands zuordnen

Eine Wiederholung des innerhalb des Status definierten Workflows. `Map`

Statusfelder in der Inline-Map

Um den Inline-Map-Status in Ihren Workflows zu verwenden, geben Sie eines oder mehrere dieser Felder an. Sie geben diese Felder zusätzlich zu den allgemeinen Statusfeldern an.

`Type` (Erforderlich)

Legt den Zustandstyp fest, z. `Map` B.

`ItemProcessor` (Erforderlich)

Enthält die folgenden JSON-Objekte, die den Verarbeitungsmodus und die Definition des `Map` Status angeben.

Die Definition enthält die Schritte, die für die Verarbeitung der einzelnen Array-Elemente wiederholt werden müssen.

- `ProcessorConfig`— Ein optionales JSON-Objekt, das den Verarbeitungsmodus für den `Map` Status angibt. Dieses Objekt enthält das `Mode` Unterfeld. Dieses Feld ist standardmäßig auf`INLINE`, was den `Map` Status im Inline-Modus verwendet.

  In diesem Modus führt der Ausfall einer Iteration dazu, dass der `Map` Status fehlschlägt. Alle Iterationen werden beendet, wenn der `Map` Status ausfällt.
- `StartAt`— Gibt eine Zeichenfolge an, die den ersten Status in einem Workflow angibt. Bei dieser Zeichenfolge wird zwischen Groß- und Kleinschreibung unterschieden und sie muss mit dem Namen eines der Statusobjekte übereinstimmen. Dieser Status wird für jedes Element im Datensatz zuerst ausgeführt. Jede Ausführungseingabe, die Sie für den `Map` Status bereitstellen, wird zuerst an den `StartAt` Status übergeben.
- `States`— Ein JSON-Objekt, das eine durch Kommas getrennte Gruppe von Zuständen enthält. In diesem Objekt definieren Sie die. Map workflow

  > **Note**
  >
  > - Staaten innerhalb des `ItemProcessor` Feldes können nur ineinander übergehen. Kein Staat außerhalb des `ItemProcessor` Feldes kann in einen Zustand innerhalb des Feldes übergehen.

- Das `ItemProcessor` Feld ersetzt das jetzt veraltete `Iterator` Feld. Sie können zwar weiterhin `Map` Bundesstaaten einbeziehen, die das `Iterator` Feld verwenden, wir empfehlen jedoch dringend, dieses Feld durch zu ersetzen. `ItemProcessor`

  Step Functions Local unterstützt das `ItemProcessor` Feld derzeit nicht. Wir empfehlen, das `Iterator` Feld mit Step Functions Local zu verwenden.

`ItemsPath` (Optional)

Gibt einen Referenzpfad unter Verwendung der JsonPathSyntax an. Dieser Pfad wählt den JSON-Knoten aus, der das Array von Elementen in der Statuseingabe enthält. Weitere Informationen finden Sie unter ItemsPath.

`ItemSelector` (Optional)

Überschreibt die Werte der Eingabe-Array-Elemente, bevor sie an jede `Map` State-Iteration weitergegeben werden.

In diesem Feld geben Sie eine gültige JSON-Datei an, die eine Sammlung von Schlüssel-Wert-Paaren enthält. Diese Paare können jedes der folgenden Elemente enthalten:

- Statische Werte, die Sie in Ihrer State-Machine-Definition definieren.
- Werte, die mithilfe eines Pfads aus der Statuseingabe ausgewählt wurden.
- Werte, auf die über das Kontextobjekt zugegriffen wird.

Weitere Informationen finden Sie unter ItemSelector.

Das `ItemSelector` Feld ersetzt das jetzt veraltete `Parameters` Feld. Sie können zwar weiterhin `Map` Bundesstaaten einbeziehen, die das `Parameters` Feld verwenden, wir empfehlen jedoch dringend, dieses Feld durch zu ersetzen. `ItemSelector`

`MaxConcurrency` (Optional)

Gibt einen Integer-Wert an, der die Obergrenze für die Anzahl der `Map` State-Iterationen angibt, die parallel ausgeführt werden können. Ein `MaxConcurrency` Wert von 10 begrenzt den `Map` Status beispielsweise auf 10 gleichzeitige Iterationen.

> **Note**
>
> Gleichzeitige Iterationen können begrenzt sein. In diesem Fall beginnen einige Iterationen erst, wenn die vorherigen Iterationen abgeschlossen sind. Die Wahrscheinlichkeit, dass dies passiert, steigt, wenn Ihr Eingabe-Array mehr als 40 Elemente enthält.

Um eine höhere Parallelität zu erreichen, sollten Sie Folgendes in Betracht ziehenVerwenden von Map State im verteilten Modus:

Der Standardwert ist`0`, wodurch die Parallelität nicht begrenzt wird. Step Functions ruft Iterationen so gleichzeitig wie möglich auf.

Ein `MaxConcurrency` Wert von `1` ruft `ItemProcessor` einmal für jedes Array-Element auf. Die Elemente im Array werden in der Reihenfolge verarbeitet, in der sie in der Eingabe erscheinen. Step Functions startet eine neue Iteration erst, wenn die vorherige Iteration abgeschlossen ist.

`MaxConcurrencyPath` (Optional)

Wenn Sie mithilfe eines Referenzpfads dynamisch aus der Statuseingabe einen maximalen Parallelitätswert angeben möchten, verwenden Sie. `MaxConcurrencyPath` Nach der Auflösung muss der Referenzpfad ein Feld auswählen, dessen Wert eine nicht negative Ganzzahl ist.

> **Note**
>
> Ein `Map` Status kann nicht sowohl als auch `MaxConcurrency` enthalten. `MaxConcurrencyPath`

`ResultPath` (Optional)

Gibt an, wo in der Eingabe die Ausgabe der Iterationen des `Map` Status gespeichert werden soll. Der Map-Status filtert dann die Eingabe gemäß den Angaben im `OutputPath`Feld, sofern angegeben. Anschließend wird die gefilterte Eingabe als Ausgabe für den Status verwendet. Weitere Informationen finden Sie unter Verarbeitung von Eingabe und Ausgabe.

`ResultSelector` (Optional)

Übergibt eine Sammlung von Schlüsselwertpaaren, wobei die Werte entweder statisch sind oder aus dem Ergebnis ausgewählt werden. Weitere Informationen finden Sie unter ResultSelector.

> **Tip**
>
> Wenn der Status Parallel oder Map, den Sie in Ihren Zustandsmaschinen verwenden, ein Array von Arrays zurückgibt, können Sie diese mit dem ResultSelector Feld in ein flaches

Array umwandeln. Weitere Informationen finden Sie unter Reduzieren eines Arrays von Arrays.

`Retry` (Optional)

Eine Reihe von Objekten, sogenannte Retriers, die eine Wiederholungsrichtlinie definieren. Staaten verwenden eine Wiederholungsrichtlinie, wenn sie auf Laufzeitfehler stoßen. Weitere Informationen finden Sie unter Beispiele für Zustandsmaschinen mit Retry und Catch.

> **Note**
>
> Wenn Sie Retrier für den Inline-Map-Status definieren, gilt die Wiederholungsrichtlinie für alle `Map` Status-Iterationen und nicht nur für fehlgeschlagene Iterationen. Ihr `Map` Status enthält beispielsweise zwei erfolgreiche Iterationen und eine fehlgeschlagene Iteration. Wenn Sie das `Retry` Feld für den Status definiert haben, gilt die Wiederholungsrichtlinie für alle drei `Map` `Map` Status-Iterationen und nicht nur für die fehlgeschlagene Iteration.

`Catch` (Optional)

Ein Array von Objekten namens Catcher, die einen Fallback-Zustand definieren. Staaten führen einen Catcher aus, wenn sie auf Laufzeitfehler stoßen und entweder keine Wiederholungsrichtlinie haben oder ihre Wiederholungsrichtlinie erschöpft ist. Weitere Informationen finden Sie unter Fallback-Zustände.

Veraltete Felder

> **Note**
>
> Sie können zwar weiterhin `Map` Bundesstaaten einbeziehen, die die folgenden Felder verwenden, wir empfehlen jedoch dringend, sie durch `ItemProcessor` und `Parameters` durch zu `Iterator` ersetzen. `ItemSelector`

`Iterator`

Gibt ein JSON-Objekt an, das eine Reihe von Schritten definiert, mit denen jedes Element des Arrays verarbeitet wird.

`Parameters`

Gibt eine Sammlung von Schlüssel-Wert-Paaren an, wobei die Werte jeden der folgenden Werte enthalten können:

- Statische Werte, die Sie in Ihrer State-Machine-Definition definieren.
- Werte, die mithilfe eines Pfads aus der Eingabe ausgewählt wurden.

Beispiel für einen Inline-Map-Status

Betrachten Sie die folgenden Eingabedaten für einen `Map` Status, der im Inlinemodus ausgeführt wird.

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

Ausgehend von der vorherigen Eingabe ruft der `Map` Status im folgenden Beispiel eine AWS Lambda Funktion auf, die für jedes Element des Arrays im `shipped` Feld `ship-val` einmal benannt wurde.

```
"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"
}
```

Jede Iteration des `Map` Status sendet ein Element im Array, das mit dem `ItemsPath`Feld ausgewählt wurde, als Eingabe für die `ship-val` Lambda-Funktion. Die folgenden Werte sind ein Beispiel für Eingaben, die der `Map` Staat an einen Aufruf der Lambda-Funktion sendet:

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

Wenn Sie fertig sind, ist die Ausgabe des `Map`-Zustands ein JSON-Array, wobei jedes Element die Ausgabe einer Iteration ist. In diesem Fall enthält dieses Array die Ausgabe der `ship-val` Lambda-Funktion.

Beispiel für einen Inline-Map-Status mit **`ItemSelector`**

Angenommen, die `ship-val` Lambda-Funktion im vorherigen Beispiel benötigt auch Informationen über den Kurierdienst der Sendung. Diese Informationen werden zusätzlich zu den Elementen im Array für jede Iteration bereitgestellt. Sie können Informationen aus der Eingabe zusammen mit Informationen einbeziehen, die für die aktuelle Iteration des `Map` Status spezifisch sind. Beachten Sie das `ItemSelector` Feld im folgenden Beispiel:

```
"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
}
```

Der `ItemSelector` Block ersetzt die Eingabe für die Iterationen durch einen JSON-Knoten. Dieser Knoten enthält sowohl die aktuellen Artikeldaten aus dem Kontextobjekt als auch die Kurierinformationen aus dem Feld der `Map` Statuseingabe. `delivery-partner` Das Folgende ist ein Beispiel für die Eingabe einer einzelnen Iteration. Der `Map` Staat übergibt diese Eingabe an einen Aufruf der `ship-val` Lambda-Funktion.

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

Im vorherigen Beispiel für einen Inline-Map-Status erzeugt das `ResultPath` Feld eine Ausgabe im gleichen Format wie die Eingabe. Es überschreibt das `detail.shipped` Feld jedoch mit einem Array, in dem jedes Element die Ausgabe des `ship-val` Lambda-Aufrufs jeder Iteration ist.

Weitere Informationen zur Verwendung des Inline Map-Statusstatus und seiner Felder finden Sie im Folgenden.

- Wiederholen Sie eine Aktion mit dem Status „Inline Map“

- Eingabe- und Ausgabeverarbeitung in Step Functions
- ItemsPath
- Context-Objektdaten für Zuordnungszustände

Eingabe- und Ausgabeverarbeitung im **`Map`** Inline-Status

`InputPath`Wählt für einen bestimmten `Map` Status eine Teilmenge der Eingaben des Status aus.

Die Eingabe eines `Map` Status muss ein JSON-Array enthalten. Der `Map` Status führt den `ItemProcessor` Abschnitt einmal für jedes Element im Array aus. Wenn Sie das `ItemsPath`Feld angeben, wählt der `Map` Status aus, an welcher Stelle in der Eingabe das Array gefunden werden soll, über das iteriert werden soll. Bei fehlender Angabe ist der Wert von `ItemsPath` `$`, und der Abschnitt `ItemProcessor` erwartet, dass das Array die einzige Eingabe ist. Wenn Sie das `ItemsPath` Feld angeben, muss es sich bei seinem Wert um einen Referenzpfad handeln. Der `Map` Staat wendet diesen Pfad auf die effektive Eingabe an, nachdem er den angewendet hat`InputPath`. Der `ItemsPath` muss ein Feld identifizieren, dessen Wert ein JSON-Array ist.

Die Eingabe für jede Iteration ist standardmäßig ein einzelnes Element des Array-Feldes, das durch den `ItemsPath` Wert identifiziert wird. Sie können diesen Wert mit dem `ItemSelector` Feld überschreiben.

Wenn Sie fertig sind, ist die Ausgabe des `Map`-Zustands ein JSON-Array, wobei jedes Element die Ausgabe einer Iteration ist.

Weitere Informationen zu Inline-Map-Statuseingaben und -ausgaben finden Sie im Folgenden:

- Wiederholen Sie eine Aktion mit dem Status „Inline Map“
- Beispiel für einen Inline-Map-Status mit `ItemSelector`
- Eingabe- und Ausgabeverarbeitung in Step Functions
- Context-Objektdaten für Zuordnungszustände
- Dynamisches Verarbeiten von Daten mit einem Map-Status

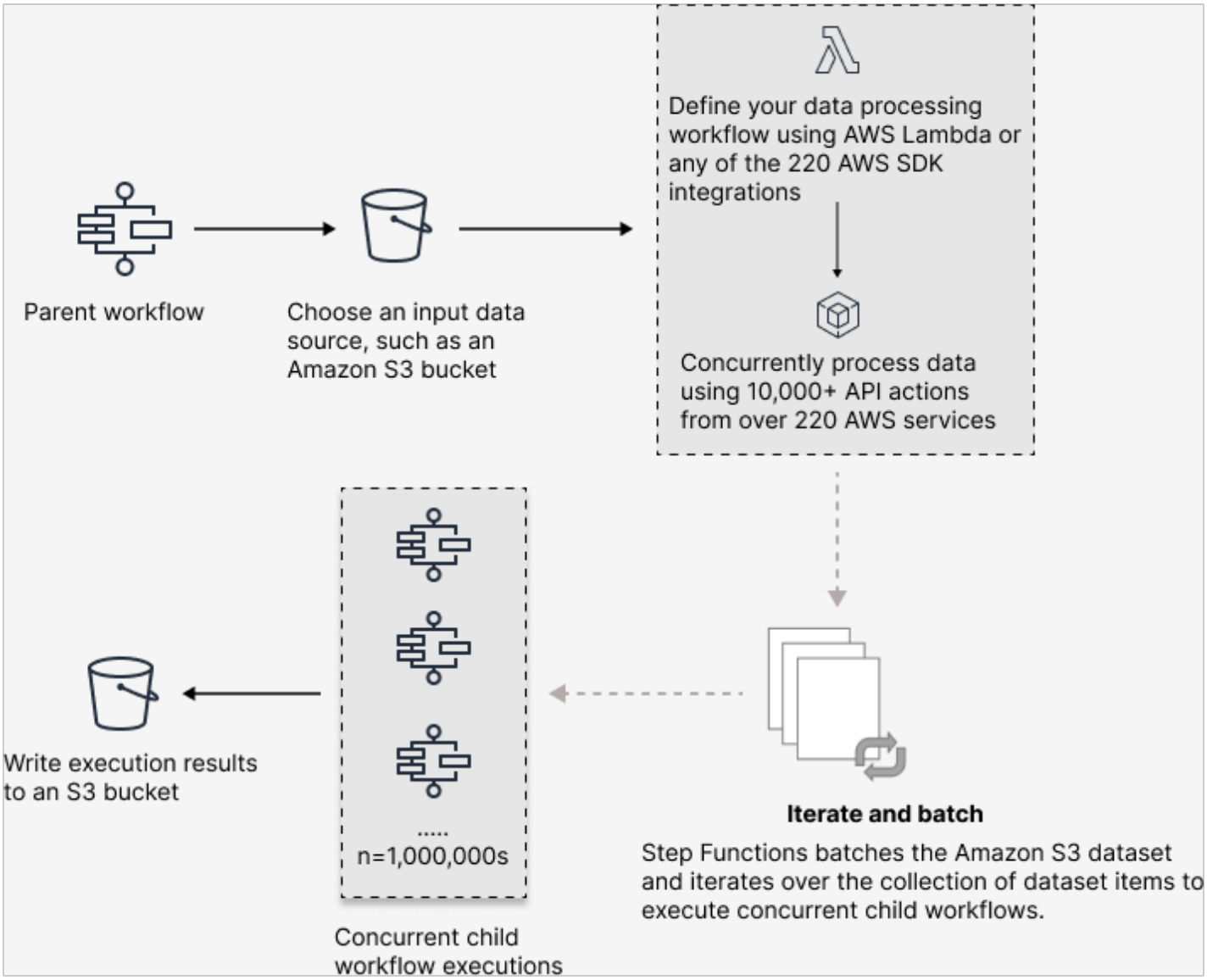
Verwenden von Map State im verteilten Modus zur Orchestrierung umfangreicher parallel Workloads

Mit Step Functions können Sie umfangreiche parallel Workloads orchestrieren, um Aufgaben wie die On-Demand-Verarbeitung halbstrukturierter Daten auszuführen. Mit diesen parallel Workloads

können Sie große Datenquellen, die in Amazon S3 gespeichert sind, gleichzeitig verarbeiten. Sie könnten beispielsweise eine einzelne JSON- oder CSV-Datei verarbeiten, die große Datenmengen enthält. Oder Sie könnten eine große Menge von Amazon S3 S3-Objekten verarbeiten.

Um eine umfangreiche parallel Arbeitslast in Ihren Workflows einzurichten, fügen Sie einen `Map` Status im Modus Verteilt hinzu. Der Kartenstatus verarbeitet Elemente in einem Datensatz gleichzeitig. Ein `Map` Status, der auf Distributed gesetzt ist, wird als Distributed Map-Status bezeichnet. Im Modus Verteilt ermöglicht der `Map` Status eine Verarbeitung mit hoher Parallelität. Im verteilten Modus verarbeitet der `Map` Status die Elemente in der Datenmenge in Iterationen, die als untergeordnete Workflow-Ausführungen bezeichnet werden. Sie können die Anzahl der untergeordneten Workflow-Ausführungen angeben, die parallel ausgeführt werden können. Jede Ausführung eines untergeordneten Workflows hat einen eigenen Ausführungsverlauf, der von dem des übergeordneten Workflows getrennt ist. Wenn Sie nichts angeben, führt Step Functions 10.000 parallele Ausführungen untergeordneter Workflows parallel aus.

In der folgenden Abbildung wird erklärt, wie Sie umfangreiche parallel Workloads in Ihren Workflows einrichten können.



In diesem Thema

Wichtige Begriffe

Verteilter Modus

Ein Verarbeitungsmodus des Kartenstatus. In diesem Modus wird jede Iteration des `Map` Status als untergeordnete Workflow-Ausführung ausgeführt, wodurch eine hohe Parallelität ermöglicht wird. Jede untergeordnete Workflow-Ausführung hat ihren eigenen Ausführungsverlauf, der vom Ausführungsverlauf des übergeordneten Workflows getrennt ist. Dieser Modus unterstützt das Lesen von Eingaben aus großen Amazon S3 S3-Datenquellen.

Status der verteilten Karte

Ein Map-Status, der auf den Modus Verteilte Verarbeitung gesetzt ist.

Arbeitsablauf zuordnen

Eine Reihe von Schritten, die ein `Map` Bundesstaat ausführt.

Übergeordneter Arbeitsablauf

Ein Workflow, der einen oder mehrere Distributed Map-Status enthält.

Ausführung eines untergeordneten Workflows

Eine Iteration des Distributed-Map-Status. Die Ausführung eines untergeordneten Workflows hat einen eigenen Ausführungsverlauf, der vom Ausführungsverlauf des übergeordneten Workflows getrennt ist.

Ordnen Sie Run zu

Wenn Sie einen `Map` Status im verteilten Modus ausführen, erstellt Step Functions eine Map Run-Ressource. Ein Map Run bezieht sich auf eine Reihe von untergeordneten Workflow-Ausführungen, die im Status Distributed Map gestartet werden, sowie auf die Laufzeiteinstellungen, die diese Ausführungen steuern. Step Functions weist Ihrem Map Run einen Amazon-Ressourcennamen (ARN) zu. Sie können einen Map Run in der Step Functions-Konsole untersuchen. Sie können die `DescribeMapRun` API-Aktion auch aufrufen. Ein Map Run sendet auch Metriken an. CloudWatch

Weitere Informationen finden Sie unter Untersuchen der Kartenausführung.

Beispiel für die Definition eines verteilten Kartenzustands

Verwenden Sie den `Map` Status im Modus Verteilt, wenn Sie umfangreiche parallel Workloads orchestrieren müssen, die eine beliebige Kombination der folgenden Bedingungen erfüllen:

- Die Größe Ihres Datensatzes übersteigt 256 KB.
- Der Verlauf der Ausführungsereignisse des Workflows umfasst mehr als 25.000 Einträge.
- Sie benötigen eine Parallelität von mehr als 40 parallel Iterationen.

Das folgende Beispiel für die Zustandsdefinition von Distributed Map spezifiziert den Datensatz als CSV-Datei, die in einem Amazon S3 S3-Bucket gespeichert ist. Es spezifiziert auch eine Lambda-Funktion, die die Daten in jeder Zeile der CSV-Datei verarbeitet. Da in diesem Beispiel eine CSV-Datei verwendet wird, wird auch die Position der CSV-Spaltenüberschriften angegeben. Die vollständige State-Machine-Definition dieses Beispiels finden Sie im Tutorial Kopieren umfangreicher CSV-Daten mit Distributed Map.

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

Berechtigungen zum Ausführen von Distributed Map

Wenn Sie einen Distributed Map-Status in Ihre Workflows aufnehmen, benötigt Step Functions die entsprechenden Berechtigungen, damit die Zustandsmaschinenrolle die `StartExecution` API-Aktion für den Distributed-Map-Status aufrufen kann.

Das folgende Beispiel für eine IAM-Richtlinie gewährt Ihrer State-Machine-Rolle die geringsten Rechte, die für die Ausführung des Status Distributed Map erforderlich sind.

> **Note**
>
> Stellen Sie sicher, dass Sie den Status *stateMachineName* durch den Namen des Zustandsmaschinen ersetzen, in dem Sie den Status Distributed Map verwenden. z. B. `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:*"
    }
  ]
}
```

Darüber hinaus müssen Sie sicherstellen, dass Sie über die geringsten Rechte verfügen, die für den Zugriff auf die im Status Distributed Map verwendeten AWS Ressourcen erforderlich sind, z. B. Amazon S3 S3-Buckets. Weitere Informationen finden Sie unter IAM-Richtlinien für die Verwendung des Distributed Map-Status.

Felder für den Status von Distributed Map

Um den Status „Distributed Map“ in Ihren Workflows zu verwenden, geben Sie eines oder mehrere dieser Felder an. Sie geben diese Felder zusätzlich zu den allgemeinen Statusfeldern an.

`Type` (Erforderlich)

Legt den Zustandstyp fest, z. `Map` B.

`ItemProcessor` (Erforderlich)

Enthält die folgenden JSON-Objekte, die den Verarbeitungsmodus und die Definition des `Map` Status angeben.

- `ProcessorConfig`— Ein JSON-Objekt, das die Konfiguration für den `Map` Status spezifiziert. Dieses Objekt enthält die folgenden Unterfelder:
  - `Mode`— Auf einstellen, **`DISTRIBUTED`** um den `Map` Status im verteilten Modus zu verwenden.

    > **Note**
    >
    > Wenn Sie den `Map` Status in Express-Workflows verwenden, können Sie den Status derzeit nicht `Mode` auf setzen`DISTRIBUTED`. Wenn Sie den `Map` Status

jedoch in Standard-Workflows verwenden, können Sie den Wert `Mode` auf setzen`DISTRIBUTED`.

- `ExecutionType`— Gibt den Ausführungstyp für den Map-Workflow entweder als STANDARD oder EXPRESS an. Sie müssen dieses Feld angeben, wenn Sie es `DISTRIBUTED` für das `Mode` Unterfeld angegeben haben. Weitere Informationen zu Workflowtypen finden Sie unterStandard- und Express-Workflows.

- `StartAt`— Gibt eine Zeichenfolge an, die den ersten Status in einem Workflow angibt. Bei dieser Zeichenfolge wird zwischen Groß- und Kleinschreibung unterschieden und sie muss mit dem Namen eines der Statusobjekte übereinstimmen. Dieser Status wird für jedes Element im Datensatz zuerst ausgeführt. Jede Ausführungseingabe, die Sie für den `Map` Status bereitstellen, wird zuerst an den `StartAt` Status übergeben.
- `States`— Ein JSON-Objekt, das eine durch Kommas getrennte Gruppe von Zuständen enthält. In diesem Objekt definieren Sie die. Map workflow

`ItemReader`

Gibt einen Datensatz und seinen Speicherort an. Der `Map` Staat erhält seine Eingabedaten aus dem angegebenen Datensatz.

Im verteilten Modus können Sie entweder eine JSON-Nutzlast, die aus einem früheren Status übergeben wurde, oder eine umfangreiche Amazon S3 S3-Datenquelle als Datensatz verwenden. Weitere Informationen finden Sie unter ItemReader.

`ItemsPath` (Optional)

Gibt einen Referenzpfad an, der die JsonPathSyntax verwendet, um den JSON-Knoten auszuwählen, der ein Array von Elementen innerhalb der Statuseingabe enthält.

Im verteilten Modus geben Sie dieses Feld nur an, wenn Sie ein JSON-Array aus einem vorherigen Schritt als Statuseingabe verwenden. Weitere Informationen finden Sie unter ItemsPath.

`ItemSelector` (Optional)

Überschreibt die Werte einzelner Datensatzelemente, bevor sie an jede `Map` State-Iteration weitergegeben werden.

In diesem Feld geben Sie eine gültige JSON-Eingabe an, die eine Sammlung von Schlüssel-Wert-Paaren enthält. Bei diesen Paaren kann es sich entweder um statische Werte handeln, die Sie in Ihrer Zustandsmaschinen-Definition definieren, um Werte, die mithilfe eines Pfads aus der

Zustandseingabe ausgewählt wurden, oder um Werte, auf die über das Kontextobjekt zugegriffen wird. Weitere Informationen finden Sie unter ItemSelector.

`ItemBatcher` (Optional)

Gibt an, dass die Datensatzelemente stapelweise verarbeitet werden sollen. Jede untergeordnete Workflow-Ausführung erhält dann einen Stapel dieser Elemente als Eingabe. Weitere Informationen finden Sie unter ItemBatcher.

`MaxConcurrency` (Optional)

Gibt die Anzahl der untergeordneten Workflow-Ausführungen an, die parallel ausgeführt werden können. Der Interpreter erlaubt nur bis zu der angegebenen Anzahl parallel untergeordneter Workflow-Ausführungen. Wenn Sie keinen Parallelitätswert angeben oder ihn auf Null setzen, begrenzt Step Functions die Parallelität nicht und führt 10.000 parallel untergeordnete Workflow-Ausführungen aus.

> **Note**
>
> Sie können zwar ein höheres Parallelitätslimit für parallel untergeordnete Workflow-Ausführungen angeben, wir empfehlen jedoch, die Kapazität eines AWS Downstream-Dienstes nicht zu überschreiten, z. B. AWS Lambda

`MaxConcurrencyPath` (Optional)

Wenn Sie mithilfe eines Referenzpfads dynamisch aus der Statuseingabe einen maximalen Parallelitätswert angeben möchten, verwenden Sie. `MaxConcurrencyPath` Nach der Auflösung muss der Referenzpfad ein Feld auswählen, dessen Wert eine nicht negative Ganzzahl ist.

> **Note**
>
> Ein `Map` Status kann nicht sowohl als auch `MaxConcurrency` enthalten. `MaxConcurrencyPath`

`ToleratedFailurePercentage` (Optional)

Definiert den Prozentsatz fehlgeschlagener Elemente, der in einem Map-Run toleriert werden soll. Der Map Run schlägt automatisch fehl, wenn er diesen Prozentsatz überschreitet. Step Functions berechnet den Prozentsatz der fehlgeschlagenen Elemente als Ergebnis der Gesamtzahl der

fehlgeschlagenen Elemente oder der Zeitüberschreitung dividiert durch die Gesamtzahl der Elemente. Sie müssen einen Wert zwischen Null und 100 angeben. Weitere Informationen finden Sie unter Schwellenwert für tolerierte Fehler im Distributed-Map-Status.

`ToleratedFailurePercentagePath` (Optional)

Wenn Sie einen prozentualen Wert für tolerierte Fehler dynamisch aus der Statuseingabe mithilfe eines Referenzpfads angeben möchten, verwenden Sie`ToleratedFailurePercentagePath`. Wenn das Problem gelöst ist, muss der Referenzpfad ein Feld auswählen, dessen Wert zwischen Null und 100 liegt.

`ToleratedFailureCount` (Optional)

Definiert die Anzahl der fehlgeschlagenen Elemente, die in einem Map-Run toleriert werden sollen. Der Map Run schlägt automatisch fehl, wenn er diese Anzahl überschreitet. Weitere Informationen finden Sie unter Schwellenwert für tolerierte Fehler im Distributed-Map-Status.

`ToleratedFailureCountPath` (Optional)

Wenn Sie einen Wert für die Anzahl tolerierter Fehler dynamisch aus der Statuseingabe mithilfe eines Referenzpfads angeben möchten, verwenden Sie`ToleratedFailureCountPath`. Wenn das Problem gelöst ist, muss der Referenzpfad ein Feld auswählen, dessen Wert eine nicht negative Ganzzahl ist.

`Label` (Optional)

Eine Zeichenfolge, die einen `Map` Bundesstaat eindeutig identifiziert. Für jeden Map Run fügt Step Functions das Label zum Map Run-ARN hinzu. Im Folgenden finden Sie ein Beispiel für einen Map Run-ARN mit einem benutzerdefinierten Label namens`demoLabel`:

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

Wenn Sie kein Label angeben, generiert Step Functions automatisch ein eindeutiges Label.

> **Note**
>
> Beschriftungen dürfen nicht länger als 40 Zeichen sein, müssen innerhalb einer State-Machine-Definition eindeutig sein und dürfen keines der folgenden Zeichen enthalten:
>
> - Leerzeichen
> - Platzhalterzeichen (`?` `*`)
> - Klammerzeichen (`<` `>` `{` `}` `[` `]`)

- Sonderzeichen (`: ; , \ | ^ ~ $ # % & ` "`)
- Steuerzeichen (`\\u0000`- `\\u001f` oder `\\u007f` -`\\u009f`).

Mit Step Functions können Sie Namen für Zustandsmaschinen, Ausführungen, Aktivitäten und Labels erstellen, die Nicht-ASCII-Zeichen enthalten. Diese Nicht-ASCII-Namen funktionieren nicht mit Amazon. CloudWatch Um sicherzustellen, dass Sie CloudWatch Messwerte verfolgen können, wählen Sie einen Namen, der nur ASCII-Zeichen verwendet.

`ResultWriter` (Optional)

Gibt den Amazon S3 S3-Speicherort an, an den Step Functions alle untergeordneten Workflow-Ausführungsergebnisse schreibt.

Step Functions konsolidiert alle Ausführungsdaten des untergeordneten Workflows, wie z. B. Ausführungseingabe und -ausgabe, ARN und Ausführungsstatus. Anschließend werden Ausführungen mit demselben Status in die entsprechenden Dateien am angegebenen Amazon S3 S3-Speicherort exportiert. Weitere Informationen finden Sie unter ResultWriter.

Wenn Sie die `Map` Statusergebnisse nicht exportieren, wird ein Array mit allen Ergebnissen der untergeordneten Workflow-Ausführung zurückgegeben. Beispielsweise:

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

`ResultPath` (Optional)

Gibt an, wo in der Eingabe die Ausgabe der Iterationen platziert werden soll. Die Eingabe wird dann gemäß den Angaben des `OutputPath`Felds gefiltert, falls vorhanden, bevor sie als Ausgabe für den Status übergeben wird. Weitere Informationen finden Sie unter Verarbeitung von Eingabe und Ausgabe.

`ResultSelector` (Optional)

Übergibt eine Sammlung von Schlüssel-Wert-Paaren, wobei die Werte statisch sind oder aus dem Ergebnis ausgewählt werden. Weitere Informationen finden Sie unter ResultSelector.

> **Tip**
>
> Wenn der Status Parallel oder Map, den Sie in Ihren Zustandsmaschinen verwenden, ein Array von Arrays zurückgibt, können Sie diese mit dem Feld in ein flaches Array

umwandeln. ResultSelector Weitere Informationen finden Sie unter Reduzieren eines Arrays von Arrays.

`Retry` (Optional)

Eine Reihe von Objekten, sogenannte Retriers, die eine Wiederholungsrichtlinie definieren. Eine Ausführung verwendet die Wiederholungsrichtlinie, wenn im Status Laufzeitfehler auftreten. Weitere Informationen finden Sie unter Beispiele für Zustandsmaschinen mit Retry und Catch.

> **Note**
>
> Wenn Sie Retrier für den Status Distributed Map definieren, gilt die Wiederholungsrichtlinie für alle untergeordneten Workflow-Ausführungen, die im Status gestartet wurden. `Map` Stellen Sie sich beispielsweise vor, Ihr `Map` Bundesstaat hat drei untergeordnete Workflow-Ausführungen gestartet, von denen eine fehlschlägt. Wenn der Fehler auftritt, verwendet die Ausführung das `Retry` Feld, sofern definiert, für den `Map` Status. Die Wiederholungsrichtlinie gilt für alle untergeordneten Workflow-Ausführungen und nicht nur für die fehlgeschlagene Ausführung. Wenn eine oder mehrere untergeordnete Workflow-Ausführungen fehlschlagen, schlägt der Map Run fehl. Wenn Sie einen `Map` Status erneut versuchen, wird ein neuer Map Run erstellt.

`Catch` (Optional)

Ein Array von Objekten namens Catcher, die einen Fallback-Zustand definieren. Step Functions verwendet die in definierten Catcher, `Catch` wenn der Status auf Laufzeitfehler stößt. Wenn ein Fehler auftritt, verwendet die Ausführung zunächst alle in definierten Retrier. `Retry` Wenn die Wiederholungsrichtlinie nicht definiert oder erschöpft ist, verwendet die Ausführung ihre Catcher, sofern definiert. Weitere Informationen finden Sie unter Fallback-Zustände.

Nächste Schritte

Weitere Informationen zum Status von Distributed Map finden Sie in den folgenden Ressourcen:

- Verarbeitung von Eingabe und Ausgabe

  Um die Eingabe zu konfigurieren, die ein Distributed-Map-Status empfängt, und die Ausgabe, die er generiert, stellt Step Functions die folgenden Felder bereit:

- ItemReader
- ItemsPath
- ItemSelector
- ItemBatcher
- ResultWriter
- Analysieren einer CSV-Eingabedatei

Zusätzlich zu diesen Feldern bietet Ihnen Step Functions auch die Möglichkeit, einen Schwellenwert für tolerierte Fehler für Distributed Map zu definieren. Mit diesem Wert können Sie die maximale Anzahl oder den Prozentsatz fehlgeschlagener Elemente als Fehlerschwellenwert für einen Map-Lauf angeben. Weitere Informationen zur Konfiguration des Schwellenwerts für tolerierte Fehler finden Sie unterSchwellenwert für tolerierte Fehler im Distributed-Map-Status.

- Der Status „Distributed Map“ wird verwendet

  In den folgenden Tutorials und Beispielprojekten finden Sie Informationen zu den ersten Schritten mit der Verwendung von Distributed Map State.

  - Erste Schritte mit der Verwendung von Distributed Map State
  - Verarbeitung des gesamten Datenstapels mit einer Lambda-Funktion
  - Verarbeiten einzelner Datenelemente mit einer Lambda-Funktion
  - Beispielprojekt: Verarbeiten Sie eine CSV-Datei mit Distributed Map
  - Beispielprojekt: Daten in einem Amazon S3 S3-Bucket mit Distributed Map verarbeiten
- Untersuchen Sie die Ausführung des Distributed Map-Status

  Die Step Functions Functions-Konsole bietet eine Seite mit den Map-Run-Details, auf der alle Informationen zur Ausführung eines Distributed Map-Status angezeigt werden. Informationen darüber, wie Sie die auf dieser Seite angezeigten Informationen überprüfen können, finden Sie unterUntersuchen der Kartenausführung.

Schwellenwert für tolerierte Fehler im Distributed-Map-Status

Wenn Sie umfangreiche parallel Workloads orchestrieren, können Sie auch einen Schwellenwert für tolerierte Fehler definieren. Mit diesem Wert können Sie die maximale Anzahl oder den Prozentsatz fehlgeschlagener Elemente als Schwellenwert für einen Kartenlauf angeben. Je nachdem, welchen Wert Sie angeben, schlägt Ihr Map Run automatisch fehl, wenn er den Schwellenwert überschreitet. Wenn Sie beide Werte angeben, schlägt der Workflow fehl, wenn er einen der Werte überschreitet.

Wenn Sie einen Schwellenwert angeben, können Sie bei einer bestimmten Anzahl von Elementen Fehler beheben, bevor der gesamte Map Run fehlschlägt. Step Functions gibt einen `States.ExceedToleratedFailureThreshold` Fehler zurück, wenn der Map Run fehlschlägt, weil der angegebene Schwellenwert überschritten wird.

> **Note**
>
> Step Functions kann weiterhin untergeordnete Workflows in einem Map Run ausführen, auch wenn der tolerierte Fehlerschwellenwert überschritten wurde, aber bevor der Map Run fehlschlägt.

Um den Schwellenwert in Workflow Studio anzugeben, wählen Sie unter Zusätzliche Konfiguration im Feld Runtime-Einstellungen die Option Schwellenwert für tolerierte Fehler festlegen aus.

Prozentsatz der tolerierten Fehler

Definiert den Prozentsatz fehlgeschlagener Elemente, der toleriert werden soll. Ihr Map Run schlägt fehl, wenn dieser Wert überschritten wird. Step Functions berechnet den Prozentsatz der fehlgeschlagenen Elemente als Ergebnis der Gesamtzahl der fehlgeschlagenen Elemente oder der Zeitüberschreitung dividiert durch die Gesamtzahl der Elemente. Sie müssen einen Wert zwischen Null und 100 angeben. Der Standardwert für den Prozentsatz ist Null, was bedeutet, dass der Workflow fehlschlägt, wenn eine der untergeordneten Workflow-Ausführungen fehlschlägt oder ein Timeout auftritt. Wenn Sie den Prozentsatz auf 100 angeben, schlägt der Workflow auch dann nicht fehl, wenn alle untergeordneten Workflow-Ausführungen fehlschlagen.

Alternativ können Sie den Prozentsatz als Referenzpfad zu einem vorhandenen Schlüssel-Wert-Paar in Ihrer Distributed Map-Statuseingabe angeben. Dieser Pfad muss zur Laufzeit in eine positive Ganzzahl zwischen 0 und 100 aufgelöst werden. Sie geben den Referenzpfad im `ToleratedFailurePercentagePath` Unterfeld an.

Beispielsweise angesichts der folgenden Eingabe:

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

Sie können den Prozentsatz mithilfe eines Referenzpfads zu dieser Eingabe wie folgt angeben:

```
{
```

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

> **Important**
>
> Sie können entweder `ToleratedFailurePercentage` oder`ToleratedFailurePercentagePath`, aber nicht beide in Ihrer Statusdefinition für Distributed Map angeben.

Anzahl tolerierter Fehler

Definiert die Anzahl der fehlgeschlagenen Elemente, die toleriert werden sollen. Ihr Map Run schlägt fehl, wenn dieser Wert überschritten wird.

Alternativ können Sie die Anzahl als Referenzpfad zu einem vorhandenen Schlüssel-Wert-Paar in Ihrer Distributed Map-Statuseingabe angeben. Dieser Pfad muss zur Laufzeit in eine positive Ganzzahl aufgelöst werden. Sie geben den Referenzpfad im `ToleratedFailureCountPath` Unterfeld an.

Beispielsweise angesichts der folgenden Eingabe:

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

Sie können die Nummer mithilfe eines Referenzpfads zu dieser Eingabe wie folgt angeben:

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

```
  }
}
```

> **Important**
>
> Sie können entweder `ToleratedFailureCount` oder`ToleratedFailureCountPath`, aber nicht beide in der Status-Definition von Distributed Map angeben.

Übergänge

Wenn Sie eine neue Ausführung Ihrer Zustandsmaschine starten, beginnt das System mit dem Status, auf den im `StartAt` Feld der obersten Ebene verwiesen wird. Dieses Feld, das als Zeichenfolge angegeben wird, muss exakt mit dem Namen eines Status im Workflow übereinstimmen, einschließlich Groß- und Kleinschreibung.

Wenn ein Status ausgeführt wurde, AWS Step Functions verwendet es den Wert des `Next` Felds, um den nächsten Status zu bestimmen, zu dem gewechselt werden soll.

`Next`Felder geben auch die Namen der Bundesstaaten als Zeichenketten an. Bei dieser Zeichenfolge wird zwischen Groß- und Kleinschreibung unterschieden und sie muss exakt mit dem Namen eines in der State-Machine-Beschreibung angegebenen Zustands übereinstimmen

Zum Beispiel umfasst der folgende Zustand einen Übergang zu `NextState`.

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

In den meisten Bundesstaaten ist nur eine einzige Übergangsregel für das `Next` Feld zulässig. In bestimmten Flusssteuerungszuständen, wie z. B. einem `Choice` Status, können Sie jedoch mehrere Übergangsregeln angeben, von denen jede über ein eigenes `Next` Feld verfügt. Amazon States Language stellt Details zu den einzelnen Zustandstypen, die Sie angeben können, bereit, einschließlich Informationen zur Angabe von Übergängen.

Zustände können mehrere eingehende Übergänge von anderen Zuständen haben.

Der Vorgang wiederholt sich, bis er entweder einen Terminalstatus (einen Zustand mit `"Type": Succeed"Type": Fail`, oder`"End": true`) erreicht oder ein Laufzeitfehler auftritt.

Wenn Sie redriveeine Ausführung ausführen, wird dies als Zustandsübergang betrachtet. Darüber hinaus werden alle Zustände, die in a erneut ausgeführt redrive werden, ebenfalls als Zustandsübergänge betrachtet.

Die folgenden Regeln gelten für Zustände in einem Zustandsautomaten:

- Zustände können innerhalb des umschließenden Blocks in beliebiger Reihenfolge auftreten. Die Reihenfolge, in der sie aufgeführt sind, hat jedoch keinen Einfluss auf die Reihenfolge, in der sie ausgeführt werden. Diese Reihenfolge wird durch die Inhalte der Zustände bestimmt.
- Innerhalb einer Zustandsmaschine kann es nur einen Zustand geben, der als `start` Staat bezeichnet wird. Der `start` Status wird durch den Wert des `StartAt` Felds in der Struktur der obersten Ebene definiert.
- Abhängig von Ihrer Zustandsmaschinenlogik — wenn Ihre Zustandsmaschine beispielsweise mehrere Logikzweige hat — können Sie mehr als einen `end` Zustand haben.
- Wenn Ihre Zustandsmaschine nur aus einem Zustand besteht, kann es sich sowohl um den Start- als auch um den Endzustand handeln.

Übergänge im Status „Verteilte Karte“

Wenn Sie den `Map` Status im Modus Verteilt verwenden, wird Ihnen für jede untergeordnete Workflow-Ausführung, die mit dem Status Distributed Map gestartet wird, ein Zustandsübergang berechnet. Wenn Sie den `Map` Status im Inline-Modus verwenden, wird Ihnen nicht für jede Iteration des Status Inline Map ein Zustandsübergang in Rechnung gestellt.

Sie können die Kosten optimieren, indem Sie den `Map` Status im Modus Verteilt verwenden und einen verschachtelten Workflow in die `Map` Statusdefinition aufnehmen. Der Status „Distributed Map“ bietet auch mehr Nutzen, wenn Sie untergeordnete Workflow-Ausführungen vom Typ Express starten. Step Functions speichert die Antwort und den Status der untergeordneten Express-Workflow-Ausführungen, wodurch die Notwendigkeit reduziert wird, Ausführungsdaten in CloudWatch Logs zu speichern. Sie können auch auf die in einem Distributed-Map-Status verfügbaren Ablaufsteuerungen zugreifen, z. B. die Definition von Fehlerschwellenwerten oder die Stapelverarbeitung einer Gruppe von Elementen. Informationen zu den Preisen von Step Functions finden Sie unter AWS Step FunctionsPreise.

Zustandsautomatendaten

Zustandsautomatendaten nehmen die folgenden Formen an:

- Die erste Eingabe in einen Zustandsautomaten
- Daten, die zwischen Zuständen übergeben werden
- Die Ausgabe eines Zustandsautomaten

In diesem Abschnitt wird beschrieben, wie Zustandsautomatendaten in AWS Step Functions formatiert und verwendet werden.

Themen

Datenformat

Zustandsmaschinendaten werden durch JSON-Text dargestellt. Sie können Werte für eine State-Maschine bereitstellen, indem Sie jeden von JSON unterstützten Datentyp verwenden.

> **Note**
>
> - Zahlen im JSON-Textformat entsprechen der JavaScript Semantik. Diese Zahlen entsprechen in der Regel IEEE-854 Werten mit doppelter Genauigkeit.
> - Das Folgende ist gültiger JSON-Text:
>   - Eigenständige, durch Anführungszeichen getrennte Zeichenketten
>   - Objekte
>   - Arrays
>   - Zahlen
>   - Boolesche Werte
>   - `null`
> - Die Ausgabe eines Zustands wird zur Eingabe für den nächsten Zustand. Mithilfe der Eingabe- und Ausgabeverarbeitung können Sie Staaten jedoch darauf beschränken, an einer Teilmenge der Eingabedaten zu arbeiten.

Eingabe/Ausgabe von Zustandsautomaten

Sie können Ihre ersten Eingabedaten auf zwei Arten an eine AWS Step Functions Zustandsmaschine weitergeben. Sie können die Daten an eine `StartExecution` Aktion übergeben, wenn Sie eine Ausführung starten. Sie können die Daten auch von der Step Functions-Konsole aus an die State-Maschine übergeben. Anfängliche Daten werden an den `StartAt`-Zustand des Zustandsautomaten übergeben. Wenn keine Eingabe getätigt wird, ist das Objekt standardmäßig leer (`{}`).

Die Ausgabe der Ausführung wird vom letzten Zustand zurückgegeben (`terminal`). Diese Ausgabe erscheint als JSON-Text im Ergebnis der Ausführung.

Bei Standard-Workflows können Sie Ausführungsergebnisse aus der Ausführungshistorie abrufen, indem Sie externe Aufrufer verwenden, z. B. die `DescribeExecution` Aktion. Sie können die Ausführungsergebnisse auf der Step Functions-Konsole einsehen.

Wenn Sie für Express-Workflows die Protokollierung aktiviert haben, können Sie Ergebnisse aus CloudWatch Protokollen abrufen oder die Ausführungen in der Step Functions-Konsole anzeigen und debuggen. Weitere Informationen erhalten Sie unter Protokollierung mitCloudWatchProtokolle und Anzeigen und Debuggen von Ausführungen in der Step Functions-Konsole.

Sie sollten auch Kontingente berücksichtigen, die sich auf Ihre Staatsmaschine beziehen. Weitere Informationen finden Sie unter Kontingente

Eingabe/Ausgabe von Zuständen

Die Eingabe jedes Zustands besteht aus JSON-Text aus dem vorherigen Zustand oder, im Falle des Zustands `StartAt`, der Eingabe in die Ausführung. Bestimmte Flusssteuerungs-Zustände geben ihre Eingabe als Echo an ihre Ausgabe weiter.

Im folgenden Beispiel fügt der Zustandsautomat zwei Zahlen zusammen.

1. Definieren Sie die AWS Lambda-Funktion.

```
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. Definieren Sie den -Zustandsautomaten.

```
{
  "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. Starten Sie eine Ausführung mit dem folgenden JSON-Text.

```
{ "numbers": [3, 4] }
```

Der Add Staat empfängt den JSON-Text und übergibt ihn an die Lambda-Funktion.

Die Lambda-Funktion gibt das Ergebnis der Berechnung an den Status zurück.

Der Zustand gibt den folgenden Wert in die Ausgabe.

```
{ "result": 7 }
```

Da Add auch der letzte Zustand in dem Zustandsautomaten ist, wird dieser Wert als Ausgabe des Zustandsautomaten zurückgegeben.

Wenn der letzte Zustand keine Ausgabe zurückgibt, gibt der Zustandsautomat ein leeres Objekt zurück ({}).

Weitere Informationen finden Sie unter Eingabe- und Ausgabeverarbeitung in Step Functions.

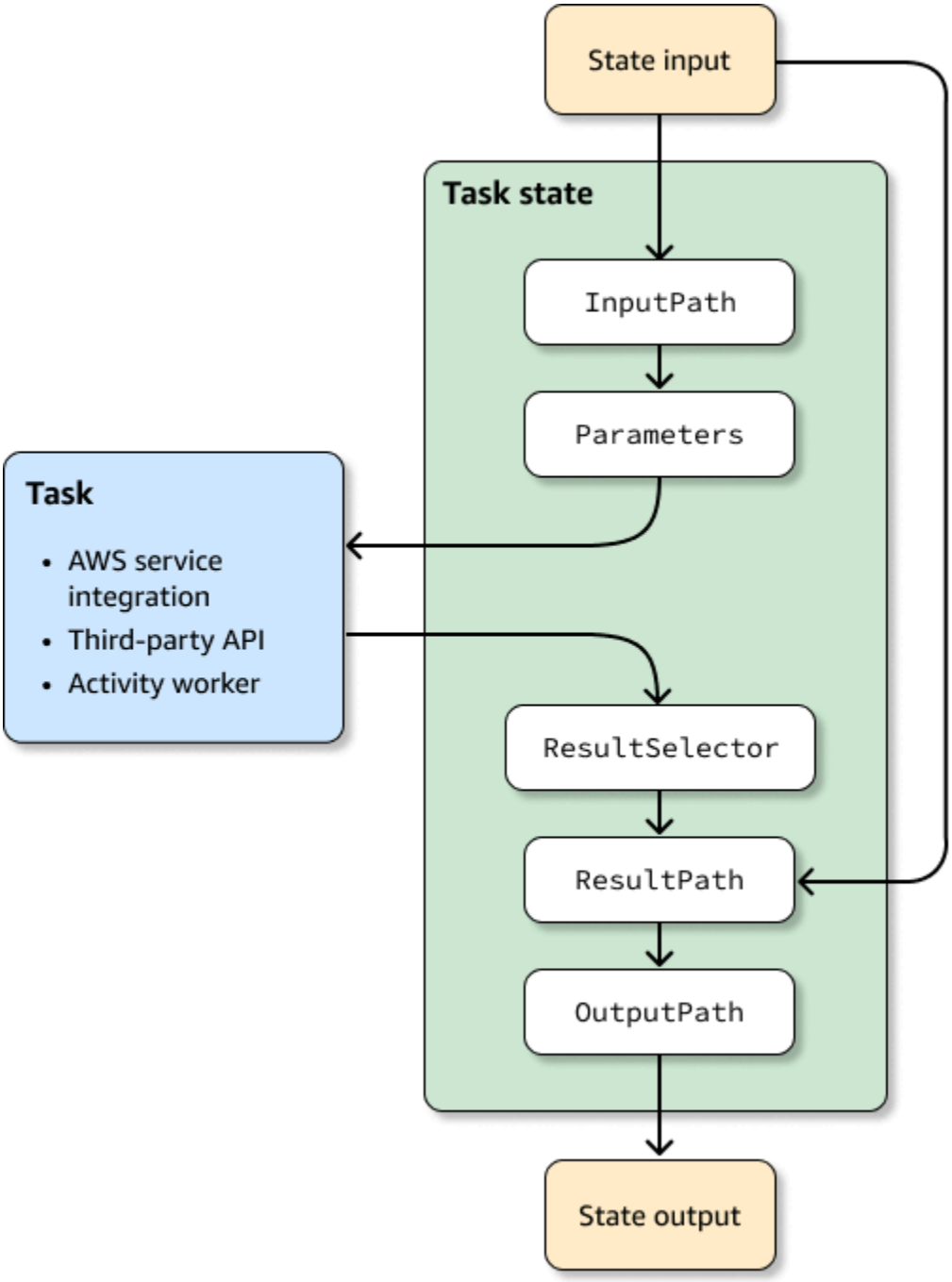
Eingabe- und Ausgabeverarbeitung in Step Functions

Eine Step Functions Functions-Ausführung empfängt einen JSON-Text als Eingabe und übergibt diese Eingabe an den ersten Status im Workflow. Einzelne Status erhalten JSON als Eingabe und leiten in der Regel JSON als Ausgabe an den nächsten Status weiter. Das Wissen, wie diese Informationen von Status zu Status fließen, und das Lernen, wie Sie diese Daten filtern und bearbeiten, ist von entscheidender Bedeutung für die effektive Gestaltung und Implementierung von Workflows in AWS Step Functions.

In der Sprache Amazon States filtern und steuern diese Felder den JSON-Fluss von Bundesstaat zu Bundesstaat:

- `InputPath`
- `Parameters`
- `ResultSelector`
- `ResultPath`
- `OutputPath`

Das folgende Diagramm zeigt, wie sich JSON-Informationen durch einen Aufgabenstatus bewegen. `InputPath`wählt aus, welche Teile der JSON-Eingabe an die Aufgabe des `Task` Zustands übergeben werden sollen (z. B. eine AWS Lambda Funktion). `ResultPath`wählt dann aus, welche Kombination aus der Statuseingabe und dem Aufgabenergebnis an die Ausgabe übergeben werden soll. `OutputPath`kann die JSON-Ausgabe filtern, um die Informationen, die an die Ausgabe übergeben werden, weiter einzuschränken.



`InputPath`,`Parameters`, `ResultSelector``ResultPath`, und `OutputPath` jeder manipuliert JSON, während es sich durch jeden Status in Ihrem Workflow bewegt.

Alle können Pfade verwenden, um Teile des JSON-Codes von der Eingabe oder vom Ergebnis auszuwählen. Ein Pfad ist eine Zeichenfolge, die mit$, beginnt und Knoten innerhalb von JSON-Text identifiziert. Step Functions Functions-Pfade verwenden JsonPathSyntax.

> **Tip**
>
> Verwenden Sie den Datenflusssimulator in der Step Functions Functions-Konsole, um die JSON-Pfadsyntax zu testen, um besser zu verstehen, wie Daten innerhalb eines Zustands manipuliert werden, und um zu sehen, wie Daten zwischen Staaten weitergegeben werden.

> **Tip**
>
> Ein Beispiel für einen Workflow, der Eingabe- und Ausgabeverarbeitung umfasst AWS-Konto, finden Sie in Modul 6 — Eingabe- und Ausgabeverarbeitung des AWS Step Functions Workshops.

Themen

Pfade

In der Sprache von Amazon States ist ein Pfad eine Zeichenfolge, $ die mit beginnt und mit der Sie Komponenten innerhalb von JSON-Text identifizieren können. Pfade folgen der JsonPathSyntax. Sie können einen Pfad für den Zugriff auf Teilsätze der Eingaben beim Festlegen von Werten für `InputPath`, `ResultPath`, und `OutputPath` angeben. Weitere Informationen finden Sie unter Eingabe- und Ausgabeverarbeitung in Step Functions.

> **Note**
>
> Sie können auch einen JSON-Knoten des Eingabe- oder des Kontextmenü-Objekts angeben, indem Sie Pfade in dem `Parameters`-Feld einer Zustandsdefinition verwenden.. Siehe Parameter an eine Service-API übergeben.

Sie müssen die Notation mit eckigen Klammern verwenden, wenn Ihr Feldname ein Zeichen enthält, das nicht in der `member-name-shorthand` Definition der JsonPath ABNF-Regel enthalten ist. Um Sonderzeichen wie Satzzeichen (ausgenommen_) zu kodieren, müssen Sie daher die Notation mit eckigen Klammern verwenden. z. B. `$.abc.['def ghi']`.

Referenzpfade

Ein Referenzpfad ist ein Pfad, dessen Syntax auf eine Weise begrenzt ist, dass er nur einen einzelnen Knoten in einer JSON-Struktur identifizieren kann:

- Sie können auf Objektfelder zugreifen, indem Sie nur Punkt (`.`)- und Klammer (`[ ]`)-Notation verwenden.
- Funktionen wie `length()` werden nicht unterstützt.
- Lexikalische Operatoren, die nicht symbolischer Natur sind, z. B. werden nicht unterstützt. `subsetof`
- Das Filtern nach regulären Ausdrücken oder durch Verweisen auf einen anderen Wert in der JSON-Struktur wird nicht unterstützt.
- Die Operatoren `@,,:`, und `?` werden nicht unterstützt

Zum Beispiel, wenn Zustandseingabedaten die folgenden Werte enthalten:

```
{
  "foo": 123,
  "bar": ["a", "b", "c"],
  "car": {
      "cdr": true
  }
}
```

In diesem Fall würden die folgenden Referenzpfade Folgendes zurückgeben:

```
$.foo => 123
$.bar => ["a", "b", "c"]
$.car.cdr => true
```

Bestimmte Zustände verwenden Pfade und Referenzpfade, um den Ablauf eines Zustandsautomaten zu steuern oder um Einstellungen oder Optionen eines Zustands zu konfigurieren. Weitere Informationen finden Sie unter Modellieren der Workflow-Eingabe- und Ausgabepfadverarbeitung mit dem Datenflusssimulator und Effektives Verwenden von JSONPath in. AWS Step Functions

Ein Array von Arrays reduzieren

Wenn der Zuordnung Status Parallel oder in Ihren Zustandsmaschinen ein Array von Arrays zurückgibt, können Sie diese mit dem Feld in ein flaches Array umwandeln. ResultSelector Sie können dieses Feld in die Zustandsdefinition Parallel oder Map aufnehmen, um das Ergebnis dieser Zustände zu manipulieren.

Um Arrays zu reduzieren, verwenden Sie die JMESPath-Syntax `[*]` in dem `ResultSelector` Feld, wie im folgenden Beispiel gezeigt.

```
"ResultSelector": {
    "flattenArray.$": "$[*][*]"
  }
```

Beispiele, die zeigen, wie ein Array geglättet wird, finden Sie in Schritt 3 der folgenden Tutorials:

- Verarbeitung des gesamten Datenstapels mit einer Lambda-Funktion
- Verarbeiten einzelner Datenelemente mit einer Lambda-Funktion

InputPath, Parameter und ResultSelector

Die `ResultSelector` Felder`InputPath`, `Parameters` und bieten eine Möglichkeit, JSON zu manipulieren, während es sich durch Ihren Workflow bewegt. `InputPath`kann die übergebene Eingabe einschränken, indem die JSON-Notation mithilfe eines Pfads gefiltert wird (siehePfade). Das Feld `Parameters` ermöglicht Ihnen die Übergabe einer Sammlung von Schlüssel-Wert-Paaren, wobei die Werte entweder statische Werte sind, die in der Definition des Zustandsautomaten ausgewählt werden, oder aus der Eingabe mithilfe eines Pfads ausgewählt werden. Das `ResultSelector` Feld bietet eine Möglichkeit, das Ergebnis des Status zu manipulieren, bevor `ResultPath` es angewendet wird.

AWS Step Functions wendet zuerst das `InputPath` Feld und dann das `Parameters` Feld an. Sie können zuerst mit `InputPath` Ihre unformatierte Eingabe zu einer gewünschten Auswahl filtern und anschließend `Parameters` anwenden, um die Eingabe weiter zu bearbeiten oder neue Werte hinzuzufügen. Anschließend können Sie das `ResultSelector` Feld verwenden, um die Ausgabe des Status zu bearbeiten, bevor er angewendet `ResultPath` wird.

> **Tip**
>
> Verwenden Sie den Datenflusssimulator in der Step Functions Functions-Konsole, um die JSON-Pfadsyntax zu testen, um besser zu verstehen, wie Daten innerhalb eines Zustands manipuliert werden, und um zu sehen, wie Daten zwischen Staaten weitergegeben werden.

InputPath

Verwenden Sie `InputPath`, um einen Teil der Zustandseingabe auszuwählen.

Nehmen Sie beispielsweise an, dass die Eingabe für Ihren zustand Folgendes enthält.

```
{
  "comment": "Example for InputPath.",
  "dataset1": {
    "val1": 1,
    "val2": 2,
    "val3": 3
  },
  "dataset2": {
    "val1": "a",
    "val2": "b",
    "val3": "c"
  }
}
```

Dann könnten Sie `InputPath` anwenden.

```
"InputPath": "$.dataset2",
```

Mit dem vorherigen `InputPath` wird der folgende JSON-Code als Eingabe übergeben.

```
{
```

```
  "val1": "a",
  "val2": "b",
  "val3": "c"
}
```

> **Note**
>
> Ein Pfad kann zu einer Auswahl von Werten führen. Betrachten Sie das folgende Beispiel.
>
> ```
> { "a": [1, 2, 3, 4] }
> ```
>
> Wenn Sie den Pfad `$.a[0:2]` anwenden, kommt es zu folgendem Ergebnis.
>
> ```
> [ 1, 2 ]
> ```

Parameter

In diesem Abschnitt werden die verschiedenen Möglichkeiten beschrieben, wie Sie das Parameterfeld verwenden können.

Schlüssel-Wert-Paare

Verwenden Sie das `Parameters` Feld, um eine Sammlung von Schlüssel-Wert-Paaren zu erstellen, die als Eingabe übergeben werden. Die Werte können jeweils entweder statische Werte sein, die Sie in der Definition des Zustandsautomaten festlegen, oder aus der Eingabe mithilfe eines Pfades ausgewählt werden. Bei Schlüssel-Wert-Paaren, bei denen der Wert mit einem Pfad ausgewählt wird, muss der Schlüsselname mit `.$` enden.

Nehmen Sie beispielsweise an, Sie stellen die folgenden Eingaben bereit.

```
{
  "comment": "Example for Parameters.",
  "product": {
    "details": {
       "color": "blue",
       "size": "small",
       "material": "cotton"
    },
```

```
    "availability": "in stock",
    "sku": "2317",
    "cost": "$23"
  }
}
```

Um einige der Informationen auszuwählen, können Sie diese in der Definition Ihres Zustandsautomaten festlegen.

```
"Parameters": {
        "comment": "Selecting what I care about.",
        "MyDetails": {
          "size.$": "$.product.details.size",
          "exists.$": "$.product.availability",
          "StaticValue": "foo"
        }
      },
```

Mit der vorherigen Eingabe und dem Feld `Parameters` wird der folgende JSON-Code übergeben.

```
{
  "comment": "Selecting what I care about.",
  "MyDetails": {
      "size": "small",
      "exists": "in stock",
      "StaticValue": "foo"
  }
},
```

Zusätzlich zu der Eingabe können Sie auf ein spezielles JSON-Objekt zugreifen, das als Kontext-Objekt bezeichnet wird. Das Kontext-Objekt enthält Informationen über die Ausführung Ihres Zustandsautomaten. Siehe Context-Objekt.

Verbundene Ressourcen

Das Feld `Parameters` kann auch Daten an verbundene Ressourcen übergeben. Wenn Ihr Aufgabenstatus beispielsweise die Orchestrierung eines AWS Batch Jobs ist, können Sie die entsprechenden API-Parameter direkt an die API-Aktionen dieses Dienstes übergeben. Weitere Informationen finden Sie hier:

- Parameter an eine Service-API übergeben

- Arbeiten mit anderen -Services

Amazon S3

Wenn die Lambda-Funktionsdaten, die Sie zwischen Staaten weitergeben, auf mehr als 262.144 Byte anwachsen könnten, empfehlen wir, Amazon S3 zum Speichern der Daten zu verwenden und eine der folgenden Methoden zu implementieren:

- Verwenden Sie den Status Distributed Map in Ihrem Workflow, damit der `Map` Status Eingaben direkt aus Amazon S3 S3-Datenquellen lesen kann. Weitere Informationen finden Sie unter Verwenden von Map State im verteilten Modus.
- Analysieren Sie den Amazon-Ressourcennamen (ARN) des Buckets im `Payload` Parameter, um den Bucket-Namen und den Schlüsselwert zu erhalten. Weitere Informationen finden Sie unter Verwenden Sie Amazon S3 S3-ARNs, anstatt große Nutzlasten weiterzuleiten.

Alternativ können Sie Ihre Implementierung so anpassen, dass kleinere Nutzlasten in Ihren Ausführungen übergeben werden.

ResultSelector

Verwenden Sie das `ResultSelector` Feld, um das Ergebnis eines Zustands zu bearbeiten, bevor er angewendet `ResultPath` wird. Mit dem `ResultSelector` Feld können Sie eine Sammlung von Schlüsselwertpaaren erstellen, wobei die Werte statisch sind oder aus dem Ergebnis des Bundesstaates ausgewählt werden. Mithilfe des `ResultSelector` Felds können Sie auswählen, welche Teile des Ergebnisses eines Bundesstaates Sie an das `ResultPath` Feld übergeben möchten.

> **Note**
>
> Mit dem `ResultPath` Feld können Sie die Ausgabe des `ResultSelector` Felds zur ursprünglichen Eingabe hinzufügen.

`ResultSelector`ist ein optionales Feld in den folgenden Zuständen:

- Zuordnung
- Status der Aufgabe

- Parallel

Beispielsweise geben Step Functions Functions-Dienstintegrationen zusätzlich zur Nutzlast im Ergebnis Metadaten zurück. `ResultSelector`kann Teile des Ergebnisses auswählen und sie mit der Statuseingabe mit zusammenführen. `ResultPath` In diesem Beispiel möchten wir nur das `resourceType` und auswählen und `ClusterId` es mit der Statuseingabe aus einem Amazon EMR CreateCluster.Sync zusammenführen. Angesichts der folgenden Bedingungen:

```
{
  "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"
  }
}
```

Sie können dann das `resourceType` und auswählen `ClusterId` mit`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"
}
```

Mit der angegebenen Eingabe `ResultSelector` erzeugt die Verwendung von:

```
{
  "OtherDataFromInput": {},
  "EMROutput": {
    "ResourceType": "elasticmapreduce",
    "ClusterId": "AKIAIOSFODNN7EXAMPLE"
  }
}
```

Reduzieren eines Arrays von Arrays

Wenn der Zuordnung Status Parallel oder in Ihren Zustandsmaschinen ein Array von Arrays zurückgibt, können Sie diese mit dem Feld in ein flaches Array umwandeln. ResultSelector Sie können dieses Feld in die Zustandsdefinition Parallel oder Map aufnehmen, um das Ergebnis dieser Zustände zu manipulieren.

Um Arrays zu reduzieren, verwenden Sie die JMESPath-Syntax `[*]` in dem `ResultSelector` Feld, wie im folgenden Beispiel gezeigt.

```
"ResultSelector": {
    "flattenArray.$": "$[*][*]"
  }
```

Beispiele, die zeigen, wie ein Array geglättet wird, finden Sie in Schritt 3 der folgenden Tutorials:

- Verarbeitung des gesamten Datenstapels mit einer Lambda-Funktion
- Verarbeiten einzelner Datenelemente mit einer Lambda-Funktion

ResultPath

Die Ausgabe eines Status kann eine Kopie seiner Eingabe, das entstandene Ergebnis (z. B. Ausgabe der Lambda-Funktion eines `Task`-Status) oder eine Kombination aus Eingabe und Ergebnis sein. Verwenden Sie `ResultPath`, um zu steuern, welche Kombination daraus an die Statusausgabe weitergeleitet wird.

Die folgenden Statustypen können ein Ergebnis generieren und `ResultPath:` einfügen.

- Pass
- Status der Aufgabe
- Parallel
- Zuordnung

Verwenden Sie `ResultPath` zum Kombinieren eines Aufgabenergebnisses mit der Aufgabeneingabe oder wählen Sie eine der folgenden Optionen aus. Der Pfad, den Sie für `ResultPath` angeben, steuert, welche Informationen an die Ausgabe weitergeleitet werden.

> **Note**
>
> `ResultPath` ist auf die Verwendung von Referenzpfaden beschränkt, die den Umfang begrenzen, damit er nur einen einzelnen Knoten in JSON identifizieren kann. Weitere Informationen finden Sie unter Referenzpfade unter Amazon States Language.

Diese Beispiele basieren auf dem im Eine Step Functions Functions-Zustandsmaschine erstellen, die Lambda verwendetTutorial beschriebenen Zustandsautomaten und der Lambda-Funktion. Arbeiten Sie die Anleitung durch und testen Sie verschiedene Ergebnisse anhand verschiedener Pfade in einem `ResultPath`-Feld.

Verwenden Sie ResultPath für Folgendes:

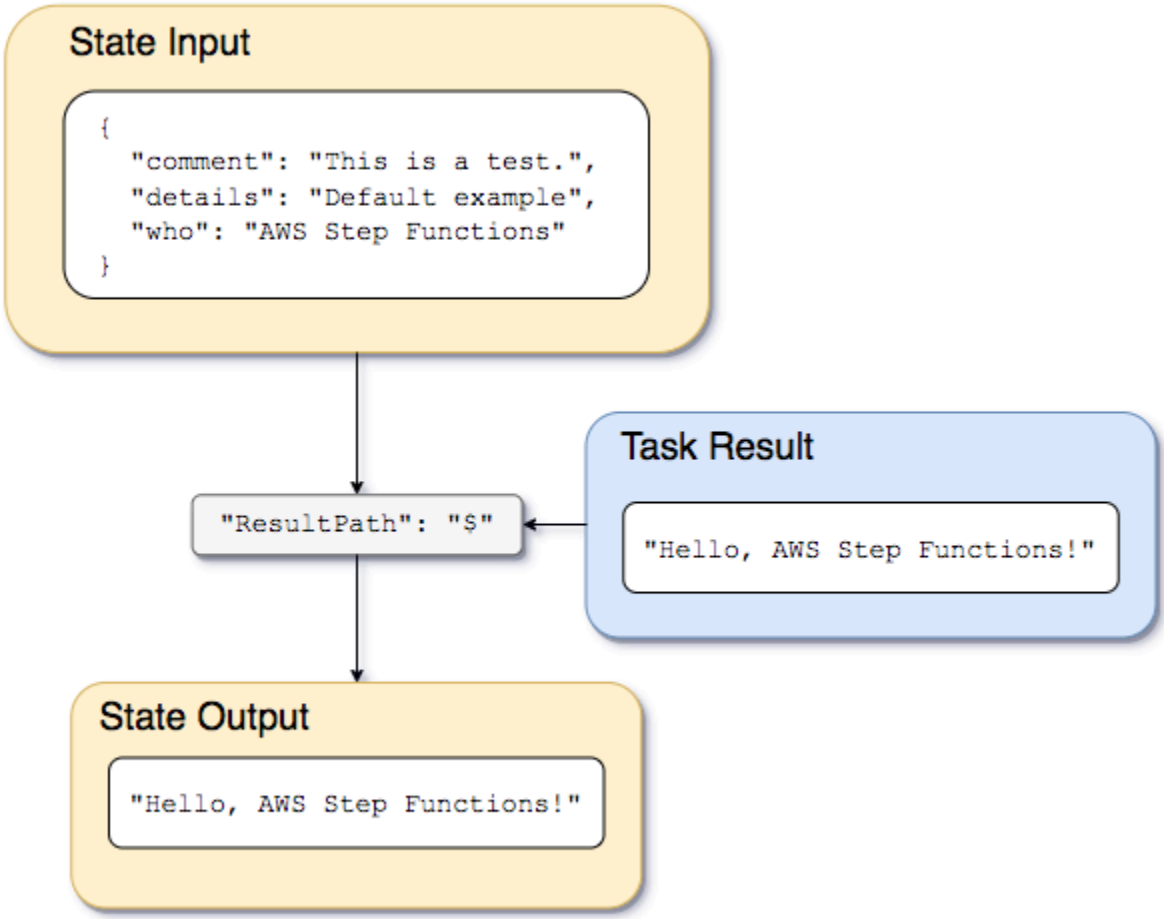
- Verwenden Sie ResultPath , um die Eingabe durch das Ergebnis zu ersetzen
- Verwerfen des Ergebnisses und Beibehalten der ursprünglichen Eingabe
- Verwenden Sie ResultPath , um das Ergebnis in die Eingabe aufzunehmen
- Verwenden Sie ResultPath , um einen Knoten in der Eingabe mit dem Ergebnis zu aktualisieren
- Verwenden Sie ResultPath , um sowohl Fehler als auch Eingabe in einen einzuschließen Catch

> **Tip**
>
> Verwenden Sie den Datenflusssimulator in der Step Functions-Konsole, um die JSON-Pfadsyntax zu testen, besser zu verstehen, wie Daten innerhalb eines -Zustands manipuliert werden, und um zu sehen, wie Daten zwischen -Zuständen übergeben werden.

Verwenden Sie ResultPath , um die Eingabe durch das Ergebnis zu ersetzen

Wenn Sie keinen `ResultPath` angeben, ist das Standardverhalten, als hätten Sie `"ResultPath": "$"` angegeben. Da dies den Status anweist, die gesamte Eingabe durch das Ergebnis zu ersetzen, wird die Statuseingabe vollständig durch das Aufgabenergebnis ersetzt.

Das folgende Diagramm zeigt, wie der `ResultPath` die Eingabe vollständig durch das Ergebnis der Aufgabe ersetzen kann.



Verwenden Sie den Zustandsautomaten und die Lambda-Funktion, die in beschrieben sindEine Step Functions Functions-Zustandsmaschine erstellen, die Lambda verwendet, und ändern Sie den Serviceintegrationstyp in SDKAWS -Integration für die Lambda-Funktion. Geben Sie dazu den Amazon-Ressourcennamen (ARN) der Lambda-Funktion im `Resource` Feld des `Task` Status an, wie im folgenden Beispiel gezeigt. Durch die Verwendung der AWS SDK-Integration wird sichergestellt, dass das `Task` Statusergebnis nur die Ausgabe der Lambda-Funktion ohne Metadaten enthält.

```
{
   "StartAt":"CallFunction",
   "States":{
      "CallFunction": {
         "Type":"Task",
         "Resource":"arn:aws:lambda:us-east-2:123456789012:function:HelloFunction",
         "End": true
      }
   }
}
```

Übergeben Sie dann die folgende Eingabe:

```
{
  "comment": "This is a test of the input and output of a Task state.",
  "details": "Default example",
  "who": "AWS Step Functions"
}
```

Die Lambda-Funktion liefert das folgende Ergebnis.

```
"Hello, AWS Step Functions!"
```

> **Tip**
>
> Sie können dieses Ergebnis in der Step Functions Konsole anzeigen. Wählen Sie dazu auf der Seite Ausführungsdetails der Konsole die Lambda Funktion in der Diagrammansicht aus. Wählen Sie dann im Schrittdetails Bereich die Registerkarte Ausgabe aus, um dieses Ergebnis zu sehen.

Wenn nicht im -Zustand angegeben `ResultPath` ist oder wenn gesetzt `"ResultPath": "$"` ist, wird die Eingabe des -Zustands durch das Ergebnis der Lambda-Funktion ersetzt, und die Ausgabe des -Zustands ist die folgende.

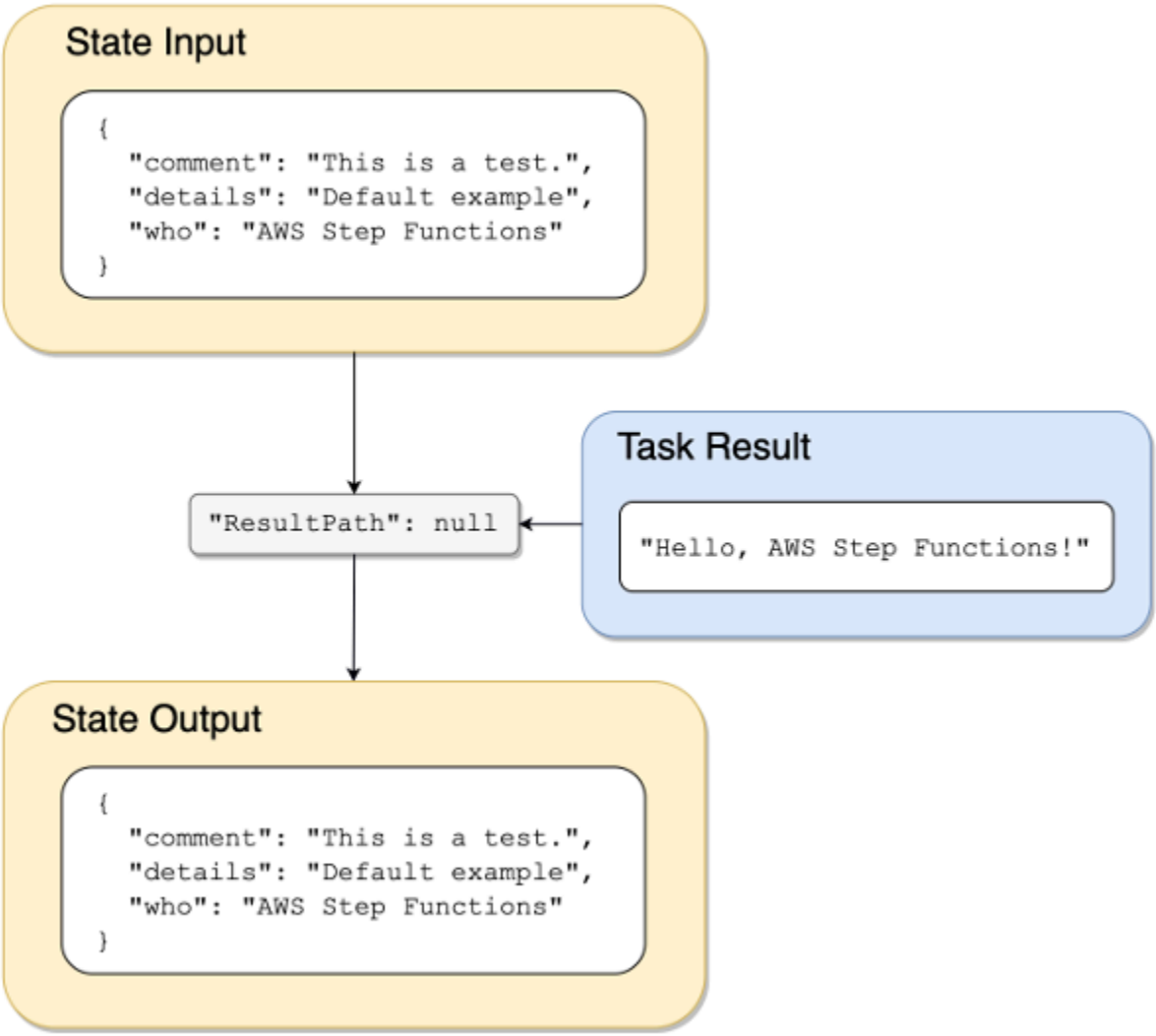
```
"Hello, AWS Step Functions!"
```

> **Note**
>
> `ResultPath` wird verwendet, um Inhalte aus dem Ergebnis in die Eingabe einzufügen, bevor sie an die Ausgabe weitergeleitet wird. Wenn `ResultPath` allerdings nicht angegeben ist, wird standardmäßig die gesamte Eingabe ersetzt.

Verwerfen des Ergebnisses und Beibehalten der urspünglichen Eingabe

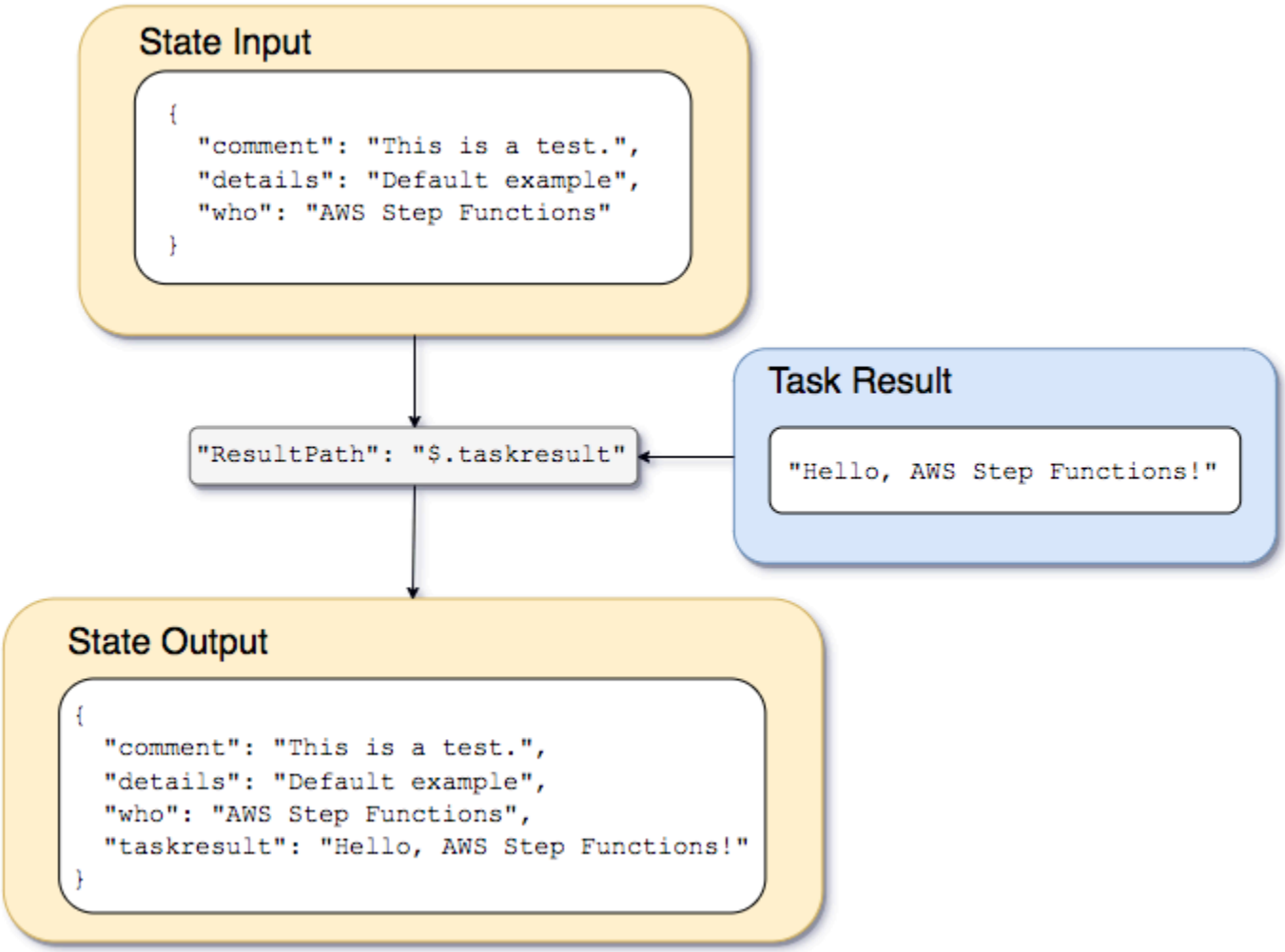
Wenn Sie `ResultPath` auf `null` einstellen, wird die ursprüngliche Eingabe an die Ausgabe übergeben. Mit `"ResultPath": null` wird die Eingabenutzlast des Zustands direkt in die Ausgabe kopiert, ohne Rücksicht auf das Ergebnis.

Das folgende Diagramm zeigt, wie ein `ResultPath` gleich Null die Eingabe direkt in die Ausgabe kopiert.



Verwenden Sie ResultPath , um das Ergebnis in die Eingabe aufzunehmen

Das folgende Diagramm zeigt, wie `ResultPath` das Ergebnis in die Eingabe einfügen kann.



Mit dem im Eine Step Functions Functions-Zustandsmaschine erstellen, die Lambda verwendetTutorial beschriebenen Zustandsautomaten und der Lambda-Funktion könnten wir die folgende Eingabe übergeben.

```
{
  "comment": "This is a test of the input and output of a Task state.",
  "details": "Default example",
  "who": "AWS Step Functions"
}
```

Das Ergebnis der Lambda-Funktion ist das folgende.

```
"Hello, AWS Step Functions!"
```

Um die Eingabe beizubehalten, das Ergebnis der Lambda-Funktion einzufügen und dann den kombinierten JSON-Code in den nächsten Zustand zu übergeben, könnten wir `ResultPath` auf Folgendes setzen.

```
"ResultPath": "$.taskresult"
```

Dies schließt das Ergebnis der Lambda-Funktion mit der ursprünglichen Eingabe ein.

```
{
  "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!"
}
```

Die Ausgabe der Lambda-Funktion wird an die ursprüngliche Eingabe als Wert für angehängt`taskresult`. Die Eingabe, einschließlich des neu eingefügten Werts, wird an den nächsten Status weitergeleitet.

Sie können das Ergebnis auch in einen untergeordneten Knoten der Eingabe einfügen. Legen Sie für `ResultPath` Folgendes fest.

```
"ResultPath": "$.strings.lambdaresult"
```

Starten einer Ausführung mit der folgenden Eingabe.

```
{
  "comment": "An input comment.",
  "strings": {
    "string1": "foo",
    "string2": "bar",
    "string3": "baz"
  },
  "who": "AWS Step Functions"
}
```

Das Ergebnis der Lambda-Funktion wird als untergeordnetes Element des `strings` Knotens in die Eingabe eingefügt.

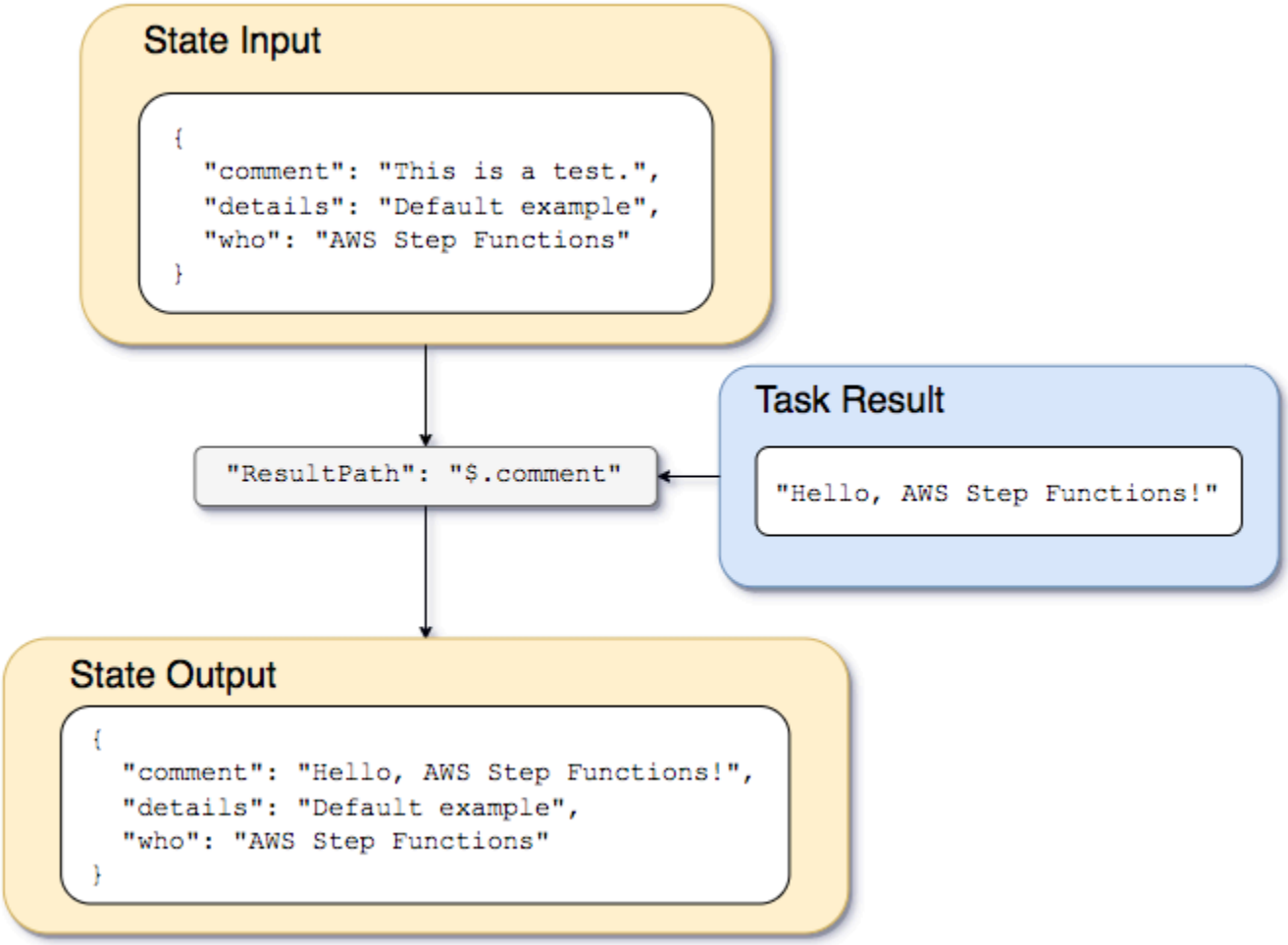
```
{
```

```
  "comment": "An input comment.",
  "strings": {
    "string1": "foo",
    "string2": "bar",
    "string3": "baz",
    "lambdaresult": "Hello, AWS Step Functions!"
  },
  "who": "AWS Step Functions"
}
```

Die Zustandsausgabe enthält jetzt die ursprüngliche JSON-Ausgabe mit dem Ergebnis als untergeordneten Knoten.

Verwenden Sie ResultPath , um einen Knoten in der Eingabe mit dem Ergebnis zu aktualisieren

Das folgende Diagramm zeigt, wie `ResultPath` den Wert der vorhandenen JSON-Knoten in der Eingabe mit Werten aus dem Aufgabenergebnis aktualisieren kann.



Anhand des Beispiels des Zustandsautomaten und der Lambda-Funktion, die im Eine Step Functions Functions-Zustandsmaschine erstellen, die Lambda verwendetTutorial beschrieben werden, könnten wir die folgende Eingabe übergeben.

```
{
  "comment": "This is a test of the input and output of a Task state.",
  "details": "Default example",
  "who": "AWS Step Functions"
}
```

Das Ergebnis der Lambda-Funktion ist das folgende.

```
Hello, AWS Step Functions!
```

Statt die Eingabe beizubehalten und das Ergebnis als neuen Knoten in JSON einzufügen, können wir einen vorhandenen Knoten überschreiben.

Beispiel: Wie beim Auslassen oder Festlegen von `"ResultPath": "$"` wird der gesamte Knoten überschrieben. Sie können einen einzelnen Knoten festlegen, der mit dem Ergebnis überschrieben wird.

```
"ResultPath": "$.comment"
```

Da der `comment` Knoten bereits in der Zustandseingabe vorhanden ist, `"$.comment"` ersetzt die Einstellung `ResultPath` auf diesen Knoten in der Eingabe durch das Ergebnis der Lambda-Funktion. Ohne weiteres Filtern nach `OutputPath` wird Folgendes an die Ausgabe weitergeleitet.

```
{
  "comment": "Hello, AWS Step Functions!",
  "details": "Default behavior example",
  "who": "AWS Step Functions",
}
```

Der Wert für den `comment` Knoten, `"This is a test of the input and output of a Task state."`, wird durch das Ergebnis der Lambda-Funktion ersetzt: `"Hello, AWS Step Functions!"` in der Zustandsausgabe.

Verwenden Sie ResultPath , um sowohl Fehler als auch Eingabe in einen einzuschließen **`Catch`**

In der Behandlung von Fehlerbedingungen mithilfe einer Step Functions Functions-Zustandsmaschine-Anleitung erfahren Sie, wie Sie einen Zustandsautomaten verwenden, um einen Fehler abzufangen. In einigen Fällen können Sie die ursprüngliche Eingabe mit dem Fehler beibehalten. Verwenden Sie `ResultPath` in einem `Catch`, um den Fehler in die ursprüngliche Eingabe aufzunehmen, anstatt ihn zu ersetzen.

```
"Catch": [{
  "ErrorEquals": ["States.ALL"],
  "Next": "NextTask",
  "ResultPath": "$.error"
}]
```

Wenn die vorherige `Catch`-Anweisung einen Fehler abfängt, wird das Ergebnis in einem `error`-Knoten innerhalb der Statuseingabe eingefügt. Nutzen Sie dazu beispielsweise folgende Eingabe:

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

Die Statusausgabe beim Abfangen eines Fehlers lautet wie folgt:

```
{
  "foo": "bar",
  "error": {
    "Error": "Error here"
  }
}
```

Weitere Informationen zur Fehlerbehandlung finden Sie im Folgenden:

- Fehlerbehandlung in Step Functions
- Behandlung von Fehlerbedingungen mithilfe einer Step Functions Functions-Zustandsmaschine

OutputPath

`OutputPath` ermöglicht Ihnen, einen Teil der Zustandsausgabe auszuwählen, die an den nächsten Zustand übergeben wird. Auf diese Weise können Sie unerwünschte Informationen herausfiltern und nur den gewünschten Teil von JSON übergeben.

Wenn Sie kein `OutputPath` angeben, ist der Standardwert $. Dieser übergibt den gesamten JSON-Knoten (bestimmt durch die Zustandseingabe, das Aufgabenergebnis und `ResultPath`) an den nächsten Zustand.

> **Tip**
>
> Verwenden Sie den Datenflusssimulator in der Step Functions Functions-Konsole, um die JSON-Pfadsyntax zu testen, um besser zu verstehen, wie Daten innerhalb eines Zustands manipuliert werden, und um zu sehen, wie Daten zwischen Staaten weitergegeben werden.

Weitere Informationen finden Sie hier:

- Pfade in der Sprache der Amazonasstaaten
- InputPath,ResultPath, und OutputPath Beispiele
- Übergeben Sie statisches JSON als Parameter

- Eingabe- und Ausgabeverarbeitung in Step Functions

InputPath,ResultPath, und OutputPath Beispiele

Jeder andere Staat als ein Fehler Bundesstaat oder ein Succeed Bundesstaat kann die Eingabe- und Ausgabeverarbeitungsfelder wie `InputPath``ResultPath`, oder enthalten`OutputPath`. Darüber hinaus unterstützen Wait die Choice Land-Staaten das `ResultPath` Feld nicht. Mit diesen Feldern können Sie a verwenden, JsonPathum die JSON-Daten zu filtern, während sie sich in Ihrem Workflow bewegen.

Sie können das `Parameters` Feld auch verwenden, um die JSON-Daten zu manipulieren, während sie sich in Ihrem Workflow bewegen. Für weitere Informationen zur Nutzung von `Parameters` siehe InputPath, Parameter und ResultSelector.

Beispiel: Beginnen Sie mit der AWS Lambda-Funktion und dem Zustandsautomaten, die in der Eine Step Functions Functions-Zustandsmaschine erstellen, die Lambda verwendet-Anleitung beschrieben sind. Ändern Sie den Zustandsautomaten so, dass er `InputPath`, `ResultPath` und `OutputPath` enthält.

```
{
  "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
    }
  }
}
```

Starten einer Ausführung mit der folgenden Eingabe.

```
{
  "comment": "An input comment.",
  "data": {
    "val1": 23,
```

```
    "val2": 17
  },
  "extra": "foo",
  "lambda": {
    "who": "AWS Step Functions"
  }
}
```

Gehen wir davon aus, dass die `extra` Knoten `comment` und verworfen werden können, wir aber die Ausgabe der Lambda-Funktion einbeziehen und die Informationen im Knoten beibehalten möchten. `data`

Im aktualisierten Zustandsautomaten wird der `Task`-Zustand geändert, damit die Aufgabeneingabe verarbeitet wird.

```
"InputPath": "$.lambda",
```

Diese Zeile in der Definition des Zustandsautomaten begrenzt die Aufgabeneingabe ausschließlich auf `lambda`-Knoten aus der Statuseingabe. Die Lambda-Funktion empfängt nur das JSON-Objekt `{"who": "AWS Step Functions"}` als Eingabe.

```
"ResultPath": "$.data.lambdaresult",
```

Dies `ResultPath` weist die State-Maschine an, das Ergebnis der Lambda-Funktion in einen Knoten mit dem Namen einzufügen`lambdaresult`, und zwar als untergeordnetes Element des `data` Knotens in der ursprünglichen State-Machine-Eingabe. Da wir keine weiteren Manipulationen an der ursprünglichen Eingabe vornehmen und das Ergebnis verwenden`OutputPath`, enthält die Ausgabe des Status jetzt das Ergebnis der Lambda-Funktion mit der ursprünglichen Eingabe.

```
{
  "comment": "An input comment.",
  "data": {
    "val1": 23,
    "val2": 17,
    "lambdaresult": "Hello, AWS Step Functions!"
  },
  "extra": "foo",
  "lambda": {
    "who": "AWS Step Functions"
  }
```

```
}
```

Unser Ziel war es jedoch, nur den `data` Knoten beizubehalten und das Ergebnis der Lambda-Funktion einzubeziehen. `OutputPath`filtert dieses kombinierte JSON, bevor es an die State-Ausgabe übergeben wird.

```
"OutputPath": "$.data",
```

Dadurch wird nur der `data`-Knoten aus der ursprünglichen Eingabe (einschließlich des untergeordneten `lambdaresult`-Elements, das von `ResultPath` eingefügt wurde) für die Weiterleitung an die Ausgabe ausgewählt. Die Statusausgabe wird auf Folgende Weise gefiltert:

```
{
  "val1": 23,
  "val2": 17,
  "lambdaresult": "Hello, AWS Step Functions!"
}
```

In diesem `Task`-Status:

1. `InputPath`sendet nur den `lambda` Knoten von der Eingabe an die Lambda-Funktion.
2. `ResultPath` fügt das Ergebnis als untergeordnetes Element des `data`-Knotens in die ursprünglichen Eingabe ein.
3. `OutputPath`filtert die Zustandseingabe (die jetzt das Ergebnis der Lambda-Funktion enthält), sodass nur der `data` Knoten an die Statusausgabe weitergegeben wird.

Example um Maschineneingabe, Ergebnis und Endausgabe im Originalzustand zu manipulieren, indem JsonPath

Stellen Sie sich die folgende staatliche Maschine vor, die die Identität und Adresse eines Versicherungsantragstellers überprüft.

> **Note**
>
> Das vollständige Beispiel finden Sie unter How to use JSON Path in Step Functions.

```
{
```

```
  "Comment": "Sample state machine to verify an applicant's ID and address",
  "StartAt": "Verify info",
  "States": {
    "Verify info": {
      "Type": "Parallel",
      "End": true,
      "Branches": [
        {
          "StartAt": "Verify identity",
          "States": {
            "Verify identity": {
              "Type": "Task",
              "Resource": "arn:aws:states:::lambda:invoke",
              "Parameters": {
                "Payload.$": "$",
                "FunctionName": "arn:aws:lambda:us-east-2:111122223333:function:check-
identity:$LATEST"
              },
              "End": true
            }
          }
        },
        {
          "StartAt": "Verify address",
          "States": {
            "Verify address": {
              "Type": "Task",
              "Resource": "arn:aws:states:::lambda:invoke",
              "Parameters": {
                "Payload.$": "$",
                "FunctionName": "arn:aws:lambda:us-east-2:111122223333:function:check-
address:$LATEST"
              },
              "End": true
            }
          }
        }
      ]
    }
  }
}
```

Wenn Sie diese State-Maschine mit der folgenden Eingabe ausführen, schlägt die Ausführung fehl, da die Lambda-Funktionen, die die Überprüfung durchführen, nur die Daten erwarten, die als Eingabe verifiziert werden müssen. Daher müssen Sie die Knoten angeben, die die zu überprüfenden Informationen enthalten, indem Sie einen entsprechenden Knoten verwendenJsonPath.

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

Um den Knoten anzugeben, den die *`check-identity`* Lambda-Funktion verwenden muss, verwenden Sie das `InputPath` Feld wie folgt:

```
"InputPath": "$.data.identity"
```

Und um den Knoten anzugeben, den die *check-address* Lambda-Funktion verwenden muss, verwenden Sie das `InputPath` Feld wie folgt:

```
"InputPath": "$.data.address"
```

Wenn Sie nun das Überprüfungsergebnis in der Maschineneingabe im Originalzustand speichern möchten, verwenden Sie das `ResultPath` Feld wie folgt:

```
"ResultPath": "$.results"
```

Wenn Sie jedoch nur die Identitäts- und Überprüfungsergebnisse benötigen und die ursprüngliche Eingabe verwerfen, verwenden Sie das `OutputPath` Feld wie folgt:

```
"OutputPath": "$.results"
```

Weitere Informationen finden Sie unter Eingabe- und Ausgabeverarbeitung in Step Functions.

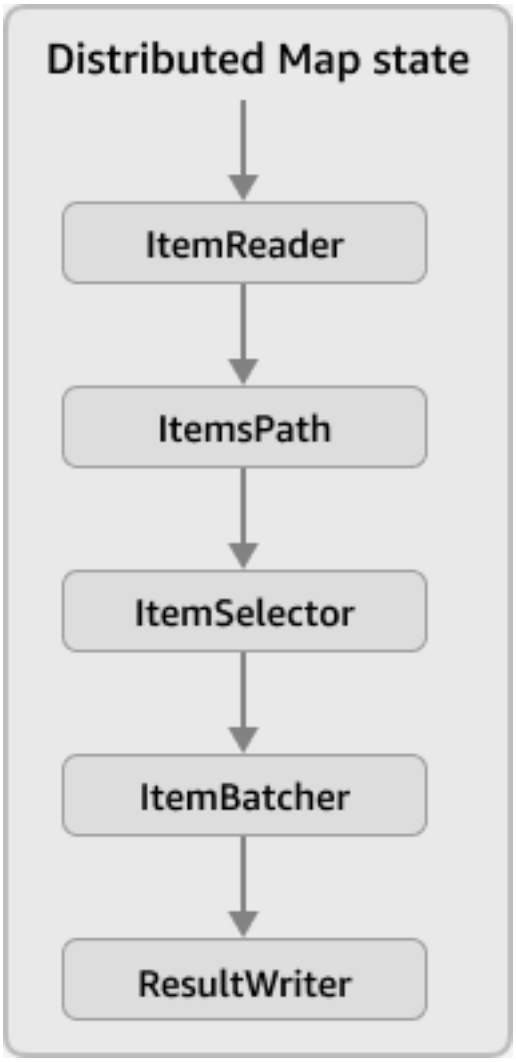
Eingabe- und Ausgabefelder des Zustands zuordnen

`Map`Zustände iterieren gleichzeitig über eine Sammlung von Elementen in einem Datensatz, z. B. ein JSON-Array, eine Liste von Amazon S3-Objekten oder die Zeilen einer CSV-Datei in einem Amazon S3-Bucket. Es wiederholt eine Reihe von Schritten für jedes Element in der Sammlung. Mithilfe dieser Felder können Sie die Eingabe, die der `Map` Staat empfängt, und die von ihm generierte Ausgabe konfigurieren. Step Functions wendet jedes Feld in Ihrem Distributed-Map-Status in der Reihenfolge an, die in der folgenden Liste und Abbildung dargestellt ist:

> **Note**
>
> Je nach Anwendungsfall müssen Sie möglicherweise nicht alle diese Felder anwenden.

1. ItemReader
2. ItemsPath
3. ItemSelector
4. ItemBatcher

5. ResultWriter



> **Note**
>
> Diese `Map` Status-Eingabe- und -Ausgabefelder sind derzeit im Datenflusssimulator in der Step Functions-Konsole nicht verfügbar.

ItemReader

Der`ItemReader`Feld ist ein JSON-Objekt, das einen Datensatz und seinen Speicherort angibt. EINStatus der verteilten Karteverwendet diesen Datensatz als Eingabe. Das folgende Beispiel zeigt die Syntax von`ItemReader`Feld, wenn Ihr Datensatz eine CSV-Datei ist, die in einem Amazon S3 S3-Bucket gespeichert ist.

```
"ItemReader": {
    "ReaderConfig": {
        "InputType": "CSV",
        "CSVHeaderLocation": "FIRST_ROW"
    },
    "Resource": "arn:aws:states:::s3:getObject",
    "Parameters": {
        "Bucket": "myBucket",
        "Key": "csvDataset/ratings.csv"
    }
}
```

> **Tip**
>
> In Workflow Studio geben Sie den Datensatz und seinen Speicherort imQuelle des ArtikelsFeld.

Inhalt

Inhalt des ItemReader Feld

Abhängig von Ihrem Datensatz ist der Inhalt des`ItemReader`Feld variiert. Wenn es sich bei Ihrem Datensatz beispielsweise um ein JSON-Array handelt, das aus einem vorherigen Schritt im Workflow übergeben wurde,`ItemReader`Feld wird weggelassen. Wenn es sich bei Ihrem Datensatz um eine Amazon S3 S3-Datenquelle handelt, enthält dieses Feld die folgenden Unterfelder.

`ReaderConfig`

Ein JSON-Objekt, das die folgenden Details spezifiziert:

- `InputType`

  Gibt den Typ der Amazon S3 S3-Datenquelle an, z. B. CSV-Datei, Objekt, JSON-Datei oder eine Amazon S3 S3-Inventarliste. In Workflow Studio können Sie einen Eingabetyp aus derAmazon S3 S3-ArtikelquelleDrop-down-Liste unterQuelle des ArtikelsFeld.

- `CSVHeaderLocation`

> **Note**
>
> Sie müssen dieses Feld nur angeben, wenn Sie eine CSV-Datei als Datensatz verwenden.

Akzeptiert einen der folgenden Werte, um die Position der Spaltenüberschrift anzugeben:

> **Important**
>
> Derzeit unterstützt Step Functions CSV-Header mit bis zu 10 KB.

- `FIRST_ROW`— Verwenden Sie diese Option, wenn die erste Zeile der Datei der Header ist.
- `GIVEN`— Verwenden Sie diese Option, um den Header innerhalb der State-Machine-Definition anzugeben. Wenn Ihre CSV-Datei beispielsweise die folgenden Daten enthält.

```
1,307,3.5,1256677221
1,481,3.5,1256677456
1,1091,1.5,1256677471
...
```

Stellen Sie das folgende JSON-Array als CSV-Header bereit.

```
"ItemReader": {
    "ReaderConfig": {
        "InputType": "CSV",
        "CSVHeaderLocation": "GIVEN",
        "CSVHeaders": [
            "userId",
            "movieId",
            "rating",
            "timestamp"
        ]
    }
}
```

> **Tip**
>
> In Workflow Studio finden Sie diese Option unterZusätzliche Konfigurationin derQuelle des ArtikelsFeld.

- `MaxItems`

  Beschränkt die Anzahl der übergebenen DatenelementeMapBundesstaat. Nehmen wir beispielsweise an, Sie stellen eine CSV-Datei bereit, die 1000 Zeilen enthält, und geben einen Grenzwert von 100 an. Dann hat der Interpreter den Test bestandennur100 Zeilen bis zumMapBundesstaat. DerMapstate verarbeitet Elemente in sequentieller Reihenfolge, beginnend nach der Kopfzeile.

  Standardmäßig ist derMapstate iteriert über alle Elemente im angegebenen Datensatz.

  > **Note**
  >
  > Derzeit können Sie einen Grenzwert von bis zu 100.000.000 angeben. DerStatus der verteilten Kartebeendet das Lesen von Elementen, die diesen Grenzwert überschreiten.

  > **Tip**
  >
  > In Workflow Studio finden Sie diese Option unterZusätzliche Konfigurationin derQuelle des ArtikelsFeld.

  Sie können auch eine angebenReferenzpfadzu einem vorhandenen Schlüssel-Wert-Paar in IhremStatus der verteilten KarteEingabe. Dieser Pfad muss in eine positive Ganzzahl aufgelöst werden. Sie geben den Referenzpfad in der`MaxItemsPath`Unterfeld.

  > **Important**
  >
  > Sie können entweder`MaxItems`oder das`MaxItemsPath`Unterfeld, aber nicht beides.

`Resource`

Die Amazon S3 S3-API-Aktion, die Step Functions aufrufen muss, hängt vom angegebenen Datensatz ab.

`Parameters`

Ein JSON-Objekt, das den Amazon S3 S3-Bucket-Namen und den Objektschlüssel angibt, in dem der Datensatz gespeichert ist.

> **Important**
>
> Stellen Sie sicher, dass sich Ihre Amazon S3 S3-Buckets unter demselben befindenAWS-KontoundAWS-Regionals Ihre Staatsmaschine.

Beispiele für Datensätze

Sie können eine der folgenden Optionen als Datensatz angeben:

- JSON-Array aus einem vorherigen Schritt
- Eine Liste von Amazon S3 S3-Objekten
- JSON-Datei in einem Amazon S3 S3-Bucket
- CSV-Datei in einem Amazon S3 S3-Bucket
- Amazon S3-Bestandsliste

> **Important**
>
> Step Functions benötigt die entsprechenden Berechtigungen, um auf die Amazon S3 S3-Datensätze zuzugreifen, die Sie verwenden. Informationen über IAM-Richtlinien für die Datensätze finden Sie unterIAM-Richtlinien für Datensätze.

JSON-Array aus einem vorherigen Schritt

EINStatus der verteilten Kartekann JSON-Eingaben akzeptieren, die von einem vorherigen Schritt im Workflow übergeben wurden. Diese Eingabe muss entweder ein Array sein oder ein Array innerhalb

eines bestimmten Knotens enthalten. Um einen Knoten auszuwählen, der das Array enthält, können Sie den`ItemsPath`Feld.

Um einzelne Elemente im Array zu verarbeiten,Status der verteilten Kartestartet eine untergeordnete Workflow-Ausführung für jedes Array-Element. Die folgenden Registerkarten zeigen Beispiele für die Eingabe, die an die übergeben wurde`Map`Status und die entsprechende Eingabe für die Ausführung eines untergeordneten Workflows.

> **Note**
>
> Step Functions lässt das aus`ItemReader`Feld, wenn es sich bei Ihrem Datensatz um ein JSON-Array aus einem vorherigen Schritt handelt.

Input passed to the Map state

Betrachten Sie das folgende JSON-Array mit drei Elementen.

```
"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

DasStatus der verteilten Kartestartet drei untergeordnete Workflow-Ausführungen. Jede Ausführung erhält ein Array-Element als Eingabe. Das folgende Beispiel zeigt die Eingabe, die von einer untergeordneten Workflow-Ausführung empfangen wurde.

```
{
  "verdict": "true",
  "statement_date": "6/11/2008",
  "statement_source": "speech"
}
```

Beispiel Amazon S3 S3-Objekte

EINStatus der verteilten Kartekann über die Objekte iterieren, die in einem Amazon S3 S3-Bucket gespeichert sind. Wenn die Workflow-Ausführung den`Map`state, Step Functions ruft dieListObjectsV2API-Aktion, die ein Array der Amazon S3 S3-Objektmetadaten zurückgibt. In diesem Array enthält jedes Element Daten, wieETagundSchlüssel, für die im Bucket gespeicherten Daten.

Um einzelne Elemente im Array zu verarbeiten,Status der verteilten Kartestartet die Ausführung eines untergeordneten Workflows. Nehmen Sie beispielsweise an, dass Ihr Amazon S3 S3-Bucket 100 Bilder enthält. Dann kehrte das Array nach dem Aufrufen von zurück`ListObjectsV2`Die API-Aktion enthält 100 Elemente. DieStatus der verteilten Kartestartet dann 100 untergeordnete Workflow-Ausführungen, um jedes Array-Element zu verarbeiten.

> **Note**
>
> - Derzeit enthält Step Functions auch einen Eintrag für jeden Ordner, den Sie in einem bestimmten Amazon S3 S3-Bucket mithilfe der Amazon S3 S3-Konsole erstellen. Dies führt zu einer zusätzlichen untergeordneten Workflow-Ausführung, die vonStatus der verteilten Karte. Um zu vermeiden, dass für den Ordner eine zusätzliche untergeordnete Workflow-Ausführung erforderlich ist, empfehlen wir die Verwendung desAWS CLIum Ordner zu erstellen. Weitere Informationen finden Sie unterAmazon S3 S3-Befehle auf hoher Ebenein derAWS Command Line InterfaceBenutzerleitfaden.
> - Step Functions benötigt die entsprechenden Berechtigungen, um auf die Amazon S3 S3-Datensätze zuzugreifen, die Sie verwenden. Informationen über IAM-Richtlinien für die Datensätze finden Sie unterIAM-Richtlinien für Datensätze.

Die folgenden Tabs zeigen Beispiele für`ItemReader`Feldsyntax und die Eingabe, die an die Ausführung eines untergeordneten Workflows für diesen Datensatz übergeben wurde.

ItemReader syntax

In diesem Beispiel haben Sie Ihre Daten, zu denen Bilder, JSON-Dateien und Objekte gehören, in einem Präfix mit dem Namen organisiert`processData`in einem Amazon S3 S3-Bucket mit dem Namen`myBucket`.

```
"ItemReader": {
    "Resource": "arn:aws:states:::s3:listObjectsV2",
    "Parameters": {
        "Bucket": "myBucket",
        "Prefix": "processData"
    }
}
```

Input passed to a child workflow execution

DasStatus der verteilten Kartestartet so viele untergeordnete Workflow-Ausführungen wie die Anzahl der im Amazon S3 S3-Bucket vorhandenen Elemente. Das folgende Beispiel zeigt die Eingabe, die von einer untergeordneten Workflow-Ausführung empfangen wurde.

```
{
  "Etag": "\"05704fbdccb224cb01c59005bebbad28\"",
  "Key": "processData/images/n02085620_1073.jpg",
  "LastModified": 1668699881,
  "Size": 34910,
  "StorageClass": "STANDARD"
}
```

JSON-Datei in einem Amazon S3 S3-Bucket

EINStatus der verteilten Kartekann eine JSON-Datei, die in einem Amazon S3 S3-Bucket gespeichert ist, als Datensatz akzeptieren. Die JSON-Datei muss ein Array enthalten.

Wenn die Workflow-Ausführung den`Map`state, Step Functions ruft die[GetObject](#)API-Aktion zum Abrufen der angegebenen JSON-Datei. Die`Map`Der Status iteriert dann über jedes Element im Array und startet für jedes Element eine untergeordnete Workflow-Ausführung. Wenn Ihre JSON-Datei beispielsweise 1000 Array-Elemente enthält,`Map`state startet 1000 untergeordnete Workflow-Ausführungen.

> **Note**
>
> - Die Ausführungseingabe, die verwendet wird, um die Ausführung eines untergeordneten Workflows zu starten, darf 256 KB nicht überschreiten. Step Functions unterstützt jedoch das Lesen eines Elements mit einer Größe von bis zu 8 MB aus einer CSV- oder JSON-Datei, wenn Sie dann das optionale`ItemSelector`Feld, um die Größe des Elements zu reduzieren.
> - Derzeit unterstützt Step Functions 10 GB als maximale Größe einer einzelnen Datei in einem Amazon S3 S3-Inventarbericht. Step Functions kann jedoch mehr als 10 GB verarbeiten, wenn jede einzelne Datei weniger als 10 GB groß ist.
> - Step Functions benötigt die entsprechenden Berechtigungen, um auf die Amazon S3 S3-Datensätze zuzugreifen, die Sie verwenden. Informationen über IAM-Richtlinien für die Datensätze finden Sie unterIAM-Richtlinien für Datensätze.

Die folgenden Tabs zeigen Beispiele für`ItemReader`Feldsyntax und die Eingabe, die an die Ausführung eines untergeordneten Workflows für diesen Datensatz übergeben wurde.

Stellen Sie sich für dieses Beispiel vor, Sie haben eine JSON-Datei mit dem Namen*`factcheck.json`*. Sie haben diese Datei in einem Präfix mit dem Namen gespeichert*`jsonDataset`*in einem Amazon S3 S3-Bucket. Das Folgende ist ein Beispiel für den JSON-Datensatz.

```
[
  {
    "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

DerStatus der verteilten Kartestartet so viele untergeordnete Workflow-Ausführungen wie die Anzahl der in der JSON-Datei vorhandenen Array-Elemente. Das folgende Beispiel zeigt die Eingabe, die von einer untergeordneten Workflow-Ausführung empfangen wurde.

```
{
  "verdict": "true",
  "statement_date": "6/11/2008",
  "statement_source": "speech"
}
```

CSV-Datei in einem Amazon S3 S3-Bucket

EINStatus der verteilten Kartekann eine CSV-Datei, die in einem Amazon S3 S3-Bucket gespeichert ist, als Datensatz akzeptieren. Wenn Sie eine CSV-Datei als Datensatz verwenden, müssen Sie eine CSV-Spaltenüberschrift angeben. Informationen zum Angeben eines CSV-Headers finden Sie unterInhalt des ItemReader Feld.

Da es kein standardisiertes Format zum Erstellen und Verwalten von Daten in CSV-Dateien gibt, analysiert Step Functions CSV-Dateien auf der Grundlage der folgenden Regeln:

- Kommas (,) sind ein Trennzeichen, das einzelne Felder voneinander trennt.
- Zeilenumbrüche sind ein Trennzeichen, das einzelne Datensätze voneinander trennt.

- Felder werden als Zeichenketten behandelt. Verwenden Sie für Datentypkonvertierungen den`States.StringToJson`intrinsische Funktion in`ItemSelector`.
- Um Zeichenketten einzuschließen, sind keine doppelten Anführungszeichen (“ „) erforderlich. Zeichenketten, die in doppelte Anführungszeichen eingeschlossen sind, können jedoch Kommas und Zeilenumbrüche enthalten, ohne dass sie als Trennzeichen dienen.
- Vermeiden Sie doppelte Anführungszeichen, indem Sie sie wiederholen.
- Wenn die Anzahl der Felder in einer Zeile geringer ist als die Anzahl der Felder in der Kopfzeile, stellt Step Functions leere Zeichenketten für die fehlenden Werte bereit.
- Wenn die Anzahl der Felder in einer Zeile größer ist als die Anzahl der Felder in der Kopfzeile, überspringt Step Functions die zusätzlichen Felder.

Weitere Informationen darüber, wie Step Functions eine CSV-Datei analysiert, finden Sie unterExample of parsing an input CSV file.

Wenn die Workflow-Ausführung den`Map`state, Step Functions ruft die`GetObject`API-Aktion zum Abrufen der angegebenen CSV-Datei. Die`Map`Der Status iteriert dann über jede Zeile in der CSV-Datei und startet eine untergeordnete Workflow-Ausführung, um die Elemente in jeder Zeile zu verarbeiten. Angenommen, Sie stellen eine CSV-Datei bereit, die 100 Zeilen als Eingabe enthält. Dann übergibt der Interpreter jede Zeile an`Map`Bundesstaat. Der`Map`state verarbeitet Elemente in serieller Reihenfolge, beginnend nach der Kopfzeile.

> **Note**
>
> - Die Ausführungseingabe, die verwendet wird, um die Ausführung eines untergeordneten Workflows zu starten, darf 256 KB nicht überschreiten. Step Functions unterstützt jedoch das Lesen eines Elements mit einer Größe von bis zu 8 MB aus einer CSV- oder JSON-Datei, wenn Sie dann das optionale`ItemSelector`Feld, um die Größe des Elements zu reduzieren.
> - Derzeit unterstützt Step Functions 10 GB als maximale Größe einer einzelnen Datei in einem Amazon S3 S3-Inventarbericht. Step Functions kann jedoch mehr als 10 GB verarbeiten, wenn jede einzelne Datei weniger als 10 GB groß ist.
> - Step Functions benötigt die entsprechenden Berechtigungen, um auf die Amazon S3 S3-Datensätze zuzugreifen, die Sie verwenden. Informationen über IAM-Richtlinien für die Datensätze finden Sie unterIAM-Richtlinien für Datensätze.

Die folgenden Tabs zeigen Beispiele für`ItemReader`Feldsyntax und die Eingabe, die an die Ausführung eines untergeordneten Workflows für diesen Datensatz übergeben wurde.

ItemReader syntax

Angenommen, Sie haben über eine CSV-Datei mit dem Namen*`ratings.csv`*. Dann haben Sie diese Datei in einem Präfix gespeichert, das benannt ist*`csvDataset`*in einem Amazon S3 S3-Bucket.

```
{
  "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

DieStatus der verteilten Kartestartet so viele untergeordnete Workflow-Ausführungen wie die Anzahl der Zeilen, die in der CSV-Datei vorhanden sind, mit Ausnahme der Kopfzeile, falls in der Datei vorhanden. Das folgende Beispiel zeigt die Eingabe, die von einer untergeordneten Workflow-Ausführung empfangen wurde.

```
{
  "rating": "3.5",
  "movieId": "307",
  "userId": "1",
  "timestamp": "1256677221"
}
```

Beispiel für ein S3-Inventar

EINStatus der verteilten Kartekann eine Amazon S3 S3-Inventarmanifestdatei, die in einem Amazon S3 S3-Bucket gespeichert ist, als Datensatz akzeptieren.

Wenn die Workflow-Ausführung den`Map`state, Step Functions ruft dieGetObjectAPI-Aktion zum Abrufen der angegebenen Amazon S3 S3-Inventarmanifestdatei. Die`Map`state iteriert dann über die Objekte im Inventar, um ein Array von Amazon S3 S3-Inventarobjekt-Metadaten zurückzugeben.

> **Note**
>
> - Derzeit unterstützt Step Functions 10 GB als maximale Größe einer einzelnen Datei in einem Amazon S3 S3-Inventarbericht. Step Functions kann jedoch mehr als 10 GB verarbeiten, wenn jede einzelne Datei weniger als 10 GB groß ist.
> - Step Functions benötigt die entsprechenden Berechtigungen, um auf die Amazon S3 S3-Datensätze zuzugreifen, die Sie verwenden. Informationen über IAM-Richtlinien für die Datensätze finden Sie unterIAM-Richtlinien für Datensätze.

Das Folgende ist ein Beispiel für eine Inventardatei im CSV-Format. Diese Datei enthält die benannten Objekte`csvDataset`und`imageDataset`, die in einem Amazon S3 S3-Bucket mit dem Namen gespeichert sind`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**
>
> Derzeit unterstützt Step Functions keinen benutzerdefinierten Amazon S3 S3-Inventarbericht als Datensatz. Sie müssen auch sicherstellen, dass das Ausgabeformat Ihres Amazon S3 S3-Inventarberichts CSV ist. Weitere Informationen zu Amazon S3 S3-Inventaren und deren Einrichtung finden Sie unterAmazon S3 Inventoryin derAmazon-S3-Benutzerhandbuch.

Das folgende Beispiel für eine Inventar-Manifestdatei zeigt die CSV-Header für die Metadaten des Inventarobjekts.

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

Die folgenden Registerkarten zeigen Beispiele für`ItemReader`Feldsyntax und die Eingabe, die an die Ausführung eines untergeordneten Workflows für diesen Datensatz übergeben wurde.

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"
}
```

Abhängig von den Feldern, die Sie bei der Konfiguration des Amazon S3-Inventarberichts ausgewählt haben, ist der Inhalt Ihres`manifest.json`Die Datei kann vom gezeigten Beispiel abweichen.

IAM-Richtlinien für Datensätze

Wenn Sie Workflows mit der Step Functions-Konsole erstellen, kann Step Functions automatisch IAM-Richtlinien auf der Grundlage der Ressourcen in Ihrer Workflow-Definition generieren. Diese Richtlinien beinhalten die geringsten Rechte, die erforderlich sind, damit die State-Machine-Rolle folgende Funktionen aufrufen kann`StartExecution`API-Aktion fürStatus der verteilten Karte. Diese Richtlinien beinhalten auch die geringsten Rechte, die für den Zugriff auf Step Functions erforderlich sind.AWSRessourcen, wie Amazon S3-Bucket und -Objekte sowie Lambda-Funktionen. Wir empfehlen dringend, dass Sie nur die Berechtigungen in Ihre IAM-Richtlinien aufnehmen, die erforderlich sind. Zum Beispiel, wenn Ihr Workflow Folgendes beinhaltet`Map`Geben Sie im Modus Verteilt an, beschränken Sie Ihre Richtlinien auf den spezifischen Amazon S3 S3-Bucket und -Ordner, der Ihren Datensatz enthält.

> **Important**
>
> Wenn Sie einen Amazon S3 S3-Bucket und ein Objekt oder ein Präfix mit einem angebenReferenzpfadzu einem vorhandenen Schlüssel-Wert-Paar in IhremStatus der verteilten KarteEingabe, stellen Sie die IAM-Richtlinie für Ihren Workflow sicher. Beschränken Sie die Richtlinien auf die Bucket- und Objektnamen, zu denen der Pfad zur Laufzeit aufgelöst wird.

Die folgenden Beispiele für IAM-Richtlinien gewähren die geringsten Rechte, die für den Zugriff auf Ihre Amazon S3 S3-Datensätze erforderlich sind, indem SieListObjectsV2undGetObjectAPI-Aktionen.

Example IAM-Richtlinie für Amazon S3 S3-Objekt als Datensatz

Das folgende Beispiel zeigt eine IAM-Richtlinie, die die geringsten Rechte für den Zugriff auf die darin organisierten Objekte gewährt*`processImages`*in einem Amazon S3 S3-Bucket mit dem Namen*`myBucket`*.

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

```
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::myBucket"
            ],
            "Condition": {
                "StringLike": {
                    "s3:prefix": [
                        "processImages"
                    ]
                }
            }
        }
    ]
}
```

Example IAM-Richtlinie für eine CSV-Datei als Datensatz

Das folgende Beispiel zeigt eine IAM-Richtlinie, die die geringsten Rechte für den Zugriff auf eine CSV-Datei gewährt*ratings.csv*.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::myBucket/csvDataset/ratings.csv"
            ]
        }
    ]
}
```

Example IAM-Richtlinie für ein Amazon S3 S3-Inventar als Datensatz

Das folgende Beispiel zeigt eine IAM-Richtlinie, die die geringsten Rechte für den Zugriff auf einen Amazon S3 S3-Inventarbericht gewährt.

```
{
    "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

Verwenden Sie das `ItemsPath` Feld, um ein Array innerhalb einer JSON-Eingabe auszuwählen, die für einen `Map` Staat bereitgestellt wird. Der `Map` Status wiederholt eine Reihe von Schritten für jedes Element im Array. Standardmäßig ist der `Map` Status `ItemsPath` auf gesetzt$, wodurch die gesamte Eingabe ausgewählt wird. Wenn die Eingabe für den `Map` Status ein JSON-Array ist, führt es eine Iteration für jedes Element im Array aus und übergibt dieses Element als Eingabe an die Iteration.

> **Note**
>
> Sie können es `ItemsPath` im Status Distributed Map nur verwenden, wenn Sie eine JSON-Eingabe verwenden, die aus einem früheren Status im Workflow übergeben wurde.

Sie können das `ItemsPath` Feld verwenden, um eine Position in der Eingabe anzugeben, die auf das JSON-Array verweist, das für Iterationen verwendet wird. Der Wert von `ItemsPath` muss ein Referenzpfad sein, und dieser Pfad muss auf ein JSON-Array zeigen. Erwägen Sie beispielsweise die Eingabe in einen `Map`-Zustand, der zwei Arrays enthält, wie im folgenden Beispiel.

```
{
  "ThingsPiratesSay": [
    {
      "say": "Avast!"
```

```
    },
    {
      "say": "Yar!"
    },
    {
      "say": "Walk the Plank!"
    }
  ],
  "ThingsGiantsSay": [
    {
      "say": "Fee!"
    },
    {
      "say": "Fi!"
    },
    {
      "say": "Fo!"
    },
    {
      "say": "Fum!"
    }
  ]
}
```

In diesem Fall könnten Sie angeben, welches Array für `Map` Zustands-Iterationen verwendet werden soll, indem Sie es mit `ItemsPath` auswählen. Die folgende State-Machine-Definition spezifiziert das `ThingsPiratesSay` Array in der Eingabe `ItemsPath` mithilfe. Anschließend wird für jedes Element im `ThingsPiratesSay` Array eine Iteration des `SayWord` Pass-Status ausgeführt.

```
{
  "StartAt": "PiratesSay",
  "States": {
    "PiratesSay": {
      "Type": "Map",
      "ItemsPath": "$.ThingsPiratesSay",
      "ItemProcessor": {
         "StartAt": "SayWord",
         "States": {
           "SayWord": {
             "Type": "Pass",
             "End": true
           }
         }
```

```
      },
      "End": true
    }
  }
}
```

Bei der Verarbeitung von Eingaben gilt der `Map` Status `ItemsPath` danach `InputPath`. Es arbeitet mit der effektiven Eingabe für den Status, nachdem die Eingabe `InputPath` gefiltert wurde.

Weitere Informationen zu `Map`-Zuständen finden Sie im Folgenden:

- Bundesstaat auf der Karte
- Zustandsverarbeitungsmodi zuordnen
- Wiederholen Sie eine Aktion mit dem Status „Inline Map“
- Eingabe- und Ausgabeverarbeitung im `Map` Inline-Status

ItemSelector

Standardmäßig ist die effektive Eingabe für den `Map` Status der Satz einzelner Datenelemente, die in der Rohzustandseingabe enthalten sind. In `ItemSelector` diesem Feld können Sie die Werte der Datenelemente überschreiben, bevor sie an den `Map` Staat weitergegeben werden. Um die Werte zu überschreiben, geben Sie eine gültige JSON-Eingabe an, die eine Sammlung von Schlüssel-Wert-Paaren enthält. Bei diesen Paaren kann es sich um statische Werte handeln, die in Ihrer State-Machine-Definition bereitgestellt werden, um Werte, die über einen Pfad aus der Zustandseingabe ausgewählt wurden, oder um Werte, auf die über das Kontextobjekt zugegriffen wird.

Wenn Sie Schlüssel-Wert-Paare mithilfe eines Pfad- oder Kontextobjekts angeben, muss der Schlüsselname mit enden. `.$`

> **Note**
>
> Das `ItemSelector` Feld ersetzt das `Parameters` Feld innerhalb des `Map` Bundesstaates. Wenn Sie das `Parameters` Feld in Ihren `Map` Statusdefinitionen verwenden, um benutzerdefinierte Eingaben zu erstellen, empfehlen wir dringend, sie durch zu ersetzen`ItemSelector`.

Sie können das `ItemSelector` Feld sowohl in einem Inline-Map-Status als auch in einem Distributed-Map-Status angeben.

Stellen Sie sich zum Beispiel die folgende JSON-Eingabe vor, die ein Array mit drei Elementen innerhalb des `imageData` Knotens enthält. Für jede *Map*Zustandsiteration wird ein Array-Element als Eingabe an die Iteration übergeben.

```
[
  {
    "resize": "true",
    "format": "jpg"
  },
  {
    "resize": "false",
    "format": "png"
  },
  {
    "resize": "true",
    "format": "jpg"
  }
]
```

Mithilfe des `ItemSelector` Felds können Sie eine benutzerdefinierte JSON-Eingabe definieren, um die ursprüngliche Eingabe zu überschreiben, wie im folgenden Beispiel gezeigt. Step Functions übergibt diese benutzerdefinierte Eingabe dann an jede *Map*Zustandsiteration. Die benutzerdefinierte Eingabe enthält einen statischen Wert für `size` und den Wert eines Kontextobjekts für den `Map` Status. Das `$$.Map.Item.Value` Kontextobjekt enthält den Wert jedes einzelnen Datenelements.

```
{
  "ItemSelector": {
    "size": 10,
    "value.$": "$$.Map.Item.Value"
  }
}
```

Das folgende Beispiel zeigt die Eingabe, die bei einer Iteration des Inline-Map-Status empfangen wurde:

```
{
  "size": 10,
```

```
  "value": {
    "resize": "true",
    "format": "jpg"
  }
}
```

> **Tip**
>
> Ein vollständiges Beispiel für einen Distributed-Map-Status, der das `ItemSelector` Feld verwendet, finden Sie unterErste Schritte mit der Verwendung von Distributed Map State.

ItemBatcher

Das `ItemBatcher` Feld ist ein JSON-Objekt, das angibt, dass eine Gruppe von Elementen in einer einzigen untergeordneten Workflow-Ausführung verarbeitet werden soll. Verwenden Sie Batching, wenn Sie große CSV-Dateien oder JSON-Arrays oder große Gruppen von Amazon S3-Objekten verarbeiten.

Das folgende Beispiel zeigt die Syntax des `ItemBatcher` Felds. In der folgenden Syntax ist die maximale Anzahl von Elementen, die jede untergeordnete Workflow-Ausführung verarbeiten soll, auf 100 festgelegt.

```
{
  "ItemBatcher": {
    "MaxItemsPerBatch": 100
  }
}
```

Standardmäßig wird jedes Element in einem Datensatz als Eingabe an einzelne untergeordnete Workflow-Ausführungen übergeben. Nehmen wir beispielsweise an, Sie geben eine JSON-Datei als Eingabe an, die das folgende Array enthält:

```
[
  {
    "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"
  },
  ...
]
```

Für die angegebene Eingabe erhält jede untergeordnete Workflow-Ausführung ein Array-Element als Eingabe. Das folgende Beispiel zeigt die Eingabe einer untergeordneten Workflow-Ausführung:

```
{
  "verdict": "true",
  "statement_date": "6/11/2008",
  "statement_source": "speech"
}
```

Um die Leistung und die Kosten Ihres Verarbeitungsauftrags zu optimieren, wählen Sie eine Stapelgröße, bei der die Anzahl der Artikel gegen die Verarbeitungszeit der Artikel abgewogen wird. Wenn Sie Batching verwenden, fügt Step Functions die Elemente einem Items-Array hinzu. Anschließend übergibt es das Array als Eingabe an jede untergeordnete Workflow-Ausführung. Das folgende Beispiel zeigt einen Stapel von zwei Elementen, die als Eingabe an eine untergeordnete Workflow-Ausführung übergeben werden:

```
{
  "Items": [
    {
      "verdict": "true",
      "statement_date": "6/11/2008",
      "statement_source": "speech"
    },
    {
      "verdict": "false",
      "statement_date": "6/7/2022",
      "statement_source": "television"
    }
  ]
```

```
}
```

> **Tip**
>
> Um mehr über die Verwendung des `ItemBatcher` Felds in Ihren Workflows zu erfahren, probieren Sie die folgenden Tutorials und Workshops aus:
>
> - Verarbeiten Sie einen ganzen Datenstapel innerhalb einer Lambda-Funktion
> - Iterieren Sie innerhalb von untergeordneten Workflow-Ausführungen über Elemente in einem Stapel
> - Großflächige Parallelisierung mit verteilter Karte in Modul 14 - Datenverarbeitung des Workshops AWS Step Functions

Inhalt

Felder zur Angabe der Artikelstapelung

Um Artikel zu stapeln, geben Sie die maximale Anzahl der zu stapelnden Artikel, die maximale Stapelgröße oder beides an. Sie müssen einen dieser Werte angeben, um Artikel zu stapeln.

Max. Artikel pro Charge

Gibt die maximale Anzahl von Elementen an, die jede untergeordnete Workflow-Ausführung verarbeitet. Der Interpreter begrenzt die Anzahl der im `Items` Array gestapelten Elemente auf diesen Wert. Wenn Sie sowohl eine Chargennummer als auch eine Chargengröße angeben, reduziert der Interpreter die Anzahl der Artikel in einem Stapel, um eine Überschreitung der angegebenen Stapelgrößenbeschränkung zu vermeiden.

Wenn Sie diesen Wert nicht angeben, sondern einen Wert für die maximale Batchgröße angeben, verarbeitet Step Functions in jeder untergeordneten Workflow-Ausführung so viele Elemente wie möglich, ohne die maximale Batchgröße in Byte zu überschreiten.

Stellen Sie sich zum Beispiel vor, Sie führen eine Ausführung mit einer JSON-Eingabedatei aus, die 1130 Knoten enthält. Wenn Sie für jeden Stapel einen maximalen Artikelwert von 100 angeben, erstellt Step Functions 12 Stapel. Davon enthalten 11 Chargen jeweils 100 Artikel, während die zwölfte Charge die restlichen 30 Artikel enthält.

Alternativ können Sie die maximale Anzahl an Elementen für jeden Stapel als Referenzpfad zu einem vorhandenen Schlüssel-Wert-Paar in Ihrer Distributed-Map-Statuseingabe angeben. Dieser Pfad muss zu einer positiven Ganzzahl führen.

Beispielsweise angesichts der folgenden Eingabe:

```
{
  "maxBatchItems": 500
}
```

Sie können die maximale Anzahl der zu stapelnden Elemente mithilfe eines Referenzpfads wie folgt angeben:

```
{
  ...
  "Map": {
    "Type": "Map",
    "MaxConcurrency": 2000,
    "ItemBatcher": {
      "MaxItemsPerBatchPath": "$.maxBatchItems"
    }
    ...
    ...
  }
}
```

> **Important**
>
> Sie können entweder das `MaxItemsPerBatch` oder das `MaxItemsPerBatchPath` Unterfeld angeben, aber nicht beide.

Max. KB pro Stapel

Gibt die maximale Größe eines Stapels in Byte an, bis zu 256 KB. Wenn Sie sowohl eine maximale Chargennummer als auch eine maximale Chargengröße angeben, reduziert Step Functions die Anzahl der Artikel in einem Stapel, um eine Überschreitung der angegebenen Stapelgrößenbeschränkung zu vermeiden.

Alternativ können Sie die maximale Batchgröße als Referenzpfad zu einem vorhandenen Schlüssel-Wert-Paar in Ihrer Distributed-Map-State-Eingabe angeben. Dieser Pfad muss zu einer positiven Ganzzahl führen.

> **Note**
>
> Wenn Sie Batching verwenden und keine maximale Batchgröße angeben, verarbeitet der Interpreter so viele Elemente, wie er verarbeiten kann, bis zu 256 KB in jeder untergeordneten Workflow-Ausführung.

Beispielsweise angesichts der folgenden Eingabe:

```
{
  "batchSize": 131072
}
```

Sie können die maximale Stapelgröße mithilfe eines Referenzpfads wie folgt angeben:

```
{
  ...
  "Map": {
    "Type": "Map",
    "MaxConcurrency": 2000,
    "ItemBatcher": {
      "MaxInputBytesPerBatchPath": "$.batchSize"
    }
    ...
    ...
  }
}
```

> **Important**
>
> Sie können entweder das `MaxInputBytesPerBatch` oder das `MaxInputBytesPerBatchPath` Unterfeld angeben, aber nicht beide.

Batch-Eingabe

Optional können Sie auch eine feste JSON-Eingabe angeben, die in jeden Batch aufgenommen werden soll, der an jede untergeordnete Workflow-Ausführung übergeben wird. Step Functions führt diese Eingabe mit der Eingabe für jede einzelne untergeordnete Workflow-Ausführung zusammen. Zum Beispiel bei der folgenden festen Eingabe eines Datums für die Faktenprüfung für eine Reihe von Elementen:

```
"ItemBatcher": {
    "BatchInput": {
        "factCheck": "December 2022"
    }
}
```

Jede untergeordnete Workflow-Ausführung erhält Folgendes als Eingabe:

```
{
  "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

Das `ResultWriter` Feld ist ein JSON-Objekt, das den Amazon S3 S3-Speicherort angibt, an den Step Functions die Ergebnisse der untergeordneten Workflow-Ausführungen schreibt, die durch einen

Distributed Map-Status gestartet wurden. Standardmäßig exportiert Step Functions diese Ergebnisse nicht.

> **Important**
>
> Stellen Sie sicher, dass sich der Amazon S3 S3-Bucket, den Sie zum Exportieren der Ergebnisse eines Map Runs verwenden, unter demselben AWS-Konto und AWS-Region wie Ihr Zustandsmaschine befindet. Andernfalls schlägt die Ausführung Ihrer Zustandsmaschine mit dem `States.ResultWriterFailed` Fehler fehl.

Das Exportieren der Ergebnisse in einen Amazon S3 S3-Bucket ist hilfreich, wenn Ihre Ausgabe-Payload-Größe 256 KB überschreitet. Step Functions konsolidiert alle Ausführungsdaten des untergeordneten Workflows, wie z. B. Ausführungseingabe und -ausgabe, ARN und Ausführungsstatus. Anschließend werden Ausführungen mit demselben Status in die entsprechenden Dateien am angegebenen Amazon S3 S3-Speicherort exportiert. Das folgende Beispiel zeigt die Syntax des `ResultWriter` Felds, wenn Sie die Ausführungsergebnisse des untergeordneten Workflows exportieren. In diesem Beispiel speichern Sie die Ergebnisse in einem Bucket, der mit einem Präfix namens benannt `myOutputBucket` ist`csvProcessJobs`.

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

> **Tip**
>
> In Workflow Studio können Sie die Ergebnisse der untergeordneten Workflow-Ausführung exportieren, indem Sie Map-Status-Ergebnisse nach Amazon S3 exportieren auswählen. Geben Sie dann den Namen des Amazon S3 S3-Buckets und das Präfix an, in das Sie die Ergebnisse exportieren möchten.

Step Functions benötigt die entsprechenden Berechtigungen, um auf den Bucket und den Ordner zuzugreifen, in den Sie die Ergebnisse exportieren möchten. Informationen zur erforderlichen IAM-Richtlinie finden Sie unterIAM-Richtlinien für ResultWriter.

Wenn Sie die Ergebnisse der untergeordneten Workflow-Ausführung exportieren, gibt die Ausführung des Status Distributed Map den Map Run-ARN und Daten über den Amazon S3 S3-Exportspeicherort im folgenden Format zurück:

```
{
  "MapRunArn": "arn:aws:states:us-
east-2:123456789012:mapRun:csvProcess/Map:ad9b5f27-090b-3ac6-9beb-243cd77144a7",
  "ResultWriterDetails": {
    "Bucket": "myOutputBucket",
    "Key": "csvProcessJobs/ad9b5f27-090b-3ac6-9beb-243cd77144a7/manifest.json"
  }
}
```

Step Functions exportiert Ausführungen mit demselben Status in ihre jeweiligen Dateien. Wenn Ihre untergeordneten Workflow-Ausführungen beispielsweise zu 500 Erfolgs- und 200 Fehlschlagsergebnissen geführt haben, erstellt Step Functions am angegebenen Amazon S3 S3-Speicherort zwei Dateien für die Erfolgs- und Fehlschlagsergebnisse. In diesem Beispiel enthält die Datei mit den Erfolgsergebnissen die 500 Erfolgsergebnisse, während die Datei mit den Fehlschlagsergebnissen die 200 Fehlerergebnisse enthält.

Für einen bestimmten Ausführungsversuch erstellt Step Functions je nach Ihrer Ausführungsausgabe die folgenden Dateien am angegebenen Amazon S3 S3-Speicherort:

- `manifest.json`— Enthält Map Run-Metadaten wie Exportadresse, Map Run-ARN und Informationen zu den Ergebnisdateien.

  Wenn Sie über redriveneinen Map Run verfügen, enthält die `manifest.json` Datei Verweise auf alle erfolgreichen untergeordneten Workflow-Ausführungen bei allen Versuchen eines Map Runs. Diese Datei enthält jedoch Verweise auf die fehlgeschlagenen und ausstehenden Ausführungen für einen bestimmten. redrive
- `SUCCEEDED_n.json`— Enthält die konsolidierten Daten für alle erfolgreichen untergeordneten Workflow-Ausführungen. n steht für die Indexnummer der Datei. Die Indexnummer beginnt bei 0. Zum Beispiel `SUCCEEDED_1.json`.
- `FAILED_n.json`— Enthält die konsolidierten Daten für alle fehlgeschlagenen, zeitüberschreitenden und abgebrochenen untergeordneten Workflow-Ausführungen. Verwenden

Sie diese Datei, um nach fehlgeschlagenen Ausführungen Daten wiederherzustellen. n steht für den Index der Datei. Die Indexnummer beginnt bei 0. Zum Beispiel `FAILED_1.json`.

- `PENDING_n.json`— Enthält die konsolidierten Daten für alle untergeordneten Workflow-Ausführungen, die nicht gestartet wurden, weil der Map Run fehlgeschlagen oder abgebrochen wurde. n steht für den Index der Datei. Die Indexnummer beginnt bei 0. Zum Beispiel `PENDING_1.json`.

Step Functions unterstützt einzelne Ergebnisdateien mit bis zu 5 GB. Wenn eine Dateigröße 5 GB überschreitet, erstellt Step Functions eine weitere Datei, um die verbleibenden Ausführungsergebnisse zu schreiben, und hängt eine Indexnummer an den Dateinamen an. Wenn die Größe der `Succeeded_0.json` Datei beispielsweise 5 GB überschreitet, erstellt Step Functions eine `Succeeded_1.json` Datei, um die verbleibenden Ergebnisse aufzuzeichnen.

Wenn Sie nicht angegeben haben, dass die Ergebnisse der untergeordneten Workflow-Ausführung exportiert werden sollen, gibt die State-Machine-Ausführung eine Reihe von Ergebnissen der untergeordneten Workflow-Ausführung zurück, wie im folgenden Beispiel gezeigt:

> **Note**
>
> Wenn die zurückgegebene Ausgabegröße 256 KB überschreitet, schlägt die Ausführung der Zustandsmaschine fehl und es wird ein `States.DataLimitExceeded` Fehler zurückgegeben.

```
[
  {
    "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"
    }
  },
  ...
]
```

IAM-Richtlinien für ResultWriter

Wenn Sie Workflows mit der Step Functions-Konsole erstellen, kann Step Functions automatisch IAM-Richtlinien auf der Grundlage der Ressourcen in Ihrer Workflow-Definition generieren. Diese Richtlinien beinhalten die geringsten Rechte, die erforderlich sind, damit die Zustandsmaschinen-Rolle die `StartExecution` API-Aktion für den Status Distributed Map aufrufen kann. Diese Richtlinien beinhalten auch die Step Functions mit den geringsten Rechten, die für den Zugriff auf AWS Ressourcen wie Amazon S3 S3-Buckets und -Objekte sowie Lambda-Funktionen erforderlich sind. Wir empfehlen dringend, dass Sie nur die Berechtigungen in Ihre IAM-Richtlinien aufnehmen, die erforderlich sind. Wenn Ihr Workflow beispielsweise einen `Map` Status im Modus „Verteilt“ umfasst, beschränken Sie Ihre Richtlinien auf den spezifischen Amazon S3 S3-Bucket und -Ordner, der Ihren Datensatz enthält.

> **Important**
>
> Wenn Sie einen Amazon S3 S3-Bucket und ein Objekt oder ein Präfix mit einem Referenzpfad zu einem vorhandenen Schlüssel-Wert-Paar in Ihrer Distributed Map-Statuseingabe angeben, stellen Sie sicher, dass Sie die IAM-Richtlinien für Ihren Workflow aktualisieren. Beschränken Sie die Richtlinien auf die Bucket- und Objektnamen, zu denen der Pfad zur Laufzeit aufgelöst wird.

Das folgende Beispiel für eine IAM-Richtlinie gewährt die geringsten Rechte, die erforderlich sind, um die Ergebnisse der Workflow-Ausführung Ihres untergeordneten Workflows mithilfe der API-Aktion in einen Ordner mit dem Namen *`csvJobs`* in einem Amazon S3 S3-Bucket zu schreiben. `PutObject`

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
```

```
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListMultipartUploadParts",
                "s3:AbortMultipartUpload"
            ],
            "Resource": [
                "arn:aws:s3:::resultBucket/csvJobs/*"
            ]
        }
    ]
}
```

Wenn der Amazon S3 S3-Bucket, in den Sie das Ergebnis der untergeordneten Workflow-Ausführung schreiben, mit einem AWS Key Management Service (AWS KMS) Schlüssel verschlüsselt ist, müssen Sie die erforderlichen AWS KMS Berechtigungen in Ihre IAM-Richtlinie aufnehmen. Weitere Informationen finden Sie unter IAM-Berechtigungen für AWS KMS key verschlüsselten Amazon S3 S3-Bucket.

Analysieren einer CSV-Eingabedatei

Da es kein standardisiertes Format zum Erstellen und Verwalten von Daten in CSV-Dateien gibt, analysiert Step Functions CSV-Dateien auf der Grundlage der folgenden Regeln:

- Kommas (,) sind ein Trennzeichen, das einzelne Felder voneinander trennt.
- Zeilenumbrüche sind ein Trennzeichen, das einzelne Datensätze voneinander trennt.
- Felder werden als Zeichenketten behandelt. Verwenden Sie für Datentypkonvertierungen den`States.StringToJson`intrinsische Funktion in`ItemSelector`.
- Doppelte Anführungszeichen (“ „) sind nicht erforderlich, um Zeichenketten einzuschließen. Zeichenketten, die in doppelte Anführungszeichen eingeschlossen sind, können jedoch Kommas und Zeilenumbrüche enthalten, ohne dass sie als Trennzeichen dienen.
- Umgehen Sie doppelte Anführungszeichen, indem Sie sie wiederholen.
- Wenn die Anzahl der Felder in einer Zeile geringer ist als die Anzahl der Felder in der Kopfzeile, stellt Step Functions leere Zeichenketten für die fehlenden Werte bereit.
- Wenn die Anzahl der Felder in einer Zeile größer ist als die Anzahl der Felder in der Kopfzeile, überspringt Step Functions die zusätzlichen Felder.

Example des Analysierens einer CSV-Eingabedatei

Angenommen, Sie haben eine CSV-Datei mit dem Namen bereitgestellt*myCSVInput.csv*das enthält eine Zeile als Eingabe. Anschließend haben Sie diese Datei in einem Amazon-S3-Bucket gespeichert.*my-bucket*. Die CSV-Datei lautet wie folgt.

```
abc,123,"This string contains commas, a double quotation marks (""), and a newline (
)",{""MyKey"":""MyValue""},"[1,2,3]"
```

Die folgende Zustandsmaschine liest diese CSV-Datei und verwendetItemSelectorum die Datentypen einiger Felder zu konvertieren.

```
{
  "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)"
      }
    }
  }
}
```

Wenn Sie diesen Zustandsmaschine ausführen, erzeugt er die folgende Ausgabe.

```
[
  {
    "MyNumbers": 123,
    "MyObject": {
      "MyKey": "MyValue"
    },
    "MyString": "This string contains commas, a double quote (\"), and a newline (\n)",
    "MyLetters": "abc",
    "MyArray": [
      1,
      2,
      3
    ]
  }
]
```

Context-Objekt

Das Kontextobjekt ist eine interne JSON-Struktur, die während einer Ausführung verfügbar ist und Informationen über Ihre Zustandsmaschine und die Ausführung enthält. Auf diese Weise können

Ihre Workflows auf Informationen zu ihrer spezifischen Ausführung zugreifen. Sie können über die folgenden Felder auf das Kontextobjekt zugreifen:

- `InputPath`
- `OutputPath`
- `ItemsPath`(in Kartenstaaten)
- `Variable`(in den Staaten Choice)
- `ResultSelector`
- `Parameters`
- Operatoren für den Vergleich von Variablen zu Variablen

Kontext-Objekt-Format

Das Kontext-Objekt enthält Informationen über den Zustandsautomaten, den Zustand, die Ausführung und die Aufgabe. Dieses JSON-Objekt enthält Knoten für jeden Typ von Daten und hat das folgende Format.

```
{
    "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"
    }
```

```
}
```

Während einer Ausführung erhält das Kontext-Objekt relevante Daten für das `Parameters`-Feld, von dem aus darauf zugegriffen wird. Der Wert für ein `Task`-Feld ist Null, wenn sich das `Parameters`-Feld außerhalb einer Aufgabe befindet.

Der Wert des `RedriveCount` Kontextobjekts ist 0, wenn Sie noch keine Ausführung redrivendurchgeführt haben. Außerdem ist das `RedriveTime` Kontextobjekt nur verfügbar, wenn Sie redriven eine Ausführung haben. Wenn ja redriven a Map Run, ist das `RedriveTime` Kontextobjekt nur für untergeordnete Workflows vom Typ Standard verfügbar. Für einen redriven Kartenlauf mit untergeordneten Workflows vom Typ Express `RedriveTime` ist es nicht verfügbar.

Inhalte aus einer laufenden Ausführung enthalten Einzelheiten im folgenden Format.

```
{
    "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**
>
> Kontextobjektdaten im Zusammenhang mit `Map`-Zuständen finden Sie unter Context-Objektdaten für Zuordnungszustände.

Zugriff auf das Kontext-Objekt

Um auf das Kontext-Objekt zuzugreifen, geben Sie zuerst den Parameternamen an, indem Sie am Ende `.$` anhängen, wie bei der Auswahl einer Zustandseingabe mit einem Pfad. Fügen Sie dann zum Zugriff auf die Kontext-Objektdaten anstelle der Eingabe vor dem Pfad `$$.` ein. Dies weist darauf AWS Step Functions hin, dass der Pfad verwendet werden soll, um einen Knoten im Kontextobjekt auszuwählen.

Die folgenden Beispiele zeigen, wie Sie auf Kontextobjekte wie Ausführungs-ID, Name, Startzeit und redrive Anzahl zugreifen können.

- Ausführungs-ARN abrufen und an einen Downstream-Dienst übergeben
- Greifen Sie im Status Pass auf die Startzeit und den Namen der Ausführung zu
- Greifen Sie auf die redrive Anzahl einer Ausführung zu

Ausführungs-ARN abrufen und an einen Downstream-Dienst übergeben

Dieses Beispiel für den Aufgabenstatus verwendet einen Pfad, um den Amazon Resource Name (ARN) der Ausführung abzurufen und an eine Amazon Simple Queue Service (Amazon SQS) - Nachricht weiterzuleiten.

```
{
  "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"
  }
}
```

Weitere Informationen zur Verwendung eines Aufgabentokens beim Aufruf eines integrierten Service finden Sie unter Warten auf einen Callback mit dem Aufgabentoken.

Greifen Sie im Status Pass auf die Startzeit und den Namen der Ausführung zu

```
{
  "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
    }
  }
}
```

Greifen Sie auf die redrive Anzahl einer Ausführung zu

Das folgende Beispiel für eine Aufgabenstatusdefinition zeigt, wie Sie auf die redriveAnzahl einer Ausführung zugreifen können.

```
{
  "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
}
```

Context-Objektdaten für Zuordnungszustände

Beim Verarbeiten eines Map-Zustands stehen zwei zusätzliche Elemente im Context-Objekt zur Verfügung: `Index` und `Value`. `Index`Enthält für jede `Map` Status-Iteration die Indexnummer für das Array-Element, das gerade verarbeitet wird, und `Value` enthält gleichzeitig das Array-Element, das gerade verarbeitet wird. Innerhalb eines `Map` Zustands enthält das Kontextobjekt die folgenden Daten:

```
"Map": {
   "Item": {
      "Index": Number,
      "Value": "String"
   }
}
```

Diese sind nur in einem `Map` Bundesstaat verfügbar und können im `ItemSelector` Feld angegeben werden.

> **Note**
>
> Sie müssen Parameter aus dem Context-Objekt im Block `ItemSelector` des `Map`-Hauptzustands definieren, nicht innerhalb der Zustände im Abschnitt `ItemProcessor`.

Bei einem Zustandsautomaten mit einem einfachen `Map`-Zustand können wir wie folgt Informationen aus dem Kontextobjekt einfügen.

```
{
  "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
    }
  }
}
```

Wenn Sie den vorherigen Zustandsautomaten mit der folgenden Eingabe ausführen, werden `Index` und `Value` in die Ausgabe eingefügt.

```
[
  {
    "who": "bob"
  },
  {
    "who": "meg"
  },
  {
    "who": "joe"
  }
]
```

Die Ausgabe für die Ausführung gibt die Werte `Index` und `Value` Elemente für jede der drei Iterationen wie folgt zurück:

```
[
  {
    "ContextIndex": 0,
    "ContextValue": {
      "who": "bob"
    }
  },
  {
    "ContextIndex": 1,
    "ContextValue": {
      "who": "meg"
```

```
    }
  },
  {

    "ContextIndex": 2,
    "ContextValue": {
      "who": "joe"
    }
  }
]
```

Datenflusssimulator

In der Step Functions-Konsole können Sie Workflows entwerfen, implementieren und debuggen. Sie können den Datenfluss in Ihren Workflows auch durch die Verarbeitung von JsonPathEingabe- und Ausgabedaten steuern. Mit dem Datenflusssimulator können Sie die Reihenfolge simulieren, in der die Status der Aufgabe Staaten in Ihrem Workflow Daten zur Laufzeit verarbeiten. Mithilfe des Simulators können Sie verstehen, wie Daten gefiltert und bearbeitet werden, während sie von einem Zustand in einen anderen fließen. Es simuliert jedes der folgenden Felder, die Step Functions verwendet, um den Fluss von JSON-Daten zu verarbeiten und zu steuern:

`InputPath`

Wählt aus, WELCHER Teil der gesamten Eingabe-Payload als Eingabe für eine Aufgabe verwendet werden soll. Wenn Sie dieses Feld angeben, wendet Step Functions dieses Feld zuerst an.

`Parameter`

Gibt an, WIE die Eingabe aussehen soll, bevor die Aufgabe aufgerufen wird. Mit dem `Parameters` Feld können Sie eine Sammlung von Schlüssel-Wert-Paaren erstellen, die als Eingabe an eine AWS-ServiceIntegration, z. B. eine AWS Lambda Funktion, übergeben werden. Diese Werte können statisch sein oder dynamisch aus der Statuseingabe oder dem Workflow-Kontextobjekt ausgewählt werden.

`ResultSelector`

Legt fest, WAS aus der Ausgabe einer Aufgabe ausgewählt werden soll. Mit dem `ResultSelector` Feld können Sie eine Sammlung von Schlüssel-Wert-Paaren erstellen, die das Ergebnis eines Zustands ersetzen, und diese Sammlung an diese übergeben. `ResultPath`

`ResultPath`

Legt fest, WO die Ausgabe einer Aufgabe abgelegt werden soll. Verwenden Sie den`ResultPath`, um festzustellen, ob die Ausgabe eines Zustands eine Kopie seiner Eingabe, des Ergebnisses, das er erzeugt, oder eine Kombination aus beidem ist.

`OutputPath`

Legt fest, WAS an den nächsten Status gesendet werden soll. Mit `OutputPath` können Sie unerwünschte Informationen herausfiltern und nur den Teil der JSON-Daten weitergeben, der Ihnen wichtig ist.

In diesem Thema

Verwenden des Datenflusssimulators

Der Simulator bietet einen side-by-side Echtzeitvergleich Ihrer Daten vor und nach dem Anwenden eines Eingabe- und Ausgabedatenverarbeitungsfeldes. Um den Simulator zu verwenden, geben Sie eine JSON-Eingabe an. Bewerten Sie es dann anhand der einzelnen Eingabe- und Ausgabeverarbeitungsfelder. Der Simulator validiert automatisch Ihre JSON-Eingabe und hebt alle Syntaxfehler hervor.

Um den Datenflusssimulator zu verwenden

In den folgenden Schritten geben Sie JSON-Eingaben ein und wenden die `Parameter` Felder `InputPath` und an. Sie können auch die anderen verfügbaren Felder anwenden und sich deren Ausgaben ansehen.

1. Öffnen Sie die Step Functions-Konsole.
2. Wählen Sie im Navigationsbereich Datenflusssimulator aus.
3. Ersetzen Sie im Eingabebereich Bundesstaat die vorab ausgefüllten JSON-Beispieldaten durch die folgenden JSON-Daten. Wählen Sie anschließend Weiter aus.

```
{
  "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. Geben Sie ein InputPath, **`$.data.address`** um den Adressknoten der JSON-Eingabedaten auszuwählen.

   Das InputPath Feld „State input after“ zeigt die folgenden Ergebnisse an.

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

5. Wählen Sie Weiter aus.
6. Wenden Sie das `Parameters` Feld an, um die resultierenden JSON-Daten in eine Zeichenfolge zu konvertieren. Gehen Sie wie folgt vor, um die Daten zu konvertieren:

   - Geben Sie im Feld Parameter den folgenden Code ein, um eine Zeichenfolge mit dem Namen zu erstellen`addressString`.

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

7. Sehen Sie sich das Ergebnis der `Parameters` Feldanwendung im Feld Gefilterte Eingabe nach Parametern an.

Überlegungen zur Verwendung des Datenflusssimulators

Bevor Sie den Datenflusssimulator verwenden, sollten Sie dessen Einschränkungen berücksichtigen, einschließlich, aber nicht beschränkt auf:

- Nicht unterstützte Filterausdrücke

  Filterausdrücke im Simulator verhalten sich anders als im Step Functions-Dienst. Dazu gehören Ausdrücke, die die folgende Syntax verwenden:`[?(expression)]`. Im Folgenden finden Sie eine Liste von Ausdrücken, die nicht unterstützt werden. Wenn diese Ausdrücke verwendet werden, liefern sie nach ihrer Auswertung möglicherweise nicht das erwartete Ergebnis.

  - `$..book[?(@.isInStock==true)]`
  - `$..book[?(@.price > 10.0)].title`
- Falsche JsonPath Bewertung für Arrays mit einem einzelnen Artikel

  Wenn Sie Ihre Daten mit einem JsonPath Ausdruck filtern, der ein einzelnes Array-Element zurückgeben würde, gibt der Simulator das Element ohne das Array zurück. Stellen Sie sich zum Beispiel das folgende Datenarray mit dem Namen vor`items`:

```
{
  "items": [
    {
      "name": "shoe",
      "color": "blue",
      "comment": "nice shoe"
    },
    {
      "name": "hat",
      "color": "red"
    },
    {
      "name": "shirt",
      "color": "yellow"
    }
  ]
}
```

  Wenn Sie bei diesem `items` Array **`$..comment`** in das InputPath Feld eingeben, würden Sie die folgende Ausgabe erwarten:

```
[
  "nice shoe"
]
```

Der Datenflusssimulator gibt jedoch stattdessen die folgende Ausgabe zurück:

```
"nice shoe"
```

Zur JsonPath Auswertung eines Arrays, das mehrere Elemente enthält, gibt der Simulator die erwartete Ausgabe zurück.

Verwaltung kontinuierlicher Bereitstellungen mit Versionen und Aliasnamen

Sie können Step Functions verwenden, um die kontinuierliche Bereitstellung Ihrer Workflows mithilfe von State-Machine-Versionen und Aliasen zu verwalten. Eine Version ist ein nummerierter, unveränderlicher Snapshot einer State-Maschine, den Sie ausführen können. Ein Alias ist ein Zeiger für bis zu zwei Versionen einer State Machine.

Sie können mehrere Versionen Ihrer State Machines verwalten und deren Bereitstellung in Ihrem Produktionsablauf verwalten. Mit Aliasen können Sie den Datenverkehr zwischen verschiedenen Workflow-Versionen weiterleiten und diese Workflows schrittweise in der Produktionsumgebung bereitstellen.

Darüber hinaus können Sie State-Machine-Ausführungen mithilfe einer Version oder eines Alias starten. Wenn Sie beim Starten einer State-Machine-Ausführung keine Version oder keinen Alias verwenden, verwendet Step Functions die neueste Version der State-Machine-Definition.

> **Revision der staatlichen Maschine**
>
> Eine Zustandsmaschine kann eine oder mehrere Revisionen haben. Wenn Sie eine State-Maschine mithilfe der UpdateStateMachineAPI-Aktion aktualisieren, erstellt sie eine neue State-Machine-Revision. Eine Revision ist ein unveränderlicher, schreibgeschützter Snapshot der Definition und Konfiguration einer State-Maschine. Sie können eine State-Machine-Ausführung nicht von einer Revision aus starten, und Revisionen haben keinen ARN.

Revisionen haben einerevisionId, bei der es sich um eine Universally Unique Identifier (UUID) handelt.

Inhalt

Versionen des Zustandsautomaten

Eine Version ist ein nummerierter, unveränderlicher Snapshot eines Zustandsautomaten. Sie veröffentlichen Versionen der letzten Revision, die an diesem Zustandsautomaten vorgenommen wurde. Jede Version hat einen eindeutigen Amazon-Ressourcennamen (ARN). Dieser ARN ist eine Kombination aus Zustandsautomat-ARN und der durch einen Doppelpunkt (:) getrennten Versionsnummer. Das folgende Beispiel zeigt das Format eines Zustandsautomaten-Versions-ARN.

```
arn:partition:states:region:account-id:stateMachine:myStateMachine:1
```

Um mit der Verwendung von Zustandsautomaten-Versionen zu beginnen, müssen Sie die erste Version veröffentlichen. Nachdem Sie eine Version veröffentlicht haben, können Sie die StartExecution API-Aktion mit dem Versions-ARN aufrufen. Sie können eine Version nicht bearbeiten, aber Sie können einen Zustandsautomaten aktualisieren und eine neue Version veröffentlichen. Sie können auch mehrere Versionen Ihres Zustandsautomaten veröffentlichen.

Wenn Sie eine neue Version Ihres Zustandsautomaten veröffentlichen, weist Step Functions ihm eine Versionsnummer zu. Versionsnummern beginnen bei 1 und nehmen für jede neue Version monoton zu. Versionsnummern werden für einen bestimmten Zustandsautomaten nicht wiederverwendet. Wenn Sie Version 10 Ihres Zustandsautomaten löschen und dann eine neue Version veröffentlichen, veröffentlicht Step Functions sie als Version 11.

Die folgenden Eigenschaften sind für alle Versionen eines Zustandsautomaten identisch:

- Alle Versionen eines Zustandsautomaten haben denselben Typ (Standard oder Express).
- Sie können den Namen oder das Erstellungsdatum eines Zustandsautomaten nicht zwischen Versionen ändern.
- Tags gelten global für Zustandsautomaten. Sie können Tags für Zustandsautomaten mithilfe der APIUntagResource-Aktionen TagResource und verwalten.

Zustandsautomaten enthalten auch Eigenschaften, die Teil jeder Version und sindrevision, aber diese Eigenschaften können sich zwischen zwei bestimmten Versionen oder Revisionen unterscheiden. Zu diesen Eigenschaften gehören die Definition des Zustandsautomaten , die IAM-Rolle , die Nachverfolgungskonfiguration und die Protokollierungskonfiguration .

Inhalt

Veröffentlichen einer Zustandsautomaten-Version (Konsole)

Sie können bis zu 1000 Versionen eines Zustandsautomaten veröffentlichen. Um eine Erhöhung dieses temporären Limits anzufordern, verwenden Sie die Seite Support Center in der AWS Management Console. Sie können ungenutzte Versionen manuell aus der Konsole oder durch Aufrufen der DeleteStateMachineVersion -API-Aktion löschen.

So veröffentlichen Sie eine Zustandsautomaten-Version

1. Öffnen Sie die Step-Functions-Konsole und wählen Sie dann einen vorhandenen Zustandsautomaten aus.
2. Wählen Sie auf der Detailseite Zustandsautomat die Option Bearbeiten aus.

3. Bearbeiten Sie die Definition des Zustandsautomaten nach Bedarf und wählen Sie dann Speichern aus.
4. Wählen Sie Publish version (Version veröffentlichen) aus.
5. (Optional) Geben Sie im daraufhin angezeigten Dialogfeld Beschreibung eine kurze Beschreibung der Version des Zustandsautomaten ein.
6. Wählen Sie Publish.

> **Note**
>
> Wenn Sie eine neue Version Ihres Zustandsautomaten veröffentlichen, weist Step Functions ihm eine Versionsnummer zu. Versionsnummern beginnen bei 1 und nehmen für jede neue Version monoton zu. Versionsnummern werden für einen bestimmten Zustandsautomaten nicht wiederverwendet. Wenn Sie Version 10 Ihres Zustandsautomaten löschen und dann eine neue Version veröffentlichen, veröffentlicht Step Functions sie als Version 11.

Verwalten von Versionen mit Step-Functions-API-Operationen

Step Functions bietet die folgenden API-Operationen zum Veröffentlichen und Verwalten von Zustandsautomatenversionen:

- PublishStateMachineVersion – Veröffentlicht eine Version aus dem aktuellen eines revision Zustandsautomaten.
- UpdateStateMachine – Veröffentlicht eine neue Zustandsautomatenversion, wenn Sie einen Zustandsautomaten aktualisieren und den `publish` Parameter `true` in derselben Anforderung auf setzen.
- CreateStateMachine – Veröffentlicht die erste Revision des Zustandsautomaten, wenn Sie den `publish` Parameter auf setzen`true`.
- ListStateMachineVersions – Listet Versionen für den angegebenen Zustandsautomaten-ARN auf.
- DescribeStateMachine – Gibt die Versionsdetails des Zustandsautomaten für einen in angegebenen Versions-ARN zurück`stateMachineArn`.
- DeleteStateMachineVersion – Löscht eine Zustandsautomatenversion.

Um eine neue Version aus der aktuellen Revision eines Zustandsautomaten namens *myStateMachine* mit der zu veröffentlichen AWS Command Line Interface, verwenden Sie den `publish-state-machine-version` Befehl :

```
aws stepfunctions publish-state-machine-version --state-machine-arn arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine
```

In der Antwort wird der `stateMachineVersionArn` zurückgegeben. Der vorherige Befehl gibt beispielsweise die Antwort zurück`arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine:1`.

> **Note**
>
> Wenn Sie eine neue Version Ihres Zustandsautomaten veröffentlichen, weist Step Functions ihm eine Versionsnummer zu. Versionsnummern beginnen bei 1 und nehmen für jede neue Version monoton zu. Versionsnummern werden für einen bestimmten Zustandsautomaten nicht wiederverwendet. Wenn Sie Version 10 Ihres Zustandsautomaten löschen und dann eine neue Version veröffentlichen, veröffentlicht Step Functions sie als Version 11.

Ausführen einer Zustandsautomaten-Version über die Konsole

Um mit der Verwendung von Zustandsautomaten-Versionen zu beginnen, müssen Sie zunächst eine Version vom aktuellen Zustandsautomaten veröffentlichenrevision. Um eine Version zu veröffentlichen, verwenden Sie die Step Functions-Konsole oder rufen Sie die PublishStateMachineVersion API-Aktion auf. Sie können die UpdateStateMachineAlias API-Aktion auch mit einem optionalen Parameter namens aufrufen`publish`, um einen Zustandsautomaten zu aktualisieren und seine Version zu veröffentlichen.

Sie können mit der Ausführung einer Version beginnen, indem Sie die -Konsole verwenden oder die StartExecution -API-Aktion aufrufen und den Versions-ARN angeben. Sie können auch einen Alias verwenden, um die Ausführung einer Version zu starten. Basierend auf seiner Routing-Konfiguration leitet ein Alias den Datenverkehr an eine bestimmte Version weiter.

Wenn Sie eine Zustandsautomaten-Ausführung starten, ohne eine Version zu verwenden, verwendet Step Functions die neueste Revision des Zustandsautomaten für die Ausführung. Informationen darüber, wie Step Functions eine Ausführung einer Version zuordnet, finden Sie unter Zuordnen von Ausführungen zu einer Version oder einem Alias.

So starten Sie eine Ausführung mit einer Zustandsautomaten-Version

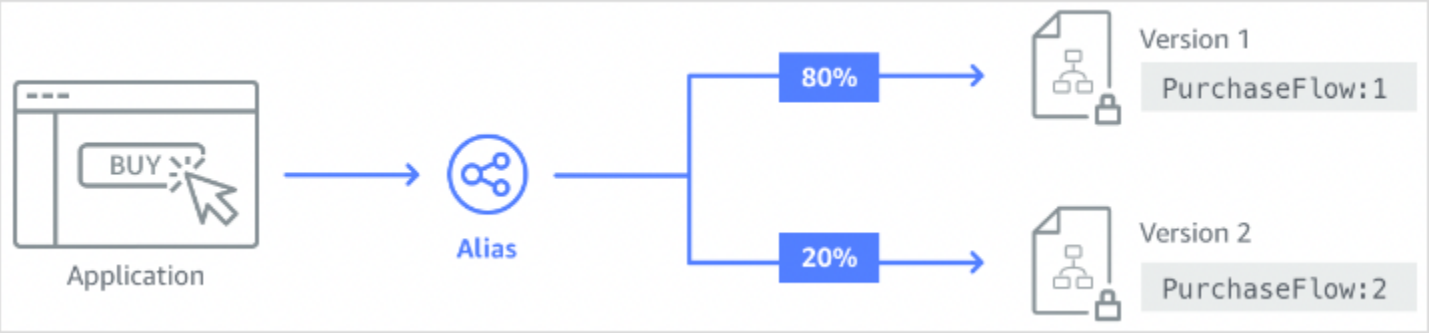
1. Öffnen Sie die Step-Functions-Konsole und wählen Sie dann einen vorhandenen Zustandsautomaten aus, für den Sie eine oder mehrere Versionen veröffentlicht haben. Informationen zum Veröffentlichen einer Version finden Sie unter Veröffentlichen einer Zustandsautomaten-Version (Konsole).
2. Wählen Sie auf der Detailseite des Zustandsautomaten die Registerkarte Versionen aus.
3. Gehen Sie im Abschnitt Versionen wie folgt vor:
   a. Wählen Sie die Version aus, mit der Sie die Ausführung starten möchten.
   b. Wählen Sie Start execution (Ausführung starten) aus.
4. (Optional) Geben Sie im Dialogfeld Ausführung starten einen Namen für die Ausführung ein.
5. (Optional) Geben Sie die Ausführungseingabe ein und wählen Sie dann Ausführung starten aus.

Aliase für Zustandsautomaten

Ein Alias ist ein Zeiger für bis zu zwei Versionen desselben Zustandsautomaten. Sie können mehrere Aliase für Ihre Zustandsautomaten erstellen. Jeder Alias hat einen eindeutigen Amazon-Ressourcennamen (ARN). Der Alias-ARN ist eine Kombination aus dem ARN des Zustandsautomaten und dem Aliasnamen, getrennt durch einen Doppelpunkt (:). Das folgende Beispiel zeigt das Format eines Zustandsautomaten-Alias-ARN.

```
arn:partition:states:region:account-id:stateMachine:myStateMachine:aliasName
```

Sie können einen Alias verwenden, um den Datenverkehr zwischen einer der beiden Zustandsautomaten-Versionen weiterzuleiten. Sie können auch einen Alias erstellen, der auf eine einzelne Version verweist. Aliase können nur auf Zustandsautomatenversionen verweisen. Sie können keinen Alias verwenden, um auf einen anderen Alias zu verweisen. Sie können auch einen Alias aktualisieren, der auf eine andere Version des Zustandsautomaten verweist.



Inhalt

Erstellen eines Zustandsautomaten-Alias (Konsole)

Sie können bis zu 100 Aliase für jeden Zustandsautomaten erstellen, indem Sie die Step-Functions-Konsole verwenden oder die CreateStateMachineAlias API-Aktion aufrufen. Um eine Erhöhung dieses Soft Limits anzufordern, verwenden Sie die Seite Support Center in der AWS Management Console. Löschen Sie nicht verwendete Aliase aus der Konsole oder durch Aufrufen der DeleteStateMachineAlias API-Aktion .

So erstellen Sie einen Zustandsautomaten-Alias

1. Öffnen Sie die Step-Functions-Konsole und wählen Sie dann einen vorhandenen Zustandsautomaten aus.
2. Wählen Sie auf der Detailseite des Zustandsautomaten die Registerkarte Aliasse aus.
3. Wählen Sie Neuen Alias erstellen aus.
4. Führen Sie auf der Seite Alias erstellen die folgenden Schritte aus:
   a. Geben Sie einen Aliasnamen ein.
   b. (Optional) Geben Sie eine Beschreibung für den Alias ein.
5. Informationen zum Konfigurieren des Routings für den Alias finden Sie unter Alias-Routing-Konfiguration.
6. Wählen Sie Alias erstellen aus.

Verwalten von Aliassen mit Step-Functions-API-Operationen

Step Functions bietet die folgenden API-Operationen, mit denen Sie Zustandsautomaten-Aliase erstellen und verwalten oder Informationen über die Aliase abrufen können:

- CreateStateMachineAlias – Erstellt einen Alias für einen Zustandsautomaten.
- DescribeStateMachineAlias – Gibt Details zu einem Zustandsautomaten-Alias zurück.

- ListStateMachineAliases – Listet Aliase für den angegebenen ARN des Zustandsautomaten auf.
- UpdateStateMachineAlias – Aktualisiert die Konfiguration eines vorhandenen Zustandsautomaten-Alias, indem dessen `description` oder geändert wird`routingConfiguration`.
- DeleteStateMachineAlias – Löscht eine Zustandsautomatenversion.

Verwenden Sie den Befehl *`PROD`*, um einen Alias mit dem Namen zu erstellen, der auf Version 1 eines Zustandsautomaten mit dem Namen *`myStateMachine`* verweist AWS Command Line Interface`create-state-machine-alias`.

```
aws stepfunctions create-state-machine-alias --name PROD  --routing-
configuration "[{\"stateMachineVersionArn\":\"arn:aws:states:us-
east-1:123456789012:stateMachine:myStateMachine:1\",\"weight\":100}]"
```

Alias-Weiterleitungskonfiguration

Sie können einen Alias verwenden, um den Ausführungsdatenverkehr zwischen zwei Versionen eines Zustandsautomaten weiterzuleiten. Angenommen, Sie möchten eine neue Version Ihres Zustandsautomaten starten. Sie können die mit der Bereitstellung der neuen Version verbundenen Risiken reduzieren, indem Sie Routing für einen Alias konfigurieren. Durch die Konfiguration von Routing können Sie den größten Teil Ihres Datenverkehrs an eine frühere, getestete Version Ihres Zustandsautomaten senden. Die neue Version kann dann einen kleineren Prozentsatz erhalten, bis Sie bestätigen können, dass es sicher ist, die neue Version vorzuziehen.

Um die Routing-Konfiguration zu definieren, stellen Sie sicher, dass Sie beide Zustandsautomatenversionen veröffentlichen, auf die Ihr Alias verweist. Wenn Sie eine Ausführung über einen Alias starten, wählt Step Functions nach dem Zufallsprinzip die Zustandsautomatenversion aus, die von den in der Routing-Konfiguration angegebenen Versionen ausgeführt werden soll. Diese Auswahl basiert auf dem Prozentsatz des Datenverkehrs, den Sie jeder Version in der Alias-Routing-Konfiguration zuweisen.

So konfigurieren Sie die Routing-Konfiguration für einen Alias

- Gehen Sie auf der Seite Alias erstellen unter Routing-Konfiguration wie folgt vor:
  a. Wählen Sie für Version die erste Zustandsautomatenversion aus, auf die der Alias verweist.
  b. Aktivieren Sie das Kontrollkästchen Datenverkehr zwischen zwei Versionen aufteilen.

> **Tip**
>
> Um auf eine einzelne Version zu verweisen, deaktivieren Sie das Kontrollkästchen Datenverkehr zwischen zwei Versionen aufteilen.

c. Wählen Sie für Version die zweite Version aus, auf die der Alias verweisen muss.

d. Geben Sie in den Feldern Datenverkehr in Prozent den Prozentsatz des Datenverkehrs an, der an jede Version weitergeleitet werden soll. Geben Sie beispielsweise **`60`** und ein, **`40`** um 60 Prozent des Ausführungsdatenverkehrs an die erste Version und 40 Prozent des Datenverkehrs an die zweite Version weiterzuleiten.

   Die kombinierten Prozentsätze des Datenverkehrs müssen 100 Prozent entsprechen.

Ausführen eines Zustandsautomaten mit einem Alias (Konsole)

Sie können die Ausführung von Zustandsautomaten mit einem Alias über die Konsole oder durch Aufrufen der StartExecution API-Aktion mit dem ARN des Alias starten. Step Functions führt dann die durch den Alias angegebene Version aus. Wenn Sie beim Starten einer Zustandsautomaten-Ausführung keine Version oder keinen Alias angeben, verwendet Step Functions standardmäßig die neueste Revision.

So starten Sie die Ausführung eines Zustandsautomaten mit einem Alias

1. Öffnen Sie die Step-Functions-Konsole und wählen Sie dann einen vorhandenen Zustandsautomaten aus, für den Sie einen Alias erstellt haben. Informationen zum Erstellen eines Alias finden Sie unter Erstellen eines Zustandsautomaten-Alias (Konsole).
2. Wählen Sie auf der Detailseite des Zustandsautomaten die Registerkarte Aliasse aus.
3. Gehen Sie im Abschnitt Aliasse wie folgt vor:

   a. Wählen Sie den Alias aus, mit dem Sie die Ausführung starten möchten.

   b. Wählen Sie Start execution (Ausführung starten) aus.
4. (Optional) Geben Sie im Dialogfeld Ausführung starten einen Namen für die Ausführung ein.
5. Geben Sie bei Bedarf die Ausführungseingabe ein und wählen Sie dann Ausführung starten aus.

Autorisierung für Versionen und Aliase

Um Step Functions API-Aktionen mit einer Version oder einem Alias aufzurufen, benötigen Sie die entsprechenden Berechtigungen. Um eine Version oder einen Alias zum Aufrufen einer API-Aktion zu autorisieren, verwendet Step Functions den ARN der State Machine, anstatt den Versions-ARN oder Alias-ARN zu verwenden. Sie können die Berechtigungen für eine bestimmte Version oder einen bestimmten Alias auch einschränken. Weitere Informationen finden Sie unter Berechtigungen einschränken.

Sie können das folgende IAM-Richtlinienbeispiel für eine State-Maschine verwenden, die benannt ist, *myStateMachine* um die CreateStateMachineAliasAPI-Aktion aufzurufen, um einen State-Machine-Alias zu erstellen.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "states:CreateStateMachineAlias",
      "Resource": "arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine"
    }
  ]
}
```

Beachten Sie beim Festlegen von Berechtigungen, um den Zugriff auf API-Aktionen mithilfe von State-Machine-Versionen oder Aliasen zuzulassen oder zu verweigern, Folgendes:

- Wenn Sie den `publish` Parameter der UpdateStateMachineAPI-Aktionen CreateStateMachineund verwenden, um eine neue State-Machine-Version zu veröffentlichen, benötigen Sie auch die `ALLOW` Genehmigung für die PublishStateMachineVersionAPI-Aktion.
- Die DeleteStateMachineAPI-Aktion löscht alle Versionen und Aliase, die einer State-Maschine zugeordnet sind.

In diesem Thema

Berechtigungen für eine Version oder einen Alias einschränken

Sie können einen Qualifier verwenden, um die Autorisierungserlaubnis, die für eine Version oder einen Alias erforderlich ist, weiter einzuschränken. Ein Qualifier bezieht sich auf eine Versionsnummer oder einen Aliasnamen. Sie verwenden den Qualifier, um eine staatliche Maschine zu qualifizieren. Das folgende Beispiel ist ein State Machine-ARN, der einen PROD Alias mit dem Namen Qualifier verwendet.

```
arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine:PROD
```

Weitere Informationen zu qualifizierten und unqualifizierten ARNs finden Sie unter. Zuordnen von Ausführungen zu einer Version oder einem Alias

Sie schränken die Berechtigungen mithilfe des optionalen Kontextschlüssels ein, der `states:StateMachineQualifier` in der `Condition` Erklärung einer IAM-Richtlinie angegeben ist. Beispielsweise `myStateMachine` verweigert die folgende IAM-Richtlinie für eine State-Maschine mit dem Namen „as“ den Zugriff auf den Aufruf der DescribeStateMachineAPI-Aktion mit einem Alias namens „as“ PROD oder „the version“. 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"
          ]
        }
      }
    }
  ]
}
```

Die folgende Liste gibt die API-Aktionen an, für die Sie die Berechtigungen mit dem `StateMachineQualifier` Kontextschlüssel einschränken können.

- CreateStateMachineAlias
- DeleteStateMachineAlias
- DeleteStateMachineVersion
- DescribeStateMachine
- DescribeStateMachineAlias
- ListExecutions
- ListStateMachineAliases
- StartExecution
- StartSyncExecution
- UpdateStateMachineAlias

Zuordnen von Zustandsautomaten-Ausführungen zu einer Version oder einem Alias

Step Functions ordnet eine Ausführung einer Version oder einem Alias basierend auf dem Amazon-Ressourcennamen (ARN) zu, den Sie zum Aufrufen der StartExecution API-Aktion verwenden. Step Functions führt diese Aktion zum Startzeitpunkt der Ausführung aus.

Sie können eine Zustandsautomaten-Ausführung mit einem qualifizierten oder einem nicht qualifizierten ARN starten.

- Qualifizierter ARN – Bezieht sich auf einen Zustandsautomaten-ARN mit einer Versionsnummer oder einem Aliasnamen.

  Das folgende qualifizierte ARN-Beispiel bezieht sich auf die Version eines 3 Zustandsautomaten mit dem Namen `myStateMachine`.

  ```
  arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine:3
  ```

  Das folgende qualifizierte ARN-Beispiel bezieht sich auf einen Alias mit dem Namen eines `PROD` Zustandsautomaten mit dem Namen `myStateMachine`.

  ```
  arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine:PROD
  ```

- Unqualifizierter ARN – Bezieht sich auf einen Zustandsautomaten-ARN ohne Versionsnummer oder Aliasnamenssuffix.

```
arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine
```

Wenn sich Ihr qualifizierter ARN beispielsweise auf Version bezieht3, ordnet Step Functions die Ausführung dieser Version zu. Die Ausführung wird nicht mit Aliassen verknüpft, die auf die Version verweisen3.

Wenn sich Ihr qualifizierter ARN auf einen Alias bezieht, ordnet Step Functions die Ausführung diesem Alias und der Version zu, auf die der Alias verweist. Eine Ausführung kann nur einem Alias zugeordnet werden.

> **Note**
>
> Wenn Sie eine Ausführung mit einem unqualifizierten ARN starten, ordnet Step Functions diese Ausführung nicht einer Version zu, selbst wenn die Version denselben Zustandsautomaten verwendetrevision. Wenn Version 3 beispielsweise die neueste Revision verwendet, Sie jedoch eine Ausführung mit einem nicht qualifizierten ARN starten, ordnet Step Functions diese Ausführung nicht der Version 3 zu.

In diesem Thema

Anzeigen von Ausführungen, die mit einer Version oder einem Alias gestartet wurden

Step Functions bietet die folgenden Möglichkeiten, um die Ausführungen anzuzeigen, die mit einer Version oder einem Alias begonnen wurden:

Verwenden von API-Aktionen

Sie können alle Ausführungen anzeigen, die einer Version oder einem Alias zugeordnet sind, indem Sie die ListExecutions API-Aktionen DescribeExecution und aufrufen. Diese API-Aktionen geben den ARN der Version oder des Alias zurück, die/der zum Starten der Ausführung verwendet wurde. Diese Aktionen geben auch andere Details zurück, einschließlich Status und ARN der Ausführung.

Sie können auch einen Zustandsautomaten-Alias-ARN oder Versions-ARN angeben, um die Ausführungen aufzulisten, die einem bestimmten Alias oder einer bestimmten Version zugeordnet sind.

Die folgende Beispielantwort der ListExecutions -API-Aktion zeigt den ARN des Alias, der zum Starten einer Zustandsautomaten-Ausführung mit dem Namen verwendet wird*myFirstExecution*.

Der *kursive* Text im folgenden Codeausschnitt stellt ressourcenspezifische Informationen dar.

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

Verwenden der Step-Functions-Konsole

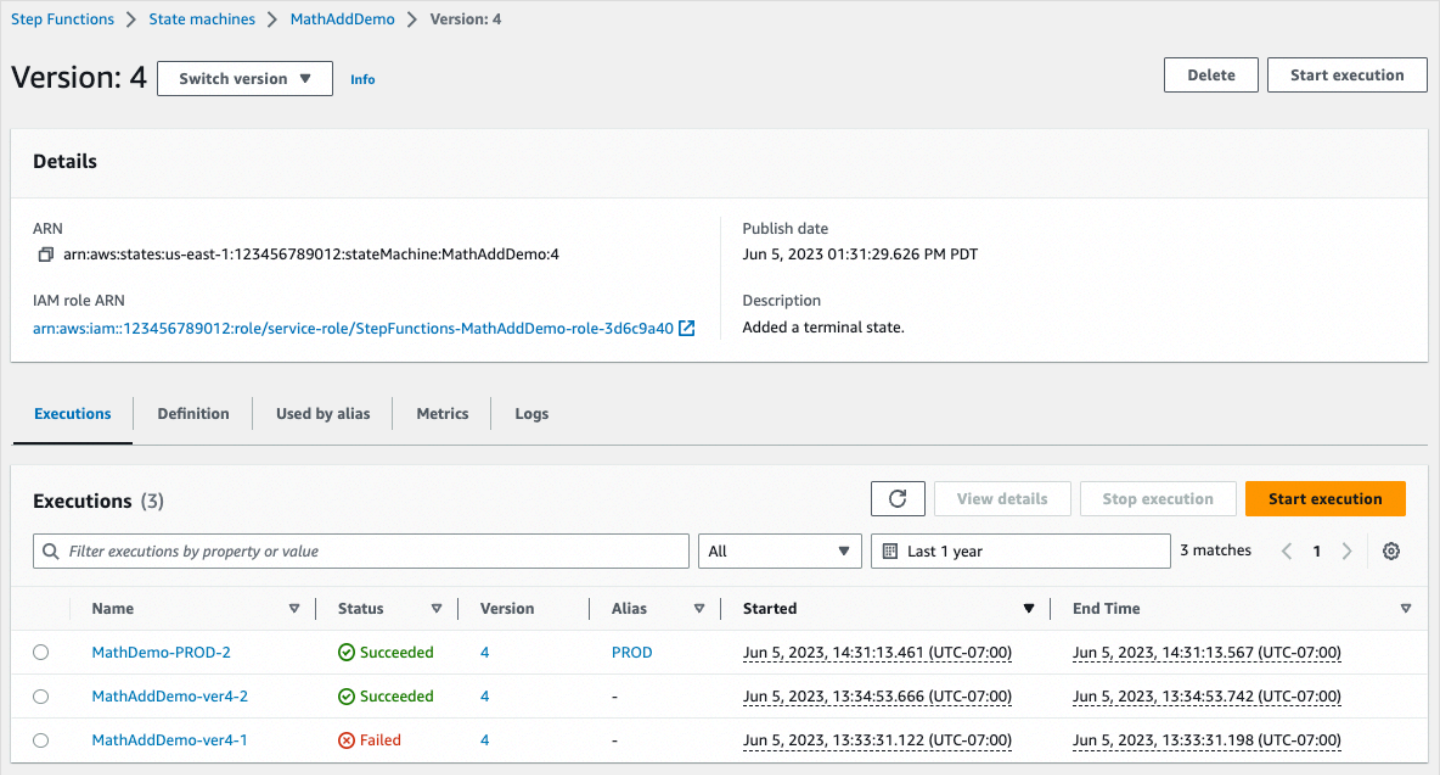
Sie können die von einer Version oder einem Alias gestarteten Ausführungen auch über die Step-Functions-Konsole anzeigen. Das folgende Verfahren zeigt, wie Sie die Ausführungen anzeigen können, die mit einer bestimmten Version gestartet wurden:

1. Öffnen Sie die Step-Functions-Konsole und wählen Sie dann einen vorhandenen Zustandsautomaten aus, für den Sie eine Version veröffentlicht oder einen Alias erstellt haben. Dieses Beispiel zeigt, wie Sie die Ausführungen anzeigen, die mit einer bestimmten Zustandsautomatenversion gestartet wurden.
2. Wählen Sie die Registerkarte Versionen und dann eine Version aus der Liste Versionen aus.

   > **Tip**
   >
   > Filtern Sie nach Eigenschaft oder Wertfeld, um nach einer bestimmten Version zu suchen.

3. Auf der Seite Versionsdetails finden Sie eine Liste aller laufenden und vergangenen Zustandsautomatenausführungen, die mit der ausgewählten Version gestartet wurden.

Die folgende Abbildung zeigt die Konsolenseite Versionsdetails. Auf dieser Seite werden Ausführungen aufgeführt, die von der Version 4 eines Zustandsautomaten mit dem Namen gestartet wurden*MathAddDemo*. Diese Liste zeigt auch eine Ausführung an, die von einem Alias namens gestartet wurde*PROD*. Dieser Alias leitete den Ausführungsdatenverkehr an Version 4 weiter.



Verwenden von CloudWatch-Metriken

Für jede Ausführung des Zustandsautomaten, die Sie mit einem beginnenQualified ARN, gibt Step Functions zusätzliche Metriken mit demselben Namen und Wert wie die Metriken aus, die derzeit ausgegeben werden. Diese zusätzlichen Metriken enthalten Dimensionen für jede der Versionskennungen und Aliasnamen, mit denen Sie eine Ausführung starten. Mit diesen Metriken können Sie die Ausführung von Zustandsautomaten auf Versionsebene überwachen und feststellen, wann ein Rollback-Szenario erforderlich sein könnte. Sie können auch Amazon CloudWatch-Alarme basierend auf diesen Metriken erstellen.

Step Functions gibt die folgenden Metriken für Ausführungen aus, die mit einem Alias oder einer Version beginnen:

- `ExecutionTime`
- `ExecutionsAborted`
- `ExecutionsFailed`

- `ExecutionsStarted`
- `ExecutionsSucceeded`
- `ExecutionsTimedOut`

Wenn Sie die Ausführung mit einem Versions-ARN gestartet haben, veröffentlicht Step Functions die Metrik mit der `StateMachineArn` und eine zweite Metrik mit den `Version` Dimensionen `StateMachineArn` und .

Wenn Sie die Ausführung mit einem Alias-ARN gestartet haben, gibt Step Functions die folgenden Metriken aus:

- Zwei Metriken für den nicht qualifizierten ARN und die Version.
- Eine Metrik mit den `Alias` Dimensionen `StateMachineArn` und .

Beispiel für die Bereitstellung von Alias und Versionen

Das folgende Beispiel für die Canary-Bereitstellungstechnik zeigt, wie Sie eine neue State-Machine-Version mit dem bereitstellen könnenAWS Command Line Interface. In diesem Beispiel leitet der von Ihnen erstellte Alias 20 Prozent des Ausführungsverkehrs an die neue Version weiter. Anschließend werden die verbleibenden 80 Prozent der früheren Version weitergeleitet. Gehen Sie wie folgt vor, um eine neue State-Machine-Version bereitzustellen und den Ausführungsverkehr mit einem Alias zu verlagern:

1. Veröffentlichen Sie eine Version der aktuellen State-Maschinenversion.

   Verwenden Sie den publish-state-machine-version Befehl in derAWS CLI, um eine Version aus der aktuellen Version einer State-Maschine mit dem Namen zu veröffentlichen *`myStateMachine`*:

   ```
   aws stepfunctions publish-state-machine-version --state-machine-arn
    arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine
   ```

   Die Antwort gibt die `stateMachineVersionArn` Version zurück, die Sie veröffentlicht haben. Zum Beispiel `arn:aws:states:us-east-1:`*`123456789012`*`:stateMachine:`*`myStateMachine`*`:1`.
2. Erstellen Sie einen Alias, der auf die State-Machine-Version verweist.

Verwenden Sie den create-state-machine-alias Befehl, um einen Alias mit dem Namen zu erstellen*PROD*, der auf Version 1 verweist von*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. Stellen Sie sicher, dass die vom Alias gestarteten Ausführungen die richtige veröffentlichte Version verwenden.

   Starten Sie eine neue Ausführung von, *myStateMachine* indem Sie den ARN des Alias **PROD** im start-execution Befehl angeben:

   ```
   aws stepfunctions start-execution
     --state-machine-arn arn:aws:states:us-
   east-1:123456789012:stateMachineAlias:myStateMachine:PROD
     --input "{}"
   ```

   Wenn Sie den State-Machine-ARN in der StartExecutionAnfrage angeben, verwendet diese die neueste Version revision der State Machine anstelle der in Ihrem Alias angegebenen Version, um die Ausführung zu starten.

4. Aktualisieren Sie die State-Machine-Definition und veröffentlichen Sie eine neue Version.

   Aktualisiere *myStateMachine* und veröffentliche die neue Version. Verwenden Sie dazu den optionalen `publish` Parameter des update-state-machine Befehls:

   ```
   aws stepfunctions update-state-machine
       --state-machine-arn arn:aws:states:us-
   east-1:123456789012:stateMachine:myStateMachine
       --definition $UPDATED_STATE_MACHINE_DEFINITION
       --publish
   ```

   Die Antwort gibt den `stateMachineVersionArn` für die neue Version zurück. Zum Beispiel `arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine:2`.

5. Aktualisieren Sie den Alias so, dass er auf beide Versionen verweist, und legen Sie die Routing-Konfiguration des Alias fest.

Verwenden Sie den update-state-machine-alias Befehl, um die Routing-Konfiguration des Alias zu aktualisierenPROD. Konfigurieren Sie den Alias so, dass 80 Prozent des Ausführungsdatenverkehrs an Version 1 und die restlichen 20 Prozent an Version 2 gehen:

```
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. Ersetzen Sie Version 1 durch Version 2.

   Nachdem Sie überprüft haben, dass Ihre neue State-Machine-Version ordnungsgemäß funktioniert, können Sie die neue State-Machine-Version bereitstellen. Aktualisieren Sie dazu den Alias erneut, um der neuen Version 100 Prozent des Ausführungsverkehrs zuzuweisen.

   Verwenden Sie den update-state-machine-alias Befehl, um die Routing-Konfiguration des Alias für Version 2 PROD auf 100 Prozent festzulegen:

```
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**
>
> Um die Bereitstellung von Version 2 rückgängig zu machen, bearbeiten Sie die Routing-Konfiguration des Alias so, dass 100 Prozent des Datenverkehrs auf die neu bereitgestellte Version umgeleitet werden.
>
> ```
> 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}]"
> ```

Sie können Versionen und Aliase verwenden, um andere Arten von Bereitstellungen durchzuführen. Sie können beispielsweise eine fortlaufende Bereitstellung einer neuen Version Ihrer State Machine durchführen. Erhöhen Sie dazu schrittweise den gewichteten Prozentsatz in der Routing-Konfiguration des Alias, der auf die neue Version verweist.

Sie können auch Versionen und Aliase verwenden, um eine blaue/grüne Bereitstellung durchzuführen. Erstellen Sie dazu einen Alias mit dem Namen`green`, auf dem die aktuelle Version 1 Ihres State-Computers ausgeführt wird. Erstellen Sie dann einen weiteren Alias mit dem Namen`blue`, auf dem die neue Version ausgeführt wird, *2* z. B. Um die neue Version zu testen, senden Sie Ausführungsverkehr an den `blue` Alias. Wenn Sie sicher sind, dass Ihre neue Version ordnungsgemäß funktioniert, aktualisieren Sie den `green` Alias so, dass er auf Ihre neue Version verweist.

Führen Sie die schrittweise Bereitstellung von State-Machine-Versionen durch

Eine fortlaufende Bereitstellung ist eine Bereitstellungsstrategie, bei der frühere Versionen einer Anwendung langsam durch neue Versionen einer Anwendung ersetzt werden. Um eine fortlaufende Bereitstellung einer State-Machine-Version durchzuführen, senden Sie schrittweise eine zunehmende Menge an Ausführungsdatenverkehr an die neue Version. Die Menge des Traffics und die Steigerungsrate sind Parameter, die Sie konfigurieren.

Sie können eine Version fortlaufend bereitstellen, indem Sie eine der folgenden Optionen verwenden:

- Step Functions Konsole— Erstellen Sie einen Alias, der auf zwei Versionen derselben State-Maschine verweist. Für diesen Alias konfigurieren Sie die Routing-Konfiguration, um den Verkehr zwischen den beiden Versionen zu verteilen. Weitere Informationen zur Verwendung der Konsole zum Rollout von Versionen finden Sie unterVersionenundAliasnamen.
- Skripte fürAWS CLIund SDK— Erstellen Sie ein Shell-Skript mit demAWS CLIoder derAWSSDK. Weitere Informationen finden Sie in den folgenden Abschnitten zur VerwendungAWS CLIundAWSSDK.
- AWS CloudFormationVorlagen— Benutze die`AWS::StepFunctions::StateMachineVersion`und`AWS::StepFunctions::StateMachineAl` um mehrere State-Machine-Versionen zu veröffentlichen und einen Alias zu erstellen, der auf eine oder zwei dieser Versionen verweist.

Benutze dieAWS CLIum eine neue State-Machine-Version bereitzustellen

Das Beispielskript in diesem Abschnitt zeigt, wie Sie das verwenden könnenAWS CLIum den Verkehr schrittweise von einer früheren State-Machine-Version auf eine neue State-Machine-Version zu verlagern. Sie können dieses Beispielskript entweder verwenden oder es entsprechend Ihren Anforderungen aktualisieren.

Dieses Skript zeigt ein Canary-Deployment zur Bereitstellung einer neuen State-Machine-Version unter Verwendung eines Alias. In den folgenden Schritten werden die Aufgaben beschrieben, die das Skript ausführt:

1. Wenn der`publish_revision`Der Parameter ist auf True gesetzt, veröffentliche den neuestenrevisionals nächste Version der Staatsmaschine. Diese Version wird zur neuen Live-Version, wenn die Bereitstellung erfolgreich ist.

   Wenn du das einstellst`publish_revision`Wenn der Parameter auf false gesetzt ist, stellt das Skript die letzte veröffentlichte Version der State Machine bereit.
2. Erstellen Sie einen Alias, falls er noch nicht existiert. Wenn der Alias nicht existiert, leiten Sie 100 Prozent des Traffics für diesen Alias auf die neue Version weiter und beenden Sie dann das Skript.
3. Aktualisieren Sie die Routing-Konfiguration des Alias, um einen kleinen Prozentsatz des Datenverkehrs von der vorherigen Version auf die neue Version zu verlagern. Du legst diesen kanarischen Prozentsatz fest mit dem`canary_percentage`Parameter.
4. Überwachen Sie standardmäßig das konfigurierbareCloudWatchAlarme alle 60 Sekunden. Wenn einer dieser Alarme ausgelöst wird, machen Sie das Deployment sofort rückgängig, indem Sie 100 Prozent des Datenverkehrs auf die vorherige Version umleiten.

   Nach jedem Zeitintervall, in Sekunden, definiert in`alarm_polling_interval`, setzen Sie die Überwachung der Alarme fort. Fahren Sie mit der Überwachung fort, bis das in definierte Zeitintervall erreicht ist`canary_interval_seconds`ist vergangen.
5. Wenn währenddessen keine Alarme ausgelöst wurden`canary_interval_seconds`, verlagern Sie 100 Prozent des Traffics auf die neue Version.
6. Wenn die neue Version erfolgreich bereitgestellt wird, löschen Sie alle Versionen, die älter sind als die im`history_max`Parameter.

```
#!/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!
```

Benutze dieAWSSDK zur Bereitstellung einer neuen State Machine-Version

Das Beispielskript unteraws-stepfunctions-exampleszeigt, wie man das benutztAWSSDK für Python zur schrittweisen Verlagerung des Datenverkehrs von einer früheren Version auf eine neue Version einer State-Maschine. Sie können dieses Beispielskript entweder verwenden oder es entsprechend Ihren Anforderungen aktualisieren.

Das Skript zeigt die folgenden Bereitstellungsstrategien:

- Kanarier— Verschiebt den Verkehr in zwei Schritten.

  In der ersten Stufe wird ein kleiner Prozentsatz des Traffics, beispielsweise 10 Prozent, auf die neue Version umgestellt. Im zweiten Schritt, bevor ein bestimmtes Zeitintervall in Sekunden abgelaufen ist, wird der verbleibende Datenverkehr auf die neue Version umgestellt. Die Umstellung auf die neue Version für den verbleibenden Traffic erfolgt nur, wenn neinCloudWatchAlarme werden während des angegebenen Zeitintervalls ausgelöst.
- Linear oder Rollend— Verschiebt den Datenverkehr in gleichen Schritten auf die neue Version, wobei zwischen den einzelnen Schritten die gleiche Anzahl von Sekunden liegt.

  Wenn Sie zum Beispiel das Inkrement in Prozent angeben als**20**mit einem`--intervalvon`**`600`**Sekunden, diese Bereitstellung erhöht den Traffic alle 600 Sekunden um 20 Prozent, bis die neue Version 100 Prozent des Datenverkehrs empfängt.

  Bei dieser Bereitstellung wird die neue Version, falls vorhanden, sofort rückgängig gemacht.CloudWatchAlarme werden ausgelöst.
- Alles auf einmal oder Blau/Grün— Leitet sofort 100 Prozent des Traffics auf die neue Version um. Dieses Deployment überwacht die neue Version und setzt sie, falls vorhanden, automatisch auf die vorherige Version zurück.CloudWatchAlarme werden ausgelöst.

BenutzenAWS CloudFormationum eine neue State-Machine-Version bereitzustellen

Das FolgendeCloudFormationIn einem Vorlagenbeispiel werden zwei Versionen einer State-Maschine mit dem Namen veröffentlicht*`MyStateMachine`*. Es erstellt einen Alias mit dem Namen*`PROD`*, das auf diese beiden Versionen verweist und dann die Version bereitstellt2.

In diesem Beispiel werden 10 Prozent des Traffics auf die Version umgestellt2alle fünf Minuten, bis diese Version 100 Prozent des Traffics erhält. Dieses Beispiel zeigt auch, wie Sie einstellen könnenCloudWatchAlarme. Wenn einer der von Ihnen eingestellten Alarme in den`ALARM`In diesem Zustand schlägt die Bereitstellung fehl und es wird sofort ein Rollback durchgeführt.

```
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
```

Ausführungen in Step Functions

Ein Zustandsautomat wird ausgeführt, wenn ein AWS Step Functions-Zustandsautomat ausgeführt wird und seine Aufgaben ausführt. Jede Step Functions Functions-Zustandsmaschine kann mehrere gleichzeitige Ausführungen haben, die Sie über die Step Functions Functions-Konsole oder mithilfe der AWS SDKs, der Step Functions Functions-API-Aktionen oder der AWS Command Line Interface () AWS CLI initiieren können. Eine Ausführung erhält JSON-Eingabe und erstellt JSON-Ausgabe. Sie können eine Step Functions Functions-Ausführung auf folgende Arten starten:

- Rufen Sie die StartExecution-API-Aktion auf.
- Starten Sie eine neue Ausführung in der Step Functions Functions-Konsole.
- Verwenden Sie Amazon EventBridge , um eine Ausführung als Reaktion auf ein Ereignis zu starten.
- Wird verwendetAmazon EventBridge Scheduler, um eine State-Machine-Ausführung nach einem Zeitplan zu starten.
- Starten Sie eine Ausführung mit Amazon API Gateway.
- Starten Sie eine verschachtelte Workflow-Ausführung von einem Task-Status aus.

Weitere Informationen zu den verschiedenen Arten der Arbeit mit Step Functions finden Sie unter Entwicklungsoptionen.

Starten von Workflow-Ausführungen über einen Aufgabenstatus

AWS Step Functions kann Workflow-Ausführungen aus einem `Task`-Zustand eines Zustandsautomaten starten. Auf diese Weise können Sie Ihre Workflows in kleinere Zustandsautomaten aufteilen und Ausführungen dieser anderen Zustandsautomaten starten. Durch das Starten dieser neuen Workflow-Ausführungen haben Sie folgenden Möglichkeiten:

- Trennen Sie den Workflow auf höherer Ebene von aufgabenspezifischen Workflows auf unterer Ebene.
- Vermeiden Sie sich wiederholende Elemente. Rufen Sie nicht mehrfach einen separaten Zustandsautomaten auf.
- Erstellen Sie eine Bibliothek mit modularen wiederverwendbaren Workflows für eine schnellere Entwicklung.
- Reduzieren Sie die Komplexität und vereinfachen Sie die Bearbeitung und Fehlerbehebung von Zustandsautomaten.

Step Functions kann diese Workflow-Ausführungen starten, indem es seine eigene API als integrierten Dienst aufruft. Rufen Sie einfach die `StartExecution`-API-Aktion aus Ihrem `Task`-Status auf und übergeben Sie die erforderlichen Parameter. Sie können die Step Functions API mithilfe eines beliebigen Dienstintegrationsmusters aufrufen.

> **Tip**
>
> Ein Beispiel für einen verschachtelten Workflow finden Sie AWS-Konto in Modul 13 — Verschachtelte Express-Workflows.

Um eine neue Ausführung einer State-Maschine zu starten, verwenden Sie einen `Task` Zustand, der dem folgenden Beispiel ähnelt:

```
{
   "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
}
```

Dieser `Task`-Status startet eine neue Ausführung des `HelloWorld`-Zustandsautomaten und übergibt den JSON-Kommentar als Eingabe.

> **Note**
>
> Die `StartExecution`-API-Aktionskontingente können die Anzahl der Ausführungen begrenzen, die Sie starten können. Verwenden Sie die `Retry` für

`StepFunctions.ExecutionLimitExceeded`, um sicherzustellen, dass Ihre Ausführung gestartet wird. Siehe unten.

- Kontingente im Zusammenhang mit der Drosselung von API-Aktionen
- Fehlerbehandlung in Step Functions

Zuordnen von Workflow-Ausführungen

Um eine gestartete Workflow-Ausführung mit der Ausführung zu verknüpfen, durch die sie gestartet wurde, übergeben Sie die Ausführungs-ID aus dem Context-Objekt an die Ausführungseingabe. Sie können aus Ihrem `Task`-Status in einer laufenden Ausführung auf die ID aus dem Context-Objekt zugreifen. Übergeben Sie die Ausführungs-ID, indem Sie `.$` an den Parameternamen anhängen und mit `$$.Execution.Id` auf die ID im Context-Objekt mit verweisen.

```
"AWS_STEP_FUNCTIONS_STARTED_BY_EXECUTION_ID.$": "$$.Execution.Id"
```

Sie können einen speziellen Parameter namens `AWS_STEP_FUNCTIONS_STARTED_BY_EXECUTION_ID` verwenden, wenn Sie eine Ausführung starten. Falls vorhanden, enthält diese Verknüpfung Links im Abschnitt mit den Schrittdetails der Step Functions-Konsole. Sofern bereitgestellt, können Sie problemlos die Ausführungen Ihrer Workflows verfolgen, vom Starten von Ausführungen bis zu den gestarteten Workflow-Ausführungen. Verknüpfen Sie im vorherigen Beispiel die Ausführungs-ID wie folgt mit der gestarteten Ausführung des `HelloWorld`-Zustandsautomaten.

```
{
   "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
}
```

Weitere Informationen finden Sie hier:

- Arbeiten mit anderen -Services
- Parameter an eine Service-API übergeben
- Zugriff auf das Kontext-Objekt
- AWS Step Functions

Verwenden von Amazon EventBridge Scheduler mit AWS Step Functions

Amazon EventBridge Scheduler ist ein serverloser Scheduler, mit dem Sie Aufgaben von einem zentralen, verwalteten Service aus erstellen, ausführen und verwalten können. Mit EventBridge Scheduler können Sie Zeitpläne mithilfe von Cron- und Rate-Ausdrücken für wiederkehrende Muster erstellen oder einmalige Aufrufe konfigurieren. Sie können flexible Zeitfenster für die Zustellung einrichten, Wiederholungslimits definieren und die maximale Aufbewahrungszeit für fehlgeschlagene API-Aufrufe festlegen.

Mit EventBridge Scheduler können Sie beispielsweise eine State-Machine-Ausführung nach einem Zeitplan starten, wenn ein sicherheitsrelevantes Ereignis eintritt, oder um einen Datenverarbeitungsjob zu automatisieren.

Auf dieser Seite wird erklärt, wie Sie den EventBridge Scheduler verwenden, um die Ausführung einer Step Functions Functions-Zustandsmaschine nach einem Zeitplan zu starten.

Themen

- Einrichten der Ausführungsrolle
- Erstellen eines Zeitplans
- Zugehörige Ressourcen

Einrichten der Ausführungsrolle

Wenn Sie einen neuen Zeitplan erstellen, muss der EventBridge Scheduler über die Berechtigung verfügen, seinen API-Zielvorgang in Ihrem Namen aufzurufen. Sie gewähren EventBridge Scheduler diese Berechtigungen mithilfe einer Ausführungsrolle. Die Berechtigungsrichtlinie, die Sie der Ausführungsrolle Ihres Zeitplans hinzufügen, definiert die erforderlichen Berechtigungen. Diese Berechtigungen hängen von der Ziel-API ab, die EventBridge Scheduler aufrufen soll.

Wenn Sie die EventBridge Scheduler-Konsole verwenden, um einen Zeitplan zu erstellen, wie im folgenden Verfahren beschrieben, richtet EventBridge Scheduler automatisch eine Ausführungsrolle auf der Grundlage Ihres ausgewählten Ziels ein. Wenn Sie einen Zeitplan mit einem der EventBridge Scheduler-SDKs, dem, oder erstellen möchten, müssen Sie über eine vorhandene Ausführungsrolle verfügenAWS CloudFormation, die die AWS CLI Berechtigungen gewährt, die EventBridge Scheduler zum Aufrufen eines Ziels benötigt. Weitere Informationen zum manuellen Einrichten einer Ausführungsrolle für Ihren Zeitplan finden Sie unter Einrichten einer Ausführungsrolle im EventBridge Scheduler-Benutzerhandbuch.

Erstellen eines Zeitplans

So erstellen Sie einen Zeitplan mithilfe der Konsole

1. Öffnen Sie die Amazon EventBridge Scheduler-Konsole unter https://console.aws.amazon.com/scheduler/home.
2. Wählen Sie auf der Seite Zeitpläne die Option Zeitplan erstellen aus.
3. Gehen Sie auf der Seite Zeitplandetails angeben im Abschnitt Zeitplanname und -beschreibung wie folgt vor:

   a. Geben Sie unter Zeitplanname einen Namen für Ihren Zeitplan ein. Zum Beispiel **`MyTestSchedule`**.

   b. (Optional) Geben Sie unter Beschreibung eine Beschreibung für Ihren Zeitplan ein. Zum Beispiel **`My first schedule`**.

   c. Wählen Sie für Zeitplangruppe eine Zeitplangruppe aus der Dropdown-Liste aus. Wenn Sie noch keine Gruppe haben, wählen Sie Standard. Um eine Zeitplangruppe zu erstellen, wählen Sie Eigenen Zeitplan erstellen.

   Sie verwenden Zeitplangruppen, um Tags zu Zeitplangruppen hinzuzufügen.

4. • Wählen Sie Ihre Zeitplanoptionen.

Vorkommen	Vorgehensweise	
Einmaliger Zeitplan Ein einmaliger Zeitplan ruft ein Ziel nur einmal zu dem von Ihnen angegebenen	Gehen Sie für Datum und Uhrzeit wie folgt vor:	

Vorkommen	Vorgehensweise	
Datum und der angegeben en Uhrzeit auf.	• Geben Sie ein gültiges Datum im `YYYY/MM/DD` -Format ein. • Geben Sie einen Zeitstempel im 24-Stunden-Format (`hh:mm`) ein. • Wählen Sie unter Zeitzone die Zeitzone aus.	

Vorkommen	Vorgehensweise	
Wiederkehrender Zeitplan Ein wiederkehrender Zeitplan ruft ein Ziel mit einer Rate auf, die Sie mit einem cron-Ausdruck oder einem Rate-Ausdruck angeben.	a. Gehen Sie bei Zeitplantyp wie folgt vor: • Um den Zeitplan mithilfe eines Cron-Ausdrucks zu definieren, wählen Sie Cron-basierter Zeitplan und geben Sie den Cron-Ausdruck ein. • Um den Zeitplan mithilfe eines Rate-Ausdrucks zu definieren, wählen Sie Rate-Ausdruck ein. Weitere Informationen zu Cron- und Rate-Ausdrücken finden Sie unter Zeitplantypen auf EventBridge Scheduler im Amazon EventBridge Scheduler-Benutzerhandbuch. b. Wählen Sie für Flexibles Zeitfenster die Option Aus, um die Option zu deaktivieren, oder wählen Sie eines der vordefinierten Zeitfenster aus. Wenn Sie beispielsweise 15 Minuten auswählen und einen wiederkehrenden	

Vorkommen	Vorgehensweise	
	Zeitplan festlegen, der sein Ziel einmal pro Stunde aufruft, wird der Zeitplan innerhalb von 15 Minuten nach Beginn jeder Stunde ausgeführt.	

5. (Optional) Wenn Sie im vorherigen Schritt Wiederkehrender Zeitplan ausgewählt haben, gehen Sie im Abschnitt Zeitrahmen wie folgt vor:

   a. Wählen Sie unter Zeitzone eine Zeitzone aus.

   b. Geben Sie für Startdatum und -uhrzeit ein gültiges Datum im `YYYY/MM/DD`-Format ein und geben Sie dann einen Zeitstempel im 24-Stunden-Format (`hh:mm`) an.

   c. Geben Sie für Enddatum und -uhrzeit ein gültiges Datum im `YYYY/MM/DD`-Format ein und geben Sie dann einen Zeitstempel im 24-Stunden-Format (`hh:mm`) an.

6. Wählen Sie Weiter aus.

7. Wählen Sie auf der Seite „Ziel auswählen“ den AWS API-Vorgang aus, den der Scheduler aufruft EventBridge :

   a. Wählen Sie. AWS Step Functions StartExecution

   b. Wählen Sie in dem StartExecutionAbschnitt einen Zustandsmaschine aus oder wählen Sie Neuen Zustandsmaschine erstellen.

      Derzeit können Sie Synchronous Express-Workflows nicht nach einem Zeitplan ausführen.

   c. Geben Sie eine JSON-Payload für die Ausführung ein. Auch wenn Ihre Zustandsmaschine keine JSON-Nutzlast benötigt, müssen Sie dennoch Eingaben im JSON-Format hinzufügen, wie im folgenden Beispiel gezeigt.

      ```
      {
          "Comment": "sampleJSONData"
      }
      ```

8. Wählen Sie Weiter aus.

9. Führen Sie auf der Seite Settings (Einstellungen) die folgenden Schritte aus:

a. Um den Zeitplan zu aktivieren, schalten Sie unter Zeitplanstatus die Option Zeitplan aktivieren ein.

b. Um eine Wiederholungsrichtlinie für Ihren Zeitplan zu konfigurieren, gehen Sie unter Wiederholungsrichtlinie und Warteschlange für unzustellbare Nachrichten (DLQ) wie folgt vor:

   - Aktivieren Sie die Option Wiederholen.
   - Geben Sie unter Höchstalter des Ereignisses die maximale (n) Stunde (n) und Minute (n) ein, für die der EventBridge Scheduler ein unbearbeitetes Ereignis speichern darf.
   - Die Höchstdauer beträgt 24 Stunden.
   - Geben Sie unter Maximale Anzahl an Wiederholungen ein, wie oft der EventBridge Scheduler den Zeitplan maximal wiederholt, wenn das Ziel einen Fehler zurückgibt.

     Der Maximalwert beträgt 185 Wiederholungen.

   Bei Wiederholungsrichtlinien führt der Scheduler den Zeitplan erneut aus, wenn ein Zeitplan sein Ziel nicht aufrufen kann. EventBridge Falls konfiguriert, müssen Sie die maximale Aufbewahrungszeit und Wiederholungsversuche für den Zeitplan festlegen.

c. Wählen Sie aus, wo der EventBridge Scheduler nicht zugestellte Ereignisse speichert.

Option für Warteschlange für unzustellbare Nachrichten (DLQ)	Vorgehensweise	
Nicht speichern	Wählen Sie None.	

Option für Warteschlange für unzustellbare Nachrichten (DLQ)	Vorgehensweise	
Speichert das Ereignis im selben AWS-Konto, in dem Sie den Zeitplan erstellen	a. Wählen Sie Eine Amazon-SQS-Warteschlange in meinem AWS-Konto als DLQ auswählen. b. Wählen Sie den Amazon-Ressourcennamen (ARN) der Amazon SQS-Warteschlange.	
Speichert das Ereignis in einem anderen AWS-Konto als dem, in dem Sie den Zeitplan erstellen	a. Wählen Sie Angeben einer Amazon-SQS-Warteschlange in anderen AWS-Konten als DLQ. b. Geben Sie den Amazon-Ressourcennamen (ARN) der Amazon-SQS-Warteschlange ein.	

d. Um einen kundenverwalteten Schlüssel zur Verschlüsselung Ihrer Zieleingabe zu verwenden, wählen Sie unter Verschlüsselung die Option Verschlüsselungseinstellungen anpassen (erweitert).

   Wenn Sie diese Option wählen, geben Sie einen vorhandenen CMK-ARN ein oder wählen Sie Erstellen eines AWS KMS key, um zur AWS KMS-Konsole zu navigieren. Weitere Informationen darüber, wie EventBridge Scheduler Ihre Daten im Ruhezustand verschlüsselt, finden Sie unter Verschlüsselung im Ruhezustand im Amazon Scheduler-Benutzerhandbuch. EventBridge

e. Um EventBridge Scheduler eine neue Ausführungsrolle für Sie erstellen zu lassen, wählen Sie Neue Rolle für diesen Zeitplan erstellen. Geben Sie dann einen Namen

für Rollenname ein. Wenn Sie diese Option wählen, ordnet EventBridge Scheduler der Rolle die erforderlichen Berechtigungen zu, die für Ihr vorgegebenes Ziel erforderlich sind.

10. Wählen Sie Weiter aus.
11. Überprüfen Sie auf der Seite Zeitplan überprüfen und erstellen die Details Ihres Zeitplans. Wählen Sie in jedem Abschnitt Bearbeiten aus, um zu diesem Schritt zurückzukehren und seine Details zu bearbeiten.
12. Wählen Sie Zeitplan erstellen.

    Auf der Seite Zeitpläne können Sie eine Liste Ihrer neuen und vorhandenen Zeitpläne anzeigen. Überprüfen Sie in der Spalte Status, ob Ihr neuer Zeitplan aktiviert ist.

Um zu überprüfen, ob der EventBridge Scheduler den Zustandsmaschine aufgerufen hat, überprüfen Sie die Amazon-Protokolle des Zustandsmaschinen. CloudWatch

Zugehörige Ressourcen

Weitere Informationen zu EventBridge Scheduler finden Sie im Folgenden:

- EventBridge Scheduler-Benutzerhandbuch
- EventBridge Scheduler-API-Referenz
- EventBridge Preise für Scheduler

Standard- und Express-Workflow-Ausführungen in der Konsole

Wenn Sie einen Zustandsautomaten erstellen, wählen Sie entweder den Typ Standard oder Express aus. Der Standardtyp für Zustandsautomaten ist Standard . Ein Zustandsautomat, dessen Typ Standard ist, wird als Standard-Workflow bezeichnet, und ein Zustandsautomat, dessen Typ Express ist, wird als Express-Workflow bezeichnet.

Sowohl für Standard- als auch für Express-Workflows definieren Sie Ihren Zustandsautomaten mithilfe der Amazon States Language. Ihre Ausführungen von Zustandsautomaten verhalten sich je nach ausgewähltem Typ unterschiedlich.

> **Important**
>
> Der von Ihnen gewählte Typ kann nicht geändert werden, nachdem Sie den Zustandsautomaten erstellt haben.

Weitere Informationen zu Standard- und Express-Workflows finden Sie unter Standard- und Express-Workflows.

Der Verlauf der Standard-Workflow-Ausführungen wird in Step Functions aufgezeichnet, während der Verlauf der Express-Workflow-Ausführungen nicht in Step Functions aufgezeichnet wird. Um den Verlauf einer Express-Workflow-Ausführung aufzuzeichnen, müssen Sie sie so konfigurieren, dass Protokolle an Amazon gesendet werden CloudWatch. Weitere Informationen finden Sie unter Protokollierung mitCloudWatchProtokolle.

Sobald die Protokollierung für einen Express-Workflow konfiguriert ist, können Sie ihre Ausführungen in der Step Functions-Konsole anzeigen. Die Konsolenerfahrung zum Anzeigen von Express-Workflow-Ausführungen und Standard-Workflow-Ausführungen ist ähnlich, mit Ausnahme der folgenden Unterschiede und Einschränkungen.

> **Note**
>
> Da Ausführungsdaten für Express-Workflows mit CloudWatch Logs Insights angezeigt werden, fallen beim Scannen der Protokolle Gebühren an. Standardmäßig listet Ihre Protokollgruppe nur Ausführungen auf, die in den letzten drei Stunden abgeschlossen wurden. Wenn Sie einen größeren Zeitraum angeben, der mehr Ausführungsereignisse umfasst, steigen Ihre Kosten. Weitere Informationen finden Sie unter Vended Logs auf der Registerkarte Protokolle auf der CloudWatch Seite Preise und Protokollierung mitCloudWatchProtokolle.

Inhalt

Unterschiede in der Konsole

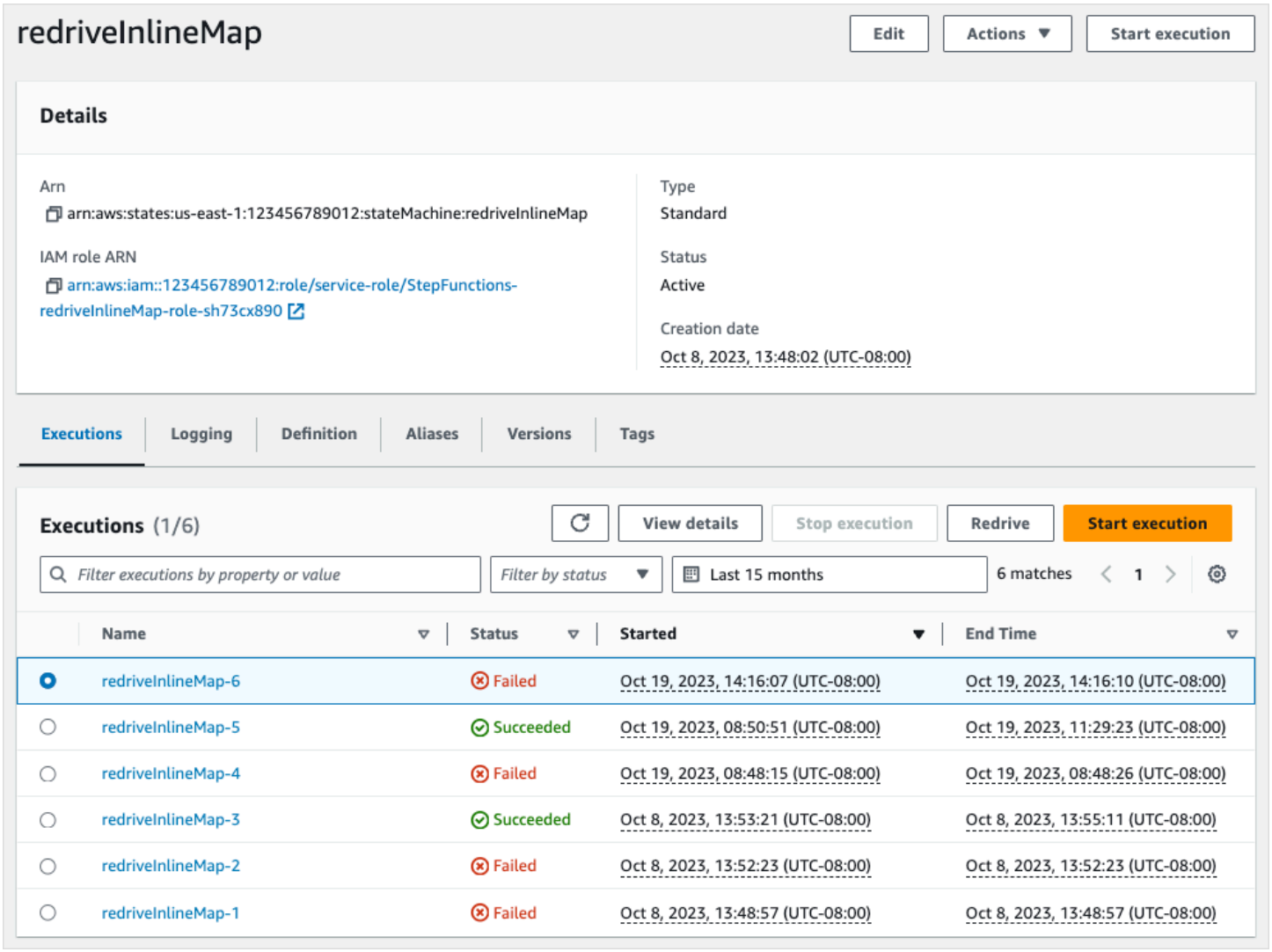
Für alle Standard- und Express-Workflows können Sie Details wie den Zustandsautomaten und seinen IAM-Rollen-ARN auf der Detailseite des Zustandsautomaten in der Step-Functions-Konsole anzeigen.

Auf der Detailseite des Zustandsautomaten finden Sie auch eine Liste der Ausführungsverläufe Ihres Zustandsautomaten auf der Registerkarte Ausführungen. Verwenden Sie das Feld Ausführungen nach Eigenschaft oder Wert filtern, um nach einer bestimmten Ausführung, Version oder einem Alias

des ausgewählten Zustandsautomaten zu suchen. Verwenden Sie das Dropdown-Menü Alle, um Ausführungsverläufe nach ihrem Status zu filtern. Sie können auch einen Ausführungsverlauf und die Schaltfläche Details anzeigen auswählen, um die Seite Ausführungsdetails zu öffnen.

Standard-Workflows

Die Ausführungsverläufe für Standard-Workflows sind immer für Ausführungen verfügbar, die in den letzten 90 Tagen abgeschlossen wurden.



Express-Workflows

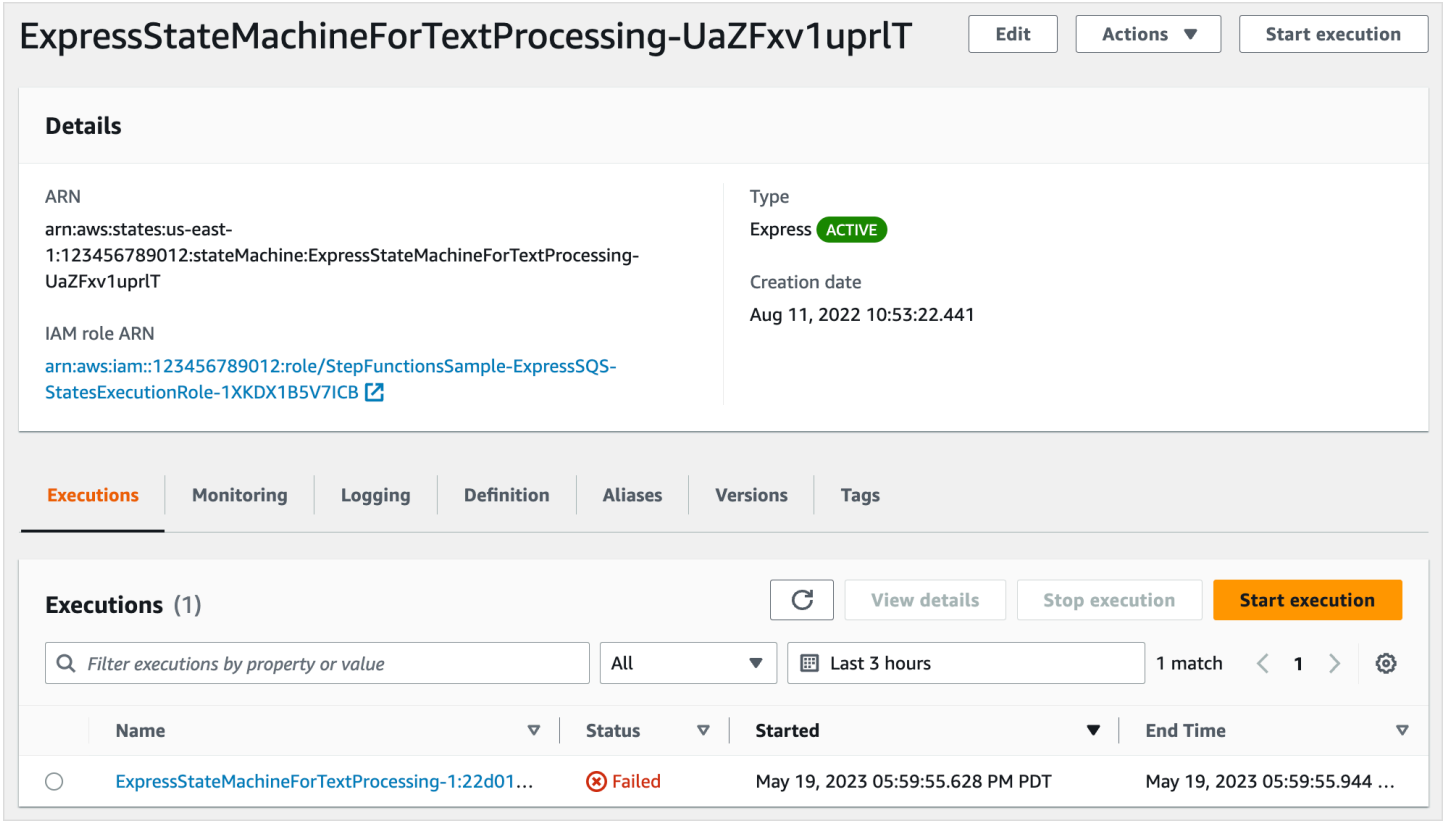
Um den Ausführungsverlauf für Express-Workflows anzuzeigen, ruft die Step-Functions-Konsole Protokolldaten ab, die über eine CloudWatch Protokollgruppe gesammelt wurden.

Sie müssen auch die neue Konsolenerfahrung aktivieren, um Express-Workflow-Ausführungen anzuzeigen. Wählen Sie dazu die Schaltfläche Aktivieren, die im Banner auf der Registerkarte

Ausführungen angezeigt wird. Sobald Sie diese Schaltfläche ausgewählt haben, wird sie nicht mehr angezeigt.

> **Tip**
>
> Um zwischen dem Aktivieren oder Deaktivieren der Konsolenerfahrung zu wechseln, verwenden Sie die Schaltfläche Express-Ausführungsverlauf aktivieren.

Die Verläufe für Ausführungen, die in den letzten drei Stunden abgeschlossen wurden, sind standardmäßig verfügbar. Sie können diesen Zeitraum anpassen oder einen benutzerdefinierten Bereich angeben. Wenn Sie einen größeren Zeitraum angeben, der mehr Ausführungsereignisse enthält, steigen die Kosten für das Scannen der Protokolle. Weitere Informationen finden Sie unter Vended Logs auf der Registerkarte Protokolle auf der CloudWatch Seite Preise und Protokollierung mitCloudWatchProtokolle.



Überlegungen und Einschränkungen für die Anzeige von Express-Workflow-Ausführungen

Beachten Sie beim Anzeigen von Express-Workflow-Ausführungen in der Step-Functions-Konsole die folgenden Überlegungen und Einschränkungen.

- Die Verfügbarkeit von Express-Workflow-Ausführungsdetails hängt von Amazon CloudWatch Logs ab
- Details zur Ausführung des Partial Express-Workflows sind verfügbar, wenn die Protokollierungsebene ERROR oder FATAL ist.
- Die Definition einer älteren Ausführung für den Zustandsautomaten kann nicht angezeigt werden, nachdem sie aktualisiert wurde

Die Verfügbarkeit von Express-Workflow-Ausführungsdetails hängt von Amazon CloudWatch Logs ab

> **Note**
>
> Wenn Sie die neue Konsolenerfahrung nicht aktivieren, um Express-Workflow-Ausführungen anzuzeigen, sind die Ausführungsverläufe und die entsprechenden Ausführungsdetails in der Step-Functions-Konsole nicht verfügbar. Um die neue Konsolenerfahrung zu aktivieren, wählen Sie die Schaltfläche Aktivieren, die im Banner auf der Registerkarte Ausführungen angezeigt wird.

Bei Express-Workflows werden der Ausführungsverlauf und die detaillierten Ausführungsinformationen über CloudWatch Logs Insights erfasst. Diese Informationen werden in der CloudWatch Protokollgruppe gespeichert, die Sie beim Erstellen des Zustandsautomaten angeben. Der Ausführungsverlauf des Zustandsautomaten wird auf der Registerkarte Ausführungen in der Step-Functions-Konsole angezeigt. Detaillierte Informationen zu jeder Ausführung des Zustandsautomaten werden auf der Seite Ausführungsdetails für die gewählte Ausführung angezeigt.

> **Warning**
>
> Wenn Sie die CloudWatch Protokolle für einen Express-Workflow löschen, wird dieser nicht auf der Registerkarte Ausführungen aufgeführt.

Wir empfehlen, dass Sie die Standardprotokollebene ALL verwenden, um alle Ausführungsereignistypen zu protokollieren. Sie können die Protokollebene nach Bedarf für Ihre vorhandenen Zustandsautomaten aktualisieren, wenn Sie sie bearbeiten. Weitere Informationen finden Sie unter Protokollierung mitCloudWatchProtokolle und Protokollstufen.

Details zur Ausführung des Partial Express-Workflows sind verfügbar, wenn die Protokollierungsebene ERROR oder FATAL ist.

Standardmäßig ist die Protokollierungsstufe für Express-Workflow-Ausführungen auf ALL festgelegt. Wenn Sie die Protokollebene ändern, sind die Ausführungsverläufe und Ausführungsdetails für abgeschlossene Ausführungen nicht betroffen. Alle neuen Ausführungen geben jedoch Protokolle aus, die auf der aktualisierten Protokollebene basieren. Weitere Informationen finden Sie unter Protokollierung mitCloudWatchProtokolle und Protokollstufen.

Wenn Sie beispielsweise die Protokollebene von ALL in ERROR oder FATAL ändern, listet die Registerkarte Ausführungen in der Step Functions-Konsole nur fehlgeschlagene Ausführungen auf. Auf der Registerkarte Ereignisansicht zeigt die Konsole nur die Ereignisdetails für die fehlgeschlagenen Schritte des Zustandsautomaten an.

Wir empfehlen, dass Sie die Standardprotokollebene ALL verwenden, um alle Ausführungsereignistypen zu protokollieren. Sie können die Protokollebene nach Bedarf für Ihre vorhandenen Zustandsautomaten aktualisieren, wenn Sie den Zustandsautomaten bearbeiten.

Die Definition einer älteren Ausführung für den Zustandsautomaten kann nicht angezeigt werden, nachdem sie aktualisiert wurde

Definitionen von Zustandsautomaten für vergangene Ausführungen werden für Express-Workflows nicht gespeichert. Wenn Sie die Definition des Zustandsautomaten ändern, können Sie die Definition des Zustandsautomaten für Ausführungen nur mit der aktuellsten Definition anzeigen.

Wenn Sie beispielsweise einen oder mehrere Schritte aus Ihrer Definition des Zustandsautomaten entfernen, erkennt Step Functions eine Nichtübereinstimmung zwischen der Definition und früheren Ausführungsereignissen. Da frühere Definitionen nicht für Express-Workflows gespeichert sind, kann Step Functions die Definition des Zustandsautomaten für Ausführungen, die auf einer früheren Version der Definition des Zustandsautomaten ausgeführt werden, nicht anzeigen. Daher sind die Registerkarten Ausführungseingabe und -ausgabe , Definition , Diagrammansicht und Tabellenansicht nicht für Ausführungen verfügbar, die auf früheren Versionen einer Zustandsautomatendefinition ausgeführt wurden.

Anzeigen und Debuggen von Ausführungen in der Step Functions-Konsole

Auf der Seite Ausführungsdetails der Step Functions-Konsole werden Informationen zu früheren und laufenden Ausführungen von Zustandsautomaten für Standard- und Express-Workflows angezeigt. Diese Informationen werden in einem Dashboard-Format angezeigt. Sie können beispielsweise die Amazon States Language-Definition des Zustandsautomaten, seinen Ausführungsstatus, seinen ARN und die Gesamtzahl der Zustandsübergänge finden. Sie können auch die Ausführungsdetails für jeden einzelnen Zustand im Zustandsautomaten anzeigen.

Inhalt

Seite mit Ausführungsdetails – Schnittstellenübersicht

Die Details zu all Ihren laufenden und vergangenen Ausführungen von Zustandsautomaten für Standard- und Express-Workflows finden Sie auf der Seite Ausführungsdetails. Wenn Sie beim Starten Ihrer Ausführung eine Ausführungs-ID angegeben haben, hat diese Seite den Titel mit dieser Ausführungs-ID. Andernfalls hat sie den Titel mit der eindeutigen Ausführungs-ID, die Step Functions automatisch für Sie generiert.

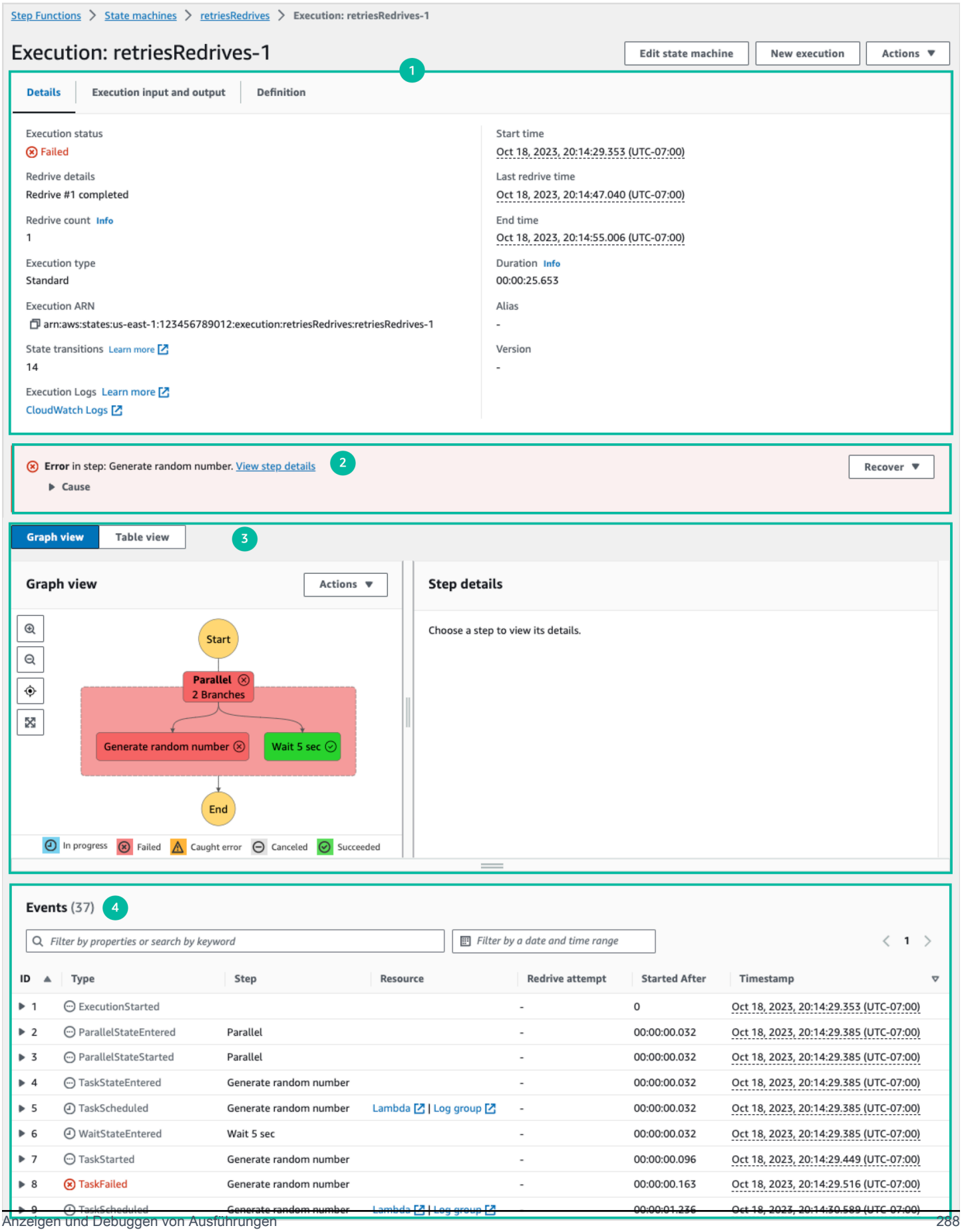
Neben den Ausführungsmetriken bietet die Seite Ausführungsdetails die folgenden Optionen für die Verwaltung Ihres Zustandsautomaten und seiner Ausführung:

Button	Wählen Sie diese Schaltfläche, um:
Bearbeiten des Zustandsautomaten	Bearbeiten Sie die Amazon States Language-Definition Ihres Zustandsautomaten.
Neue Ausführung	Starten Sie eine neue Ausführung Ihres Zustandsautomaten.
Aktionen	Bietet die folgenden Optionen zur Auswahl: • Ausführung beenden – Anhalten einer laufenden Ausführung. Diese Option ist für abgeschlossene Ausführungen nicht zugänglich. • Redrive – Redrive Ausführungen von Standard-Workflows, die in den letzten 14 Tagen nicht erfolgreich abgeschlossen wurden. Dazu gehören fehlgeschlagene, abgebrochene oder abgelaufene Ausführungen. Weitere Informationen finden Sie unter Redriving -Ausführungen. • Exportieren– Exportieren Sie die Ausführungsdetails im JSON-Format, um sie mit anderen zu teilen oder eine Offline-Analyse durchzuführen. • Feedback senden – Teilen Sie Feedback über die Schnittstelle.

Anzeigen von Ausführungen, die mit einer Version oder einem Alias gestartet wurden

Sie können die Ausführungen, die mit einer Version oder einem Alias gestartet wurden, auch in der Step Functions-Konsole anzeigen. Weitere Informationen finden Sie unter Auflisten von Ausführungen für Versionen und Aliasse.

Die Konsolenseite Ausführungsdetails enthält die folgenden Abschnitte:


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. Ausführungsübersicht
2. Fehlermeldung
3. Anzeigemodus
4. Schrittdetails
5. Ereignisse

Ausführungsübersicht

Der Abschnitt Ausführungsübersicht wird oben auf der Seite Ausführungsdetails angezeigt. Dieser Abschnitt bietet Ihnen einen Überblick über die Ausführungsdetails Ihres Workflows. Diese Informationen sind auf die folgenden drei Registerkarten aufgeteilt:

Details

Zeigt Informationen an, z. B. den Status der Ausführung, den ARN und Zeitstempel für die Start- und Endzeit der Ausführung. Sie können auch die Gesamtzahl der Zustandsübergänge anzeigen, die während der Ausführung des Zustandsautomaten stattgefunden haben. Sie können auch die Links für die X-Ray-Ablaufverfolgungszuordnung und Amazon CloudWatch -Ausführungsprotokolle anzeigen, wenn Sie die Nachverfolgung oder Protokolle für Ihren Zustandsautomaten aktiviert haben.

Wenn die Ausführung Ihres Zustandsautomaten von einem anderen Zustandsautomaten gestartet wurde, können Sie den Link für den übergeordneten Zustandsautomaten auf dieser Registerkarte anzeigen.

Wenn die Ausführung Ihres Zustandsautomaten warredriven, werden auf dieser Registerkarte redrive zugehörige Informationen angezeigt, z. B. die Redrive Anzahl .

Eingabe und Ausgabe der Ausführung

Zeigt die Eingabe und Ausgabe der Ausführung des Zustandsautomaten an side-by-side.

Definition

Zeigt die Amazon States Language-Definition des Zustandsautomaten an.

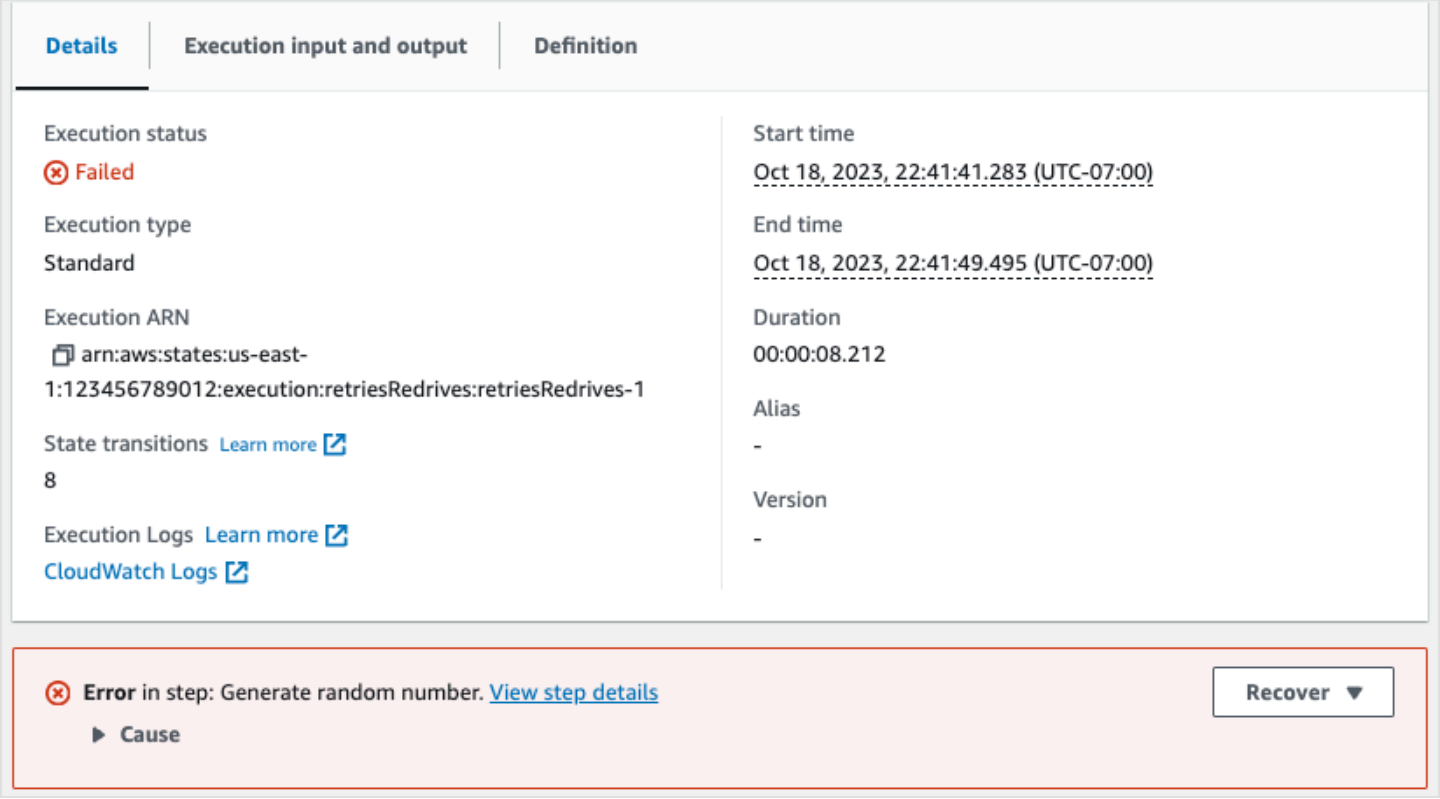
Fehlermeldung

Wenn die Ausführung Ihres Zustandsautomaten fehlgeschlagen ist, wird auf der Seite Ausführungsdetails eine Fehlermeldung angezeigt. Wählen Sie Ursache oder Schrittdetails anzeigen

in der Fehlermeldung, um den Grund für den Ausführungsfehler oder den Schritt anzuzeigen, der den Fehler verursacht hat.

Wenn Sie Schrittdetails anzeigen wählen, hebt Step Functions den Schritt hervor, der den Fehler auf den Registerkarten Schrittdetails, Diagrammansicht und Tabellenansicht verursacht hat. Wenn der Schritt ein Aufgaben-, Zuordnungs- oder Parallelstatus ist, für den Sie Wiederholungsversuche definiert haben, zeigt der Bereich Schrittdetails die Registerkarte Wiederholen für den Schritt an. Wenn Sie redriven die Ausführung haben, können Sie außerdem die Wiederholungs- und redrive Ausführungsdetails auf der Registerkarte Wiederholungen und redrives im Bereich Schrittdetails sehen.

Über die Dropdown-Schaltfläche Wiederherstellen in dieser Fehlermeldung können Sie entweder redrive Ihre erfolglosen Ausführungen verwenden oder eine neue Ausführung starten. Weitere Informationen finden Sie unter Redriving -Ausführungen.



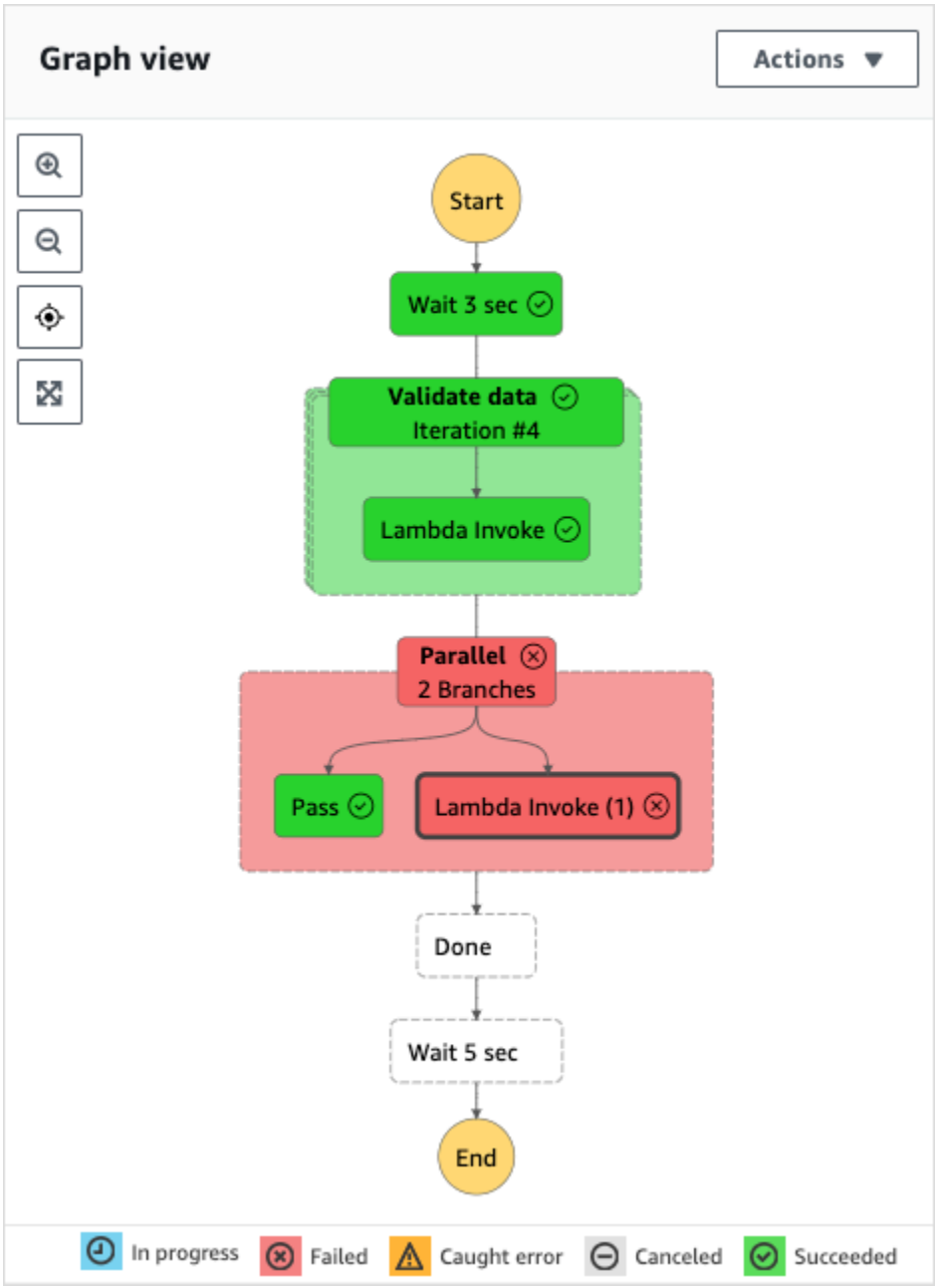
Anzeigemodus

Der Abschnitt Ansichtsmodus enthält zwei verschiedene Visualisierungen für Ihren Zustandsautomaten. Sie können wählen, ob Sie eine grafische Darstellung des Workflows, eine Tabelle, die die Zustände in Ihrem Workflow beschreibt, oder eine Liste der Ereignisse anzeigen möchten, die mit der Ausführung Ihres Zustandsautomaten verknüpft sind:

> **Note**
>
> Wählen Sie eine Registerkarte aus, um ihren Inhalt anzuzeigen.

Graph view

Im Modus Diagrammansicht wird eine grafische Darstellung Ihres Workflows angezeigt. Unten befindet sich eine Legende, die den Ausführungsstatus des Zustandsautomaten angibt. Sie enthält auch Schaltflächen, mit denen Sie den gesamten Workflow vergrößern, verkleinern, zentrieren oder den Workflow im Vollbildmodus anzeigen können.



In dieser Ansicht können Sie einen beliebigen Schritt in Ihrem Workflow auswählen, um Details zu seiner Ausführung in der Komponente Schrittdetails anzuzeigen. Wenn Sie einen Schritt in der Diagrammansicht ausgewählt haben, zeigt die Tabellenansicht diesen Schritt ebenfalls an. Dies gilt auch in umgekehrter Hinsicht. Wenn Sie einen Schritt aus der Tabellenansicht auswählen, zeigt die Diagrammansicht denselben Schritt an.

Wenn Ihr Zustandsautomat einen `Map` -Zustand, einen -`Parallel`Zustand oder beides enthält, können Sie deren Namen im Workflow in der Diagrammansicht anzeigen. Darüber hinaus können Sie für den -`Map`Zustand in der Diagrammansicht über verschiedene Iterationen der

Ausführungsdaten des Zuordnungsstatus hinweg wechseln. Wenn Ihr Map-Status beispielsweise fünf Iterationen hat und Sie die Ausführungsdaten für die dritte und vierte Iteration anzeigen möchten, gehen Sie wie folgt vor:

1. Wählen Sie den Zuordnungsstatus aus, dessen Iterationsdaten Sie anzeigen möchten.
2. Wählen Sie unter Iterationsbetrachter zuordnen die Option 2 aus der Dropdown-Liste für die dritte Iteration aus. Dies liegt daran, dass Iterationen von Null gezählt werden. Wählen Sie ebenfalls #3 aus der Dropdown-Liste für die vierte Iteration des Zuordnungsstatus aus.

   Alternativ können Sie die
   ▽
   Steuerelemente
   △
   und verwenden, um zwischen verschiedenen Iterationen des Zuordnungsstatus zu wechseln.

   > **Note**
   >
   > Wenn Ihr Zustandsautomat verschachtelte `Map` Zustände enthält, werden die Dropdown-Listen für die übergeordneten und untergeordneten `Map` Zustandsiterationen angezeigt, wie im folgenden Beispiel gezeigt:
   >
   > 
   >

3. (Optional) Wenn eine oder mehrere Ihrer Iterationen des Zuordnungsstatus nicht ausgeführt werden konnten oder die Ausführung gestoppt wurde, können Sie ihre Daten anzeigen, indem Sie diese Iterationsnummern unter Fehlgeschlagen oder Abgebrochen in der Dropdown-Liste auswählen.

Schließlich können Sie die Schaltflächen Export ierenund Layout verwenden, um das Workflow-Diagramm als SVG- oder PNG-Bild zu exportieren. Sie können auch zwischen horizontalen und vertikalen Ansichten Ihres Workflows wechseln.

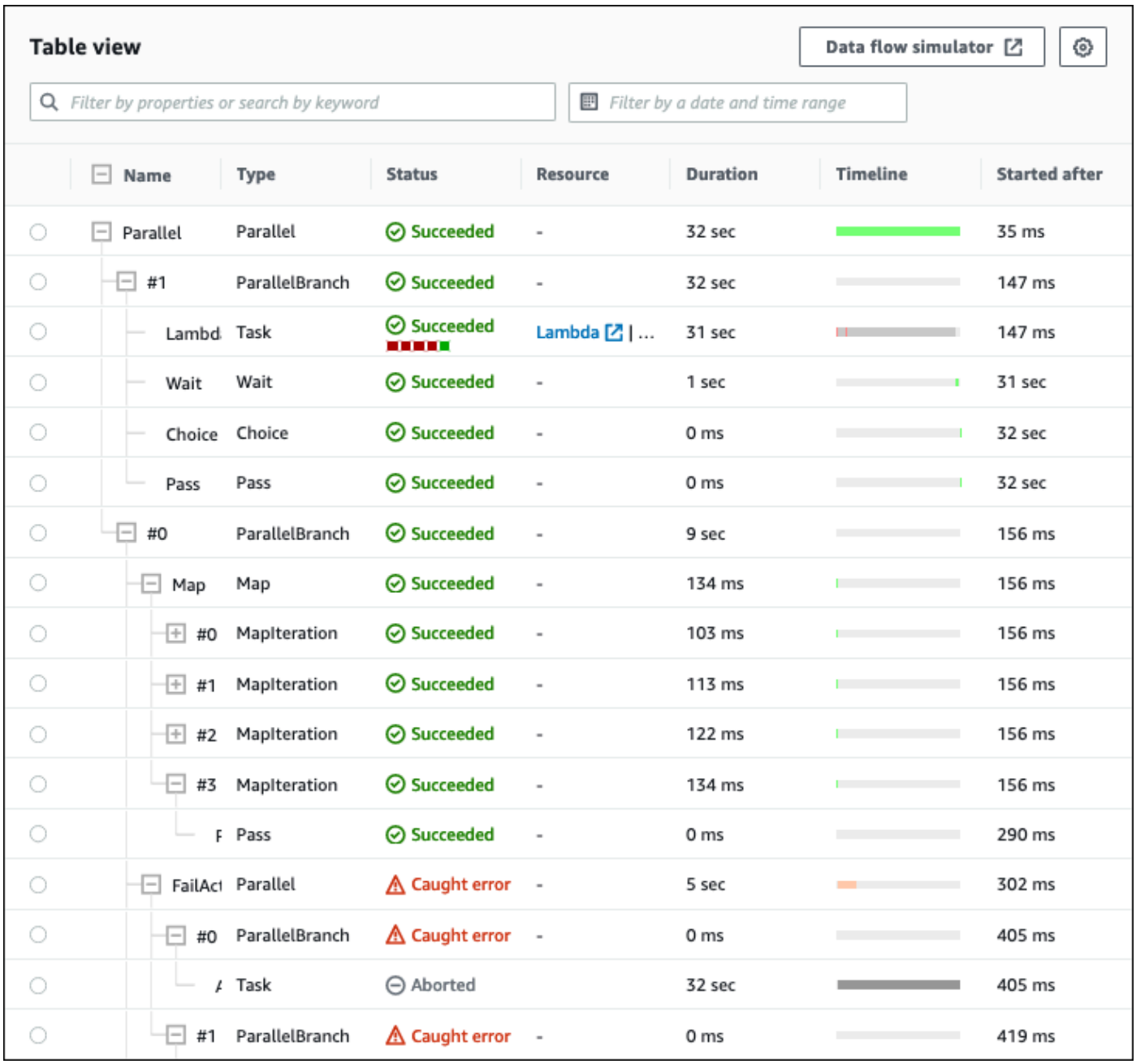
Table view

Der Tabellenansichtsmodus zeigt eine tabellarische Darstellung der Zustände in Ihrem Workflow an. In diesem Ansichtsmodus können Sie die Details zu jedem Status anzeigen, der in Ihrem

Workflow ausgeführt wurde, einschließlich seines Namens, des Namens jeder verwendeten Ressource (z. B. einer - AWS Lambda Funktion) und ob der Status erfolgreich ausgeführt wurde.

In dieser Ansicht können Sie einen beliebigen Status in Ihrem Workflow auswählen, um Details zu seiner Ausführung in der Komponente Schrittdetails anzuzeigen. Wenn Sie einen Schritt in der Tabellenansicht ausgewählt haben, zeigt die Diagrammansicht diesen Schritt ebenfalls an. Dies gilt auch in umgekehrter Hinsicht. Wenn Sie einen Schritt aus der Diagrammansicht auswählen, zeigt die Tabellenansicht denselben Schritt an.

Sie können auch die im Tabellenansichtsmodus angezeigte Datenmenge begrenzen, indem Sie Filter auf die Ansicht anwenden. Sie können einen Filter für eine bestimmte Eigenschaft erstellen, z. B. Status oder Redrive Versuch . Weitere Informationen finden Sie unter Tutorial: Untersuchen von Ausführungen von Zustandsautomaten mithilfe der Step-Functions-Konsole.

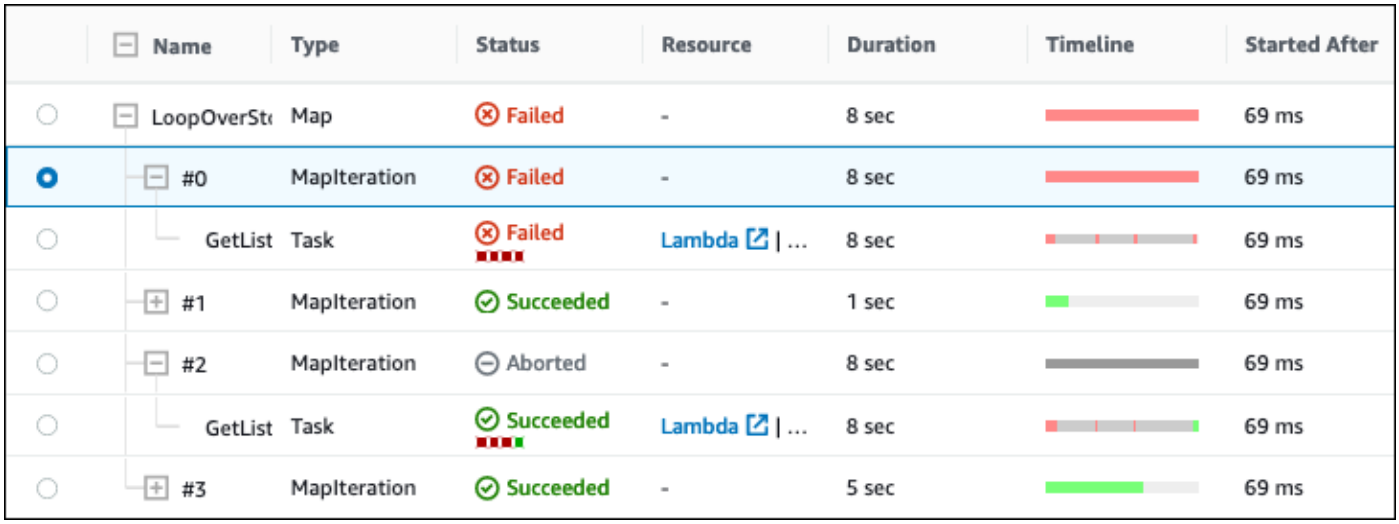


Standardmäßig zeigt dieser Modus die Spalten Name , Typ , Status , Ressource und Started After an. Sie können die Spalten, die Sie anzeigen möchten, über das Dialogfeld Einstellungen konfigurieren. Die Auswahl, die Sie in diesem Dialogfeld treffen, bleibt für zukünftige Ausführungen von Zustandsautomaten bestehen, bis sie erneut geändert werden.

Wenn Sie die Spalte Timeline hinzufügen, wird die Ausführungsdauer jedes Zustands in Bezug auf die Laufzeit für die gesamte Ausführung angezeigt. Dies wird als farbige, lineare Zeitleiste angezeigt. Dies kann Ihnen helfen, leistungsbezogene Probleme bei der Ausführung eines bestimmten Status zu identifizieren. Anhand der farbigen Segmente für jeden Status auf

der Zeitleiste können Sie den Ausführungsstatus des Status ermitteln, z. B. in Bearbeitung, fehlgeschlagen oder abgebrochen.

Wenn Sie beispielsweise Ausführungswiederholungen für einen -Zustand in Ihrem Zustandsautomaten definiert haben, werden diese Wiederholungen in der Zeitlinie angezeigt. Rote Segmente stellen die fehlgeschlagenen `Retry` Versuche dar, während hellgraue Segmente die `BackoffRate` zwischen jedem `Retry` Versuch darstellen.



Wenn Ihr Zustandsautomat einen `Map` Status, einen `Parallel` Status oder beides enthält, können Sie ihre Namen im Workflow in der Tabellenansicht anzeigen. Für die `Parallel` Zustände `Map` und zeigt der Tabellenansichtsmodus die Ausführungsdaten für ihre Iterationen und parallelen Verzweigungen als Knoten innerhalb einer Baumansicht an. Sie können jeden Knoten in diesen Status auswählen, um seine einzelnen Details im Abschnitt Schrittdetails anzuzeigen. Sie können beispielsweise die Daten für eine bestimmte Iteration des Zuordnungsstatus überprüfen, die dazu führte, dass der Status fehlschlug. Erweitern Sie den Knoten für den Map-Status und zeigen Sie dann den Status für jede Iteration in der Spalte Status an.

Schrittdetails

Der Abschnitt Schrittdetails wird rechts geöffnet, wenn Sie einen Status in der Diagrammansicht oder Tabellenansicht auswählen. Dieser Abschnitt enthält die folgenden Registerkarten, auf denen Sie detaillierte Informationen über den ausgewählten Status erhalten:

Eingabe

Zeigt die Eingabedetails des ausgewählten Status an. Wenn in der Eingabe ein Fehler auftritt, wird dieser mit einem

auf der Registerkartenüberschrift angezeigt. Darüber hinaus können Sie den Grund für den Fehler auf dieser Registerkarte anzeigen.

Sie können auch die Schaltfläche Erweiterte Ansicht wählen, um den Eingabedatenübertragungspfad als die Daten anzuzeigen, die durch den ausgewählten Status übergeben werden. Auf diese Weise können Sie feststellen, wie Ihre Eingabe verarbeitet wurde, wenn eines oder mehrere der Felder, z. B. `InputPath`, `ResultSelectorParameters`, `OutputPath`, und `ResultPath`, auf die Daten angewendet wurden.

Ausgabe

Zeigt die Ausgabe des ausgewählten Status an. Wenn es einen Fehler in der Ausgabe gibt, wird dieser mit einem

auf der Registerkartenüberschrift angezeigt. Darüber hinaus können Sie den Grund für den Fehler auf dieser Registerkarte anzeigen.

Sie können auch die Schaltfläche Erweiterte Ansicht wählen, um den Ausgabedatenübertragungspfad als die Daten anzuzeigen, die durch den ausgewählten Status übergeben werden. Auf diese Weise können Sie feststellen, wie Ihre Eingabe verarbeitet wurde, wenn eines oder mehrere der Felder, z. B. `InputPath`, `ResultSelectorParameters`, `OutputPath`, und `ResultPath`, auf die Daten angewendet wurden.

Details

Zeigt Informationen an, z. B. den Statustyp, seinen Ausführungsstatus und seine Ausführungsdauer.

Bei `Task` Zuständen, die eine Ressource verwenden, z. B. AWS Lambda, enthält diese Registerkarte Links zur Ressourcendefinitionsseite und zur Amazon CloudWatch-Protokollseite für den Ressourcenaufruf. Sie zeigt auch Werte, falls angegeben, für die `HeartbeatSeconds` Felder `TimeoutSeconds` und des `Task` Status an.

Für -`Map`Zustände zeigt diese Registerkarte Informationen zur Gesamtzahl der Iterationen eines `Map` Zustands an. Iterationen werden als Fehlgeschlagen, Abgebrochen, Erfolgreich oder kategorisiertInProgress.

Definition

Zeigt die Amazon States Language-Definition an, die dem ausgewählten Status entspricht.

Wiederholen

> **Note**
>
> Diese Registerkarte wird nur angezeigt, wenn Sie ein `Retry` Feld im `Parallel` Status `Task` oder Ihres Zustandsautomaten definiert haben.

Zeigt die ersten und nachfolgenden Wiederholungsversuche für einen ausgewählten Status im ursprünglichen Ausführungsversuch an. Wählen Sie für den ersten und alle nachfolgenden fehlgeschlagenen Versuche den

neben Typ aus, um den Grund für den Fehler anzuzeigen, der in einem Dropdown-Feld angezeigt wird. Wenn der Wiederholungsversuch erfolgreich ist, können Sie die Ausgabe anzeigen, die in einem Dropdown-Feld angezeigt wird.

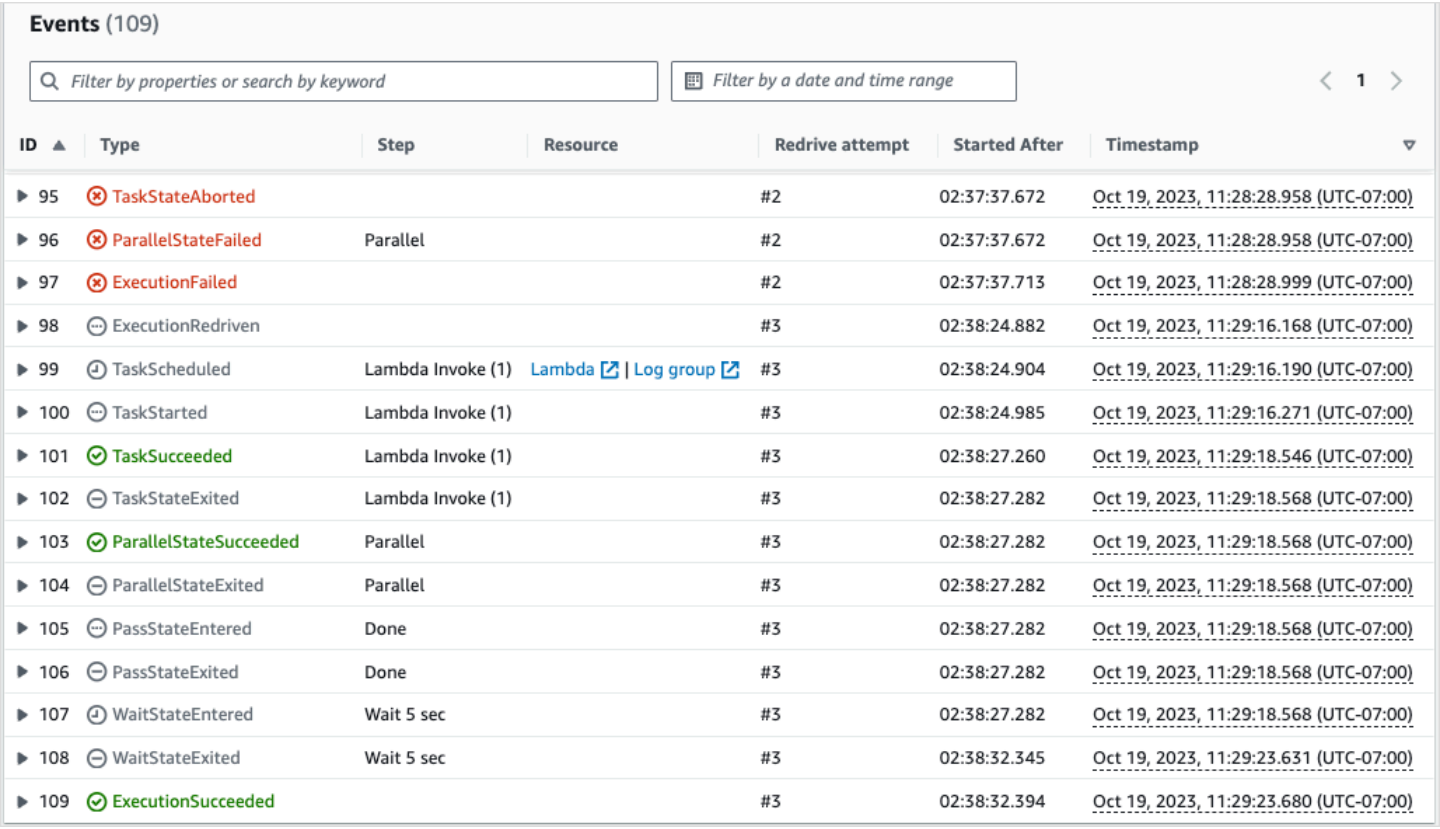
Wenn Sie redriven Ihre Ausführung haben, zeigt dieser Tabulator-Header den Namen Wiederholungsversuche & redrives und die Wiederholungsversuchsdetails für jede anredrive.

Ereignisse

Zeigt eine gefilterte Liste der Ereignisse an, die dem ausgewählten Status in einer Ausführung zugeordnet sind. Die Informationen, die Sie auf dieser Registerkarte sehen, sind eine Teilmenge des vollständigen Ausführungsereignisverlaufs, den Sie in der Tabelle Ereignisse sehen.

Ereignisse

Die Tabelle Ereignisse zeigt den vollständigen Verlauf für die ausgewählte Ausführung als Liste von Ereignissen an, die sich über mehrere Seiten erstrecken. Jede Seite enthält bis zu 25 Ereignisse. In diesem Abschnitt wird auch die Gesamtzahl der Ereignisse angezeigt, anhand derer Sie feststellen können, ob Sie die maximale Anzahl von Ereignissen im Ereignisverlauf von 25.000 überschritten haben.



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)

Standardmäßig werden die Ergebnisse in der Tabelle Ereignisse in aufsteigender Reihenfolge angezeigt, basierend auf dem Zeitstempel der Ereignisse. Sie können die Sortierung des Verlaufs des Ausführungsereignisses in absteigende Reihenfolge ändern, indem Sie auf die Spaltenüberschrift Zeitstempel klicken.

In der Tabelle Ereignisse ist jedes Ereignis farbenkodiert, um seinen Ausführungsstatus anzugeben. Beispielsweise werden fehlgeschlagene Ereignisse rot angezeigt. Um weitere Details zu einem Ereignis anzuzeigen, wählen Sie die

neben der Ereignis-ID aus. Nach dem Öffnen zeigen die Ereignisdetails die Eingabe, Ausgabe und den Ressourcenaufruf für das Ereignis an.

Darüber hinaus können Sie in der Tabelle Ereignisse Filter anwenden, um die angezeigten Ergebnisse des Ausführungsereignisverlaufs einzuschränken. Sie können Eigenschaften wie die ID auswählen oder Redrive versuchen. Weitere Informationen finden Sie unter Tutorial: Untersuchen von Ausführungen von Zustandsautomaten mithilfe der Step-Functions-Konsole.

Tutorial: Untersuchen von Ausführungen von Zustandsautomaten mithilfe der Step-Functions-Konsole

In diesem Tutorial erfahren Sie, wie Sie die auf der Seite Ausführungsdetails angezeigten Ausführungsinformationen überprüfen und den Grund für eine fehlgeschlagene Ausführung anzeigen. Anschließend erfahren Sie, wie Sie auf verschiedene Iterationen einer `Map` Zustandsausführung zugreifen. Schließlich erfahren Sie, wie Sie die Spalten in der Tabellenansicht konfigurieren und geeignete Filter anwenden, um nur die für Sie interessanten Informationen anzuzeigen.

In diesem Tutorial erstellen Sie einen Zustandsautomaten vom Typ Standard, der den Preis für einen Satz von Lebensmitteln abruft. Dazu verwendet der Zustandsautomat drei AWS Lambda Funktionen, die eine zufällige Liste von vier Lebensmitteln, den Preis jeder Lebensmittel und die durchschnittlichen Kosten der Lebensmittel zurückgeben. Die Lambda-Funktionen sind so konzipiert, dass sie einen Fehler auslösen, wenn der Preis der Lebensmittel kleiner oder gleich einem Schwellenwert ist.

> **Note**
>
> Das folgende Verfahren enthält zwar Anweisungen zur Untersuchung der Details einer Standard-Workflow-Ausführung, Sie können aber auch die Details für Express-Workflow-Ausführungen untersuchen. Informationen zu den Unterschieden zwischen den Ausführungsdetails für Standard- und Express-Workflow-Typen finden Sie unter Standard- und Express-Workflow-Ausführungen in der Konsole.

Inhalt

Schritt 1: Erstellen und Testen der erforderlichen Lambda-Funktionen

1. Öffnen Sie die Lambda-Konsole und führen Sie dann die Schritte 1 bis 4 im Schritt 1: Erstellen einer Lambda-Funktion Abschnitt aus. Stellen Sie sicher, dass Sie die Lambda-Funktion benennen**`GetListOfFruits`**.

2. Nachdem Sie Ihre Lambda-Funktion erstellt haben, kopieren Sie den Amazon-Ressourcennamen (ARN) der Funktion, der in der oberen rechten Ecke der Seite angezeigt wird. Um den ARN zu kopieren, klicken Sie auf

.

Im Folgenden finden Sie ein Beispiel für einen ARN, wobei der Name der Lambda-Funktion *`function-name`* ist (in diesem Fall `GetListOfFruits`):

```
arn:aws:lambda:us-east-1:123456789012:function:function-name
```

3. Kopieren Sie den folgenden Code für die Lambda-Funktion in den Bereich Codequelle der GetListOfFruits Seite.

```
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. Wählen Sie Bereitstellen und dann Testen aus, um die Änderungen bereitzustellen und die Ausgabe Ihrer Lambda-Funktion anzuzeigen.
5. Erstellen Sie zwei zusätzliche Lambda-Funktionen mit dem Namen **`GetFruitPrice`** bzw**`CalculateAverage`**. mit den folgenden Schritten:

   a. Kopieren Sie den folgenden Code in den Bereich Codequelle der GetFruitPrice Lambda-Funktion:

   ```
   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. Kopieren Sie den folgenden Code in den Bereich Codequelle der CalculateAverage Lambda-Funktion:

   ```
   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. Stellen Sie sicher, dass Sie die ARNs dieser beiden Lambda-Funktionen kopieren und sie dann bereitstellen und testen.

Schritt 2: Erstellen und Ausführen des Zustandsautomaten

Verwenden Sie die Step Functions-Konsole, um einen Zustandsautomaten zu erstellen, der die Lambda-Funktionen aufruft, die Sie in Schritt 1 erstellt haben. In diesem Zustandsautomaten sind drei `Map` Zustände definiert. Jeder dieser `Map` Zustände enthält einen `Task` Status, der eine Ihrer Lambda-Funktionen aufruft. Darüber hinaus wird in jedem `Task` Status ein `Retry` Feld mit einer Reihe von Wiederholungsversuchen definiert, die für jeden Status definiert sind. Wenn bei einem `Task` Status ein Laufzeitfehler auftritt, wird er erneut bis zu der für diesen definierten Anzahl von Wiederholungsversuchen ausgeführt`Task`.

1. Öffnen Sie die Step-Functions-Konsole und wählen Sie Workflow in Code schreiben aus.

> **Important**
>
> Stellen Sie sicher, dass sich Ihr Zustandsautomat unter demselben AWS Konto und derselben Region befindet wie die zuvor erstellte Lambda-Funktion.

2. Behalten Sie für Typ die Standardauswahl von Standard bei.
3. Kopieren Sie die folgende Amazon States Language-Definition und fügen Sie sie unter Definition ein. Stellen Sie sicher, dass Sie die angezeigten ARNs durch die der zuvor erstellten Lambda-Funktionen ersetzen.

```
{
    "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. Geben Sie einen Namen für Ihren Zustandsautomaten ein. Behalten Sie die Standardauswahl für die anderen Optionen auf dieser Seite bei und wählen Sie Zustandsautomat erstellen aus.

5. Öffnen Sie die Seite mit dem Titel mit dem Namen Ihres Zustandsautomaten. Führen Sie die Schritte 1 bis 4 im Schritt 4: Führen Sie die Zustandsmaschine aus Abschnitt aus, verwenden Sie jedoch die folgenden Daten als Ausführungseingabe:

```
{
    "stores": [
      "Store A",
      "Store B",
      "Store C",
      "Store D"
    ]
}
```

Schritt 3: Anzeigen der Ausführungsdetails des Zustandsautomaten

Auf der Seite mit dem Titel Ihrer Ausführungs-ID können Sie die Ergebnisse Ihrer Ausführung überprüfen und Fehler debuggen.

1. (Optional) Wählen Sie aus den Registerkarten, die auf der Seite Ausführungsdetails angezeigt werden, um die Informationen anzuzeigen, die in jeder von ihnen vorhanden sind. Um beispielsweise die Eingabe des Zustandsautomaten und seine Ausführungsausgabe anzuzeigen, wählen Sie Ausführungseingabe und -ausgabe im Abschnitt Ausführungsübersicht aus.
2. Wenn die Ausführung Ihres Zustandsautomaten fehlgeschlagen ist, wählen Sie in der Fehlermeldung die Option Ursache oder Schrittdetail anzeigen aus. Details zum Fehler werden im Abschnitt Schrittdetails angezeigt. Beachten Sie, dass der Schritt, der den Fehler verursacht hat, bei dem es sich um einen `Task` Status mit dem Namen handeltGetListofFruits, in der Diagramm- und Tabellenansicht hervorgehoben ist.

> **Note**
>
> Da der GetListofFruits Schritt innerhalb eines -`Map`Zustands definiert ist und der Schritt nicht erfolgreich ausgeführt werden konnte, wird der `Map` Statusschritt als Fehlgeschlagen angezeigt.

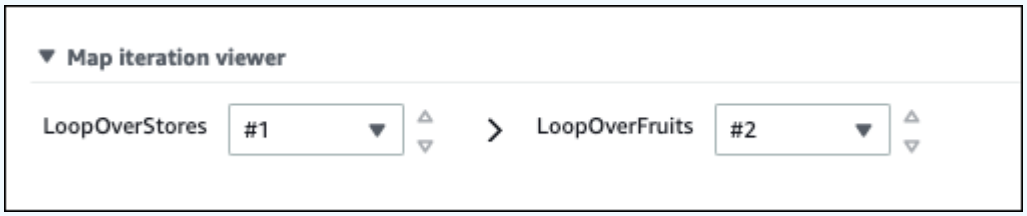
Schritt 4: Erkunden der verschiedenen Ansichtsmodi

Sie können einen bevorzugten Modus auswählen, um entweder den Workflow des Zustandsautomaten oder den Verlauf des Ausführungsereignisses anzuzeigen. Einige der Aufgaben, die Sie in diesen Ansichtsmodi ausführen können, sind:

Diagrammansicht – Wechseln Sie zwischen verschiedenen **`Map`** Statusiterationen

Wenn Ihr Map-Status fünf Iterationen hat und Sie die Ausführungsdetails für die dritte und vierte Iteration anzeigen möchten, gehen Sie wie folgt vor:

1. Wählen Sie den `Map` Status aus, für den Sie die Iterationsdaten anzeigen möchten.
2. Wählen Sie unter Iterationsbetrachter zuordnen die Iteration aus, die Sie anzeigen möchten. Iterationen werden ab Null gezählt. Um die dritte Iteration aus fünf auszuwählen, wählen Sie #2 aus der Dropdown-Liste neben dem Namen des Zuordnungsstatus aus.

> **Note**
>
> Wenn Ihr Zustandsautomat verschachtelte `Map` Zustände enthält, zeigt Step Functions die Iterationen des übergeordneten und untergeordneten `Map` Zustands als zwei separate Dropdown-Listen an:
>
> 
>

3. (Optional) Wenn eine oder mehrere Ihrer `Map` Statusiterationen nicht ausgeführt werden konnten oder in einem abgebrochenen Zustand gestoppt wurden, können Sie Details zur fehlgeschlagenen Iteration anzeigen. Um diese Details anzuzeigen, wählen Sie die betroffenen Iterationsnummern unter Fehlgeschlagen oder Abgebrochen in der Dropdown-Liste aus.

Tabellenansicht – Wechseln Sie zwischen verschiedenen **`Map`** Statusiterationen

Wenn Ihr Map-Status fünf Iterationen hat und Sie die Ausführungsdetails für die Iterationsnummer drei und vier anzeigen möchten, gehen Sie wie folgt vor:

1. Wählen Sie den `Map` Status aus, für den Sie die verschiedenen Iterationsdaten anzeigen möchten.
2. Wählen Sie in der Baumansicht der `Map` Zustandsiterationen die Zeile für die Iteration mit dem Namen #2 für Iterationsnummer drei aus. Wählen Sie in ähnlicher Weise die Zeile mit dem Namen #3 für Iterationsnummer 4 aus.

Tabellenansicht – Konfigurieren Sie die anzuzeigenden Spalten

Wählen Sie

.

Wählen Sie dann im Dialogfeld Einstellungen die Spalten aus, die Sie unter Sichtbare Spalten auswählen anzeigen möchten.

Standardmäßig zeigt dieser Modus die Spalten Name , Typ , Status , Ressource und Started After an.

Tabellenansicht – Filtern der Ergebnisse

Beschränken Sie die Menge der angezeigten Informationen, indem Sie einen oder mehrere Filter anwenden, die auf einer Eigenschaft wie Status oder einem Datums- und Zeitbereich basieren. Um beispielsweise die Schritte anzuzeigen, bei denen die Ausführung fehlgeschlagen ist, wenden Sie den folgenden Filter an:

1. Wählen Sie Nach Eigenschaften filtern oder nach Schlüsselwörtern suchen und dann Status unter Eigenschaften aus.
2. Wählen Sie unter Operatoren die Option Status = aus.
3. Wählen Sie Status = Fehlgeschlagen aus.
4. (Optional) Wählen Sie Filter löschen, um die angewendeten Filter zu entfernen.

Ereignisansicht – Filtern der Ergebnisse

Beschränken Sie die Menge der angezeigten Informationen, indem Sie einen oder mehrere Filter basierend auf einer Eigenschaft wie Typ oder einem Datums- und Zeitbereich anwenden. Um beispielsweise die `Task` Statusschritte anzuzeigen, bei denen die Ausführung fehlgeschlagen ist, wenden Sie den folgenden Filter an:

1. Wählen Sie Nach Eigenschaften filtern oder nach Schlüsselwörtern suchen und dann unter Eigenschaften die Option Typ aus.
2. Wählen Sie unter Operatoren die Option Typ = aus.
3. Wählen Sie Typ = TaskFailed aus.
4. (Optional) Wählen Sie Filter löschen, um die angewendeten Filter zu entfernen.

Ereignisansicht – Überprüfen eines TaskFailed Ereignisdetails

Wählen Sie die

neben der ID eines TaskFailed Ereignisses aus, um dessen Details zu überprüfen, einschließlich Eingabe, Ausgabe und Ressourcenaufruf, die in einem Dropdown-Feld angezeigt werden.

Redriving -Ausführungen

Sie können verwendenredrive, um Ausführungen von Standard-Workflows neu zu starten, die in den letzten 14 Tagen nicht erfolgreich abgeschlossen wurden. Dazu gehören fehlgeschlagene, abgebrochene oder abgelaufene Ausführungen.

Wenn Sie redrive eine Ausführung ausführen, wird die fehlgeschlagene Ausführung ab dem erfolglosen Schritt fortgesetzt und dieselbe Eingabe verwendet. Step Functions behält die Ergebnisse und den Ausführungsverlauf der erfolgreichen Schritte bei und diese Schritte werden nicht erneut ausgeführt, wenn Sie redrive eine Ausführung ausführen. Angenommen, Ihr Workflow enthält zwei Status: einen Pass Status gefolgt von einem Status der Aufgabe Status. Wenn Ihre Workflow-Ausführung im Status Aufgabe fehlschlägt und Sie redrive die Ausführung durchführen, wird die Ausführung neu geplant und dann erneut ausgeführt.

Redriven -Ausführungen verwenden dieselbe Definition und denselben Ausführungs-ARN des Zustandsautomaten, der für den ursprünglichen Ausführungsversuch verwendet wurde. Wenn Ihr ursprünglicher Ausführungsversuch mit einer Version , einem Alias oder beidem verknüpft war, wird die redriven Ausführung mit derselben Version, demselben Alias oder beidem verknüpft. Auch wenn Sie Ihren Alias aktualisieren, sodass er auf eine andere Version verweist, verwendet die redriven Ausführung weiterhin die Version, die dem ursprünglichen Ausführungsversuch zugeordnet ist. Da redriven Ausführungen dieselbe Definition für den Zustandsautomaten verwenden, müssen Sie eine neue Ausführung starten, wenn Sie Ihre Definition für den Zustandsautomaten aktualisieren.

Wenn Sie redrive eine Ausführung ausführen, wird das Timeout auf Zustandsautomatenebene, falls definiert, auf 0 zurückgesetzt. Weitere Informationen zum Timeout auf Zustandsautomatenebene finden Sie unter `TimeoutSeconds`.

Die Ausführung redrives wird als Zustandsübergänge betrachtet. Informationen darüber, wie sich Statusübergänge auf die Abrechnung auswirken, finden Sie unter Step Functions – Preise.

Themen

- Redrive Berechtigung für erfolglose Ausführungen
- Redrive Verhalten einzelner Status
- IAM-Berechtigung für redrive eine Ausführung
- Redriving -Ausführungen in der -Konsole
- Redriving -Ausführungen mit der API
- Untersuchen von redriven Ausführungen

- Wiederholungsverhalten von redriven Ausführungen

Redrive Berechtigung für erfolglose Ausführungen

Sie können redrive Ausführungen ausführen, wenn Ihr ursprünglicher Ausführungsversuch die folgenden Bedingungen erfüllt:

- Sie haben die Ausführung am oder nach dem 15. November 2023 gestartet. Ausführungen, die Sie vor diesem Datum gestartet haben, kommen nicht für in Frageredrive.
- Der Ausführungsstatus lautet nicht `SUCCEEDED`.
- Die Workflow-Ausführung hat den redrivable Zeitraum von 14 Tagen nicht überschritten. -RedrivableZeitraum bezieht sich auf die Zeit, in der Sie redrive eine bestimmte Ausführung ausführen können. Dieser Zeitraum beginnt an dem Tag, an dem ein Zustandsautomat seine Ausführung abschließt.
- Die Workflow-Ausführung hat die maximale Öffnungszeit von einem Jahr nicht überschritten. Informationen zu Ausführungskontingenten für Zustandsautomaten finden Sie unter Kontingente im Zusammenhang mit der Ausführung von Zustandsmaschinen.
- Die Anzahl der Ausführungsereignisse beträgt weniger als 24.999. -RedrivenAusführungen fügen ihren Ereignisverlauf an den vorhandenen Ereignisverlauf an. Stellen Sie sicher, dass Ihre Workflow-Ausführung weniger als 24.999 Ereignisse enthält, um das `ExecutionRedriven` Verlaufsereignis und mindestens ein anderes Verlaufsereignis zu berücksichtigen.

Redrive Verhalten einzelner Status

Je nach Status, der in Ihrem Workflow fehlgeschlagen ist, variiert das redrive Verhalten für alle nicht erfolgreichen Status. In der folgenden Tabelle wird das redrive Verhalten für alle Status beschrieben.

Statusname	Redrive Ausführungsverhalten
Pass	Wenn ein vorangehender Schritt fehlschlägt oder der Zustandsautomat eine Zeitüberschreitung aufweist, wird der Status „Bestanden“ beendet und nicht auf ausgeführtredrive.
Status der Aufgabe	Plant und startet den Aufgabenstatus erneut.

Statusname	Redrive Ausführungsverhalten
	Wenn Sie redrive eine Ausführung ausführen, die einen Aufgabenstatus erneut ausführt, wird der `TimeoutSeconds` für den Status, falls definiert, auf 0 zurückgesetzt. Weitere Informationen zum Timeout finden Sie unter Aufgabenstatus.
Choice	Bewertet die Choice-Statusregeln neu.
Wait	Wenn der Status `Timestamp` oder angibt`TimestampPath`, die sich auf einen Zeitstempel in der Vergangenheit bezieht, redrive bewirkt, dass der Wartestatus beendet wird und in den im `Next` Feld angegebenen Status übergeht.
Succeed	Statt redrive Maschinenausführungen, die in den Status Erfolgreich wechseln, nicht zu.
Fehler	Setzt den Status Fehlgeschlagen erneut ein und schlägt erneut fehl.
Parallel	Ändert und redrives nur die Verzweigungen, die fehlgeschlagen oder abgebrochen wurden. Wenn der Status aufgrund eines `States.DataLimitExceeded` Fehlers fehlgeschlagen ist, wird der Parallelstatus erneut ausgeführt, einschließlich der Verzweigungen, die beim ursprünglichen Ausführungsversuch erfolgreich waren.

Statusname	Redrive Ausführungsverhalten
Inline-Zuweisungsstatus	plant und redrives nur die Iterationen, die fehlgeschlagen oder abgebrochen wurden. Wenn der Status aufgrund eines `States.DataLimitExceeded` Fehlers fehlgeschlagen ist, wird der Inline Map-Status erneut ausgeführt, einschließlich der Iterationen, die beim ursprünglichen Ausführungsversuch erfolgreich waren.
Status der verteilten Zuordnung	redrives Die erfolglosen untergeordneten Workflow-Ausführungen in einer Map-Ausführung. Weitere Informationen finden Sie unter RedrivingKarte läuft. Wenn der Status aufgrund eines `States.DataLimitExceeded` Fehlers fehlgeschlagen ist, wird der Status Distributed Map erneut ausgeführt. Dazu gehören die untergeordneten Workflows, die beim ursprünglichen Ausführungsversuch erfolgreich waren.

IAM-Berechtigung für redrive eine Ausführung

Step Functions benötigt die entsprechende Berechtigung für redrive eine Ausführung. Das folgende Beispiel für eine IAM-Richtlinie gewährt Ihrem Zustandsautomaten die geringste Berechtigung für redriving eine Ausführung. Denken Sie daran, den *`kursiv gedruckten`* Text durch Ihre ressourcenspezifischen Informationen zu ersetzen.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "states:RedriveExecution"
```

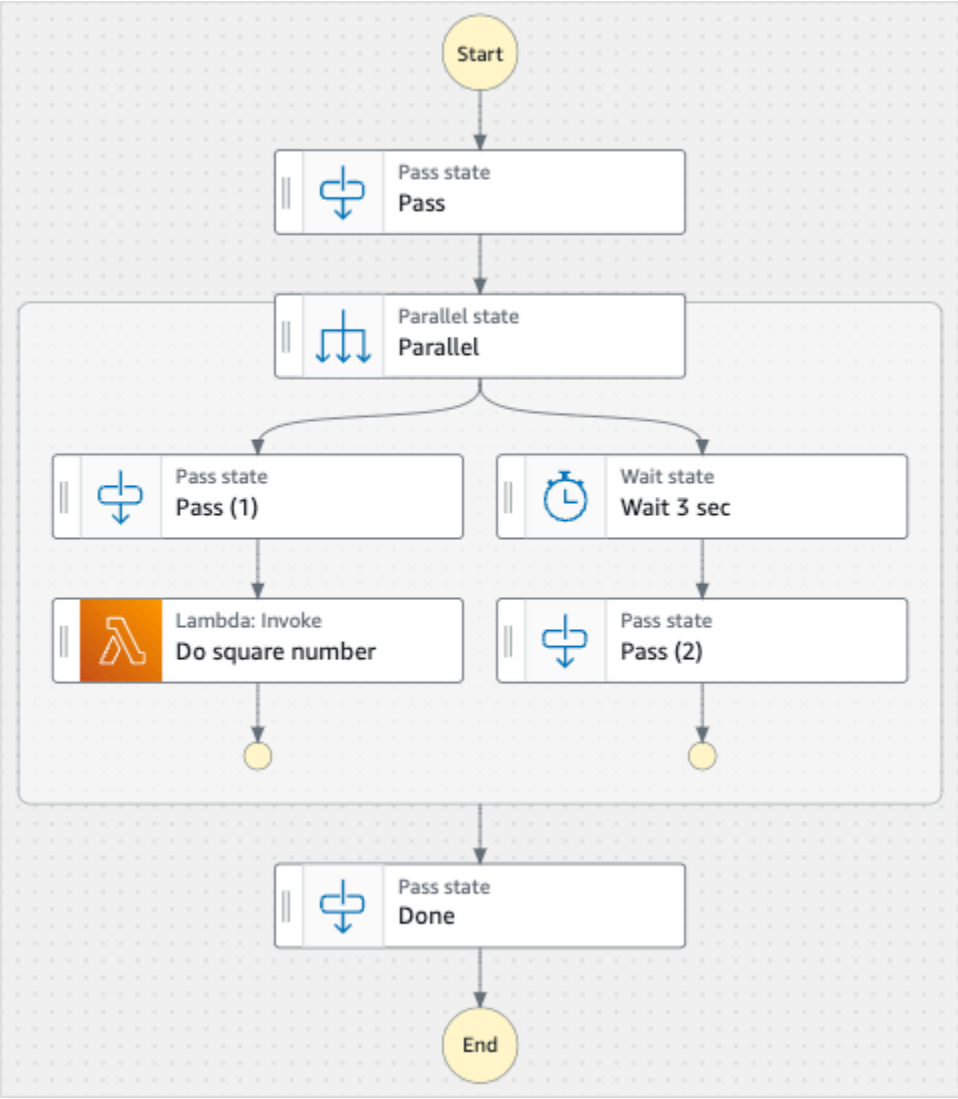
```
            ],
            "Resource": "arn:aws:states:us-
east-2:123456789012:execution:myStateMachine:*"
        }
    ]
}
```

Ein Beispiel für die Berechtigung, die Sie für redrive eine Map-Ausführung benötigen, finden Sie unter Beispiel für eine IAM-Richtlinie für redriving eine Distributed Map.

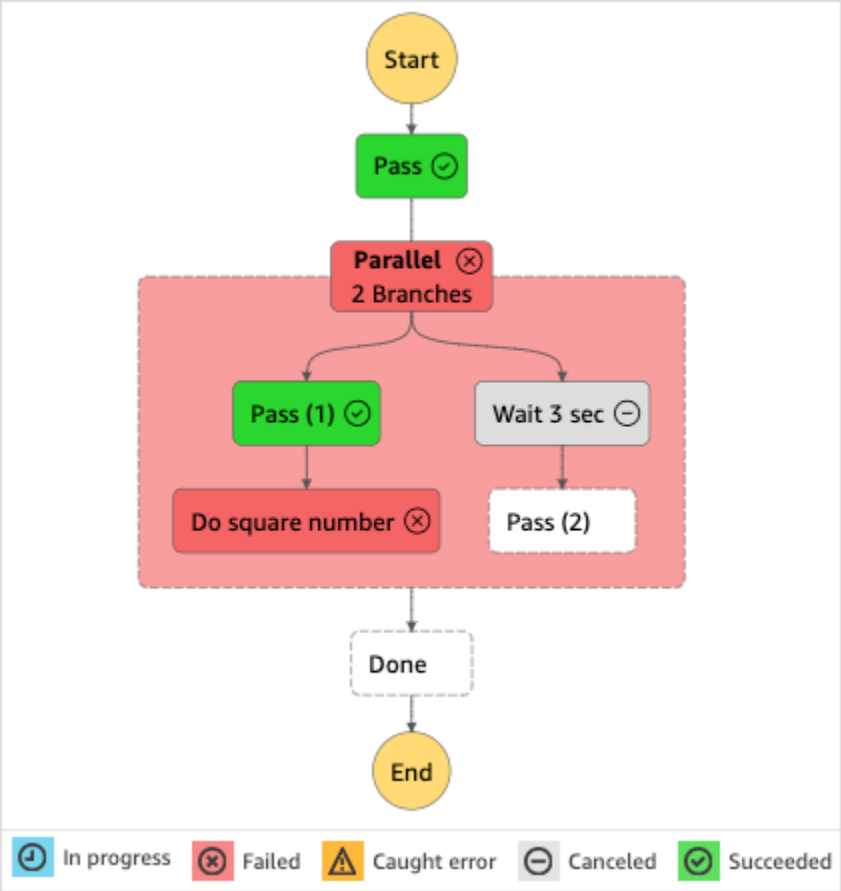
Redriving -Ausführungen in der -Konsole

Sie können redriveberechtigte Ausführungen über die Step Functions Konsole vornehmen.

Angenommen, die folgende Abbildung stellt das Workflow-Diagramm Ihres Zustandsautomaten dar.



Angenommen, Sie führen diesen Zustandsautomaten aus. Die folgende Abbildung zeigt das Diagramm für die Ausführung des Zustandsautomaten.



Wie in dieser Abbildung gezeigt, hat der Lambda Aufrufschritt mit dem Namen Quadratzahl innerhalb des Parallel-Zustands einen Fehler zurückgegeben. Dies führte dazu, dass der parallele Status fehlschlug. Die Verzweigungen, deren Ausführung in Bearbeitung war oder nicht gestartet wurde, werden gestoppt und die Ausführung des Zustandsautomaten schlägt fehl.

Zu redrive einer Ausführung von der Konsole aus

1. Öffnen Sie die Step-Functions-Konsole und wählen Sie dann einen vorhandenen Zustandsautomaten aus, dessen Ausführung fehlgeschlagen ist.
2. Wählen Sie auf der Detailseite des Zustandsautomaten unter Ausführungen eine fehlgeschlagene Ausführungs-Instance aus.
3. Wählen Sie Redrive.
4. Wählen Sie im Redrive Dialogfeld Redrive Ausführung aus.

> **Tip**
>
> Wenn Sie sich auf der Seite Ausführungsdetails einer fehlgeschlagenen Ausführung befinden, führen Sie einen der folgenden Schritte für redrive die Ausführung aus:
>
> - Wählen Sie Wiederherstellen und dann Redrive aus Fehler aus.
> - Wählen Sie Aktionen und dann ausRedrive.

Beachten Sie, dass dieselbe Zustandsautomatendefinition und denselben ARN redrive verwendet. Es führt die Ausführung ab dem Schritt aus, der beim ursprünglichen Ausführungsversuch fehlgeschlagen ist. In diesem Beispiel ist es der Schritt Quadratzahl ausführen und 3 Sekunden warten innerhalb des parallelen Zustands. Nach dem Neustart der Ausführung dieser erfolglosen Schritte im Parallel-Status redrive setzt die Ausführung für den Schritt Fertig fort.

5. Wählen Sie die Ausführung aus, um die Seite Ausführungsdetails zu öffnen.

   Auf dieser Seite können Sie die Ergebnisse der redriven Ausführung anzeigen. Im Ausführungsübersicht Abschnitt können Sie beispielsweise die Redrive Anzahl sehen, die angibt, wie oft eine Ausführung warredriven. Im Abschnitt Ereignisse können Sie die redrive zugehörigen Ausführungsereignisse sehen, die an die Ereignisse des ursprünglichen Ausführungsversuchs angehängt sind. Zum Beispiel das -`ExecutionRedriven`Ereignis.

Redriving -Ausführungen mit der API

Sie können redrive berechtigte Ausführungen mit der RedriveExecution-API verwenden. Diese API startet erfolglose Ausführungen von Standard-Workflows ab dem Schritt neu, der fehlgeschlagen, abgebrochen oder abgelaufen ist.

Führen Sie in der AWS Command Line Interface (AWS CLI) den folgenden Befehl für redrive eine erfolglose Ausführung des Zustandsautomaten aus. Denken Sie daran, den *kursiv gedruckten* Text durch Ihre ressourcenspezifischen Informationen zu ersetzen.

```
aws stepfunctions redrive-execution --execution-arn arn:aws:states:us-
east-2:123456789012:execution:myStateMachine:foo
```

Untersuchen von redriven Ausführungen

Sie können eine redriven Ausführung in der Konsole oder mit den APIs untersuchen: GetExecutionHistory und DescribeExecution.

Untersuchen von redriven Ausführungen auf der Konsole

1. Öffnen Sie die Step-Functions-Konsole und wählen Sie dann einen vorhandenen Zustandsautomaten aus, für den Sie redriven eine Ausführung haben.
2. Öffnen Sie die Seite Ausführungsdetails.

   Auf dieser Seite können Sie die Ergebnisse der redriven Ausführung anzeigen. Im Ausführungsübersicht Abschnitt können Sie beispielsweise die Redrive Anzahl sehen, die angibt, wie oft eine Ausführung warredriven. Im Abschnitt Ereignisse können Sie die redrive zugehörigen Ausführungsereignisse sehen, die an die Ereignisse des ursprünglichen Ausführungsversuchs angehängt sind. Zum Beispiel das -`ExecutionRedriven`Ereignis.

Untersuchen von redriven Ausführungen mithilfe von APIs

Wenn Sie redriven eine Zustandsautomaten-Ausführung haben, können Sie eine der folgenden APIs verwenden, um Details zur redriven Ausführung anzuzeigen. Denken Sie daran, den *kursiv gedruckten* Text durch Ihre ressourcenspezifischen Informationen zu ersetzen.

- GetExecutionHistory – Gibt den Verlauf der angegebenen Ausführung als Liste von Ereignissen zurück. Diese API gibt auch die Details zum redrive Versuch einer Ausführung zurück, falls verfügbar.

  AWS CLIFühren Sie in der den folgenden Befehl aus.

  ```
  aws stepfunctions get-execution-history --execution-arn arn:aws:states:us-
  east-2:123456789012:execution:myStateMachine:foo
  ```

- DescribeExecution – Stellt Informationen über die Ausführung eines Zustandsautomaten bereit. Dies kann der der Ausführung zugeordnete Zustandsautomat, die Ausführungseingabe und -ausgabe, redrive Ausführungsdetails, falls verfügbar, und relevante Ausführungsmetadaten sein.

  AWS CLIFühren Sie in der den folgenden Befehl aus.

```
aws stepfunctions describe-execution --execution-arn arn:aws:states:us-
east-2:123456789012:execution:myStateMachine:foo
```

Wiederholungsverhalten von redriven Ausführungen

Wenn Ihre redriven Ausführung einen Status der Aufgabe-, - Parallelloder Inline Map-Status erneut ausführt, für den Sie Wiederholungsversuche definiert haben, wird die Anzahl der Wiederholungsversuche für diese Zustände auf 0 zurückgesetzt. Dies ermöglicht die maximale Anzahl von Versuchen auf redrive. Für eine redriven Ausführung können Sie einzelne Wiederholungsversuche dieser Zustände mithilfe der Konsole verfolgen.

So untersuchen Sie die einzelnen Wiederholungsversuche in der Konsole

1. Wählen Sie auf der Seite Ausführungsdetails der Step-Functions-Konsole einen Status aus, der auf erneut versucht wurderedrive.
2. Wählen Sie die Registerkarte Wiederholungen und redrives aus.
3. Wählen Sie die

   ▶

   neben jedem Wiederholungsversuch aus, um dessen Details anzuzeigen. Wenn der Wiederholungsversuch erfolgreich war, können Sie die Ergebnisse in Ausgabe anzeigen, die in einem Dropdown-Feld angezeigt wird.

Die folgende Abbildung zeigt ein Beispiel für die Wiederholungsversuche, die für einen -Zustand im ursprünglichen Ausführungsversuch durchgeführt wurden, und für die redrives dieser Ausführung. In diesem Image werden drei Wiederholungsversuche im Original und in redrive Ausführungsversuchen durchgeführt. Die Ausführung ist beim vierten redrive Versuch erfolgreich und gibt eine Ausgabe von 16 zurück.

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   }
```

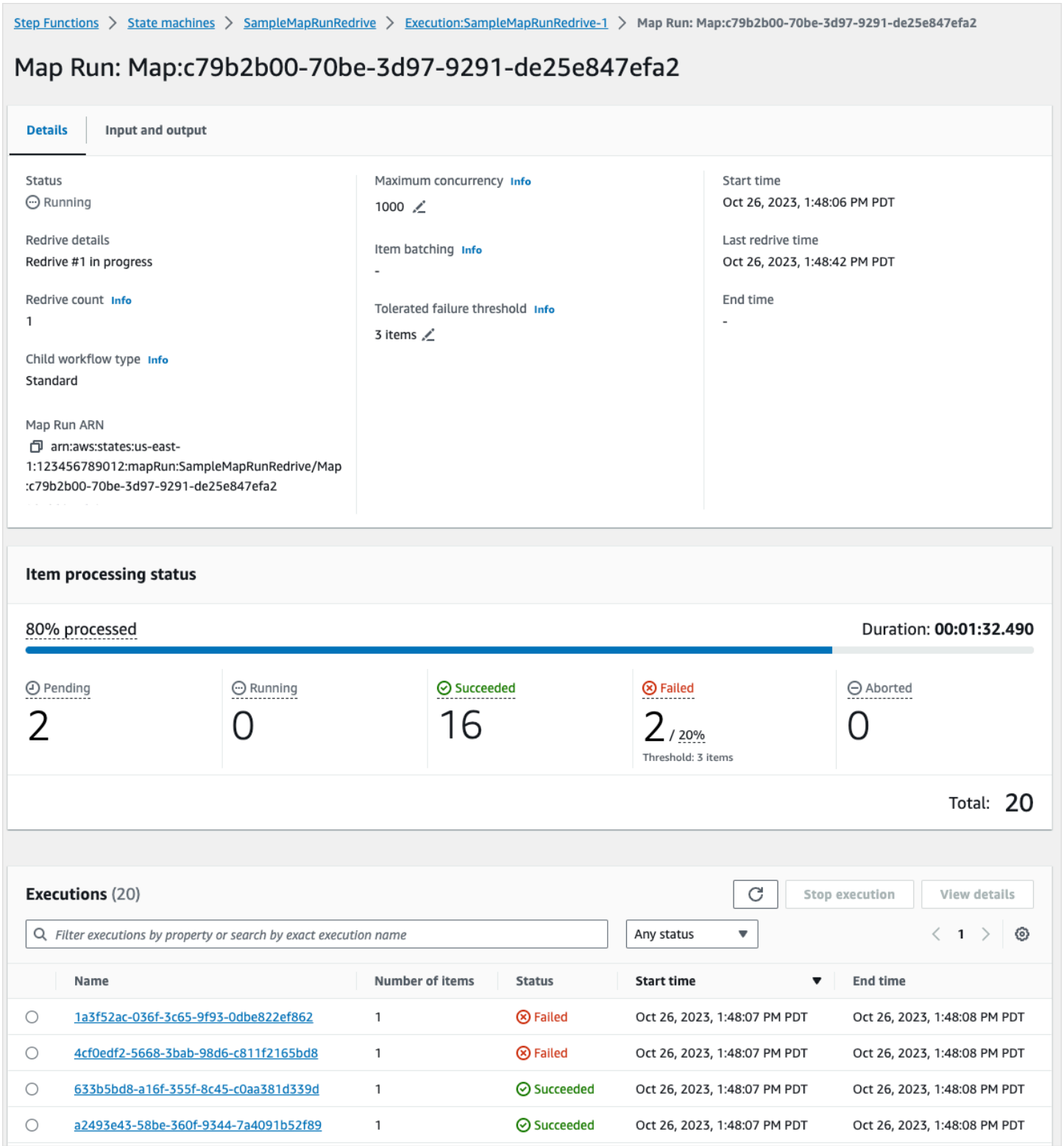
Formatted

Untersuchen der Map-Ausführung einer Ausführung des verteilten Map-Zustands

Wenn Sie einen -`Map`Status im verteilten Modus ausführen, erstellt Step Functions eine Map-Run-Ressource. Eine Map-Ausführung bezieht sich auf eine Reihe untergeordneter Workflow-Ausführungen, die ein Distributed Map-Status startet, sowie auf die Laufzeiteinstellungen, die diese Ausführungen steuern. Step Functions weist Ihrer Map-Ausführung einen Amazon-Ressourcennamen (ARN) zu. Sie können einen Map Run in der Step Functions-Konsole untersuchen. Sie können auch die `DescribeMapRun` -API-Aktion aufrufen. Ein Map Run gibt auch Metriken an aus CloudWatch.

Die Step-Functions-Konsole bietet eine Seite mit den Details zur Zuordnung von Ausführungsdetails, auf der alle Informationen zu einer Ausführung des verteilten Zuordnungsstatus angezeigt werden. Sie können beispielsweise den Status der Ausführung des Status der verteilten Zuordnung , den ARN der Zuordnungsausführung und den Status der in den untergeordneten Workflow-Ausführungen verarbeiteten Elemente anzeigen, die mit dem Status der verteilten Zuordnung gestartet wurden. Sie können auch eine Liste aller untergeordneten Workflow-Ausführungen anzeigen und auf deren Details zugreifen. Wenn Ihre Map-Ausführung warredriven, können Sie außerdem die redrive Details der Map-Ausführung im Zusammenfassung der Ausführung von Map Run Abschnitt sehen. Zum Beispiel das letzte redrive Mal . Die Konsole zeigt diese Informationen in einem Dashboard-Format an.

Die Seite Details zur Zuordnung von Ausführung enthält die folgenden Abschnitte:



Inhalt

- Status der Elementverarbeitung
- Auflistung der Ausführungen
- RedrivingKarte läuft
  - RedriveEignung für untergeordnete Workflows in einem Map Run
  - redriveVerhalten bei der Ausführung eines untergeordneten Workflows
  - Bei Map Run verwendete Eingabeszenarien redrive
  - IAM-Erlaubnis für redrive einen Map Run
  - RedrivingKarte In der Konsole ausführen
  - RedrivingMap Run mithilfe der API

Zusammenfassung der Ausführung von Map Run

Der Abschnitt Zusammenfassung der Map-Ausführungsausführung wird oben auf der Seite Details zur Map-Ausführung angezeigt. Dieser Abschnitt bietet Ihnen einen Überblick über die Ausführungsdetails des Status Verteilte Zuordnung . Diese Informationen sind auf die folgenden Registerkarten aufgeteilt:

Details

Zeigt Informationen an, z. B. den Ausführungsstatus des Status „Verteilte Zuordnung“, den ARN „Zuordnungsausführung“ und den Typ der untergeordneten Workflow-Ausführungen, die mit dem Status „Verteilte Zuordnung“ gestartet wurden. Sie können zusätzliche Konfigurationen anzeigen, z. B. den Schwellenwert für tolerierende Fehler für die Map-Ausführung und die maximale Gleichzeitigkeit, die für untergeordnete Workflow-Ausführungen angegeben ist. Sie können diese Konfigurationen auch bearbeiten.

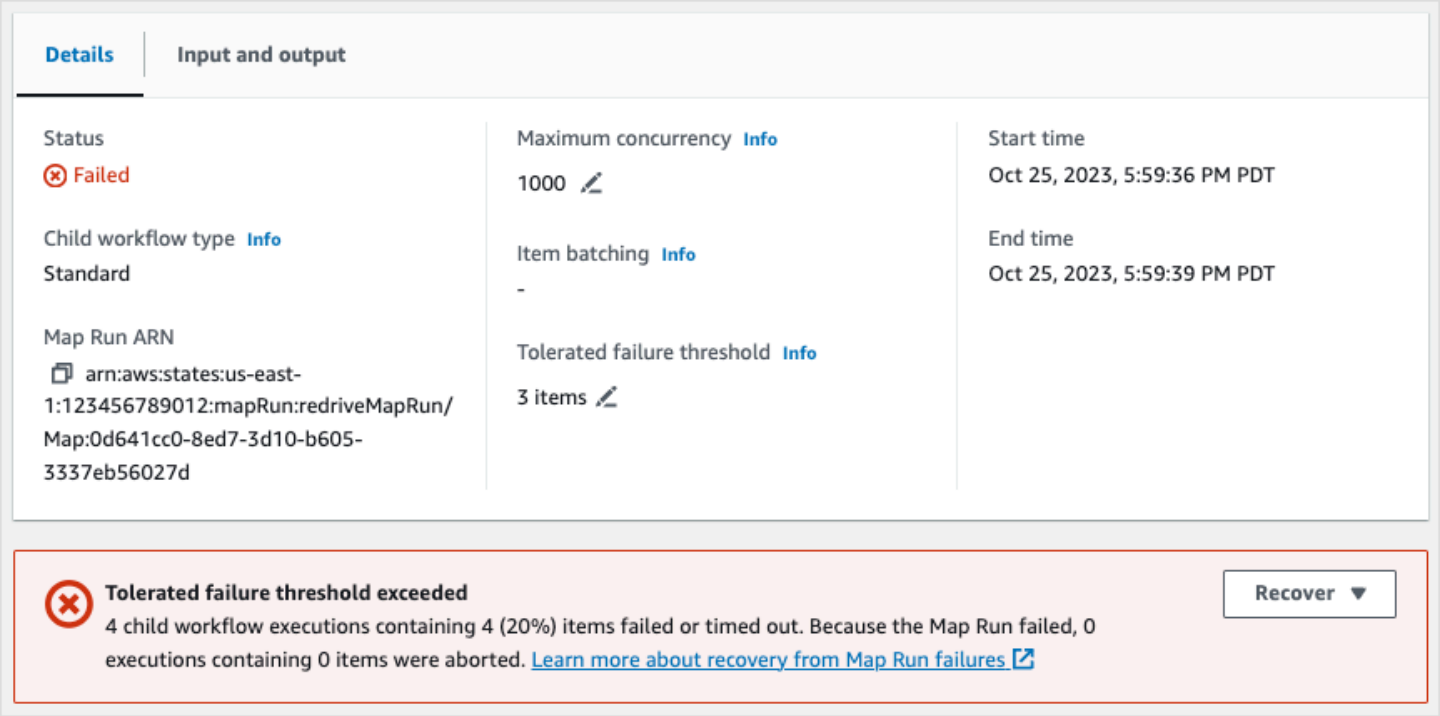
Eingabe und Ausgabe

Zeigt die vom Status Distributed Map empfangene Eingabe und die entsprechende Ausgabe an, die generiert wird. Sie können beispielsweise den Eingabedatensatz und seinen Speicherort sowie die Eingabefilter anzeigen, die auf die einzelnen Datenelemente in diesem Datensatz angewendet werden. Wenn Sie die Ausgabe der Ausführung des Status Distributed Map exportieren, zeigt diese Registerkarte den Pfad zum Amazon S3-Bucket an, der die Ausführungsergebnisse enthält. Andernfalls werden Sie auf die Seite Ausführungsdetails des übergeordneten Workflows weitergeleitet, um die Ausführungsausgabe anzuzeigen.

Fehlermeldung

Wenn Ihre Map-Ausführung fehlgeschlagen ist, zeigt die Seite Details zur Map-Ausführung eine Fehlermeldung mit dem Grund für den Fehler an.

Über die Dropdown-Schaltfläche Wiederherstellen in dieser Fehlermeldung können Sie entweder redrive die erfolglosen untergeordneten Workflow-Ausführungen starten, die mit dieser Map-Ausführung gestartet wurden, oder eine neue Ausführung des übergeordneten Workflows starten. Weitere Informationen finden Sie unter RedrivingKarte läuft.



Status der Elementverarbeitung

Im Abschnitt Status der Elementverarbeitung wird der Status der in einer Kartenausführung verarbeiteten Elemente angezeigt. Ausstehend zeigt beispielsweise an, dass eine untergeordnete Workflow-Ausführung noch nicht mit der Verarbeitung des Elements begonnen hat.

Der Elementstatus hängt vom Status der untergeordneten Workflow-Ausführungen ab, die die Elemente verarbeiten. Wenn eine untergeordnete Workflow-Ausführung fehlgeschlagen ist, eine Zeitüberschreitung auftritt oder wenn ein Benutzer die Ausführung abbricht, erhält Step Functions keine Informationen über das Verarbeitungsergebnis der Elemente innerhalb dieser untergeordneten Workflow-Ausführung. Alle von dieser Ausführung verarbeiteten Elemente teilen sich den Status der untergeordneten Workflow-Ausführung.

Angenommen, Sie möchten 100 Elemente in zwei untergeordneten Workflow-Ausführungen verarbeiten, wobei jede Ausführung einen Stapel von 50 Elementen verarbeitet. Wenn eine der Ausführungen fehlschlägt und die andere erfolgreich ist, haben Sie 50 erfolgreiche und 50 fehlgeschlagene Elemente.

In der folgenden Tabelle werden die Arten von Verarbeitungsstatus erläutert, die für alle Elemente verfügbar sind:

Status	Description
Ausstehend	Gibt ein Element an, dass die untergeordnete Workflow-Ausführung nicht mit der Verarbeitung begonnen hat. Wenn eine Map-Ausführung angehalten wird, fehlschlägt oder ein Benutzer die Ausführung abbricht, bevor die Verarbeitung eines Elements beginnt, bleibt das Element im Status Ausstehend. Wenn eine Map-Ausführung beispielsweise fehlschlägt und 10 Elemente noch verarbeitet werden müssen, verbleiben diese 10 Elemente im Status Ausstehend.
In Ausführung	Gibt ein Element an, das derzeit von der untergeordneten Workflow-Ausführung verarbeitet wird.
Erfolgreich	Zeigt an, dass die untergeordnete Workflow-Ausführung das Element erfolgreich verarbeitet hat. Eine erfolgreiche untergeordnete Workflow-Ausführung darf keine fehlgeschlagenen Elemente enthalten. Wenn ein Element im Datensatz während der Ausführung fehlschlägt, schlägt die gesamte untergeordnete Workflow-Ausführung fehl.

Status	Description
Fehlgeschlagen	Zeigt an, dass die untergeordnete Workflow-Ausführung das Element entweder nicht verarbeiten konnte oder dass die Ausführung abgelaufen ist. Wenn ein von einer untergeordneten Workflow-Ausführung verarbeitetes Element fehlschlägt, schlägt die gesamte untergeordnete Workflow-Ausführung fehl. Betrachten Sie beispielsweise eine untergeordnete Workflow-Ausführung, die 1000 Elemente verarbeitet hat. Wenn ein Element in diesem Datensatz während der Ausführung fehlschlägt, betrachtet Step Functions die gesamte untergeordnete Workflow-Ausführung als fehlgeschlagen. Wenn Sie redrive eine Map-Ausführung ausführen, wird die Anzahl der Elemente mit diesem Status auf 0 zurückgesetzt.

Status	Description
Abgebrochen	Zeigt an, dass die untergeordnete Workflow-Ausführung mit der Verarbeitung des Elements begonnen hat, aber entweder der Benutzer die Ausführung abgebrochen hat oder Step Functions die Ausführung gestoppt hat, weil die Zuordnungsausführung fehlgeschlagen ist. Betrachten Sie beispielsweise eine Ausführun g eines laufenden untergeordneten Workflows , die 50 Elemente verarbeitet. Wenn die Map-Ausführung aufgrund eines Fehlers oder weil ein Benutzer die Ausführung abgebrochen hat, wird die untergeordnete Workflow-Ausführung und der Status aller 50 Elemente in Abgebroch en geändert. Wenn Sie eine untergeordnete Workflow-Ausführung vom Typ Express verwenden, können Sie die Ausführung nicht beenden. Wenn Sie redrive eine Map-Ausführung ausführen, die untergeordnete Workflow-Ausführungen vom Typ Express startet, wird die Anzahl der Elemente mit diesem Status auf 0 zurückgesetzt. Dies liegt daran, dass untergeordnete Express-Workflows mit der StartExecution -API-Aktion neu gestartet werden, anstatt zu seinredriven.

Auflistung der Ausführungen

Im Abschnitt Ausführungen werden alle untergeordneten Workflow-Ausführungen für eine bestimmte Map-Ausführung aufgelistet. Verwenden Sie das Feld Nach exaktem Ausführungsnamen suchen, um nach einer bestimmten untergeordneten Workflow-Ausführung zu suchen. Sie können auch das Dropdown-Menü Beliebiger Status verwenden, um die Verlauf der untergeordneten Workflow-

Ausführung nach ihrem Status zu filtern. Um Details zu einer bestimmten Ausführung anzuzeigen, wählen Sie eine untergeordnete Workflow-Ausführung aus der Liste aus und klicken Sie auf die Schaltfläche Details anzeigen, um die Seite Ausführungsdetails zu öffnen.

> **Important**
>
> Die Aufbewahrungsrichtlinie für untergeordnete Workflow-Ausführungen beträgt 90 Tage. Abgeschlossene untergeordnete Workflow-Ausführungen, die älter als dieser Aufbewahrungszeitraum sind, werden in der Tabelle Ausführungen nicht angezeigt. Dies gilt auch dann, wenn der Status Verteilte Zuordnung oder der übergeordnete Workflow länger als die Aufbewahrungsfrist ausgeführt wird. Sie können Ausführungsdetails, einschließlich der Ergebnisse, dieser untergeordneten Workflow-Ausführungen anzeigen, wenn Sie die Ausgabe des Status der verteilten Zuordnung mithilfe von in einen Amazon S3-Bucket exportieren`ResultWriter`.

> **Tip**
>
> Wählen Sie die Schaltfläche Aktualisieren,
>
> 
>
> um die aktuelle Liste aller untergeordneten Workflow-Ausführungen anzuzeigen.

RedrivingKarte läuft

Sie können erfolglose untergeordnete Workflow-Ausführungen in einem Map Run durch redrivingIhren übergeordneten Workflow neu starten. Ein redriven übergeordneter Workflow mit redrives allen erfolglosen Status, einschließlich Distributed Map. Ein übergeordneter Workflow gibt den Status „Nicht erfolgreich“ wieder aus, wenn es kein `<stateType>Exited` Ereignis gibt, das dem `<stateType>Entered` Ereignis für einen Status entspricht, in dem der übergeordnete Workflow seine Ausführung abgeschlossen hat. Wenn der Ereignisverlauf beispielsweise das Ereignis für ein `MapStateExited` `MapStateEntered` Ereignis nicht enthält, können Sie redrive den übergeordneten Workflow für redrive alle erfolglosen untergeordneten Workflow-Ausführungen in der Map Run übernehmen.

Ein Map Run wird entweder nicht gestartet oder schlägt beim ursprünglichen Ausführungsversuch fehl, wenn der Zustandsmaschine nicht über die erforderliche Zugriffsberechtigung für den

ItemReaderResultWriter, oder beides verfügt. Wenn der Map Run beim ursprünglichen Ausführungsversuch des übergeordneten Workflows nicht gestartet wurde, startet redriving der übergeordnete Workflow den Map Run zum ersten Mal. Um dieses Problem zu beheben, fügen Sie Ihrer State-Machine-Rolle und dann redrive dem übergeordneten Workflow die erforderlichen Berechtigungen hinzu. Wenn Sie redrive den übergeordneten Workflow verwenden, ohne die erforderlichen Berechtigungen hinzuzufügen, versucht er, einen neuen Map Run-Lauf zu starten, der erneut fehlschlägt. Informationen zu den Berechtigungen, die Sie möglicherweise benötigen, finden Sie unterIAM-Richtlinien für die Verwendung des Distributed Map-Status.

Themen

- RedriveEignung für untergeordnete Workflows in einem Map Run
- redriveVerhalten bei der Ausführung eines untergeordneten Workflows
- Bei Map Run verwendete Eingabeszenarien redrive
- IAM-Erlaubnis für redrive einen Map Run
- RedrivingKarte In der Konsole ausführen
- RedrivingMap Run mithilfe der API

RedriveEignung für untergeordnete Workflows in einem Map Run

Sie können redrive erfolglose untergeordnete Workflow-Ausführungen in einem Map Run durchführen, wenn die folgenden Bedingungen erfüllt sind:

- Sie haben die Ausführung des übergeordneten Workflows am oder nach dem 15. November 2023 gestartet. Ausführungen, die Sie vor diesem Datum gestartet haben, kommen nicht in Frage. redrive
- Sie haben das feste Limit redrives von 1000 für einen bestimmten Map Run nicht überschritten. Wenn du dieses Limit überschritten hast, erhältst du die `States.Runtime` Fehlermeldung.
- Der übergeordnete Workflow istredrivable. Wenn der übergeordnete Workflow dies nicht istredrivable, können Sie redrive den untergeordneten Workflow nicht in einem Map Run ausführen. Weitere Informationen zur redrive Eignung eines Workflows finden Sie unter. Redrive Berechtigung für erfolglose Ausführungen
- Die untergeordneten Workflow-Ausführungen des Typs Standard in Ihrem Map Run haben das Limit von 25.000 Ausführungsereignissen im Verlauf nicht überschritten. Untergeordnete Workflow-Ausführungen, die das Limit für den Ereignisverlauf überschritten haben, werden auf den Schwellenwert für tolerierte Fehler angerechnet und als fehlgeschlagen betrachtet. Weitere

Informationen zur redrive Eignung einer Ausführung finden Sie unter. Redrive Berechtigung für erfolglose Ausführungen

In den folgenden Fällen wird ein neuer Map Run gestartet und der bestehende Map Run wird nicht redriven gestartet, auch wenn der Map Run beim ursprünglichen Ausführungsversuch fehlgeschlagen ist:

- Map Run ist aufgrund des `States.DataLimitExceeded` Fehlers fehlgeschlagen.
- Map Run ist aufgrund des JSON-Dateninterpolationsfehlers fehlgeschlagen,. `States.Runtime` Sie haben beispielsweise einen nicht vorhandenen JSON-Knoten in ausgewählt. OutputPath

Ein Map-Run kann auch dann weiter ausgeführt werden, wenn der übergeordnete Workflow beendet oder das Timeout überschritten wurde. In diesen Szenarien passiert redrive das nicht sofort:

- Map Run storniert möglicherweise immer noch laufende untergeordnete Workflow-Ausführungen vom Typ Standard oder wartet darauf, dass untergeordnete Workflow-Ausführungen vom Typ Express ihre Ausführungen abgeschlossen haben.
- Map Run schreibt möglicherweise immer noch Ergebnisse in dasResultWriter, wenn Sie es für den Export von Ergebnissen konfiguriert haben.

In diesen Fällen schließt der laufende Map Run seine Operationen ab, bevor er versucht, dies zu tunredrive.

redriveVerhalten bei der Ausführung eines untergeordneten Workflows

Die redriven untergeordneten Workflow-Ausführungen in einem Map Run weisen das in der folgenden Tabelle beschriebene Verhalten auf.

Untergeordneter Express-Arbeitsablauf	Standardmäßiger Arbeitsablauf für Kinder
Alle untergeordneten Workflow-Ausführungen, die beim ursprünglichen Ausführungsversuch fehlgeschlagen sind oder deren Timeout überschritten wurde, werden mithilfe der StartExecutionAPI-Aktion gestartet. Der erste Status in ItemProcessor wird zuerst ausgeführt.	Alle untergeordneten Workflow-Ausführungen, die beim ursprünglichen Ausführungsversuch fehlgeschlagen sind, deren Timeout überschri tten wurde oder die abgebrochen wurden, redriven verwenden die RedriveExecutionAPI-Aktion. Diese untergeordneten Workflows stammen redriven aus dem letzten Status

Untergeordneter Express-Arbeitsablauf	Standardmäßiger Arbeitsablauf für Kinder
	ItemProcessor , in dem sie nicht erfolgreich ausgeführt wurden.
Fehlgeschlagene Ausführungen können immer sein. redriven Das liegt daran, dass untergeordnete Express-Workflow-Ausführungen immer als neue Ausführung mithilfe der StartExecution API-Aktion gestartet werden.	Untergeordnete Standard-Workflow-Ausführungen können nicht immer erfolglos sein. redriven Wenn eine Ausführung nicht erfolgtredrivable, wird sie nicht erneut versucht. Der letzte Fehler oder die letzte Ausgabe der Ausführung ist dauerhaft. Dies ist möglich, wenn eine Ausführung mehr als 25.000 historische Ereignisse umfasst oder der redrivable Zeitraum von 14 Tagen abgelaufen ist. Eine untergeordnete Standardworkflow-Ausführung ist möglicherweise nicht möglich, redrivable wenn die Ausführung des übergeordneten Workflows innerhalb von 14 Tagen abgeschlossen wurde, die Ausführung des untergeordneten Workflows jedoch vor 14 Tagen.
Untergeordnete Express-Workflow-Ausführungen verwenden denselben Ausführungs-ARN wie der ursprüngliche Ausführungsversuch, aber Sie können ihre Person nicht eindeutig identifizieren. redrives	Standardausführungen untergeordneter Workflows verwenden denselben Ausführungs-ARN wie der ursprüngliche Ausführungsversuch. Sie können die Person redrives in der Konsole und mithilfe von APIs wie GetExecutionHistoryund eindeutig identifizieren. DescribeExecution Weitere Informationen finden Sie unter Untersuchen von redriven Ausführungen.

Wenn Sie über redriven einen Map Run verfügen und dieser sein Parallelitätslimit erreicht hat, gehen die untergeordneten Workflow-Ausführungen in dieser Map Run-Ausführung in den Status Ausstehend über. Der Ausführungsstatus von Map Run geht ebenfalls in den Status Ausstehend

redrive über. Bis das angegebene Parallelitätslimit die Ausführung weiterer untergeordneter Workflow-Ausführungen ermöglicht, verbleibt die Ausführung im Status Ausstehend redrive.

Nehmen wir beispielsweise an, dass das Parallelitätslimit für die verteilte Map in Ihrem Workflow 3000 beträgt und die Anzahl der erneut auszuführenden untergeordneten Workflows 6000 beträgt. Dies führt dazu, dass 3000 untergeordnete Workflows parallel ausgeführt werden, während die verbleibenden 3000 Workflows im Status Pending Redrive verbleiben. Nachdem die Ausführung des ersten Batches von 3000 untergeordneten Workflows abgeschlossen ist, werden die verbleibenden 3000 untergeordneten Workflows ausgeführt.

Wenn die Ausführung eines Map Run abgeschlossen oder abgebrochen wurde, wird die Anzahl der Ausführungen untergeordneter Workflows im redrive Status Ausstehend auf 0 zurückgesetzt.

Bei Map Run verwendete Eingabeszenarien redrive

Je nachdem, wie Sie beim ursprünglichen Ausführungsversuch Eingaben für die Distributed Map gemacht haben, verwendet ein redriven Map Run die Eingabe wie in der folgenden Tabelle beschrieben.

Eingabe beim ursprünglichen Ausführungsversuch	Bei Map Run verwendete Eingabe redrive
Eingabe, die aus einem vorherigen Status oder der Ausführungseingabe übergeben wurde.	Der redriven Map Run verwendet dieselbe Eingabe.
Die Eingabe wurde mit übergeben ItemReader und der Map Run hat die Ausführung des untergeordneten Workflows nicht gestartet, da eine der folgenden Bedingungen zutrifft: • Map Run ist mit dem `States.ItemReaderFailed` Fehler fehlgeschlagen. • Map Run ist mit dem `States.ResultWriterFailed` Fehler fehlgeschlagen. • Bei der Ausführung des übergeordneten Workflows wurde das Timeout überschritten	Der redriven Map Run verwendet die Eingabe im Amazon S3 S3-Bucket.

Eingabe beim ursprünglichen Ausführun gsversuch	Bei Map Run verwendete Eingabe redrive
oder sie wurde abgebrochen, bevor Map Run gestartet wurde.	
Eingabe wurde mit übergeben ItemReader. Der Map Run schlug fehl, nachdem untergeordnete Workflow-Ausführungen gestartet oder versucht wurden, sie zu starten.	Der redriven Map Run verwendet dieselbe Eingabe wie beim ursprünglichen Ausführun gsversuch.

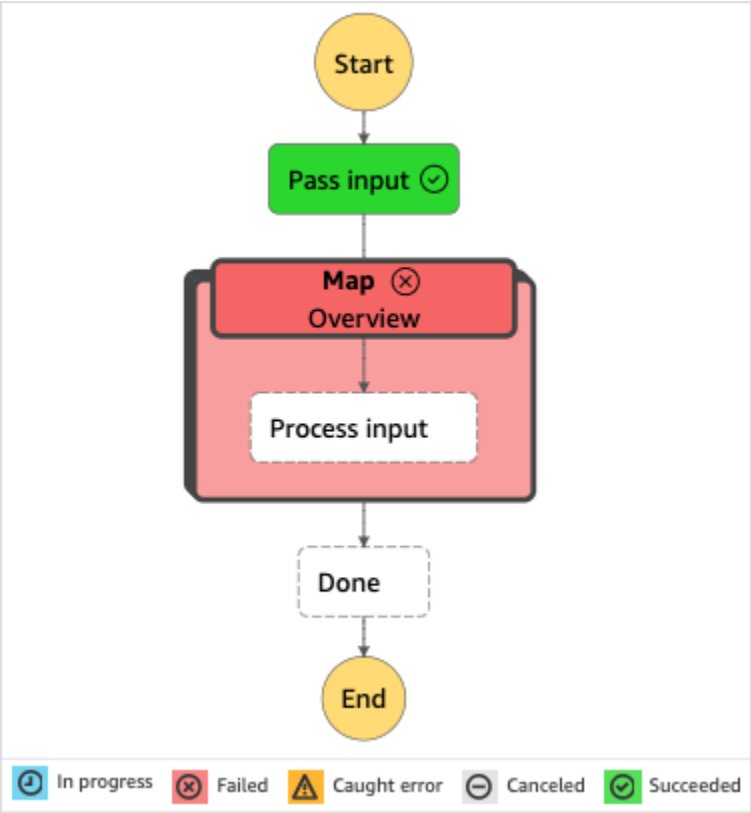
IAM-Erlaubnis für redrive einen Map Run

Step Functions benötigt die entsprechende Erlaubnis für redrive einen Map Run. Das folgende Beispiel für eine IAM-Richtlinie gewährt Ihrem Zustandsmaschine die geringste Berechtigung, die für redriving einen Map Run erforderlich ist. Denken Sie daran, den *kursiv geschriebenen* Text durch Ihre ressourcenspezifischen Informationen zu ersetzen.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:RedriveExecution"
      ],
      "Resource": "arn:aws:states:us-
east-2:123456789012:execution:myStateMachine/myMapRunLabel:*"
    }
  ]
}
```

RedrivingKarte In der Konsole ausführen

Die folgende Abbildung zeigt das Ausführungsdiagramm einer Zustandsmaschine, die eine Distributed Map enthält. Diese Ausführung ist fehlgeschlagen, weil der Map Run fehlgeschlagen ist. redriveZum Map Run müssen Sie redrive den übergeordneten Workflow verwenden.



Zu redrive einem Map Run von der Konsole aus

1. Öffnen Sie die Step Functions Functions-Konsole und wählen Sie dann eine vorhandene Zustandsmaschine aus, die eine verteilte Map enthält, deren Ausführung fehlgeschlagen ist.
2. Wählen Sie auf der Detailseite der Zustandsmaschine unter Ausführungen eine Instanz dieser Zustandsmaschine aus, deren Ausführung fehlgeschlagen ist.
3. Wählen Sie Redrive.
4. Wählen RedriveSie im Dialogfeld RedriveAusführung aus.

> **Tip**
>
> Sie können redrive einen Map-Lauf auch auf der Seite „Ausführungsdetails“ oder „Map-Run-Details“ ausführen.
> Wenn Sie sich auf der Seite mit den Ausführungsdetails befinden, führen Sie für die Ausführung einen redrive der folgenden Schritte aus:
>
> - Wählen Sie „Wiederherstellen“ und dann „Fehler“ Redrive aus.
> - Wählen Sie Aktionen und dann aus Redrive.

> Wenn Sie sich auf der Seite „Kartenlaufdetails“ befinden, wählen Sie „Wiederherstellen“ und dann „Fehler“ Redrive aus.

Beachten Sie, dass dieselbe Zustandsmaschinen-Definition und derselben ARN redrive verwendet werden. Es setzt die Ausführung ab dem Schritt fort, der beim ursprünglichen Ausführungsversuch fehlgeschlagen ist. In diesem Beispiel handelt es sich um den Schritt Distributed Map mit dem Namen Map und den darin enthaltenen Eingabeschritt Process. Nach dem Neustart der erfolglosen untergeordneten Workflow-Ausführungen von Map Run redrive wird die Ausführung für den Schritt Fertig fortgesetzt.

5. Wählen Sie auf der Seite mit den Ausführungsdetails die Option Map Run aus, um die Details des redriven Map Run anzuzeigen.

   Auf dieser Seite können Sie sich die Ergebnisse der redriven Ausführung ansehen. In Zusammenfassung der Ausführung von Map Run diesem Abschnitt können Sie beispielsweise die RedriveAnzahl sehen, die angibt, wie oft der Map Run durchgeführt wurderedriven. Im Abschnitt Ereignisse können Sie die redrive zugehörigen Ausführungsereignisse an die Ereignisse des ursprünglichen Ausführungsversuchs angehängt sehen. Zum Beispiel das `MapRunRedriven` Ereignis.

Nachdem Sie redriven einen Map Run ausgeführt haben, können Sie dessen redrive Details in der Konsole oder mithilfe der API-Aktionen GetExecutionHistoryund der DescribeExecutionAPI-Aktionen überprüfen. Weitere Hinweise zur Untersuchung einer redriven Ausführung finden Sie unterUntersuchen von redriven Ausführungen.

RedrivingMap Run mithilfe der API

Sie können redrive einen geeigneten Map Run mithilfe der RedriveExecutionAPI für den übergeordneten Workflow ausführen. Diese API startet erfolglose untergeordnete Workflow-Ausführungen in einem Map Run neu.

Führen Sie in AWS Command Line Interface (AWS CLI) den folgenden Befehl für redrive eine erfolglose State-Machine-Ausführung aus. Denken Sie daran, den *`kursiv geschriebenen`* Text durch Ihre ressourcenspezifischen Informationen zu ersetzen.

```
aws stepfunctions redrive-execution --execution-arn arn:aws:states:us-
east-2:123456789012:execution:myStateMachine:foo
```

Nachdem Sie redriven einen Map Run ausgeführt haben, können Sie dessen redrive Details in der Konsole oder mithilfe der API-Aktion überprüfen. DescribeMapRun Um die redrive Details von Standard-Workflow-Ausführungen in einem Map-Run zu untersuchen, können Sie die DescribeExecutionAPI-Aktion GetExecutionHistoryoder verwenden. Weitere Hinweise zur Untersuchung einer redriven Ausführung finden Sie unterthe section called “Untersuchen von redriven Ausführungen”.

Sie können die redrive Details der Express-Workflow-Ausführungen in einem Map Run in der Step Functions Functions-Konsole überprüfen, wenn Sie die Protokollierung für den übergeordneten Workflow aktiviert haben. Weitere Informationen finden Sie unter Protokollierung mitCloudWatchProtokolle.

Fehlerbehandlung in Step Functions

In allen Zuständen, mit Ausnahme `Pass` `Wait` von Staaten, kann es zu Laufzeitfehlern kommen. Fehler können aus verschiedenen Gründen auftreten, z. B. in den folgenden Beispielen:

- Probleme bei der Definition des Zustandsautomaten (z. B. keine übereinstimmende Regel in einem `Choice`-Zustand)
- Aufgabenausfälle (z. B. eine Ausnahme in einer AWS Lambda-Funktion)
- Vorübergehende Probleme (z. B. Netzwerkpartitionsereignisse)

Standardmäßig gilt Folgendes: Wenn ein Zustand einen Fehler meldet, sorgt AWS Step Functions dafür, dass die Ausführung vollständig fehlschlägt.

> **Tip**
>
> Ein Beispiel für einen Workflow, der eine Fehlerbehandlung beinhaltetAWS-Konto, finden Sie in Modul 8 — Fehlerbehandlung von The AWS Step Functions Workshop.

Themen

- Namen der Fehler

- Wiederholter Versuch nach einem Fehler
- Fallback-Staaten
- Beispiele für Zustandsmaschinen mit Retry und Catch

Namen der Fehler

Step Functions identifiziert Fehler in der Sprache der Amazon-Staaten mithilfe von Zeichenfolgen, bei denen Groß- und Kleinschreibung beachtet wird, die als Fehlernamen bezeichnet werden. Die Amazon States Language definiert eine Reihe von integrierten Zeichenketten, die bekannte Fehler benennen, die alle mit dem `States.` Präfix beginnen.

`States.ALL`

Ein Platzhalter, der mit jedem bekannten Fehlernamen übereinstimmt.

> **Note**
>
> Dieser Fehlertyp kann den `States.DataLimitExceeded` Terminalfehlertyp und die Laufzeitfehlertypen nicht abfangen. Weitere Hinweise zu diesen Fehlertypen finden Sie unter `States.DataLimitExceeded`und `States.Runtime`.

`States.DataLimitExceeded`

Step Functions meldet unter den folgenden Bedingungen eine `States.DataLimitExceeded` Ausnahme:

- Wenn die Ausgabe eines Connectors das Kontingent für die Payload-Größe überschreitet.
- Wenn die Ausgabe eines Zustands größer als das Payload-Größenkontingent ist.
- Wenn nach der `Parameters` Verarbeitung die Eingabe eines Zustands das Payload-Größenkontingent überschreitet.

Weitere Informationen zu Kontingenten finden Sie unterKontingente.

> **Note**
>
> Dies ist ein Terminalfehler, der durch den `States.ALL` Fehlertyp nicht abgefangen werden kann.

`States.ExceedToleratedFailureThreshold`

Ein `Map` Status ist fehlgeschlagen, weil die Anzahl der ausgefallenen Elemente den in der Zustandsmaschinen-Definition angegebenen Schwellenwert überschritten hat. Weitere Informationen finden Sie unter Schwellenwert für tolerierte Fehler im Distributed-Map-Status.

`States.HeartbeatTimeout`

Ein `Task` Status konnte für einen Zeitraum, der länger als der `HeartbeatSeconds` Wert war, keinen Heartbeat senden.

> **Note**
>
> Dieser Fehler ist nur in den `Retry` Feldern `Catch` und verfügbar.

`States.IntrinsicFailure`

Ein Versuch, eine systeminterne Funktion innerhalb einer Payload-Vorlage aufzurufen, ist fehlgeschlagen.

`States.ItemReaderFailed`

Ein `Map` Status ist fehlgeschlagen, weil er nicht aus der im Feld angegebenen Elementquelle lesen konnte. `ItemReader` Weitere Informationen finden Sie unter `ItemReader`.

`States.NoChoiceMatched`

Ein `Choice` Status konnte die Eingabe nicht mit den in der Auswahlregel definierten Bedingungen abgleichen, und es wurde kein Standardübergang angegeben.

`States.ParameterPathFailure`

Der Versuch, ein Feld innerhalb eines `Parameters` Statusfeldes zu ersetzen, dessen Name mit der `.$` Verwendung eines Pfads endet, schlägt fehl.

`States.Permissions`

Ein `Task` Status schlug fehl, weil er nicht über ausreichende Rechte verfügte, um den angegebenen Code auszuführen.

`States.ResultPathMatchFailure`

Step Functions konnte das `ResultPath` Feld eines Bundesstaates nicht auf die Eingabe anwenden, die der Status empfangen hat.

`States.ResultWriterFailed`

Ein `Map` Status ist fehlgeschlagen, weil er keine Ergebnisse in das im `ResultWriter` Feld angegebene Ziel schreiben konnte. Weitere Informationen finden Sie unter `ResultWriter`.

`States.Runtime`

Eine Ausführung ist aufgrund einer Ausnahme fehlgeschlagen, die nicht verarbeitet werden konnte. Oft werden diese durch Laufzeitfehler verursacht, z. B. durch den Versuch, `InputPath` oder `OutputPath` auf eine JSON-Nutzlast gleich Null anzuwenden. Ein `States.Runtime` Fehler kann nicht behoben werden und führt immer dazu, dass die Ausführung fehlschlägt. Ein erneuter Versuch oder ein catch On `States.ALL` deckt keine `States.Runtime` Fehler ab.

`States.TaskFailed`

Ein `Task`-Zustand ist während der Ausführung fehlgeschlagen. Wenn es in einem Wiederholungsversuch oder catch verwendet wird, `States.TaskFailed` fungiert es als Platzhalter, der jedem bekannten Fehlernamen entspricht, mit Ausnahme von. `States.Timeout`

`States.Timeout`

Ein `Task`-Zustand dauerte entweder länger als im Wert `TimeoutSeconds` festgelegt oder er hat für längere Zeit kein Heartbeat gesendet als im Wert `HeartbeatSeconds` festgelegt.

Wenn eine Zustandsmaschine länger als der angegebene `TimeoutSeconds` Wert läuft, schlägt die Ausführung außerdem mit einem `States.Timeout` Fehler fehl.

Zustände können Fehler mit anderen Namen melden. Diese Fehlernamen dürfen jedoch nicht mit dem `States.` Präfix beginnen.

Stellen Sie als bewährte Methode sicher, dass der Produktionscode AWS Lambda-Service-Ausnahmen (`Lambda.ServiceException` und `Lambda.SdkClientException`) verarbeiten kann. Weitere Informationen finden Sie unter Lambda-Serviceausnahmen behandeln.

> **Note**
>
> Unbehandelte Fehler in Lambda werden wie `Lambda.Unknown` in der Fehlerausgabe gemeldet. Dazu gehören out-of-memory Fehler und Funktions-Timeouts. Sie können nach, oder abgleichen `Lambda.Unknown`States.ALL, `States.TaskFailed` um diese Fehler zu behandeln. Wenn Lambda die maximale Anzahl von Aufrufen erreicht, lautet

der Fehler. `Lambda.TooManyRequestsException` Weitere Informationen zu Lambda-Funktionsfehlern finden Sie unter Fehlerbehandlung und automatische Wiederholungen im AWS LambdaEntwicklerhandbuch.

Wiederholter Versuch nach einem Fehler

`Task`, `Parallel`, und `Map` Staaten können ein benanntes Feld haben`Retry`, dessen Wert ein Array von Objekten sein muss, die als Retrier bezeichnet werden. Eine einzelner Retrier steht für eine bestimmte Anzahl von erneuten Versuchen, in der Regel in zunehmenden Zeitintervallen.

Wenn einer dieser Zustände einen Fehler meldet und es ein `Retry` Feld gibt, durchsucht Step Functions die Retrier in der Reihenfolge, die im Array aufgeführt ist. Wenn der Name des Fehlers im Wert eines `ErrorEquals` Retrier-Felds erscheint, versucht die State Machine erneut, wie im Feld definiert. `Retry`

Wenn Ihre redriven Ausführung einenStatus der Aufgabe, Parallel oder Inline-Map-Status wiederholt, für den Sie Wiederholungen definiert haben, wird die Anzahl der Wiederholungsversuche für diese Zustände auf 0 zurückgesetzt, um die maximale Anzahl von Versuchen zu ermöglichen. redrive Bei einer redriven Ausführung können Sie einzelne Wiederholungsversuche in diesen Zuständen mithilfe der Konsole verfolgen. Weitere Informationen finden Sie unter Wiederholungsverhalten von redriven Ausführungen in Redriving -Ausführungen.

Ein Retrier enthält die folgenden Felder:

> **Note**
>
> Wiederholungen werden als Zustandsübergänge behandelt. Informationen darüber, wie sich Statusübergänge auf die Abrechnung auswirken, finden Sie unter Step Functions — Preisgestaltung.

`ErrorEquals` (Erforderlich)

Ein nicht leeres Array von Zeichenfolgen, die mit Fehlernamen übereinstimmen. Wenn ein Bundesstaat einen Fehler meldet, durchsucht Step Functions die Retrier. Wenn der Fehlername in diesem Array erscheint, implementiert es die in diesem Retrier beschriebene Wiederholungsrichtlinie.

`IntervalSeconds` (Optional)

Eine positive Ganzzahl, die die Anzahl der Sekunden vor dem ersten Wiederholungsversuch darstellt (`1`standardmäßig). `IntervalSeconds`hat einen Höchstwert von 99999999.

`MaxAttempts` (Optional)

Eine positive Ganzzahl, die die maximale Anzahl von erneuten Versuchen angibt (standardmäßig `3`). Wenn der Fehler öfter als angegeben erneut auftritt, erfolgen keine erneuten Versuche und die normale Fehlerbehandlung wird fortgesetzt. Ein Wert von `0` gibt an, dass der Fehler nie erneut versucht wird. `MaxAttempts`hat einen Höchstwert von 99999999.

`BackoffRate` (Optional)

Der Multiplikator, um den sich das mit bezeichnete Wiederholungsintervall nach jedem Wiederholungsversuch erhöht. `IntervalSeconds` Standardmäßig erhöht sich der Wert um. `BackoffRate` `2.0`

Nehmen wir zum Beispiel an, Ihr `IntervalSeconds` Wert `MaxAttempts` ist 3, ist 3 und `BackoffRate` ist 2. Der erste Wiederholungsversuch erfolgt drei Sekunden nach dem Auftreten des Fehlers. Der zweite Versuch erfolgt sechs Sekunden nach dem ersten Wiederholungsversuch. Während der dritte Wiederholungsversuch 12 Sekunden nach dem zweiten Wiederholungsversuch stattfindet.

`MaxDelaySeconds` (Optional)

Eine positive Ganzzahl, die den Höchstwert in Sekunden festlegt, bis zu dem ein Wiederholungsintervall verlängert werden kann. Es ist hilfreich, dieses Feld zusammen mit dem `BackoffRate` Feld zu verwenden. Der Wert, den Sie in diesem Feld angeben, begrenzt die exponentiellen Wartezeiten, die sich aus dem Multiplikator für die Backoff-Rate ergeben, der auf jeden aufeinanderfolgenden Wiederholungsversuch angewendet wird. Sie müssen einen Wert größer als 0 und kleiner als 31622401 für angeben. `MaxDelaySeconds`

Wenn Sie diesen Wert nicht angeben, begrenzt Step Functions die Wartezeiten zwischen Wiederholungsversuchen nicht.

`JitterStrategy` (Optional)

Eine Zeichenfolge, die bestimmt, ob Jitter in die Wartezeiten zwischen aufeinanderfolgenden Wiederholungsversuchen einbezogen werden soll oder nicht. Jitter reduziert gleichzeitige Wiederholungsversuche, indem diese über ein zufälliges Verzögerungsintervall verteilt werden. Diese Zeichenfolge akzeptiert `FULL` oder `NONE` als ihre Werte. Der Standardwert ist `NONE`.

Nehmen wir zum Beispiel an, Sie haben den Wert `MaxAttempts` als 3, `IntervalSeconds` als 2 und `BackoffRate` als 2 festgelegt. Der erste Wiederholungsversuch erfolgt zwei Sekunden nach dem Auftreten des Fehlers. Die zweite Wiederholung erfolgt vier Sekunden nach dem ersten Wiederholungsversuch und die dritte Wiederholung erfolgt acht Sekunden nach dem zweiten Wiederholungsversuch. Wenn Sie dies `JitterStrategy` als festlegen`FULL`, wird das erste Wiederholungsintervall nach dem Zufallsprinzip zwischen 0 und 2 Sekunden, das zweite Wiederholungsintervall zwischen 0 und 4 Sekunden und das dritte Wiederholungsintervall zwischen 0 und 8 Sekunden randomisiert.

Beispiele für Wiederholungsfelder

Dieser Abschnitt enthält die folgenden `Retry` Feldbeispiele.

- Retry with BackoffRate
- Retry with MaxDelaySeconds
- Retry all errors except States.Timeout
- Complex retry scenario

> **Tip**
>
> Ein Beispiel für einen Workflow zur Fehlerbehandlung finden Sie AWS-Konto im Modul Fehlerbehandlung von The AWS Step Functions Workshop.

Beispiel 1 — Versuchen Sie es erneut mit BackoffRate

Das folgende Beispiel für a `Retry` führt zwei Wiederholungsversuche durch, wobei der erste Versuch nach einer Wartezeit von drei Sekunden erfolgt. Je nachdem, was `BackoffRate` Sie angeben, erhöht Step Functions das Intervall zwischen den einzelnen Wiederholungsversuchen, bis die maximale Anzahl von Wiederholungsversuchen erreicht ist. Im folgenden Beispiel beginnt der zweite Wiederholungsversuch, nachdem nach dem ersten Versuch drei Sekunden gewartet wurden.

```
"Retry": [ {
   "ErrorEquals": [ "States.Timeout" ],
   "IntervalSeconds": 3,
   "MaxAttempts": 2,
   "BackoffRate": 1
```

```
} ]
```

Beispiel 2 — Versuchen Sie es erneut mit MaxDelaySeconds

Im folgenden Beispiel werden drei Wiederholungsversuche durchgeführt und die daraus resultierende Wartezeit `BackoffRate` auf 5 Sekunden begrenzt. Der erste Wiederholungsversuch erfolgt nach einer Wartezeit von drei Sekunden. Der zweite und dritte Wiederholungsversuch erfolgen, nachdem fünf Sekunden nach dem vorherigen Wiederholungsversuch gewartet wurden, da die maximale Wartezeit von festgelegt ist. `MaxDelaySeconds`

```
"Retry": [ {
    "ErrorEquals": [ "States.Timeout" ],
    "IntervalSeconds": 3,
    "MaxAttempts": 3,
    "BackoffRate":2,
    "MaxDelaySeconds": 5,
    "JitterStrategy": "FULL"
} ]
```

`MaxDelaySeconds`Andernfalls würde der zweite Wiederholungsversuch sechs Sekunden nach dem ersten Wiederholungsversuch und der dritte Wiederholungsversuch 12 Sekunden nach dem zweiten Versuch erfolgen.

Beispiel 3 — Alle Fehler außer States.Timeout wiederholen

Der reservierte Name `States.ALL`, der im Feld `ErrorEquals` eines Retriers angezeigt wird, ist ein Platzhalter, der mit jedem Fehlernamen übereinstimmt. Er muss allein im Array `ErrorEquals` erscheinen und er muss im letzten Retrier im Array `Retry` erscheinen. Der Name dient `States.TaskFailed` auch als Platzhalter und entspricht allen Fehlern mit Ausnahme von. `States.Timeout`

Das folgende Beispiel für ein `Retry` Feld wiederholt jeden Fehler mit Ausnahme von. `States.Timeout`

```
"Retry": [ {
   "ErrorEquals": [ "States.Timeout" ],
   "MaxAttempts": 0
}, {
   "ErrorEquals": [ "States.ALL" ]
} ]
```

Beispiel 4 — Komplexes Wiederholungsszenario

Die Parameter eines Retriers gelten für alle Besuche des Retriers im Kontext einer einzelnen Zustandsausführung.

Beachten Sie den folgenden `Task`-Zustand.

```
"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"
   } ]
}
```

Diese Aufgabe schlägt viermal hintereinander fehl und gibt die folgenden Fehlernamen aus:`ErrorA`, `ErrorBErrorC`, und. `ErrorB` Infolgedessen geschieht Folgendes:

- Die ersten beiden Fehler stimmen mit dem ersten Abruf überein und führen zu Wartezeiten von ein bis zwei Sekunden.
- Der dritte Fehler entspricht dem zweiten Retrier und führt zu einer Wartezeit von fünf Sekunden.
- Der vierte Fehler entspricht auch dem ersten Retrier. Für diesen bestimmten Fehler wurde jedoch bereits das Maximum von zwei Wiederholungen (`MaxAttempts`) erreicht. Daher schlägt dieser Abruf fehl und die Ausführung leitet den Workflow über das Feld in den Z Status um. `Catch`

Fallback-Staaten

`Task`, `Map` und für jeden `Parallel` Bundesstaat kann ein Feld benannt `Catch` werden. Der Wert dieses Felds muss eine Reihe von Objekten sein, die als Catcher bekannt sind.

Ein Catcher enthält die folgenden Felder.

`ErrorEquals` (Erforderlich)

Ein nicht leeres Array von Zeichenfolgen, das mit Fehlernamen übereinstimmt, genauso angegeben wie bei dem Retrier-Feld desselben Namens.

`Next` (Erforderlich)

Eine Zeichenfolge, die mit genau einem der Zustandsnamen des Zustandsautomaten übereinstimmen muss.

`ResultPath` (Optional)

Ein Pfad, der bestimmt, welche Eingabe der Catcher an den im `Next` Feld angegebenen Status sendet.

Wenn ein Status einen Fehler meldet und entweder kein `Retry` Feld vorhanden ist oder wenn der Fehler durch erneute Versuche nicht behoben werden kann, durchsucht Step Functions die Catcher in der Reihenfolge, die im Array aufgeführt ist. Wenn der Fehlername im Feld `ErrorEquals` eines Catchers erscheint, wechselt der Zustandsautomat in den Zustand, der im Feld `Next` genannt ist.

Der reservierte Name `States.ALL`, der im Feld `ErrorEquals` des Catchers erscheint, ist ein Platzhalter, der mit jedem Fehlernamen übereinstimmt. Er muss allein im Array `ErrorEquals` erscheinen und er muss im letzten Catcher im Array `Catch` erscheinen. Der Name dient `States.TaskFailed` auch als Platzhalter und entspricht allen Fehlern mit Ausnahme von. `States.Timeout`

Das folgende Beispiel zeigt, wie ein `Catch` Feld in den angegebenen Zustand übergeht, `RecoveryState` wenn eine Lambda-Funktion eine unbehandelte Java-Ausnahme ausgibt. Andernfalls wechselt das Feld in den Zustand `EndState`.

```
"Catch": [ {
   "ErrorEquals": [ "java.lang.Exception" ],
   "ResultPath": "$.error-info",
   "Next": "RecoveryState"
}, {
   "ErrorEquals": [ "States.ALL" ],
   "Next": "EndState"
} ]
```

> **Note**
>
> Jeder Catcher kann mehrere zu behandelnde Fehler angeben.

Fehlerausgabe

Wenn Step Functions in den in einem Fangnamen angegebenen Status übergeht, enthält das Objekt normalerweise das Feld`Cause`. Der Wert dieses Felds ist eine für Menschen lesbare Beschreibung des Fehlers. Das Objekt ist als Fehlerausgabe bekannt.

In diesem Beispiel enthält der erste Catcher ein Feld `ResultPath`. Dies funktioniert ähnlich wie ein `ResultPath`-Feld im obersten Level eines Zustands, wodurch sich zwei Möglichkeiten ergeben:

- Es verwendet die Ergebnisse der Ausführung dieses Zustands und überschreibt entweder die gesamte Eingabe des Status oder einen Teil davon.
- Es nimmt die Ergebnisse und fügt sie der Eingabe hinzu. Im Falle eines Fehlers, der von einem Catcher behandelt wird, ist das Ergebnis der Ausführung des Zustands die Fehlerausgabe.

Für den ersten Catcher im Beispiel fügt der Catcher der Eingabe also die Fehlerausgabe als Feld mit dem Namen hinzu, `error-info` falls es nicht bereits ein Feld mit diesem Namen in der Eingabe gibt. Dann sendet der Catcher die gesamte Eingabe an`RecoveryState`. Beim zweiten Catcher überschreibt die Fehlerausgabe die Eingabe und der Catcher sendet nur die Fehlerausgabe an. `EndState`

> **Note**
>
> Wenn Sie das Feld `ResultPath` nicht angeben, wird es standardmäßig mit $ ausgefüllt, was die gesamte Eingabe auswählt und so überschreibt.

Wenn ein Status `Retry` sowohl als auch `Catch` Felder enthält, verwendet Step Functions zuerst alle geeigneten Retrier. Wenn die Wiederholungsrichtlinie den Fehler nicht beheben kann, wendet Step Functions den passenden Catcher-Übergang an.

Verursacht Payloads und Serviceintegrationen

Ein Catcher gibt eine Zeichenketten-Payload als Ausgabe zurück. Wenn Sie mit Serviceintegrationen wie Amazon Athena oder arbeitenAWS CodeBuild, möchten Sie die `Cause` Zeichenfolge möglicherweise in JSON konvertieren. Das folgende Beispiel für einen `Pass` Status mit systemeigenen Funktionen zeigt, wie eine `Cause` Zeichenfolge in JSON konvertiert wird.

```
"Handle escaped JSON with JSONtoString": {
  "Type": "Pass",
  "Parameters": {
    "Cause.$": "States.StringToJson($.Cause)"
  },
  "Next": "Pass State with Pass Processing"
},
```

Beispiele für Zustandsmaschinen mit Retry und Catch

Die in den folgenden Beispielen definierten Zustandsmaschinen setzen die Existenz von zwei Lambda-Funktionen voraus: eine, die immer fehlschlägt, und eine, die lange genug wartet, bis ein in der Zustandsmaschine definierter Timeout eintritt.

Dies ist eine Definition einer Lambda-Funktion von Node.js, die immer fehlschlägt und die Nachricht `error` zurückgibt. In den folgenden State-Machine-Beispielen wird diese Lambda-Funktion benannt`FailFunction`. Informationen zum Erstellen einer Lambda-Funktion finden Sie Schritt 1: Erstellen einer Lambda-Funktion im Abschnitt.

```
exports.handler = (event, context, callback) => {
    callback("error");
};
```

Dies ist eine Definition einer Lambda-Funktion von Node.js, die 10 Sekunden lang ruht. In den folgenden State-Machine-Beispielen wird diese Lambda-Funktion benannt`sleep10`.

> **Note**
>
> Denken Sie beim Erstellen dieser Lambda-Funktion in der Lambda-Konsole daran, den Timeout-Wert im Abschnitt Erweiterte Einstellungen von 3 Sekunden (Standard) auf 11 Sekunden zu ändern.

```
exports.handler = (event, context, callback) => {
    setTimeout(function(){
    }, 11000);
};
```

Behandlung eines Fehlers mithilfe von Retry

Dieser Zustandsautomat verwendet ein `Retry`-Feld, das eine fehlgeschlagene Funktion erneut versucht und den Fehlernamen `HandledError` ausgibt. Diese Funktion wird zweimal wiederholt, wobei zwischen den Wiederholungen ein exponentieller Backoff auftritt.

```
{
   "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
      }
   }
}
```

Diese Variante verwendet den vordefinierten Fehlercode`States.TaskFailed`, der jedem Fehler entspricht, den eine Lambda-Funktion ausgibt.

```
{
   "Comment": "A Hello World example of the Amazon States Language using an AWS Lambda
 function",
   "StartAt": "HelloWorld",
   "States": {
      "HelloWorld": {
         "Type": "Task",
         "Resource": "arn:aws:lambda:us-east-1:123456789012:function:FailFunction",
```

```
         "Retry": [ {
            "ErrorEquals": ["States.TaskFailed"],
            "IntervalSeconds": 1,
            "MaxAttempts": 2,
            "BackoffRate": 2.0
         } ],
         "End": true
      }
   }
}
```

> **Note**
>
> Es hat sich bewährt, dass Aufgaben, die auf eine Lambda-Funktion verweisen, Lambda-Serviceausnahmen behandeln sollten. Weitere Informationen finden Sie unter Lambda-Serviceausnahmen behandeln.

Behandlung eines Fehlers mit Catch

In diesem Beispiel wird ein `Catch`-Feld verwendet. Wenn eine Lambda-Funktion einen Fehler ausgibt, fängt sie den Fehler ab und die Zustandsmaschine wechselt in den `fallback` Status.

```
{
   "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
      }
```

```
   }
}
```

Diese Variante verwendet den vordefinierten Fehlercode`States.TaskFailed`, der jedem Fehler entspricht, den eine Lambda-Funktion ausgibt.

```
{
   "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
      }
   }
}
```

Behandlung eines Timeouts mit Retry

Diese Zustandsmaschine verwendet ein `Retry` Feld, um einen `Task` Status zu wiederholen, bei dem das Timeout überschritten wurde, basierend auf dem in angegebenen Timeout-Wert. `TimeoutSeconds` Step Functions wiederholt den Lambda-Funktionsaufruf in diesem `Task` Zustand zweimal, mit einem exponentiellen Backoff zwischen den Wiederholungen.

```
{
   "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
      }
   }
}
```

Behandlung eines Timeouts mit Catch

In diesem Beispiel wird ein `Catch`-Feld verwendet. Wenn ein Timeout aufgetreten ist, wechsel der Zustandsautomat in den Zustand `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**
>
> Sie können die Statuseingabe und den Fehler beibehalten, indem Sie `ResultPath` verwenden. Siehe Verwenden Sie ResultPath , um sowohl Fehler als auch Eingabe in einen einzuschließen `Catch`.

AufrufenAWS Step Functionsvon anderen Diensten

Sie können mehrere andere Dienste konfigurieren, um State-Maschinen aufzurufen. Basierend auf der StaatsmaschineWorkflow-Art, Sie können State-Maschinen asynchron oder synchron aufrufen. Um State-Maschinen synchron aufzurufen, verwenden Sie den`StartSyncExecution`API-Aufruf oder Amazon API Gateway-Integration mit Express Workflows. Beim asynchronen Aufruf unterbricht Step Functions die Workflow-Ausführung, bis ein Task-Token zurückgegeben wird. Das Warten auf ein Task-Token macht den Workflow jedoch synchron.

Zu den Diensten, die Sie für den Aufruf von Step Functions konfigurieren können, gehören:

- AWS Lambda, unter Verwendung der`StartExecution`anrufen.
- Amazon API Gateway
- Amazon EventBridge
- AWS CodePipeline
- AWS IoTRegel-Engine
- AWS Step Functions

Die Aufrufe von Step Functions werden gesteuert durch`StartExecution`Quote. Weitere Informationen finden Sie unter:

- Kontingente

Lesen Sie Consistency in Step Functions

Zustandsautomatenaktualisierung in AWS Step Functions sind eventually consistent. Bei allen `StartExecution` Aufrufen innerhalb weniger Sekunden werden die aktualisierte Definition und `roleArn` (der Amazon-Ressourcenname für die IAM-Rolle) verwendet. Ausführungen,

die unmittelbar nach dem Aufruf von `UpdateStateMachine` gestartet werden, verwenden möglicherweise noch die vorherige Definition des Zustandsautomaten und den vorherigen `roleArn`.

Weitere Informationen finden Sie hier:

- `UpdateStateMachine` in der AWS Step Functions-API-Referenz
- Aktualisieren Sie einen Workflow inErste Schritte mit AWS Step Functions.

Funktionen zum Taggen in Step

AWS Step Functions unterstützt das Markieren von Zustandsautomaten (sowohl Standard als auch Express) und von Aktivitäten. Dies kann Ihnen helfen, die mit Ihren Ressourcen verbundenen Kosten zu verfolgen und zu verwalten und für mehr Sicherheit in Ihren AWS Identity and Access Management (IAM-) Richtlinien zu sorgen. Durch das Taggen von Step Functions-Ressourcen können sie von AWS Resource Groups verwaltet werden. Weitere Informationen zu Ressourcengruppen finden Sie im AWS Resource GroupsBenutzerhandbuch.

Bei der tagbasierten Autorisierung erben die Ausführungsressourcen der State Machine, wie im folgenden Beispiel gezeigt, die Tags, die einer State-Maschine zugeordnet sind.

```
arn:<partition>:states:<Region>:<account-id>:execution:<StateMachineName>:<ExecutionId>
```

Wenn Sie DescribeExecutionoder andere APIs aufrufen, in denen Sie den Ausführungsressource ARN angeben, verwendet Step Functions Tags, die der State Machine zugeordnet sind, um die Anfrage anzunehmen oder abzulehnen und gleichzeitig die tagbasierte Autorisierung durchzuführen. Auf diese Weise können Sie den Zugriff auf State-Machine-Ausführungen auf State-Machine-Ebene zulassen oder verweigern.

Weitere Informationen zu Einschränkungen in Bezug auf das Markieren von Ressourcen finden Sie unter Einschränkungen im Zusammenhang mit dem Tagging.

Themen

- Markieren für die Kostenzuordnung
- Markieren für-Sicherheit
- Tags in der Step Functions Console anzeigen und verwalten
- Tags mit API-Aktionen von Step Functions verwalten

Markieren für die Kostenzuordnung

Um Ihre Step Functions-Ressourcen für die Kostenzuweisung zu organisieren und zu identifizieren, können Sie Metadaten-Tags hinzufügen, die den Zweck einer State-Maschine oder Aktivität angeben. Dies ist vor allem nützlich, wenn Sie viele Ressourcen haben. Organisieren Sie Ihre AWS-Rechnung mit Kostenzuordnungs-Tags, damit Sie Ihre eigene Kostenstruktur wiedergeben können. Melden Sie sich an, um Ihre AWS-Kontorechnung mit Tag-Schlüsseln und Werten zu erhalten. Weitere Informationen finden Sie unter Einrichten Ihres monatlichen Kostenzuordnungsberichts im AWS BillingBenutzerhandbuch.

Sie könnten beispielsweise wie folgt Tags hinzufügen, die die Kostenstelle und den Zweck Ihrer Step Functions-Ressourcen darstellen.

Ressource	Schlüssel	Value (Wert)
`StateMachine1`	`Cost Center`	`34567`
	`Application`	`Image processing`
`StateMachine2`	`Cost Center`	`34567`
	`Application`	`Rekognition processing`
`Activity1`	`Cost Center`	`12345`
	`Application`	`Legacy database`

Dieses Markierungsschema ermöglicht es Ihnen, zwei Zustandsautomaten zu gruppieren, die verwandte Aufgaben auf derselben Kostenstelle ausführen, während Sie eine unabhängige Aktivität mit einem anderen Tag für die Kostenzuordnung markieren.

Markieren für-Sicherheit

IAM unterstützt die Steuerung des Zugriffs auf Ressourcen anhand von Tags. Um den Zugriff anhand von Tags zu steuern, geben Sie Informationen zu Ihren Ressourcen-Tags im Bedingungselement einer IAM-Richtlinie an.

Sie könnten beispielsweise den Zugriff auf alle Step Functions-Ressourcen einschränken, die ein Tag mit dem Schlüssel `environment` und dem Wert enthalten`production`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "states:TagResource",
                "states:DeleteActivity",
                "states:DeleteStateMachine",
                "states:StopExecution"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {"aws:ResourceTag/environment": "production"}
            }
        }
    ]
}
```

Weitere Informationen finden Sie unter Zugriffssteuerung mit Tags im IAM-Benutzerhandbuch.

Tags in der Step Functions Console anzeigen und verwalten

Mit Step Functions können Sie die Tags für Ihre State-Maschinen in der Step Functions-Konsole anzeigen und verwalten. Wählen Sie auf der Seite Details eines Zustandsautomaten die Option Tags aus. Hier werden Ihnen bestehende Tags angezeigt, die Ihrem Zustandsautomaten zugeordnet sind.

> **Note**
>
> Informationen zum Verwalten von Tags für Aktivitäten finden Sie unter Tags mit API-Aktionen von Step Functions verwalten.

Um Tags hinzuzufügen oder zu löschen, die Ihrem Zustandsautomaten zugeordnet sind, wählen Sie die Schaltfläche Manage Tags (Tags verwalten).

1. Navigieren Sie zur Detailseite eines Zustandsautomaten.
2. Wählen Sie die Tags neben Executions (Ausführungen) und Definition.

3. Wählen Sie Manage tags (Tags (Markierungen) verwalten) aus.
   - Bearbeiten Sie zum Ändern bestehender Tags die Werte Key (Schlüssel) und Value (Wert).
   - Um vorhandene Tags zu entfernen, wählen Sie Remove Tag (Tag entfernen) aus.
   - Um ein neues Tag hinzuzufügen, wählen Sie Add tag (Tag hinzufügen) aus und geben Sie einen Key (Schlüssel) und Value (Wert) ein.
4. Wählen Sie Speichern aus.

Tags mit API-Aktionen von Step Functions verwalten

Verwenden Sie die folgenden API-Aktionen, um Tags mithilfe der Step Functions API zu verwalten:

- `ListTagsForResource`
- `TagResource`
- `UntagResource`

AWS Step Functions Workflow-Studio

Workflow Studio for AWS Step Functions ist ein visueller Low-Code-Workflow-Designer, mit dem Sie serverlose Workflows erstellen können, indem Sie Dienste orchestrieren AWS . Mithilfe dieser drag-and-drop Funktion oder des integrierten Code-Editors können Sie Workflows erstellen und bearbeiten, steuern, wie Eingabe und Ausgabe für jeden Status gefiltert oder transformiert werden, und die Fehlerbehandlung konfigurieren. Wenn Sie Status per Drag-and-Drop verschieben, um Ihren Workflow zu erstellen, validiert Workflow Studio Ihre Arbeit und generiert automatisch Code. Sie können den generierten Code überprüfen oder die State-Machine-Definition im Code-Editor aktualisieren. Wenn Sie fertig sind, können Sie Ihren Workflow speichern, ausführen und dann die Ergebnisse in der Step Functions Functions-Konsole überprüfen. Sie können Workflows visuell hinzufügen und ändern, um die verschiedenen Dienste in Ihrer Anwendung zu orchestrieren.

Um Step Functions Workflow Studio verwenden zu können AWS-Konto, benötigen Sie ein und Anmeldeinformationen, die die richtigen Berechtigungen für alle Ressourcen bereitstellen, die Sie verwenden möchten. Weitere Informationen finden Sie unter Voraussetzungen für den Einstieg in AWS Step Functions.

> **Note**
>
> Workflow Studio unterstützt Internet Explorer 11 nicht. Wenn Sie Internet Explorer 11 verwenden und Probleme mit Workflow Studio haben, versuchen Sie es mit einem anderen Browser.

Sie können über die Step Functions-Konsole auf Workflow Studio zugreifen, wenn Sie einen Workflow in Step Functions erstellen oder bearbeiten. Sie können auch von dort aus auf Workflow Studio zugreifenAWS Application Composer. Workflow Studio in Application Composer bietet eine visuelle IaC-Umgebung, die es Ihnen leicht macht, Workflows in Ihre serverlosen Anwendungen zu integrieren, die mit IaC-Tools wie Vorlagen erstellt wurden. AWS CloudFormation Mit Workflow Studio in Application Composer können Sie Workflows mithilfe von Vorlagen erstellen. AWS CloudFormation Darin Application Composer können Sie einen neuen Workflow hinzufügen, einen vorhandenen Workflow ändern und einzelne Workflow-Schritte mit anderen Anwendungsressourcen verbinden. Application Composererstellt und aktualisiert automatisch die benötigten CloudFormation Ressourcen und Konfigurationen. Auf diese Weise können Sie alle in Ihren Workflows verwendeten Ressourcen an einem Ort erstellen und verwalten. Dies hilft Ihnen auch dabei, Ihren Weg vom Workflow-Prototyping zur Produktionsbereitstellung zu beschleunigen.

Wenn Sie Workflow Studio in verwendenApplication Composer, können Sie sich auch direkt mit Ihrem lokalen Projekt verbinden. Dies hilft Ihnen, in Ihrer integrierten Entwicklungsumgebung (IDE) neben der visuellen Leinwand zu arbeiten. Weitere Informationen finden Sie unter Verwenden von Workflow Studio in Application Composer.

Themen

Übersicht über die Oberfläche

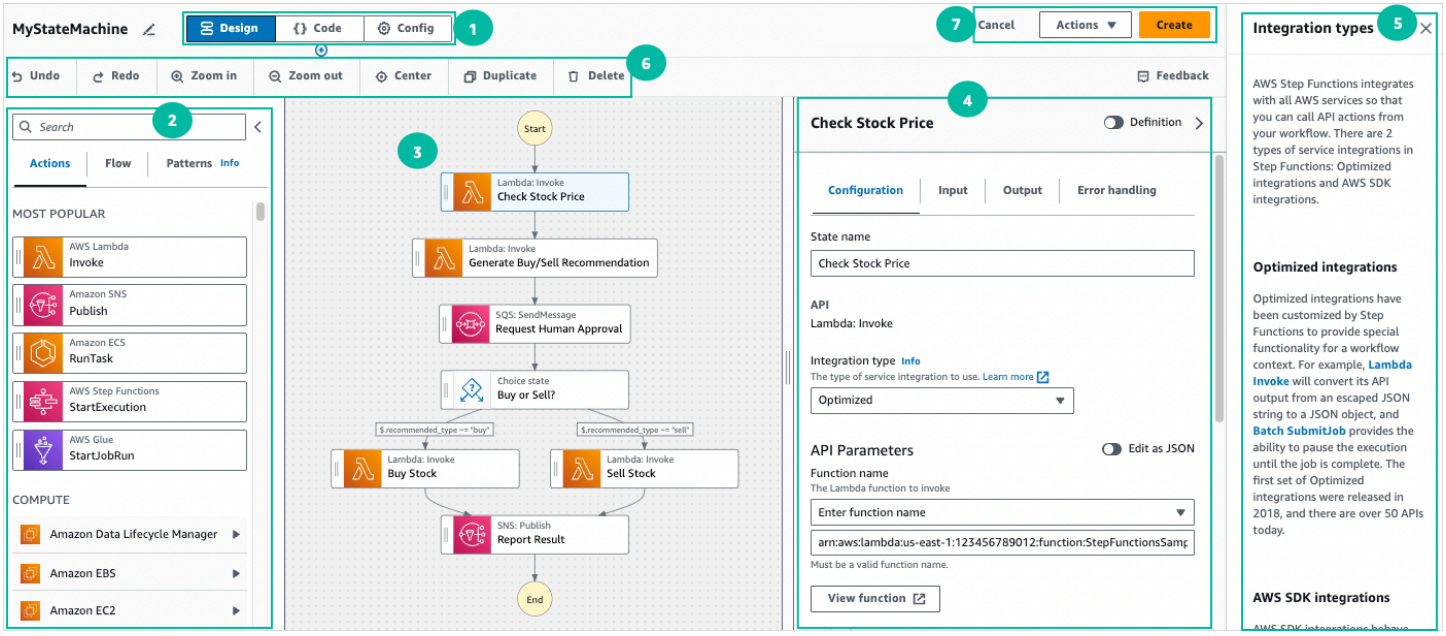
Workflow Studio for AWS Step Functions ist ein visueller Low-Code-Workflow-Designer, mit dem Sie serverlose Workflows durch Orchestrierung erstellen können. AWS-Services Mit seiner Drag-and-Drop-Funktion erleichtert Ihnen Workflow Studio das Erstellen, Bearbeiten und Visualisieren Ihrer Workflow-Prototypen. Workflow Studio bietet auch einen integrierten Code-Editor zum Schreiben und Bearbeiten Ihrer Workflow-Definitionen mithilfe von Amazon States Language (ASL) in der Step Functions Functions-Konsole.

Um Sie bei der Erstellung und Visualisierung Ihrer Workflows, der Bearbeitung ihrer Definitionen und der Verwaltung ihrer Konfiguration zu unterstützen, bietet Workflow Studio drei Modi: Design, Code und Config. In den folgenden Abschnitten werden diese Modi detailliert beschrieben.

In diesem Thema

Entwurfsmodus

Der Designmodus von Workflow Studio bietet eine grafische Oberfläche zur Visualisierung Ihrer Workflows, während Sie deren Prototypen erstellen. Die folgende Abbildung zeigt die verschiedenen Komponenten, die im Designmodus verfügbar sind.



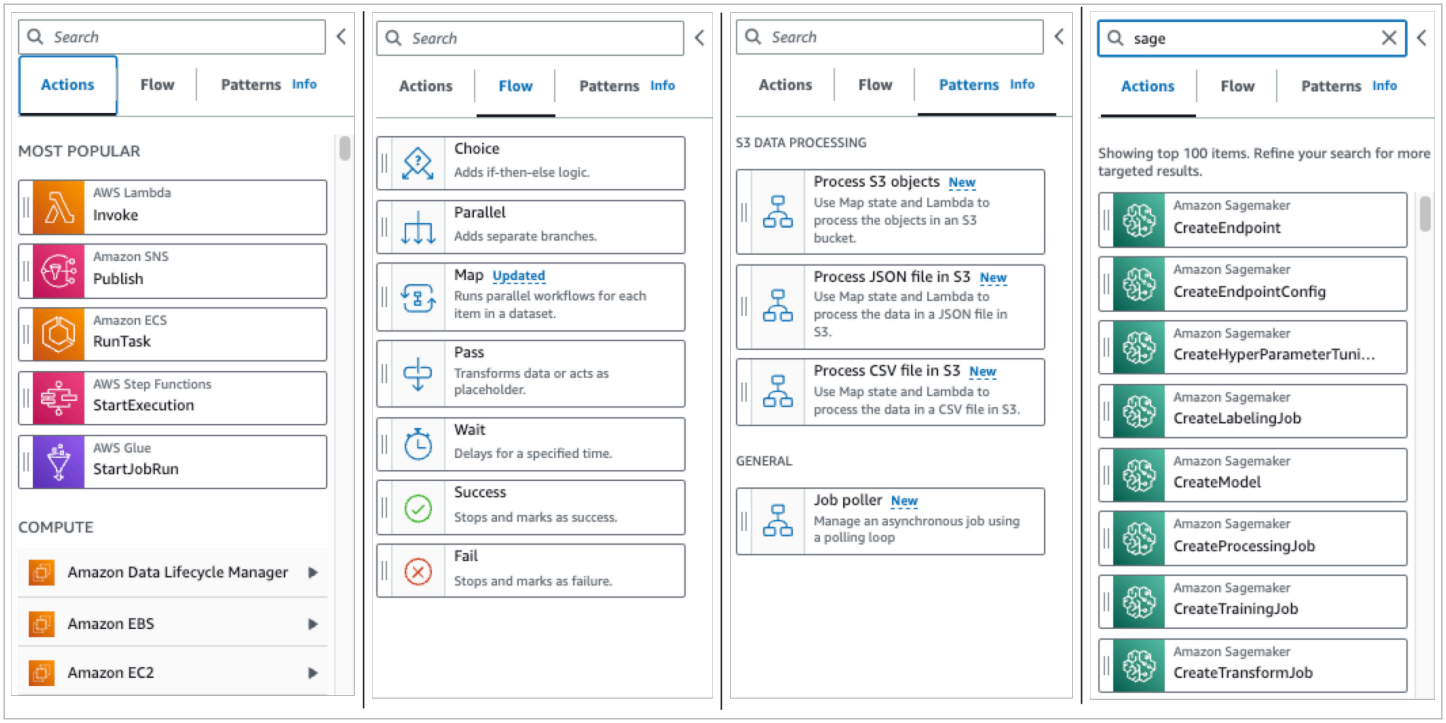
1. Modus-Tasten — Wechseln Sie mit den Modustasten zwischen den Design -, Code - und Config-Modi von Workflow Studio. Sie können den Modus nicht wechseln, wenn das JSON in der ASL-Definition Ihres Workflows ungültig ist.
2. Der Bundesstaaten-Browser enthält die folgenden drei Registerkarten:
   - Die Registerkarte Aktionen enthält eine Liste von AWS APIs, die Sie per Drag-and-Drop in Ihr Workflow-Diagramm auf der Arbeitsfläche ziehen können. Jede Aktion steht für einen Status der Aufgabe Status.
   - Die Registerkarte Flow enthält eine Liste von Flow-Status, die Sie per Drag-and-Drop in Ihr Workflow-Diagramm auf der Arbeitsfläche ziehen können.
   - Die Registerkarte Muster enthält mehrere ready-to-use wiederverwendbare Bausteine, die Sie für eine Vielzahl von Anwendungsfällen verwenden können. Sie können diese Muster beispielsweise verwenden, um Daten in einem Amazon S3 S3-Bucket iterativ zu verarbeiten.
3. LeinwandHier ziehen Sie Status per Drag-and-Drop in Ihr Workflow-Diagramm, ändern die Reihenfolge der Status und wählen Status aus, die konfiguriert oder angezeigt werden sollen.
4. In Inspector diesem Bereich können Sie die Eigenschaften jedes Status, den Sie auf der Leinwand ausgewählt haben, anzeigen und bearbeiten. Aktivieren Sie den Schalter Definition,

um den Sprachcode für die Amazon-Staaten für Ihren Workflow anzuzeigen, wobei der aktuell ausgewählte Bundesstaat hervorgehoben ist.

5. Mit Informationslinks wird ein Fenster mit Kontextinformationen geöffnet, falls Sie Hilfe benötigen. Diese Bereiche enthalten auch Links zu verwandten Themen in der Step Functions Functions-Dokumentation.
6. Entwurfssymbolleiste — Enthält eine Reihe von Schaltflächen für allgemeine Aktionen wie Rückgängigmachen, Löschen und Vergrößern.
7. Hilfsschaltflächen — Eine Reihe von Schaltflächen, mit denen Sie Aufgaben ausführen können, z. B. das Speichern Ihrer Workflows oder das Exportieren ihrer ASL-Definitionen in eine JSON- oder YAML-Datei.

Bundesstaaten-Browser

Im Statusbrowser wählen Sie Bundesstaaten aus, die Sie per Drag-and-Drop in Ihr Workflow-Diagramm ziehen möchten. Die Registerkarte „Aktionen“ enthält eine Liste von AWS APIs, und die Registerkarte „Flow“ enthält eine Liste der Flow-Status. Auf der Registerkarte „Muster“ finden Sie mehrere ready-to-use wiederverwendbare Bausteine, die Sie für eine Vielzahl von Anwendungsfällen verwenden können. Mit dem Suchfeld oben können Sie alle Bundesstaaten im States-Browser durchsuchen.



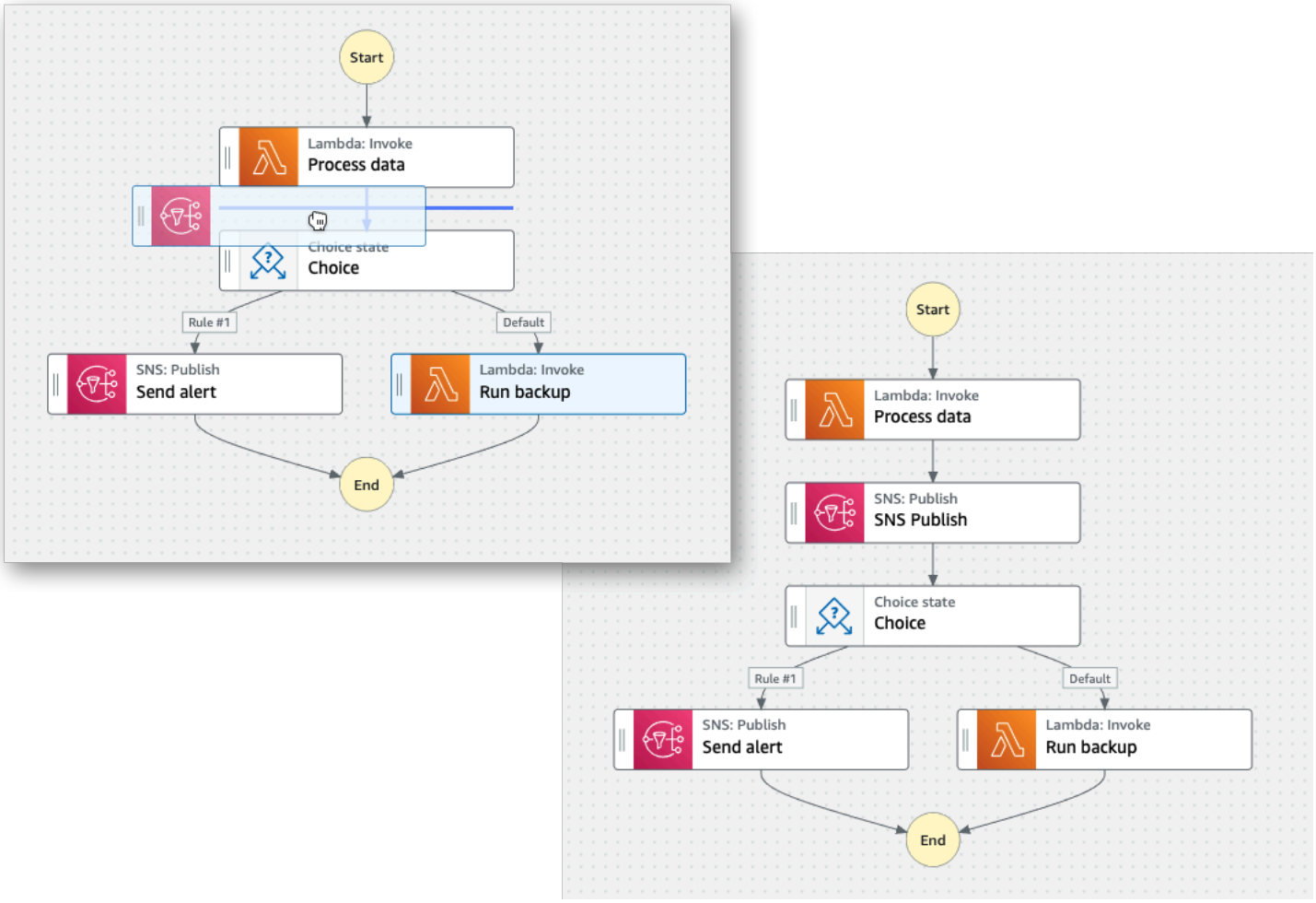
Es gibt sieben Flow-Status, mit denen Sie Ihren Workflow steuern und steuern können. Alle verwenden Eingaben aus dem vorherigen Status, und viele ermöglichen es Ihnen, die Eingabe aus dem vorherigen Status und die Ausgabe in den folgenden Status zu filtern. Die Flusszustände sind:

- Choice: Fügen Sie Ihrem Workflow eine Auswahl zwischen verschiedenen Ausführungszweigen hinzu. Auf der Registerkarte Konfiguration des Inspector können Sie Regeln konfigurieren, um zu bestimmen, in welchen Status der Workflow übergeht.
- Parallel: Fügen Sie Ihrem Workflow parallel Ausführungszweige hinzu.
- Zuordnung: Schritte für jedes Element eines Eingabe-Arrays dynamisch iterieren. Im Gegensatz zu einem `Parallel` `Map` Flow-Status führt ein Status dieselben Schritte für mehrere Einträge eines Arrays in der Statuseingabe aus.
- Pass: Ermöglicht es Ihnen, seine Eingabe an seine Ausgabe zu übergeben. (Optional) Sie können der Ausgabe feste Daten hinzufügen.
- Wait: Lassen Sie Ihren Workflow für eine bestimmte Zeit oder bis zu einer bestimmten Uhrzeit oder einem bestimmten Datum pausieren.
- Succeed: Beendet Ihren Workflow erfolgreich.
- Fehler: Beendet Ihren Workflow mit einem Fehler.

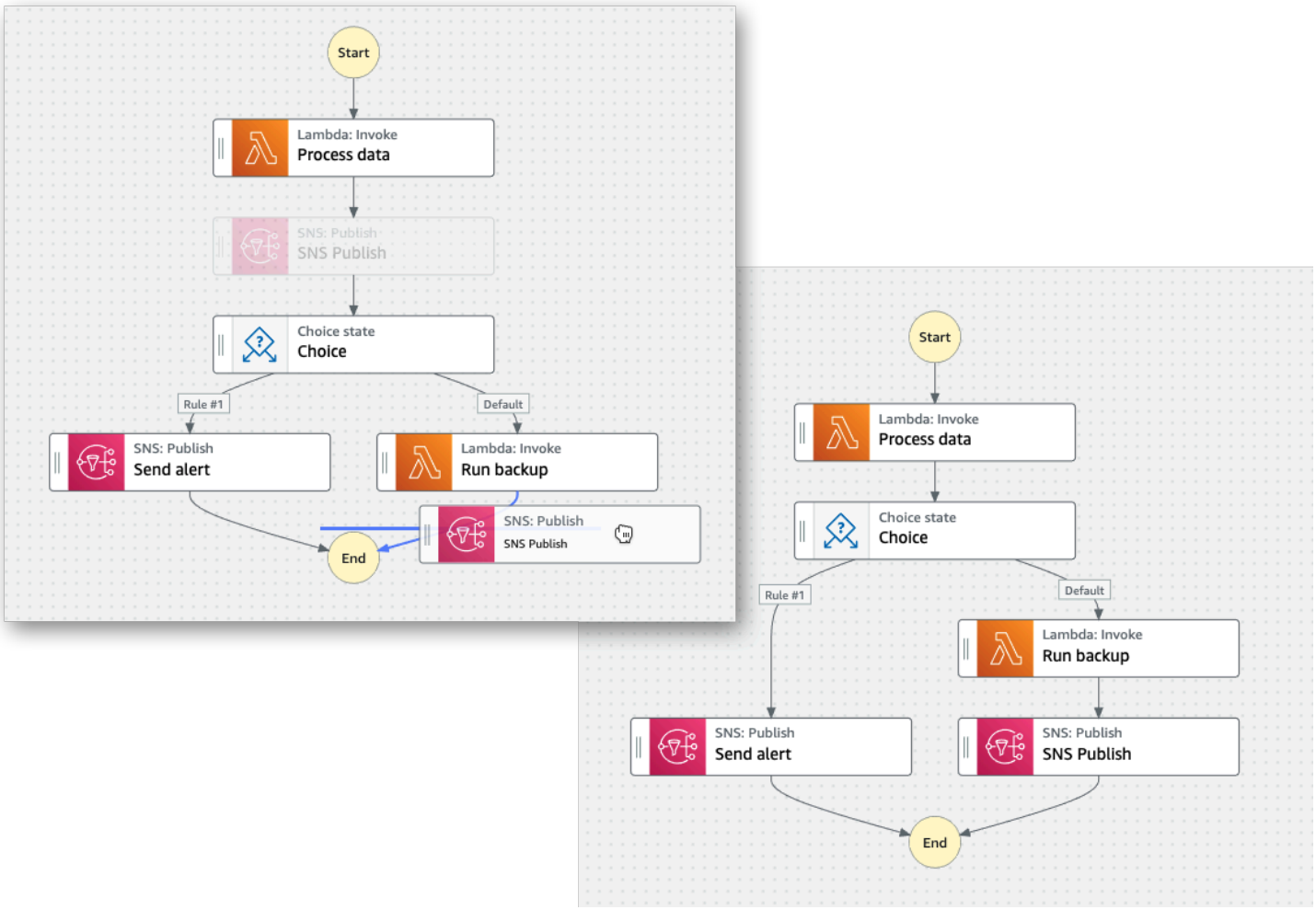
Leinwand

Nachdem Sie einen Status ausgewählt haben, den Sie Ihrem Workflow hinzufügen möchten, ziehen Sie ihn auf die Arbeitsfläche und legen Sie ihn in Ihrem Workflow-Diagramm ab. Sie können Status auch per Drag-and-Drop an verschiedene Stellen in Ihrem Workflow verschieben. Wenn Ihr Arbeitsablauf komplex ist, können Sie ihn möglicherweise nicht vollständig im Canvas-Bereich anzeigen. Verwenden Sie die Steuerelemente oben auf der Arbeitsfläche, um die Ansicht zu vergrößern oder zu verkleinern. Um verschiedene Teile eines Workflow-Diagramms anzuzeigen, können Sie das Workflow-Diagramm auf die Arbeitsfläche ziehen.

Ziehen Sie einen Workflow-Status von der Registerkarte Aktionen oder Flow und legen Sie ihn in Ihren Workflow ab. Eine Linie zeigt, wo sie in Ihrem Workflow platziert wird. Der neue Workflow-Status wurde zu Ihrem Workflow hinzugefügt, und sein Code wird automatisch generiert.

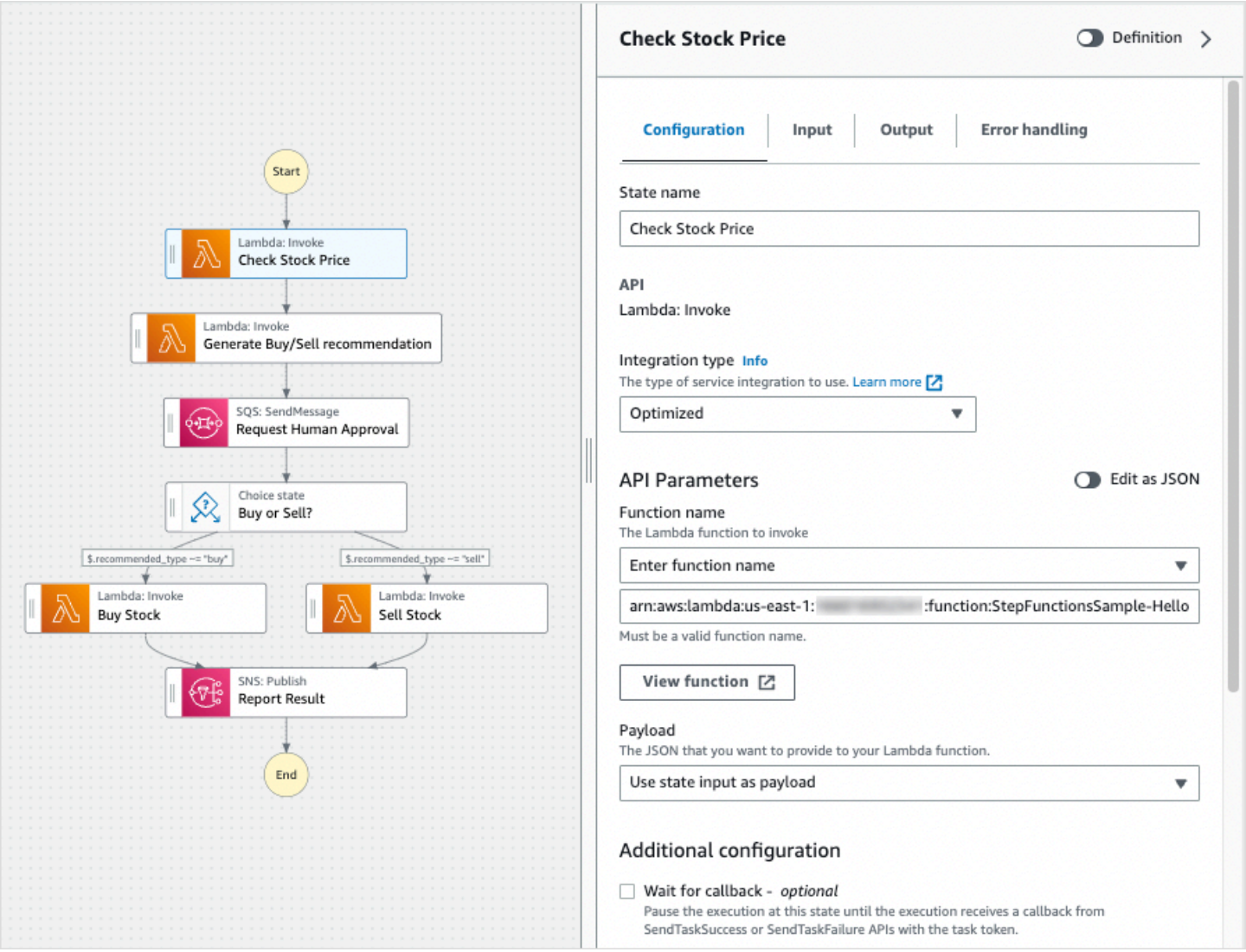


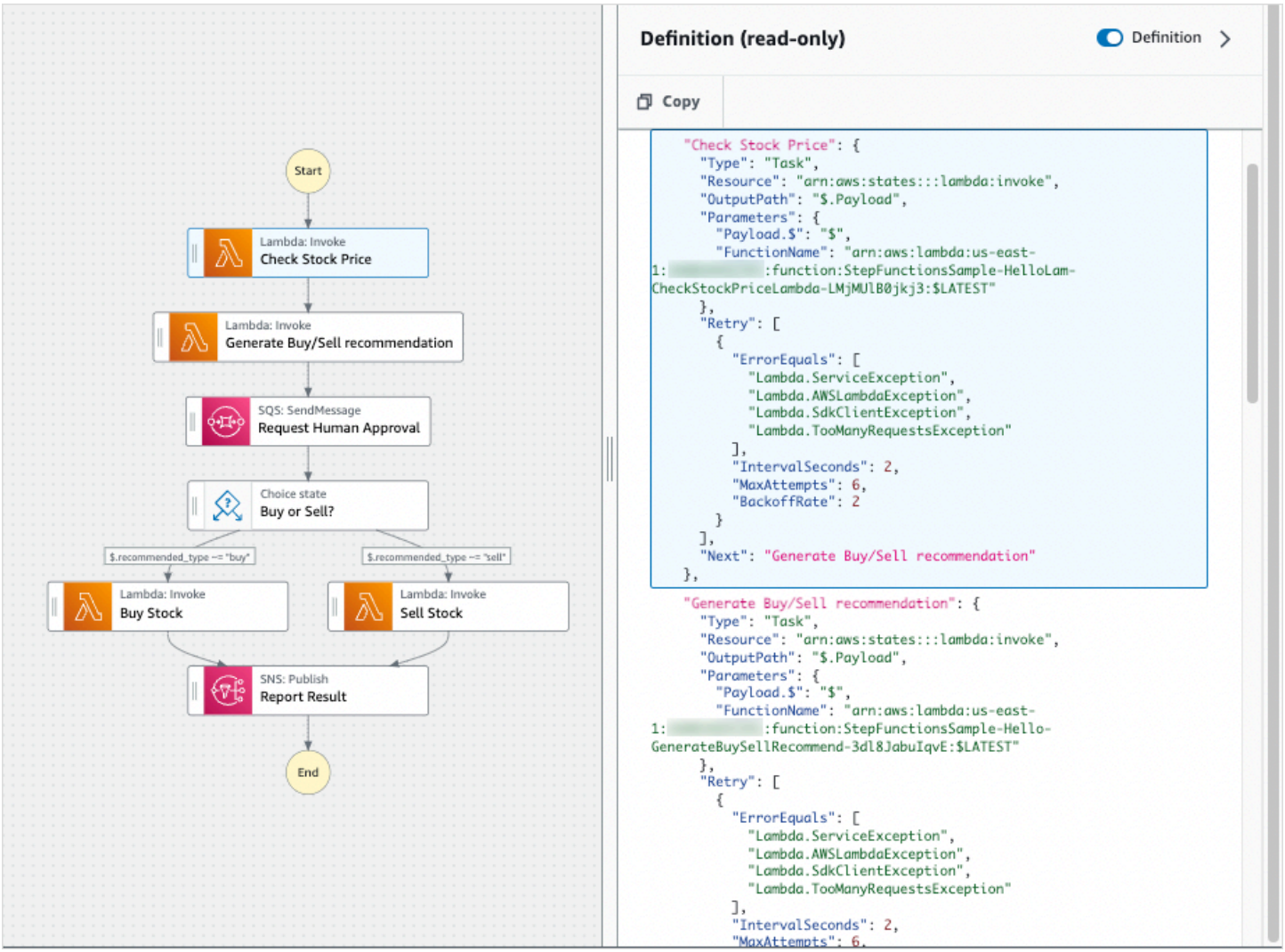
Um die Reihenfolge eines Status zu ändern, können Sie ihn an eine andere Stelle in Ihrem Workflow ziehen.



Inspector

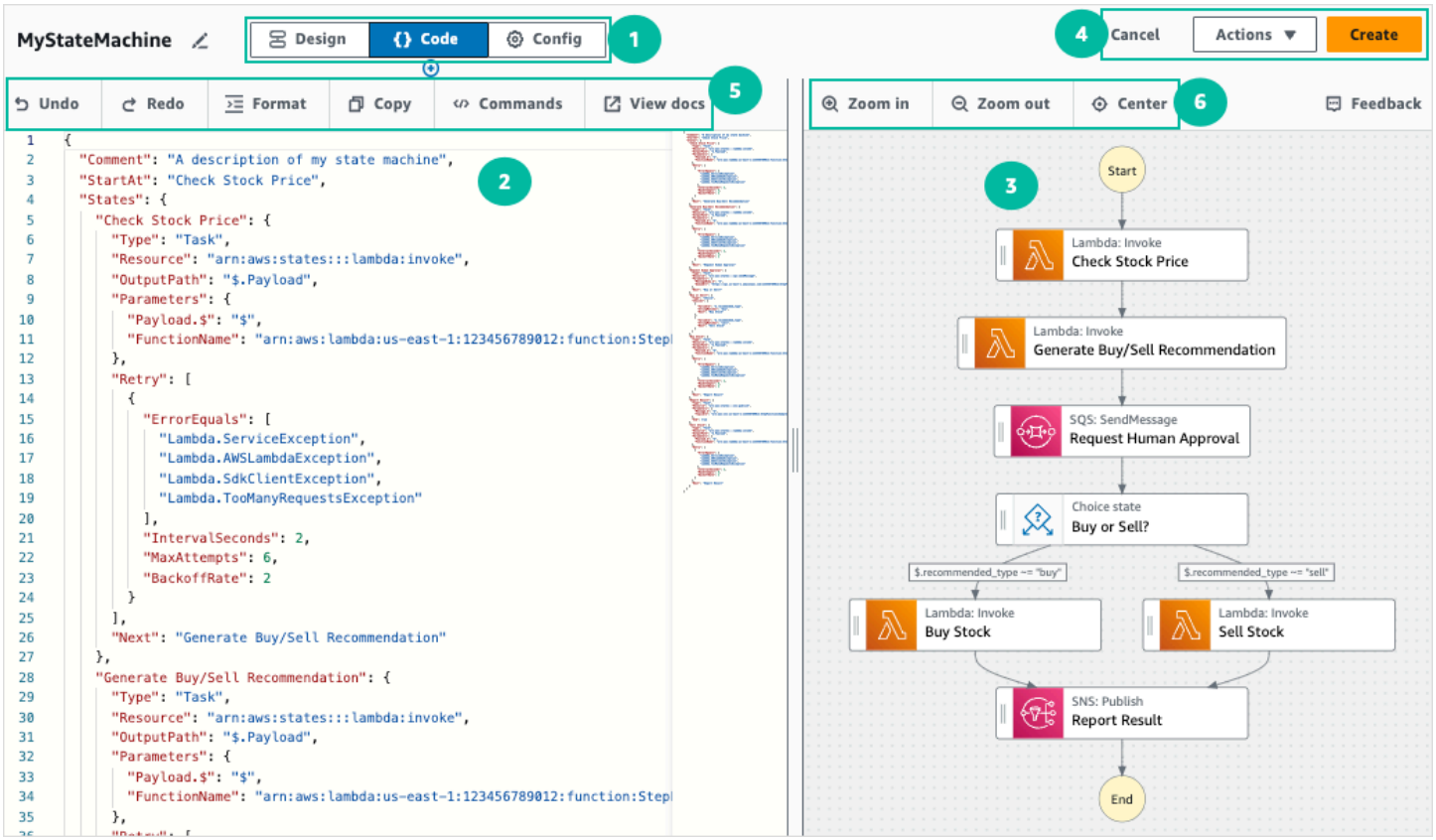
Sie können jeden Status konfigurieren, den Sie Ihrem Workflow hinzufügen. Wählen Sie den Status aus, den Sie konfigurieren möchten, und Sie werden seine Konfigurationsoptionen im Inspektorfenster sehen. Um die automatisch generierte ASL-Definition für Ihren Workflow-Code zu sehen, aktivieren Sie den Schalter Definition. Die ASL-Definition, die dem ausgewählten Status zugeordnet ist, wird hervorgehoben angezeigt.


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.



Codemodus

Der Codemodus von Workflow Studio bietet einen integrierten Code-Editor zum Anzeigen, Schreiben und Bearbeiten der Amazon States Language (ASL-) Definition Ihrer Workflows in der Step Functions Functions-Konsole. Die folgende Abbildung zeigt die verschiedenen Komponenten, die im Codemodus verfügbar sind.



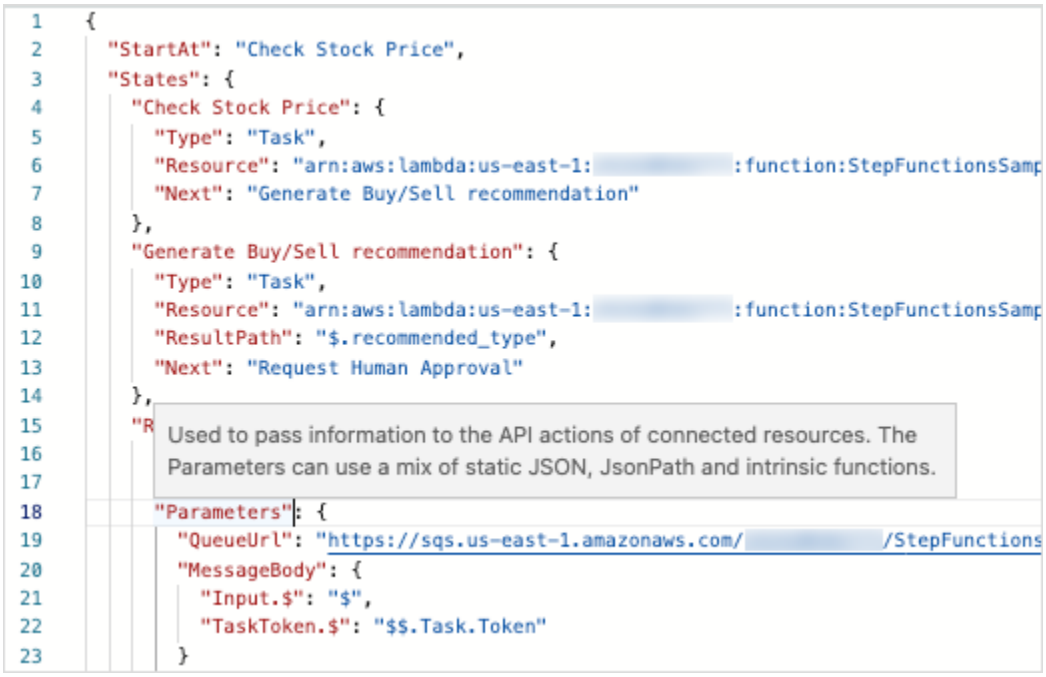
1. Modus-Tasten — Wechseln Sie mit den Modustasten zwischen den Design -, Code - und Config-Modi von Workflow Studio. Sie können den Modus nicht wechseln, wenn das JSON in der ASL-Definition Ihres Workflows ungültig ist.
2. Code-EditorHier schreiben und bearbeiten Sie die ASL-Definition Ihrer Workflows im Workflow Studio. Der Code-Editor bietet auch Funktionen wie Syntaxhervorhebung und automatische Vervollständigung.
3. Bereich zur Grafikvisualisierung— Zeigt eine grafische Echtzeitvisualisierung Ihres Workflows.
4. Hilfsschaltflächen — Eine Reihe von Schaltflächen zum Ausführen von Aufgaben, z. B. zum Speichern Ihrer Workflows oder zum Exportieren ihrer ASL-Definitionen in eine JSON- oder YAML-Datei.
5. Codesymbolleiste — Enthält eine Reihe von Schaltflächen, mit denen allgemeine Aktionen ausgeführt werden können, z. B. das Rückgängigmachen einer Aktion oder das Formatieren des Codes.
6. Grafiksymbolleiste — Enthält eine Reihe von Schaltflächen für allgemeine Aktionen, wie z. B. das Vergrößern und Verkleinern des Workflow-Diagramms.

Code-Editor

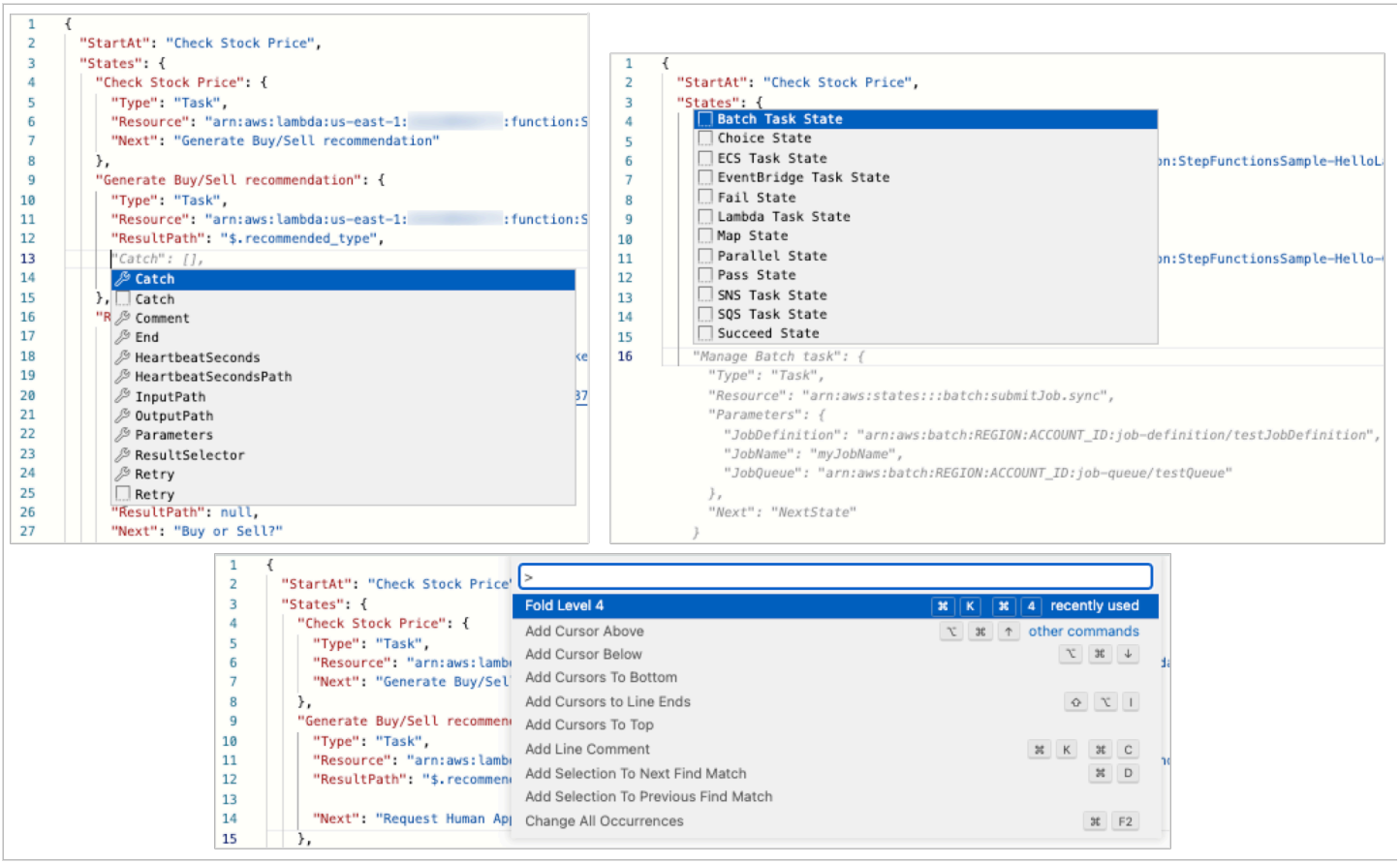
Der Code-Editor bietet eine IDE-ähnliche Oberfläche zum Schreiben und Bearbeiten Ihrer Workflow-Definitionen mithilfe von JSON innerhalb von Workflow Studio. Der Code-Editor enthält mehrere Funktionen, z. B. Syntaxhervorhebung, Vorschläge zur automatischen Vervollständigung, Überprüfung der ASL-Definition und kontextsensitive Hilfeanzeige. Wenn Sie Ihre Workflow-Definition aktualisieren, wird ein Echtzeitdiagramm Ihres Workflows Bereich zur Grafikvisualisierung gerendert. Sie können das aktualisierte Workflow-Diagramm auch in der Entwurfsmodus sehen.

Wenn Sie im Bereich Entwurfsmodus oder im Bereich der Grafikvisualisierung einen Status auswählen, wird die ASL-Definition dieses Status im Code-Editor hervorgehoben angezeigt. Die ASL-Definition Ihres Workflows wird automatisch aktualisiert, wenn Sie einen Status im Entwurfsmodus oder im Bereich der Grafikvisualisierung neu anordnen, löschen oder hinzufügen.

Platzieren Sie den Mauszeiger über einem beliebigen Feld in der Workflow-Definition, um die zugehörige kontextsensitive Hilfe als Tooltip anzuzeigen.



In den Vorschlägen zur automatischen Vervollständigung werden Codefragmente für die Felder oder Status angezeigt, die Sie in Ihre Workflows aufnehmen können. Um eine Liste der Felder anzuzeigen, die Sie in einen bestimmten Status aufnehmen können, drücken Sie auf. **`Ctrl+Space`** Um einen Codeausschnitt für einen neuen Status in Ihrem Workflow zu generieren, drücken Sie **`Ctrl+Space`** hinter der Definition des aktuellen Status. Sie können auch drücken**`F1`**, um eine Liste der verfügbaren Befehle anzuzeigen.



Bereich zur Grafikvisualisierung

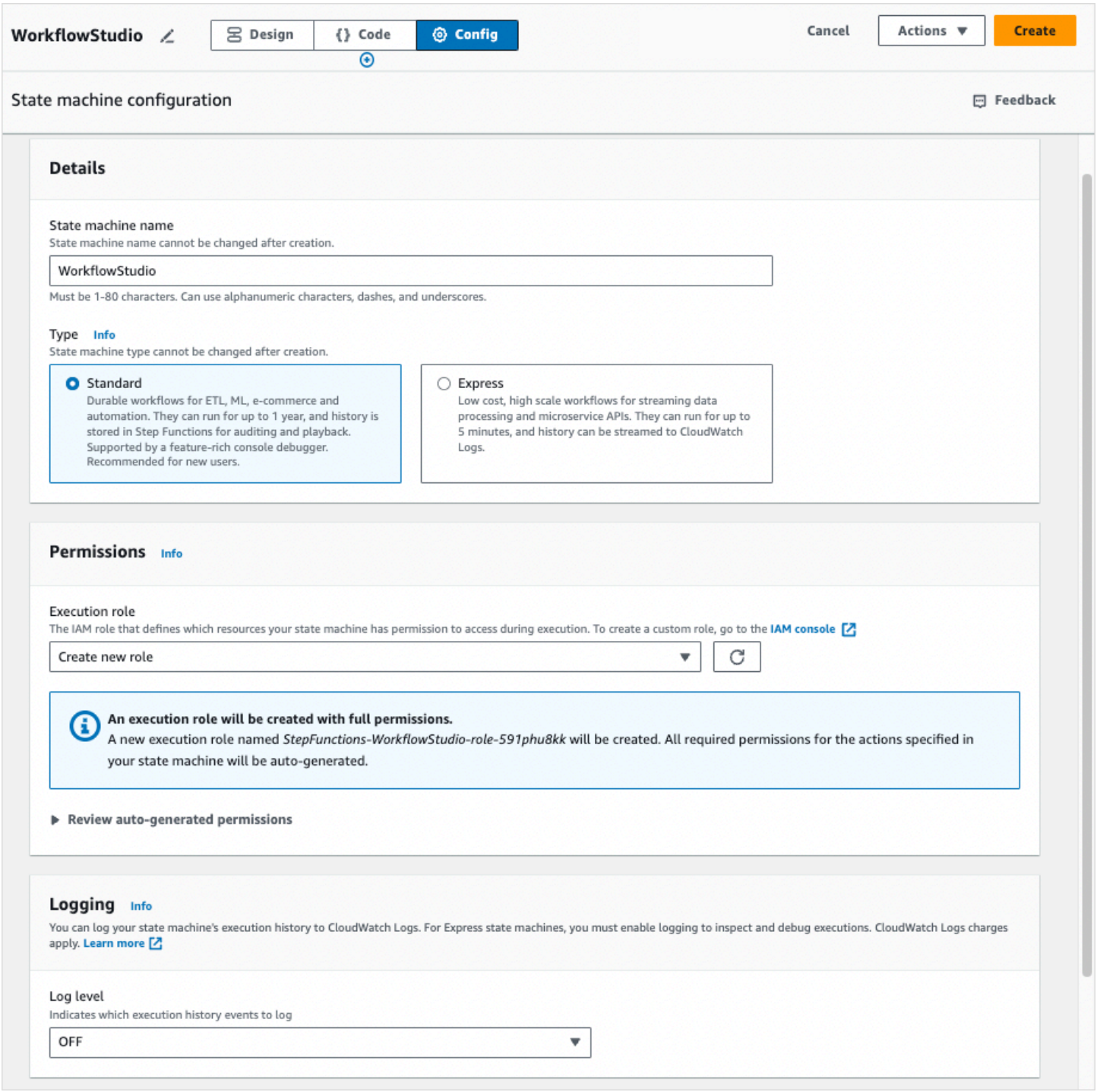
Mithilfe von grafischen Visualisierungen können Sie sehen, wie Ihr Arbeitsablauf in grafischer Form aussieht. Wenn Sie Ihre Workflow-Definitionen in Workflow Studio schreiben, wird im Bereich der Grafikvisualisierung ein Echtzeitdiagramm Ihres Workflows dargestellt. Code-Editor Wenn Sie einen Status im Bereich der Grafikvisualisierung neu anordnen, löschen oder duplizieren, wird die Workflow-Definition im Code-Editor automatisch aktualisiert. Ebenso wird die Visualisierung automatisch aktualisiert, wenn Sie Ihre Workflow-Definitionen aktualisieren, neu anordnen, löschen oder einen Status im Code-Editor hinzufügen.

Wenn der JSON-Code in der ASL-Definition Ihres Workflows ungültig ist, unterbricht der Bereich zur Grafikvisualisierung das Rendern und zeigt am unteren Rand des Bereichs eine Statusmeldung an.

Konfigurationsmodus

Im Konfigurationsmodus von Workflow Studio können Sie die Konfiguration Ihrer Zustandsmaschinen verwalten. In diesem Modus können Sie Details wie den Namen und Typ der Zustandsmaschine, die IAM-Berechtigungen und die Protokollierungskonfiguration für die Zustandsmaschine angeben.

Andere zusätzliche Konfigurationen, die Sie in diesem Modus angeben können, umfassen das Aktivieren der AWS X-Ray Ablaufverfolgung und das Veröffentlichen einer Version, wenn Sie den Zustandsmaschine erstellen. Nachdem Sie den Zustandsmaschine erstellt haben, können Sie alle Zustandsmaschinen-Konfigurationsoptionen mit Ausnahme des Namens und des Typs des Zustandsmaschinen bearbeiten. Die folgende Abbildung zeigt einige der Konfigurationen, die Sie im Konfigurationsmodus angeben können.



State-Machine-Konfiguration verwalten

Gehen Sie wie folgt vor, um Ihre State-Machine-Konfiguration zu verwalten:

1. Geben Sie im Feld State Machine Name einen Namen für Ihren State Machine ein.

> **Tip**
>
> Wählen Sie alternativ das Bearbeitungssymbol neben dem Standardnamen der Zustandsmaschine von MyStateMachine. Geben Sie dann unter State-Machine-Konfiguration einen Namen an.

> **Important**
>
> Sie können den Namen der Zustandsmaschine nicht bearbeiten, nachdem Sie die Zustandsmaschine erstellt haben.

2. Wählen Sie unter Typ den Zustandsmaschinentyp Standard oder Express aus. Informationen zu Zustandsmaschinen finden Sie unterStandard- und Express-Workflows.

> **Important**
>
> Sie können den Zustandsmaschinentyp nicht mehr bearbeiten, nachdem Sie den Zustandsmaschine erstellt haben.

3. Wählen Sie unter Berechtigungen die IAM-Rolle aus, die als Ausführungsrolle für die Zustandsmaschine verwendet werden soll.
   - Neue Rolle erstellen (empfohlen): Wenn Sie diese Option auswählen, erstellt Step Functions automatisch eine Ausführungsrolle für Ihre Zustandsmaschinen mit den geringsten Rechten, die beim Erstellen der Zustandsmaschinen erforderlich sind. Diese automatisch generierten IAM-Rollen gelten für den, AWS-Region in dem Sie den Zustandsmaschine erstellen.

> **Tip**
>
> Um die Berechtigungen zu überprüfen, die Step Functions automatisch für Ihre Zustandsmaschine generiert, wählen Sie Automatisch generierte Berechtigungen überprüfen.

> **Note**
>
> Wenn Sie die von Step Functions erstellte IAM-Rolle löschen, kann Step Functions sie später nicht mehr neu erstellen. Ebenso kann Step Functions ihre ursprünglichen Einstellungen später nicht wiederherstellen, wenn Sie die Rolle ändern (z. B. indem Sie Step Functions aus den Principals in der IAM-Richtlinie entfernen).

- Wählen Sie eine vorhandene Rolle aus: Erstellen Sie Ihre eigene IAM-Rolle für die Zustandsmaschine und wählen Sie sie dann aus den unten aufgeführten Optionen Wählen Sie eine vorhandene Rolle aus. Stellen Sie sicher, dass die Rollenrichtlinie die Berechtigungen enthält, die der Zustandsmaschine zugewiesen werden sollen.

  Informationen zum Erstellen und Bearbeiten von IAM-Richtlinien finden Sie unter Erstellen von IAM-Richtlinien im IAM-Benutzerhandbuch.

- Geben Sie einen Rollen-ARN ein: Geben Sie den Amazon-Ressourcennamen (ARN) einer vorhandenen IAM-Rolle an, die für diese Zustandsmaschine verwendet werden soll. z. B. `arn:aws:iam::123456789012:role/service-role/StepFunctions-WorkflowStudio-role-777f4027`.

4. Legen Sie unter Logging die Protokollebene für Ihren State-Machine fest. Step Functions protokolliert die Ereignisse des Ausführungsverlaufs auf der Grundlage Ihrer Auswahl. Sie können eine der folgenden Optionen auswählen:

   - ALL: Alle Ereignistypen werden protokolliert.
   - FEHLER: Alle Fehlerereignistypen werden protokolliert, z. B. TaskFailed und ExecutionFailed.
   - SCHWERWIEGEND: Alle Ereignistypen mit schwerwiegenden Fehlern werden protokolliert, z. B. ExecutionAborted und ExecutionFailed.
   - AUS: Es werden keine Ereignistypen protokolliert.

   Weitere Hinweise zu Protokollebenen finden Sie unterProtokollstufen.

5. Stellen Sie unter Zusätzliche Konfiguration eine oder mehrere der folgenden optionalen Konfigurationen ein:

   - X-RayAblaufverfolgung aktivieren: Wählen Sie dieses Kontrollkästchen für Step Functions, an die Traces X-Ray für State-Machine-Ausführungen gesendet werden sollen, auch wenn keine Trace-ID von einem Upstream-Dienst übergeben wird. Weitere Informationen finden Sie unter AWS X-Ray und Step Functions.

- Version bei Erstellung veröffentlichen: Eine Version ist ein nummerierter, unveränderlicher Snapshot einer Zustandsmaschine, den Sie ausführen können. Wählen Sie dieses Kontrollkästchen, um eine Version Ihres Zustandsmaschinen zu veröffentlichen, während Sie den Zustandsmaschine erstellen. Step Functions veröffentlicht Version 1 als erste Revision der State Machine.

  Weitere Informationen zu Versionen erhalten Sie unter Versionen des Zustandsautomaten.
- Neues Tag hinzufügen: Wählen Sie dieses Feld, um Ihrer Zustandsmaschine Tags hinzuzufügen. Durch das Hinzufügen von Tags können Sie die mit Ihren Ressourcen verbundenen Kosten verfolgen und verwalten und Ihre IAM-Richtlinien besser schützen. Weitere Informationen zu Tags erhalten Sie unter Funktionen zum Taggen in Step.

6. Wählen Sie Erstellen.
7. Wählen Sie im Dialogfeld zur Bestätigung der Rollenerstellung die Option Bestätigen aus, um fortzufahren.

   Sie können auch Rollenkonfiguration anzeigen wählen, um zum Konfigurationsmodus zurückzukehren.

Tastenkombinationen

Workflow Studio unterstützt die folgenden Tastenkombinationen:

Tastaturkürzel	Funktion
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

Tastaturkürzel	Funktion
Backspace	Remove all selected states
Delete	Remove all selected states
Ctrl+D	Duplicate selected state

Verwenden von Workflow Studio

Erfahren Sie, wie Sie Workflows mit Step Functions Workflow Studio erstellen, bearbeiten und ausführen. Sobald Ihr Workflow fertig ist, können Sie ihn exportieren. Sie können Workflow Studio auch für Rapid Prototyping verwenden.

In diesem Thema

Erstellen Sie einen Workflow

In Workflow Studio können Sie entweder eine Startvorlage oder eine leere Vorlage wählen, um einen Workflow von Grund auf neu zu erstellen. Für leere Vorlagen können Sie den Entwurfs - oder Codemodus verwenden, um Ihren Workflow zu erstellen.

Eine Startvorlage ist ein ready-to-run Beispielprojekt, das automatisch den Workflow-Prototyp und die Workflow-Definition erstellt und alle zugehörigen AWS Ressourcen bereitstellt, die Ihr Projekt für Sie benötigt. AWS-Konto Sie können diese Startvorlagen verwenden, um sie unverändert bereitzustellen und auszuführen, oder die Workflow-Prototypen verwenden, um darauf aufzubauen. Weitere Informationen zu Startvorlagen finden Sie unterBeispielprojekte für Step Functions.

Erstellen Sie einen Workflow mithilfe von Startvorlagen

1. Öffnen Sie die Step Functions Functions-Konsole und wählen Sie Create State Machine.
2. Führen Sie im Dialogfeld „Vorlage auswählen“ einen der folgenden Schritte aus, um ein Beispielprojekt auszuwählen, z. B. das Task Timer-Beispielprojekt:

   - Geben Sie **`Task Timer`** in das Feld Nach Schlüsselwort suchen einen Text ein, und wählen Sie dann in den zurückgegebenen Suchergebnissen die Option Task-Timer aus.
   - Durchsuchen Sie die Beispielprojekte, die im rechten Bereich unter Alle aufgeführt sind, und wählen Sie dann Task Timer aus.
3. Wählen Sie Next (Weiter), um fortzufahren.
4. Step Functions listet die Funktionen auf, die in dem von Ihnen ausgewählten Beispielprojekt AWS-Services verwendet wurden. Außerdem wird ein Workflow-Diagramm für das Beispielprojekt angezeigt. Stellen Sie dieses Projekt für Ihr Projekt bereit AWS-Konto oder verwenden Sie es als Ausgangspunkt für die Erstellung Ihrer eigenen Projekte. Je nachdem, wie Sie vorgehen möchten, wählen Sie „Demo ausführen“ oder „Darauf aufbauen“.
5. Wählen Sie Vorlage verwenden, um mit Ihrer Auswahl fortzufahren.
6. Führen Sie eine der folgenden Aktionen aus:

   - Wenn Sie „Darauf aufbauen“ ausgewählt haben, erstellt Step Functions den Workflow-Prototyp für das von Ihnen ausgewählte Beispielprojekt. Step Functions stellt die in der Workflow-Definition aufgeführten Ressourcen nicht bereit. Ziehen Sie die Entwurfsmodus Status per Drag-and-Drop aus, Bundesstaaten-Browser um mit der Erstellung Ihres Workflow-Prototyps fortzufahren. Oder wechseln Sie zu demCodemodus, um die Amazon States Language (ASL-) Definition Ihres Workflows zu aktualisieren.

     > **Important**
     >
     > Denken Sie daran, den Platzhalter Amazon Resource Name (ARN) für die im Beispielprojekt verwendeten Ressourcen zu aktualisieren, bevor Sie Ihren Workflow ausführen.

   - Wenn Sie Eine Demo ausführen ausgewählt haben, erstellt Step Functions ein schreibgeschütztes Beispielprojekt, das eine AWS CloudFormation Vorlage verwendet, um die in dieser Vorlage aufgeführten AWS Ressourcen für Ihr Projekt bereitzustellen. AWS-Konto

> **Tip**
>
> Um die State-Machine-Definition des Beispielprojekts anzuzeigen, wählen Sie Code.

Wenn Sie bereit sind, wählen Sie Deploy and run aus, um das Beispielprojekt bereitzustellen und die Ressourcen zu erstellen.

> **Note**
>
> Es kann bis zu 10 Minuten dauern, bis diese Ressourcen und die zugehörigen IAM-Berechtigungen erstellt sind. Während der Bereitstellung Ihrer Ressourcen können Sie den CloudFormation Stack-ID-Link öffnen, um zu sehen, welche Ressourcen bereitgestellt werden.

> **Important**
>
> Für jeden in der CloudFormation Vorlage verwendeten Dienst fallen Standardgebühren an.

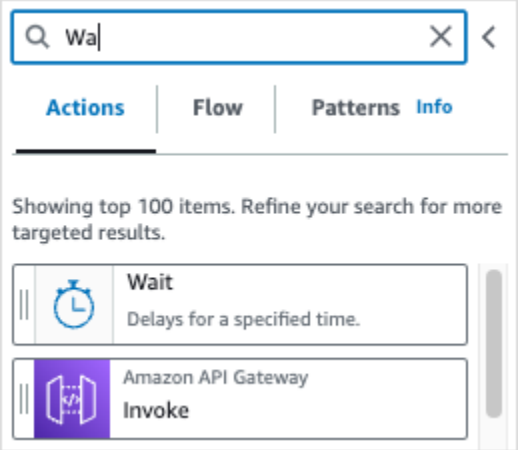
Erstellen Sie einen Workflow mit einer leeren Vorlage

1. Öffnen Sie die Step Functions Functions-Konsole.
2. Wählen Sie Create State Machine (Zustandsautomaten erstellen).
3. Wählen Sie im Dialogfeld Vorlage auswählen die Option Leer aus.
4. Wählen Sie Select (Auswählen). Dadurch wird Workflow Studio in geöffnetEntwurfsmodus.

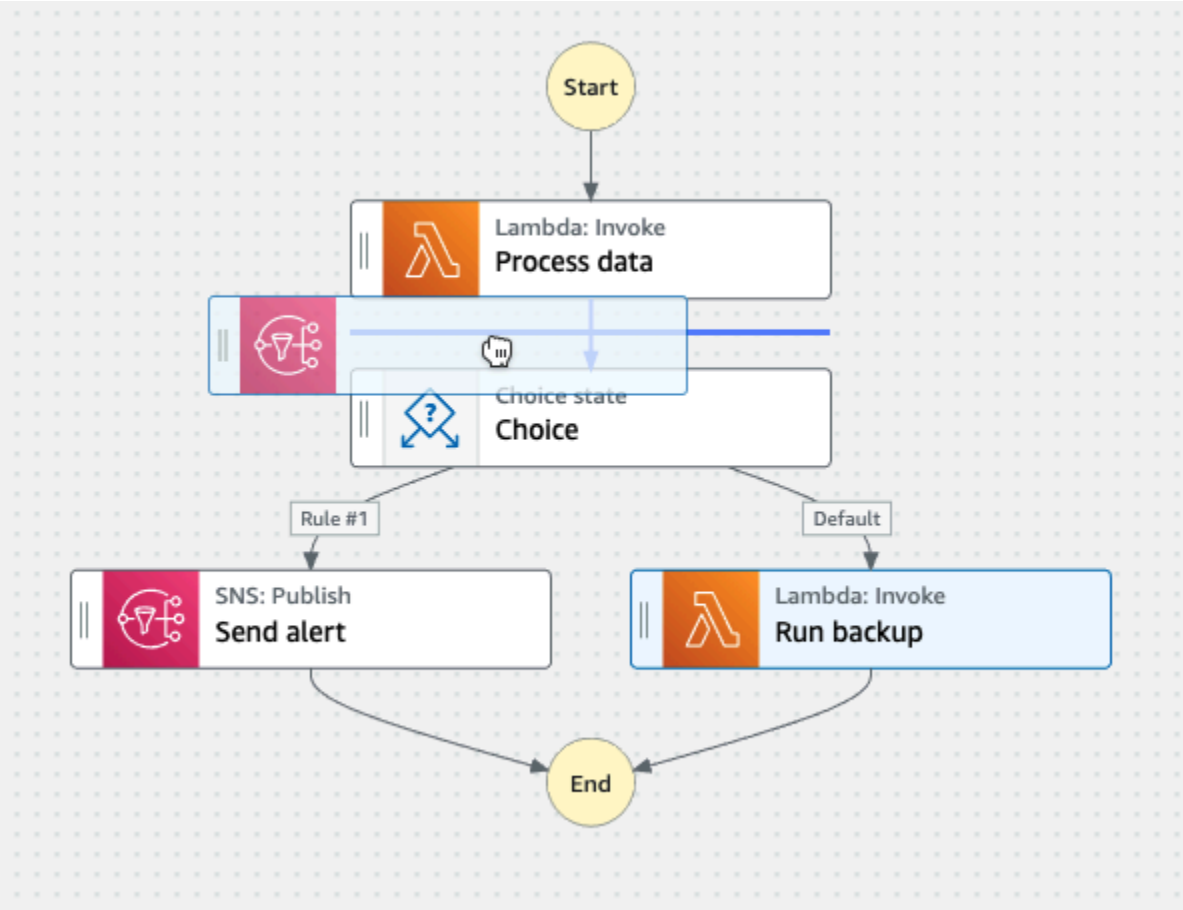
   Sie können jetzt mit dem Entwerfen Ihres Workflows beginnen Entwurfsmodus oder Ihre Workflow-Definition in schreibenCodemodus.
5. Wählen Sie Config, um die Konfiguration Ihres Workflows in der zu verwaltenKonfigurationsmodus. Geben Sie beispielsweise einen Namen für Ihren Workflow ein und wählen Sie dessen Typ aus.

Entwerfen Sie einen Workflow

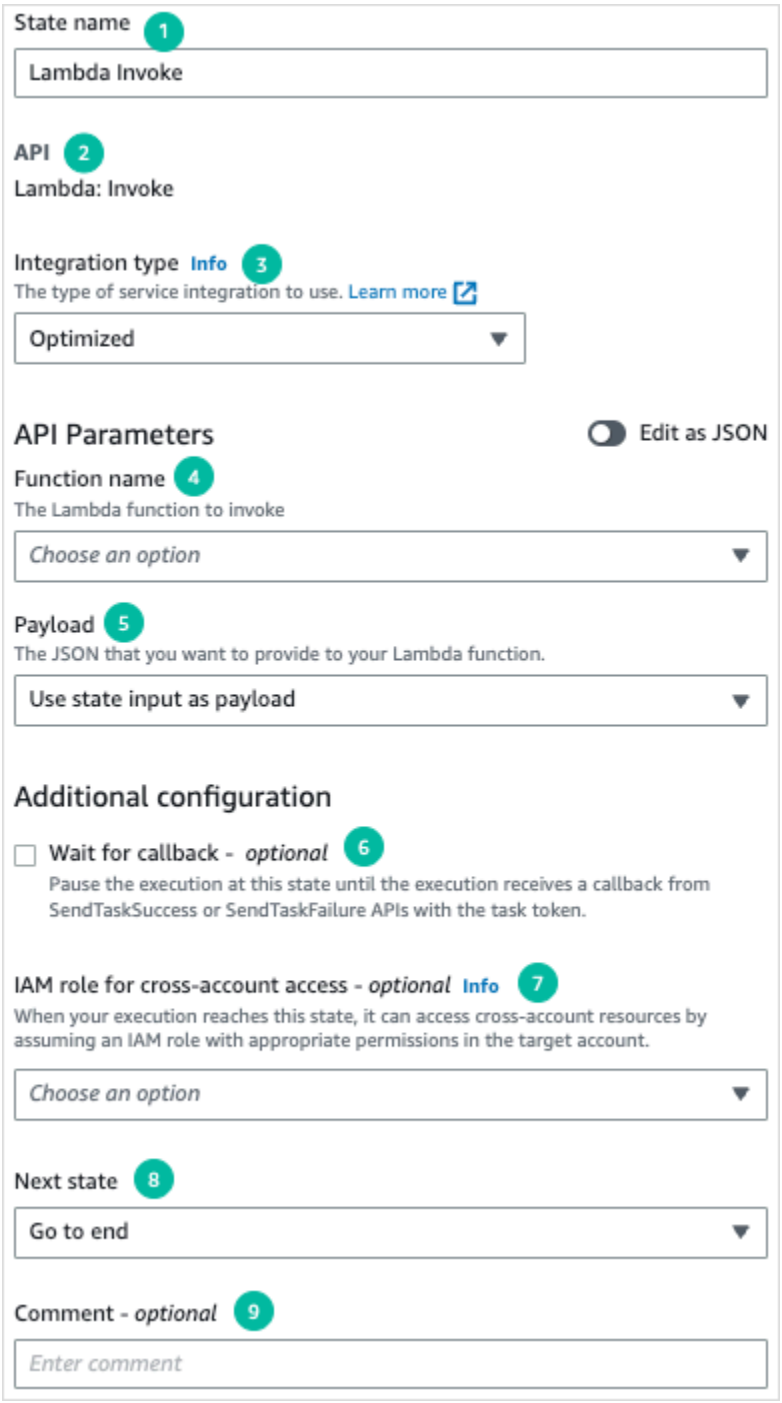
Wenn Sie den Namen des Status kennen, den Sie hinzufügen möchten, verwenden Sie das Suchfeld oben in der, Bundesstaaten-Browser um diesen Status auf den Registerkarten Aktionen und Flow von zu findenEntwurfsmodus.



Wählen Sie andernfalls einen Status aus dem Statusbrowser aus und ziehen Sie ihn per Drag & Drop auf die Arbeitsfläche. Platzieren Sie ihn an der gewünschten Stelle in Ihrem Workflow. Sie können Status in Ihrem Workflow auch neu anordnen, indem Sie sie an eine andere Stelle in Ihrem Workflow ziehen. Wenn Sie einen Status auf die Arbeitsfläche ziehen, wird an der Stelle, an der Sie ihn in Ihrem Workflow ablegen können, eine Linie angezeigt. Nachdem ein Status auf der Arbeitsfläche abgelegt wurde, wird sein Code automatisch generiert und zu Ihrer Workflow-Definition hinzugefügt. Um die Definition zu sehen, aktivieren Sie den Schalter Definition im Inspektorfenster. Um Änderungen an Ihrer Workflow-Definition vorzunehmen, wählen Sie denCodemodus, der einen integrierten Code-Editor bietet.



Nachdem Sie einen Status auf der Arbeitsfläche abgelegt haben, können Sie ihn im Inspector Bereich auf der rechten Seite konfigurieren. Dieser Bereich enthält die Registerkarten Konfiguration, Eingabe, Ausgabe und Fehlerbehandlung für jeden Status oder jede API-Aktion, die Sie auf der Arbeitsfläche platzieren. Sie konfigurieren die Status, die Sie in Ihre Workflows aufnehmen, auf der Registerkarte Konfiguration. Beispielsweise besteht die Registerkarte „Konfiguration“ für die Lambda-Aktion „API aufrufen“ aus den folgenden Optionen:



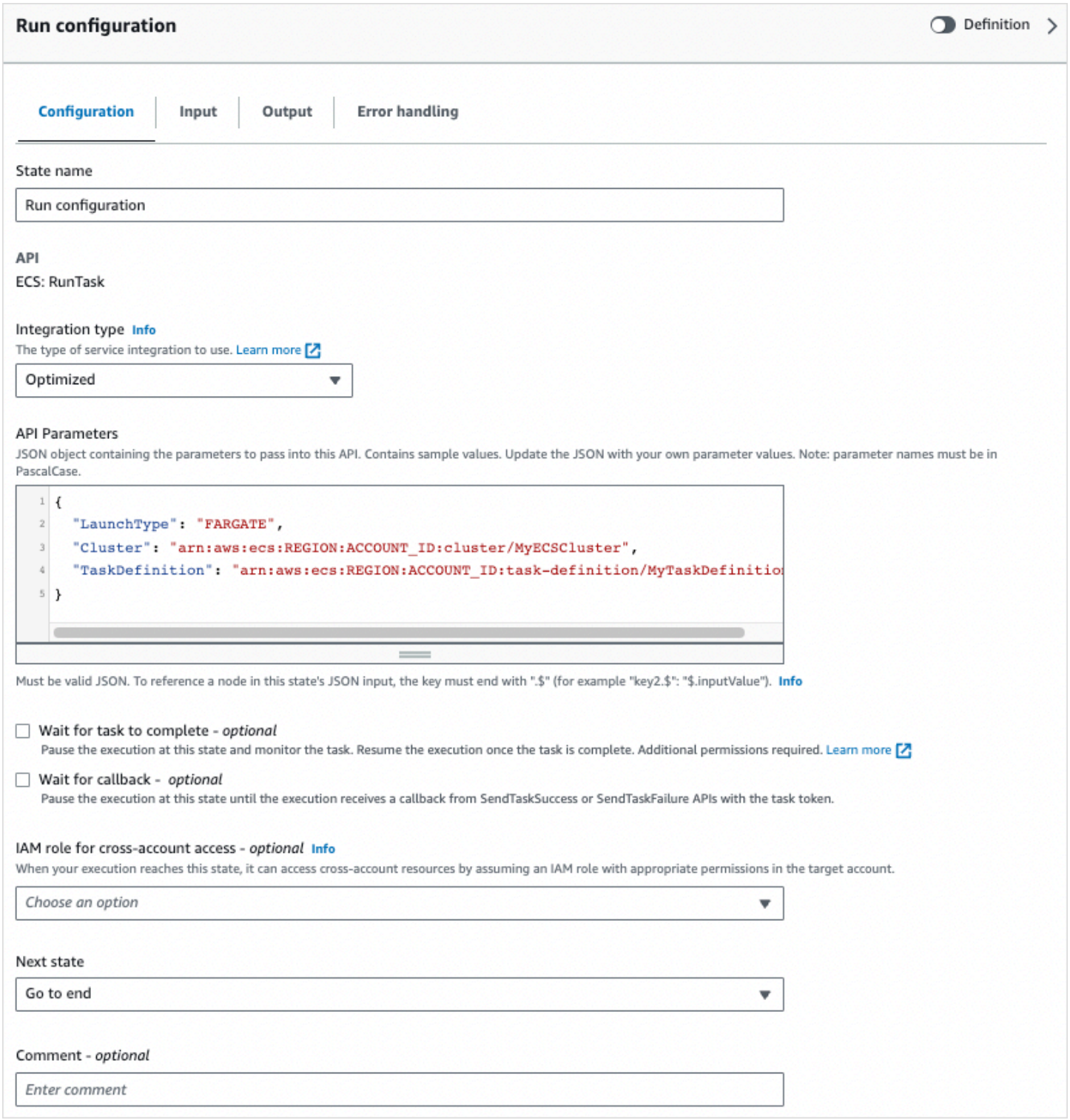
1. Der Name des Bundesstaates identifiziert den Staat. Sie können Ihren eigenen Namen verwenden oder den generierten Standardnamen akzeptieren.
2. Die API zeigt die vom Staat verwendete API-Aktion.
3. Die Dropdownliste Integrationstyp bietet Optionen zur Auswahl der Art der Serviceintegrationen, die in Step Functions verfügbar sind. Der von Ihnen gewählte Integrationstyp wird verwendet, um API-Aktionen einer bestimmten Person AWS-Service aus Ihrem Workflow aufzurufen.

4. Der Funktionsname bietet Optionen für:
   - Geben Sie einen Funktionsnamen ein: Sie können Ihren Funktionsnamen oder seinen ARN eingeben.
   - Funktionsnamen zur Laufzeit aus der Statuseingabe abrufen: Sie können diese Option verwenden, um den Funktionsnamen dynamisch aus der Statuseingabe abzurufen, basierend auf dem von Ihnen angegebenen Pfad.
   - Funktionsnamen wählen: Sie können direkt aus den Funktionen auswählen, die in Ihrem Konto und Ihrer Region verfügbar sind.
5. Mit der Payload können Sie aus den folgenden Optionen wählen:
   - Statuseingabe als Nutzlast verwenden: Sie können diese Option verwenden, um die Eingabe des Status als Nutzlast für Ihre Lambda-Funktion zu übergeben.
   - Geben Sie Ihre eigene Nutzlast ein: Sie können diese Option verwenden, um ein JSON-Objekt zu erstellen, das als Nutzlast an Ihre Lambda-Funktion übergeben wird. Dieses JSON kann sowohl statische Werte als auch Werte enthalten, die aus der Statuseingabe ausgewählt wurden.
   - Keine Nutzlast: Sie können diese Option verwenden, wenn Sie keine Nutzlast an Ihre Lambda-Funktion übergeben möchten.
6. (Optional) In einigen Bundesstaaten gibt es die Option „Auf Abschluss der Aufgabe warten“ oder „Auf Rückruf warten“. Sofern verfügbar, wählen diese Optionen eines der folgenden Serviceintegrationsmuster aus:
   - Keine Option ausgewählt: Step Functions verwendet das Request Response (Antwort anfordern) Integrationsmuster. Step Functions wartet auf eine HTTP-Antwort und wechselt dann zum nächsten Status. Step Functions wartet nicht darauf, dass ein Job abgeschlossen ist. Wenn keine Optionen verfügbar sind, verwendet der Staat dieses Muster.
   - Warten Sie, bis die Aufgabe abgeschlossen ist: Step Functions verwendet das Ausführen einer Aufgabe (.sync) Integrationsmuster.
   - Auf Rückruf warten: Step Functions verwendet das Warten auf einen Callback mit dem Aufgabentoken Integrationsmuster.
7. (Optional) Um auf Ressourcen zuzugreifen, die AWS-Konten in Ihren Workflows unterschiedlich konfiguriert sind, bietet Step Functions kontenübergreifenden Zugriff. Die IAM-Rolle für kontoübergreifenden Zugriff bietet Optionen für:
   - Geben Sie den ARN für die IAM-Rolle an: Geben Sie die IAM-Rolle an, die die entsprechenden Ressourcenzugriffsberechtigungen enthält. Diese Ressourcen sind in einem Zielkonto verfügbar, AWS-Konto auf das Sie kontenübergreifende Anrufe tätigen.

- Rufen Sie den ARN der IAM-Rolle zur Laufzeit aus der Statuseingabe ab: Geben Sie einen Referenzpfad zu einem vorhandenen Schlüssel-Wert-Paar in der JSON-Eingabe des Bundesstaates an, das die IAM-Rolle enthält.

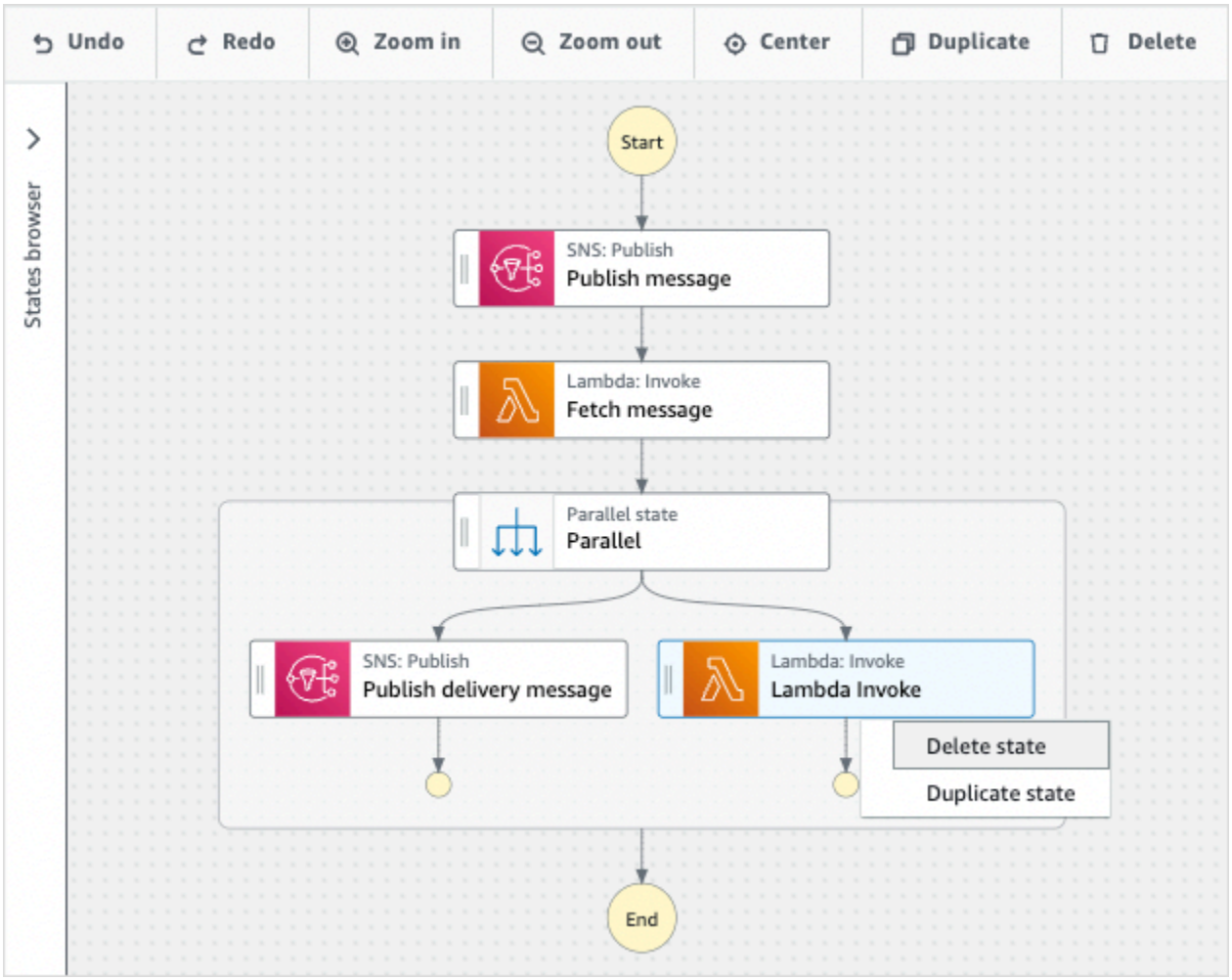
8. Im nächsten Status können Sie den Status auswählen, in den Sie als Nächstes wechseln möchten.
9. (Optional) Das Feld Kommentar kann verwendet werden, um Ihren eigenen Kommentar hinzuzufügen. Es hat keinen Einfluss auf den Arbeitsablauf, kann aber verwendet werden, um Ihren Workflow mit Anmerkungen zu versehen.

In einigen Staaten werden allgemeinere Konfigurationsoptionen verfügbar sein. Beispielsweise enthält die Amazon `RunTask` ECS-State-Konfiguration ein `API Parameters` Feld, das mit Platzhalterwerten gefüllt ist.



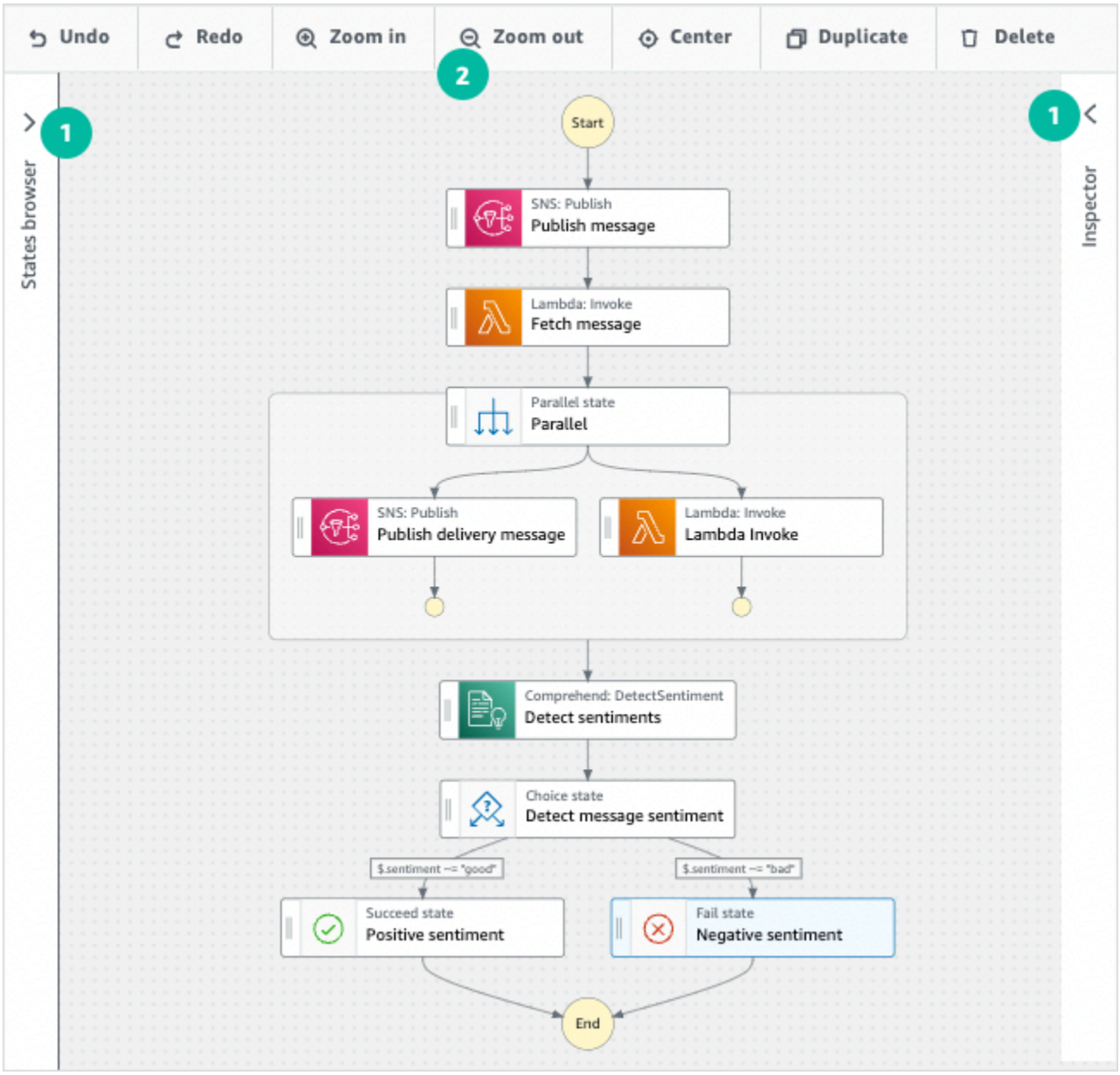
Für diese Bundesstaaten können Sie die Platzhalterwerte durch Konfigurationen ersetzen, die Ihren Anforderungen entsprechen.

Um einen Status zu löschen, können Sie die Rücktaste verwenden, mit der rechten Maustaste klicken und Status löschen wählen oder in der Entwurfssymbolleiste Löschen wählen.



Wenn Ihr Workflow wächst, passt er möglicherweise nicht in die Arbeitsfläche. Sie haben folgende Möglichkeiten:

1. Verwenden Sie die Steuerelemente an den Seitenbereichen, um die Größe der Bedienfelder zu ändern oder sie zu schließen.
2. Verwenden Sie die Steuerelemente der Design-Symbolleiste oben inLeinwand, um das Workflow-Diagramm zu vergrößern oder zu verkleinern.



Führen Sie Ihren Workflow aus

Nachdem Sie Ihren Workflow mit dem Workflow Studio erstellt oder bearbeitet haben, können Sie ihn ausführen und seine Ausführung in der Step Functions Functions-Konsole anzeigen.

Um einen Workflow in Workflow Studio auszuführen

1. Wählen Sie im Design -, Code - oder Konfigurationsmodus die Option Ausführen.

   Das Dialogfeld Ausführung starten wird auf einer neuen Registerkarte geöffnet.
2. Gehen Sie im Dialogfeld Ausführung starten wie folgt vor:

1. (Optional) Um Ihre Ausführung zu identifizieren, können Sie im Feld Name einen Namen dafür angeben. Standardmäßig generiert Step Functions automatisch einen eindeutigen Ausführungsnamen.

   > **Note**
   >
   > Mit Step Functions können Sie Namen für Zustandsmaschinen, Ausführungen und Aktivitäten sowie Beschriftungen erstellen, die Nicht-ASCII-Zeichen enthalten. Diese Nicht-ASCII-Namen funktionieren nicht mit Amazon. CloudWatch Um sicherzustellen, dass Sie CloudWatch Messwerte verfolgen können, wählen Sie einen Namen, der nur ASCII-Zeichen verwendet.

2. (Optional) Geben Sie in das Eingabefeld Eingabewerte im JSON-Format ein, um Ihren Workflow auszuführen.
3. Wählen Sie Start execution (Ausführung starten) aus.
4. Die Step Functions Functions-Konsole leitet Sie zu einer Seite weiter, die mit Ihrer Ausführungs-ID betitelt ist. Diese Seite wird als Seite mit den Ausführungsdetails bezeichnet. Auf dieser Seite können Sie die Ausführungsergebnisse im Verlauf oder nach Abschluss der Ausführung überprüfen.

   Um die Ausführungsergebnisse zu überprüfen, wählen Sie in der Diagrammansicht einzelne Status aus und wählen Sie dann die einzelnen Registerkarten im Schrittdetails Bereich, um die Details der einzelnen Status, einschließlich Eingabe, Ausgabe und Definition, anzuzeigen. Einzelheiten zu den Ausführungsinformationen, die Sie auf der Seite mit den Ausführungsdetails einsehen können, finden Sie unterSeite mit Ausführungsdetails – Schnittstellenübersicht.

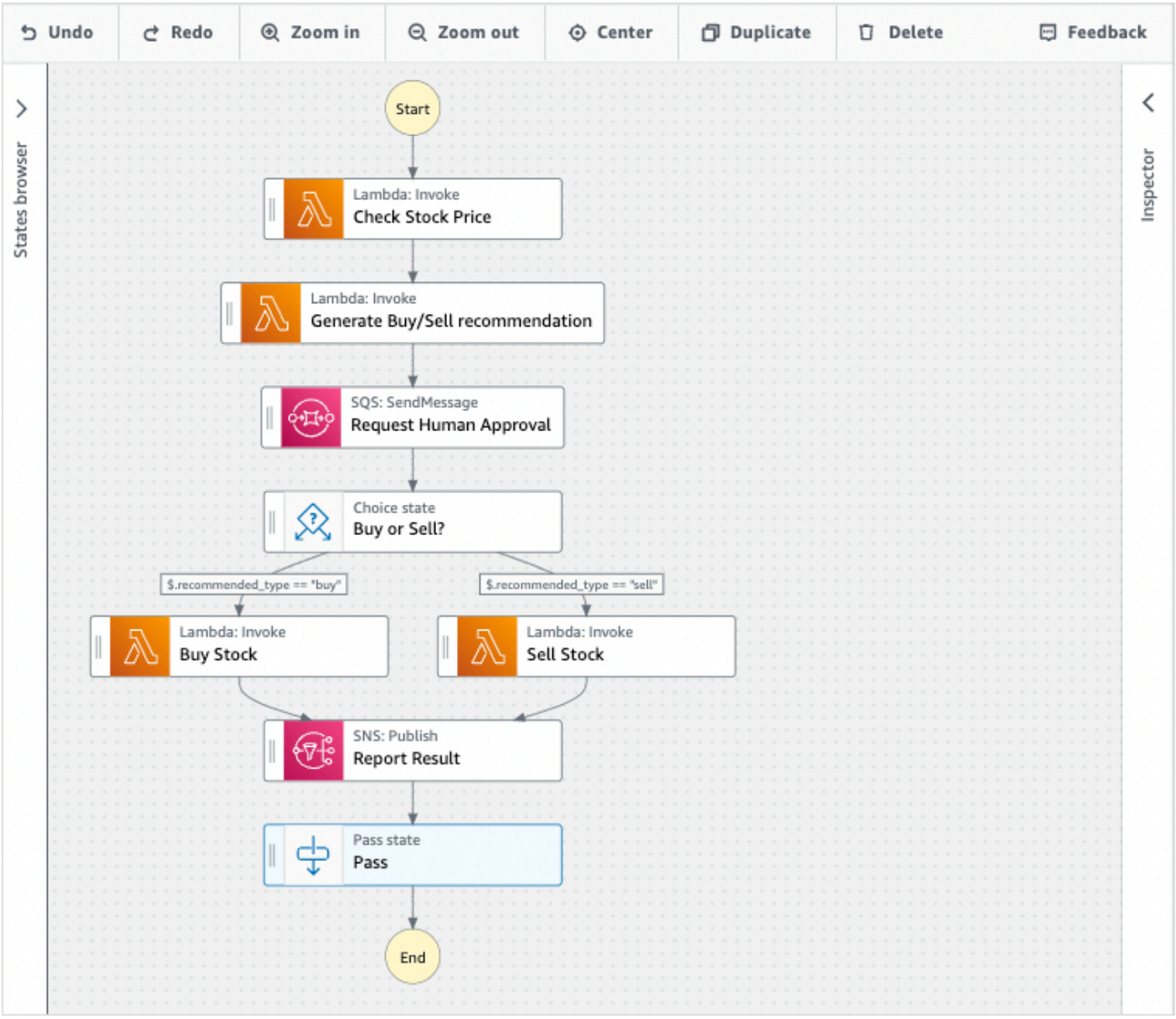
Bearbeiten Sie Ihren Workflow

Sie können einen vorhandenen Workflow visuell im Entwurfsmodus Workflow Studio bearbeiten. Sie können die Workflow-Definition auch in Codemodus Workflow Studio bearbeiten.

Um einen vorhandenen Workflow zu bearbeiten:

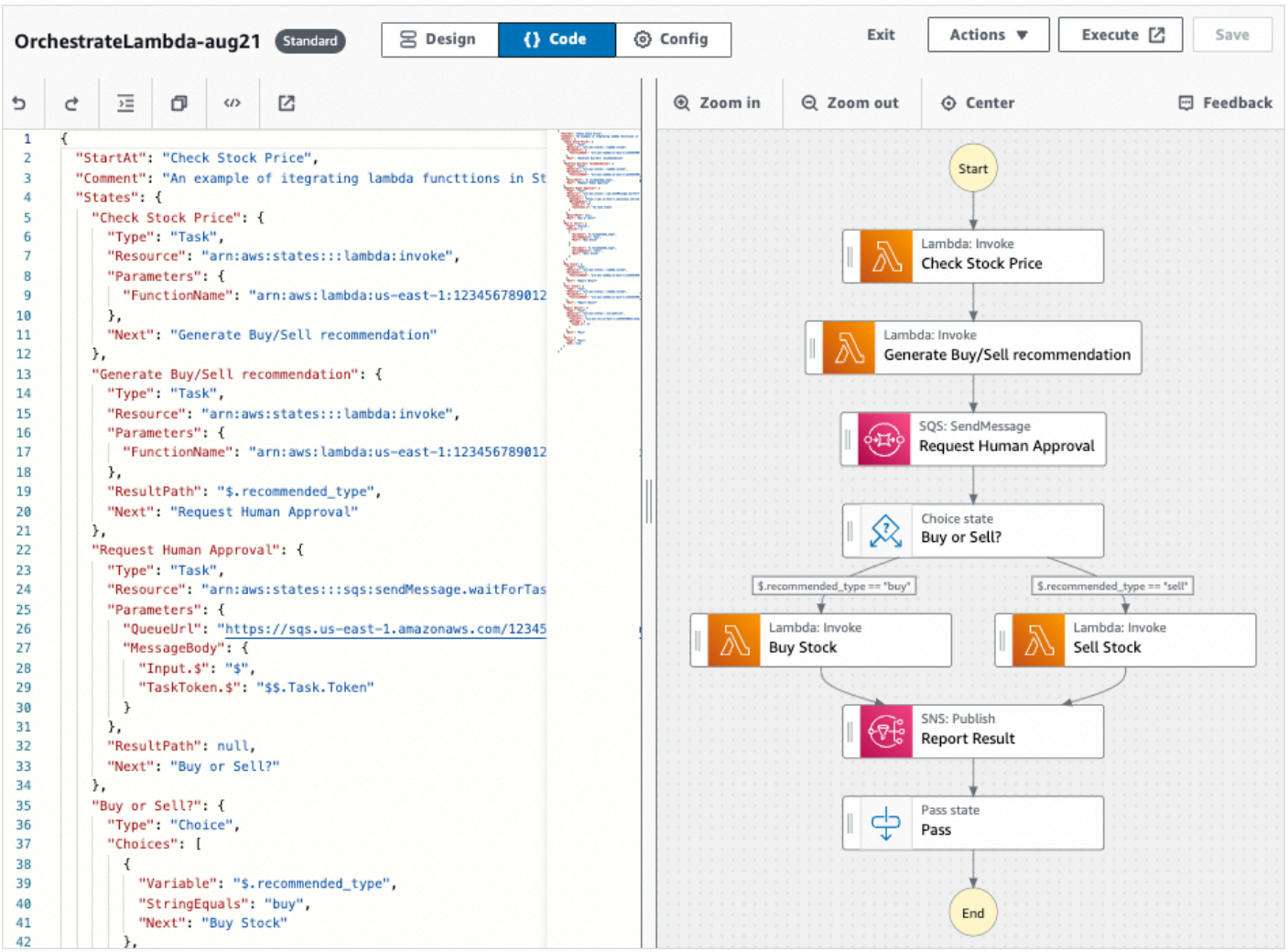
1. Öffnen Sie die Step Functions Functions-Konsole.
2. Wählen Sie auf der Seite State Machines den Workflow aus, den Sie bearbeiten möchten.

3. Wählen Sie auf der Detailseite der Zustandsmaschine die Option Bearbeiten aus.
4. Der Workflow wird im Entwurfsmodus von Workflow Studio geöffnet. Bearbeiten Sie den Workflow nach Bedarf.



> **Note**
>
> Wenn Sie Fehler in Ihrem Workflow sehen, müssen Sie diese im Entwurfsmodus beheben. Sie können nicht in den Code - oder Konfigurationsmodus wechseln, wenn in Ihrem Workflow Fehler auftreten.

5. (Optional) Wählen Sie die Code-Schaltfläche, um die Workflow-Definition in Workflow Studio anzuzeigen oder zu bearbeiten.



6. Wenn Sie fertig sind, wählen Sie Speichern, um Ihren aktualisierten Workflow zu speichern.
7. (Optional) Um Ihren aktualisierten Workflow auszuführen, wählen Sie Ausführen. Das Dialogfeld „Ausführung starten“ wird auf einer neuen Registerkarte geöffnet.

Exportieren Sie Ihren Workflow

Sie können die Definition Ihres Workflows Amazon States Language (ASL) und Ihr Workflow-Diagramm exportieren:

1. Wählen Sie Ihren Workflow in der Step Functions Functions-Konsole aus.
2. Wählen Sie auf der Seite mit den State-Machine-Details die Option Bearbeiten aus.

3. (Optional) Ihr Workflow wird im Entwurfsmodus von Workflow Studio geöffnet. Bearbeiten Sie Ihren Workflow im Designmodus oder wechseln Sie in den Codemodus.
4. Wählen Sie die Dropdownschaltfläche „Aktionen“ und führen Sie dann eine oder beide der folgenden Aktionen aus:
   - Um das Workflow-Diagramm in eine SVG- oder PNG-Datei zu exportieren, wählen Sie unter Diagramm exportieren das gewünschte Format aus.
   - Um die Workflow-Definition als JSON- oder YAML-Datei zu exportieren, wählen Sie unter Exportdefinition das gewünschte Format aus.

Erstellen Sie Ihren Workflow-Prototyp

Sie können Workflow Studio verwenden, um Prototypen neuer Workflows zu erstellen, die Platzhalterressourcen enthalten. Sie können Ihre Workflows auch mit Workflow Studio in Application Composer erstellen. Um einen Prototyp zu erstellen:

1. Melden Sie sich bei der Step Functions Functions-Konsole an.
2. Wählen Sie Create State Machine (Zustandsautomaten erstellen).
3. Wählen Sie im Dialogfeld Vorlage auswählen die Option Leer aus.
4. Wählen Sie Select (Auswählen). Dadurch wird Workflow Studio in geöffnetEntwurfsmodus.
5. Der Entwurfsmodus von Workflow Studio wird geöffnet. Entwerfen Sie Ihren Workflow in Workflow Studio. Um Platzhalter-Ressourcen einzubeziehen:
   a. Wählen Sie den Bundesstaat aus, für den Sie eine Platzhalterressource hinzufügen möchten, und gehen Sie dann unter Konfiguration wie folgt vor:
      - Wählen Sie für Lambda Invoke-Status die Option Funktionsname und dann Funktionsname eingeben aus. Sie können auch einen benutzerdefinierten Namen für Ihre Funktion eingeben.
      - Wählen Sie für den Status „Nachricht senden“ von Amazon SQS die Option „Warteschlangen-URL“ und anschließend „Warteschlangen-URL eingeben“. Geben Sie eine Platzhalter-URL für die Warteschlange ein.
      - Wählen Sie für Amazon SNS Publish States unter Thema einen Themen-ARN aus.
      - Für alle anderen Staaten, die unter Aktionen aufgeführt sind, können Sie die Standardkonfiguration verwenden.

> **Note**
>
> Wenn Sie Fehler in Ihrem Arbeitsablauf feststellen, müssen Sie diese im Entwurfsmodus beheben. Sie können nicht in den Code - oder Konfigurationsmodus wechseln, wenn in Ihrem Workflow Fehler auftreten.

b. (Optional) Um die automatisch generierte ASL-Definition Ihres Workflows anzuzeigen, wählen Sie Definition.

c. (Optional) Um die Workflow-Definition in Workflow Studio zu aktualisieren, wählen Sie die Code-Schaltfläche.

> **Note**
>
> Wenn Sie Fehler in Ihrer Workflow-Definition sehen, müssen Sie diese im Codemodus beheben. Sie können nicht in den Entwurfs - oder Konfigurationsmodus wechseln, wenn in Ihrer Workflow-Definition Fehler auftreten.

6. (Optional) Um den Namen des Zustandsmaschinen zu bearbeiten, wählen Sie das Bearbeitungssymbol neben dem Standardnamen des Zustandsmaschinen von MyStateMachineund geben Sie im Feld Name des Zustandsmaschinen einen Namen ein.

   Sie können auch zum wechseln, Konfigurationsmodus um den Standardnamen des Zustandsmaschinen zu bearbeiten.

7. Geben Sie Ihre Workflow-Einstellungen an, z. B. den Zustandsmaschinentyp und seine Ausführungsrolle.

8. Wählen Sie Erstellen.

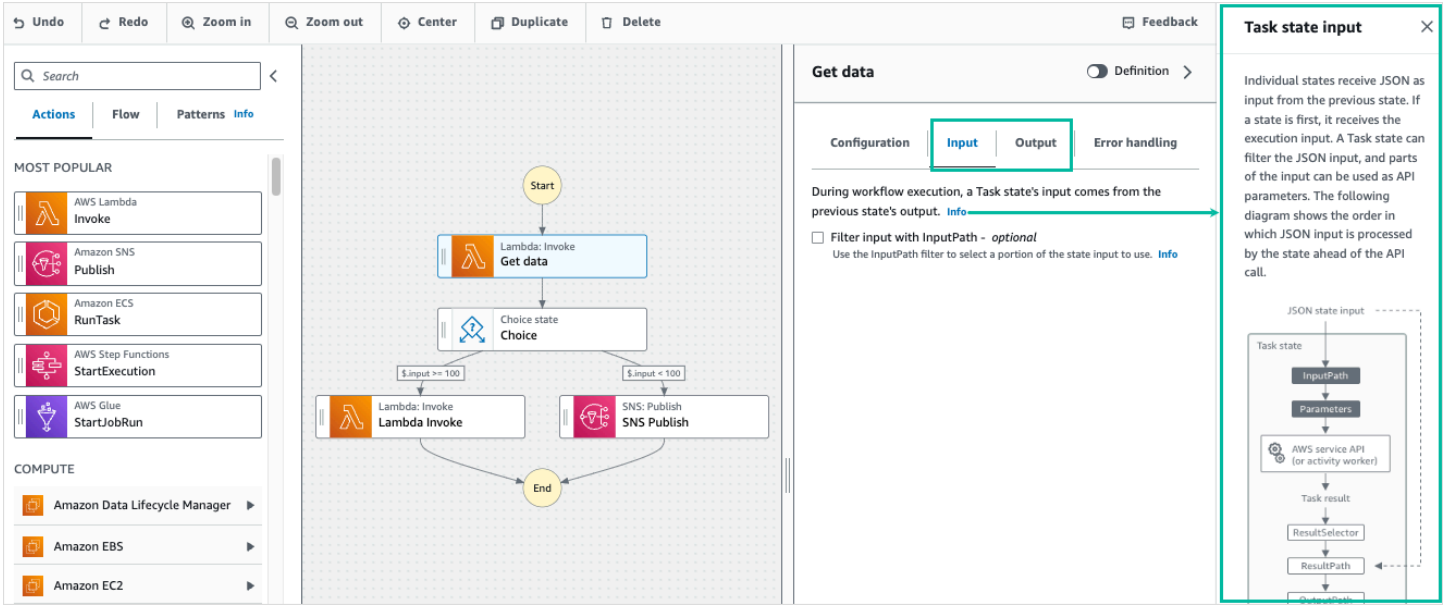
Sie haben jetzt einen neuen Workflow mit Platzhalterressourcen erstellt, die für Prototypen verwendet werden können. Sie können Ihre Workflow-Definition und das Workflow-Diagramm exportieren.

- Um Ihre Workflow-Definition als JSON- oder YAML-Datei zu exportieren, wählen Sie im Design - oder Codemodus die Drop-down-Schaltfläche Aktionen aus. Wählen Sie dann unter Definition exportieren das Format aus, das Sie exportieren möchten. Sie können diese exportierte Definition als Ausgangspunkt für die lokale Entwicklung mit dem verwenden AWS Toolkit for Visual Studio Code.

- Um Ihr Workflow-Diagramm in eine SVG- oder PNG-Datei zu exportieren, wählen Sie im Design - oder Codemodus die Dropdownschaltfläche Aktionen aus. Wählen Sie dann unter Exportdefinition das gewünschte Format aus.

Konfigurieren Sie Eingaben und Ausgaben für Ihre Bundesstaaten

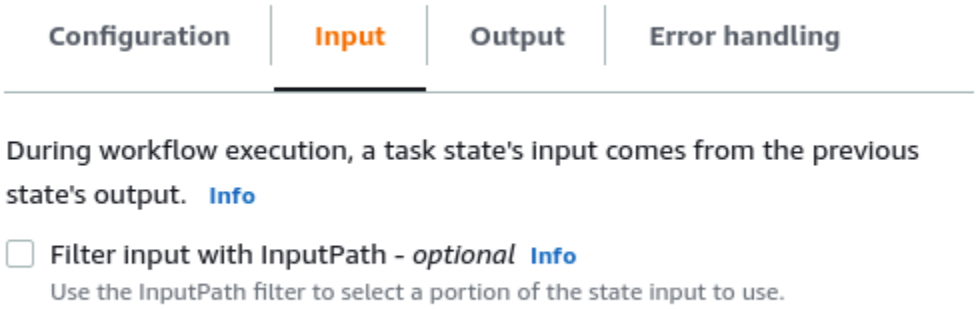
Jeder Staat trifft auf der Grundlage der eingegangenen Eingaben eine Entscheidung oder führt eine Aktion durch. In den meisten Fällen leitet er dann die Ausgabe an andere Staaten weiter. In Workflow Studio können Sie auf den Registerkarten Eingabe und Ausgabe des Bedienfelds konfigurieren, wie ein Status seine Eingabe- und Ausgabedaten filtert und manipuliert. Inspector Verwenden Sie die Informationslinks, um bei der Konfiguration von Eingaben und Ausgaben auf die kontextuelle Hilfe zuzugreifen.



Ausführliche Informationen darüber, wie Step Functions Eingabe und Ausgabe verarbeitet, finden Sie unterEingabe- und Ausgabeverarbeitung in Step Functions.

Konfigurieren Sie die Eingabe für einen Status

Jeder Status erhält Eingaben aus dem vorherigen Status als JSON. Wenn Sie die Eingabe filtern möchten, können Sie den `InputPath` Filter auf der Registerkarte Eingabe im Inspector Bereich verwenden. Das `InputPath` ist eine Zeichenfolge, die mit beginnt $ und einen bestimmten JSON-Knoten identifiziert. Diese werden als Referenzpfade bezeichnet und folgen der JsonPath Syntax.



Um die Eingabe zu filtern:

- Wählen Sie Eingabe filtern mit InputPath.
- Geben Sie einen JsonPathfür den `InputPath` Filter gültigen Wert ein. z. B. **`$.data`**.

Ihr `InputPath` Filter wird zu Ihrem Workflow hinzugefügt.

Example Beispiel 1: Verwenden Sie den InputPath Filter in Workflow Studio

Angenommen, die Eingabe für Ihren Status enthält die folgenden JSON-Daten.

```
{
  "comment": "Example for InputPath",
  "dataset1": {
    "val1": 1,
    "val2": 2,
    "val3": 3
  },
  "dataset2": {
    "val1": "a",
    "val2": "b",
    "val3": "c"
  }
}
```

Um den `InputPath` Filter anzuwenden, wählen Sie Eingabe filtern mit InputPath und geben Sie dann einen entsprechenden Referenzpfad ein. Wenn Sie eingeben**`$.dataset2.val1`**, wird der folgende JSON-Code als Eingabe an den Status übergeben.

```
{"a"}
```

Ein Referenzpfad kann auch eine Auswahl von Werten enthalten. Wenn es sich bei den Daten, auf die Sie verweisen, um Daten handelt `{ "a": [1, 2, 3, 4] }` und Sie den Referenzpfad `$.a[0:2]` als `InputPath` Filter anwenden, ist das folgende Ergebnis.

```
[ 1, 2 ]
```

ParallelZuordnung, und Pass Flow-Status verfügen über eine zusätzliche Eingabefilteroption, die auf der jeweiligen `Parameters` Registerkarte Eingabe aufgerufen wird. Dieser Filter wird nach dem InputPath Filter wirksam und kann verwendet werden, um ein benutzerdefiniertes JSON-Objekt zu erstellen, das aus einem oder mehreren Schlüssel-Wert-Paaren besteht. Bei den Werten der einzelnen Paare kann es sich entweder um statische Werte handeln, sie können aus der Eingabe ausgewählt werden, oder sie können Context-Objekt mit einem Pfad aus dem ausgewählt werden.

> **Note**
>
> Um anzugeben, dass ein Parameter einen Referenzpfad verwendet, um auf einen JSON-Knoten in der Eingabe zu verweisen, muss der Parametername mit enden `.$`.

Example Beispiel 2: Erstellen Sie eine benutzerdefinierte JSON-Eingabe für den Parallel-Status

Angenommen, die folgenden JSON-Daten sind die Eingabe für einen Parallelstatus.

```
{
  "comment": "Example for Parameters",
  "product": {
    "details": {
      "color": "blue",
      "size": "small",
      "material": "cotton"
    },
    "availability": "in stock",
    "sku": "2317",
    "cost": "$23"
  }
}
```

Um einen Teil dieser Eingabe auszuwählen und zusätzliche Schlüssel-Wert-Paare mit einem statischen Wert zu übergeben, können Sie im Feld Parameter auf der Registerkarte Eingabe des Status Parallel Folgendes angeben.

```
{
 "comment": "Selecting what I care about.",
 "MyDetails": {
    "size.$": "$.product.details.size",
    "exists.$": "$.product.availability",
    "StaticValue": "foo"
    }
 }
```

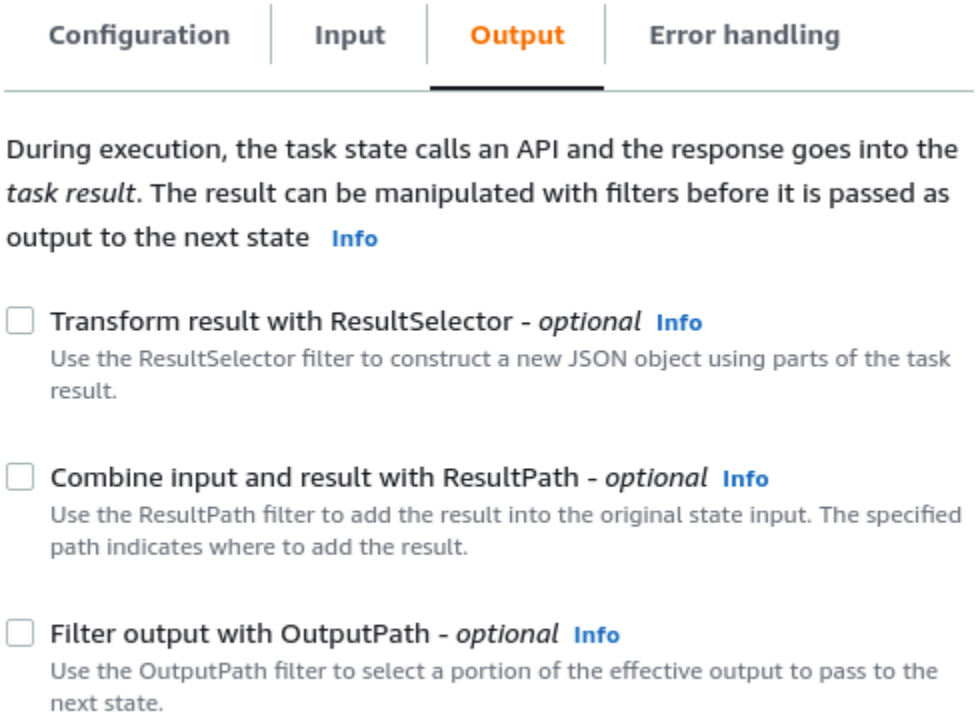
Die folgenden JSON-Daten werden das Ergebnis sein.

```
{
  "comment": "Selecting what I care about.",
  "MyDetails": {
    "size": "small",
    "exists": "in stock",
    "StaticValue": "foo"
  }
}
```

Konfigurieren Sie die Ausgabe eines Zustands

Jeder Status erzeugt eine JSON-Ausgabe, die gefiltert werden kann, bevor sie an den nächsten Status übergeben wird. Es sind mehrere Filter verfügbar, und jeder wirkt sich auf unterschiedliche Weise auf die Ausgabe aus. Die für jeden Status verfügbaren Ausgabefilter sind auf der Registerkarte „Ausgabe“ im Inspektorfenster aufgeführt. Für Status der Aufgabe Bundesstaaten werden alle von Ihnen ausgewählten Ausgabefilter in dieser Reihenfolge verarbeitet:

1. `ResultSelector`: Verwenden Sie diesen Filter, um das Ergebnis des Status zu manipulieren. Sie können ein neues JSON-Objekt mit Teilen des Ergebnisses erstellen.
2. `ResultPath`: Verwenden Sie diesen Filter, um eine Kombination aus der Statuseingabe und dem Aufgabenergebnis auszuwählen, die an die Ausgabe übergeben werden soll.
3. `OutputPath`: Verwenden Sie diesen Filter, um die JSON-Ausgabe zu filtern und auszuwählen, welche Informationen aus dem Ergebnis an den nächsten Status übergeben werden.



Benutze ResultSelector

`ResultSelector`ist ein optionaler Ausgabefilter für die folgenden Staaten:

- Status der AufgabeStaaten, bei denen es sich ausschließlich um Staaten handelt, die auf der Registerkarte Aktionen von aufgeführt sindBundesstaaten-Browser.
- ZuordnungStaaten, auf der Registerkarte Flow des States-Browsers.
- ParallelStaaten, auf der Registerkarte Flow des States-Browsers.

`ResultSelector`kann verwendet werden, um ein benutzerdefiniertes JSON-Objekt zu erstellen, das aus einem oder mehreren Schlüssel-Wert-Paaren besteht. Die Werte jedes Paares können entweder statische Werte sein oder aus dem Ergebnis des Zustands mit einem Pfad ausgewählt werden.

> **Note**
>
> Um anzugeben, dass ein Parameter einen Pfad verwendet, um auf einen JSON-Knoten im Ergebnis zu verweisen, muss der Parametername mit enden`.$`.

Example Beispiel für die Verwendung eines ResultSelector Filters

In diesem Beispiel verwenden Sie, `ResultSelector` um die Antwort des Amazon CreateCluster EMR-API-Aufrufs für einen Amazon `CreateCluster` EMR-Status zu manipulieren. Das Folgende ist das Ergebnis des Amazon `CreateCluster` EMR-API-Aufrufs.

```
{
  "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"
  }
}
```

Um einen Teil dieser Informationen auszuwählen und ein zusätzliches Schlüssel-Wert-Paar mit einem statischen Wert zu übergeben, geben Sie Folgendes in das ResultSelectorFeld auf der Registerkarte Ausgabe des Bundesstaates ein.

```
{
 "result": "found",
 "ClusterId.$": "$.output.ClusterId",
 "ResourceType.$": "$.resourceType"
 }
```

Die Verwendung `ResultSelector` führt zu dem folgenden Ergebnis.

```
{
 "result": "found",
 "ClusterId": "AKIAIOSFODNN7EXAMPLE",
 "ResourceType": "elasticmapreduce"
```

```
}
```

Benutzen ResultPath

Die Ausgabe eines Zustands kann eine Kopie seiner Eingabe, des von ihm erzeugten Ergebnisses oder eine Kombination aus Eingabe und Ergebnis sein. Verwenden Sie `ResultPath`, um zu steuern, welche Kombination daraus an die Statusausgabe weitergeleitet wird. Weitere Anwendungsfälle von finden `ResultPath` Sie unterResultPath.

`ResultPath`ist ein optionaler Ausgabefilter für die folgenden Zustände:

- Status der AufgabeStaaten, das sind alles Staaten, die auf der Registerkarte „Aktionen“ des Status-Browsers aufgeführt sind.
- ZuordnungStaaten, auf der Registerkarte Flow des States-Browsers.
- ParallelStaaten, auf der Registerkarte Flow des States-Browsers.
- PassStaaten, auf der Registerkarte Flow des States-Browsers.

`ResultPath`kann verwendet werden, um das Ergebnis zur ursprünglichen Statuseingabe hinzuzufügen. Der angegebene Pfad gibt an, wo das Ergebnis hinzugefügt werden soll.

Example Beispiel für die Verwendung eines ResultPath Filters

Nehmen wir an, das Folgende ist die Eingabe für einen Task-Status.

```
{
  "details": "Default example",
  "who": "AWS Step Functions"
}
```

Das Ergebnis des Task-Status ist das Folgende.

```
Hello, AWS Step Functions
```

Sie können dieses Ergebnis zur Eingabe des Status hinzufügen, indem Sie einen Referenzpfad anwenden `ResultPath` und eingeben, der angibt, wo das Ergebnis hinzugefügt werden soll, z. B.`$.taskresult`:

Damit `ResultPath` ist das Folgende das JSON, das als Ausgabe des Status übergeben wird.

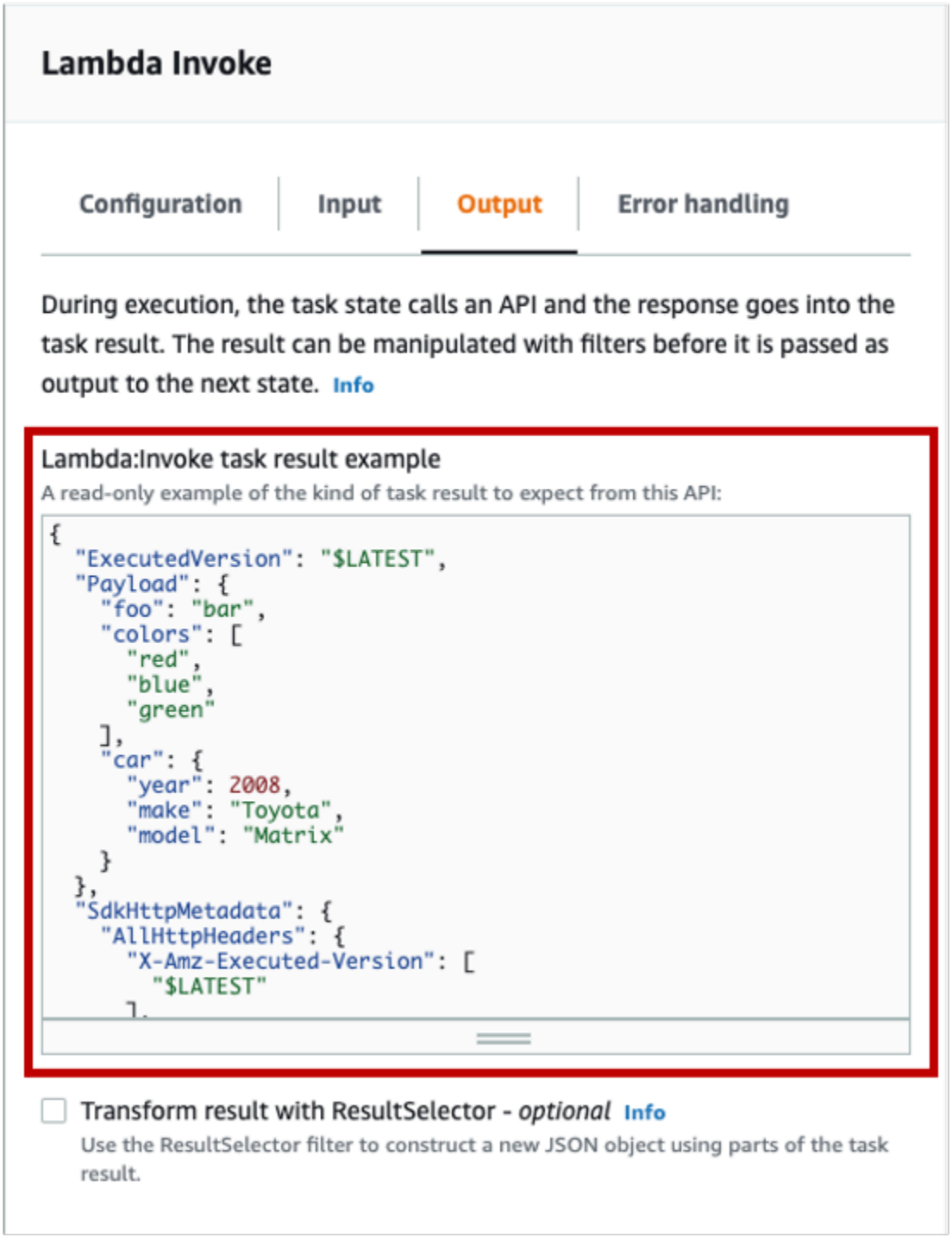
```
{
  "details": "Default example",
  "who": "AWS Step Functions",
  "taskresult": "Hello, AWS Step Functions!"
}
```

Benutze OutputPath

Mit dem `OutputPath` Filter können Sie unerwünschte Informationen herausfiltern und nur den Teil von JSON übergeben, der Ihnen wichtig ist. Das `OutputPath` ist eine Zeichenfolge, die mit beginnt $ und Knoten innerhalb von JSON-Text identifiziert.

Example Beispiel für die Verwendung eines OutputPath Filters

Ein Lambda Invoke-API-Aufruf gibt zusätzlich zur Nutzlast, die das Ergebnis der Lambda-Funktion ist, Metadaten zurück. Ein Beispiel für die Antwort auf diesen API-Aufruf wird auf der Registerkarte Ausgabe des Bundesstaates angezeigt.



Sie können `OutputPath` es verwenden, um die zusätzlichen Metadaten herauszufiltern. Standardmäßig lautet der Wert des OutputPathFilters für Lambda Invoke-Zustände, die mit dem Workflow Studio erstellt wurden. `$.Payload` Dieser Standardwert entfernt die zusätzlichen Metadaten und gibt eine Ausgabe zurück, die der direkten Ausführung der Lambda-Funktion entspricht.

Das Beispiel für das Ergebnis der Lambda-Invoke-Aufgabe und der Wert von `$.Payload` für den Ausgabefilter übergeben die folgenden JSON-Daten als Ausgabe.

```
{
 "foo": "bar",
 "colors": [
      "red",
      "blue",
      "green"
 ],
 "car": {
      "year": 2008,
      "make": "Toyota",
      "model": "Matrix"
 }
}
```

> **Note**
>
> Da der `OutputPath` Filter der letzte Ausgabefilter ist, der wirksam wird, sollten Sie, wenn Sie zusätzliche Ausgabefilter wie `ResultSelector` oder verwenden`ResultPath`, den Standardwert von `$.Payload` für den `OutputPath` Filter entsprechend ändern.

Ausführungsrollen in Workflow Studio

Jede Step Functions Zustandsmaschine benötigt eine AWS Identity and Access Management (IAM) -Rolle, die der Zustandsmaschine die Berechtigung erteilt, Aktionen auf AWS-Services Ressourcen auszuführen oder APIs von Drittanbietern aufzurufen. Diese Rolle wird als Ausführungsrolle bezeichnet. Diese Rolle muss IAM Richtlinien für jede Aktion enthalten, z. B. Richtlinien, die es der Zustandsmaschine ermöglichen, eine AWS Lambda Funktion aufzurufen, einen AWS Batch Job auszuführen oder die Stripe-API aufzurufen. Step Functionserfordert in den folgenden Fällen, dass Sie eine Ausführungsrolle angeben:

- Sie erstellen eine Zustandsmaschine in der Konsole, in den AWS SDKs oder AWS CLI mithilfe der CreateStateMachineAPI.
- Sie testen einen Status in der Konsole, in den AWS SDKs oder AWS CLI mithilfe der TestStateAPI.

Workflow Studio verfügt über Funktionen, mit denen Sie die Ausführungsrollen für Ihre Workflows einfach verwalten können.

Themen

Über automatisch generierte Rollen

Wenn Sie in der Step Functions Konsole eine Zustandsmaschine erstellen, kann Workflow Studio automatisch eine Ausführungsrolle für Sie erstellen, die die erforderlichen IAM Richtlinien enthält. Workflow Studio analysiert Ihre State-Machine-Definition und generiert Richtlinien mit den geringsten Rechten, die zur Ausführung Ihres Workflows erforderlich sind.

Workflow Studio kann IAM Richtlinien für Folgendes generieren:

- HTTP-Aufgaben, die APIs von Drittanbietern aufrufen.
- Aufgabenstatus, die andere AWS-Services mithilfe optimierter Integrationen aufrufen, z. B. LambdaInvoke DynamoDB GetItem, oder. AWS Glue StartJobRun
- Aufgabenstatus, die verschachtelte Workflows ausführen.
- Status verteilter Karten, einschließlich Richtlinien zum Starten untergeordneter Workflow-Ausführungen, zum Auflisten von Amazon S3 Buckets und zum Lesen oder Schreiben von S3-Objekten.
- X-RayAblaufverfolgung. Jede Rolle, die in Workflow Studio automatisch generiert wird, enthält eine Richtlinie, die dem Zustandsmaschine Berechtigungen zum Senden von Traces erteilt. X-Ray
- Protokollierung mitCloudWatchProtokollewenn die Protokollierung auf der Zustandsmaschine aktiviert ist.

Workflow Studio kann AWS-Services mithilfe von AWS SDK-Integrationen keine IAM Richtlinien für Task-Status generieren, die andere aufrufen.

Automatisches Generieren von Rollen

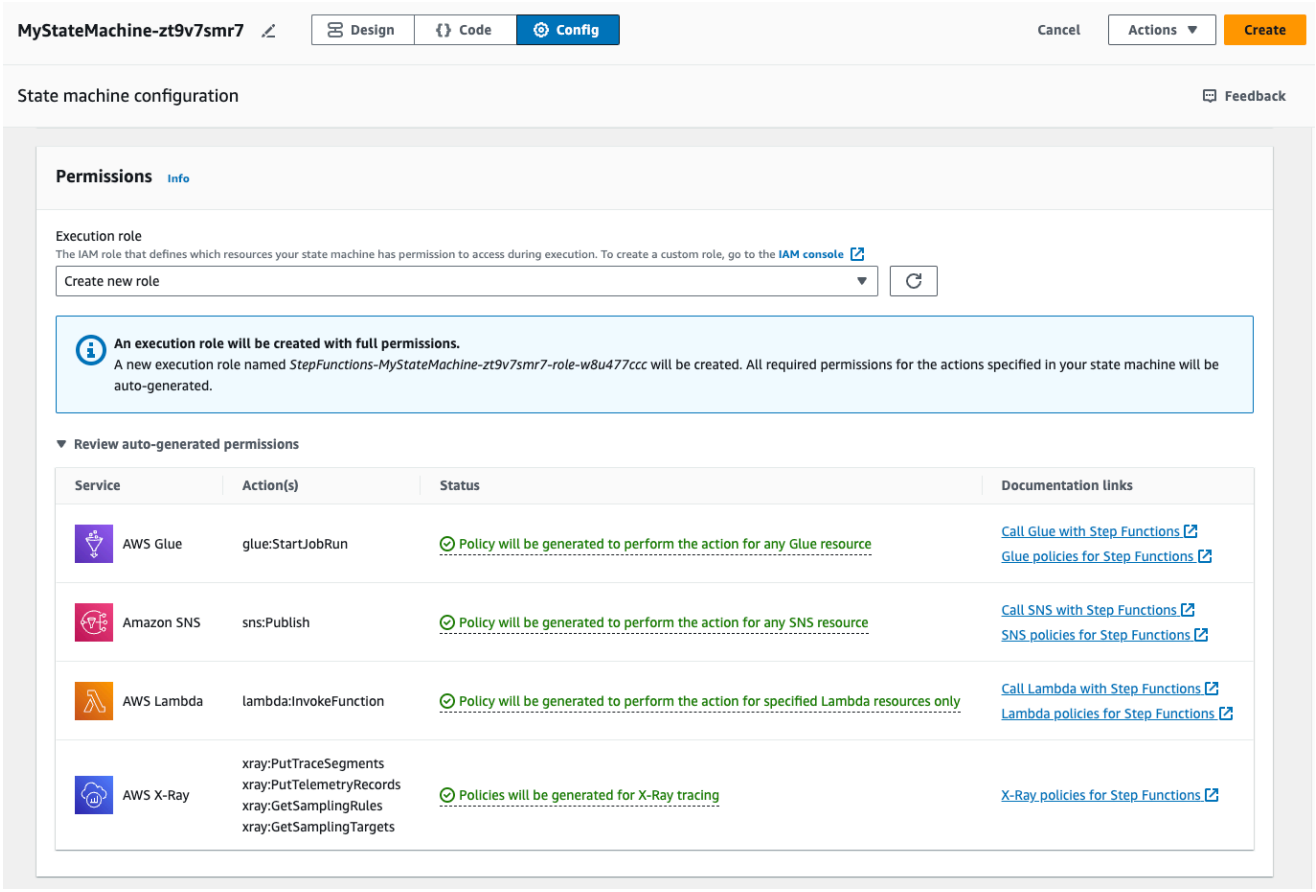
1. Öffnen Sie die Step Functions Functions-Konsole und wählen Sie Create State Machine.

   Sie können auch eine vorhandene Zustandsmaschine aktualisieren. Lesen Sie Schritt 4, wenn Sie eine Zustandsmaschine aktualisieren.

2. Wählen Sie im Dialogfeld Vorlage auswählen die Option Leer aus.
3. Wählen Sie Select (Auswählen). Dadurch wird Workflow Studio in geöffnetEntwurfsmodus.
4. Wählen Sie den Tab Config.
5. Scrollen Sie nach unten zum Abschnitt „Berechtigungen“ und gehen Sie wie folgt vor:

   a. Stellen Sie sicher, dass Sie für die Ausführungsrolle die Standardauswahl Neue Rolle erstellen beibehalten.

   Workflow Studio generiert automatisch alle erforderlichen IAM Richtlinien für jeden gültigen Status in Ihrer Zustandsmaschinen-Definition. Es zeigt ein Banner mit der Nachricht an: Eine Ausführungsrolle wird mit vollen Rechten erstellt.



> **Tip**
>
> Um die Berechtigungen zu überprüfen, die Workflow Studio automatisch für Ihren Zustandsmaschine generiert, wählen Sie Automatisch generierte Berechtigungen überprüfen aus.

> **Note**
>
> Wenn Sie die von Step Functions erstellte IAM-Rolle löschen, kann Step Functions sie später nicht mehr neu erstellen. Ebenso kann Step Functions ihre ursprünglichen Einstellungen später nicht wiederherstellen, wenn Sie die Rolle ändern (z. B. indem Sie Step Functions aus den Principals in der IAM-Richtlinie entfernen).

Wenn Workflow Studio nicht alle erforderlichen IAM Richtlinien generieren kann, wird ein Banner mit der Meldung Berechtigungen für bestimmte Aktionen können nicht automatisch generiert werden angezeigt. Eine IAM Rolle wird nur mit Teilberechtigungen erstellt. Informationen zum Hinzufügen der fehlenden Berechtigungen finden Sie unterProbleme bei der Rollengenerierung lösen.

b. Wählen Sie Erstellen, wenn Sie eine Zustandsmaschine erstellen. Wählen Sie andernfalls Speichern aus.

c. Wählen Sie im daraufhin angezeigten Dialogfeld die Option Bestätigen aus.

Workflow Studio speichert Ihren Zustandsmaschine und erstellt die neue Ausführungsrolle.

Probleme bei der Rollengenerierung lösen

In den folgenden Fällen kann Workflow Studio nicht automatisch eine Ausführungsrolle mit allen erforderlichen Berechtigungen generieren:

- In Ihrer Zustandsmaschine sind Fehler aufgetreten. Stellen Sie sicher, dass Sie alle Validierungsfehler in Workflow Studio behoben haben. Stellen Sie außerdem sicher, dass Sie alle serverseitigen Fehler beheben, die beim Speichern auftreten.
- Ihr Zustandsmaschine enthält Aufgaben, die AWS SDK-Integrationen verwenden. Workflow Studio kann in diesem Fall keine IAM Richtlinien automatisch generieren. Workflow Studio zeigt ein

Banner mit der Nachricht an. Berechtigungen für bestimmte Aktionen können nicht automatisch generiert werden. Eine IAM Rolle wird nur mit Teilberechtigungen erstellt. Wählen Sie in der Tabelle Automatisch generierte Berechtigungen überprüfen den Inhalt unter Status aus, um weitere Informationen zu den Richtlinien zu erhalten, die Ihrer Ausführungsrolle fehlen. Workflow Studio kann immer noch eine Ausführungsrolle generieren, aber diese Rolle enthält nicht IAM Richtlinien für alle Aktionen. Mithilfe der Links unter den Links zur Dokumentation können Sie Ihre eigenen Richtlinien schreiben und sie der Rolle hinzufügen, nachdem sie generiert wurde. Diese Links sind auch nach dem Speichern der Zustandsmaschine verfügbar.

Rolle zum Testen von HTTP-Aufgaben in Workflow Studio

Sie benötigen eine Ausführungsrolle, um einen HTTP-Task-Status zu testen. Wenn Sie keine Rolle mit ausreichenden Berechtigungen haben, verwenden Sie eine der folgenden Optionen, um eine Rolle zu erstellen:

- Automatische Generierung einer Rolle mit Workflow Studio (empfohlen) — Dies ist die sichere Option. Schließen Sie das Dialogfeld Teststatus und folgen Sie den Anweisungen unterAutomatisches Generieren von Rollen. Dazu müssen Sie zuerst Ihren Zustandsmaschine erstellen oder aktualisieren und dann zu Workflow Studio zurückkehren, um Ihren Status zu testen.
- Verwenden Sie eine Rolle mit Administratorzugriff — Wenn Sie berechtigt sind, eine Rolle mit Vollzugriff auf alle Dienste und Ressourcen zu erstellen AWS, können Sie diese Rolle verwenden, um jede Art von Status in Ihrem Workflow zu testen. Zu diesem Zweck können Sie in der IAM Konsole https://console.aws.amazon.com/iam/ eine Step Functions Servicerolle erstellen und ihr die AdministratorAccess Richtlinie hinzufügen.

Rolle zum Testen einer optimierten Serviceintegration in Workflow Studio

Sie benötigen eine Ausführungsrolle für Aufgabenstatus, die optimierte Serviceintegrationen aufrufen. Wenn Sie keine Rolle mit ausreichenden Berechtigungen haben, verwenden Sie eine der folgenden Optionen, um eine Rolle zu erstellen:

- Verwenden Sie die Dokumentationslinks in Workflow Studio, um Ihre eigenen IAM Richtlinien zu schreiben (empfohlen) — Dies ist die sichere Option. Schließen Sie das Dialogfeld Teststatus und folgen Sie den Anweisungen unterAutomatisches Generieren von Rollen. Dazu müssen Sie zuerst Ihren Zustandsmaschine erstellen oder aktualisieren und dann zu Workflow Studio zurückkehren, um Ihren Status zu testen.

- Verwenden Sie eine Rolle mit Administratorzugriff — Wenn Sie berechtigt sind, eine Rolle mit Vollzugriff auf alle Dienste und Ressourcen zu erstellen AWS, können Sie diese Rolle verwenden, um jede Art von Status in Ihrem Workflow zu testen. Zu diesem Zweck können Sie in der IAM Konsole https://console.aws.amazon.com/iam/ eine Step Functions Servicerolle erstellen und ihr die AdministratorAccess Richtlinie hinzufügen.

Rolle zum Testen einer AWS SDK-Dienstintegration in Workflow Studio

Sie benötigen eine Ausführungsrolle für Aufgabenstatus, die AWS SDK-Integrationen aufrufen. Wenn Sie keine Rolle mit ausreichenden Berechtigungen haben, verwenden Sie eine der folgenden Optionen, um eine Rolle zu erstellen:

- Verwenden Sie die Dokumentationslinks in Workflow Studio, um Ihre eigenen IAM Richtlinien zu schreiben (empfohlen) — Dies ist die sichere Option. Schließen Sie das Dialogfeld Teststatus und folgen Sie den Anweisungen unterAutomatisches Generieren von Rollen. Dazu müssen Sie zuerst Ihren Zustandsmaschine erstellen oder aktualisieren und dann zu Workflow Studio zurückkehren, um Ihren Status zu testen. Gehen Sie wie folgt vor:
  1. Schließen Sie das Dialogfeld Status testen
  2. Wählen Sie die Registerkarte Config, um den Konfigurationsmodus anzuzeigen.
  3. Scrollen Sie nach unten zum Abschnitt „Berechtigungen“.
  4. Workflow Studio zeigt ein Banner mit der Nachricht an. Berechtigungen für bestimmte Aktionen können nicht automatisch generiert werden. Eine IAM Rolle wird nur mit Teilberechtigungen erstellt. Wählen Sie Automatisch generierte Berechtigungen überprüfen aus.
  5. In der Tabelle „Automatisch generierte Berechtigungen überprüfen“ wird eine Zeile mit der Aktion angezeigt, die dem Aufgabenstatus entspricht, den Sie testen möchten. Unter den Links zur Dokumentation finden Sie Links, um Ihre eigenen IAM Richtlinien in eine benutzerdefinierte Rolle zu schreiben.
- Verwenden Sie eine Rolle mit Administratorzugriff — Wenn Sie berechtigt sind, eine Rolle mit uneingeschränktem Zugriff auf alle Dienste und Ressourcen zu erstellen AWS, können Sie diese Rolle verwenden, um jeden Status in Ihrem Workflow zu testen. Zu diesem Zweck können Sie in der IAM Konsole https://console.aws.amazon.com/iam/ eine Step Functions Servicerolle erstellen und ihr die AdministratorAccess Richtlinie hinzufügen.

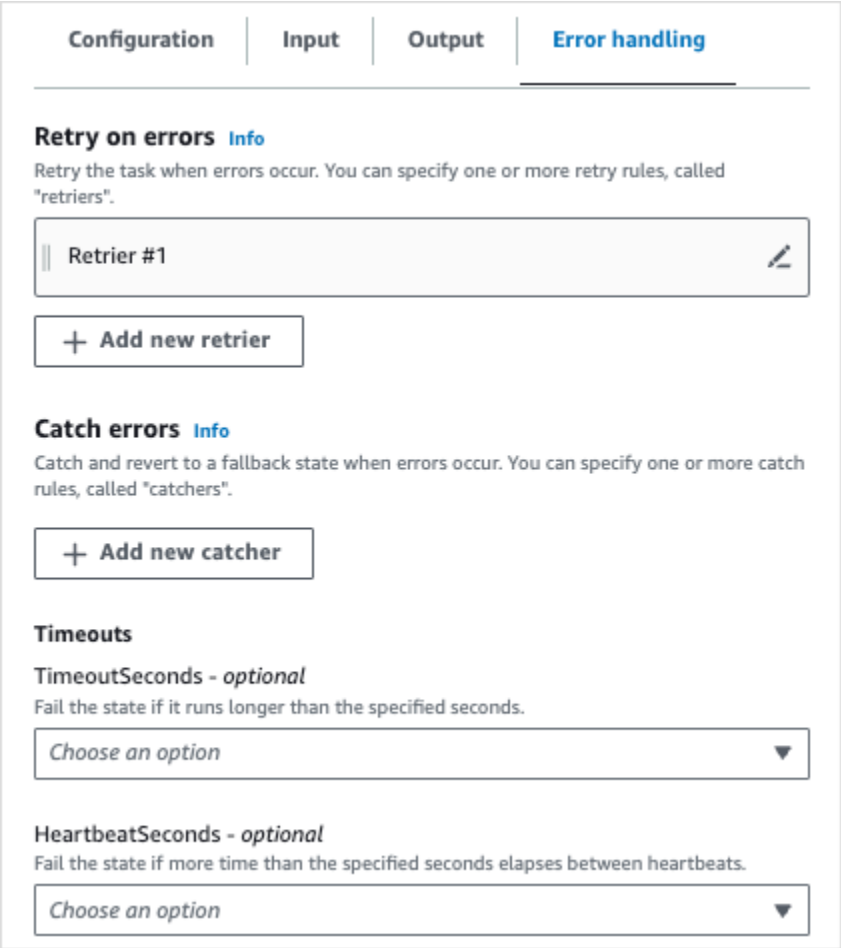
Rolle zum Testen von Ablaufzuständen in Workflow Studio

Sie benötigen eine Ausführungsrolle, um Flussstatus in Workflow Studio zu testen. Ablaufstatus sind die Zustände, die den Ausführungsablauf steuernChoice, wieParallel,Zuordnung,Pass,Wait,Succeed, oderFehler. Die TestStateAPI funktioniert nicht mit Zuordnungs- oder Parallel-Status. Verwenden Sie eine der folgenden Optionen, um eine Rolle zum Testen eines Flow-Status zu erstellen:

- Verwenden Sie eine beliebige Rolle in Ihrem AWS-Konto (empfohlen) — Flow-Status erfordern keine spezifischen IAM Richtlinien, da sie keine AWS Aktionen oder Ressourcen aufrufen. Daher kannst du jede IAM Rolle in deinem verwenden AWS-Konto.
  1. Wählen Sie im Dialogfeld Teststatus eine beliebige Rolle aus der Dropdownliste Ausführungsrolle aus.
  2. Wenn in der Dropdownliste keine Rollen angezeigt werden, gehen Sie wie folgt vor:
     a. Wählen Sie in der IAM Konsole https://console.aws.amazon.com/iam/ die Option Rollen aus.
     b. Wählen Sie eine Rolle aus der Liste aus und kopieren Sie ihren ARN von der Seite mit den Rollendetails. Sie müssen diesen ARN im Dialogfeld Teststatus angeben.
     c. Wählen Sie im Dialogfeld Teststatus in der Dropdownliste Ausführungsrolle die Option Einen Rollen-ARN eingeben aus.
     d. Fügen Sie den ARN in die Rolle ARN ein.
- Verwenden Sie eine Rolle mit Administratorzugriff — Wenn Sie berechtigt sind, eine Rolle mit Vollzugriff auf alle Dienste und Ressourcen zu erstellen AWS, können Sie diese Rolle verwenden, um jeden Status in Ihrem Workflow zu testen. Zu diesem Zweck können Sie in der IAM Konsole https://console.aws.amazon.com/iam/ eine Step Functions Servicerolle erstellen und ihr die AdministratorAccess Richtlinie hinzufügen.

Fehlerbehandlung

Wenn ein Status einen Fehler meldet, führt Step Functions standardmäßig dazu, dass die Workflow-Ausführung vollständig fehlschlägt. Für Aktionen und einige Ablaufstatus können Sie konfigurieren, wie Step Functions mit Fehlern umgeht. Auch wenn Sie die Fehlerbehandlung konfiguriert haben, können einige Fehler dennoch dazu führen, dass die Ausführung eines Workflows fehlschlägt. Weitere Informationen finden Sie unter Fehlerbehandlung in Step Functions. Konfigurieren Sie in Workflow Studio die Fehlerbehandlung auf der Registerkarte Fehlerbehandlung des Inspector Fensters.



Versuchen Sie es bei Fehlern erneut

Sie können dem Aktionsstatus und dem Parallel Flow-Status eine oder mehrere Regeln hinzufügen, um die Aufgabe erneut zu versuchen, wenn ein Fehler auftritt. Diese Regeln werden Retrier genannt. Um einen Retrier hinzuzufügen, wählen Sie das Bearbeitungssymbol im Feld Retrier #1 und konfigurieren Sie dann die entsprechenden Optionen:

- (Optional) Fügen Sie im Feld Kommentar Ihren Kommentar hinzu. Es hat keinen Einfluss auf den Arbeitsablauf, kann aber verwendet werden, um Ihren Arbeitsablauf mit Anmerkungen zu versehen.
- Platzieren Sie den Cursor in das Feld Fehler und wählen Sie einen Fehler aus, durch den der Abruf ausgelöst wird, oder geben Sie einen benutzerdefinierten Fehlernamen ein. Sie können mehrere Fehler auswählen oder hinzufügen.
- (Optional) Legen Sie ein Intervall fest. Dies ist die Zeit in Sekunden, bevor Step Functions seinen ersten Wiederholungsversuch durchführt. Zusätzliche Wiederholungen folgen in Intervallen, die Sie mit Max. Versuche und Backoff-Rate konfigurieren können.

- (Optional) Legen Sie „Max. Versuche“ fest. Dies ist die maximale Anzahl von Wiederholungsversuchen, bevor Step Functions dazu führt, dass die Ausführung fehlschlägt.
- (Optional) Stellen Sie die Backoff-Rate ein. Dies ist ein Multiplikator, der bestimmt, um wie viel sich das Wiederholungsintervall bei jedem Versuch verlängert.

> **Note**
>
> Nicht alle Optionen zur Fehlerbehandlung sind für alle Bundesstaaten verfügbar. Bei Lambda Invoke ist standardmäßig ein Retrier konfiguriert.

Fehler abfangen

Sie können den Aktionszuständen und den Zuordnung Ablaufstatus eine oder mehrere Regeln hinzufügen, um einen Fehler abzufangen. Parallel Diese Regeln werden Catcher genannt. Um einen Catcher hinzuzufügen, wählen Sie Neuen Catcher hinzufügen und konfigurieren Sie dann seine Optionen:

- (Optional) Fügen Sie im Feld Kommentar Ihren Kommentar hinzu. Es hat keinen Einfluss auf den Arbeitsablauf, kann aber verwendet werden, um Ihren Arbeitsablauf mit Anmerkungen zu versehen.
- Platzieren Sie den Cursor in das Feld Fehler und wählen Sie einen Fehler aus, durch den der Catcher ausgelöst wird, oder geben Sie einen benutzerdefinierten Fehlernamen ein. Sie können mehrere Fehler auswählen oder hinzufügen.
- Wählen Sie im Feld Fallback-Status einen Fallback-Status aus. Dies ist der Status, in den der Workflow als Nächstes wechselt, nachdem ein Fehler erkannt wurde.
- (Optional) Fügen Sie in dem ResultPathFeld einen `ResultPath` Filter hinzu, um den Fehler zur ursprünglichen Statuseingabe hinzuzufügen. Der `ResultPath`muss gültig sein JsonPath. Dies wird in den Fallback-Status gesendet.

Timeouts

Sie können ein Timeout für Aktionsstatus konfigurieren, um die maximale Anzahl von Sekunden festzulegen, für die Ihr Status ausgeführt werden kann, bevor er fehlschlägt. Verwenden Sie Timeouts, um zu verhindern, dass Ausführungen hängenbleiben. Um ein Timeout zu konfigurieren, geben Sie die Anzahl der Sekunden ein, die Ihr Bundesstaat warten soll, bevor die Ausführung

fehlschlägt. Weitere Informationen zu Timeouts finden Sie unter `TimeoutSeconds` im Status der Aufgabe Status.

HeartbeatSeconds

Sie können einen Heartbeat oder eine regelmäßige Benachrichtigung konfigurieren, die von Ihrer Aufgabe gesendet wird. Wenn Sie ein Taktintervall festlegen und Ihr Bundesstaat in den konfigurierten Intervallen keine Heartbeat-Benachrichtigungen sendet, wird die Aufgabe als fehlgeschlagen markiert. Um einen Heartbeat zu konfigurieren, legen Sie eine positive Ganzzahl ungleich Null für Sekunden fest. Weitere Informationen finden Sie unter `HeartBeatSeconds` in Status der Aufgabe state.

Tutorial: Lernen Sie, das AWS Step Functions Workflow Studio zu verwenden

In diesem Tutorial lernen Sie die Grundlagen der Arbeit mit Workflow Studio für kennen AWS Step Functions. In Entwurfsmodus Workflow Studio erstellen Sie eine Zustandsmaschine mit mehreren Status, darunter `Pass`,`Choice`,`Fail`,`Wait`, und`Parallel`. Sie verwenden die Drag-and-Drop-Funktion, um nach diesen Status zu suchen, sie auszuwählen und zu konfigurieren. Anschließend sehen Sie sich die automatisch generierte Amazon States Language (ASL) Definition Ihres Workflows an. Sie werden auch Workflow Studio verwenden, um die Codemodus Workflow-Definition zu bearbeiten. Anschließend beenden Sie Workflow Studio, führen die Zustandsmaschine aus und überprüfen die Ausführungsdetails.

In diesem Tutorial erfahren Sie auch, wie Sie die Zustandsmaschine aktualisieren und die Änderungen in der Ausführungsausgabe anzeigen. Schließlich führen Sie einen Bereinigungsschritt durch und löschen Ihren State Machine.

Nachdem Sie dieses Tutorial abgeschlossen haben, wissen Sie, wie Sie Workflow Studio verwenden, um einen Workflow sowohl im Entwurfs - als auch im Codemodus zu erstellen und zu konfigurieren. Sie werden auch wissen, wie Sie Ihren State Machine aktualisieren, ausführen und löschen.

> **Note**
>
> Bevor Sie beginnen, stellen Sie sicher, dass Sie die Voraussetzungen für dieses Tutorial erfüllt haben.

Themen

Schritt 1: Navigieren Sie zu Workflow Studio

1. Öffnen Sie die Step Functions Functions-Konsole und wählen Sie Create State Machine.
2. Wählen Sie im Dialogfeld Vorlage auswählen die Option Leer aus.
3. Wählen Sie Select (Auswählen). Dadurch wird Workflow Studio in geöffnetEntwurfsmodus.

Schritt 2: Erstellen Sie eine Zustandsmaschine

In Workflow Studio ist eine State Machine eine grafische Darstellung Ihres Workflows. Mit Workflow Studio können Sie die einzelnen Schritte Ihres Workflows definieren, konfigurieren und untersuchen. In den folgenden Schritten verwenden Sie den Entwurfsmodus von Workflow Studio, um Ihre Zustandsmaschine zu erstellen.

Erstellen eines -Zustandsautomaten

1. Stellen Sie sicher, dass Sie sich im Designmodus von Workflow Studio befinden.
2. Wählen Sie Bundesstaaten-Browser auf der linken Seite die Registerkarte Flow aus. Ziehen Sie dann einen Pass-Status in den leeren Status mit der Bezeichnung Drag First state here.
3. Ziehen Sie einen Choice-Status aus dem Flow-Tab und legen Sie ihn unter den Status Pass ab.
4. Ersetzen Sie für den Namen des Bundesstaates den Standardnamen Choice. Verwenden Sie in diesem Tutorial den Namen **`IsHelloWorldExample`**.
5. Ziehen Sie einen anderen Pass-Status und legen Sie ihn in einen Zweig des IsHelloWorldExampleBundesstaates ab. Ziehen Sie dann den Status „Fehler“ und legen Sie ihn unter den anderen Zweig des IsHelloWorldExampleZustands ab.

6. Wählen Sie den Status Pass (1) und benennen Sie ihn um in**`Yes`**. Benennen Sie den Status „Fehlgeschlagen“ um in**`No`**.
7. Geben Sie die Verzweigungslogik des IsHelloWorldExampleStatus mithilfe der booleschen Variablen an. `IsHelloWorldExample`

   Wenn ja **`IsHelloWorldExampleFalse`**, wechselt der Workflow in den Status Nein. Andernfalls setzt der Workflow seinen Ausführungsablauf im Status Ja fort.

   Um die Branch-Logik zu definieren, führen Sie die folgenden Schritte aus:

   a. Wählen Sie den IsHelloWorldExampleStatus aufLeinwand, und wählen Sie dann unter Auswahlregeln das Bearbeitungssymbol im Feld Regel #1 aus, um die Regel erster Wahl zu definieren.

   b. Wählen Sie Bedingungen hinzufügen.

   c. Geben Sie im Dialogfeld Bedingungen für Regel #1 **`$.IsHelloWorldExample`** unter Variable den Wert ein.

   d. Wählen Sie entspricht unter Operator.

   e. Wählen Sie unter Wert die Option Boolesche Konstante und dann in der Dropdownliste die Option True aus.

   f. Wählen Sie Bedingungen speichern aus.

   g. Vergewissern Sie sich, dass in der Dropdownliste Der nächste Status lautet: Ja ausgewählt ist.

   h. Wählen Sie Neue Auswahlregel hinzufügen und anschließend Bedingungen hinzufügen aus.

   i. Definieren Sie im Feld Regel #2 die Regel zweiter Wahl, wenn der Wert der `IsHelloWorldExample` Variablen falsch ist, indem Sie die Teilschritte 7.c bis 7.f wiederholen. Wählen Sie für Schritt 7.e Falsch statt Wahr aus.

   j. Wählen Sie im Feld Regel #2 in der Dropdownliste Dann ist der nächste Status: die Option Nein aus.

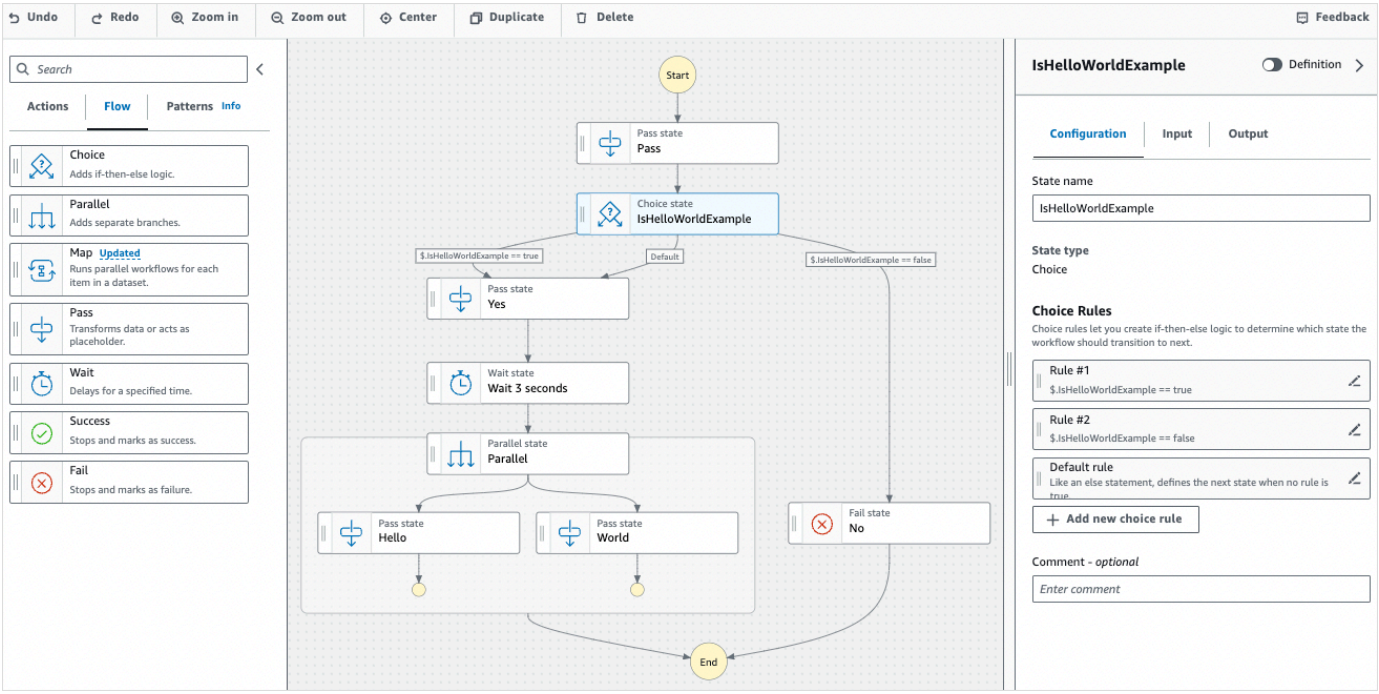
   k. Wählen Sie im Feld Standardregel das Bearbeitungssymbol aus, um die Standard-Auswahlregel zu definieren, und wählen Sie dann in der Dropdownliste Ja aus.
8. Fügen Sie nach dem Status Ja einen Wartestatus hinzu und geben Sie ihm **`Wait 3 sec`** einen Namen. Konfigurieren Sie dann die Wartezeit auf drei Sekunden, indem Sie die folgenden Schritte ausführen:

   a. Behalten Sie unter Optionen die Standardauswahl „Auf ein festes Intervall warten“ bei.

b. Vergewissern Sie sich, dass unter Sekunden die Option Sekunden eingeben ausgewählt ist, und geben Sie dann **3** in das Feld ein.

9. Fügen Sie nach dem Status 3 Sekunden warten den Status Parallel hinzu. Fügen Sie in den beiden Zweigen zwei Pass-Status hinzu. Nennen Sie den ersten Pass-Status**`Hello`**. Nennen Sie den zweiten Pass-Status**`World`**.

   Der abgeschlossene Workflow wird wie folgt aussehen:



Schritt 3: Überprüfen Sie die automatisch generierte Amazon States-Sprachdefinition

Wenn Sie Status von der Registerkarte Flow auf die Arbeitsfläche ziehen und dort ablegen, erstellt Workflow Studio automatisch die Amazon States Language (ASL) -Definition Ihres Workflows in Echtzeit. Wählen Sie im Inspector Bedienfeld die Umschaltfläche Definition, um diese Definition anzuzeigen, oder wechseln Sie zu, um diese Definition Codemodus nach Bedarf zu bearbeiten. Informationen zur Bearbeitung der Workflow-Definition finden Sie in Schritt 4 dieses Tutorials.

- (Optional) Wählen Sie im Inspektorfenster „Definition“ und sehen Sie sich den Arbeitsablauf der Zustandsmaschine an.

  Der folgende Beispielcode zeigt die automatisch generierte Amazon States-Sprachdefinition für den `IsHelloWorldExample` Zustandsmaschine. Der `Choice` Status, den Sie in Workflow

Studio hinzugefügt haben, wird verwendet, um den Ausführungsablauf auf der Grundlage der Verzweigungslogik zu bestimmen, die Sie in Schritt 2 definiert haben.

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

Schritt 4: Bearbeiten Sie die Workflow-Definition im Codemodus

Der Code-Modus von Workflow Studio bietet einen integrierten Code-Editor, mit dem Sie die ASL-Definition Ihrer Workflows anzeigen und bearbeiten können.

1. Wählen Sie Code, um in den Codemodus zu wechseln.
2. Platzieren Sie nach der Definition des Parallelstatus den Cursor und drücken Sie**`Enter`**.
3. Drücken Sie**`Ctrl+space`**, um die Liste der Status anzuzeigen, die Sie nach dem Status Parallel hinzufügen können.

4. Wählen Sie Pass State aus der Liste der Optionen aus. Der Code-Editor fügt Standardcode für den Pass-Status hinzu.
5. Das Hinzufügen dieses Status führt zu Fehlern in Ihrer Workflow-Definition. Ersetzen Sie in der Definition des Status Parallel `"End": true` durch**`"Next": "PassState"`**.
6. Nehmen Sie in der Definition Pass State, die Sie hinzugefügt haben, die folgenden Änderungen vor:

   a. Entfernen Sie den Ergebnisknoten.

   b. Entfernen Sie `"ResultPath": "$.result"`, und`"Next": "NextState"`.

   c. Danach `"Type": "Pass"`, geben Sie ein**`"End": true`**.

   d. Fügen Sie **`,`** nach der Pass-State-Definition eine hinzu.

Ihre Workflow-Definition sollte jetzt der folgenden Definition ähneln.

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

Schritt 5: Speichern Sie die Zustandsmaschine

1. Wählen Sie den Konfigurationsmodus oder klicken Sie auf das Bearbeitungssymbol neben dem Standardnamen der Zustandsmaschine von MyStateMachine. Geben Sie unter State-Machine-Konfiguration einen Namen an. Geben Sie z. B. ei **`HelloWorld`**.
2. (Optional) Geben Sie weitere Workflow-Einstellungen an, z. B. den Zustandsmaschinentyp und seine Ausführungsrolle. Behalten Sie für dieses Tutorial alle Standardauswahlen in der State-Machine-Konfiguration bei.
3. Wählen Sie Erstellen.
4. Wählen Sie im Dialogfeld „Rollenerstellung bestätigen“ die Option „Bestätigen“, um fortzufahren.

   Sie können auch Rollenkonfiguration anzeigen wählen, um zum Konfigurationsmodus zurückzukehren.

Weitere Informationen zum Konfigurationsmodus finden Sie unter Konfigurationsmodus von Workflow Studio.

Schritt 6: Führen Sie die Zustandsmaschine aus

State-Machine-Ausführungen sind Instanzen, in denen Sie Ihren Workflow ausführen, um Aufgaben auszuführen.

1. Wählen Sie auf der Seite State Machines den HelloWorldState Machine aus.
2. Wählen Sie auf der HelloWorldSeite die Option Ausführung starten aus.
3. (Optional) Um Ihre Ausführung zu identifizieren, können Sie im Feld Name einen Namen dafür angeben. Standardmäßig generiert Step Functions automatisch einen eindeutigen Ausführungsnamen.

   > **Note**
   >
   > Mit Step Functions können Sie Namen für Zustandsmaschinen, Ausführungen und Aktivitäten sowie Beschriftungen erstellen, die Nicht-ASCII-Zeichen enthalten. Diese Nicht-ASCII-Namen funktionieren nicht mit Amazon. CloudWatch Um sicherzustellen, dass Sie CloudWatch Messwerte verfolgen können, wählen Sie einen Namen, der nur ASCII-Zeichen verwendet.

4. Geben Sie im Eingabefeld Eingabewerte für Ihre Ausführung im JSON-Format ein. Basierend auf Ihrer Eingabe bestimmt die `IsHelloWorldExample` Variable, welcher State-Machine-Flow ausgeführt wird. Verwenden Sie vorerst den folgenden Eingabewert:

```
{
   "IsHelloWorldExample": true
}
```

> **Note**
>
> Die Angabe einer Ausführungseingabe ist zwar optional, in diesem Tutorial ist es jedoch erforderlich, eine Ausführungseingabe anzugeben, die der obigen Beispieleingabe ähnelt. Auf diesen Eingabewert wird im `Choice` Status verwiesen, wenn Sie die Zustandsmaschine ausführen.

5. Wählen Sie Start execution (Ausführung starten) aus.
6. Die Step Functions Functions-Konsole leitet Sie zu einer Seite weiter, die mit Ihrer Ausführungs-ID betitelt ist. Diese Seite wird als Seite mit den Ausführungsdetails bezeichnet. Auf dieser Seite können Sie die Ausführungsergebnisse im Verlauf oder nach Abschluss der Ausführung überprüfen.

   Um die Ausführungsergebnisse zu überprüfen, wählen Sie in der Diagrammansicht einzelne Status und dann die einzelnen Registerkarten im Schrittdetails Bereich aus, um die Details der einzelnen Status, einschließlich Eingabe, Ausgabe und Definition, anzuzeigen. Einzelheiten zu den Ausführungsinformationen, die Sie auf der Seite mit den Ausführungsdetails einsehen können, finden Sie unterSeite mit Ausführungsdetails – Schnittstellenübersicht.

   Wenn Sie für dieses Tutorial einen Eingabewert von eingegeben haben`"IsHelloWorldExample": true`, sollten Sie die folgende Ausgabe sehen:

```
{
   "IsHelloWorldExample": true
},
{
   "IsHelloWorldExample": true
}
```

Schritt 7: Aktualisieren Sie Ihren State Machine

Wenn Sie eine Zustandsmaschine aktualisieren, sind Ihre Updates letztendlich konsistent. Nach kurzer Zeit entsprechen alle neu gestarteten Ausführungen der aktualisierten Definition Ihres Zustandsmaschinen. Alle derzeit laufenden Ausführungen werden gemäß der vorherigen Definition bis zum Abschluss ausgeführt.

In diesem Schritt aktualisieren Sie Ihren Zustandsmaschine im Entwurfsmodus Modus von Workflow Studio. Sie fügen im Pass-Status ein `Result` Feld mit dem Namen World hinzu.

1. Wählen Sie auf der Seite mit Ihrer Ausführungs-ID die Option Zustandsmaschine bearbeiten aus.
2. Stellen Sie sicher, dass Sie sich im Designmodus befinden.
3. Wählen Sie auf der Arbeitsfläche den Pass-Status mit dem Namen World aus und wählen Sie dann Output aus.
4. Geben Sie in das Feld „Ergebnis“ ein **`"World has been updated!"`**.
5. Wählen Sie Speichern.
6. (Optional) Sehen Sie sich im Bereich Definition die aktualisierte Amazon States-Sprachdefinition Ihres Workflows an.

```
{
      "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. Wählen Sie Execute (Ausführen).
8. Geben Sie im Dialogfeld Ausführung starten, das auf einer neuen Registerkarte geöffnet wird, die folgenden Ausführungseingaben ein.

```
{
   "IsHelloWorldExample": true
}
```

9. Wählen Sie Start Execution aus.
10. (Optional) Wählen Sie in der Diagrammansicht den Schritt Welt und dann Ausgabe aus. Die Ausgabe lautet „Welt wurde aktualisiert“!

Schritt 8: Bereinigen

Um deinen State Machine zu löschen

1. Wählen Sie im Navigationsmenü State Machines aus.
2. Wählen Sie auf der Seite State Machines HelloWorlddie Option und anschließend Löschen aus.
3. Geben Sie im Dialogfeld „Zustandsmaschine löschen“ den Text ein, **`delete`** um den Löschvorgang zu bestätigen.
4. Wählen Sie Löschen aus.

   Wenn der Löschvorgang erfolgreich war, wird oben auf dem Bildschirm eine grüne Statusleiste angezeigt. Die grüne Statusleiste informiert Sie darüber, dass Ihr State Machine zum Löschen markiert ist. Ihr Zustandsmaschine wird gelöscht, wenn alle laufenden Ausführungen nicht mehr ausgeführt werden.

Um Ihre Ausführungsrolle zu löschen

1. Öffnen Sie die Rollenseite für IAM.
2. Wählen Sie die IAM-Rolle aus, die Step Functions für Sie erstellt hat. Beispiel: StepFunctions-HelloWorld -Role-example.

3. Wählen Sie Delete role (Rolle löschen) aus.
4. Wählen Sie Yes, delete (Ja, löschen) aus.

Tutorials für Step Functions

Die Tutorials in diesem Abschnitt helfen Ihnen dabei, verschiedene Aspekte der Arbeit mit AWS Step Functions zu verstehen.

Um diese Tutorials abzuschließen, benötigen Sie ein - AWS Konto. Wenn Sie kein - AWS Konto haben, navigieren Sie zu https://aws.amazon.com/ und wählen Sie AWS Konto erstellen aus.

Themen

Eine Step Functions Functions-Zustandsmaschine erstellen, die Lambda verwendet

In diesem Tutorial erstellen Sie einen einstufigen Workflow, mit dem AWS Step Functions Sie eine AWS Lambda Funktion aufrufen.

> **Note**
>
> Step Functions basiert auf Zustandsmaschinen und Aufgaben. In Step Functions werden Zustandsmaschinen als Workflows bezeichnet. Dabei handelt es sich um eine Reihe von ereignisgesteuerten Schritten. Jeder Schritt in einem Workflow wird als Status bezeichnet. Ein Aufgabenstatus steht beispielsweise für eine Arbeitseinheit, die ein anderer AWS Dienst ausführt, z. B. das Aufrufen eines anderen Dienstes AWS-Service oder einer API.
> Weitere Informationen finden Sie hier:
>
> - Was ist AWS Step Functions?
> - Rufen Sie andere Dienste AWS an

Lambda eignet sich gut für `Task` Staaten, da Lambda-Funktionen serverlos und einfach zu schreiben sind. Sie können Code im AWS Management Console oder in Ihrem Lieblingseditor schreiben. AWS kümmert sich um die Einzelheiten der Bereitstellung einer Computerumgebung für Ihre Funktion und deren Ausführung.

In diesem Thema:

Schritt 1: Erstellen einer Lambda-Funktion

Ihre Lambda-Funktion empfängt Ereignisdaten und gibt eine Begrüßungsnachricht zurück.

> **Important**
>
> Stellen Sie sicher, dass sich Ihre Lambda-Funktion unter demselben AWS Konto und derselben AWS Region wie Ihr State Machine befindet.

1. Öffnen Sie die Lambda-Konsole und wählen Sie Create function.
2. Wählen Sie auf der Seite Create function die Option Author from scratch.

3. Geben Sie für Function name (Funktionsname) `HelloFunction` ein.
4. Behalten Sie die Standardauswahl für alle anderen Optionen bei und wählen Sie dann Funktion erstellen.
5. Nachdem Ihre Lambda-Funktion erstellt wurde, kopieren Sie den Amazon-Ressourcennamen (ARN) der Funktion, der in der oberen rechten Ecke der Seite angezeigt wird. Um den ARN zu kopieren, klicken Sie auf
   Im Folgenden finden Sie ein Beispiel für einen ARN:

   ```
   arn:aws:lambda:us-east-1:123456789012:function:HelloFunction
   ```

6. Kopieren Sie den folgenden Code für die Lambda-Funktion in den Abschnitt Codequelle der ***HelloFunction***Seite.

   ```
   export const handler = async(event, context, callback) => {
       callback(null, "Hello from " + event.who + "!");
   };
   ```

   Dieser Code erstellt eine Grußformel aus dem Feld `who` der Eingabedaten, die durch das in Ihre Funktion übergebene `event`-Objekt bereitgestellt wird. Sie fügen später Eingabedaten für diese Funktion hinzu, wenn Sie eine neue Ausführung starten. Die `callback`-Methode gibt die erstellte Grußformel aus Ihrer Funktion zurück.
7. Wählen Sie Bereitstellen.

Schritt 2: Testen Sie die Lambda-Funktion

Testen Sie Ihre Lambda-Funktion, um zu sehen, wie sie in Betrieb ist.

1. Wählen Sie Test aus.
2. Geben Sie für Event name (Ereignisname) `HelloEvent` ein.
3. Ersetzen Sie die Event-JSON-Daten durch die folgenden.

   ```
   {
       "who": "AWS Step Functions"
   }
   ```

Der `"who"` Eintrag entspricht dem `event.who` Feld in Ihrer Lambda-Funktion und vervollständigt die Begrüßung. Sie geben dieselben Eingabedaten ein, wenn Sie Ihre Zustandsmaschine ausführen.

4. Wählen Sie Speichern und dann Test.
5. Erweitern Sie unter Execution result (Ausführungsergebnis) die Option Details, um die Testergebnisse anzuzeigen.

Schritt 3: Erstellen Sie eine Zustandsmaschine

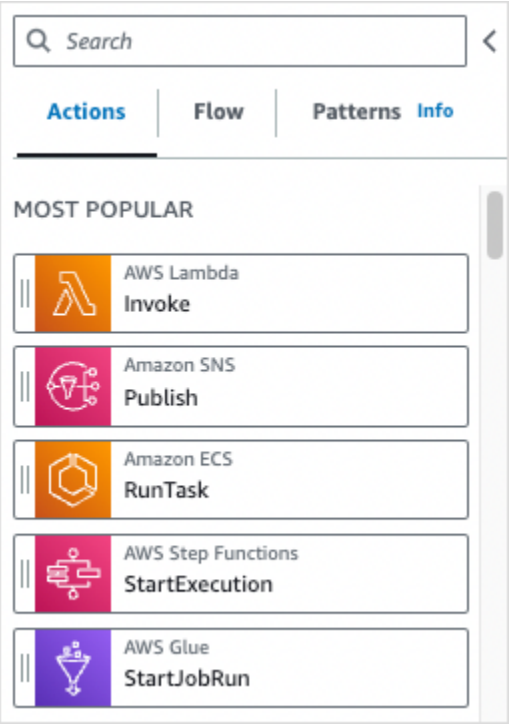
Verwenden Sie die Step Functions Functions-Konsole, um eine Zustandsmaschine zu erstellen, die die Lambda-Funktion aufruft, die Sie in Schritt 1 erstellt haben.

1. Öffnen Sie die Step Functions Functions-Konsole und wählen Sie Create State Machine.

> **Important**
>
> Stellen Sie sicher, dass sich Ihr State Machine unter demselben AWS Konto und derselben Region befindet wie die zuvor erstellte Lambda-Funktion.

2. Wählen Sie im Dialogfeld Vorlage auswählen die Option Leer aus.
3. Wählen Sie Select (Auswählen). Dadurch wird Workflow Studio in geöffnetEntwurfsmodus.
4. Vergewissern Sie sich, dass Sie im States-Browser auf der linken Seite die Registerkarte Aktionen ausgewählt haben. Führen Sie dann die folgenden Schritte aus:

   - Ziehen Sie die AWS Lambda Invoke API per Drag & Drop in den leeren Status mit der Bezeichnung Drag first state here.



5. Konfigurieren Sie im Inspektor-Panel auf der rechten Seite die Lambda-Funktion:

   a. Wählen Sie im Abschnitt API-Parameter die Lambda-Funktion aus, die Sie zuvor in der Dropdownliste Funktionsname erstellt haben.

   b. Behalten Sie die Standardauswahl in der Payload-Dropdown-Liste bei.

6. (Optional) Wählen Sie Definition, um die Definition der Zustandsmaschine Amazon States Language (ASL) anzuzeigen, die automatisch auf der Grundlage Ihrer Auswahl auf der Registerkarte Aktionen und im Inspektorfenster generiert wird.

7. Geben Sie einen Namen für Ihre Zustandsmaschine ein. Wählen Sie dazu das Bearbeitungssymbol neben dem Standardnamen der Zustandsmaschine von MyStateMachine. Geben Sie dann unter State-Machine-Konfiguration einen Namen in das Feld State-Machine-Name ein.

   Geben Sie beispielsweise den Namen **`LambdaStateMachine`** ein.

   > **Note**
   >
   > Die Namen von Zustandsmaschinen, Ausführungen und Aktivitätsaufgaben dürfen nicht länger als 80 Zeichen sein. Diese Namen müssen für Ihr Konto und Ihre AWS Region eindeutig sein und dürfen keine der folgenden Angaben enthalten:

- Leerraum
- Platzhalterzeichen () ? *
- Klammerzeichen () < > { } [ ]
- Sonderzeichen (" # % \ ^ | ~ ` $ & , ; : /)
- Steuerzeichen (\\u0000- \\u001f oder \\u007f -\\u009f).

Wenn Ihre Zustandsmaschine vom Typ Express ist, können Sie denselben Namen für mehrere Ausführungen der Zustandsmaschine angeben. Step Functions generiert für jede Ausführung von Express State Machine einen eindeutigen Ausführungs-ARN, auch wenn mehrere Ausführungen denselben Namen haben.
Mit Step Functions können Sie Namen für Zustandsmaschinen, Ausführungen und Aktivitäten sowie Beschriftungen erstellen, die Nicht-ASCII-Zeichen enthalten. Diese Nicht-ASCII-Namen funktionieren nicht mit Amazon. CloudWatch Um sicherzustellen, dass Sie CloudWatch Messwerte verfolgen können, wählen Sie einen Namen, der nur ASCII-Zeichen verwendet.

8. (Optional) Geben Sie unter State-Machine-Konfiguration weitere Workflow-Einstellungen an, z. B. den Zustandsmaschinentyp und seine Ausführungsrolle.

   Behalten Sie für dieses Tutorial alle Standardauswahlen in den State-Machine-Einstellungen bei.

9. Wählen Sie Erstellen.
10. Wählen Sie im Dialogfeld „Rollenerstellung bestätigen“ die Option „Bestätigen“, um fortzufahren.

    Sie können auch Rolleneinstellungen anzeigen wählen, um zur State-Machine-Konfiguration zurückzukehren.

> **Note**
>
> Wenn Sie die von Step Functions erstellte IAM-Rolle löschen, kann Step Functions sie später nicht mehr neu erstellen. Ebenso kann Step Functions ihre ursprünglichen Einstellungen später nicht wiederherstellen, wenn Sie die Rolle ändern (z. B. indem Sie Step Functions aus den Principals in der IAM-Richtlinie entfernen).

Schritt 4: Führen Sie die Zustandsmaschine aus

Nachdem Sie Ihren Zustandsmaschine erstellt haben, können Sie ihn ausführen.

1. Wählen Sie auf der Seite State Machines die Option LambdaStateMachine.
2. Wählen Sie Start execution (Ausführung starten) aus.

   Das Dialogfeld „Ausführung starten“ wird angezeigt.
3. (Optional) Um Ihre Ausführung zu identifizieren, können Sie im Feld Name einen Namen dafür angeben. Standardmäßig generiert Step Functions automatisch einen eindeutigen Ausführungsnamen.

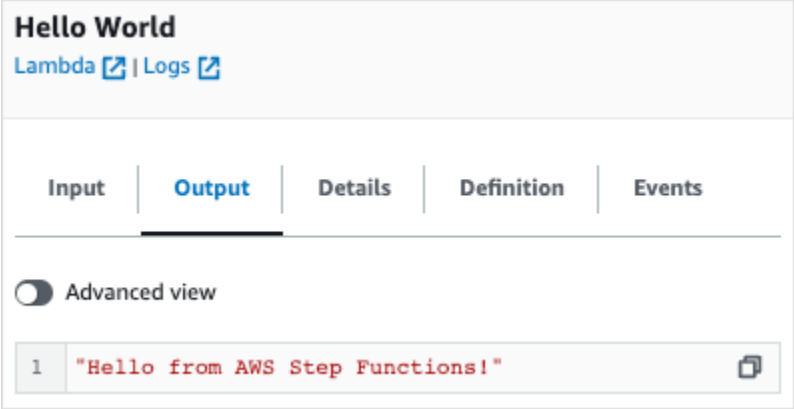
   > **Note**
   >
   > Mit Step Functions können Sie Namen für Zustandsmaschinen, Ausführungen und Aktivitäten sowie Beschriftungen erstellen, die Nicht-ASCII-Zeichen enthalten. Diese Nicht-ASCII-Namen funktionieren nicht mit Amazon. CloudWatch Um sicherzustellen, dass Sie CloudWatch Messwerte verfolgen können, wählen Sie einen Namen, der nur ASCII-Zeichen verwendet.
4. Ersetzen Sie im Eingabebereich die Ausführungsdaten des Beispiels durch die folgenden.

   ```
   {
       "who" : "AWS Step Functions"
   }
   ```

   `"who"`ist der Schlüsselname, den Ihre Lambda-Funktion verwendet, um den Namen der zu begrüßenden Person abzurufen.
5. Wählen Sie Start Execution aus.

   Die Ausführung Ihrer Zustandsmaschine wird gestartet, und eine neue Seite mit Ihrer laufenden Ausführung wird angezeigt.
6. Die Step Functions Functions-Konsole leitet Sie zu einer Seite weiter, die mit Ihrer Ausführungs-ID betitelt ist. Diese Seite wird als Seite mit den Ausführungsdetails bezeichnet. Auf dieser Seite können Sie die Ausführungsergebnisse im Verlauf der Ausführung oder nach deren Abschluss überprüfen.

Um die Ausführungsergebnisse zu überprüfen, wählen Sie in der Diagrammansicht einzelne Status aus und wählen Sie dann die einzelnen Registerkarten im Schrittdetails Bereich, um die Details der einzelnen Status, einschließlich Eingabe, Ausgabe und Definition, anzuzeigen. Einzelheiten zu den Ausführungsinformationen, die Sie auf der Seite mit den Ausführungsdetails einsehen können, finden Sie unterSeite mit Ausführungsdetails – Schnittstellenübersicht.



> **Note**
>
> Sie können auch Nutzlasten übergeben, während Sie Lambda von einer Zustandsmaschine aufrufen. Weitere Informationen und Beispiele zum Aufrufen von Lambda durch Übergabe von Nutzdaten in das `Parameters` Feld finden Sie unter. Lambda mit Step-Funktionen aufrufen

Behandlung von Fehlerbedingungen mithilfe einer Step Functions Functions-Zustandsmaschine

In diesem Tutorial erstellen Sie eine AWS Step Functions Zustandsmaschine mit einem Fallback-Staaten Feld. Das `Catch` Feld verwendet eine AWS Lambda Funktion, um mit bedingter Logik zu antworten, die auf dem Typ der Fehlermeldung basiert. Diese Technik wird als Funktionsfehlerbehandlung bezeichnet.

Weitere Informationen finden Sie unter AWS Lambda Funktionsfehler in Node.js im AWS Lambda Entwicklerhandbuch.

> **Note**
>
> Sie können auch Zustandsmaschinen erstellen, die es bei Timeouts erneut versuchen, oder solche, die bei `Catch` Auftreten eines Fehlers oder einer Zeitüberschreitung in einen bestimmten Zustand übergehen. Beispiele für diese Fehlerbehandlungstechniken finden Sie unter Beispiele für die Verwendung von Retry und von Catch.

In diesem Thema:

Schritt 1: Erstellen Sie eine Lambda-Funktion, die fehlschlägt

Verwenden Sie eine Lambda-Funktion, um eine Fehlerbedingung zu simulieren.

> **Important**
>
> Stellen Sie sicher, dass sich Ihre Lambda-Funktion unter demselben AWS Konto und derselben AWS Region wie Ihr State Machine befindet.

1. Öffnen Sie die AWS Lambda Konsole unter https://console.aws.amazon.com/lambda/.
2. Wählen Sie Funktion erstellen.
3. Wählen Sie Blueprint verwenden aus, geben Sie den **`step-functions`** Text in das Suchfeld ein und wählen Sie dann den Blueprint Einen benutzerdefinierten Fehler auslösen aus.
4. Geben Sie für Function name (Funktionsname) `FailFunction` ein.
5. Behalten Sie für Rolle die Standardauswahl bei (Neue Rolle mit grundlegenden Lambda-Berechtigungen erstellen).
6. Der folgende Code wird im Codebereich der Lambda-Funktion angezeigt.

```
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!');
};
```

Das `context`-Objekt gibt die Fehlermeldung `This is a custom error!` zurück.

7. Wählen Sie Funktion erstellen.
8. Nachdem Ihre Lambda-Funktion erstellt wurde, kopieren Sie den Amazon-Ressourcennamen (ARN) der Funktion, der in der oberen rechten Ecke der Seite angezeigt wird. Um den ARN zu kopieren, klicken Sie auf
   Im Folgenden finden Sie ein Beispiel für einen ARN:

   ```
   arn:aws:lambda:us-east-1:123456789012:function:FailFunction
   ```

9. Wählen Sie Bereitstellen.

Schritt 2: Testen Sie die Lambda-Funktion

Testen Sie Ihre Lambda-Funktion, um zu sehen, wie sie in Betrieb ist.

1. Wählen Sie auf der FailFunctionSeite die Registerkarte Test und dann Test aus. Sie müssen kein Testereignis erstellen.
2. Um die Testergebnisse (den simulierten Fehler) zu überprüfen, erweitern Sie unter Ausführungsergebnis die Option Details.

Schritt 3: Erstellen Sie eine Zustandsmaschine mit einem Catch-Feld

Verwenden Sie die Step Functions Functions-Konsole, um eine Zustandsmaschine zu erstellen, die einen Status der Aufgabe Status mit einem `Catch` Feld verwendet. Fügen Sie im Task-Status einen Verweis auf Ihre Lambda-Funktion hinzu. Die Zustandsmaschine ruft die Lambda-Funktion auf, die während der Ausführung fehlschlägt. Step Functions wiederholt die Funktion zweimal, wobei ein exponentieller Backoff zwischen den Wiederholungen verwendet wird.

1. Öffnen Sie die Step Functions Functions-Konsole und wählen Sie Create State Machine.

2. Wählen Sie im Dialogfeld Vorlage auswählen die Option Leer aus.
3. Wählen Sie Select (Auswählen). Dadurch wird Workflow Studio in geöffnetEntwurfsmodus.
4. Wählen Sie Code, um den Code-Editor zu öffnen. Im Code-Editor schreiben und bearbeiten Sie die Amazon States Language (ASL-) Definition Ihrer Workflows.
5. Fügen Sie den folgenden Code ein, ersetzen Sie jedoch den ARN der Lambda-Funktion, die Sie zuvor in dem `Resource` Feld erstellt haben.

```
{
   "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
      }
```

```
    }
}
```

Dies ist eine Beschreibung Ihrer Zustandsmaschine in der Sprache Amazon States. Sie definiert einen einzelnen `Task`-Zustand namens `CreateAccount`. Weitere Informationen finden Sie unter State Machine Structure.

Weitere Informationen zur Syntax des `Retry`-Feldes finden Sie unter Beispiele für Zustandsmaschinen mit Retry und Catch.

> **Note**
>
> Unbehandelte Fehler in Lambda werden wie `Lambda.Unknown` in der Fehlerausgabe gemeldet. Dazu gehören out-of-memory Fehler und Funktions-Timeouts. Sie können nach, oder abgleichen `Lambda.UnknownStates.ALL`, `States.TaskFailed` um diese Fehler zu behandeln. Wenn Lambda die maximale Anzahl von Aufrufen erreicht, lautet der Fehler. `Lambda.TooManyRequestsException` Weitere Informationen zu Lambda-Funktionsfehlern finden Sie unter Fehlerbehandlung und automatische Wiederholungen im AWS Lambda Entwicklerhandbuch.

6. (Optional) Sehen Sie sich im die Bereich zur Grafikvisualisierung grafische Echtzeitvisualisierung Ihres Workflows an.
7. Geben Sie einen Namen für Ihre Zustandsmaschine an. Wählen Sie dazu das Bearbeitungssymbol neben dem Standardnamen der Zustandsmaschine von MyStateMachine. Geben Sie dann unter State-Machine-Konfiguration einen Namen in das Feld State-Machine-Name ein.

   Geben Sie für dieses Tutorial **`Catchfailure`** ein.
8. (Optional) Geben Sie unter Zustandsmaschinen-Konfiguration weitere Workflow-Einstellungen an, z. B. den Zustandsmaschinentyp und seine Ausführungsrolle.

   Behalten Sie für dieses Tutorial alle Standardauswahlen in den State-Machine-Einstellungen bei.
9. Wählen Sie im Dialogfeld „Rollenerstellung bestätigen“ die Option „Bestätigen“, um fortzufahren.

   Sie können auch Rolleneinstellungen anzeigen wählen, um zur State-Machine-Konfiguration zurückzukehren.

> **Note**
>
> Wenn Sie die von Step Functions erstellte IAM-Rolle löschen, kann Step Functions sie später nicht mehr neu erstellen. Ebenso kann Step Functions ihre ursprünglichen Einstellungen später nicht wiederherstellen, wenn Sie die Rolle ändern (z. B. indem Sie Step Functions aus den Principals in der IAM-Richtlinie entfernen).

Schritt 4: Führen Sie die Zustandsmaschine aus

Nachdem Sie Ihren Zustandsmaschine erstellt haben, können Sie ihn ausführen.

1. Wählen Sie auf der Seite State Machines die Option Catchfailure aus.
2. Wählen Sie auf der Seite Catchfailure die Option Ausführung starten aus. Das Dialogfeld „Ausführung starten“ wird angezeigt.
3. Gehen Sie im Dialogfeld Ausführung starten wie folgt vor:

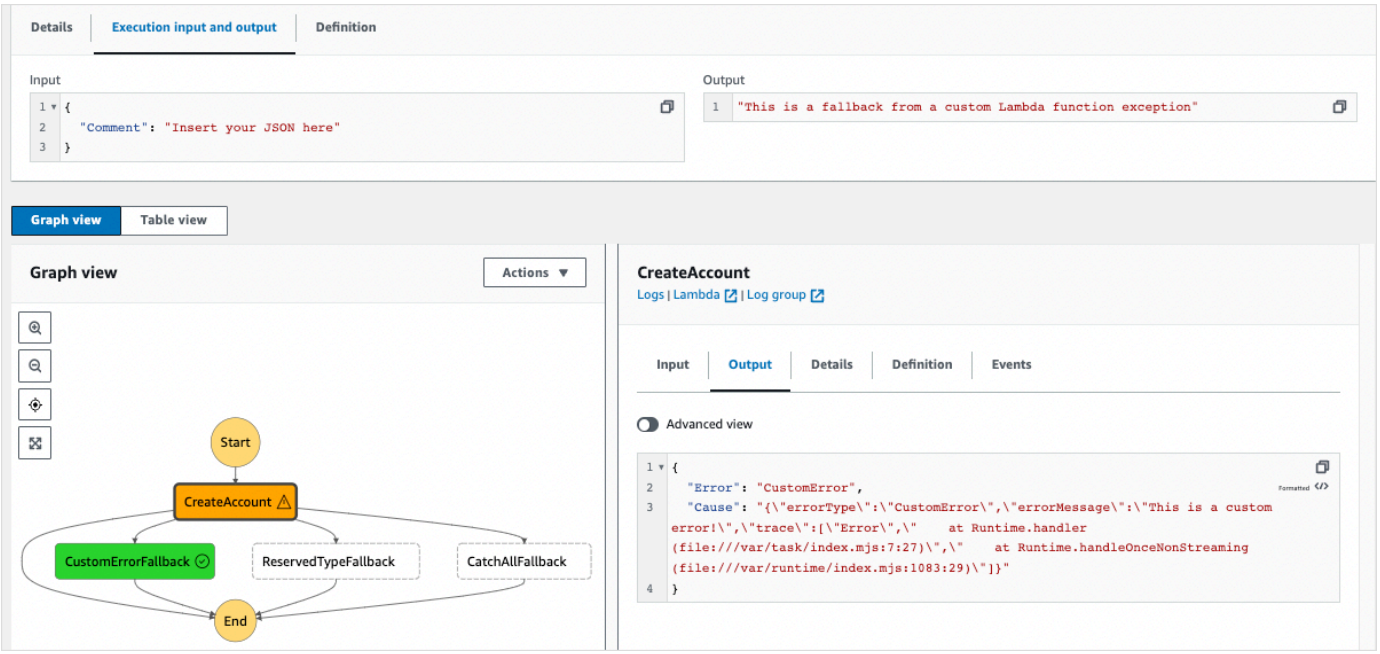
    1. (Optional) Um Ihre Ausführung zu identifizieren, können Sie im Feld Name einen Namen dafür angeben. Standardmäßig generiert Step Functions automatisch einen eindeutigen Ausführungsnamen.

        > **Note**
        >
        > Mit Step Functions können Sie Namen für Zustandsmaschinen, Ausführungen und Aktivitäten sowie Beschriftungen erstellen, die Nicht-ASCII-Zeichen enthalten. Diese Nicht-ASCII-Namen funktionieren nicht mit Amazon. CloudWatch Um sicherzustellen, dass Sie CloudWatch Messwerte verfolgen können, wählen Sie einen Namen, der nur ASCII-Zeichen verwendet.

    2. (Optional) Geben Sie in das Eingabefeld Eingabewerte im JSON-Format ein, um Ihren Workflow auszuführen.
    3. Wählen Sie Start execution (Ausführung starten) aus.
    4. Die Step Functions Functions-Konsole leitet Sie zu einer Seite weiter, die mit Ihrer Ausführungs-ID betitelt ist. Diese Seite wird als Seite mit den Ausführungsdetails bezeichnet. Auf dieser Seite können Sie die Ausführungsergebnisse im Verlauf der Ausführung oder nach deren Abschluss überprüfen.

Um die Ausführungsergebnisse zu überprüfen, wählen Sie in der Diagrammansicht einzelne Status aus und wählen Sie dann die einzelnen Registerkarten im Schrittdetails Bereich, um die Details der einzelnen Status, einschließlich Eingabe, Ausgabe und Definition, anzuzeigen. Einzelheiten zu den Ausführungsinformationen, die Sie auf der Seite mit den Ausführungsdetails einsehen können, finden Sie unterSeite mit Ausführungsdetails – Schnittstellenübersicht.

Um beispielsweise Ihre benutzerdefinierte Fehlermeldung anzuzeigen, wählen Sie den CreateAccountSchritt in der Diagrammansicht und dann die Registerkarte Ausgabe aus.



> **Note**
>
> Sie können die Statuseingabe mit dem Fehler beibehalten, indem Sie `ResultPath` verwenden. Siehe Verwenden Sie ResultPath , um sowohl Fehler als auch Eingabe in einen einzuschließen `Catch`.

Verwenden des Inline-Map-Status zur Wiederholung einer Aktion

Dieses Tutorial hilft Ihnen bei den ersten Schritten mit der Verwendung des `Map` Status im Inline-Modus. Sie verwenden den Inline-Map-Status in Ihren Workflows, um wiederholt eine Aktion

auszuführen. Weitere Informationen zum Inline-Modus finden Sie unter Zuordnen des Status im Inline-Modus.

In diesem Tutorial verwenden Sie den Inline-Map-Status, um wiederholt Universally Unique Identifiers (v4-UUID) der Version 4 zu generieren. Sie beginnen mit der Erstellung eines Workflows, der zwei Pass Status und einen Inline Map-Status im Workflow Studio enthält. Anschließend konfigurieren Sie die Eingabe und Ausgabe, einschließlich des Eingabe-JSON-Arrays für den `Map` Status. Der `Map` Status gibt ein Ausgabearray zurück, das die v4-UUIDs enthält, die für jedes Element im Eingabearray generiert wurden.

Inhalt

Schritt 1: Erstellen Sie den Workflow-Prototyp

In diesem Schritt erstellen Sie den Prototyp für Ihren Workflow mit Workflow Studio. Workflow Studio ist ein visueller Workflow-Designer, der in der Step Functions-Konsole verfügbar ist. Sie wählen die erforderlichen Status auf der Registerkarte Flow aus und verwenden die Drag-and-Drop-Funktion von Workflow Studio, um den Workflow-Prototyp zu erstellen.

1. Öffnen Sie die Step Functions Functions-Konsole und wählen Sie Create State Machine.
2. Wählen Sie im Dialogfeld Vorlage auswählen die Option Leer aus.
3. Wählen Sie Select (Auswählen). Dadurch wird Workflow Studio in geöffnetEntwurfsmodus.
4. Ziehen Sie auf der Registerkarte Flow einen Pass-Status und legen Sie ihn in den leeren Status mit der Bezeichnung Drag First state here ab.
5. Ziehen Sie einen Kartenstatus und legen Sie ihn unter den Status Pass ab. Benennen Sie den Kartenstatus um in**`Map demo`**.
6. Ziehen Sie einen zweiten Pass-Status und legen Sie ihn innerhalb des Map-Demo-Status ab.
7. Benennen Sie den zweiten Pass-Status in um**`Generate UUID`**.

Schritt 2: Eingabe und Ausgabe konfigurieren

In diesem Schritt konfigurieren Sie Eingabe und Ausgabe für alle Status in Ihrem Workflow-Prototyp. Zunächst fügen Sie mithilfe des ersten Pass-Status einige feste Daten in den Workflow ein. Dieser Status „Pass“ gibt diese Daten als Eingabe an den Map-Demo-Status weiter. In dieser Eingabe geben Sie den Knoten an, der das Eingabearray enthält, über das der Map-Demostatus iteriert werden soll. Dann definieren Sie den Schritt, den der Map-Demostatus wiederholen soll, um die v4-UUIDs zu generieren. Schließlich konfigurieren Sie die Ausgabe so, dass sie bei jeder Wiederholung zurückgegeben wird.

1. Wählen Sie den ersten Pass-Status in Ihrem Workflow-Prototyp aus. Geben Sie auf der Registerkarte Ausgabe unter Ergebnis Folgendes ein:

   ```
   {
     "foo": "bar",
     "colors": [
       "red",
       "green",
       "blue",
       "yellow",
       "white"
     ]
   }
   ```

2. Wählen Sie den Map-Demostatus und gehen Sie auf der Registerkarte Konfiguration wie folgt vor:

   a. Wählen Sie „Pfad zum Artikelarray angeben“ aus.

   b. Geben Sie den folgenden Referenzpfad an, um den Knoten auszuwählen, der das Eingabe-Array enthält:

      ```
      $.colors
      ```

3. Wählen Sie den Status „UUID generieren“ und gehen Sie auf der Registerkarte Eingabe wie folgt vor:

   a. Wählen Sie „Eingabe mit Parametern transformieren“.

   b. Geben Sie die folgende JSON-Eingabe ein, um die v4-UUIDs für jedes der Eingabe-Array-Elemente zu generieren. Sie verwenden die `States.UUID` systeminterne Funktion, um die UUIDs zu generieren.

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

4. Wählen Sie für den Status „UUID generieren“ die Registerkarte „Ausgabe“ und gehen Sie wie folgt vor:

   a. Wählen Sie „Ausgabe filtern mit“. OutputPath

   b. Geben Sie den folgenden Referenzpfad ein, um den JSON-Knoten auszuwählen, der die Ausgabe-Array-Elemente enthält:

```
$.uuid
```

Schritt 3: Überprüfen Sie die automatisch generierte Amazon States-Sprachdefinition und speichern Sie den Workflow

Wenn Sie Status aus dem Flow-Bedienfeld auf die Arbeitsfläche ziehen und dort ablegen, erstellt Workflow Studio automatisch die Amazon States Language (ASL) -Definition Ihres Workflows in Echtzeit. Sie können diese Definition nach Bedarf bearbeiten.

1. (Optional) Wählen Sie im Inspector Fenster Definition aus, um die automatisch generierte Amazon States-Sprachdefinition Ihres Workflows anzuzeigen.

   > **Tip**
   >
   > Sie können die ASL-Definition auch im Workflow Studio einsehen. Code-Editor Im Code-Editor können Sie auch die ASL-Definition Ihres Workflows bearbeiten.

   Das folgende Beispiel zeigt die automatisch generierte Amazon States-Sprachdefinition für Ihren Workflow.

```
{
    "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. Geben Sie einen Namen für Ihre Zustandsmaschine an. Wählen Sie dazu das Bearbeitungssymbol neben dem Standardnamen der Zustandsmaschine von MyStateMachine. Geben Sie dann unter State-Machine-Konfiguration einen Namen in das Feld State-Machine-Name ein.

   Geben Sie für dieses Tutorial den Namen **`InlineMapDemo`** ein.

3. (Optional) Geben Sie unter State-Machine-Konfiguration weitere Workflow-Einstellungen an, z. B. den Zustandsmaschinentyp und seine Ausführungsrolle.

   Behalten Sie für dieses Tutorial alle Standardauswahlen in der State-Machine-Konfiguration bei.

4. Wählen Sie im Dialogfeld „Rollenerstellung bestätigen“ die Option „Bestätigen“, um fortzufahren.

   Sie können auch Rolleneinstellungen anzeigen wählen, um zur State-Machine-Konfiguration zurückzukehren.

   > **Note**
   >
   > Wenn Sie die von Step Functions erstellte IAM-Rolle löschen, kann Step Functions sie später nicht mehr neu erstellen. Ebenso kann Step Functions ihre ursprünglichen Einstellungen später nicht wiederherstellen, wenn Sie die Rolle ändern (z. B. indem Sie Step Functions aus den Principals in der IAM-Richtlinie entfernen).

Schritt 4: Führen Sie die Zustandsmaschine aus

State-Machine-Ausführungen sind Instanzen, in denen Sie Ihren Workflow ausführen, um Aufgaben auszuführen.

1. Wählen Sie auf der InlineMapDemoSeite Ausführung starten aus.
2. Gehen Sie im Dialogfeld Ausführung starten wie folgt vor:

   1. (Optional) Um Ihre Ausführung zu identifizieren, können Sie im Feld Name einen Namen dafür angeben. Standardmäßig generiert Step Functions automatisch einen eindeutigen Ausführungsnamen.

      > **Note**
      >
      > Mit Step Functions können Sie Namen für Zustandsmaschinen, Ausführungen, Aktivitäten und Labels erstellen, die Nicht-ASCII-Zeichen enthalten. Diese Nicht-ASCII-Namen funktionieren nicht mit Amazon. CloudWatch Um sicherzustellen, dass Sie CloudWatch Messwerte verfolgen können, wählen Sie einen Namen, der nur ASCII-Zeichen verwendet.

2. (Optional) Geben Sie in das Eingabefeld Eingabewerte im JSON-Format ein, um Ihren Workflow auszuführen.
3. Wählen Sie Start execution (Ausführung starten) aus.
4. Die Step Functions Functions-Konsole leitet Sie zu einer Seite weiter, die mit Ihrer Ausführungs-ID betitelt ist. Diese Seite wird als Seite mit den Ausführungsdetails bezeichnet. Auf dieser Seite können Sie die Ausführungsergebnisse im Verlauf der Ausführung oder nach deren Abschluss überprüfen.

   Um die Ausführungsergebnisse zu überprüfen, wählen Sie in der Diagrammansicht einzelne Status aus und wählen Sie dann die einzelnen Registerkarten im Schrittdetails Bereich, um die Details der einzelnen Status, einschließlich Eingabe, Ausgabe und Definition, anzuzeigen. Einzelheiten zu den Ausführungsinformationen, die Sie auf der Seite mit den Ausführungsdetails einsehen können, finden Sie unterSeite mit Ausführungsdetails – Schnittstellenübersicht.

Um die Eingabe und Ausgabe der Ausführung anzuzeigen side-by-side, wählen Sie „Eingabe und Ausgabe der Ausführung“. Sehen Sie sich unter Ausgabe das vom `Map` Status zurückgegebene Ausgabe-Array an. Im Folgenden finden Sie ein Beispiel für das Ausgabe-Array:

```
[
  "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"
]
```

Kopieren umfangreicher CSV-Daten mit Distributed Map

Dieses Tutorial hilft Ihnen beim Einstieg in die Verwendung des `Map` Status im verteilten Modus. Ein `Map` Status, der auf Verteilt gesetzt ist, wird als Distributed-Map-Status bezeichnet. Sie verwenden den Status Distributed Map in Ihren Workflows, um über umfangreiche Amazon S3 S3-Datenquellen zu iterieren. Der `Map` Status führt jede Iteration als untergeordnete Workflow-Ausführung aus, wodurch eine hohe Parallelität ermöglicht wird. Weitere Informationen zum verteilten Modus finden Sie unter Status im verteilten Modus zuordnen.

In diesem Tutorial verwenden Sie den Status Distributed Map, um über eine CSV-Datei in einem Amazon S3 S3-Bucket zu iterieren. Anschließend geben Sie seinen Inhalt zusammen mit dem ARN einer untergeordneten Workflow-Ausführung in einem anderen Amazon S3 S3-Bucket zurück. Sie beginnen mit der Erstellung eines Workflow-Prototyps im Workflow Studio. Als Nächstes setzen Sie den Verarbeitungsmodus des `Map` Status auf Verteilt, geben die CSV-Datei als Datensatz an und geben dem `Map` Status ihren Speicherort an. Sie geben auch den Workflowtyp für die untergeordneten Workflow-Ausführungen an, bei denen der Status Distributed Map als Express gestartet wird.

Zusätzlich zu diesen Einstellungen geben Sie auch andere Konfigurationen an, z. B. die maximale Anzahl gleichzeitiger untergeordneter Workflow-Ausführungen und den Speicherort, an dem das `Map` Ergebnis exportiert werden soll, für den in diesem Tutorial verwendeten Beispiel-Workflow.

Inhalt

Voraussetzungen

- Laden Sie eine CSV-Datei in einen Amazon S3 S3-Bucket hoch. Sie müssen eine Kopfzeile in Ihrer CSV-Datei definieren. Informationen zu den Größenbeschränkungen für die CSV-Datei und zur Angabe der Kopfzeile finden Sie unterCSV-Datei in einem Amazon S3 S3-Bucket.
- Erstellen Sie einen weiteren Amazon S3 S3-Bucket und einen Ordner innerhalb dieses Buckets, in den das `Map` Statusergebnis exportiert werden soll.

> **Important**
>
> Stellen Sie sicher, dass sich Ihre Amazon S3 S3-Buckets unter demselben AWS-Konto und AWS-Region Ihrem Zustandsmaschine befinden.

Schritt 1: Erstellen Sie den Workflow-Prototyp

In diesem Schritt erstellen Sie den Prototyp für Ihren Workflow mit Workflow Studio. Workflow Studio ist ein visueller Workflow-Designer, der in der Step Functions-Konsole verfügbar ist. Sie wählen den erforderlichen Status und die API-Aktion auf den Registerkarten Flow bzw. Aktionen aus. Sie verwenden die Drag-and-Drop-Funktion von Workflow Studio, um den Workflow-Prototyp zu erstellen.

1. Öffnen Sie die Step Functions Functions-Konsole und wählen Sie Create State Machine.
2. Wählen Sie im Dialogfeld Vorlage auswählen die Option Leer aus.
3. Wählen Sie Select (Auswählen). Dadurch wird Workflow Studio in geöffnetEntwurfsmodus.
4. Ziehen Sie auf der Registerkarte Flow einen Map-Status und legen Sie ihn in den leeren Status mit der Bezeichnung Drag First state here ab.
5. Geben **`Process data`** Sie auf der Registerkarte Konfiguration als Statusname den Wert ein.
6. Ziehen Sie auf der Registerkarte Aktionen eine Aktion „API AWS Lambda aufrufen“ und legen Sie sie im Status Prozessdaten ab.
7. Benennen Sie den Status „AWS Lambda Aufrufen“ in um. **`Process CSV data`**

Schritt 2: Konfigurieren Sie die erforderlichen Felder für den Kartenstatus

In diesem Schritt konfigurieren Sie die folgenden Pflichtfelder für den Status Distributed Map:

- ItemReader— Gibt den Datensatz und seinen Speicherort an, aus dem der `Map` Bundesstaat Eingaben lesen kann.
- ItemProcessor— Gibt die folgenden Werte an:
  - `ProcessorConfig`— Legt die `Mode` Werte `DISTRIBUTED` und `ExecutionType` auf `EXPRESS` bzw. fest. Dadurch werden der Verarbeitungsmodus des `Map` Status und der Workflowtyp für untergeordnete Workflow-Ausführungen festgelegt, die mit dem Status Distributed Map gestartet werden.
  - `StartAt`— Der erste Status im Map-Workflow.

- `States`— Definiert den Map-Workflow, bei dem es sich um eine Reihe von Schritten handelt, die bei der Ausführung jedes untergeordneten Workflows wiederholt werden müssen.
- ResultWriter— Gibt den Amazon S3 S3-Speicherort an, an den Step Functions die Distributed Map-Zustandsergebnisse schreibt.

> **Important**
>
> Stellen Sie sicher, dass sich der Amazon S3 S3-Bucket, den Sie zum Exportieren der Ergebnisse eines Map Runs verwenden, unter demselben AWS-Konto und AWS-Region wie Ihr Zustandsmaschine befindet. Andernfalls schlägt die Ausführung Ihrer Zustandsmaschine mit dem `States.ResultWriterFailed` Fehler fehl.

Gehen Sie wie folgt vor, um die erforderlichen Felder zu konfigurieren:

1. Wählen Sie den Status Prozessdaten und gehen Sie auf der Registerkarte Konfiguration wie folgt vor:
   a. Wählen Sie für den Verarbeitungsmodus die Option Verteilt aus.
   b. Wählen Sie als Artikelquelle Amazon S3 und wählen Sie dann CSV-Datei in S3 aus der Dropdownliste S3-Artikelquelle aus.
   c. Gehen Sie wie folgt vor, um den Amazon S3 S3-Speicherort Ihrer CSV-Datei anzugeben:
      i. Wählen Sie für das S3-Objekt die Option Bucket und Schlüssel eingeben aus der Drop-down-Liste aus.
      ii. Geben Sie für Bucket den Namen des Amazon S3 S3-Buckets ein, der die CSV-Datei enthält. z. B. **`sourceBucket`**.
      iii. Geben Sie für Key den Namen des Amazon S3 S3-Objekts ein, in dem Sie die CSV-Datei gespeichert haben. Sie müssen in diesem Feld auch den Namen der CSV-Datei angeben. z. B. **`csvDataset/ratings.csv`**.
   d. Bei CSV-Dateien müssen Sie auch den Speicherort der Spaltenüberschrift angeben. Wählen Sie dazu Zusätzliche Konfiguration und behalten Sie dann für Speicherort der CSV-Überschrift die Standardauswahl Erste Zeile bei, wenn die erste Zeile Ihrer CSV-Datei die Kopfzeile ist. Wählen Sie andernfalls Given aus, um den Header innerhalb der State-Machine-Definition anzugeben. Weitere Informationen finden Sie unter `ReaderConfig`.
   e. Wählen Sie als Ausführungstyp Kind die Option Express aus.

2. Um die Map Run-Ergebnisse an einen bestimmten Amazon S3-Standort zu exportieren, wählen Sie unter Standort exportieren die Option Ausgabe des Kartenstatus nach Amazon S3 exportieren.
3. Gehen Sie wie folgt vor:

   a. Wählen Sie für den S3-Bucket die Option Bucket-Namen und -Präfix eingeben aus der Drop-down-Liste aus.
   b. Geben Sie für Bucket den Namen des Amazon S3 S3-Buckets ein, in den Sie die Ergebnisse exportieren möchten. z. B. **`mapOutputs`**.
   c. Geben Sie unter Präfix den Namen des Ordners ein, in dem Sie die Ergebnisse speichern möchten. z. B. **`resultData`**.

Schritt 3: Konfigurieren Sie zusätzliche Optionen

Zusätzlich zu den erforderlichen Einstellungen für einen Distributed-Map-Status können Sie auch andere Optionen angeben. Dazu können die maximale Anzahl gleichzeitiger Ausführungen untergeordneter Workflows und der Speicherort gehören, an den das `Map` Statusergebnis exportiert werden soll.

1. Wählen Sie den Status Prozessdaten aus. Wählen Sie dann unter Artikelquelle die Option Zusätzliche Konfiguration aus.
2. Gehen Sie wie folgt vor:

   a. Wählen Sie Elemente ändern mit aus ItemSelector, um eine benutzerdefinierte JSON-Eingabe für jede untergeordnete Workflow-Ausführung anzugeben.
   b. Geben Sie die folgende JSON-Eingabe ein:

   ```
   {
     "index.$": "$$.Map.Item.Index",
     "value.$": "$$.Map.Item.Value"
   }
   ```

   Informationen zum Erstellen einer benutzerdefinierten Eingabe finden Sie unter`ItemSelector`.
3. Geben Sie in den Laufzeiteinstellungen für Parallelitätslimit die Anzahl gleichzeitiger untergeordneter Workflow-Ausführungen an, die der Status Distributed Map starten kann. Geben Sie z. B. ei **`100`**.

4. Öffnen Sie ein neues Fenster oder eine neue Registerkarte in Ihrem Browser und schließen Sie die Konfiguration der Lambda-Funktion ab, die Sie in diesem Workflow verwenden werden, wie unter erklärt. Schritt 4: Lambda-Funktion konfigurieren

Schritt 4: Lambda-Funktion konfigurieren

> **Important**
>
> Stellen Sie sicher, dass sich Ihre Lambda-Funktion unter derselben befindet AWS-Region wie Ihre Zustandsmaschine.

1. Öffnen Sie die Lambda-Konsole und wählen Sie Create function.
2. Wählen Sie auf der Seite Create function die Option Author from scratch.
3. Konfigurieren Sie im Abschnitt Grundinformationen Ihre Lambda-Funktion:

   a. Geben Sie für Function name (Funktionsname) **`distributedMapLambda`** ein.

   b. Wählen Sie unter Laufzeit die Option Node.js 16.x aus.

   c. Behalten Sie alle Standardauswahlen bei und wählen Sie Funktion erstellen.

   d. Nachdem Sie Ihre Lambda-Funktion erstellt haben, kopieren Sie den Amazon-Ressourcennamen (ARN) der Funktion, der in der oberen rechten Ecke der Seite angezeigt wird. Sie müssen dies in Ihrem Workflow-Prototyp angeben. Um den ARN zu kopieren, klicken Sie auf
   Im Folgenden finden Sie ein Beispiel für einen ARN:

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

4. Kopieren Sie den folgenden Code für die Lambda-Funktion und fügen Sie ihn in den Abschnitt Codequelle der distributedMapLambdaSeite ein.

```
exports.handler = async function(event, context) {
    console.log("Received Input:\n", event);

    return {
        'statusCode' : 200,
        'inputReceived' : event //returns the input that it received
```

```
    }
};
```

5. Wählen Sie Bereitstellen. Sobald Ihre Funktion bereitgestellt ist, wählen Sie Test, um die Ausgabe Ihrer Lambda-Funktion zu sehen.

Schritt 5: Aktualisieren Sie den Workflow-Prototyp

In der Step Functions Functions-Konsole aktualisieren Sie Ihren Workflow, um den ARN der Lambda-Funktion hinzuzufügen.

1. Kehren Sie zu der Registerkarte oder dem Fenster zurück, in dem Sie den Workflow-Prototyp erstellt haben.
2. Wählen Sie den Schritt CSV-Daten verarbeiten aus und gehen Sie auf der Registerkarte Konfiguration wie folgt vor:

   a. Wählen Sie als Integrationstyp die Option Optimiert aus.

   b. Beginnen Sie mit der Eingabe des Namens Ihrer Lambda-Funktion als Funktionsname. Wählen Sie die Funktion aus der angezeigten Dropdownliste aus, oder wählen Sie Funktionsnamen eingeben und geben Sie den ARN der Lambda-Funktion ein.

Schritt 6: Überprüfen Sie die automatisch generierte Amazon States-Sprachdefinition und speichern Sie den Workflow

Wenn Sie Status aus den Registerkarten Aktion und Ablauf per Drag-and-Drop auf die Arbeitsfläche ziehen, erstellt Workflow Studio automatisch die Amazon States-Sprachdefinition Ihres Workflows in Echtzeit. Sie können diese Definition nach Bedarf bearbeiten.

1. (Optional) Wählen Sie im Inspector Fenster „Definition“ aus und sehen Sie sich die State-Machine-Definition an.

> **Tip**
>
> Sie können die ASL-Definition auch im Code-Editor Workflow Studio anzeigen. Im Code-Editor können Sie auch die ASL-Definition Ihres Workflows bearbeiten.

Der folgende Beispielcode zeigt die automatisch generierte Amazon States-Sprachdefinition für Ihren Workflow.

```
{
  "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. Geben Sie einen Namen für Ihre Zustandsmaschine an. Wählen Sie dazu das Bearbeitungssymbol neben dem Standardnamen der Zustandsmaschine von MyStateMachine. Geben Sie dann unter State-Machine-Konfiguration einen Namen in das Feld State-Machine-Name ein.

   Geben Sie für dieses Tutorial den Namen **`DistributedMapDemo`** ein.

3. (Optional) Geben Sie unter State-Machine-Konfiguration weitere Workflow-Einstellungen an, z. B. den Zustandsmaschinentyp und seine Ausführungsrolle.

   Behalten Sie für dieses Tutorial alle Standardauswahlen in der State-Machine-Konfiguration bei.

4. Wählen Sie im Dialogfeld „Rollenerstellung bestätigen“ die Option „Bestätigen“, um fortzufahren.

   Sie können auch Rolleneinstellungen anzeigen wählen, um zur State-Machine-Konfiguration zurückzukehren.

> **Note**
>
> Wenn Sie die von Step Functions erstellte IAM-Rolle löschen, kann Step Functions sie später nicht mehr neu erstellen. Ebenso kann Step Functions ihre ursprünglichen Einstellungen später nicht wiederherstellen, wenn Sie die Rolle ändern (z. B. indem Sie Step Functions aus den Principals in der IAM-Richtlinie entfernen).

Schritt 7: Führen Sie die Zustandsmaschine aus

Eine Ausführung ist eine Instanz Ihrer Zustandsmaschine, in der Sie Ihren Workflow ausführen, um Aufgaben auszuführen.

1. Wählen Sie auf der DistributedMapDemoSeite Ausführung starten aus.
2. Gehen Sie im Dialogfeld Ausführung starten wie folgt vor:

   1. (Optional) Um Ihre Ausführung zu identifizieren, können Sie im Feld Name einen Namen dafür angeben. Standardmäßig generiert Step Functions automatisch einen eindeutigen Ausführungsnamen.

      > **Note**
      >
      > Mit Step Functions können Sie Namen für Zustandsmaschinen, Ausführungen, Aktivitäten und Labels erstellen, die Nicht-ASCII-Zeichen enthalten. Diese Nicht-ASCII-Namen funktionieren nicht mit Amazon. CloudWatch Um sicherzustellen, dass Sie CloudWatch Messwerte verfolgen können, wählen Sie einen Namen, der nur ASCII-Zeichen verwendet.

   2. (Optional) Geben Sie in das Eingabefeld Eingabewerte im JSON-Format ein, um Ihren Workflow auszuführen.
   3. Wählen Sie Start execution (Ausführung starten) aus.
   4. Die Step Functions Functions-Konsole leitet Sie zu einer Seite weiter, die mit Ihrer Ausführungs-ID betitelt ist. Diese Seite wird als Seite mit den Ausführungsdetails bezeichnet. Auf dieser Seite können Sie die Ausführungsergebnisse im Verlauf der Ausführung oder nach deren Abschluss überprüfen.

      Um die Ausführungsergebnisse zu überprüfen, wählen Sie in der Diagrammansicht einzelne Status aus und wählen Sie dann die einzelnen Registerkarten im Schrittdetails Bereich, um die Details der einzelnen Status, einschließlich Eingabe, Ausgabe und Definition, anzuzeigen. Einzelheiten zu den Ausführungsinformationen, die Sie auf der Seite mit den Ausführungsdetails einsehen können, finden Sie unterSeite mit Ausführungsdetails – Schnittstellenübersicht.

   Wählen Sie beispielsweise den `Map` Bundesstaat und anschließend „Ausführung zuordnen“, um die Seite „Kartenlaufdetails“ zu öffnen. Auf dieser Seite können Sie alle Ausführungsdetails des

Status „Distributed Map“ und die damit gestarteten untergeordneten Workflow-Ausführungen anzeigen. Informationen zu dieser Seite finden Sie unterUntersuchen der Kartenausführung.

Verarbeitung des gesamten Datenstapels mit einer Lambda-Funktion

In diesem Tutorial verwenden Sie das ItemBatcher Feld des Status „Verteilte Zuordnung“ von , um einen gesamten Stapel von Elementen innerhalb einer Lambda-Funktion zu verarbeiten. Jeder Batch enthält maximal drei Elemente. Der Status Distributed Map startet vier untergeordnete Workflow-Ausführungen, wobei jede Ausführung drei Elemente verarbeitet, während eine Ausführung ein einzelnes Element verarbeitet. Jede untergeordnete Workflow-Ausführung ruft eine Lambda-Funktion auf, die über die einzelnen im Batch vorhandenen Elemente iteriert.

Sie erstellen einen Zustandsautomaten, der eine Multiplikation für ein Array von Ganzzahlen durchführt. Angenommen, das Ganzzahl-Array, das Sie als Eingabe angeben, ist `[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]` und der Multiplikationsfaktor ist `7`. Dann ist das resultierende Array, das nach dem Multiplizieren dieser Ganzzahlen mit einem Faktor von 7 gebildet wird, `[7, 14, 21, 28, 35, 42, 49, 56, 63, 70]`.

Themen

- Schritt 1: Erstellen des Zustandsautomaten
- Schritt 2: Erstellen der Lambda-Funktion
- Schritt 3: Ausführen des Zustandsautomaten

Schritt 1: Erstellen des Zustandsautomaten

In diesem Schritt erstellen Sie den Workflow-Prototyp des Zustandsautomaten, der einen gesamten Datenstapel an die Lambda-Funktion übergibt, die Sie in Schritt 2 erstellen.

- Verwenden Sie die folgende Definition, um einen Zustandsautomaten mit der Step-Functions-Konsole zu erstellen. Informationen zum Erstellen eines Zustandsautomaten finden Sie Schritt 1: Erstellen Sie den Workflow-Prototyp unter im Tutorial Erste Schritte mit der Verwendung von Distributed Map.

  In diesem Zustandsautomaten definieren Sie einen Distributed Map-Zustand, der ein Array von 10 Ganzzahlen als Eingabe akzeptiert und dieses Array in Batches von an eine Lambda-

Funktion übergibt`3`. Die Lambda-Funktion iteriert über die einzelnen im Batch vorhandenen Elemente und gibt ein Ausgabe-Array mit dem Namen zurück`multiplied`. Das Ausgabe-Array enthält das Ergebnis der Multiplikation, die für die im Eingabe-Array übergebenen Elemente durchgeführt wurde.

> **Important**
>
> Stellen Sie sicher, dass Sie den Amazon-Ressourcennamen (ARN) der Lambda-Funktion im folgenden Code durch den ARN der Funktion ersetzen, die Sie in Schritt 2 erstellen.

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

Schritt 2: Erstellen der Lambda-Funktion

In diesem Schritt erstellen Sie die Lambda-Funktion, die alle im Batch übergebenen Elemente verarbeitet.

> **Important**
>
> Stellen Sie sicher, dass sich Ihre Lambda-Funktion unter derselben AWS-Region wie Ihr Zustandsautomat befindet.

So erstellen Sie die Lambda-Funktion:

1. Verwenden Sie die Lambda-Konsole, um eine Python-3.9-Lambda-Funktion mit dem Namen zu erstellen**`ProcessEntireBatch`**. Informationen zum Erstellen einer Lambda-Funktion finden Sie unter Schritt 4: Konfigurieren der Lambda-Funktion im Tutorial Erste Schritte mit der Verwendung von Distributed Map State.
2. Kopieren Sie den folgenden Code für die Lambda-Funktion und fügen Sie ihn in den Abschnitt Codequelle Ihrer Lambda-Funktion ein.

```
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. Nachdem Sie Ihre Lambda-Funktion erstellt haben, kopieren Sie den ARN der Funktion, der in der oberen rechten Ecke der Seite angezeigt wird. Um den ARN zu kopieren, klicken Sie auf  .

   Im Folgenden finden Sie ein Beispiel für einen ARN, wobei der Name der Lambda-Funktion *`function-name`* ist (in diesem Fall `ProcessEntireBatch`):

```
arn:aws:lambda:us-east-1:123456789012:function:function-name
```

   Sie müssen den Funktions-ARN in dem Zustandsautomaten angeben, den Sie in Schritt 1 erstellt haben.
4. Wählen Sie Bereitstellen, um die Änderungen bereitzustellen.

Schritt 3: Ausführen des Zustandsautomaten

Wenn Sie den Zustandsautomaten ausführen, startet der Status Verteilte Zuordnung vier untergeordnete Workflow-Ausführungen, wobei jede Ausführung drei Elemente verarbeitet, während eine Ausführung ein einzelnes Element verarbeitet.

Das folgende Beispiel zeigt die Daten, die von einer der untergeordneten Workflow-Ausführungen an die `ProcessEntireBatch` Funktion übergeben wurden.

```
{
  "BatchInput": {
    "MyMultiplicationFactor": 7
  },
  "Items": [1, 2, 3]
}
```

Bei dieser Eingabe zeigt das folgende Beispiel das Ausgabearray mit dem Namen `multiplied`, das von der Lambda-Funktion zurückgegeben wird.

```
{
  "statusCode": 200,
  "multiplied": [7, 14, 21]
}
```

Der Zustandsautomat gibt die folgende Ausgabe zurück, die vier Arrays mit dem Namen `multiplied` für die vier untergeordneten Workflow-Ausführungen enthält. Diese Arrays enthalten die Multiplikationsergebnisse der einzelnen Eingabeelemente.

```
[
  {
    "statusCode": 200,
    "multiplied": [7, 14, 21]
  },
  {
    "statusCode": 200,
    "multiplied": [28, 35, 42]
  },
  {
    "statusCode": 200,
    "multiplied": [49, 56, 63]
```

```
  },
  {
    "statusCode": 200,
    "multiplied": [70]
  }
]
```

Um alle zurückgegebenen Array-Elemente in einem einzigen Ausgabe-Array zu kombinieren, können Sie das ResultSelector Feld verwenden. Definieren Sie dieses Feld im Status Verteilte Zuordnung, um alle `multiplied`Arrays zu finden, alle Elemente innerhalb dieser Arrays zu extrahieren und sie dann zu einem einzigen Ausgabe-Array zu kombinieren.

Um das `ResultSelector` Feld zu verwenden, aktualisieren Sie Ihre Definition des Zustandsautomaten wie im folgenden Beispiel gezeigt.

```
{
  "StartAt": "Pass",
  "States": {
    ...
    ...
    "Map": {
      "Type": "Map",
      ...
      ...
      "ItemsPath": "$.MyItems",
      "ResultSelector": {
        "multiplied.$": "$..multiplied[*]"
      }
    }
  }
}
```

Der aktualisierte Zustandsautomat gibt ein konsolidiertes Ausgabearray zurück, wie im folgenden Beispiel gezeigt.

```
{
  "multiplied": [7, 14, 21, 28, 35, 42, 49, 56, 63, 70]
}
```

Verarbeiten einzelner Datenelemente mit einer Lambda-Funktion

In diesem Tutorial verwenden Sie das ItemBatcher Feld des Status „Verteilte Zuordnung“ von , um mithilfe einer Lambda-Funktion über einzelne Elemente zu iterieren, die in einem Batch vorhanden sind. Der Status Distributed Map startet vier untergeordnete Workflow-Ausführungen. Jeder dieser untergeordneten Workflows führt einen Inline Map-Status aus. Für jede Iteration ruft der Inline Map-Status eine Lambda-Funktion auf und übergibt ein einzelnes Element aus dem Stapel an die Funktion. Die Lambda-Funktion verarbeitet dann das Element und gibt das Ergebnis zurück.

Sie erstellen einen Zustandsautomaten, der eine Multiplikation für ein Array von Ganzzahlen durchführt. Angenommen, das Ganzzahl-Array, das Sie als Eingabe angeben, ist `[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]` und der Multiplikationsfaktor ist `7`. Dann ist das resultierende Array, das nach dem Multiplizieren dieser Ganzzahlen mit einem Faktor von 7 gebildet wird, `[7, 14, 21, 28, 35, 42, 49, 56, 63, 70]`.

Themen

- Schritt 1: Erstellen des Zustandsautomaten
- Schritt 2: Erstellen der Lambda-Funktion
- Schritt 3: Ausführen des Zustandsautomaten

Schritt 1: Erstellen des Zustandsautomaten

In diesem Schritt erstellen Sie den Workflow-Prototyp des Zustandsautomaten, der ein einzelnes Element aus einem Stapel von Elementen an jeden Aufruf der Lambda-Funktion übergibt, die Sie in Schritt 2 erstellen.

- Verwenden Sie die folgende Definition, um einen Zustandsautomaten mit der Step-Functions-Konsole zu erstellen. Informationen zum Erstellen eines Zustandsautomaten finden Sie Schritt 1: Erstellen Sie den Workflow-Prototyp unter im Tutorial Erste Schritte mit der Verwendung von Distributed Map.

  In diesem Zustandsautomaten definieren Sie einen Distributed Map-Status, der ein Array von 10 Ganzzahlen als Eingabe akzeptiert und diese Array-Elemente in Batches an die untergeordneten Workflow-Ausführungen übergibt. Jede untergeordnete Workflow-Ausführung erhält einen Stapel von drei Elementen als Eingabe und führt einen Inline-Map-Status aus. Jede Iteration des Inline Map-Zustands ruft eine Lambda-Funktion auf und übergibt ein Element aus dem Stapel an die

Funktion. Diese Funktion multipliziert dann das Element mit dem Faktor 7 und gibt das Ergebnis zurück.

Die Ausgabe jeder untergeordneten Workflow-Ausführung ist ein JSON-Array, das das Multiplikationsergebnis für jedes übergebene Element enthält.

> **Important**
>
> Stellen Sie sicher, dass Sie den Amazon-Ressourcennamen (ARN) der Lambda-Funktion im folgenden Code durch den ARN der Funktion ersetzen, die Sie in Schritt 2 erstellen.

```
{
  "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"
        }
      }
```

```
    }
  }
}
```

Schritt 2: Erstellen der Lambda-Funktion

In diesem Schritt erstellen Sie die Lambda-Funktion, die jedes Element verarbeitet, das aus dem Batch übergeben wurde.

> **Important**
>
> Stellen Sie sicher, dass sich Ihre Lambda-Funktion unter derselben AWS-Region wie Ihr Zustandsautomat befindet.

So erstellen Sie die Lambda-Funktion:

1. Verwenden Sie die Lambda-Konsole, um eine Python-3.9-Lambda-Funktion mit dem Namen zu erstellen**`ProcessSingleItem`**. Informationen zum Erstellen einer Lambda-Funktion finden Sie unter Schritt 4: Konfigurieren der Lambda-Funktion im Tutorial Erste Schritte mit der Verwendung von Distributed Map State.
2. Kopieren Sie den folgenden Code für die Lambda-Funktion und fügen Sie ihn in den Abschnitt Codequelle Ihrer Lambda-Funktion ein.

   ```
   import json

   def lambda_handler(event, context):

       multiplication_factor = event['MyMultiplicationFactor']
       item = event['MyItem']

       result = multiplication_factor * item

       return {
           'statusCode': 200,
           'multiplied': result
       }
   ```

3. Nachdem Sie Ihre Lambda-Funktion erstellt haben, kopieren Sie den ARN der Funktion, der in der oberen rechten Ecke der Seite angezeigt wird. Um den ARN zu kopieren, klicken Sie auf

Im Folgenden finden Sie ein Beispiel für einen ARN, wobei der Name der Lambda-Funktion *`function-name`* ist (in diesem Fall `ProcessSingleItem`):

```
arn:aws:lambda:us-east-1:123456789012:function:function-name
```

Sie müssen den Funktions-ARN in dem Zustandsautomaten angeben, den Sie in Schritt 1 erstellt haben.

4. Wählen Sie Bereitstellen, um die Änderungen bereitzustellen.

Schritt 3: Ausführen des Zustandsautomaten

Wenn Sie den Zustandsautomaten ausführen, startet der Status Verteilte Zuordnung vier untergeordnete Workflow-Ausführungen, wobei jede Ausführung drei Elemente verarbeitet, während eine Ausführung ein einzelnes Element verarbeitet.

Das folgende Beispiel zeigt die Daten, die an einen der `ProcessSingleItem` Funktionsaufrufe innerhalb einer untergeordneten Workflow-Ausführung übergeben wurden.

```
{
  "MyMultiplicationFactor": 7,
  "MyItem": 1
}
```

Bei dieser Eingabe zeigt das folgende Beispiel die Ausgabe, die von der Lambda-Funktion zurückgegeben wird.

```
{
  "statusCode": 200,
  "multiplied": 7
}
```

Das folgende Beispiel zeigt das JSON-Ausgabearray für eine der untergeordneten Workflow-Ausführungen.

```
[
  {
    "statusCode": 200,
```

```
    "multiplied": 7
  },
  {
    "statusCode": 200,
    "multiplied": 14
  },
  {
    "statusCode": 200,
    "multiplied": 21
  }
]
```

Der Zustandsautomat gibt die folgende Ausgabe zurück, die vier Arrays für die vier untergeordneten Workflow-Ausführungen enthält. Diese Arrays enthalten die Multiplikationsergebnisse der einzelnen Eingabeelemente.

Schließlich ist die Ausgabe des Zustandsautomaten ein Array mit dem Namen , das alle Multiplikationsergebnisse kombiniert`multiplied`, die für die vier untergeordneten Workflow-Ausführungen zurückgegeben wurden.

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

Um alle von den untergeordneten Workflow-Ausführungen zurückgegebenen Multiplikationsergebnisse zu einem einzigen Ausgabe-Array zu kombinieren, können Sie das ResultSelector Feld verwenden. Definieren Sie dieses Feld im Status Verteilte Zuordnung, um alle Ergebnisse zu finden, die einzelnen Ergebnisse zu extrahieren und sie dann zu einem einzigen Ausgabe-Array mit dem Namen zu kombinieren`multiplied`.

Um das `ResultSelector` Feld zu verwenden, aktualisieren Sie Ihre Definition des Zustandsautomaten wie im folgenden Beispiel gezeigt.

```
{
  "StartAt": "Pass",
  "States": {
    ...
    ...
    "Map": {
```

```
      "Type": "Map",
      ...
      ...
      "ItemBatcher": {
        "MaxItemsPerBatch": 3,
        "BatchInput": {
          "MyMultiplicationFactor.$": "$.MyMultiplicationFactor"
        }
      },
      "ItemsPath": "$.MyItems",
      "ResultSelector": {
        "multiplied.$": "$..multiplied"
      }
    }
  }
}
```

Der aktualisierte Zustandsautomat gibt ein konsolidiertes Ausgabearray zurück, wie im folgenden Beispiel gezeigt.

```
{
  "multiplied": [7, 14, 21, 28, 35, 42, 49, 56, 63, 70]
}
```

Starten einer State Machine-Ausführung als Reaktion auf Amazon S3 S3-Ereignisse

Sie können eine AWS Step Functions Zustandsmaschine als Reaktion auf eine EventBridge Amazon-Regel ausführen.

Dieses Tutorial zeigt Ihnen, wie Sie einen State Machine als Ziel für eine EventBridge Amazon-Regel konfigurieren. Diese Regel startet eine State-Machine-Ausführung, wenn Dateien zu einem Amazon Simple Storage Service (Amazon S3) -Bucket hinzugefügt werden.

Für eine praktische Anwendung könnten Sie eine Zustandsmaschine starten, die Operationen an Dateien ausführt, die Sie dem Bucket hinzufügen, z. B. das Erstellen von Miniaturansichten oder das Ausführen von Amazon Rekognition Rekognition-Analysen für Bild- und Videodateien.

In diesem Tutorial starten Sie die Ausführung einer `Helloworld` State Machine, indem Sie eine Datei in einen Amazon S3 S3-Bucket hochladen. Anschließend überprüfen Sie die Beispieleingabe

dieser Ausführung, um die Informationen zu identifizieren, die in der Eingabe der Amazon S3 S3-Ereignisbenachrichtigung enthalten sind, an die gesendet wurde EventBridge.

Themen

Voraussetzung: Erstellen eines Zustandsautomaten

Bevor Sie eine Zustandsmaschine als EventBridge Amazon-Ziel konfigurieren können, müssen Sie die Zustandsmaschine erstellen.

- Verwenden Sie das Tutorial Erstellen einer Zustandsmaschine, die eine Lambda-Funktion verwendet, um eine grundlegende Zustandsmaschine zu erstellen.
- Wenn Sie bereits über einen `Helloworld`-Zustandsautomaten verfügen, fahren Sie mit dem nächsten Schritt fort.

Schritt 1: Erstellen Sie einen Bucket in Amazon S3

Da Sie nun über eine `Helloworld` Zustandsmaschine verfügen, müssen Sie einen Amazon S3 S3-Bucket erstellen, in dem Ihre Dateien gespeichert werden. In Schritt 3 dieses Tutorials richten Sie eine Regel ein, sodass beim Hochladen einer Datei in diesen Bucket eine Ausführung Ihres State Machine EventBridge ausgelöst wird.

1. Navigieren Sie zur Amazon S3 S3-Konsole und wählen Sie dann Bucket erstellen, um den Bucket zu erstellen, in dem Sie Ihre Dateien speichern und eine Amazon S3 S3-Ereignisregel auslösen möchten.
2. Geben Sie unter Bucket name (Bucket-Name) einen Namen ein, beispielsweise *`username`*`-sfn-tutorial`.

> **Note**
>
> Bucket-Namen müssen für alle vorhandenen Bucket-Namen in allen AWS Regionen in Amazon S3 eindeutig sein. Verwenden Sie Ihren eigenen *Benutzernamen*, damit dieser Namen eindeutig ist. Sie müssen alle Ressourcen in der gleichen AWS-Region erstellen.

3. Behalten Sie alle Standardauswahlen auf der Seite bei und wählen Sie Create Bucket.

Schritt 2: Aktivieren Sie die Amazon S3 S3-Ereignisbenachrichtigung mit EventBridge

Nachdem Sie den Amazon S3 S3-Bucket erstellt haben, konfigurieren Sie ihn so, dass Ereignisse immer dann gesendet werden, EventBridge wenn bestimmte Ereignisse in Ihrem S3-Bucket eintreten, z. B. Datei-Uploads.

1. Navigieren Sie zur Amazon S3 S3-Konsole.
2. Wählen Sie in der Liste Buckets den Namen des Buckets aus, für den Sie Ereignisse aktivieren möchten.
3. Wählen Sie Properties (Eigenschaften).
4. Scrollen Sie auf der Seite nach unten, um den Abschnitt Event-Benachrichtigungen aufzurufen, und wählen Sie dann im EventBridgeAmazon-Unterabschnitt Bearbeiten aus.
5. Wählen Sie unter Benachrichtigungen EventBridge für alle Ereignisse in diesem Bucket an Amazon senden die Option Aktiviert aus.
6. Wählen Sie Änderungen speichern aus.

> **Note**
>
> Nach der Aktivierung dauert es etwa fünf Minuten EventBridge, bis die Änderungen wirksam werden.

Schritt 3: Erstellen Sie eine EventBridge Amazon-Regel

Nachdem Sie über eine Zustandsmaschine verfügen und den Amazon S3 S3-Bucket erstellt und für das Senden von Ereignisbenachrichtigungen konfiguriert haben EventBridge, erstellen Sie eine EventBridge Regel.

> **Note**
>
> Sie müssen die EventBridge Regel in derselben AWS Region wie der Amazon S3 S3-Bucket konfigurieren.

So erstellen Sie die -Regel

1. Navigieren Sie zur EventBridge Amazon-Konsole und wählen Sie Regel erstellen aus.

   > **Tip**
   >
   > Sie können auch im Navigationsbereich der EventBridge Konsole unter Busse die Option Regeln und dann Regel erstellen auswählen.

2. Geben Sie einen Namen für Ihre Regel ein (z. B.*`S3Step Functions`*) und geben Sie optional eine Beschreibung für die Regel ein.
3. Behalten Sie für Eventbus und Regeltyp die Standardauswahlen bei.
4. Wählen Sie Weiter aus. Dadurch wird die Seite Event-Pattern erstellen geöffnet.
5. Scrollen Sie nach unten zum Abschnitt Event-Pattern und gehen Sie wie folgt vor:

   a. Behalten Sie unter Ereignisquelle die Standardauswahl für AWSEreignisse oder EventBridge Partnerereignisse bei.

   b. Wählen Sie als AWSService Simple Storage Service (S3).

   c. Wählen Sie als Ereignistyp Amazon S3 Event Notification aus.

   d. Wählen Sie Bestimmte Ereignisse und anschließend Objekt erstellt aus.

   e. Wählen Sie Bestimmte Buckets nach Namen aus und geben Sie den Bucket-Namen ein, den Sie in Schritt 1 (*`username`*`-sfn-tutorial`) zum Speichern Ihrer Dateien erstellt haben.

   f. Wählen Sie Weiter aus. Dadurch wird die Seite Ziel (e) auswählen geöffnet.

So erstellen Sie das Ziel

1. Behalten Sie in Ziel 1 die Standardauswahl des AWSDienstes bei.
2. Wählen Sie in der Dropdownliste Ziel auswählen die Option Step Functions state machine aus.
3. Wählen Sie in der Liste State Machine den State Machine aus, den Sie zuvor erstellt haben (z. B.`Helloworld`).
4. Behalten Sie alle Standardauswahlen auf der Seite bei und wählen Sie Weiter. Dadurch wird die Seite „Tags konfigurieren“ geöffnet.
5. Wählen Sie erneut Next (Weiter). Dadurch wird die Seite Überprüfen und erstellen geöffnet.
6. Überprüfen Sie die Details der Regel und wählen Sie dann Create rule (Regel erstellen) aus.

   Die Regel wird erstellt und die Seite „Regeln“ wird angezeigt, auf der alle Ihre EventBridge Amazon-Regeln aufgeführt sind.

Schritt 4: Testen der Regel

Jetzt, da alles an seinem Platz ist, testen Sie das Hinzufügen einer Datei zum Amazon S3 S3-Bucket und schauen Sie sich dann die Eingabe der resultierenden State-Machine-Ausführung an.

1. Fügen Sie Ihrem Amazon S3 S3-Bucket eine Datei hinzu.

   Navigieren Sie zur Amazon S3 S3-Konsole, wählen Sie den Bucket aus, den Sie zum Speichern von Dateien (*`username`*`-sfn-tutorial`) erstellt haben, und wählen Sie dann Upload.
2. Fügen Sie beispielsweise *`test.png`* eine Datei hinzu und wählen Sie dann Upload.

   Dadurch wird die Ausführung Ihres Zustandsautomaten gestartet, der Informationen aus AWS CloudTrail als Eingabe übergibt.
3. Überprüfen Sie die Ausführung für Ihren Zustandsautomaten.

   Navigieren Sie zur Step Functions Functions-Konsole und wählen Sie den Zustandsmaschine aus, der in Ihrer EventBridge Amazon-Regel verwendet wird (`Helloworld`).
4. Wählen Sie die letzte Ausführung dieser Zustandsmaschine aus und erweitern Sie den Abschnitt Ausführungseingabe.

   Diese Eingabe enthält Informationen wie den Bucket-Namen und den Objektnamen. In einem realen Anwendungsfall kann ein Zustandsautomat diese Eingabe verwenden, um Aktionen für das betreffende Objekt auszuführen.

Beispiel der Ausführungseingabe

Das folgende Beispiel zeigt eine typische Eingabe für die State-Machine-Ausführung.

```
{
    "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"
    }
  }
```

Erstellen einer Step-Functions-API mit API Gateway

Sie können Amazon API Gateway verwenden, um Ihre AWS Step Functions APIs mit Methoden in einer API Gateway-API zu verknüpfen. Wenn eine HTTPS-Anforderung an eine API-Methode gesendet wird, ruft API Gateway Ihre Step-Functions-API-Aktionen auf.

In diesem Tutorial erfahren Sie, wie Sie eine API erstellen, die eine Ressource und die `POST`-Methode verwendet, um mit der API-Aktion `StartExecution` zu kommunizieren. Sie verwenden die

AWS Identity and Access Management (IAM)-Konsole, um eine Rolle für API Gateway zu erstellen. Anschließend verwenden Sie die API Gateway-Konsole, um eine API Gateway-API zu erstellen, eine Ressource und Methode zu erstellen und die Methode der `StartExecution` API-Aktion zuzuordnen. Schließlich stellen Sie Ihre API bereit und testen sie.

> **Note**
>
> Obwohl Amazon API Gateway eine Step-Functions-Ausführung starten kann, indem es aufruft`StartExecution`, müssen Sie aufrufen, `DescribeExecution` um das Ergebnis zu erhalten.

Themen

Schritt 1: Erstellen einer IAM-Rolle für API Gateway

Bevor Sie Ihre API Gateway-API erstellen, müssen Sie API Gateway die Berechtigung zum Aufrufen von Step Functions-API-Aktionen erteilen.

So richten Sie Berechtigungen für API Gateway ein

1. Melden Sie sich bei der IAM-Konsole an und wählen Sie Rollen , Rolle erstellen aus.
2. Gehen Sie auf der Seite Select trusted entity (Vertrauenswürdige Entität auswählen) wie folgt vor:
   a. Behalten Sie für Vertrauenswürdiger Entitätstyp die Standardauswahl von beiAWS-Service.
   b. Wählen Sie für Anwendungsfall die Option API Gateway aus der Dropdown-Liste aus.
3. Wählen Sie API Gateway und dann Weiter aus.
4. Wählen Sie auf der Seite Add permissions (Berechtigungen hinzufügen) die Option Next (Weiter) aus.
5. (Optional) Geben Sie auf der Seite Name, Überprüfung und Erstellung Details ein, z. B. den Rollennamen. Geben Sie z. B. ei **`APIGatewayToStepFunctions`**.
6. Wählen Sie Rolle erstellen aus.

Die IAM-Rolle wird in der Liste der Rollen angezeigt.

7. Wählen Sie den Namen Ihrer Rolle und notieren Sie den Role ARN (Rollen-ARN), wie im folgenden Beispiel gezeigt.

```
arn:aws:iam::123456789012:role/APIGatewayToStepFunctions
```

So fügen Sie eine Richtlinie an die IAM-Rolle an

1. Suchen Sie auf der Seite Roles (Rollen) Ihre Rolle (`APIGatewayToStepFunctions`) und wählen Sie dann die Rolle aus.
2. Wählen Sie auf der Registerkarte Berechtigungen die Option Berechtigungen hinzufügen und dann Richtlinien anfügen aus.
3. Suchen Sie auf der Seite Richtlinie anfügen nach `AWSStepFunctionsFullAccess`, wählen Sie die Richtlinie und dann Berechtigungen hinzufügen aus.

Schritt 2: API Gateway-API erstellen

Nachdem Sie Ihre IAM-Rolle erstellt haben, können Sie Ihre benutzerdefinierte API Gateway-API erstellen.

So erstellen Sie die API

1. Öffnen Sie die Amazon API Gateway-Konsole und wählen Sie dann API erstellen aus.
2. Wählen Sie auf der Seite API-Typ auswählen im Bereich REST API die Option Erstellen aus.
3. Wählen Sie auf der Seite REST-API erstellen die Option Neue API aus und geben Sie dann ***StartExecutionAPI*** für den API-Namen ein.
4. Behalten Sie den API-Endpunkttyp als Regional bei und wählen Sie dann API erstellen aus.

Erstellen Sie eine Ressource

1. Wählen Sie auf der Seite Ressourcen der ***StartExecutionAPI*** *die* Option Ressource erstellen aus.

2. Geben Sie auf der Seite Ressource erstellen **`execution`** für Ressourcenname ein und wählen Sie dann Ressource erstellen aus.

So erstellen Sie eine POST-Methode

1. Wählen Sie die /execution-Ressource und dann Methode erstellen aus.
2. Wählen Sie für Methodentyp `POST`.
3. Wählen Sie für Integrationstyp die Option -AWS Service aus.
4. AWS-RegionWählen Sie für eine Region aus der Liste aus.

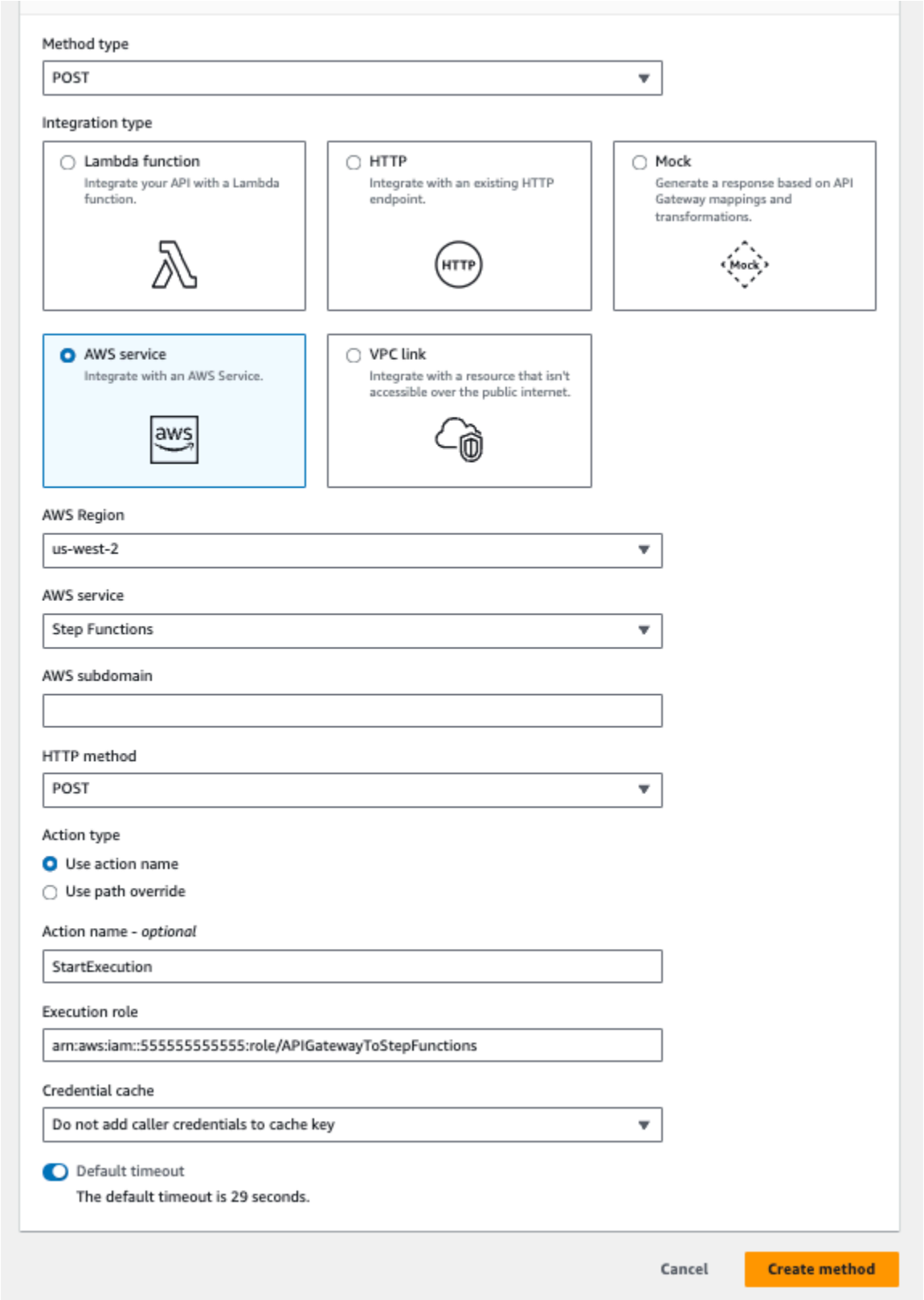
> **Note**
>
> Informationen zu Regionen, die derzeit Step Functions unterstützen, finden Sie unter Unterstützte Regionen.

5. AWS-ServiceWählen Sie für Step Functions aus der Liste aus.
6. Lassen Sie die AWS -Subdomain leer.
7. Wählen Sie für HTTP-Methode POST aus der Liste aus.

> **Note**
>
> Alle Step-Functions-API-Aktionen verwenden die HTTP-`POST`Methode.

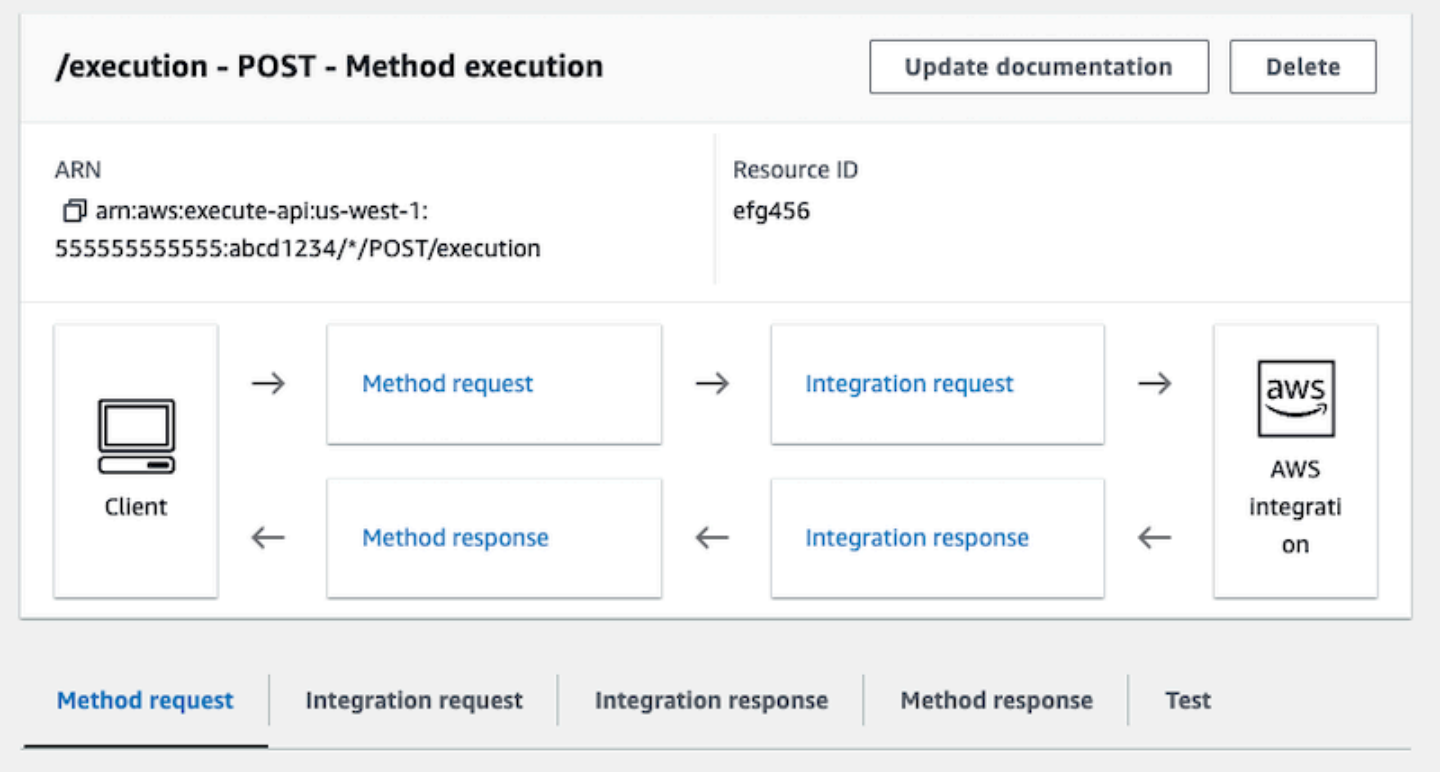
8. Wählen Sie für Aktionstyp die Option Aktionsnamen verwenden aus.
9. Für Aktionsname geben Sie **`StartExecution`** ein.
10. Geben Sie für Ausführungsrolle den Rollen-ARN der IAM-Rolle ein, die Sie zuvor erstellt haben, wie im folgenden Beispiel gezeigt.

```
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. Behalten Sie die Standardoptionen für Anmeldeinformations-Cache und Standard-Timeout bei und wählen Sie dann Speichern aus.

Die visuelle Zuordnung zwischen API Gateway und Step Functions wird auf der Seite /execution – POST – Method execution angezeigt.



Schritt 3: Testen und Bereitstellen der API Gateway-API

Wenn Sie die API erstellt haben, können Sie sie testen und bereitstellen.

So testen Sie die Kommunikation zwischen API Gateway und Step Functions

1. Wählen Sie auf der Seite /execution – POST – Method Execution die Registerkarte Test aus. Möglicherweise müssen Sie die rechte Pfeiltaste wählen, um die Registerkarte anzuzeigen.
2. Kopieren Sie auf der Registerkarte /execution - POST - Method Test die folgenden Anforderungsparameter in den Abschnitt Anforderungstext mit dem ARN eines vorhandenen Zustandsautomaten (oder erstellen Sie einen neuen Zustandsautomaten, der eine Lambda-Funktion verwendet) und wählen Sie dann Test aus.

```
{
   "input": "{}",
```

```
   "name": "MyExecution",
   "stateMachineArn": "arn:aws:states:us-
east-1:123456789012:stateMachine:HelloWorld"
}
```

Weitere Informationen finden Sie unter `StartExecution` Anforderungssyntax in der APIAWS Step Functions -Referenz zu .

> **Note**
>
> Wenn Sie den ARN Ihres Zustandsautomaten nicht in den Hauptteil Ihres API Gateway-Aufrufs aufnehmen möchten, können Sie eine Zuweisungsvorlage auf der Registerkarte Integrationsanforderung konfigurieren, wie im folgenden Beispiel gezeigt.
>
> ```
> {
>     "input": "$util.escapeJavaScript($input.json('$'))",
>     "stateMachineArn": "$util.escapeJavaScript($stageVariables.arn)"
> }
> ```
>
> Mit diesem Ansatz können Sie ARNs verschiedener Zustandsautomaten basierend auf Ihrer Entwicklungsphase angeben (z. B. `devtest`, und `prod`). Weitere Informationen zum Angeben von Stufenvariablen in einer Zuweisungsvorlage finden Sie unter `$stageVariables` im API Gateway-Entwicklerhandbuch.

3. Die Ausführung beginnt und der Ausführungs-ARN und sein Epochendatum werden unter Antworttext angezeigt.

```
{
   "executionArn": "arn:aws:states:us-
east-1:123456789012:execution:HelloWorld:MyExecution",
   "startDate": 1486768956.878
}
```

> **Note**
>
> Sie können die Ausführung anzeigen, indem Sie Ihren Zustandsautomaten auf der AWS Step Functions -Konsole auswählen.

Stellen Sie Ihre API bereit

1. Wählen Sie auf der Seite Ressourcen der ***StartExecutionAPI*** *die* Option API bereitstellen aus.
2. Wählen Sie für Stufe die Option Neue Stufe aus.
3. Geben Sie für Stage name (Stufenname) **alpha** ein.
4. (Optional) Geben Sie unter Description (Beschreibung) eine Beschreibung ein.
5. Wählen Sie Bereitstellen.

Testen der Bereitstellung

1. Erweitern Sie auf der Seite Stufen der ***StartExecutionAPI*** *die* Optionen alpha , /, /execution , POST und wählen Sie dann die POST-Methode aus.
2. Wählen Sie unter Methodenüberschreibungen das Kopiersymbol aus, um die Aufruf-URL Ihrer API zu kopieren. Die vollständige URL sollte wie im folgenden Beispiel aussehen.

```
https://a1b2c3d4e5.execute-api.us-east-1.amazonaws.com/alpha/execution
```

3. Führen Sie in der Befehlszeile mithilfe des ARN Ihres Zustandsautomaten den Befehl `curl` aus und rufen Sie dann den URL Ihrer Bereitstellung auf, wie im folgenden Beispiel gezeigt.

```
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
```

Der Ausführung-ARN und sein Epochendatum werden zurückgegeben, wie im folgenden Beispiel gezeigt.

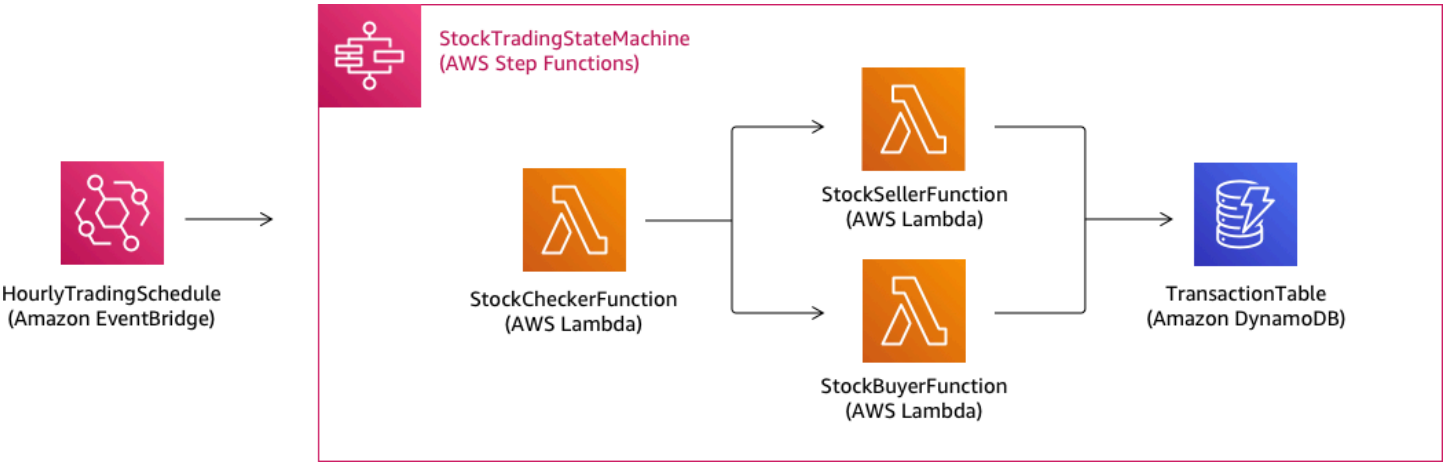
```
{"executionArn":"arn:aws:states:us-
east-1:123456789012:execution:HelloWorld:MyExecution","startDate":1.486772644911E9}
```

> **Note**
>
> Wenn die Fehlermeldung „Fehlendes Authentifizierungstoken“ angezeigt wird, stellen Sie sicher, dass die Aufruf-URL mit /execution endet.

Erstellen einer Step Functions Functions-ZustandsmaschineAWS SAM

Diese Anleitung zeigt Ihnen, wie Sie eine AWS SAM-Beispielanwendung herunterladen, erstellen und bereitstellen, die einen AWS Step Functions-Zustandsautomaten enthält. Diese Anwendung erstellt einen Pseudo-Aktienhandel-Workflow, der nach einem vordefinierten Zeitplan ausgeführt wird (beachten Sie, dass der Zeitplan standardmäßig deaktiviert ist, um Gebühren zu vermeiden).

Das folgende Diagramm zeigt die Komponenten dieser Anwendung:



Im Folgenden finden Sie eine Vorschau der Befehle, die Sie zum Erstellen der Beispielanwendung ausführen. Weitere Informationen zu den einzelnen Befehlen finden Sie in den Abschnitten weiter unten auf dieser Seite.

```
# 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
```

Voraussetzungen

In diesem Handbuch wird davon ausgegangen, dass Sie die Schritte unter Installieren der AWS SAM-CLI für Ihr Betriebssystem ausgeführt haben. Es wird davon ausgegangen, dass Sie Folgendes getan haben:

1. ErstelltAWSKonto.
2. Konfigurierte IAM-Berechtigungen
3. Homebrew wurde installiert. Hinweis: Homebrew ist nur eine Voraussetzung für Linux und macOS.
4. Die AWS SAM-CLI wurde installiert. Hinweis: Stellen Sie sicher, dass Sie Version 0.52.0 oder höher besitzen. Sie können überprüfen, über welche Version Sie verfügen, indem Sie den Befehl `sam --version` ausführen.

Schritt 1: Herunterladen einer AWS SAM-Beispielanwendung

Befehl zum Ausführen:

```
sam init
```

Folgen Sie den Anweisungen auf dem Bildschirm, um Folgendes auszuwählen:

1. Vorlage: AWSSchnellstart-Vorlagen
2. Sprache: Python, Ruby, NodeJS, Go, Java oder .NET
3. Projektname: (Name Ihrer Wahl – Standard ist `sam-app`)
4. Schnellstart-Anwendung:Mehrstufiger Arbeitsablauf

WasAWS SAMmacht:

Mit diesem Befehl wird ein Verzeichnis mit dem Namen erstellt, den Sie für die Eingabeaufforderung „Projektname“ angegeben haben (Standardeinstellung ist `sam-app`). Der spezifische Inhalt des Verzeichnisses hängt von der gewählten Sprache ab.

Im Folgenden sind die Verzeichnisinhalte aufgeführt, wenn Sie eine der Python-Laufzeitumgebungen auswählen:

```
### 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
```

Es gibt zwei besonders interessante Dateien, die Sie sich ansehen können:

- `template.yaml`: Enthält dieAWS SAMVorlage, die Ihre Anwendung definiertAWSRessourcen.
- `statemachine/stockTrader.asl.json`: Enthält die Zustandsautomaten-Definition der Anwendung, die in Amazon States Language geschrieben ist.

Sie können den folgenden Eintrag in der `template.yaml`-Datei sehen, der auf die Definitionsdatei des Zustandsautomaten verweist:

```
    Properties:
      DefinitionUri: statemachine/stock_trader.asl.json
```

Es kann hilfreich sein, die State-Machine-Definition als separate Datei beizubehalten, anstatt sie in die Datei einzubettenAWS SAMVorlage. Beispielsweise ist es einfacher, Änderungen an der State-Machine-Definition nachzuverfolgen, wenn Sie die Definition nicht in die Vorlage aufnehmen. Sie können Workflow Studio verwenden, um die Zustandsmaschinen-Definition zu erstellen und zu

verwalten und die Definition von der Konsole direkt in die Sprachspezifikationsdatei von Amazon States zu exportieren, ohne sie mit der Vorlage zusammenzuführen.

Weitere Informationen zur Beispielanwendung finden Sie in der Datei `README.md` im Projektverzeichnis.

Schritt 2: Erstellen Ihrer Anwendung

Befehl zum Ausführen:

Wechseln Sie zuerst in das Projektverzeichnis (d. h. das Verzeichnis, in dem sich die `template.yaml`-Datei für die Beispielanwendung befindet; standardmäßig `sam-app`) und führen Sie dann folgenden Befehl aus:

```
sam build
```

Beispielausgabe:

```
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
```

WasAWS SAMmacht:

DasAWS SAMCLI enthält Abstraktionen für eine Reihe von Lambda-Laufzeiten zum Erstellen Ihrer Abhängigkeiten und kopiert alle Build-Artefakte in Staging-Ordner, sodass alles bereit ist, gepackt und bereitgestellt zu werden. Der `sam build`-Befehl erstellt alle Abhängigkeiten Ihrer Anwendung und kopiert die Build-Artefakte in Ordner unter `.aws-sam/build`.

Schritt 3: Bereitstellen derAWSCloud

Befehl zum Ausführen:

```
sam deploy --guided
```

Folgen Sie den Anweisungen auf dem Bildschirm. Sie können einfach mit `Enter` antworten, um die in der interaktiven Erfahrung bereitgestellten Standardoptionen zu akzeptieren.

WasAWS SAMmacht:

Mit diesem Befehl wird Ihre Anwendung inAWSWolke. Es benötigt die Bereitstellungsartefakte, die Sie mit dem erstellen`sam build`Befehl, Paket und lädt sie in einen Amazon S3 S3-BucketAWS SAMCLI und stellt die Anwendung bereit mitAWS CloudFormation. In der Ausgabe des Deploy-Befehls können Sie die Änderungen sehen, die an Ihrem AWS CloudFormation-Stack vorgenommen werden.

Sie können überprüfen, ob die Step Functions Functions-Beispiel-Zustandsmaschine erfolgreich bereitgestellt wurde, indem Sie die folgenden Schritte ausführen:

1. Melden Sie sich anAWS Management Consoleund öffnen Sie die Step Functions Functions-Konsole unterhttps://console.aws.amazon.com/states/.
2. Wählen Sie in der linken Navigation die Option Zustandsautomaten aus.
3. Suchen Sie Ihren neuen Zustandsautomaten in der Liste und wählen Sie ihn aus. Es wird benannt StockTradingStateMachine-*<unique-hash>*.
4. Wählen Sie die Registerkarte Definition.

Sie sollten nun eine visuelle Darstellung Ihres Zustandsautomaten sehen. Sie können überprüfen, ob die visuelle Darstellung mit der Zustandsautomatendefinition in der Datei `statemachine/stockTrader.asl.json` Ihres Projektverzeichnisses übereinstimmt.

Fehlerbehebung

SAM CLI-Fehler: „keine solche Option: --guided“

Bei der Ausführung von `sam deploy` wird der folgende Fehler angezeigt:

```
Error: no such option: --guided
```

Dies bedeutet, dass Sie eine ältere Version der AWS SAM-CLI verwenden, die den Parameter `--guided` nicht unterstützt. Um dies zu beheben, können Sie entweder Ihre Version von AWS SAM-CLI auf 0.33.0 oder höher aktualisieren oder den Parameter `--guided` aus im Befehl `sam deploy` weglassen.

SAM-CLI-Fehler: „Fehler beim Erstellen verwalteter Ressourcen: Anmeldeinformationen konnten nicht gefunden werden“

Bei der Ausführung von `sam deploy` wird der folgende Fehler angezeigt:

```
Error: Failed to create managed resources: Unable to locate credentials
```

Das bedeutet, dass Sie nicht eingerichtet habenAWSZugangsdaten zur Aktivierung desAWS SAMCLI zum ErstellenAWSServiceanrufe. Um dies zu beheben, müssen SieAWSReferenzen. Weitere Informationen finden Sie unterEinrichtenAWSReferenzenin derAWS Serverless Application ModelLeitfaden für Entwickler.

Bereinigen

Wenn Sie das nicht mehr benötigenAWSRessourcen, die Sie beim Ausführen dieses Tutorials erstellt haben, können Sie entfernen, indem Sie dieAWS CloudFormationStapel, den Sie bereitgestellt haben.

Gehen Sie folgendermaßen vor, um den AWS CloudFormation-Stack zu löschen, der mit diesem Tutorial über die AWS Management Console erstellt wurde:

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die AWS CloudFormation-Konsole unter https://console.aws.amazon.com/cloudformation.
2. Wählen Sie im linken Navigationsbereich Stack aus.
3. Wählen Sie in der Liste der Stacks sam-app (oder den Namen des von Ihnen erstellten Stacks) aus.
4. Wählen Sie Löschen.

Wenn Sie fertig sind, ändert sich der Status des Stacks in DELETE_COMPLETE.

Alternativ können Sie den AWS CloudFormation-Stack löschen, indem Sie den folgenden AWS CLI-Befehl ausführen:

```
aws cloudformation delete-stack --stack-name sam-app --region region
```

Gelöschten Stack überprüfen

Für beide Methoden zum Löschen vonAWS CloudFormationStapel, Sie können überprüfen, ob er gelöscht wurde, indem Sie zuhttps://console.aws.amazon.com/cloudformation, wählendStapelim linken Navigationsbereich und wählenGelöschtin der Drop-down-Liste rechts neben dem Such-Textfeld Sie sollten Ihren Stacknamen sam-app (oder den Namen des Stacks, den Sie erstellt haben) in der Liste der gelöschten Stacks sehen.

Einen Activity State Machine mithilfe von Step Functions erstellen

In diesem Tutorial erfahren Sie, wie Sie einen aktivitätsbasierten Zustandsautomaten mithilfe von Java und AWS Step Functions erstellen. Mithilfe von Aktivitäten können Sie Worker-Code, der woanders ausgeführt wird, von Ihrem Zustandsmaschine aus steuern. Eine Übersicht finden Sie unter Aktivitäten im Abschnitt So funktioniert Step Functions.

Zum Durcharbeiten dieses Tutorials ist Folgendes erforderlich:

- Das SDK für Java. Die Beispielaktivität in diesem Tutorial ist eine Java-Anwendung, die verwendet, AWS SDK for Java um mit zu kommunizieren AWS.
- AWS Anmeldeinformationen in der Umgebung oder in der AWS Standardkonfigurationsdatei. Weitere Informationen finden Sie unter Einrichten Ihrer AWS Anmeldeinformationen im AWS SDK for Java Entwicklerhandbuch.

Themen

- Schritt 1: Erstellen einer Aktivität
- Schritt 2: Erstellen Sie eine Zustandsmaschine
- Schritt 3: Implementieren eines Workers
- Schritt 4: Führen Sie die Zustandsmaschine aus
- Schritt 5: Ausführen und Beenden des Workers

Schritt 1: Erstellen einer Aktivität

Sie müssen Step Functions auf die Aktivität aufmerksam machen, deren Worker (ein Programm) Sie erstellen möchten. Step Functions antwortet mit einem Amazon-Ressourcennamen (ARN), der eine

Identität für die Aktivität festlegt. Verwenden Sie diese Identität, um die Informationen zwischen Ihrem Zustandsautomaten und dem Worker zu koordinieren.

> **Important**
>
> Stellen Sie sicher, dass sich Ihre Aktivitätsaufgabe unter demselben AWS Konto befindet wie Ihr State Machine.

1. Wählen Sie in der Step Functions Functions-Konsole im Navigationsbereich auf der linken Seite Aktivitäten aus.
2. Wählen Sie Create activity (Aktivität erstellen) aus.
3. Geben Sie beispielsweise einen Namen für die Aktivität ein und wählen Sie dann Aktivität erstellen aus. *`get-greeting`*
4. Wenn Ihre Aktivitätsaufgabe erstellt wird, notieren Sie sich deren ARN, wie im folgenden Beispiel gezeigt.

   ```
   arn:aws:states:us-east-1:123456789012:activity:get-greeting
   ```

Schritt 2: Erstellen Sie eine Zustandsmaschine

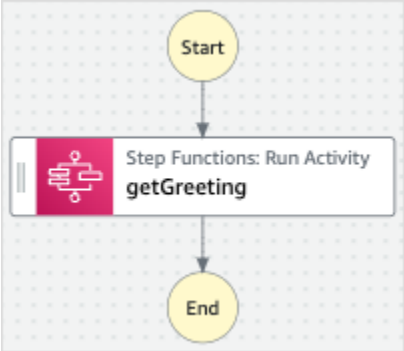
Erstellen Sie einen Zustandsautomaten, der festlegt, wann Ihre Aktivität aufgerufen wird und wann Ihr Worker seine primäre Arbeit ausführen, die Ergebnisse sammeln und zurückgeben soll. Um die Zustandsmaschine zu erstellen, verwenden Sie Code-Editor den Workflow Studio.

1. Wählen Sie in der Step Functions Functions-Konsole im Navigationsbereich auf der linken Seite State Machines aus.
2. Wählen Sie auf der Seite State Machines die Option Create State Machine aus.
3. Wählen Sie im Dialogfeld Vorlage auswählen die Option Leer aus.
4. Wählen Sie Select (Auswählen). Dadurch wird Workflow Studio in geöffnetEntwurfsmodus.
5. Für dieses Tutorial schreiben Sie die Amazon States Language (ASL-) Definition Ihrer Zustandsmaschine in den Code-Editor. Wählen Sie dazu Code.
6. Entfernen Sie den vorhandenen Boilerplate-Code und fügen Sie den folgenden Code ein. Denken Sie daran, den Beispiel-ARN in diesem Code durch den ARN der Aktivitätsaufgabe zu ersetzen, die Sie zuvor in dem `Resource` Feld erstellt haben.

```
{
  "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
    }
  }
}
```

Dies ist eine Beschreibung Ihrer Zustandsmaschine, die die Amazon States Language (ASL) verwendet. Sie definiert einen einzelnen `Task`-Zustand namens `getGreeting`. Weitere Informationen finden Sie unter State Machine Structure.

7. Vergewissern Sie sichBereich zur Grafikvisualisierung, dass das Workflow-Diagramm für die ASL-Definition, die Sie hinzugefügt haben, dem folgenden Diagramm ähnelt.



8. Geben Sie einen Namen für Ihre Zustandsmaschine ein. Wählen Sie dazu das Bearbeitungssymbol neben dem Standardnamen der Zustandsmaschine von MyStateMachine. Geben Sie dann unter State-Machine-Konfiguration einen Namen in das Feld State-Machine-Name ein.

   Geben Sie für dieses Tutorial den Namen **`ActivityStateMachine`** ein.

9. (Optional) Geben Sie unter State-Machine-Konfiguration weitere Workflow-Einstellungen an, z. B. den Zustandsmaschinentyp und seine Ausführungsrolle.

   Behalten Sie für dieses Tutorial alle Standardauswahlen in den State-Machine-Einstellungen bei.

Wenn Sie zuvor eine IAM-Rolle mit den richtigen Berechtigungen für Ihren Zustandsmaschine erstellt haben und diese verwenden möchten, wählen Sie unter Berechtigungen die Option Vorhandene Rolle auswählen und dann eine Rolle aus der Liste aus. Oder wählen Sie Einen Rollen-ARN eingeben aus und geben Sie dann einen ARN für diese IAM-Rolle ein.

10. Wählen Sie im Dialogfeld „Rollenerstellung bestätigen“ die Option Bestätigen aus, um fortzufahren.

    Sie können auch Rolleneinstellungen anzeigen wählen, um zur State-Machine-Konfiguration zurückzukehren.

> **Note**
>
> Wenn Sie die von Step Functions erstellte IAM-Rolle löschen, kann Step Functions sie später nicht mehr neu erstellen. Ebenso kann Step Functions ihre ursprünglichen Einstellungen später nicht wiederherstellen, wenn Sie die Rolle ändern (z. B. indem Sie Step Functions aus den Principals in der IAM-Richtlinie entfernen).

Schritt 3: Implementieren eines Workers

Erstellen Sie einen Worker. Ein Worker ist ein Programm, das für Folgendes verantwortlich ist:

- Polling-Step-Funktionen für Aktivitäten, die die `GetActivityTask` API-Aktion verwenden.
- Durchführen der Arbeit der Aktivität mit Ihrem Code (z. B. die Methode `getGreeting()` im folgenden Code).
- Zurückgeben der Ergebnisse mithilfe der API-Aktionen `SendTaskSuccess`, `SendTaskFailure` und `SendTaskHeartbeat`.

> **Note**
>
> Ein umfassendes Beispiel für einen Aktivitäts-Worker finden Sie unter Beispiel für einen Aktivitäts-Worker in Ruby. Dieses Beispiel verwendet eine auf bewährten Methoden basierende Implementierung, die Sie als Referenz für Ihren Aktivitäts-Worker verwenden können. Der Code implementiert ein Verbraucher/Erzeuger-Muster mit einer konfigurierbaren Anzahl an Threads für Poller und Aktivitäts-Worker.

Implementieren des Workers

1. Erstellen Sie eine Datei namens `GreeterActivities.java`.
2. Fügen Sie ihr folgenden Code hinzu.

```
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**
>
> Bei der Klasse `EnvironmentVariableCredentialsProvider` in diesem Beispiel wird davon ausgegangen, dass die Umgebungsvariablen `AWS_ACCESS_KEY_ID` (oder `AWS_ACCESS_KEY`) und `AWS_SECRET_KEY` (oder `AWS_SECRET_ACCESS_KEY`) gesetzt sind. Weitere Informationen zur Bereitstellung der erforderlichen Anmeldeinformationen für das Werk finden Sie AWSCredentialsProviderin der AWS SDK for Java API-Referenz und unter AWS Anmeldeinformationen und Region für die Entwicklung einrichten im AWS SDK for Java Entwicklerhandbuch.
>
> Standardmäßig wartet das AWS SDK bei jedem Vorgang bis zu 50 Sekunden, bis es Daten vom Server empfängt. Die Operation `GetActivityTask` ist eine Langabfrage-Operation, die bis zu 60 Sekunden auf die nächste verfügbare Aufgabe wartet.
> Stellen Sie den Wert auf 70 Sekunden ein, SocketTimeout um zu verhindern, dass ein `SocketTimeoutException` Fehler angezeigt wird.

3. Ersetzen Sie in der Liste der Parameter des Konstruktors `GetActivityTaskRequest().withActivityArn()` den Wert `ACTIVITY_ARN` mit dem ARN der Aktivitätsaufgabe, die Sie zuvor erstellt haben.

Schritt 4: Führen Sie die Zustandsmaschine aus

Wenn Sie die Ausführung der Zustandsmaschine starten, fragt Ihr Worker Step Functions nach Aktivitäten ab, führt seine Arbeit aus (unter Verwendung der von Ihnen bereitgestellten Eingaben) und gibt die Ergebnisse zurück.

1. Wählen Sie auf der ***ActivityStateMachine***Seite die Option Ausführung starten aus.

   Das Dialogfeld Ausführung starten wird angezeigt.

2. Gehen Sie im Dialogfeld Ausführung starten wie folgt vor:

   a. (Optional) Um Ihre Ausführung zu identifizieren, können Sie im Feld Name einen Namen dafür angeben. Standardmäßig generiert Step Functions automatisch einen eindeutigen Ausführungsnamen.

   > **Note**
   >
   > Mit Step Functions können Sie Namen für Zustandsmaschinen, Ausführungen und Aktivitäten sowie Beschriftungen erstellen, die Nicht-ASCII-Zeichen enthalten. Diese Nicht-ASCII-Namen funktionieren nicht mit Amazon. CloudWatch Um sicherzustellen, dass Sie CloudWatch Messwerte verfolgen können, wählen Sie einen Namen, der nur ASCII-Zeichen verwendet.

   b. Geben Sie im Eingabefeld die folgende JSON-Eingabe ein, um Ihren Workflow auszuführen.

   ```
   {
     "who": "AWS Step Functions"
   }
   ```

   c. Wählen Sie Start execution (Ausführung starten) aus.

   d. Die Step Functions Functions-Konsole leitet Sie zu einer Seite weiter, die mit Ihrer Ausführungs-ID betitelt ist. Diese Seite wird als Seite mit den Ausführungsdetails bezeichnet. Auf dieser Seite können Sie die Ausführungsergebnisse im Verlauf der Ausführung oder nach deren Abschluss überprüfen.

   Um die Ausführungsergebnisse zu überprüfen, wählen Sie in der Diagrammansicht einzelne Status aus und wählen Sie dann die einzelnen Registerkarten im Schrittdetails Bereich, um die Details der einzelnen Status, einschließlich Eingabe, Ausgabe und Definition, anzuzeigen. Einzelheiten zu den Ausführungsinformationen, die Sie auf der Seite mit

den Ausführungsdetails einsehen können, finden Sie unterSeite mit Ausführungsdetails – Schnittstellenübersicht.

Schritt 5: Ausführen und Beenden des Workers

Damit der Worker Ihren Zustandsautomaten nach Aktivitäten abfragen kann, müssen Sie den Worker ausführen.

1. Navigieren Sie über die Befehlszeile zu dem Verzeichnis, in dem Sie `GreeterActivities.java` erstellt haben.
2. Um das AWS SDK zu verwenden, fügen Sie den vollständigen Pfad der `third-party` Verzeichnisse `lib` und zu den Abhängigkeiten Ihrer Build-Datei und zu Ihrer Java-Datei hinzu`CLASSPATH`. Weitere Informationen finden Sie unter Herunterladen und Extrahieren des SDK im AWS SDK for Java Entwicklerhandbuch.
3. Kompilieren Sie die Datei.

   ```
   $ javac GreeterActivities.java
   ```

4. Führen Sie die Datei aus.

   ```
   $ java GreeterActivities
   ```

5. Navigieren Sie in der Step Functions Functions-Konsole zur Seite mit den Ausführungsdetails.
6. Wenn die Ausführung abgeschlossen ist, überprüfen Sie die Ergebnisse Ihrer Ausführung.
7. Beenden Sie den Worker.

Iteriere eine Schleife mit Lambda

In diesem Tutorial implementieren Sie ein Entwurfsmuster, das einen Zustandsautomaten und eine AWS Lambda -Funktion verwendet, um eine bestimmte Anzahl von Schleifendurchgängen durchzuführen.

Verwenden Sie dieses Entwurfsmuster stets, wenn Sie die Anzahl der Schleifen in einem Zustandsautomaten nachverfolgen müssen. Diese Implementierung kann dabei helfen, größere Aufgaben oder lange andauernde Ausführungen in kleinere Datenmengen aufzubrechen oder eine Ausführung nach einer bestimmten Anzahl von Ereignissen zu beenden. Sie können eine ähnliche Implementierung verwenden, um eine Ausführung mit langer Laufzeit regelmäßig zu beenden und

neu zu starten, um zu verhindern, dass die Dienstkontingente für AWS Step Functions AWS Lambda, oder andere AWS Dienste überschritten werden.

Bevor Sie beginnen, sollten Sie das Eine Step Functions Functions-Zustandsmaschine erstellen, die Lambda verwendet Tutorial durchgehen, um sicherzustellen, dass Sie mit der gemeinsamen Verwendung von Lambda und Step Functions vertraut sind.

Schritt 1: Erstellen Sie eine Lambda-Funktion, um eine Zählung zu iterieren

Mithilfe einer Lambda-Funktion können Sie die Anzahl der Iterationen einer Schleife in Ihrer Zustandsmaschine verfolgen. Die folgende Lambda-Funktion empfängt Eingabewerte für `countindex`, und`step`. Sie gibt diese Werte mit einem aktualisierten `index` und einem Booleschen Wert namens `continue` zurück. Die Lambda-Funktion wird `continue` auf gesetzt, `true` wenn der kleiner als `index` `count` ist.

Ihr Zustandsautomat implementiert dann einen `Choice`-Zustand, der eine bestimmte Anwendungslogik ausführt, wenn `continue` gleich `true` ist, oder sich beendet, wenn es `false` ist.

So erstellen Sie die Lambda-Funktion:

1. Melden Sie sich bei der Lambda-Konsole an und wählen Sie dann Create function aus.
2. Wählen Sie auf der Seite Create function die Option Author from scratch.
3. Konfigurieren Sie im Abschnitt Grundinformationen Ihre Lambda-Funktion wie folgt:
   a. Geben Sie für Function name (Funktionsname) `Iterator` ein.
   b. Wählen Sie unter Laufzeit die Option Node.js.
   c. Wählen Sie unter Standardausführungsrolle ändern die Option Neue Rolle mit grundlegenden Lambda-Berechtigungen erstellen aus.
   d. Wählen Sie Funktion erstellen.
4. Kopieren Sie den folgenden Code für die Lambda-Funktion in die Codequelle.

```
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
  })
}
```

Dieser Code nimmt Eingabewerte für `count`, `index` und `step` entgegen. Er inkrementiert `index` um den Wert von `step` und gibt diese Werte sowie den Booleschen Wert `continue` zurück. Der Wert von `continue` ist `true`, wenn `index` kleiner als `count` ist.

5. Wählen Sie Bereitstellen.

Schritt 2: Testen Sie die Lambda-Funktion

Führen Sie Ihre Lambda-Funktion mit numerischen Werten aus, um zu sehen, wie sie in Betrieb ist. Sie können Eingabewerte für Ihre Lambda-Funktion angeben, die eine Iteration nachahmen.

Um Ihre Lambda-Funktion zu testen

1. Wählen Sie Test aus.
2. Geben Sie im Dialogfeld „Testereignis konfigurieren“ `TestIterator` in das Feld „Ereignisname“ ein.
3. Ersetzen Sie die Beispieldaten durch Folgendes.

```
{
  "Comment": "Test my Iterator function",
  "iterator": {
    "count": 10,
    "index": 5,
    "step": 1
  }
}
```

Diese Werte ahmen nach, was während einer Iteration aus Ihrem Zustandsautomaten kommt. Die Lambda-Funktion erhöht den Index und kehrt zurück`true`, `continue` wenn der Index kleiner als ist. `count` Für diesen Test wurde der Index bereits auf 5 inkrementiert. Der Test wird auf inkrementiert 6 und `index` auf gesetzt. `continue` `true`

4. Wählen Sie Erstellen.

5. Wählen Sie Test, um Ihre Lambda-Funktion zu testen.

   Die Ergebnisse des Tests werden auf der Registerkarte Ausführungsergebnisse angezeigt.

6. Wählen Sie die Registerkarte Ausführungsergebnisse, um die Ausgabe zu sehen.

```
{
  "index": 6,
  "step": 1,
  "count": 10,
  "continue": true
}
```

> **Note**
>
> Wenn Sie den Wert `index` auf festlegen `9` und erneut testen, werden `index` `false` die Werte auf `10` und `continue` erhöht.

Schritt 3: Erstellen eines Zustandsautomaten

> **Bevor Sie die Lambda-Konsole verlassen...**
>
> Kopieren Sie die Lambda-Funktion ARN. Fügen Sie es in eine Notiz ein. Sie benötigen ihn im nächsten Schritt.

Als Nächstes erstellen Sie eine Zustandsmaschine mit den folgenden Zuständen:

- `ConfigureCount`— Legt Standardwerte für `count`, `index`, und fest`step`.
- `Iterator`— Bezieht sich auf die Lambda-Funktion, die Sie zuvor erstellt haben und die in `ConfigureCount` konfigurierten Werte übergeben.
- `IsCountReached`— Ein Auswahlstatus, der die Schleife fortsetzt oder in den `Done` Status übergeht, basierend auf dem von Ihrer `Iterator` Funktion zurückgegebenen Wert.
- `ExampleWork`— Ein Stummel für Arbeit, die erledigt werden muss. In diesem Beispiel hat der Workflow einen `Pass` Status, aber in einer echten Lösung würden Sie wahrscheinlich einen `Task` verwenden.
- `Done`— Endstatus Ihres Workflows.

Um die Zustandsmaschine in der Konsole zu erstellen:

1. Öffnen Sie die Step Functions Functions-Konsole und wählen Sie dann Create a State Machine aus.

   > **Important**
   >
   > Ihr State Machine muss sich in demselben AWS Konto und derselben Region wie Ihre Lambda-Funktion befinden.

2. Wählen Sie die leere Vorlage aus.
3. Fügen Sie im Codebereich den folgenden JSON-Code ein, der den Zustandsmaschine definiert.

   Weitere Informationen zur Sprache von Amazon States finden Sie unter State Machine Structure.

```
{
    "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. Ersetzen Sie das `Iterator Resource` Feld durch den ARN für Ihre `Iterator` Lambda-Funktion, den Sie zuvor erstellt haben.
5. Wählen Sie Config und geben Sie einen Namen für Ihre Zustandsmaschine ein, z. *`IterateCount`* B.

> **Note**
>
> Die Namen von Zustandsmaschinen, Ausführungen und Aktivitätsaufgaben dürfen nicht länger als 80 Zeichen sein. Diese Namen müssen für Ihr Konto und Ihre AWS Region eindeutig sein und dürfen keine der folgenden Angaben enthalten:
>
> - Leerraum
> - Platzhalterzeichen () ? *
> - Klammerzeichen () < > { } [ ]
> - Sonderzeichen (" # % \ ^ | ~ ` $ & , ; : /)
> - Steuerzeichen (\\u0000- \\u001f oder \\u007f -\\u009f).

Wenn Ihre Zustandsmaschine vom Typ Express ist, können Sie denselben Namen für mehrere Ausführungen der Zustandsmaschine angeben. Step Functions generiert für jede Ausführung von Express State Machine einen eindeutigen Ausführungs-ARN, auch wenn mehrere Ausführungen denselben Namen haben.
Mit Step Functions können Sie Namen für Zustandsmaschinen, Ausführungen und Aktivitäten sowie Beschriftungen erstellen, die Nicht-ASCII-Zeichen enthalten. Diese Nicht-ASCII-Namen funktionieren nicht mit Amazon. CloudWatch Um sicherzustellen, dass Sie CloudWatch Messwerte verfolgen können, wählen Sie einen Namen, der nur ASCII-Zeichen verwendet.

6. Akzeptieren Sie als Typ den Standardwert Standard. Wählen Sie für Berechtigungen die Option Neue Rolle erstellen aus.
7. Wählen Sie Erstellen und dann Bestätigen Sie die Rollenerstellungen.

Schritt 4: Starten einer neuen Ausführung

Nach dem Erstellen des Zustandsautomaten können Sie eine Ausführung starten.

1. Wählen Sie auf der IterateCountSeite Ausführung starten aus.
2. (Optional) Um Ihre Ausführung zu identifizieren, können Sie im Feld Name einen Namen dafür angeben. Standardmäßig generiert Step Functions automatisch einen eindeutigen Ausführungsnamen.

   Note

   Mit Step Functions können Sie Namen für Zustandsmaschinen, Ausführungen und Aktivitäten sowie Beschriftungen erstellen, die Nicht-ASCII-Zeichen enthalten. Diese Nicht-ASCII-Namen funktionieren nicht mit Amazon. CloudWatch Um sicherzustellen, dass Sie CloudWatch Messwerte verfolgen können, wählen Sie einen Namen, der nur ASCII-Zeichen verwendet.

3. Wählen Sie Start Execution aus.

   Eine neue Ausführung Ihres Zustandsautomaten beginnt und zeigt Ihre laufende Ausführung. Ansicht des State-Machine-Diagramms, in dem der Iterator-Status blau angezeigt wird, um den Status „In Bearbeitung“ anzuzeigen.

Die Ausführung wird schrittweise inkrementiert, wobei die Anzahl mithilfe Ihrer Lambda-Funktion verfolgt wird. Bei jeder Iteration wird die Beispielarbeit ausgeführt, auf die im `ExampleWork`-Zustand in Ihrem Zustandsautomaten verwiesen wird.

Wenn der Zähler die im `ConfigureCount`-Zustand in Ihrem Zustandsautomaten festgelegte Anzahl erreicht hat, verlässt die Ausführung den Durchlauf und wird beendet.
Diagrammansicht der Zustandsmaschine, in der der Status „Iterator“ und der Status „Fertig“ grün angezeigt werden, um anzuzeigen, dass beide erfolgreich waren.

Fortsetzung lang andauernder Workflow-Ausführungen als neue Ausführung

AWS Step Functions ist zum Ausführen von Workflows konzipiert, die eine begrenzte Dauer und eine begrenzte Anzahl von Schritten aufweisen. Ausführungen haben eine maximale Dauer von einem Jahr und maximal 25.000 Ereignisse (siehe Kontingente).

Um bei lang andauernden Ausführungen zu vermeiden, dass das feste Kontingent von 25.000 Einträgen im Ausführungsereignisverlauf erreicht wird, empfehlen wir, eine neue Workflow-Ausführung direkt vom `Task` Status einer State-Maschine aus zu starten. Auf diese Weise können Sie Ihre Workflows in kleinere State-Maschinen unterteilen und Ihre laufende Arbeit in einer neuen Ausführung fortsetzen. Um diese Workflow-Ausführungen zu starten, rufen Sie die `StartExecution` API-Aktion von Ihrem `Task` Status aus auf und übergeben Sie die erforderlichen Parameter.

Alternativ können Sie auch ein Muster implementieren, das eine Lambda-Funktion verwendet, um eine neue Ausführung Ihrer State Machine zu starten, um die laufende Arbeit auf mehrere Workflow-Ausführungen aufzuteilen.

Dieses Tutorial zeigt Ihnen beide Methoden, um Workflow-Ausführungen fortzusetzen, ohne die Servicekontingente zu überschreiten.

Themen

- Verwenden einer Step Functions Functions-API-Aktion, um eine neue Ausführung fortzusetzen (empfohlen)
- Verwenden einer Lambda-Funktion, um eine neue Ausführung fortzusetzen

Verwenden einer Step Functions Functions-API-Aktion, um eine neue Ausführung fortzusetzen (empfohlen)

Step Functions kann Workflow-Ausführungen starten, indem es seine eigene API als integrierten Dienst aufruft. Wir empfehlen Ihnen, diesen Ansatz zu verwenden, um zu vermeiden, dass die Servicekontingenten bei Ausführungen mit langer Laufzeit überschritten werden.

Schritt 1: Erstellen Sie einen State Machine mit langer Laufzeit

Erstellen Sie eine Zustandsmaschine mit langer Laufzeit, die Sie vom `Task` Status einer anderen Zustandsmaschine aus starten möchten. Verwenden Sie für dieses Tutorial die Zustandsmaschine, die eine Lambda-Funktion verwendet.

> **Note**
>
> Stellen Sie sicher, dass Sie den Namen und den Amazon-Ressourcennamen dieses Zustandsmaschinen zur späteren Verwendung in eine Textdatei kopieren.

Schritt 2: Erstellen Sie eine Zustandsmaschine, um die Step Functions Functions-API-Aktion aufzurufen

Um Workflow-Ausführungen von einem **`Task`** Status aus zu starten

1. Öffnen Sie die Step Functions Functions-Konsole und wählen Sie Create State Machine.
2. Wählen Sie im Dialogfeld Vorlage auswählen die Option Leer aus.
3. Wählen Sie Select (Auswählen). Dadurch wird Workflow Studio in geöffnetEntwurfsmodus.
4. Ziehen Sie die StartExecutionAPI-Aktion von der Registerkarte Aktionen in den leeren Status mit der Bezeichnung Drag First state here.
5. Wählen Sie den StartExecutionStatus aus und gehen Sie auf der Registerkarte Konfiguration unter wie folgt vorEntwurfsmodus:

    a. Benennen Sie den Status um in**`Start nested execution`**.

    b. Wählen Sie als Integrationstyp die Option AWS SDK — neu aus der Dropdownliste aus.

    c. Gehen Sie unter API-Parameter wie folgt vor:

i. Ersetzen Sie das `StateMachineArn` Beispiel für den Amazon-Ressourcennamen durch den ARN Ihres Zustandsmaschinen. Geben Sie beispielsweise den ARN der Zustandsmaschine ein, die Lambda verwendet.

ii. Ersetzen Sie für `Input` Node den vorhandenen Platzhaltertext durch den folgenden Wert:

```
"Comment": "Starting workflow execution using a Step Functions API action"
```

iii. Stellen Sie sicher, dass Ihre Eingaben in API-Parametern wie folgt aussehen:

```
{
  "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. (Optional) Wählen Sie im Inspector Fenster Definition aus, um die automatisch generierte Amazon States Language (ASL) Definition Ihres Workflows anzuzeigen.

> **Tip**
>
> Sie können die ASL-Definition auch im Workflow Studio anzeigen. Code-Editor Im Code-Editor können Sie auch die ASL-Definition Ihres Workflows bearbeiten.

7. Geben Sie einen Namen für Ihre Zustandsmaschine an. Wählen Sie dazu das Bearbeitungssymbol neben dem Standardnamen der Zustandsmaschine von MyStateMachine. Geben Sie dann unter State-Machine-Konfiguration einen Namen in das Feld State-Machine-Name ein.

   Geben Sie für dieses Tutorial den Namen **`ParentStateMachine`** ein.

8. (Optional) Geben Sie unter State-Machine-Konfiguration weitere Workflow-Einstellungen an, z. B. den Zustandsmaschinentyp und seine Ausführungsrolle.

   Behalten Sie für dieses Tutorial alle Standardauswahlen in den State-Machine-Einstellungen bei.

Wenn Sie zuvor eine IAM-Rolle mit den richtigen Berechtigungen für Ihren Zustandsmaschine erstellt haben und diese verwenden möchten, wählen Sie unter Berechtigungen die Option Vorhandene Rolle auswählen und dann eine Rolle aus der Liste aus. Oder wählen Sie Einen Rollen-ARN eingeben aus und geben Sie dann einen ARN für diese IAM-Rolle ein.

9. Wählen Sie im Dialogfeld „Rollenerstellung bestätigen“ die Option Bestätigen aus, um fortzufahren.

   Sie können auch Rolleneinstellungen anzeigen wählen, um zur State-Machine-Konfiguration zurückzukehren.

   > **Note**
   >
   > Wenn Sie die von Step Functions erstellte IAM-Rolle löschen, kann Step Functions sie später nicht mehr neu erstellen. Ebenso kann Step Functions ihre ursprünglichen Einstellungen später nicht wiederherstellen, wenn Sie die Rolle ändern (z. B. indem Sie Step Functions aus den Principals in der IAM-Richtlinie entfernen).

Schritt 3: Aktualisieren Sie die IAM-Richtlinie

Um sicherzustellen, dass Ihre Zustandsmaschine berechtigt ist, die Ausführung der Zustandsmaschine zu starten, die eine Lambda-Funktion verwendet, müssen Sie der IAM-Rolle Ihrer Zustandsmaschine eine Inline-Richtlinie anhängen. Weitere Informationen finden Sie unter Einbetten von Inline-Richtlinien im IAM-Benutzerhandbuch.

1. Wählen Sie auf der ParentStateMachineSeite die IAM-Rolle ARN aus, um zur Seite mit den IAM-Rollen für Ihren Zustandsmaschine zu navigieren.
2. Weisen Sie der IAM-Rolle von eine entsprechende Berechtigung zu, ParentStateMachinedamit sie die Ausführung einer anderen Zustandsmaschine starten kann. Gehen Sie wie folgt vor, um die Berechtigung zuzuweisen:

   a. Wählen Sie auf der Seite „IAM-Rollen“ die Option Berechtigungen hinzufügen und dann Inline-Richtlinie erstellen aus.

   b. Wählen Sie auf der Seite Richtlinie erstellen die Registerkarte JSON aus.

   c. Ersetzen Sie den vorhandenen Text durch die folgende Richtlinie.

   ```
   {
   ```

```
  "Version": "2012-10-17",
  "Statement": [
      {
          "Effect": "Allow",
          "Action": [
              "states:StartExecution"
          ],
          "Resource": [
              "arn:aws:states:us-
east-2:123456789012:stateMachine:LambdaStateMachine"
          ]
      }
  ]
}
```

d. Wählen Sie Richtlinie prüfen.

e. Geben Sie einen Namen für die Richtlinie an, und wählen Sie dann Richtlinie erstellen aus.

Schritt 4: Führen Sie die Zustandsmaschine aus

State-Machine-Ausführungen sind Instanzen, in denen Sie Ihren Workflow ausführen, um Aufgaben auszuführen.

1. Wählen Sie auf der ParentStateMachineSeite Ausführung starten aus.

   Das Dialogfeld Ausführung starten wird angezeigt.

2. Gehen Sie im Dialogfeld Ausführung starten wie folgt vor:

   a. (Optional) Um Ihre Ausführung zu identifizieren, können Sie im Feld Name einen Namen dafür angeben. Standardmäßig generiert Step Functions automatisch einen eindeutigen Ausführungsnamen.

   > **Note**
   >
   > Mit Step Functions können Sie Namen für Zustandsmaschinen, Ausführungen und Aktivitäten sowie Beschriftungen erstellen, die Nicht-ASCII-Zeichen enthalten. Diese Nicht-ASCII-Namen funktionieren nicht mit Amazon. CloudWatch Um sicherzustellen, dass Sie CloudWatch Messwerte verfolgen können, wählen Sie einen Namen, der nur ASCII-Zeichen verwendet.

b. (Optional) Geben Sie in das Eingabefeld Eingabewerte im JSON-Format ein, um Ihren Workflow auszuführen.

c. Wählen Sie Start execution (Ausführung starten) aus.

d. Die Step Functions Functions-Konsole leitet Sie zu einer Seite weiter, die mit Ihrer Ausführungs-ID betitelt ist. Diese Seite wird als Seite mit den Ausführungsdetails bezeichnet. Auf dieser Seite können Sie die Ausführungsergebnisse im Verlauf der Ausführung oder nach deren Abschluss überprüfen.

   Um die Ausführungsergebnisse zu überprüfen, wählen Sie in der Diagrammansicht einzelne Status aus und wählen Sie dann die einzelnen Registerkarten im Schrittdetails Bereich, um die Details der einzelnen Status, einschließlich Eingabe, Ausgabe und Definition, anzuzeigen. Einzelheiten zu den Ausführungsinformationen, die Sie auf der Seite mit den Ausführungsdetails einsehen können, finden Sie unterSeite mit Ausführungsdetails – Schnittstellenübersicht.

3. Öffnen Sie die LambdaStateMachineSeite und stellen Sie fest, dass eine neue Ausführung durch den ausgelöst wurde ParentStateMachine.

Verwenden einer Lambda-Funktion, um eine neue Ausführung fortzusetzen

Sie können eine Zustandsmaschine erstellen, die eine Lambda-Funktion verwendet, um eine neue Ausführung zu starten, bevor die aktuelle Ausführung beendet wird. Wenn Sie diesen Ansatz verwenden, um Ihre laufende Arbeit in einer neuen Ausführung fortzusetzen, erhalten Sie eine Zustandsmaschine, die große Jobs in kleinere Workflows aufteilen kann, oder eine Zustandsmaschine, die unbegrenzt läuft.

Dieses Tutorial baut auf dem Konzept auf, eine externe Lambda-Funktion zur Änderung Ihres Workflows zu verwenden, das im Iteriere eine Schleife mit Lambda Tutorial demonstriert wurde. Sie verwenden dieselbe Lambda-Funktion (`Iterator`), um eine Schleife für eine bestimmte Anzahl von Malen zu iterieren. Darüber hinaus erstellen Sie eine weitere Lambda-Funktion, um eine neue Ausführung Ihres Workflows zu starten und bei jedem Start einer neuen Ausführung eine Anzahl zu verringern. Indem die Anzahl der Ausführungen in der Eingabe festgelegt wird, beendet dieser Zustandsautomat eine Ausführung mit der angegebenen Häufigkeit und startet sie neu.

Der Zustandsautomat, den Sie erstellen, implementiert die folgenden Zustände.

Status	Zweck
`ConfigureCount`	Ein `Pass` Zustand, der die `step` Werte, und konfiguriert `countindex`, die die `Iterator` Lambda-Funktion verwendet, um schrittweise Arbeitsiterationen zu durchlaufen.
`Iterator`	Ein `Task` Zustand, der auf die `Iterator` Lambda-Funktion verweist.
`IsCountReached`	Ein `Choice` Zustand, der anhand eines booleschen Werts aus der `Iterator` Funktion entscheidet, ob die Zustandsmaschine die Beispielarbeit fortsetzen oder in den Status wechseln soll. `ShouldRestart`
`ExampleWork`	Ein `Pass` Zustand, der den `Task` Status darstellt, der in einer tatsächlichen Implementierung Arbeit verrichten würde.
`ShouldRestart`	Ein `Choice` Zustand, der anhand des `executionCount` Werts entscheidet, ob eine Ausführung beendet und eine andere gestartet oder einfach beendet werden soll.
`Restart`	Ein `Task` Zustand, der eine Lambda-Funktion verwendet, um eine neue Ausführung Ihrer Zustandsmaschine zu starten. Wie die `Iterator`-Funktion dekrementiert auch diese Funktion einen Zähler. Der `Restart` Status übergibt den dekrementierten Wert der Anzahl an die Eingabe der neuen Ausführung.

Voraussetzungen

Bevor Sie beginnen, sollten Sie das Eine Step Functions Functions-Zustandsmaschine erstellen, die Lambda verwendet Tutorial durchgehen, um sicherzustellen, dass Sie mit der gemeinsamen Verwendung von Lambda und Step Functions vertraut sind.

Themen

- Schritt 1: Erstellen Sie eine Lambda-Funktion, um eine Zählung zu iterieren
- Schritt 2: Erstellen Sie eine Lambda-Funktion neu starten, um eine neue Step Functions Functions-Ausführung zu starten

- Schritt 3: Erstellen Sie eine Zustandsmaschine
- Schritt 4: Aktualisieren Sie die IAM-Richtlinie
- Schritt 5: Führen Sie die Zustandsmaschine aus

Schritt 1: Erstellen Sie eine Lambda-Funktion, um eine Zählung zu iterieren

> **Note**
>
> Wenn Sie das Iteriere eine Schleife mit Lambda Tutorial abgeschlossen haben, können Sie diesen Schritt überspringen und diese Lambda-Funktion verwenden.

Dieser Abschnitt und das Iteriere eine Schleife mit Lambda Tutorial zeigen, wie Sie eine Lambda-Funktion verwenden können, um eine Anzahl zu verfolgen, z. B. die Anzahl der Iterationen einer Schleife in Ihrer Zustandsmaschine.

Die folgende Lambda-Funktion empfängt Eingabewerte für `count`index, und`step`. Sie gibt diese Werte mit einem aktualisierten `index` und einem Booleschen Wert namens `continue` zurück. Die Lambda-Funktion wird `continue` auf gesetzt, `true` wenn der kleiner als `index` `count` ist.

Ihr Zustandsautomat implementiert dann einen `Choice`-Zustand, der eine bestimmte Anwendungslogik ausführt, wenn `true` für `continue` angegeben ist, oder mit `ShouldRestart` fortfährt, wenn `false` für `continue` angegeben ist.

Erstellen Sie die Funktion Iterate Lambda

1. Öffnen Sie die Lambda-Konsole und wählen Sie Create function (Funktion erstellen) aus.
2. Wählen Sie auf der Seite Create function die Option Author from scratch.
3. Konfigurieren Sie im Abschnitt Grundinformationen Ihre Lambda-Funktion wie folgt:

    a. Geben Sie für Function name (Funktionsname) `Iterator` ein.

    b. Wählen Sie unter Laufzeit die Option Node.js 16.x aus.

    c. Behalten Sie alle Standardauswahlen auf der Seite bei und wählen Sie dann Funktion erstellen.

    Wenn Ihre Lambda-Funktion erstellt wurde, notieren Sie sich ihren Amazon-Ressourcennamen (ARN) in der oberen rechten Ecke der Seite, zum Beispiel:

```
arn:aws:lambda:us-east-1:123456789012:function:Iterator
```

4. Kopieren Sie den folgenden Code für die Lambda-Funktion in den Abschnitt Codequelle der ***Iterator-Seite*** in der Lambda-Konsole.

```
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
  })
}
```

   Dieser Code nimmt Eingabewerte für `count`, `index` und `step` entgegen. Er inkrementiert `index` um den Wert von `step` und gibt diese Werte sowie den Booleschen Wert `continue` zurück. Der Wert von `continue` ist `true`, wenn `index` kleiner als `count` ist.

5. Wählen Sie Deploy, um den Code bereitzustellen.

Testen Sie die Funktion Iterate Lambda

Um zu sehen, wie Ihre `Iterate`-Funktion arbeitet, führen Sie sie mit numerischen Werten aus. Sie können Eingabewerte für Ihre Lambda-Funktion angeben, die eine Iteration nachahmen, um zu sehen, welche Ausgabe Sie mit bestimmten Eingabewerten erhalten.

Um Ihre Lambda-Funktion zu testen

1. Wählen Sie im Dialogfeld Configure test event die Option Create new test event und geben Sie dann `TestIterator` für Event name ein.
2. Ersetzen Sie die Beispieldaten durch Folgendes.

```
{
  "Comment": "Test my Iterator function",
```

```
  "iterator": {
    "count": 10,
    "index": 5,
    "step": 1
  }
}
```

Diese Werte ahmen nach, was während einer Iteration aus Ihrem Zustandsautomaten kommt. Die Lambda-Funktion erhöht den Index und gibt als zurück`continue`. `true` Wenn der Index nicht geringer ist als `count`, wird `continue` als `false` zurückgegeben. Für diesen Test wurde der Index bereits auf 5 inkrementiert. Die Ergebnisse sollten `index` auf 6 inkrementieren und `continue` auf `true` setzen.

3. Wählen Sie Erstellen.
4. Vergewissern Sie sich, dass auf der ***`Iterator-Seite`*** in Ihrer Lambda-Konsole aufgeführt TestIteratorist, und wählen Sie dann Test aus.

   Die Ergebnisse des Tests werden oben auf der Seite angezeigt. Wählen Sie Details und sehen Sie sich das Ergebnis an.

```
{
  "index": 6,
  "step": 1,
  "count": 10,
  "continue": true
}
```

> **Note**
>
> Wenn Sie `index` für diesen Test auf 9 setzen, wird `index` auf `10` inkrementiert und `continue` ist `false`.

Schritt 2: Erstellen Sie eine Lambda-Funktion neu starten, um eine neue Step Functions Functions-Ausführung zu starten

1. Öffnen Sie die Lambda-Konsole und wählen Sie Create function (Funktion erstellen) aus.
2. Wählen Sie auf der Seite Create function die Option Author from scratch.
3. Konfigurieren Sie im Abschnitt Grundinformationen Ihre Lambda-Funktion wie folgt:

a. Geben Sie für Function name (Funktionsname) `Restart` ein.

b. Wählen Sie unter Laufzeit die Option Node.js 16.x aus.

4. Behalten Sie alle Standardauswahlen auf der Seite bei und wählen Sie dann Funktion erstellen.

   Wenn Ihre Lambda-Funktion erstellt wurde, notieren Sie sich ihren Amazon-Ressourcennamen (ARN) in der oberen rechten Ecke der Seite, zum Beispiel:

   ```
   arn:aws:lambda:us-east-1:123456789012:function:Iterator
   ```

5. Kopieren Sie den folgenden Code für die Lambda-Funktion in den Abschnitt Codequelle der ***Neustart-Seite*** in der Lambda-Konsole.

   Der folgende Code dekrementiert einen Zähler für die Ausführungsanzahl und startet eine neue Ausführung Ihres Zustandsautomaten, einschließlich des dekrementierten Wertes.

   ```
   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. Wählen Sie Deploy, um den Code bereitzustellen.

Schritt 3: Erstellen Sie eine Zustandsmaschine

Nachdem Sie Ihre beiden Lambda-Funktionen erstellt haben, erstellen Sie eine Zustandsmaschine. In diesem Zustandsautomaten benennen die Zustände `ShouldRestart` und `Restart` die Art und Weise, wie Sie Ihre Arbeit auf mehrere Ausführungen aufteilen.

Example ShouldRestart Status „Wahl“

Der folgende Auszug zeigt den `ShouldRestart` `Choice` Bundesstaat. Dieser Status bestimmt, ob Sie die Ausführung neu starten sollten oder nicht.

```
"ShouldRestart": {
"Type": "Choice",
"Choices": [
  {
    "Variable": "$.restart.executionCount",
    "NumericGreaterThan": 1,
    "Next": "Restart"
  }
],
```

Der `$.restart.executionCount`-Wert ist in der Eingabe der ersten Ausführung enthalten. Er wird bei jedem Aufruf der `Restart`-Funktion um eins dekrementiert und dann für jede nachfolgende Ausführung in die Eingabe platziert.

Example Task-Zustand Restart

Der folgende Auszug zeigt den `Restart` `Task` Status. Dieser Status verwendet die Lambda-Funktion, die Sie zuvor erstellt haben, um die Ausführung neu zu starten und die Anzahl zu verringern, um die verbleibende Anzahl der zu startenden Ausführungen zu verfolgen.

```
"Restart": {
  "Type": "Task",
  "Resource": "arn:aws:lambda:us-east-1:123456789012:function:Restart",
  "Next": "Done"
},
```

So erstellen Sie den -Zustandsautomaten

1. Öffnen Sie die Step Functions Functions-Konsole und wählen Sie Create State Machine.

> **Important**
>
> Stellen Sie sicher, dass sich Ihr State Machine unter demselben AWS Konto und derselben Region befindet wie die Lambda-Funktionen, die Sie zuvor in Schritt 1 und Schritt 2 erstellt haben.

2. Wählen Sie im Dialogfeld Vorlage auswählen die Option Leer aus.
3. Wählen Sie Select (Auswählen). Dadurch wird Workflow Studio in geöffnetEntwurfsmodus.
4. Für dieses Tutorial schreiben Sie die Amazon States Language (ASL-) Definition Ihrer Zustandsmaschine in denCode-Editor. Wählen Sie dazu Code.
5. Entfernen Sie den vorhandenen Boilerplate-Code und fügen Sie den folgenden Code ein. Denken Sie daran, die ARNs in diesem Code durch die ARNs der von Ihnen erstellten Lambda-Funktionen zu ersetzen.

```
{
    "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. Geben Sie einen Namen für Ihre Zustandsmaschine an. Wählen Sie dazu das Bearbeitungssymbol neben dem Standardnamen der Zustandsmaschine von MyStateMachine. Geben Sie dann unter State-Machine-Konfiguration einen Namen in das Feld State-Machine-Name ein.

   Geben Sie für dieses Tutorial den Namen **`ContinueAsNew`** ein.

7. (Optional) Geben Sie unter State-Machine-Konfiguration weitere Workflow-Einstellungen an, z. B. den Zustandsmaschinentyp und seine Ausführungsrolle.

   Behalten Sie für dieses Tutorial alle Standardauswahlen in den State-Machine-Einstellungen bei.

   Wenn Sie zuvor eine IAM-Rolle mit den richtigen Berechtigungen für Ihren Zustandsmaschine erstellt haben und diese verwenden möchten, wählen Sie unter Berechtigungen die Option Vorhandene Rolle auswählen und dann eine Rolle aus der Liste aus. Oder wählen Sie Einen Rollen-ARN eingeben aus und geben Sie dann einen ARN für diese IAM-Rolle ein.

8. Wählen Sie im Dialogfeld „Rollenerstellung bestätigen“ die Option Bestätigen aus, um fortzufahren.

   Sie können auch Rolleneinstellungen anzeigen wählen, um zur State-Machine-Konfiguration zurückzukehren.

   > **Note**
   >
   > Wenn Sie die von Step Functions erstellte IAM-Rolle löschen, kann Step Functions sie später nicht mehr neu erstellen. Ebenso kann Step Functions ihre ursprünglichen Einstellungen später nicht wiederherstellen, wenn Sie die Rolle ändern (z. B. indem Sie Step Functions aus den Principals in der IAM-Richtlinie entfernen).

9. Speichern Sie den Amazon-Ressourcennamen (ARN) dieser Zustandsmaschine in einer Textdatei. Sie müssen den ARN angeben und gleichzeitig der Lambda-Funktion die Erlaubnis erteilen, eine neue Step Functions Functions-Ausführung zu starten.

Schritt 4: Aktualisieren Sie die IAM-Richtlinie

Um sicherzustellen, dass Ihre Lambda-Funktion berechtigt ist, eine neue Step Functions Functions-Ausführung zu starten, fügen Sie der IAM-Rolle, die Sie für Ihre `Restart` Lambda-Funktion verwenden, eine Inline-Richtlinie hinzu. Weitere Informationen finden Sie unter Einbetten von Inline-Richtlinien im IAM-Benutzerhandbuch.

> **Note**
>
> Sie können die `Resource`-Zeile im vorherigen Beispiel aktualisieren, damit sie auf den ARN Ihres `ContinueAsNew`-Zustandsautomaten verweist. Dies beschränkt die Richtlinie, so dass sie nur eine Ausführung dieses spezifischen Zustandsautomaten starten kann.

```
{
 "Version": "2012-10-17",
 "Statement": [
     {
         "Sid": "VisualEditor0",
         "Effect": "Allow",
         "Action": [
             "states:StartExecution"
         ],
         "Resource": "arn:aws:states:us-east-2:123456789012stateMachine:ContinueAsNew"
     }
 ]
}
```

Schritt 5: Führen Sie die Zustandsmaschine aus

Zum Starten einer Ausführung stellen Sie eine Eingabe bereit, die den ARN des Zustandsautomaten enthält und einen `executionCount`, der angibt, wie oft eine neue Ausführung gestartet werden soll.

1. Wählen Sie auf der ContinueAsNewSeite Ausführung starten aus.

   Das Dialogfeld Ausführung starten wird angezeigt.
2. Gehen Sie im Dialogfeld Ausführung starten wie folgt vor:

   a. (Optional) Um Ihre Ausführung zu identifizieren, können Sie im Feld Name einen Namen dafür angeben. Standardmäßig generiert Step Functions automatisch einen eindeutigen Ausführungsnamen.

   > **Note**
   >
   > Mit Step Functions können Sie Namen für Zustandsmaschinen, Ausführungen und Aktivitäten sowie Beschriftungen erstellen, die Nicht-ASCII-Zeichen enthalten. Diese Nicht-ASCII-Namen funktionieren nicht mit Amazon. CloudWatch Um

sicherzustellen, dass Sie CloudWatch Messwerte verfolgen können, wählen Sie einen Namen, der nur ASCII-Zeichen verwendet.

b. Geben Sie im Eingabefeld die folgende JSON-Eingabe ein, um Ihren Workflow auszuführen.

```
{
  "restart": {
    "StateMachineArn": "arn:aws:states:us-
east-1:123456789012:stateMachine:ContinueAsNew",
    "executionCount": 4
  }
}
```

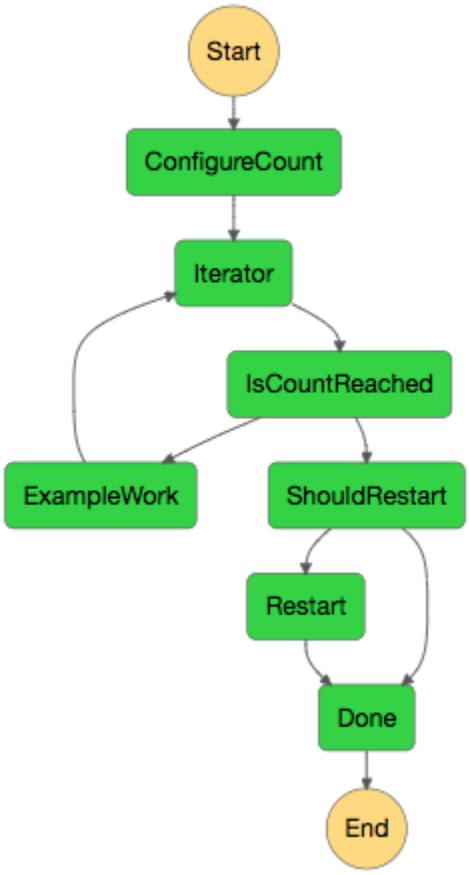
c. Aktualisieren Sie das Feld `StateMachineArn` mit dem ARN für den Zustandsautomaten `ContinueAsNew`.

d. Wählen Sie Start execution (Ausführung starten) aus.

e. Die Step Functions Functions-Konsole leitet Sie zu einer Seite weiter, die mit Ihrer Ausführungs-ID betitelt ist. Diese Seite wird als Seite mit den Ausführungsdetails bezeichnet. Auf dieser Seite können Sie die Ausführungsergebnisse im Verlauf oder nach Abschluss der Ausführung überprüfen.

   Um die Ausführungsergebnisse zu überprüfen, wählen Sie in der Diagrammansicht einzelne Status und dann die einzelnen Registerkarten im Schrittdetails Bereich aus, um die Details der einzelnen Status, einschließlich Eingabe, Ausgabe und Definition, anzuzeigen. Einzelheiten zu den Ausführungsinformationen, die Sie auf der Seite mit den Ausführungsdetails einsehen können, finden Sie unterSeite mit Ausführungsdetails – Schnittstellenübersicht.

   In der Diagrammansicht wird die erste der vier Ausführungen angezeigt. Bevor sie abgeschlossen ist, durchläuft sie den `Restart`-Zustand und startet eine neue Ausführung.



Wenn diese Ausführung abgeschlossen ist, können Sie sich die nächste Ausführung ansehen, die gerade ausgeführt wird. Wählen Sie den ContinueAsNewLink oben aus, um die Liste der Ausführungen zu sehen. Sie sollten sowohl die kürzlich abgeschlossene Ausführung als auch eine laufende Ausführung sehen, die die `Restart` Lambda-Funktion gestartet hat.

Wenn alle Ausführungen abgeschlossen sind, sollten Sie vier erfolgreiche Ausführungen in der Liste sehen. Die erste gestartete Ausführung zeigt den von Ihnen gewählten Namen. Nachfolgende Ausführungen haben einen generierten Namen.

Bereitstellen eines Beispielprojekts für Genehmigungen durch Menschen

In diesem Tutorial erfahren Sie, wie Sie ein Projekt mit Genehmigungen durch Menschen bereitstellen, das das Anhalten einer AWS Step Functions -Ausführung während einer Aufgabe und das Warten auf die E-Mail-Antwort eines Benutzers ermöglicht. Der Workflow wird mit der nächsten Phase fortgesetzt, sobald der Benutzer die Aufgabe genehmigt hat.

Durch die Bereitstellung des in diesem Tutorial enthaltenen AWS CloudFormation Stacks werden alle erforderlichen Ressourcen bereitgestellt, darunter:

- Ressourcen für Amazon API Gateway
- Und AWS Lambda Funktionen
- Eine AWS Step Functions Staatsmaschine
- Ein E-Mail-Thema von Amazon Simple Notification Service
- Verwandte AWS Identity and Access Management Rollen und Berechtigungen

> **Note**
>
> Sie müssen eine gültige E-Mail-Adresse angeben, auf die Sie Zugriff haben, wenn Sie den AWS CloudFormation Stack erstellen.

Weitere Informationen finden Sie unter Arbeiten mit CloudFormation Vorlagen und in der `AWS::StepFunctions::StateMachine` Ressource im AWS CloudFormation Benutzerhandbuch.

Themen

Schritt 1: Erstellen Sie eine AWS CloudFormation Vorlage

1. Kopieren Sie den Beispielcode aus dem Abschnitt AWS CloudFormation Quellcode der Vorlage.

2. Fügen Sie die Quelle der AWS CloudFormation Vorlage in eine Datei auf Ihrem lokalen Computer ein.

   In diesem Beispiel hat die Datei den Namen `human-approval.yaml`.

Schritt 2: Erstellen Sie einen Stapel

1. Melden Sie sich an der AWS CloudFormation -Konsole an.
2. Wählen Sie Stack erstellen und dann Mit neuen Ressourcen (Standard).
3. Gehen Sie auf der Seite Create stack (Stack erstellen) wie folgt vor:

   a. Vergewissern Sie sich, dass im Abschnitt Voraussetzung — Vorlage vorbereiten die Option Vorlage ist bereit ausgewählt ist.

   b. Wählen Sie im Abschnitt Vorlage angeben die Option Vorlagendatei hochladen und dann Datei auswählen aus, um die zuvor erstellte `human-approval.yaml` Datei hochzuladen, die den Quellcode der Vorlage enthält.
4. Wählen Sie Weiter aus.
5. Führen Sie auf der Seite Specify DB Details (Festlegen von DB-Detail) die folgenden Schritte aus:

a. Geben Sie unter Stackname einen Namen für Ihren Stack ein.

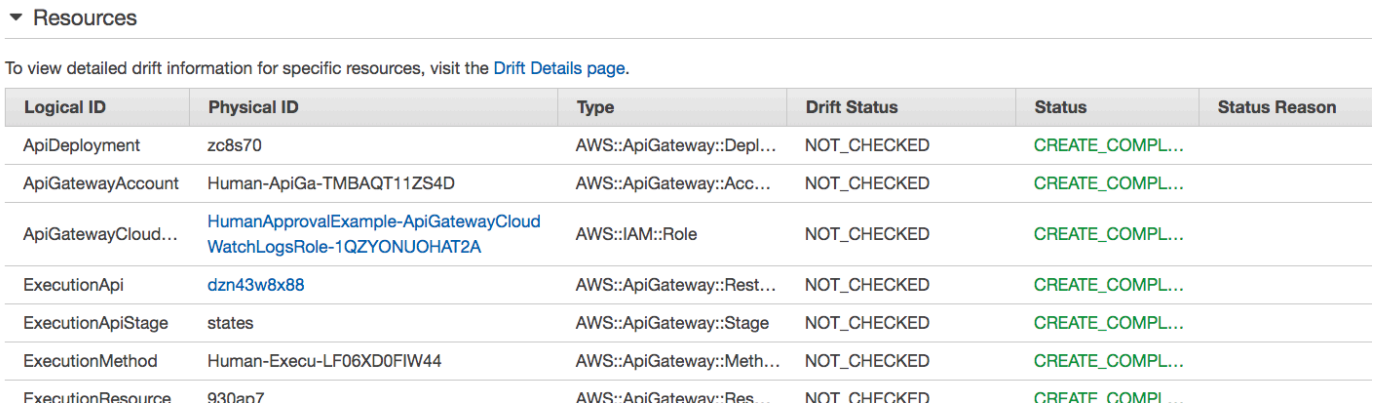
b. Geben Sie unter Parameter eine gültige E-Mail-Adresse ein. Sie verwenden diese E-Mail-Adresse, um das Amazon SNS SNS-Thema zu abonnieren.

6. Wählen Sie Weiter und dann erneut Weiter.

7. Wählen Sie auf der Seite Überprüfen die Option Ich bestätige, dass AWS CloudFormation möglicherweise IAM-Ressourcen erstellt werden, und wählen Sie dann Erstellen aus.

   AWS CloudFormation beginnt mit der Erstellung Ihres Stacks und zeigt den Status CREATE_IN_PROGRESS an. Wenn der Vorgang abgeschlossen ist, wird der Status CREATE_COMPLETE AWS CloudFormation angezeigt.

8. (Optional) Wählen Sie zum Anzeigen der Ressourcen in Ihrem Stack den Stack und anschließend die Registerkarte Resources aus.



Schritt 3: Genehmigen Sie das Amazon SNS SNS-Abonnement

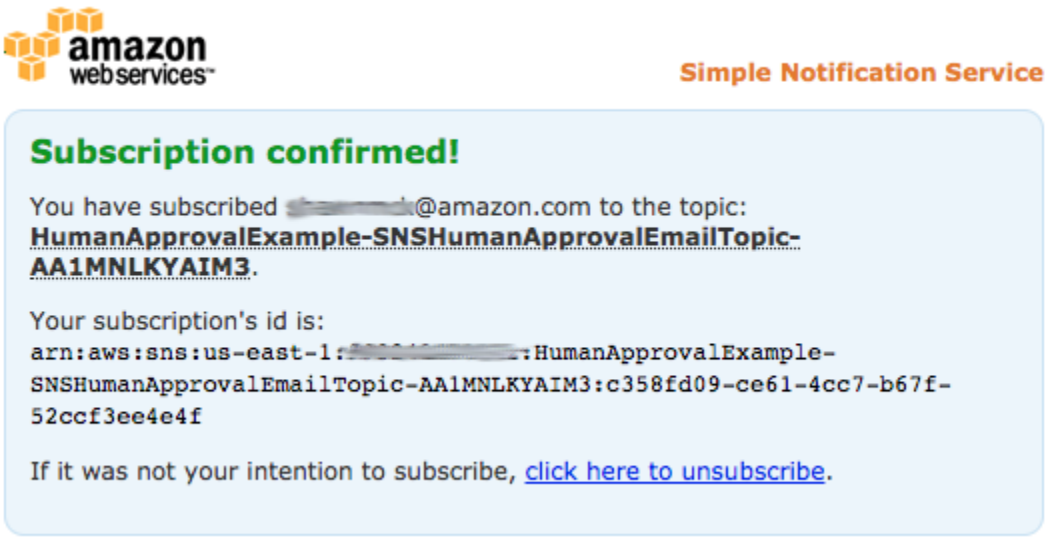
Sobald das Amazon SNS SNS-Thema erstellt wurde, erhalten Sie eine E-Mail mit der Aufforderung, das Abonnement zu bestätigen.

1. Öffnen Sie das E-Mail-Konto, das Sie bei der Erstellung des AWS CloudFormation Stacks angegeben haben.

2. Öffnen Sie die Nachricht AWS Benachrichtigung — Abonnementbestätigung von no-reply@sns.amazonaws.com

   In der E-Mail werden der Amazon-Ressourcenname für das Amazon SNS-Thema sowie ein Bestätigungslink aufgeführt.

3. Wählen Sie den Link confirm subscription (Abonnement bestätigen) aus.



Schritt 4: Führen Sie die Zustandsmaschine aus

1. Wählen Sie auf der HumanApprovalLambdaStateMachineSeite Ausführung starten aus.

   Das Dialogfeld Ausführung starten wird angezeigt.
2. Gehen Sie im Dialogfeld Ausführung starten wie folgt vor:

   a. (Optional) Um Ihre Ausführung zu identifizieren, können Sie im Feld Name einen Namen dafür angeben. Standardmäßig generiert Step Functions automatisch einen eindeutigen Ausführungsnamen.

   > **Note**
   >
   > Mit Step Functions können Sie Namen für Zustandsmaschinen, Ausführungen und Aktivitäten sowie Labels erstellen, die Nicht-ASCII-Zeichen enthalten. Diese Nicht-ASCII-Namen funktionieren nicht mit Amazon. CloudWatch Um sicherzustellen, dass Sie CloudWatch Messwerte verfolgen können, wählen Sie einen Namen, der nur ASCII-Zeichen verwendet.

   b. Geben Sie im Eingabefeld die folgende JSON-Eingabe ein, um Ihren Workflow auszuführen.

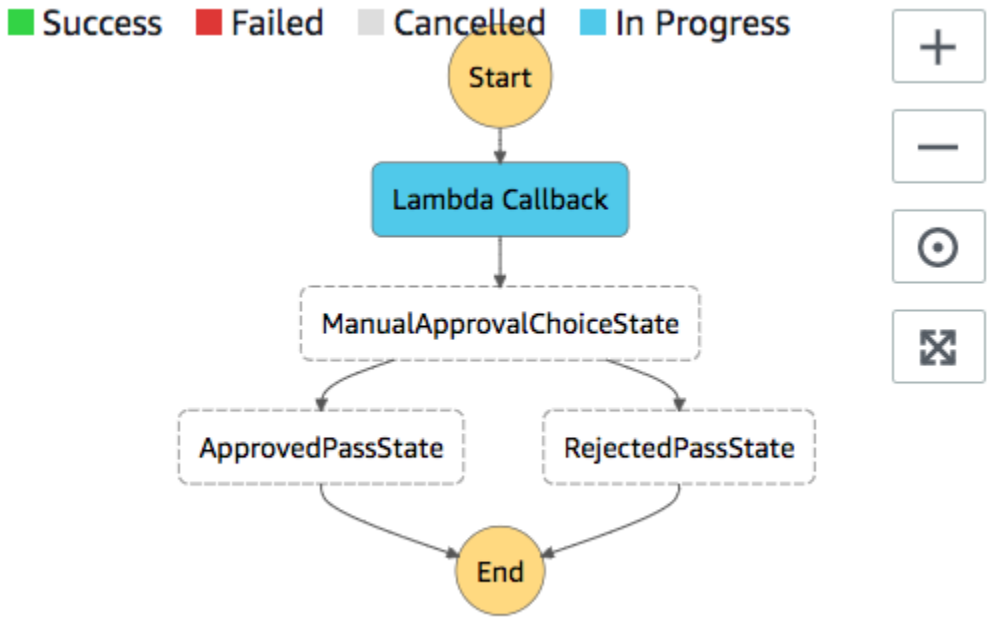
```
{
    "Comment": "Testing the human approval tutorial."
}
```

c. Wählen Sie Start execution (Ausführung starten) aus.

Die Ausführung der ApprovalTestZustandsmaschine wird bei der Lambda-Callback-Aufgabe gestartet und angehalten.

d. Die Step Functions Functions-Konsole leitet Sie zu einer Seite weiter, die mit Ihrer Ausführungs-ID betitelt ist. Diese Seite wird als Seite mit den Ausführungsdetails bezeichnet. Auf dieser Seite können Sie die Ausführungsergebnisse im Verlauf oder nach Abschluss der Ausführung überprüfen.

Um die Ausführungsergebnisse zu überprüfen, wählen Sie in der Diagrammansicht einzelne Status und dann die einzelnen Registerkarten im Schrittdetails Bereich aus, um die Details der einzelnen Status, einschließlich Eingabe, Ausgabe und Definition, anzuzeigen. Einzelheiten zu den Ausführungsinformationen, die Sie auf der Seite mit den Ausführungsdetails einsehen können, finden Sie unterSeite mit Ausführungsdetails – Schnittstellenübersicht.

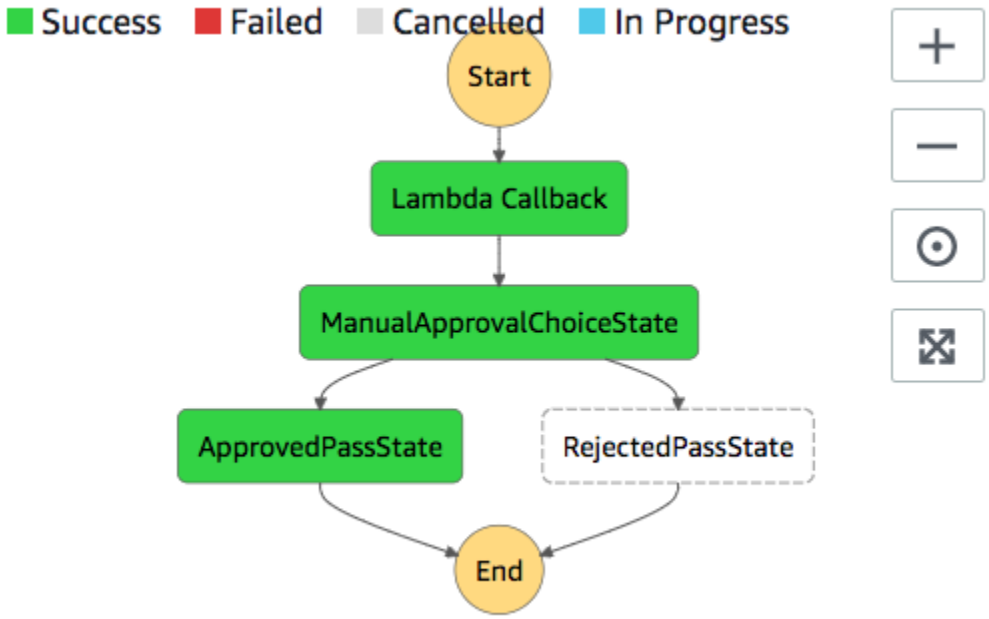


3. Öffnen Sie in dem E-Mail-Konto, das Sie zuvor für das Amazon SNS SNS-Thema verwendet haben, die Nachricht mit dem Betreff Genehmigung erforderlich von AWS Step Functions.

Die Nachricht enthält getrennte URLs für Approve (Genehmigen) und Reject (Ablehnen).

4. Wählen Sie die URL für Approve (Genehmigen) aus.

Der Workflow wird abhängig von Ihrer Wahl fortgesetzt.



AWS CloudFormation Quellcode der Vorlage

Verwenden Sie diese AWS CloudFormation Vorlage, um ein Beispiel für einen menschlichen Genehmigungsprozess bereitzustellen.

```
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
```

Röntgenspuren in Step Functions anzeigen

In diesem Tutorial erfahren Sie, wie Sie mit X-Ray Fehler verfolgen können, die beim Ausführen einer Zustandsmaschine auftreten. Sie können AWS X-Raydamit die Komponenten Ihrer Zustandsmaschine visualisieren, Leistungsengpässe identifizieren und Anfragen, die zu einem Fehler geführt haben, beheben. In diesem Tutorial erstellen Sie mehrere Lambda-Funktionen, die nach dem Zufallsprinzip Fehler erzeugen, die Sie dann mit X-Ray verfolgen und analysieren können.

Das Eine Step Functions Functions-Zustandsmaschine erstellen, die Lambda verwendet Tutorial führt Sie durch die Erstellung einer Zustandsmaschine, die eine Lambda-Funktion aufruft. Wenn Sie dieses Tutorial abgeschlossen haben, fahren Sie mit Schritt 2 fort und verwenden Sie die AWS Identity and Access Management (IAM-) Rolle, die Sie zuvor erstellt haben.

Themen

- Schritt 1: Erstellen Sie eine IAM-Rolle für Lambda
- Schritt 2: Erstellen einer Lambda-Funktion
- Schritt 3: Erstellen Sie zwei weitere Lambda-Funktionen
- Schritt 4: Erstellen Sie eine Zustandsmaschine
- Schritt 5: Führen Sie die Zustandsmaschine aus

Schritt 1: Erstellen Sie eine IAM-Rolle für Lambda

Beides AWS Lambda und AWS Step Functions kann Code ausführen und auf AWS Ressourcen zugreifen (z. B. auf Daten, die in Amazon S3 S3-Buckets gespeichert sind). Um die Sicherheit zu gewährleisten, müssen Sie Lambda und Step Functions Zugriff auf diese Ressourcen gewähren.

Lambda verlangt, dass Sie beim Erstellen einer Lambda-Funktion eine AWS Identity and Access Management (IAM-) Rolle zuweisen, genauso wie Step Functions verlangt, dass Sie beim Erstellen einer Zustandsmaschine eine IAM-Rolle zuweisen.

Sie verwenden die IAM-Konsole, um eine serviceverknüpfte Rolle zu erstellen.

So erstellen Sie eine Rolle (Konsole)

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie im Navigationsbereich der IAM Console Roles. Wählen Sie dann Create Role.
3. Wählen Sie den Rollentyp AWS Service und dann Lambda aus.
4. Wählen Sie den Lambda-Anwendungsfall. Anwendungsfälle werden durch den Service definiert, damit die für den Service erforderliche Vertrauensrichtlinie enthalten ist. Wählen Sie dann Next: Permissions.
5. Wählen Sie eine oder mehrere Berechtigungsrichtlinien aus, die Sie an die Rolle anfügen möchten (z. B. `AWSLambdaBasicExecutionRole`). Siehe AWS Lambda - Berechtigungsmodell.

   Aktivieren Sie das Kontrollkästchen neben der Richtlinie, die die Berechtigungen zuweist, die der Rolle gewährt werden sollen. Wählen Sie dann Next: Review aus.
6. Geben Sie einen Role name ein.
7. (Optional:) Bearbeiten Sie in Role description die Beschreibung für die neue serviceverknüpfte Rolle.
8. Prüfen Sie die Rolle und klicken Sie dann auf Create Role (Rolle erstellen).

Schritt 2: Erstellen einer Lambda-Funktion

Ihre Lambda-Funktion gibt nach dem Zufallsprinzip Fehler oder Timeout aus und erzeugt Beispieldaten, die in X-Ray angezeigt werden können.

> **Important**
>
> Stellen Sie sicher, dass sich Ihre Lambda-Funktion unter demselben AWS Konto und derselben AWS Region wie Ihr State Machine befindet.

1. Öffnen Sie die Lambda-Konsole und wählen Sie Create function.
2. Wählen Sie im Abschnitt Create function die Option Author from scratch.
3. Konfigurieren Sie im Abschnitt Grundinformationen Ihre Lambda-Funktion:

   a. Geben Sie für Function name (Funktionsname) `TestFunction1` ein.

   b. Wählen Sie unter Laufzeit die Option Node.js 18.x aus.

   c. Wählen Sie für Role (Rolle) die Option Choose an existing role (Eine vorhandene Rolle wählen) aus.

   d. Wählen Sie unter Existing role die Lambda-Rolle aus, die Sie zuvor erstellt haben.

   > **Note**
   >
   > Wenn die von Ihnen erstellte IAM-Rolle nicht in der Liste erscheint, dauert es möglicherweise dennoch einige Minuten, bis die Rolle an Lambda weitergegeben wird.

   e. Wählen Sie Funktion erstellen.

   Wenn Ihre Lambda-Funktion erstellt wird, notieren Sie sich ihren Amazon-Ressourcennamen (ARN) in der oberen rechten Ecke der Seite. Beispielsweise:

   ```
   arn:aws:lambda:us-east-1:123456789012:function:TestFunction1
   ```

4. Kopieren Sie den folgenden Code für die Lambda-Funktion in den Abschnitt Funktionscode der ***TestFunction1-Seite***.

```
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 }
};
```

Dieser Code erzeugt nach dem Zufallsprinzip zeitlich begrenzte Fehler, die verwendet werden, um Beispielfehler in Ihrer Zustandsmaschine zu generieren, die mithilfe von Röntgenspuren angezeigt und analysiert werden können.

5. Wählen Sie Speichern.

Schritt 3: Erstellen Sie zwei weitere Lambda-Funktionen

Erstellen Sie zwei weitere Lambda-Funktionen.

1. Wiederholen Sie Schritt 2, um zwei weitere Lambda-Funktionen zu erstellen. Geben Sie für die nächste Funktion im Feld **`TestFunction2`**Funktionsname den Wert ein. Geben Sie für die letzte Funktion im Feld Funktionsname den Wert ein`TestFunction3`.
2. Vergewissern Sie sich in der Lambda-Konsole, dass Sie jetzt über drei Lambda-Funktionen verfügen: `TestFunction1``TestFunction2`, und. `TestFunction3`

Schritt 4: Erstellen Sie eine Zustandsmaschine

In diesem Schritt verwenden Sie die Step Functions-Konsole, um eine Zustandsmaschine mit drei `Task` Zuständen zu erstellen. Jeder `Task` Status wird auf eine Ihrer drei Lambda-Funktionen verweisen.

1. Öffnen Sie die Step Functions Functions-Konsole und wählen Sie Create State Machine.

> **Important**
>
> Stellen Sie sicher, dass sich Ihr State Machine unter demselben AWS Konto und derselben Region befindet wie die Lambda-Funktionen, die Sie zuvor in Schritt 2 und Schritt 3 erstellt haben.

2. Wählen Sie im Dialogfeld Vorlage auswählen die Option Leer aus.
3. Wählen Sie Select (Auswählen). Dadurch wird Workflow Studio in geöffnetEntwurfsmodus.
4. Für dieses Tutorial schreiben Sie die Amazon States Language (ASL-) Definition Ihrer Zustandsmaschine in denCode-Editor. Wählen Sie dazu Code.
5. Entfernen Sie den vorhandenen Boilerplate-Code und fügen Sie den folgenden Code ein. Denken Sie daran, in der Aufgabenstatusdefinition die Beispiel-ARNs durch die ARNs der Lambda-Funktionen zu ersetzen, die Sie erstellt haben.

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

Dies ist eine Beschreibung Ihrer Zustandsmaschine in der Sprache Amazon States. Es definiert drei `Task` Staaten mit dem Namen`CallTestFunction1`, `CallTestFunction2` und`CallTestFunction3`. Jede Funktion ruft eine Ihrer drei Lambda-Funktionen auf. Weitere Informationen finden Sie unter State Machine Structure.

6. Geben Sie einen Namen für Ihre Zustandsmaschine an. Wählen Sie dazu das Bearbeitungssymbol neben dem Standardnamen der Zustandsmaschine von MyStateMachine. Geben Sie dann unter State-Machine-Konfiguration einen Namen in das Feld State-Machine-Name ein.

   Geben Sie für dieses Tutorial den Namen **`TraceFunctions`** ein.

7. (Optional) Geben Sie unter State-Machine-Konfiguration weitere Workflow-Einstellungen an, z. B. den Zustandsmaschinentyp und seine Ausführungsrolle.

Wählen Sie für dieses Tutorial unter Zusätzliche Konfiguration die Option X-Ray-Tracing aktivieren aus. Behalten Sie alle anderen Standardauswahlen in den State-Machine-Einstellungen bei.

Wenn Sie zuvor eine IAM-Rolle mit den richtigen Berechtigungen für Ihren Zustandsmaschine erstellt haben und diese verwenden möchten, wählen Sie unter Berechtigungen die Option Vorhandene Rolle auswählen und dann eine Rolle aus der Liste aus. Oder wählen Sie Einen Rollen-ARN eingeben aus und geben Sie dann einen ARN für diese IAM-Rolle ein.

8. Wählen Sie im Dialogfeld „Rollenerstellung bestätigen“ die Option Bestätigen aus, um fortzufahren.

   Sie können auch Rolleneinstellungen anzeigen wählen, um zur State-Machine-Konfiguration zurückzukehren.

   > **Note**
   >
   > Wenn Sie die von Step Functions erstellte IAM-Rolle löschen, kann Step Functions sie später nicht mehr neu erstellen. Ebenso kann Step Functions ihre ursprünglichen Einstellungen später nicht wiederherstellen, wenn Sie die Rolle ändern (z. B. indem Sie Step Functions aus den Principals in der IAM-Richtlinie entfernen).

Schritt 5: Führen Sie die Zustandsmaschine aus

State-Machine-Ausführungen sind Instanzen, in denen Sie Ihren Workflow ausführen, um Aufgaben auszuführen.

1. Wählen Sie auf der ***TraceFunctions***Seite Ausführung starten aus.

   Die Seite New execution wird angezeigt.

2. Gehen Sie im Dialogfeld Ausführung starten wie folgt vor:

   a. (Optional) Um Ihre Ausführung zu identifizieren, können Sie im Feld Name einen Namen dafür angeben. Standardmäßig generiert Step Functions automatisch einen eindeutigen Ausführungsnamen.

> **Note**
>
> Mit Step Functions können Sie Namen für Zustandsmaschinen, Ausführungen und Aktivitäten sowie Beschriftungen erstellen, die Nicht-ASCII-Zeichen enthalten. Diese Nicht-ASCII-Namen funktionieren nicht mit Amazon. CloudWatch Um sicherzustellen, dass Sie CloudWatch Messwerte verfolgen können, wählen Sie einen Namen, der nur ASCII-Zeichen verwendet.

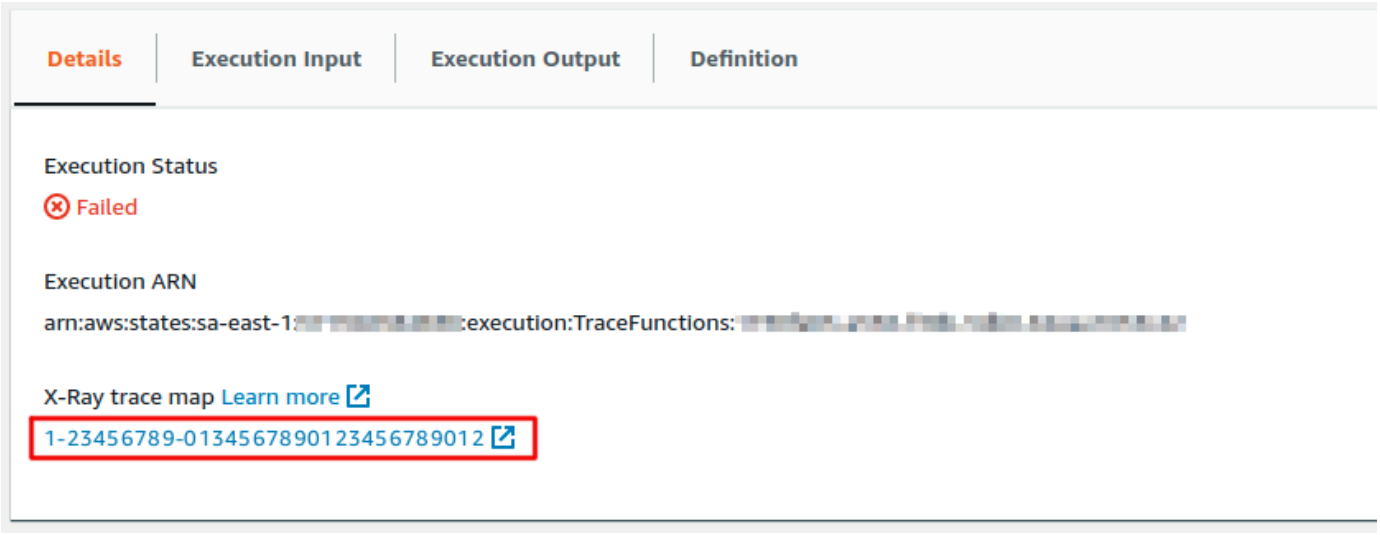
b. Wählen Sie Start execution (Ausführung starten) aus.

c. Die Step Functions Functions-Konsole leitet Sie zu einer Seite weiter, die mit Ihrer Ausführungs-ID betitelt ist. Diese Seite wird als Seite mit den Ausführungsdetails bezeichnet. Auf dieser Seite können Sie die Ausführungsergebnisse im Verlauf der Ausführung oder nach deren Abschluss überprüfen.

   Um die Ausführungsergebnisse zu überprüfen, wählen Sie in der Diagrammansicht einzelne Status aus und wählen Sie dann die einzelnen Registerkarten im Schrittdetails Bereich, um die Details der einzelnen Status, einschließlich Eingabe, Ausgabe und Definition, anzuzeigen. Einzelheiten zu den Ausführungsinformationen, die Sie auf der Seite mit den Ausführungsdetails einsehen können, finden Sie unterSeite mit Ausführungsdetails – Schnittstellenübersicht.

   Führen Sie mehrere (mindestens drei) Ausführungen aus.

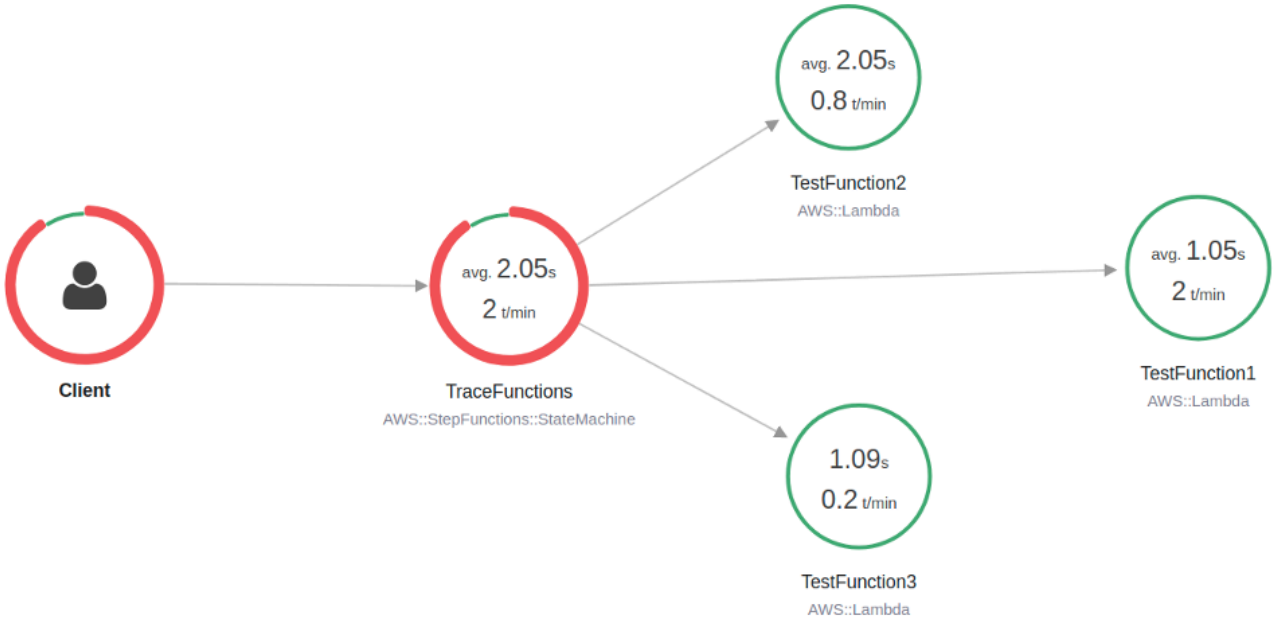
3. Nachdem die Ausführungen abgeschlossen sind, folgen Sie dem Link zur X-Ray-Trace-Map. Sie können den Trace anzeigen, während eine Ausführung noch läuft, aber vielleicht möchten Sie sich die Ausführungsergebnisse ansehen, bevor Sie die X-Ray-Trace-Map aufrufen.



4. Sehen Sie sich die Service-Map an, um festzustellen, wo Fehler auftreten, Verbindungen mit hoher Latenz hergestellt wurden oder ob Anfragen nicht erfolgreich waren. In diesem Beispiel können Sie sehen, wie viel Traffic jede Funktion empfängt. `TestFunction2`wurde häufiger aufgerufen als `TestFunction3` und `TestFunction1` wurde mehr als doppelt so oft aufgerufen wie`TestFunction2`.

   Die Service-Karte zeigt den Zustand jedes Knotens an und markiert ihn farblich auf Grundlage des Verhältnisses zwischen erfolgreichen Anrufen und Fehlern:

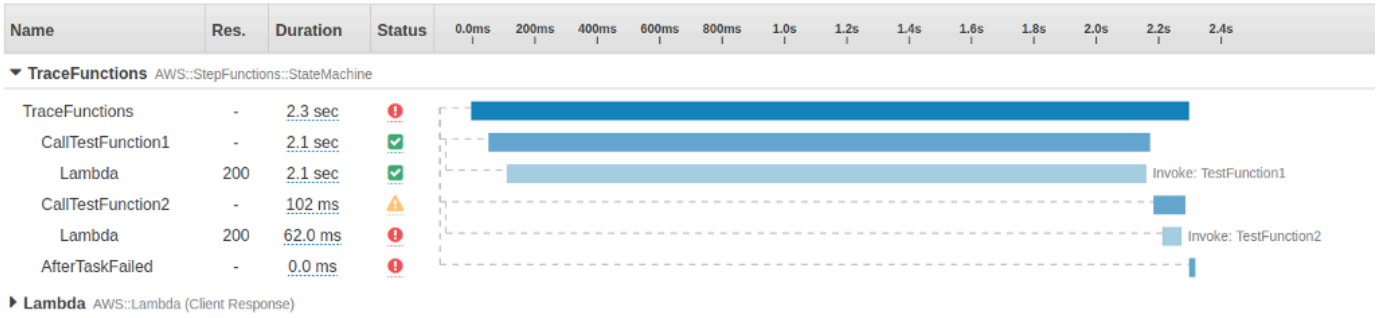
   - Grün für erfolgreiche Anrufe
   - Rot für Server-Fehler (500er-Fehler)
   - Gelb für Client-Fehler (400er-Fehler)
   - Violett für Ablehnungsfehler („Fehler 429 – Zu viele Anfragen”)



   Sie können auch einen Dienstknoten auswählen, um Anfragen für diesen Knoten anzuzeigen, oder eine Schnittstelle zwischen zwei Knoten, um Anfragen anzuzeigen, die über diese Verbindung übertragen wurden.

5. Sehen Sie sich die X-Ray-Trace-Map an, um zu sehen, was bei jeder Ausführung passiert ist. Die Timeline-Ansicht zeigt eine Hierarchie der Segmente und Untersegmente. Der erste Eintrag in der Liste ist das Segment, das alle vom Service für eine einzelne Anforderung

aufgezeichneten Daten darstellt. Unter dem Segment befinden sich Untersegmente. Dieses Beispiel zeigt Untersegmente, die von den Lambda-Funktionen aufgezeichnet wurden.



Weitere Informationen zum Verständnis von Röntgenspuren und zur Verwendung von X-Ray with Step Functions finden Sie im AWS X-Ray und Step Functions

Sammeln Sie Amazon S3 S3-Bucket-Informationen mithilfe von AWS SDK-Serviceintegrationen

Dieses Tutorial zeigt Ihnen, wie Sie eine AWS SDK-Integration mit Amazon Simple Storage Service durchführen. Der Zustandsmaschine, den Sie in diesem Tutorial erstellen, sammelt Informationen über Ihre Amazon S3 S3-Buckets und listet dann Ihre Buckets zusammen mit Versionsinformationen für jeden Bucket in der aktuellen Region auf.

Themen

- Schritt 1: Erstellen Sie den State Machine
- Schritt 2: Fügen Sie die erforderlichen IAM-Rollenberechtigungen hinzu
- Schritt 3: Führen Sie eine Standard-State-Machine-Ausführung aus
- Schritt 4: Führen Sie eine Express-State-Machine-Ausführung aus

Schritt 1: Erstellen Sie den State Machine

Mithilfe der Step Functions Functions-Konsole erstellen Sie eine Zustandsmaschine, die einen `Task` Status enthält, um alle Amazon S3 S3-Buckets im aktuellen Konto und in der aktuellen Region aufzulisten. Anschließend fügen Sie einen weiteren `Task` Status hinzu, der die `HeadBucket` API aufruft, um zu überprüfen, ob der zurückgegebene Bucket in der aktuellen Region zugänglich ist. Wenn auf den Bucket nicht zugegriffen werden kann, gibt der `HeadBucket` API-Aufruf den

`S3.S3Exception` Fehler zurück. Sie fügen einen `Catch` Block zum Abfangen dieser Ausnahme und einen `Pass` Status als Fallback-Status hinzu.

1. Öffnen Sie die Step Functions Functions-Konsole und wählen Sie Create State Machine.
2. Wählen Sie im Dialogfeld Vorlage auswählen die Option Leer aus.
3. Wählen Sie Select (Auswählen). Dadurch wird Workflow Studio in geöffnetEntwurfsmodus.
4. Für dieses Tutorial schreiben Sie die Amazon States Language (ASL-) Definition Ihrer Zustandsmaschine in denCode-Editor. Wählen Sie dazu Code.
5. Entfernen Sie den vorhandenen Standardcode und fügen Sie die folgende State-Machine-Definition ein.

```
{
  "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. Geben Sie einen Namen für Ihre Zustandsmaschine an. Wählen Sie dazu das Bearbeitungssymbol neben dem Standardnamen der Zustandsmaschine von MyStateMachine. Geben Sie dann unter State-Machine-Konfiguration einen Namen in das Feld State-Machine-Name ein.

   Geben Sie für dieses Tutorial den Namen **`Gather-S3-Bucket-Info-Standard`** ein.

7. (Optional) Geben Sie unter State-Machine-Konfiguration weitere Workflow-Einstellungen an, z. B. den Zustandsmaschinentyp und seine Ausführungsrolle.

   Behalten Sie alle Standardauswahlen in den State-Machine-Einstellungen bei.

Wenn Sie zuvor eine IAM-Rolle mit den richtigen Berechtigungen für Ihren Zustandsmaschine erstellt haben und diese verwenden möchten, wählen Sie unter Berechtigungen die Option Vorhandene Rolle auswählen und dann eine Rolle aus der Liste aus. Oder wählen Sie Einen Rollen-ARN eingeben aus und geben Sie dann einen ARN für diese IAM-Rolle ein.

8. Wählen Sie im Dialogfeld „Rollenerstellung bestätigen“ die Option Bestätigen aus, um fortzufahren.

   Sie können auch Rolleneinstellungen anzeigen wählen, um zur State-Machine-Konfiguration zurückzukehren.

   > **Note**
   >
   > Wenn Sie die von Step Functions erstellte IAM-Rolle löschen, kann Step Functions sie später nicht mehr neu erstellen. Ebenso kann Step Functions ihre ursprünglichen Einstellungen später nicht wiederherstellen, wenn Sie die Rolle ändern (z. B. indem Sie Step Functions aus den Principals in der IAM-Richtlinie entfernen).

   In Schritt 2 fügen Sie der State-Machine-Rolle die fehlenden Berechtigungen hinzu.

Schritt 2: Fügen Sie die erforderlichen IAM-Rollenberechtigungen hinzu

Um Informationen über die Amazon S3 S3-Buckets in Ihrer aktuellen Region zu sammeln, müssen Sie Ihrem State Machine die erforderlichen Berechtigungen für den Zugriff auf die Amazon S3 S3-Buckets gewähren.

1. Wählen Sie auf der Zustandsmaschinenseite die IAM-Rolle ARN aus, um die Seite Rollen für die Zustandsmaschinenrolle zu öffnen.
2. Wählen Sie Add permissions (Berechtigungen hinzufügen) und dann Create inline policy (Inline-Richtlinie erstellen) aus.
3. Wählen Sie die Registerkarte JSON und fügen Sie dann die folgenden Berechtigungen in den JSON-Editor ein.

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

```
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "s3:ListAllMyBuckets",
                "s3:ListBucket",
                "s3:GetBucketVersioning"
            ],
            "Resource": "*"
        }
    ]
}
```

4. Wählen Sie Richtlinie prüfen.
5. Geben Sie unter Review policy (Richtlinie prüfen) für den Richtlinien-Namen **`s3-bucket-permissions`** ein.
6. Wählen Sie Richtlinie erstellen aus.

Schritt 3: Führen Sie eine Standard-State-Machine-Ausführung aus

1. Wählen Sie auf der Seite Gather-S3-Bucket-Info-Standard die Option Ausführung starten aus.
2. Gehen Sie im Dialogfeld Ausführung starten wie folgt vor:

   a. (Optional) Um Ihre Ausführung zu identifizieren, können Sie im Feld Name einen Namen dafür angeben. Standardmäßig generiert Step Functions automatisch einen eindeutigen Ausführungsnamen.

   > **Note**
   >
   > Mit Step Functions können Sie Namen für Zustandsmaschinen, Ausführungen und Aktivitäten sowie Beschriftungen erstellen, die Nicht-ASCII-Zeichen enthalten. Diese Nicht-ASCII-Namen funktionieren nicht mit Amazon. CloudWatch Um sicherzustellen, dass Sie CloudWatch Messwerte verfolgen können, wählen Sie einen Namen, der nur ASCII-Zeichen verwendet.

   b. Wählen Sie Start execution (Ausführung starten) aus.

   c. Die Step Functions Functions-Konsole leitet Sie zu einer Seite weiter, die mit Ihrer Ausführungs-ID betitelt ist. Diese Seite wird als Seite mit den Ausführungsdetails

bezeichnet. Auf dieser Seite können Sie die Ausführungsergebnisse im Verlauf der Ausführung oder nach deren Abschluss überprüfen.

Um die Ausführungsergebnisse zu überprüfen, wählen Sie in der Diagrammansicht einzelne Status aus und wählen Sie dann die einzelnen Registerkarten im Schrittdetails Bereich, um die Details der einzelnen Status, einschließlich Eingabe, Ausgabe und Definition, anzuzeigen. Einzelheiten zu den Ausführungsinformationen, die Sie auf der Seite mit den Ausführungsdetails einsehen können, finden Sie unterSeite mit Ausführungsdetails – Schnittstellenübersicht.

Schritt 4: Führen Sie eine Express-State-Machine-Ausführung aus

1. Erstellen Sie eine Express-Zustandsmaschine mithilfe der in Schritt 1 bereitgestellten Zustandsmaschinen-Definition. Stellen Sie sicher, dass Sie auch die erforderlichen IAM-Rollenberechtigungen angeben, wie in Schritt 2 beschrieben.

   > **Tip**
   >
   > Benennen Sie den Express-Zustandsmaschine als**`Gather-S3-Bucket-Info-Express`**, um ihn von dem zuvor erstellten Standardcomputer zu unterscheiden.

2. Wählen Sie auf der Seite Gather-S3-Bucket-Info-Standard die Option Ausführung starten aus.
3. Gehen Sie im Dialogfeld Ausführung starten wie folgt vor:

   a. (Optional) Um Ihre Ausführung zu identifizieren, können Sie im Feld Name einen Namen dafür angeben. Standardmäßig generiert Step Functions automatisch einen eindeutigen Ausführungsnamen.

      > **Note**
      >
      > Mit Step Functions können Sie Namen für Zustandsmaschinen, Ausführungen und Aktivitäten sowie Beschriftungen erstellen, die Nicht-ASCII-Zeichen enthalten. Diese Nicht-ASCII-Namen funktionieren nicht mit Amazon. CloudWatch Um sicherzustellen, dass Sie CloudWatch Messwerte verfolgen können, wählen Sie einen Namen, der nur ASCII-Zeichen verwendet.

   b. Wählen Sie Start execution (Ausführung starten) aus.

c. Die Step Functions Functions-Konsole leitet Sie zu einer Seite weiter, die mit Ihrer Ausführungs-ID betitelt ist. Diese Seite wird als Seite mit den Ausführungsdetails bezeichnet. Auf dieser Seite können Sie die Ausführungsergebnisse im Verlauf der Ausführung oder nach deren Abschluss überprüfen.

   Um die Ausführungsergebnisse zu überprüfen, wählen Sie in der Diagrammansicht einzelne Status aus und wählen Sie dann die einzelnen Registerkarten im Schrittdetails Bereich, um die Details der einzelnen Status, einschließlich Eingabe, Ausgabe und Definition, anzuzeigen. Einzelheiten zu den Ausführungsinformationen, die Sie auf der Seite mit den Ausführungsdetails einsehen können, finden Sie unterSeite mit Ausführungsdetails – Schnittstellenübersicht.

Entwickler-Tools

Die folgenden Ressourcen enthalten zusätzliche Informationen zum Erstellen serverloser Workflows und zum Arbeiten mit Zustandsmaschinen:

- AWS CDK
- AWS Toolkit for VS Code

In den folgenden Themen finden Sie Informationen zum Erstellen, Testen und Debuggen von Zustandsmaschinen.

Themen

- Optionen für die Entwicklung
- AWS Step Functions und AWS SAM
- Verwenden von Workflow Studio in Application Composer
- Erstellen einer Lambda-Zustandsmaschine für Step Functions mit AWS CloudFormation
- Erstellen einer Lambda Zustandsmaschine für die Step Functions Verwendung AWS CDK
- Erstellen einer API Gateway-REST-API mit einem synchronen Express-Zustandsautomaten mithilfe der AWS CDK
- AWS Step Functions Datenwissenschaft-SDK für Python
- Bereitstellen von Zustandsmaschinen mit Terraform

Optionen für die Entwicklung

Sie können Ihre AWS Step Functions Zustandsmaschinen auf verschiedene Weise implementieren, z. B. mithilfe der Konsole, der SDKs oder einer lokalen Version von Step Functions für Tests und Entwicklung.

Themen

- Step Functions Functions-Konsole
- AWS SDKs
- Standard- und Express-Workflows

- HTTPS-Dienst-API
- Entwicklungsumgebungen
- Endpunkte
- AWS CLI
- Step Functions Lokal
- AWS Toolkit for Visual Studio Code
- AWS Serverless Application Model und Step Functions
- Terraform- und Step Functions
- Unterstützung für Definitionsformate

Step Functions Functions-Konsole

Sie können eine Zustandsmaschine mithilfe der Step Functions Functions-Konsole definieren. Sie können komplexe Zustandsmaschinen in der Cloud schreiben, ohne eine lokale Entwicklungsumgebung AWS Lambda zu verwenden, indem Sie Code für Ihre Aufgaben bereitstellen. Nach dem Schreiben können Sie dann die Step Functions Functions-Konsole verwenden, um Ihre Zustandsmaschine mithilfe der Amazon States-Sprache zu definieren.

Das Tutorial „Lambda State Machine erstellen“ verwendet diese Technik, um eine einfache Zustandsmaschine zu erstellen, sie auszuführen und ihre Ergebnisse anzuzeigen.

Datenflusssimulator

Sie können Workflows in der Step Functions Functions-Konsole entwerfen, implementieren und debuggen. Sie können den Datenfluss in Ihren Workflows auch mithilfe der `JsonPath` Eingabe- und Ausgabeverarbeitung steuern. Verwenden Sie den Datenflusssimulator in der Step Functions Functions-Konsole, um zu erfahren, wie Informationen von Status zu Status fließen, und um zu verstehen, wie Daten gefiltert und bearbeitet werden. Dieses Werkzeug simuliert jedes der Felder, die Step Functions zur Verarbeitung von Daten verwendet, z. B.`InputPath`,`Parameters`, `ResultSelector`,`OutputPath`, und`ResultPath`.

Weitere Informationen finden Sie unter Datenflusssimulator.

AWS SDKs

Step Functions wird von den AWS SDKs für Java, .NET, Ruby, PHP, Python (Boto 3) JavaScript, Go und C++ unterstützt. Diese SDKs bieten eine bequeme Möglichkeit, die HTTPS-API-Aktionen von Step Functions in mehreren Programmiersprachen zu verwenden.

Sie können Zustandsautomaten, Aktivitäten oder Zustandsautomat-Starter mithilfe der API-Aktionen entwickeln, die von diesen SDK-Bibliotheken bereitgestellt werden. Mithilfe dieser Bibliotheken können Sie auch auf Sichtbarkeitsoperationen zugreifen, um Ihre eigenen Überwachungs- und Berichtstools von Step Functions zu entwickeln.

Informationen zur Verwendung von Step Functions mit anderen AWS Services finden Sie in der Referenzdokumentation zu den aktuellen AWS SDKs und Tools für Amazon Web Services.

> **Note**
>
> Step Functions unterstützt nur HTTPS-Endpunkte.

Standard- und Express-Workflows

Wenn Sie einen neuen Zustandsautomaten erstellen, müssen Sie als `Type` entweder Standard oder Express auswählen. In beiden Fällen definieren Sie Ihren Zustandsmaschine mithilfe der Amazon States-Sprache. Die Ausführungen des Zustandsautomaten verhalten sich abhängig von dem von Ihnen ausgewählten Type (Typ) jeweils anders. Der von Ihnen gewählte Typ kann nach der Erstellung Ihres Zustandsmaschinen nicht mehr geändert werden.

Weitere Informationen finden Sie unter Protokollierung mitCloudWatchProtokolle.

HTTPS-Dienst-API

Step Functions bietet Serviceoperationen, auf die über HTTPS-Anfragen zugegriffen werden kann. Sie können diese Operationen verwenden, um direkt mit Step Functions zu kommunizieren und Ihre eigenen Bibliotheken in jeder Sprache zu entwickeln, die über HTTPS mit Step Functions kommunizieren kann.

Sie können mithilfe der Service-API-Aktionen Zustandsautomaten, Worker oder Zustandsautomaten-Starter entwickeln. Sie können über die API-Aktionen auch auf Sichtbarkeitsoperationen zugreifen, um Ihre eigenen Überwachungs- und Reporting-Tools zu entwickeln.

Ausführliche Informationen zu API-Aktionen finden Sie in der AWS Step Functions API-Referenz.

Entwicklungsumgebungen

Sie müssen eine Entwicklungsumgebung einrichten, die mit der Programmiersprache kompatibel ist, die Sie verwenden möchten.

Um beispielsweise für Step Functions mit Java zu entwickeln, müssen Sie auf jeder Ihrer Entwicklungs-Workstations eine Java-Entwicklungsumgebung wie die AWS SDK for Java installieren. Wenn Sie Eclipse IDE for Java Developers verwenden, sollten Sie auch die AWS Toolkit for Eclipse installieren. Dieses Eclipse-Plug-in fügt für die Entwicklung mit AWS nützliche Funktionen hinzu.

Wenn Ihre Programmiersprache eine Laufzeitumgebung erfordert, müssen Sie die Umgebung auf jedem Computer einrichten, auf dem diese Prozesse ausgeführt werden.

Endpunkte

Um die Latenz zu reduzieren und Daten an einem Ort zu speichern, der Ihren Anforderungen entspricht, bietet Step Functions Endpunkte in verschiedenen AWS Regionen.

Jeder Endpunkt in Step Functions ist völlig unabhängig. Zustandsautomaten oder Aktivitäten existieren nur in der Region, in der sie erstellt wurden. Zustandsautomaten und Aktivitäten, die Sie in einer Region erstellen, teilen keine Daten oder Attribute mit denjenigen, die in einer anderen Region erstellt wurden. Sie können beispielsweise eine Zustandsmaschine registrieren, die `STATES-Flows-1` in zwei verschiedenen Regionen benannt ist. Die `STATES-Flows-1` Zustandsmaschine in einer Region teilt keine Daten oder Attribute mit der `STATES-Flow-1` Zustandsmaschine in der anderen Region.

Eine Liste der Step Functions Functions-Endpunkte finden Sie unter AWS Step Functions Regionen und Endpunkte in der. Allgemeine AWS-Referenz

AWS CLI

Über AWS Command Line Interface (AWS CLI) können Sie auf viele Funktionen von Step Functions zugreifen. Dies AWS CLI ist eine Alternative zur Verwendung der Step Functions Functions-Konsole oder in einigen Fällen zur Programmierung mithilfe der Step Functions Functions-API-Aktionen. Sie können den beispielsweise verwenden, AWS CLI um eine Zustandsmaschine zu erstellen und dann Ihre vorhandenen Zustandsmaschinen aufzulisten.

Sie können Step Functions Functions-Befehle in verwenden, AWS CLI um Ausführungen zu starten und zu verwalten, Aktivitäten abzufragen, Task-Heartbeats aufzuzeichnen und vieles mehr. Eine

vollständige Liste der Step Functions Functions-Befehle, Beschreibungen der verfügbaren Argumente und Beispiele für ihre Verwendung finden Sie in der AWS CLI Befehlsreferenz.

AWS CLI Die Befehle orientieren sich eng an der Sprache von Amazon States, sodass Sie die verwenden können, AWS CLI um mehr über die API-Aktionen von Step Functions zu erfahren. Sie können auch Ihre vorhandenen API-Kenntnisse verwenden, um Code zu prototypisieren oder Step Functions Functions-Aktionen von der Befehlszeile aus auszuführen.

Step Functions Lokal

Zu Test- und Entwicklungszwecken können Sie Step Functions auf Ihrem lokalen Computer installieren und ausführen. Mit Step Functions Local können Sie eine Ausführung auf jedem Computer starten.

Die lokale Version von Step Functions kann AWS Lambda Funktionen aufrufen, AWS sowohl in als auch bei der lokalen Ausführung. Sie können auch andere unterstützte AWS Dienste koordinieren. Weitere Informationen finden Sie unter Zustandsmaschinen lokal testen.

> **Note**
>
> Step Functions Local verwendet Dummy-Konten, um zu funktionieren.

AWS Toolkit for Visual Studio Code

Sie können VS Code verwenden, um mit Remote-Zustandsmaschinen zu interagieren und Zustandsmaschinen lokal zu entwickeln. Sie können Zustandsmaschinen erstellen oder aktualisieren, vorhandene Zustandsmaschinen auflisten und Zustandsmaschinen ausführen oder herunterladen. Mit VS Code können Sie auch neue Zustandsautomaten aus Vorlagen erstellen, eine Visualisierung Ihres Zustandsautomaten anzeigen und Codeausschnitte, Codevervollständigung und Codevalidierung bereitstellen.

Weitere Informationen finden Sie im AWS Toolkit for Visual Studio Code Benutzerhandbuch

AWS Serverless Application Model und Step Functions

Step Functions ist in die integriert AWS Serverless Application Model, sodass Sie Workflows mit Lambda-Funktionen, APIs und Ereignissen integrieren können, um serverlose Anwendungen zu erstellen.

Sie können die AWS SAM CLI auch in Verbindung mit der AWS Toolkit for Visual Studio Code als Teil einer integrierten Erfahrung verwenden.

Weitere Informationen finden Sie unter AWS Step Functions und AWS SAM.

Terraform- und Step Functions

Terraform by HashiCorp ist ein Framework für die Erstellung von Anwendungen unter Verwendung von Infrastructure as Code (IaC). Mit Terraform können Sie Zustandsmaschinen erstellen und Funktionen wie die Vorschau von Infrastrukturbereitstellungen und die Erstellung wiederverwendbarer Vorlagen verwenden. Terraform-Vorlagen helfen Ihnen dabei, den Code zu verwalten und wiederzuverwenden, indem sie ihn in kleinere Abschnitte aufteilen.

Weitere Informationen finden Sie unter Bereitstellen von Zustandsmaschinen mit Terraform.

Unterstützung für Definitionsformate

Step Functions bietet eine Vielzahl von Tools, mit denen Sie Ihre State-Machine-Definitionen in verschiedenen Formaten bereitstellen können. Eine Amazon States Language (ASL) -Definition, die die Details Ihres State Machine spezifiziert, kann entweder als Zeichenfolge oder als serialisiertes Objekt mit JSON oder YAML bereitgestellt werden.

> **Note**
>
> YAML erlaubt einzeilige Kommentare. Alle YAML-Kommentare, die in der Zustandsmaschine-Definition einer Vorlage bereitgestellt werden, werden nicht in die Definition der erstellten Ressource übernommen. Stattdessen können Sie die `Comment` Eigenschaft innerhalb der State-Machine-Definition verwenden. Weitere Informationen finden Sie auf der Seite Struktur der Zustandsmaschine.

Die folgende Tabelle zeigt, welche Tools ASL-basierte Definitionen unterstützen.

Unterstützung für Definitionsformate nach Tool

	JSON	YAML	Stringifizierte Sprache der Amazonas-Staaten		
Step Functions Functions-Konsole	✓				
HTTPS-Dienst-API			✓		
AWS CLI			✓		
Step Functions Lokal			✓		
AWS Toolkit for Visual Studio Code	✓	✓			
AWS SAM	✓	✓			
AWS CloudFormation	✓	✓	✓		

> **Note**
>
> AWS CloudFormation und ermöglicht es Ihnen AWS SAM auch, Ihre State-Machine-Definitionen im JSON- oder YAML-Format auf Amazon S3 hochzuladen und den Amazon S3 S3-Speicherort der Definition in der Vorlage anzugeben. Dies kann die Lesbarkeit Ihrer Vorlagen verbessern, wenn Ihre State-Machine-Definition komplex ist. Weitere Informationen finden Sie auf der AWS::StepFunctions::StateMachine S3Location-Seite.

Die folgenden AWS CloudFormation Beispielvorlagen zeigen, wie Sie dieselbe State-Machine-Definition mit unterschiedlichen Eingabeformaten bereitstellen können.

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 und AWS SAM

Sie können die AWS SAM CLI in Verbindung mit der AWS Toolkit for Visual Studio Code als Teil einer integrierten Erfahrung verwenden, um Zustandsmaschinen lokal zu erstellen. Sie können eine serverlose Anwendung mit AWS SAM der VS Code IDE erstellen und dann Ihre Zustandsmaschine ausbauen. Anschließend können Sie Ihre Ressourcen validieren, verpacken und bereitstellen. Optional können Sie auch auf dem veröffentlichen AWS Serverless Application Repository.

> **Tip**
>
> Informationen zur Bereitstellung einer serverlosen Beispielanwendung, die einen Step Functions Functions-Workflow mit AWS SAM Ihrem startet AWS-Konto, finden Sie in Modul 11 — Bereitstellen mit AWS SAM von The AWS Step Functions Workshop.

Themen

Warum sollten Sie Step Functions mit verwenden AWS SAM?

Wenn Sie Step Functions mit verwenden AWS SAM , können Sie:

- Beginnen Sie mit einer AWS SAM Beispielvorlage.
- Integrieren Sie Ihren Zustandsautomaten in Ihre serverlose Anwendung.
- Verwenden Sie die Variablenersetzung, um ARNs zum Zeitpunkt der Bereitstellung in Ihrem State-Machine zu ersetzen.

  AWS CloudFormationunterstützt `DefinitionSubstitutions`, mit denen Sie dynamische Verweise in Ihrer Workflow-Definition zu einem Wert hinzufügen können, den Sie in Ihrer CloudFormation Vorlage angeben. Sie können dynamische Verweise hinzufügen, indem Sie Ihrer Workflow-Definition mithilfe der `${dollar_sign_brace}` Notation Substitutionen hinzufügen. Sie müssen diese dynamischen Verweise auch in der `DefinitionSubstitutions` Eigenschaft für die StateMachine Ressource in Ihrer CloudFormation Vorlage definieren. Diese Ersetzungen werden bei der Erstellung des CloudFormation Stacks durch tatsächliche Werte ersetzt. Weitere Informationen finden Sie unter DefinitionSubstitutions in Vorlagen AWS SAM.
- Geben Sie die Rolle Ihres Zustandsmaschinen mithilfe von AWS SAM Richtlinienvorlagen an.
- Initiieren Sie State-Machine-Ausführungen mit API Gateway, EventBridge Ereignissen oder nach einem Zeitplan innerhalb Ihrer AWS SAM Vorlage.

Step Functions Integration mit der AWS SAM Spezifikation

Sie können die AWS SAM Richtlinienvorlagen verwenden, um Ihrem Zustandsmaschine Berechtigungen hinzuzufügen. Mit diesen Berechtigungen können Sie Lambda-Funktionen und andere AWS Ressourcen orchestrieren, um komplexe und robuste Workflows zu bilden.

Integration von Schrittfunktionen in die SAM-CLI

Step Functions ist in die AWS SAM CLI integriert. Verwenden Sie diese Option, um einen Zustandsautomaten schnell in Ihre serverlose Anwendung zu entwickeln.

Probieren Sie das Erstellen einer Step Functions Functions-ZustandsmaschineAWS SAM Tutorial aus, um zu erfahren, wie Sie AWS SAM Zustandsmaschinen erstellen.

Zu den unterstützten AWS SAM CLI-Funktionen gehören:

CLI-Befehl	Beschreibung
sam init	Initialisiert eine serverlose Anwendung mit einer Vorlage. AWS SAM Kann mit einer SAM-Vorla ge für Schrittfunktionen verwendet werden.
sam validate	Validiert eine Vorlage. AWS SAM
sam package	Paketiert eine AWS SAM Anwendung. Es erstellt eine ZIP-Datei mit Ihrem Code und Ihren Abhängigkeiten und lädt sie dann auf Amazon S3 hoch. Anschließend wird eine Kopie Ihrer AWS SAM -Vorlage zurückgeg eben, wobei Verweise auf lokale Artefakte durch den Amazon S3-Speicherort ersetzt werden, zu dem der Befehl die Artefakte hochgeladen hat.
sam deploy	Stellt eine AWS SAM Anwendung bereit.
sam publish	Veröffentlichen Sie eine AWS SAM Anwendung auf der AWS Serverless Application Repositor y. Dieser Befehl verwendet ein AWS

CLI-Befehl	Beschreibung
	SAM Vorlagenpaket und veröffentlicht die Anwendung in der angegebenen Region.

> **Note**
>
> Wenn Sie AWS SAM local verwenden, können Sie Lambda und API Gateway lokal emulieren. Sie können Step Functions jedoch nicht lokal emulieren, indem Sie AWS SAM.

DefinitionSubstitutions in Vorlagen AWS SAM

Sie können Zustandsmaschinen mithilfe von CloudFormation Vorlagen mit definierenAWS SAM. Unter Verwendung von AWS SAM können Sie den Zustandsautomaten inline in der Vorlage oder in einer separaten Datei definieren. Die folgende AWS SAM Vorlage enthält eine Zustandsmaschine, die einen Aktienhandelsablauf simuliert. Diese Zustandsmaschine ruft drei Lambda Funktionen auf, um den Kurs einer Aktie zu überprüfen und festzustellen, ob die Aktie gekauft oder verkauft werden soll. Diese Transaktion wird dann in einer Amazon DynamoDB Tabelle aufgezeichnet. Die ARNs für die Lambda Funktionen und die DynamoDB Tabelle in der folgenden Vorlage werden mit `DefinitionSubstitutions`angegeben.

```
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
```

Der folgende Code ist die Zustandsmaschinen-Definition in der Datei`stock_trader.asl.json`, die im Erstellen einer Step Functions Functions-ZustandsmaschineAWS SAM Tutorial verwendet wird. Diese Zustandsmaschinen-Definition enthält mehrere Zustandsmaschinen, die durch die Notation `DefinitionSubstitutions` gekennzeichnet sind. `${dollar_sign_brace}` Anstatt

beispielsweise einen statischen Lambda Funktions-ARN für die `Check Stock Value` Aufgabe anzugeben, `${StockCheckerFunctionArn}` wird die Substitution verwendet. Diese Substitution ist in der DefinitionSubstitutions Eigenschaft der Vorlage definiert. `DefinitionSubstitutions`ist eine Zuordnung von Schlüssel-Wert-Paaren für die Zustandsmaschinen-Ressource. In`DefinitionSubstitutions`, $ {StockCheckerFunctionArn} wird mithilfe der systemeigenen Funktion dem ARN der `StockCheckerFunction` CloudFormation Ressource zugeordnet. `!GetAtt` Wenn Sie die AWS SAM Vorlage bereitstellen, werden die `DefinitionSubstitutions` in der Vorlage enthaltenen Werte durch die tatsächlichen Werte ersetzt.

```
{
    "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
        }
    }
}
```

Nächste Schritte

In den folgenden Ressourcen können Sie mehr über die Verwendung von Step Functions erfahren: AWS SAM

- Schließen Sie das Erstellen einer Step Functions Functions-ZustandsmaschineAWS SAM Tutorial ab, um eine Zustandsmaschine mit zu erstellen AWS SAM.
- Geben Sie eine AWS::Serverless::StateMachineRessource an.
- Suchen Sie die zu verwendenden AWS SAM -Richtlinienvorlagen.
- AWS Toolkit for Visual Studio CodeMit Step Functions verwenden.
- Lesen Sie die AWS SAM CLI-Referenz, um mehr über die Funktionen zu erfahren, die in verfügbar sind AWS SAM.

Sie können Ihre Workflows auch in Infrastructure as Code (IaC) mithilfe von Visual Buildern wie Workflow Studio in Application Composer entwerfen und erstellen. Weitere Informationen finden Sie unter Verwenden von Workflow Studio in Application Composer.

Verwenden von Workflow Studio in Application Composer

AWS Application Composerist ein Visual Builder, mit dem Sie AWS CloudFormation Vorlagen mithilfe einer einfachen grafischen Oberfläche entwickeln AWS SAM können. Mit entwerfen Sie eine AnwendungsarchitekturApplication Composer, indem Sie sie auf einer visuellen Leinwand ziehen, gruppieren und verbindenAWS-Services. Application Composererstellt dann aus Ihrem Entwurf eine IaC-Vorlage (Infrastructure as Code), die Sie verwenden können, um Ihre Anwendung mit der AWS SAM Befehlszeilenschnittstelle (AWS SAMCLI) oder CloudFormation bereitzustellen. Weitere Informationen über Application Composer finden Sie unter Was ist Application Composer.

Workflow Studio ist verfügbar inApplication Composer, um Sie beim Entwerfen und Erstellen Ihrer Workflows zu unterstützen. Workflow Studio in Application Composer bietet eine visuelle IaC-

Umgebung, mit der Sie Workflows auf einfache Weise in Ihre serverlosen Anwendungen integrieren können, die mit IaC-Tools wie Vorlagen erstellt wurden. CloudFormation Wenn Sie Workflow Studio in verwendenApplication Composer, verbindet es die einzelnen Workflow-Schritte mit AWS Ressourcen und generiert die Ressourcenkonfigurationen in einer Vorlage. AWS SAM Außerdem werden die IAM Berechtigungen hinzugefügt, die für die Ausführung Ihres Workflows erforderlich sind. Mit Workflow Studio in Application Composer können Sie Prototypen Ihrer Anwendungen erstellen und sie in produktionsreife Anwendungen umwandeln.

Wenn Sie Workflow Studio in verwendenApplication Composer, können Sie zwischen der Application Composer Arbeitsfläche und Workflow Studio hin und her wechseln.

Themen

Verwenden Sie Workflow Studio inApplication Composer, um einen serverlosen Workflow zu erstellen

1. Öffnen Sie die Application Composer-Konsole und wählen Sie Projekt erstellen, um ein Projekt zu erstellen.
2. Geben Sie im Suchfeld in der Ressourcenpalette **`state machine`** ein.
3. Ziehen Sie die Step FunctionsState Machine-Ressource auf die Arbeitsfläche.
4. Wählen Sie In Workflow Studio bearbeiten, um Ihre State Machine-Ressource zu bearbeiten.

   Die folgende Animation zeigt, wie Sie zum Workflow Studio wechseln können, um Ihre State-Machine-Definition zu bearbeiten.
   Eine Animation, die veranschaulicht, wie Sie Workflow Studio in verwenden könnenApplication Composer.

   Die Integration mit Workflow Studio zur Bearbeitung von Zustandsmaschinen, in denen Ressourcen erstellt wurden, Application Composer ist nur für

`AWS::Serverless::StateMachine`Ressourcen verfügbar. Diese Integration ist nicht für Vorlagen verfügbar, die die `AWS::StepFunctions::StateMachine`Ressource verwenden.

Verweisen Sie dynamisch auf Ressourcen mithilfe von CloudFormation Definitionsersetzungen in Workflow Studio

In Workflow Studio können Sie Definitionsersetzungen in Ihrer CloudFormation Workflow-Definition verwenden, um dynamisch auf Ressourcen zu verweisen, die Sie in Ihrer IaC-Vorlage definiert haben. Sie können Ihrer Workflow-Definition mithilfe der `${dollar_sign_brace}` Notation Platzhalterersetzungen hinzufügen, die bei der Stapelerstellung durch tatsächliche Werte ersetzt werden. CloudFormation Weitere Informationen zu Definitionsersetzungen finden Sie unter. DefinitionSubstitutions in Vorlagen AWS SAM

Die folgende Animation zeigt, wie Sie Platzhalterersetzungen für die Ressourcen in Ihrer State-Machine-Definition hinzufügen können.
Eine Animation, die veranschaulicht, wie Sie dynamisch auf Ressourcen wie AWS Lambda Funktionen und Definitionsersetzungen verweisen, wenn Sie Workflow Studio in verwenden. Application Composer

Connect Aufgaben zur Serviceintegration mit erweiterten Komponentenkarten

Sie können die Aufgaben, die optimierte Serviceintegrationen aufrufen, mit erweiterten Komponentenkarten in Application Composer Canvas verbinden. Dadurch werden automatisch alle Platzhalterersetzungen zugeordnet, die durch die `${dollar_sign_brace}` Notation in Ihrer Workflow-Definition und der `DefinitionSubstitution` Eigenschaft für Ihre Ressource angegeben sind. `StateMachine` Außerdem werden die entsprechenden AWS SAM Richtlinien für die Zustandsmaschine hinzugefügt.

Wenn Sie Aufgaben zur optimierten Serviceintegration Standardkomponentenkarten zuordnen, wird die Verbindungslinie nicht auf der Application Composer Arbeitsfläche angezeigt.

Die folgende Animation zeigt, wie Sie eine optimierte Aufgabe mit einer erweiterten Komponentenkarte verbinden und die Änderungen im Change Inspector anzeigen können.
Eine Animation, die veranschaulicht, wie Sie Aufgaben, die optimierte Serviceintegrationen aufrufen, mit erweiterten Komponentenkarten verbinden, wenn Sie Workflow Studio in Application Composer verwenden.

Sie können AWSSDK-Integrationen in Ihrem Task-Status nicht mit erweiterten Komponentenkarten oder optimierten Serviceintegrationen mit Standardkomponentenkarten verbinden. Für diese Aufgaben können Sie die Ersetzungen im Bereich „Ressourceneigenschaften“ auf der Application Composer Arbeitsfläche zuordnen und Richtlinien zur Vorlage hinzufügen. AWS SAM

> **Tip**
>
> Alternativ können Sie auch Platzhalterersetzungen für Ihren Zustandsmaschine unter Definitionsersetzungen im Bereich Ressourceneigenschaften zuordnen. Wenn Sie dies tun, müssen Sie die erforderlichen Berechtigungen für AWS-Service Ihre Task-State-Aufrufe in der Rolle State-Machine-Ausführung hinzufügen. Informationen zu den Berechtigungen, die Ihre Ausführungsrolle möglicherweise benötigt, finden Sie unterAusführungsrollen in Workflow Studio.

Die folgende Animation zeigt, wie Sie die Zuordnung der Platzhalterersetzung im Bereich „Ressourceneigenschaften“ manuell aktualisieren können.
Eine Animation, die veranschaulicht, wie Sie die Platzhalterersetzungszuordnung im Bereich mit den Ressourceneigenschaften manuell aktualisieren, wenn Sie Workflow Studio in verwenden. Application Composer

Importieren Sie bestehende Projekte und synchronisieren Sie sie lokal

Sie können bestehende CloudFormation AWS SAM Projekte öffnen, um sie Application Composer zum besseren Verständnis zu visualisieren und ihre Entwürfe zu ändern. Mit Application Composer der lokalen Synchronisierungsfunktion können Sie Ihre Vorlagen- und Codedateien automatisch synchronisieren und auf Ihrem lokalen Build-Computer speichern. Die Verwendung des lokalen Synchronisierungsmodus kann Ihre bestehenden Entwicklungsabläufe ergänzen. Stellen Sie sicher, dass Ihr Browser die File System Access API unterstützt, die es Webanwendungen ermöglicht, Dateien in Ihrem lokalen Dateisystem zu lesen, zu schreiben und zu speichern. Wir empfehlen, entweder Google Chrome oder Microsoft Edge zu verwenden.

Nicht verfügbare Funktionen von Workflow Studio in AWS Application Composer

Wenn Sie Workflow Studio in verwendenApplication Composer, sind einige der Funktionen von Workflow Studio nicht verfügbar. Darüber hinaus unterstützt der Abschnitt API-Parameter, der

Inspector im Bereich verfügbar ist, CloudFormation Definitionsersetzungen. Sie können die Substitutionen Codemodus mithilfe der `${dollar_sign_brace}` Notation hinzufügen. Weitere Informationen zu dieser Notation finden Sie unterDefinitionSubstitutions in Vorlagen AWS SAM.

In der folgenden Liste werden die Funktionen von Workflow Studio beschrieben, die nicht verfügbar sind, wenn Sie Workflow Studio in verwendenApplication Composer:

- Starter-Vorlagen — Starter-Vorlagen sind ready-to-run Beispielprojekte, mit denen die Workflow-Prototypen und -Definitionen automatisch erstellt werden. Diese Vorlagen stellen alle zugehörigen AWS Ressourcen bereit, die Ihr Projekt für Sie benötigt. AWS-Konto
- Konfigurationsmodus — In diesem Modus können Sie die Konfiguration Ihrer Zustandsmaschinen verwalten. Sie können Ihre State-Machine-Konfigurationen in Ihren IaC-Vorlagen aktualisieren oder den Bereich mit den Ressourceneigenschaften Application Composer auf der Leinwand verwenden. Informationen zum Aktualisieren von Konfigurationen im Bereich mit den Ressourceneigenschaften finden Sie unterConnect Aufgaben zur Serviceintegration mit erweiterten Komponentenkarten.
- TestState-API
- Option zum Importieren oder Exportieren von Workflow-Definitionen über die Dropdownschaltfläche „Aktionen“ in Workflow Studio. Wählen Sie stattdessen im Application Composer Menü Öffnen > Projektordner. Stellen Sie sicher, dass Sie den lokalen Synchronisierungsmodus aktiviert haben, damit Ihre Änderungen im Application Composer Canvas automatisch direkt auf Ihrem lokalen Computer gespeichert werden.
- Schaltfläche „Ausführen“. Wenn Sie Workflow Studio in verwendenApplication Composer, Application Composer generiert es den IaC-Code für Ihren Workflow. Daher müssen Sie zuerst die Vorlage bereitstellen. Führen Sie dann den Workflow in der Konsole oder über die aus AWS Command Line Interface(AWS CLI).

Erstellen einer Lambda-Zustandsmaschine für Step Functions mit AWS CloudFormation

Dieses Tutorial zeigt Ihnen, wie Sie eine grundlegende AWS Lambda Funktion mit erstellen AWS CloudFormation. Sie verwenden die AWS CloudFormation Konsole und eine YAML-Vorlage, um den Stack (IAM-Rollen, die Lambda-Funktion und die Zustandsmaschine) zu erstellen. Anschließend verwenden Sie die AWS Step Functions Konsole, um die Ausführung der Zustandsmaschine zu starten.

Weitere Informationen finden Sie unter Arbeiten mit CloudFormation Vorlagen und in der entsprechenden `AWS::StepFunctions::StateMachine` Ressource im AWS CloudFormation Benutzerhandbuch.

Themen

Schritt 1: Richten Sie Ihre AWS CloudFormation Vorlage ein

Bevor Sie die Beispielvorlagen verwenden, sollten Sie verstehen, wie die verschiedenen Teile einer AWS CloudFormation -Vorlage deklariert werden.

Themen

So erstellen Sie eine IAM-Rolle für Lambda

Definieren Sie die Vertrauensrichtlinie, die der IAM-Rolle für die Lambda-Funktion zugeordnet ist. Die folgenden Beispiele definieren eine Vertrauensrichtlinie, die entweder YAML oder JSON verwendet.

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"
                        }
                    ]
                }
            }
```

Eine Lambda-Funktion erstellen

Definieren Sie die folgenden Eigenschaften für eine Lambda-Funktion, die die Nachricht `Hello World` ausgibt.

> **Important**
>
> Stellen Sie sicher, dass sich Ihre Lambda-Funktion unter demselben AWS Konto und derselben AWS Region wie Ihr State Machine befindet.

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"
            }
        },
```

Um eine IAM-Rolle für die State-Machine-Ausführung zu erstellen

Definieren Sie die Vertrauensrichtlinie, die der IAM-Rolle für die State-Machine-Ausführung zugeordnet ist.

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": "*"
                                }
                            ]
                        }
                    }
                ]
            }
        },
```

So erstellen Sie eine Lambda-Zustandsmaschine

Definieren Sie die Lambda-Zustandsmaschine.

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"
                    ]
                }
            }
        }
```

Schritt 2: Verwenden Sie die AWS CloudFormation Vorlage, um eine Lambda State Machine zu erstellen

Sobald Sie die Komponenten der AWS CloudFormation Vorlage verstanden haben, können Sie sie zusammenfügen und die Vorlage verwenden, um einen AWS CloudFormation Stack zu erstellen.

Um die Lambda-Zustandsmaschine zu erstellen

1. Kopieren Sie die folgenden Beispieldaten in eine Datei mit dem Namen `MyStateMachine.yaml` für das YAML-Beispiel oder `MyStateMachine.json` für 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",
```

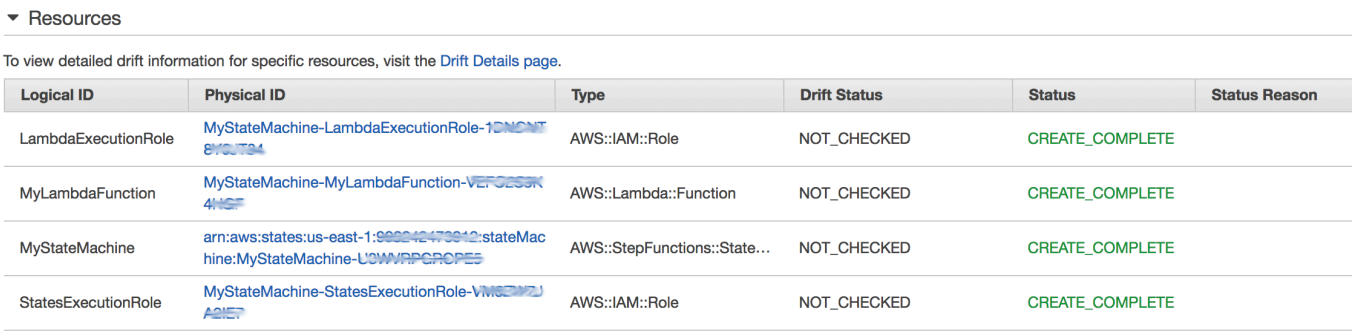
```
                            "Principal": {
                                "Service": "lambda.amazonaws.com"
                            },
                            "Action": "sts:AssumeRole"
                        }
                    ]
                }
            }
        },
        "MyLambdaFunction": {
            "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. Öffnen Sie die AWS CloudFormation -Konsole und wählen Sie Create Stack (Stack erstellen) aus.
3. Wählen Sie auf der Seite Select Template (Vorlage auswählen) Upload a template to Amazon S3 (Eine Vorlage zu Amazon S3 hochladen) aus. Wählen Sie Ihre Datei `MyStateMachine` und anschließend Next aus.
4. Geben Sie auf der Seite Specify Details (Details angeben) unter Stack name (Stack-Name) `MyStateMachine` ein und wählen Sie dann Next (Weiter) aus.
5. Wählen Sie auf der Seite Optionen Weiter aus.
6. Wählen Sie auf der Seite „Überprüfen“ die Option Ich bestätige, dass AWS CloudFormation möglicherweise IAM-Ressourcen erstellt werden. und wählen Sie dann Erstellen aus.

   AWS CloudFormation beginnt mit der Erstellung des `MyStateMachine` Stacks und zeigt den Status CREATE_IN_PROGRESS an. Wenn der Prozess abgeschlossen ist, zeigt AWS CloudFormation den Status CREATE_COMPLETE an.
7. (Optional) Wählen Sie zum Anzeigen der Ressourcen in Ihrem Stack den Stack und anschließend die Registerkarte Resources aus.



Schritt 3: Starten Sie eine State Machine-Ausführung

Nachdem Sie Ihre Lambda-Zustandsmaschine erstellt haben, können Sie deren Ausführung starten.

Starten der Ausführung des Zustandsautomaten

1. Öffnen Sie die Step Functions Functions-Konsole und wählen Sie den Namen der Zustandsmaschine aus, mit der Sie erstellt haben AWS CloudFormation.
2. Wählen Sie auf der Seite ***MyStateMachine-ABCDEFGHIJ1K*** die Option Neue Ausführung aus.

   Die Seite New execution wird angezeigt.
3. (Optional) Um Ihre Ausführung zu identifizieren, können Sie im Feld Name einen Namen dafür angeben. Standardmäßig generiert Step Functions automatisch einen eindeutigen Ausführungsnamen.

   > **Note**
   >
   > Mit Step Functions können Sie Namen für Zustandsmaschinen, Ausführungen und Aktivitäten sowie Beschriftungen erstellen, die Nicht-ASCII-Zeichen enthalten. Diese Nicht-ASCII-Namen funktionieren nicht mit Amazon. CloudWatch Um sicherzustellen, dass Sie CloudWatch Messwerte verfolgen können, wählen Sie einen Namen, der nur ASCII-Zeichen verwendet.
4. Wählen Sie Start Execution aus.

   Eine neue Ausführung Ihres Zustandsautomaten startet und eine neue Seite mit Ihrer laufenden Ausführung wird angezeigt.
5. (Optional) Überprüfen Sie unter Execution Details (Ausführungsdetails) den Execution Status (Ausführungsstatus) sowie die Zeitstempel Started (Gestartet) und Closed (Geschlossen).
6. Um die Ergebnisse Ihrer Ausführung anzuzeigen, wählen Sie Output (Ausgabe).

Erstellen einer Lambda Zustandsmaschine für die Step Functions Verwendung AWS CDK

Dieses Tutorial zeigt, wie Sie mithilfe von eine Funktion eine AWS Step Functions Zustandsmaschine erstellen, die eine AWS Lambda Funktion enthältAWS Cloud Development Kit (AWS CDK). Das AWS CDK ist ein Infrastructure as Code (IAC) -Framework, mit dem Sie die AWS Infrastruktur mithilfe einer vollwertigen Programmiersprache definieren können. Sie können eine App in einer der unterstützten Sprachen schreiben, die CDK einen oder mehrere Stapel enthält. Anschließend können Sie es zu einer AWS CloudFormation Vorlage zusammenfassen und in Ihrem AWS Konto bereitstellen.

Wir verwenden diese Methode, um eine Step Functions Zustandsmaschine zu definieren, die eine Lambda Funktion enthält, und verwenden dann die, AWS Management Console um die Zustandsmaschine auszuführen.

Bevor Sie mit diesem Tutorial beginnen, müssen Sie Ihre AWS CDK Entwicklungsumgebung wie unter Erste Schritte mit den AWS CDK — Voraussetzungen im AWS Cloud Development Kit (AWS CDK) Entwicklerhandbuch beschrieben einrichten. Installieren Sie dann das AWS CDK mit dem folgenden Befehl unterAWS CLI:

```
npm install -g aws-cdk
```

Dieses Tutorial liefert das gleiche Ergebnis wiethe section called “Erstellen einer Lambda-Zustandsmaschine mit AWS CloudFormation”. In diesem Tutorial müssen Sie jedoch AWS CDK keine IAM Rollen erstellen, der AWS CDK erledigt das für Sie. Die AWS CDK Version enthält auch einen Succeed Schritt, der veranschaulicht, wie Sie Ihrer Zustandsmaschine zusätzliche Schritte hinzufügen können.

> **Tip**
>
> Informationen zur Bereitstellung einer serverlosen Beispielanwendung, die einen Step Functions Workflow mithilfe von AWS CDK with TypeScript zu Ihrem startet AWS-Konto, finden Sie in Modul 10 — Deploy with AWS CDK von The AWS Step Functions Workshop.

Themen

Schritt 1: Einrichten des Projekts AWS CDK

1. Führen Sie in Ihrem Home-Verzeichnis oder, falls Sie es vorziehen, in einem anderen Verzeichnis den folgenden Befehl aus, um ein Verzeichnis für Ihre neue AWS CDK App zu erstellen.

> **Important**
>
> Achten Sie darauf, dem Verzeichnis einen Namen zu geben`step`. Die AWS CDK Anwendungsvorlage verwendet den Namen des Verzeichnisses, um Namen für Quelldateien und Klassen zu generieren. Wenn Sie einen anderen Namen verwenden, wird Ihre App nicht mit diesem Lernprogramm übereinstimmen.

TypeScript

```
mkdir step && cd step
```

JavaScript

```
mkdir step && cd step
```

Python

```
mkdir step && cd step
```

Java

```
mkdir step && cd step
```

C#

Stellen Sie sicher, dass Sie.NET Version 6.0 oder höher installiert haben. Weitere Informationen finden Sie unter Unterstützte Versionen.

```
mkdir step && cd step
```

2. Initialisieren Sie die App mit dem Befehl cdk init. Geben Sie die gewünschte Vorlage („App“) und die gewünschte Programmiersprache an, wie in den folgenden Beispielen gezeigt.

TypeScript

```
cdk init --language typescript
```

JavaScript

```
cdk init --language javascript
```

Python

```
cdk init --language python
```

Nachdem das Projekt initialisiert wurde, aktivieren Sie die virtuelle Umgebung des Projekts und installieren Sie die AWS CDK Basisabhängigkeiten.

```
source .venv/bin/activate
python -m pip install -r requirements.txt
```

Java

```
cdk init --language java
```

C#

```
cdk init --language csharp
```

Schritt 2: Verwenden Sie diese OptionAWS CDK, um eine Zustandsmaschine zu erstellen

Zunächst stellen wir die einzelnen Codeteile vor, die die Lambda Funktion und die Step Functions Zustandsmaschine definieren. Dann erklären wir, wie Sie sie in Ihrer AWS CDK App zusammenfügen. Schließlich erfahren Sie, wie Sie diese Ressourcen synthetisieren und einsetzen können.

So erstellen Sie eine Lambda-Funktion

Der folgende AWS CDK Code definiert die Lambda Funktion und stellt ihren Quellcode inline bereit.

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)
});
```

Sie können in diesem kurzen Beispielcode sehen:

- Der logische Name der Funktion,`MyLambdaFunction`.
- Der Quellcode für die Funktion, eingebettet als Zeichenfolge in den Quellcode der AWS CDK App.
- Andere Funktionsattribute, wie die zu verwendende Laufzeit (Node 18.x), der Einstiegspunkt der Funktion und ein Timeout.

Erstellen eines -Zustandsautomaten

Unsere Zustandsmaschine hat zwei Zustände: eine Lambda Funktionsaufgabe und einen Succeed Status. Die Funktion erfordert, dass wir eine erstellen Step Functionsthe section called “Aufgabe”, die unsere Funktion aufruft. Dieser Task-Status wird als erster Schritt in der Zustandsmaschine verwendet. Der Erfolgsstatus wird der Zustandsmaschine mithilfe der `next()` Methode des Task-Status hinzugefügt. Der folgende Code ruft zuerst die angegebene Funktion auf und verwendet dann die `next()` Methode`MyLambdaTask`, um einen Erfolgsstatus mit dem Namen `GreetedWorld` zu definieren.

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")))
});
```

Um die App zu erstellen und bereitzustellen AWS CDK

Bearbeiten Sie in Ihrem neu erstellten AWS CDK Projekt die Datei, die die Definition des Stacks enthält, sodass sie wie im folgenden Beispielcode aussieht. Sie kennen die Definitionen der Lambda Funktion und der Step Functions Zustandsmaschine aus den vorherigen Abschnitten.

1. Aktualisieren Sie den Stack wie in den folgenden Beispielen gezeigt.

   TypeScript

   Aktualisieren Sie `lib/step-stack.ts` mit dem folgenden Code.

```
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

   Aktualisieren Sie `lib/step-stack.js` mit dem folgenden Code.

```
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

Aktualisieren Sie `step/step_stack.py` mit dem folgenden Code.

```
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

Aktualisieren Sie `src/main/java/com.myorg/StepStack.java` mit dem folgenden Code.

```
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#

Aktualisieren Sie `scr/Step/StepStack.cs` mit dem folgenden Code.

```
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. Speichern Sie die Quelldatei und führen Sie dann den `cdk synth` Befehl im Hauptverzeichnis der App aus.

   AWS CDKführt die App aus und synthetisiert daraus eine AWS CloudFormation Vorlage. AWS CDKzeigt dann die Vorlage an.

   > **Note**
   >
   > Wenn Sie Ihr AWS CDK Projekt TypeScript früher erstellt haben, kann beim Ausführen des `cdk synth` Befehls der folgende Fehler zurückgegeben werden.
   >
   > ```
   > TSError: # Unable to compile TypeScript:
   > bin/step.ts:7:33 - error TS2554: Expected 2 arguments, but got 3.
   > ```
   >
   > Ändern Sie die `bin/step.ts` Datei wie im folgenden Beispiel gezeigt, um diesen Fehler zu beheben.
   >
   > ```
   > #!/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. Um die Lambda-Funktion und die Step Functions Functions-Zustandsmaschine für Ihr AWS Konto bereitzustellen, geben `cdk deploy` Sie Folgendes ein. Sie werden aufgefordert, die von ihm generierten IAM-Richtlinien zu genehmigen. AWS CDK

Schritt 3: Starten Sie eine State-Machine-Ausführung

Nachdem Sie Ihre Zustandsmaschine erstellt haben, können Sie deren Ausführung starten.

Starten der Ausführung des Zustandsautomaten

1. Öffnen Sie die Step Functions Functions-Konsole und wählen Sie den Namen der Zustandsmaschine aus, mit der Sie erstellt habenAWS CDK.
2. Wählen Sie auf der State-Machine-Seite die Option Ausführung starten aus.

   Das Dialogfeld Ausführung starten wird angezeigt.
3. (Optional) Um Ihre Ausführung zu identifizieren, können Sie im Feld Name einen Namen dafür angeben. Standardmäßig generiert Step Functions automatisch einen eindeutigen Ausführungsnamen.

   > **Note**
   >
   > Mit Step Functions können Sie Namen für Zustandsmaschinen, Ausführungen und Aktivitäten sowie Beschriftungen erstellen, die Nicht-ASCII-Zeichen enthalten. Diese Nicht-ASCII-Namen funktionieren nicht mit Amazon. CloudWatch Um sicherzustellen, dass Sie CloudWatch Messwerte verfolgen können, wählen Sie einen Namen, der nur ASCII-Zeichen verwendet.
4. Wählen Sie Start Execution aus.

Die Ausführung Ihrer Zustandsmaschine wird gestartet, und eine neue Seite mit Ihrer laufenden Ausführung wird angezeigt.

5. Die Step Functions Functions-Konsole leitet Sie zu einer Seite weiter, die mit Ihrer Ausführungs-ID betitelt ist. Diese Seite wird als Seite mit den Ausführungsdetails bezeichnet. Auf dieser Seite können Sie die Ausführungsergebnisse im Verlauf oder nach Abschluss der Ausführung überprüfen.

   Um die Ausführungsergebnisse zu überprüfen, wählen Sie in der Diagrammansicht einzelne Status aus und wählen Sie dann die einzelnen Registerkarten im Schrittdetails Bereich, um die Details der einzelnen Status, einschließlich Eingabe, Ausgabe und Definition, anzuzeigen. Einzelheiten zu den Ausführungsinformationen, die Sie auf der Seite mit den Ausführungsdetails einsehen können, finden Sie unterSeite mit Ausführungsdetails – Schnittstellenübersicht.

Schritt 4: Bereinigen

Nachdem Sie Ihre Zustandsmaschine getestet haben, empfehlen wir Ihnen, sowohl Ihre Zustandsmaschine als auch die zugehörige Lambda-Funktion zu entfernen, um Ressourcen in Ihrem AWS-Konto freizugeben. Führen Sie den `cdk destroy` Befehl im Hauptverzeichnis Ihrer App aus, um Ihre Zustandsmaschine zu entfernen.

Nächste Schritte

Weitere Informationen zur Entwicklung von AWS Infrastrukturen mithilfe von Cookies AWS CDK finden Sie im AWS CDKEntwicklerhandbuch.

Informationen zum Schreiben von AWS CDK-Apps in der Sprache Ihrer Wahl finden Sie unter:

TypeScript

- Arbeiten mit AWS CDK in TypeScript

JavaScript

- Arbeitet mit AWS CDK in JavaScript

Python

- Arbeiten mit AWS CDK in Python

Java

Arbeiten mit AWS CDK in Java

C#

Arbeiten mit AWS CDK in C#

Weitere Informationen zu den in diesem Tutorial verwendeten AWS Construct Library-Modulen finden Sie in den folgenden AWS CDK API-Referenzübersichten:

- aws-lambda
- aws-step-Funktionen
- aws-stepfunctions-tasks

Erstellen einer API Gateway-REST-API mit einem synchronen Express-Zustandsautomaten mithilfe der AWS CDK

Dieses Tutorial zeigt Ihnen, wie Sie eine API Gateway-REST-API mit Synchronous Express State Machine als Backend-Integration mithilfe der erstellen AWS Cloud Development Kit (AWS CDK). In diesem Tutorial wird das `StepFunctionsRestApi` Konstrukt verwendet, um den Zustandsautomaten mit dem API Gateway zu verbinden. Das `StepFunctionsRestApi`Konstrukt richtet ein Standard-Eingabe/Ausgabe-Mapping und die API Gateway-REST-API mit den erforderlichen Berechtigungen und einer HTTP-Methode „ANY“ ein. AWS CDK ist ein Infrastructure as Code (IAC)-Framework, mit dem Sie AWS Infrastruktur mit einer vollwertigen Programmiersprache definieren können. Sie schreiben eine App in einer der unterstützten Sprachen des CDK, die einen oder mehrere Stacks enthält, synthetisieren sie dann in einer - AWS CloudFormation Vorlage und stellen sie in Ihrem AWS Konto bereit. Wir verwenden sie, um eine API Gateway-REST-API zu definieren, die in Synchronous Express State Machine als Backend integriert ist, und dann die zu verwenden, AWS Management Console um die Ausführung zu initiieren.

Bevor Sie mit diesem Tutorial beginnen, richten Sie Ihre AWS CDK Entwicklungsumgebung wie unter Erste Schritte mit der AWS CDK – Voraussetzungen beschrieben ein und installieren Sie dann die , AWS CDK indem Sie Folgendes ausgeben:

```
npm install -g aws-cdk
```

Themen

Schritt 1: Einrichten Ihres AWS CDK Projekts

Erstellen Sie zunächst ein Verzeichnis für Ihre neue AWS CDK App und initialisieren Sie das Projekt.

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
```

Nachdem das Projekt initialisiert wurde, aktivieren Sie die virtuelle Umgebung des Projekts und installieren Sie die Basisabhängigkeiten AWS CDKvon .

```
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**
>
> Achten Sie darauf, das Verzeichnis zu benennen`stepfunctions-rest-api`. Die AWS CDK Anwendungsvorlage verwendet den Namen des Verzeichnisses, um Namen für Quelldateien und Klassen zu generieren. Wenn Sie einen anderen Namen verwenden, wird Ihre App nicht mit diesem Lernprogramm übereinstimmen.

Installieren Sie jetzt die Konstruktbibliotheksmodule für AWS Step Functions und 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

Bearbeiten Sie `pom.xml` des Projekts, um die folgenden Abhängigkeiten innerhalb des vorhandenen `<dependencies>`-Container hinzuzufügen.

```
        <dependency>
            <groupId>software.amazon.awscdk</groupId>
            <artifactId>stepfunctions</artifactId>
            <version>${cdk.version}</version>
        </dependency>
        <dependency>
            <groupId>software.amazon.awscdk</groupId>
            <artifactId>apigateway</artifactId>
            <version>${cdk.version}</version>
        </dependency>
```

Maven installiert diese Abhängigkeiten automatisch, wenn Sie Ihre App das nächste Mal erstellen. Um zu erstellen, stellen Sie `mvn compile` aus oder verwenden Sie den Entwicklung-Befehl Ihrer Java-IDE.

C#

```
dotnet add src/StepfunctionsRestApi package Amazon.CDK.AWS.Stepfunctions
dotnet add src/StepfunctionsRestApi package Amazon.CDK.AWS.APIGateway
```

Sie können die angegebenen Pakete auch mithilfe der Visual Studio NuGet-Benutzeroberfläche installieren, die über Tools > NuGet Package Manager > Manage NuGet Packages for Solution verfügbar ist.

Sobald Sie die Module installiert haben, können Sie sie in Ihrer AWS CDK App verwenden, indem Sie die folgenden Pakete importieren.

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

Fügen Sie Folgendes zu `import` in hinzu`stepfunctions-rest-api.go`.

```
"github.com/aws/aws-cdk-go/awscdk/awsapigateway"
"github.com/aws/aws-cdk-go/awscdk/awsstepfunctions"
```

Schritt 2: Verwenden der AWS CDK zum Erstellen einer API Gateway-REST-API mit Synchronous Express State Machine-Backend-Integration

Zuerst präsentieren wir die einzelnen Codeteile, die den Synchronous Express State Machine und die API Gateway-REST-API definieren, und erläutern dann, wie Sie sie in Ihre AWS CDK App einfügen. Anschließend erfahren Sie, wie Sie diese Ressourcen synthetisieren und bereitstellen.

> **Note**
>
> Der Zustandsautomat, den wir hier anzeigen werden, ist ein einfacher Zustandsautomat mit dem `Pass` Status .

So erstellen Sie einen Express-Zustandsautomaten

Dies ist der AWS CDK Code, der einen einfachen Zustandsautomaten mit einem -`Pass`Zustand definiert.

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,
})
```

Sie können in diesem kurzen Snippet sehen:

- Die Computerdefinition mit dem Namen `PassState`, bei der es sich um einen `Pass` Zustand handelt.
- Der logische Name des Zustandsautomaten, `MyStateMachine`.
- Die Computerdefinition wird als State Machine-Definition verwendet.

- Der Typ des Zustandsautomaten ist auf gesetzt, `EXPRESS` da `StepFunctionsRestApi` nur einen Synchronous Express-Zustandsautomaten zulässt.

So erstellen Sie die API Gateway-REST-API mithilfe des **`StepFunctionsRestApi`**Konstrukts

Wir werden das -`StepFunctionsRestApi`Konstrukt verwenden, um die API Gateway-REST-API mit den erforderlichen Berechtigungen und der Standard-Eingabe/Ausgabe-Zuweisung zu erstellen.

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,
})
```

So erstellen und stellen Sie die AWS CDK App bereit

Bearbeiten Sie in dem von Ihnen erstellten AWS CDK Projekt die Datei mit der Definition des Stacks so, dass sie wie der folgende Code aussieht. Sie werden die Definitionen des Step-Functions-Zustandsautomaten und des API Gateways von oben erkennen.

TypeScript

Aktualisieren Sie `lib/stepfunctions-rest-api-stack.ts` damit darauf folgendes steht.

```
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

Aktualisieren Sie `lib/stepfunctions-rest-api-stack.js` damit darauf folgendes steht.

```
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

Aktualisieren Sie `stepfunctions_rest_api/stepfunctions_rest_api_stack.py` damit darauf folgendes steht.

```
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

Aktualisieren Sie `src/main/java/com.myorg/StepfunctionsRestApiStack.java` damit darauf folgendes steht.

```
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#

Aktualisieren Sie `src/StepfunctionsRestApi/StepfunctionsRestApiStack.cs` damit darauf folgendes steht.

```
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

Aktualisieren Sie `stepfunctions-rest-api.go` damit darauf folgendes steht.

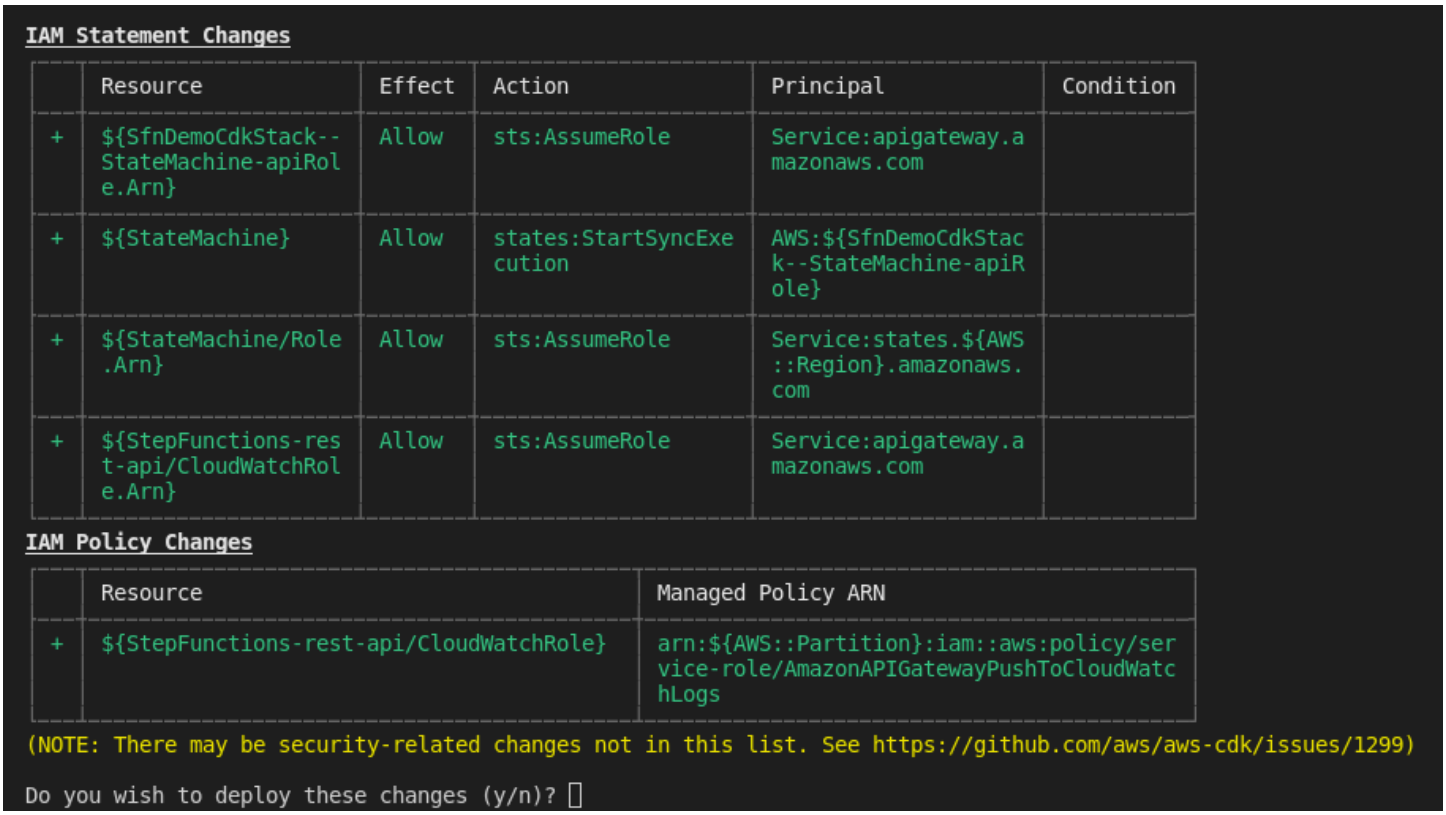
```
package main
import (
    "github.com/aws/aws-cdk-go/awscdk"
    "github.com/aws/aws-cdk-go/awscdk/awsapigateway"
    "github.com/aws/aws-cdk-go/awscdk/awsstepfunctions"
    "github.com/aws/constructs-go/constructs/v3"
    "github.com/aws/jsii-runtime-go"
)


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")),
    // }
}
```

Speichern Sie die Quelldatei, und geben Sie `cdk synth` im Hauptverzeichnis der App heraus. Der AWS CDK führt die App aus und synthetisiert daraus eine - AWS CloudFormation Vorlage und zeigt dann die Vorlage an.

Um das Amazon API Gateway und den AWS Step Functions Zustandsautomaten tatsächlich in Ihrem AWS-Konto bereitzustellen, geben Sie aus`cdk deploy`. Sie werden aufgefordert, die von AWS CDK generierten IAM-Richtlinien zu genehmigen. Die erstellten Richtlinien sehen etwa wie folgt aus:



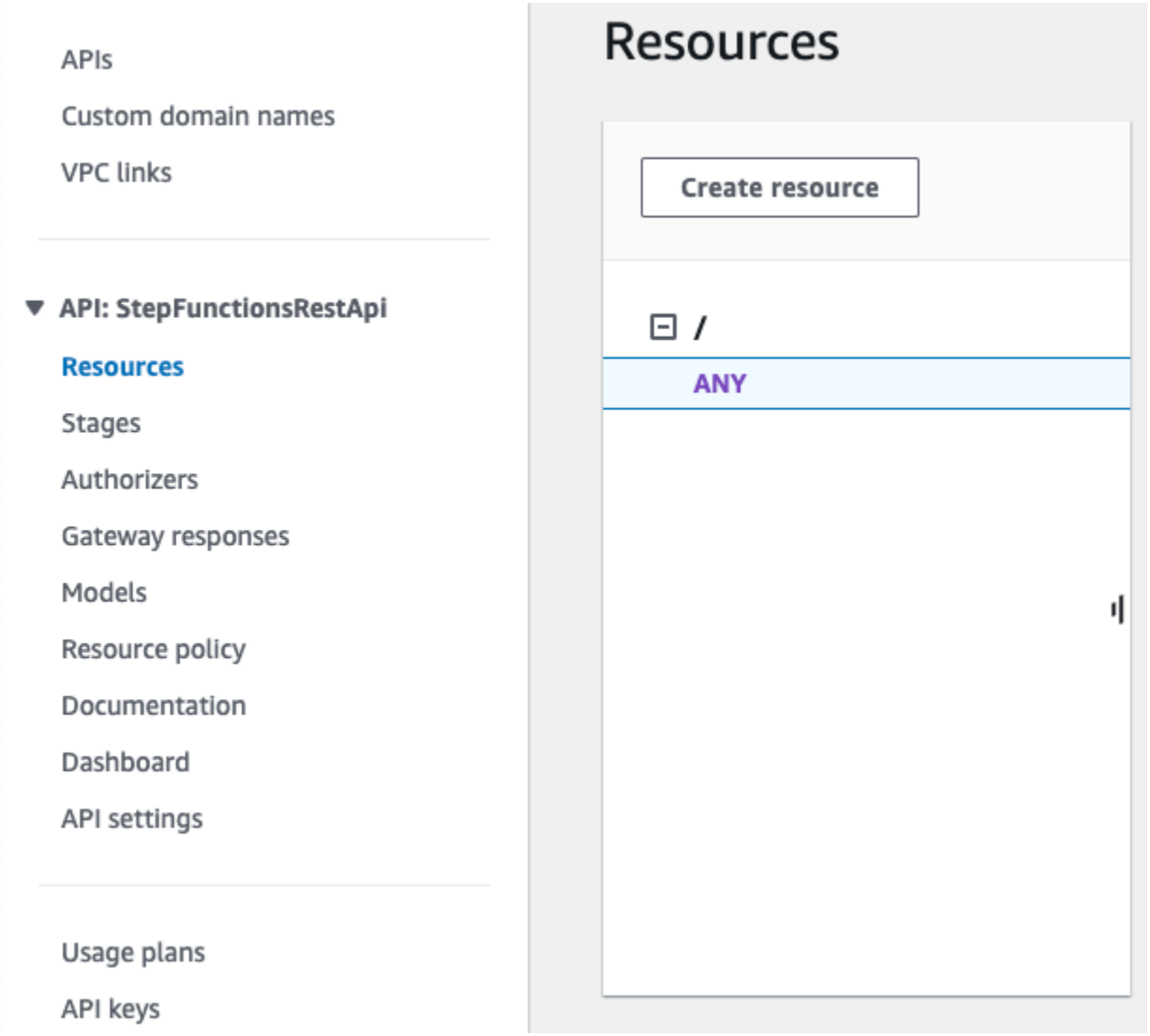
Schritt 3: Testen des API Gateways

Nachdem Sie Ihre API Gateway-REST-API mit Synchronous Express State Machine als Backend-Integration erstellt haben, können Sie das API Gateway testen.

So testen Sie das bereitgestellte API Gateway mit der API Gateway-Konsole

1. Öffnen Sie die Amazon API Gateway-Konsole und melden Sie sich an.

2. Wählen Sie Ihre REST-API mit dem Namen `StepFunctionsRestApi`.
3. Wählen Sie im Bereich Ressourcen die ANY Methode aus.



4. Wählen Sie die Registerkarte Test. Möglicherweise müssen Sie die rechte Pfeiltaste wählen, um die Registerkarte anzuzeigen.
5. Wählen Sie für Method (Methode) die Option POST aus.
6. Kopieren Sie für Anforderungstext die folgenden Anforderungsparameter.

```
{
    "key": "Hello"
}
```

7. Wählen Sie Test aus. Sie bekommen die folgenden Informationen angezeigt:

   - Request ist der Pfad der Ressource, die für die Methode aufgerufen wurde.

- Status ist der HTTP-Statuscode der Antwort.
- Latenz ist die Zeit, die zwischen dem Empfang der Anforderung des Aufrufers und der zurückgegebenen Antwort vergeht.
- Antworttext ist der Text der HTTP-Antwort.
- Antwort-Header sind die Header der HTTP-Antwort.
- Das Protokoll zeigt die simulierten Amazon- CloudWatch Logs-Einträge, die geschrieben worden wären, wenn diese Methode außerhalb der API-Gateway-Konsole aufgerufen worden wäre.

> **Note**
>
> Obwohl die CloudWatch Logs-Einträge simuliert werden, sind die Ergebnisse des Methodenaufrufs echt.

Die Ausgabe des Antworttexts sollte etwa wie folgt aussehen:

```
"Hello"
```

> **Tip**
>
> Testen Sie das API Gateway mit verschiedenen Methoden und einer ungültigen Eingabe, um die Fehlerausgabe zu sehen. Möglicherweise möchten Sie den Zustandsautomaten so ändern, dass er nach einem bestimmten Schlüssel sucht, und während Tests geben Sie den falschen Schlüssel an, um die Ausführung des Zustandsautomaten zu fehlschlagen und eine Fehlermeldung in der Antworttextausgabe zu generieren.

So testen Sie die bereitgestellte API mithilfe von cURL

1. Öffnen Sie ein Terminal-Fenster.
2. Kopieren Sie den folgenden cURL-Befehl und fügen Sie ihn in das Terminalfenster ein. Ersetzen Sie dabei `<api-id>` mit der API-ID Ihrer API und `<region>` mit der Region, in der Ihre API bereitgestellt ist.

```
curl -X POST\
```

```
'https://<api-id>.execute-api.<region>.amazonaws.com/prod' \
-d '{"key":"Hello"}' \
-H 'Content-Type: application/json'
```

Die Ausgabe des Antworttexts sollte etwa wie folgt aussehen:

```
"Hello"
```

> **Tip**
>
> Testen Sie das API Gateway mit verschiedenen Methoden und einer ungültigen Eingabe, um die Fehlerausgabe zu sehen. Möglicherweise möchten Sie den Zustandsautomaten ändern, um nach einem bestimmten Schlüssel zu suchen, und während Tests stellen Sie den falschen Schlüssel bereit, um die Ausführung des Zustandsautomaten zu fehlschlagen und eine Fehlermeldung in der Antworttextausgabe zu generieren.

Schritt 4: Bereinigen

Wenn Sie mit dem Ausprobieren Ihres API Gateways fertig sind, können Sie sowohl den Zustandsautomaten als auch das API Gateway mit dem AWS-CDK außer Betrieb nehmen. Geben Sie `cdk destroy` im Hauptverzeichnis Ihrer App heraus.

AWS Step Functions Datenwissenschaft-SDK für Python

Das AWS Step Functions Data Science SDK ist eine Open-Source-Bibliothek für Datenwissenschaftler. Mit diesem SDK können Sie Workflows erstellen, die Modelle für maschinelles Lernen mithilfe von Step Functions verarbeiten SageMaker und veröffentlichen. Sie können auch mehrstufige Workflows für maschinelles Lernen in Python erstellen, die die AWS Infrastruktur in großem Umfang orchestrieren, ohne die AWS Dienste separat bereitstellen und integrieren zu müssen.

Das AWS Step Functions Data Science SDK bietet eine Python-API, mit der Step Functions Functions-Workflows erstellt und aufgerufen werden können. Sie können diese Workflows direkt in Python sowie in Jupyter-Notebooks verwalten und ausführen.

Mit dem AWS Step Functions Data Science SDK können Sie nicht nur produktionsreife Workflows direkt in Python erstellen, sondern auch diesen Workflow kopieren, mit neuen Optionen experimentieren und den verfeinerten Workflow dann in die Produktion überführen.

Weitere Informationen zum AWS Step Functions Data Science SDK finden Sie im Folgenden:

- Projekt auf Github
- SDK-Dokumentation
- Die folgenden Beispiel-Notebooks, die in Jupyter-Notebook-Instanzen in der SageMaker Konsole und im zugehörigen Projekt verfügbar sind: GitHub
  - `hello_world_workflow.ipynb`
  - `machine_learning_workflow_abalone.ipynb`
  - `training_pipeline_pytorch_mnist.ipynb`

Bereitstellen von Zustandsmaschinen mit Terraform

Terraform by HashiCorp ist ein Framework für die Erstellung von Anwendungen unter Verwendung von Infrastructure as Code (IaC). Mit Terraform können Sie Zustandsmaschinen erstellen und Funktionen wie die Vorschau von Infrastrukturbereitstellungen und die Erstellung wiederverwendbarer Vorlagen verwenden. Terraform-Vorlagen helfen Ihnen dabei, den Code zu verwalten und wiederzuverwenden, indem sie ihn in kleinere Abschnitte aufteilen.

Wenn Sie mit Terraform vertraut sind, können Sie den in diesem Thema beschriebenen Entwicklungszyklus als Modell für die Erstellung und Bereitstellung Ihrer Zustandsmaschinen in Terraform verfolgen. Wenn Sie mit Terraform nicht vertraut sind, empfehlen wir Ihnen, zunächst den Workshop Einführung in Terraform zu absolvieren, um sich mit Terraform vertraut zu machen. AWS

> **Tip**
>
> Ein Beispiel für eine mit Terraform erstellte Zustandsmaschine für Sie bereitstellenAWS-Konto, finden Sie im Modul Verwaltung von Zustandsmaschinen mit Infrastruktur als Code von The Workshop. AWS Step Functions

In diesem Thema

- Voraussetzungen
- Analysieren Sie den Lebenszyklus der Maschinenentwicklung mit Terraform

- IAM-Rollen und -Richtlinien für Ihren State Machine

Voraussetzungen

Bevor Sie beginnen, stellen Sie sicher, dass Sie die folgenden Voraussetzungen erfüllen:

- Installieren Sie Terraform auf Ihrem Computer. Informationen zur Installation von Terraform finden Sie unter Terraform installieren.
- Installieren Sie Step Functions Local auf Ihrem Computer. Wir empfehlen, das Docker-Image Step Functions Local zu installieren, um Step Functions Local zu verwenden. Weitere Informationen finden Sie unter Zustandsmaschinen lokal testen.
- Installieren Sie AWS SAM CLI. Informationen zur Installation finden Sie unter Installation der AWS SAM CLI im AWS Serverless Application ModelEntwicklerhandbuch.
- Installieren Sie denAWS Toolkit for Visual Studio Code, um das Workflow-Diagramm Ihrer Zustandsmaschinen anzuzeigen. Informationen zur Installation finden Sie unter Installation von AWS Toolkit for Visual Studio Code im AWS Toolkit for Visual Studio CodeBenutzerhandbuch.

Analysieren Sie den Lebenszyklus der Maschinenentwicklung mit Terraform

Das folgende Verfahren erklärt, wie Sie einen State-Machine-Prototyp, den Sie mit Workflow Studio in der Step Functions Functions-Konsole erstellen, als Ausgangspunkt für die lokale Entwicklung mit Terraform und dem verwenden können. AWS Toolkit for Visual Studio Code

Das vollständige Beispiel, in dem die Zustandsmaschine-Entwicklung mit Terraform erörtert und die Best Practices im Detail vorgestellt werden, finden Sie unter Bewährte Methoden für das Schreiben von Step Functions Terraform-Projekten.

Um den Entwicklungszyklus einer State Machine mit Terraform zu beginnen

1. Starten Sie ein neues Terraform-Projekt mit dem folgenden Befehl.

   ```
   terraform init
   ```

2. Öffnen Sie die Step Functions Functions-Konsole, um einen Prototyp für Ihre State Machine zu erstellen.
3. Gehen Sie in Workflow Studio wie folgt vor:

   a. Erstellen Sie Ihren Workflow-Prototyp.

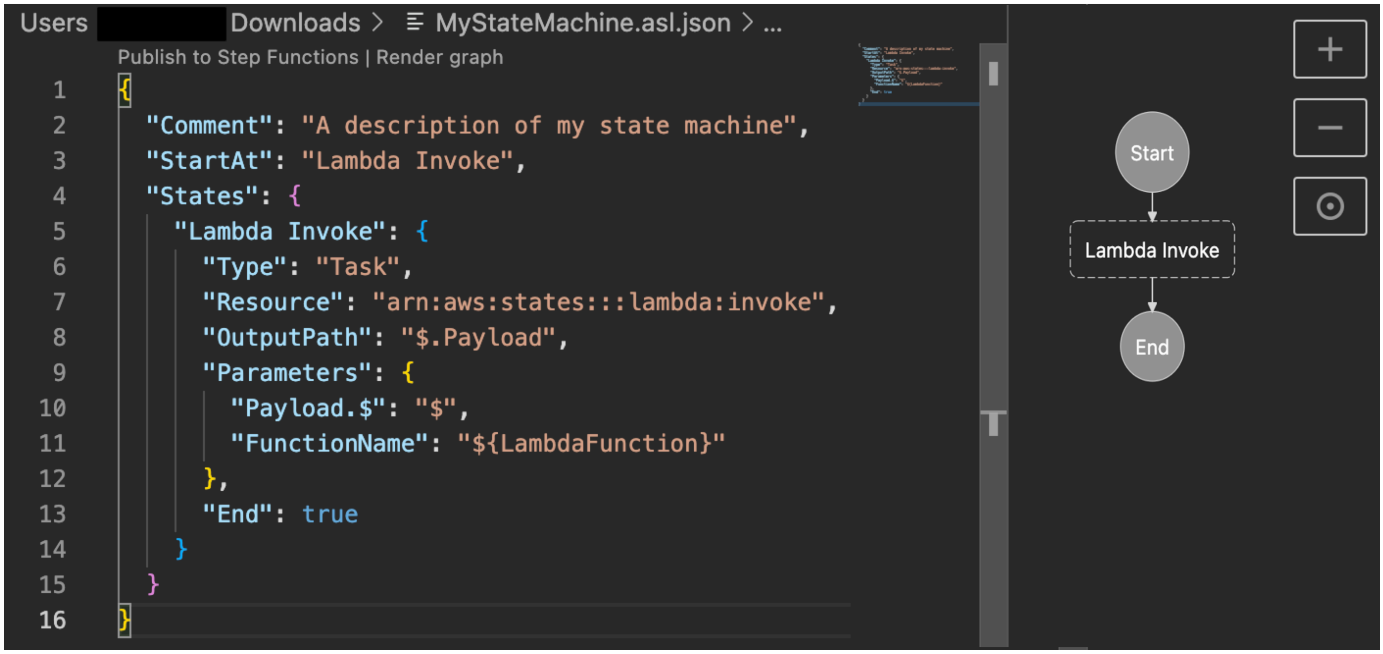
b. Exportieren Sie die Amazon States Language (ASL) -Definition Ihres Workflows. Wählen Sie dazu die Dropdownliste Import/Export und dann JSON-Definition exportieren aus.

4. Speichern Sie die exportierte ASL-Definition in Ihrem Projektverzeichnis.

   Sie übergeben die exportierte ASL-Definition als Eingabeparameter an die `aws_sfn_state_machine`Terraform-Ressource, die die Funktion verwendet. `templatefile` Diese Funktion wird innerhalb des Definitionsfeldes verwendet, das die exportierte ASL-Definition und alle Variablenersetzungen weitergibt.

   > **Tip**
   >
   > Da die ASL-Definitionsdatei lange Textblöcke enthalten kann, empfehlen wir, die Inline-EOF-Methode zu vermeiden. Dadurch ist es einfacher, Parameter in Ihre Zustandsmaschinen-Definition einzufügen.

5. (Optional) Aktualisieren Sie die ASL-Definition in Ihrer IDE und visualisieren Sie Ihre Änderungen mithilfe von. AWS Toolkit for Visual Studio Code



   Um zu vermeiden, dass Sie Ihre Definition ständig exportieren und in Ihr Projekt umgestalten, empfehlen wir Ihnen, Aktualisierungen lokal in Ihrer IDE vorzunehmen und diese Aktualisierungen mit Git zu verfolgen.

6. Testen Sie Ihren Workflow mit Step Functions Local.

> **Tip**
>
> Mit AWS SAMCLI Local können Sie Serviceintegrationen mit Lambda-Funktionen und API-Gateway-APIs auch lokal in Ihrer Zustandsmaschine testen.

7. Zeigen Sie eine Vorschau Ihrer Zustandsmaschine und anderer AWS Ressourcen an, bevor Sie die Zustandsmaschine bereitstellen. Führen Sie dazu den folgenden Befehl aus.

```
terraform plan
```

8. Stellen Sie Ihre Zustandsmaschine mithilfe des folgenden Befehls von Ihrer lokalen Umgebung oder über CI/CD-Pipelines aus bereit.

```
terraform apply
```

9. (Optional) Bereinigen Sie Ihre Ressourcen und löschen Sie die Zustandsmaschine mit dem folgenden Befehl.

```
terraform destroy
```

IAM-Rollen und -Richtlinien für Ihren State Machine

Verwenden Sie die Terraform-Dienstintegrationsrichtlinien, um Ihrer Zustandsmaschine die erforderlichen IAM-Berechtigungen hinzuzufügen, z. B. die Berechtigung zum Aufrufen von Lambda-Funktionen. Sie können auch explizite Rollen und Richtlinien definieren und diese Ihrer Zustandsmaschine zuordnen.

Das folgende Beispiel für eine IAM-Richtlinie gewährt Ihrer Zustandsmaschine Zugriff, um eine Lambda-Funktion mit dem Namen aufzurufen. *myFunction*

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "lambda:InvokeFunction"
      ],
```

```
      "Resource": "arn:aws:lambda:us-east-1:123456789012:function:myFunction"
    }
  ]
}
```

Wir empfehlen außerdem, die aws_iam_policy_documentDatenquelle bei der Definition von IAM-Richtlinien für Ihre Zustandsmaschinen in Terraform zu verwenden. Auf diese Weise können Sie überprüfen, ob Ihre Richtlinie falsch formatiert ist, und alle Ressourcen durch Variablen ersetzen.

Das folgende Beispiel für eine IAM-Richtlinie verwendet die `aws_iam_policy_document` Datenquelle und gewährt Ihrer Zustandsmaschine Zugriff, um eine Lambda-Funktion mit dem Namen aufzurufen. *myFunction*

```
data "aws_iam_policy_document" "state_machine_role_policy" {

  statement {
    effect = "Allow"

    actions = [
      "lambda:InvokeFunction"
    ]

    resources = ["${aws_lambda_function.[[myFunction]].arn}:*"]
  }

}
```

> **Tip**
>
> Weitere erweiterte AWS Architekturmuster, die mit Terraform bereitgestellt wurden, finden Sie unter Terraform-Beispiele unter Serverless Land Workflows Collection.

Testen und Debuggen

Step Functionsbietet verschiedene Möglichkeiten zum Testen und Debuggen von Zustandsmaschinen. Sie können beispielsweise Ihre Zustandsmaschinen in der Konsole testen und debuggen, die TestStateAPI verwenden, um einen einzelnen Status zu testen, oder Step Functions Local verwenden, um Zustandsmaschinen lokal zu testen.

Mithilfe der TestStateAPI stellen Sie die Definition eines einzelnen Zustands bereit und führen ihn aus. Sie können einen einzelnen Status testen, ohne einen Zustandsmaschine zu erstellen oder einen vorhandenen Zustandsmaschine zu aktualisieren.

Step FunctionsLocal ist eine herunterladbare Version vonStep Functions, mit der Sie Anwendungen entwickeln und testen können, indem Sie eine Version von verwenden, die in Ihrer eigenen Entwicklungsumgebung Step Functions ausgeführt wird. Mit Step Functions Local können Sie Ihre Zustandsmaschinen ausführen, um deren Eingabe- und Ausgabedatenflüsse, Integrationen mit unterstützten Diensten und mehr in Ihrer lokalen Entwicklungsumgebung zu testen.

Themen

- Verwenden einer TestState API zum Testen eines Zustands
- Zustandsmaschinen lokal testen

Verwenden einer TestState API zum Testen eines Zustands

Die TestState API akzeptiert die Definition eines einzelnen Status und führt sie aus. Sie können einen Zustand testen, ohne einen Zustandsautomaten zu erstellen oder einen vorhandenen Zustandsautomaten zu aktualisieren.

Mit der TestState API können Sie Folgendes testen:

- Der Eingabe- und Ausgabeverarbeitungsdatenfluss eines Zustands .
- Eine AWS-Service Integration mit anderen AWS-Services Anforderungen und Antworten
- Eine HTTP-Aufgabenanforderung und -antwort

Um einen Status zu testen, können Sie auch die Step Functions Konsole , AWS Command Line Interface (AWS CLI)oder das SDK verwenden.

Die API übernimmt eine IAM`TestState`-Rolle, die die erforderlichen IAM Berechtigungen für die Ressourcen enthalten muss, auf die Ihr Status zugreift. Informationen zu den Berechtigungen, die ein Status möglicherweise benötigt, finden Sie unter IAM -Berechtigungen für die Verwendung der TestState API.

Themen

Überlegungen zur Verwendung der TestState API

Mit der TestState API können Sie jeweils nur einen Status testen. Zu den Status, die Sie testen können, gehören die folgenden:

- Alle Aufgabentypen, außer Aktivitäten
- Pass
- Wait
- Choice
- Succeed
- Fehler

Beachten Sie bei der Verwendung der `TestState` API die folgenden Überlegungen.

- Die TestState API bietet keine Unterstützung für Folgendes:
  - Status der Aufgabe gibt an, die die folgenden Ressourcentypen verwenden:
    - Aktivität
    - Serviceintegrationsmuster vom Typ `.sync` oder `.waitForTaskToken`
  - Parallel Status
  - Zuordnung Status

- Ein Test kann bis zu fünf Minuten lang ausgeführt werden. Wenn ein Test diese Dauer überschreitet, schlägt er mit dem `States.Timeout` Fehler fehl.

Verwenden von Inspektionsstufen in der TestState API

Um einen Status mithilfe der TestState API zu testen, geben Sie die Definition dieses Status an. Der Test gibt dann eine Ausgabe zurück. Für jeden Status können Sie den Detailumfang angeben, den Sie in den Testergebnissen anzeigen möchten. Diese Details enthalten zusätzliche Informationen über den Status, den Sie testen. Wenn Sie beispielsweise Eingabe- und Ausgabedatenverarbeitungsfilter wie `InputPath` oder `ResultPath` in einem -Zustand verwendet haben, können Sie die Zwischen- und Enddatenverarbeitungsergebnisse anzeigen.

Step Functions bietet die folgenden Ebenen, um die Details anzugeben, die Sie anzeigen möchten:

- INFO
- FEHLER
- TRACE

All diese Ebenen geben auch die `nextState` Felder `status` und zurück. `status` gibt den Status der Zustandsausführung an. Zum Beispiel , `SUCCEEDEDFAILEDRETRIABLE`, und `CAUGHT_ERROR`. `nextState` gibt den Namen des nächsten Status an, in den gewechselt werden soll. Wenn Sie in Ihrer Definition keinen nächsten Status definiert haben, gibt dieses Feld einen leeren Wert zurück.

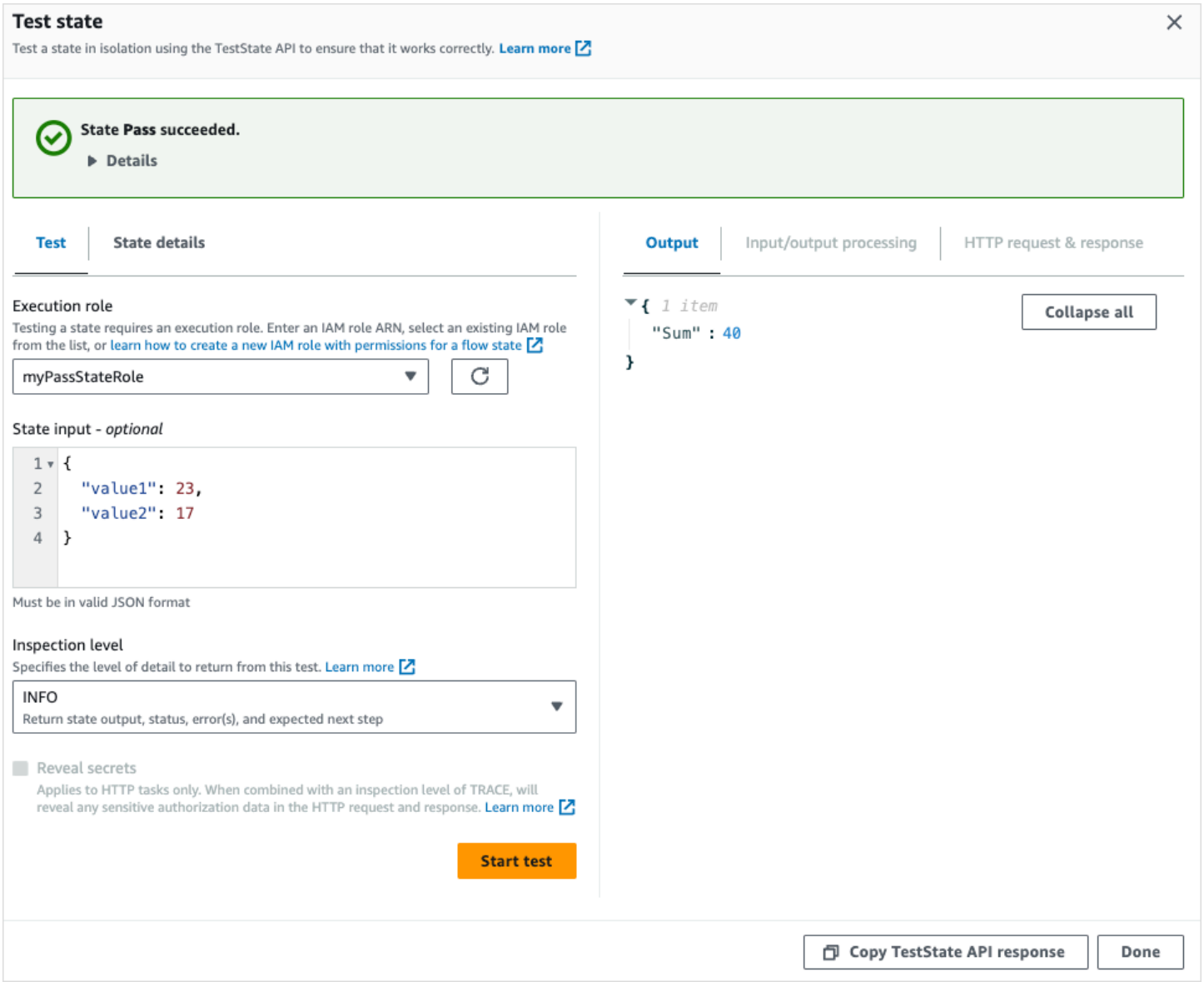
Informationen zum Testen eines Zustands mit diesen Prüfstufen in der Step FunctionsKonsole und AWS CLIfinden Sie unter Testen eines Zustands (Konsole) und Testen eines Zustands mit AWS CLI.

INFO-inspectionLevel

Wenn der Test erfolgreich ist, zeigt diese Stufe die Statusausgabe an. Wenn der Test fehlschlägt, zeigt diese Stufe die Fehlerausgabe an. Standardmäßig Step Functions legt die Überprüfungsstufe auf INFO fest, wenn Sie keine Stufe angeben.

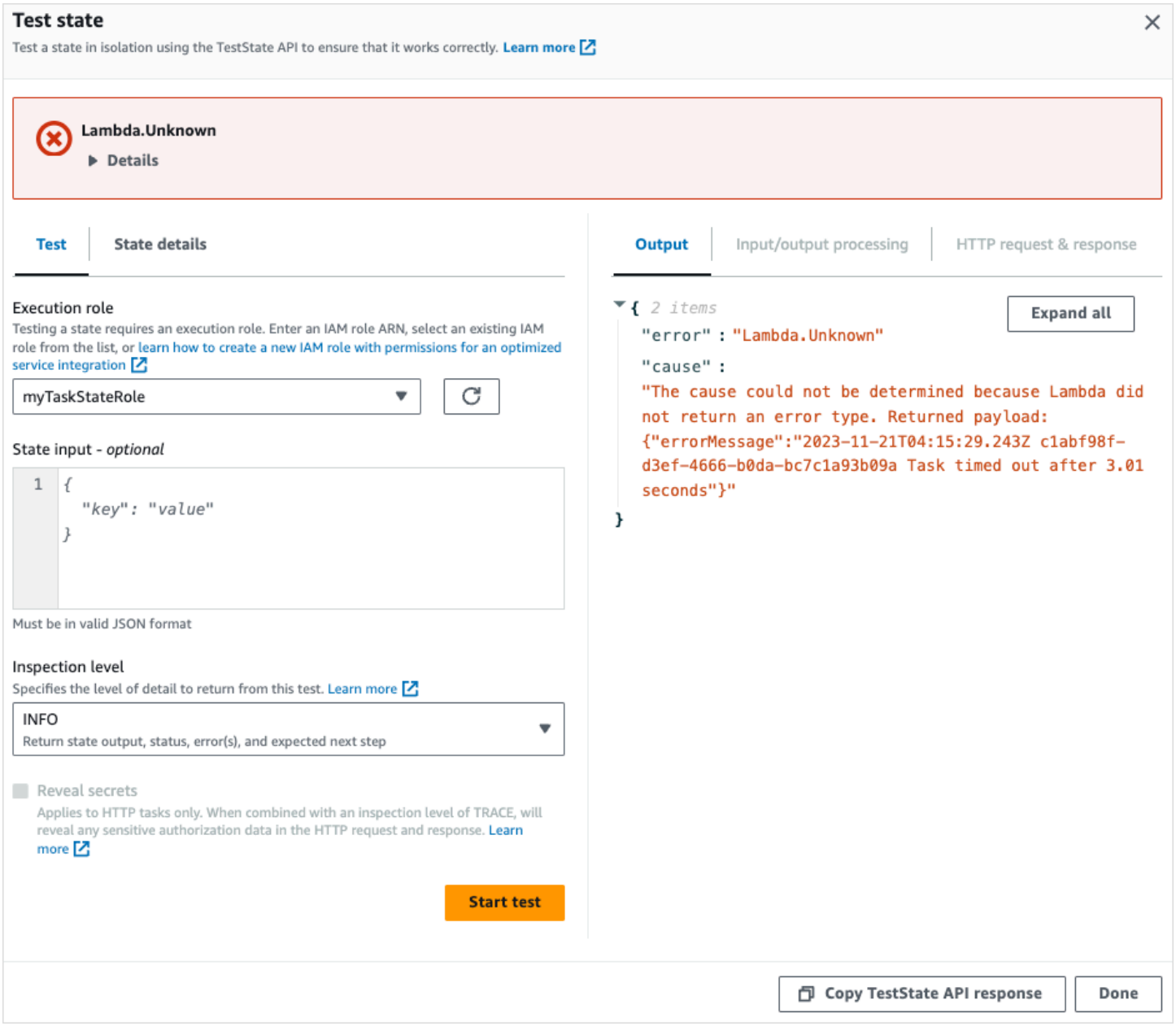
Beispiel für einen Test mit einer INFO-Ebene, die erfolgreich ist

Die folgende Abbildung zeigt einen Test auf einen erfolgreichen Status „Bestanden“. Die Überprüfungsstufe für diesen Zustand ist auf INFO gesetzt und die Ausgabe für den Zustand wird auf der Registerkarte Ausgabe angezeigt.



Beispiel für einen Test mit einer INFO-Ebene, die fehlschlägt

Die folgende Abbildung zeigt einen Test, der für einen Aufgabenstatus fehlgeschlagen ist, wenn die Überprüfungsstufe auf INFO gesetzt ist. Auf der Registerkarte Ausgabe wird die Fehlerausgabe angezeigt, die den Fehlernamen und eine detaillierte Erklärung der Ursache für diesen Fehler enthält.



DEBUG-inspectionLevel

Wenn der Test erfolgreich ist, zeigt diese Stufe die Statusausgabe und das Ergebnis der Eingabe- und Ausgabedatenverarbeitung an.

Wenn der Test fehlschlägt, zeigt diese Stufe die Fehlerausgabe an. Diese Stufe zeigt die Zwischenergebnisse der Datenverarbeitung bis zum Zeitpunkt des Fehlers an. Angenommen, Sie haben einen Aufgabenstatus getestet, der eine Lambda Funktion aufruft. Angenommen, Sie hätten die OutputPath Filter InputPath, ParameterResultPath, und auf den Aufgabenstatus angewendet. Angenommen, der Aufruf ist fehlgeschlagen. In diesem Fall zeigt die DEBUG Ebene

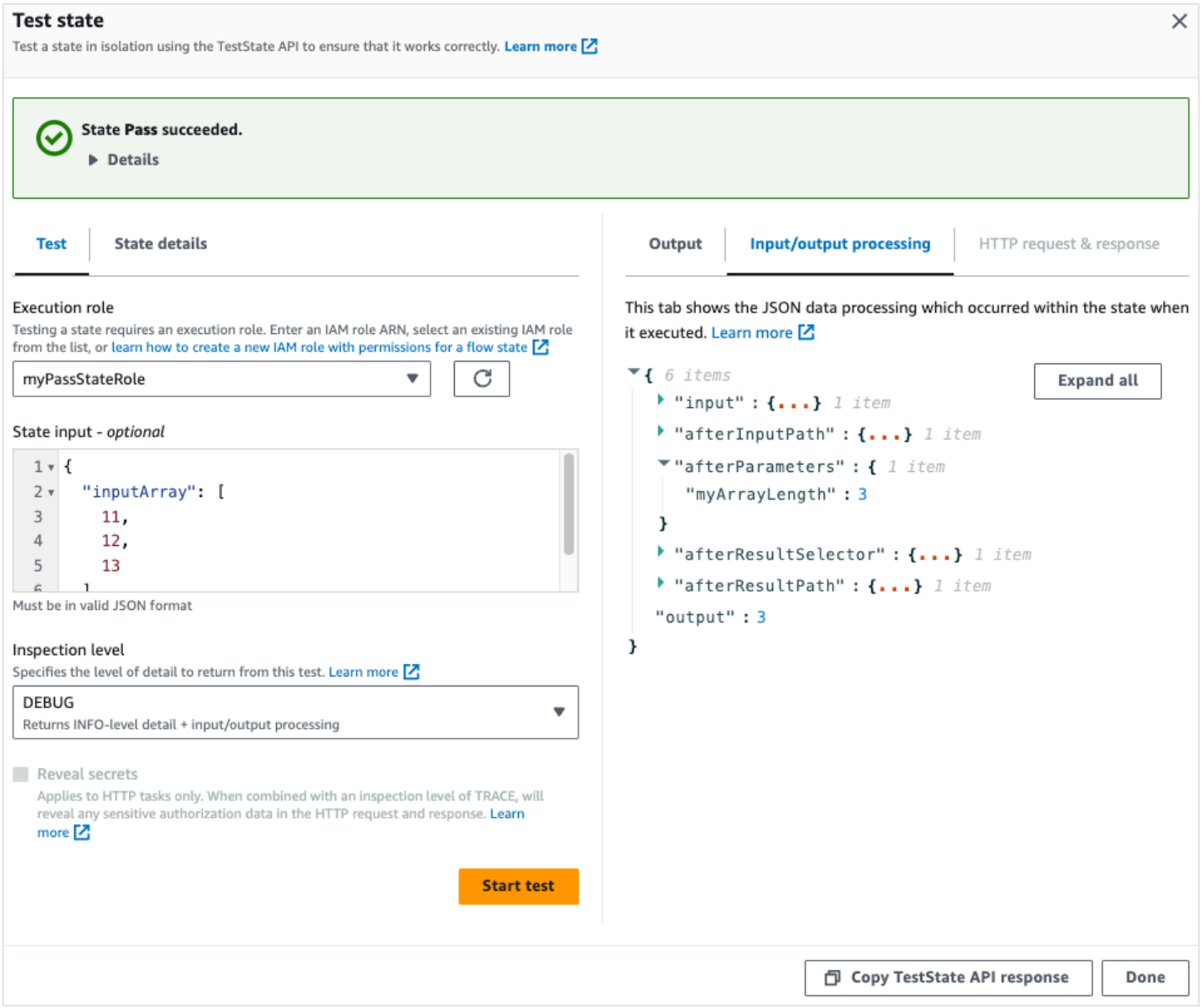
Datenverarbeitungsergebnisse basierend auf der Anwendung der Filter in der folgenden Reihenfolge an:

- `input` – Rohstatuseingabe
- `afterInputPath` – Eingabe nach Step Functions wendet den `InputPath` Filter an.
- `afterParameters` – Die effektive Eingabe nach Step Functions wendet den `Parameters` Filter an.

Die in dieser Ebene verfügbaren Diagnoseinformationen können Ihnen helfen, Probleme im Zusammenhang mit einer Serviceintegration oder einem Eingabe- und Ausgabedatenverarbeitungsablauf zu beheben, den Sie möglicherweise definiert haben.

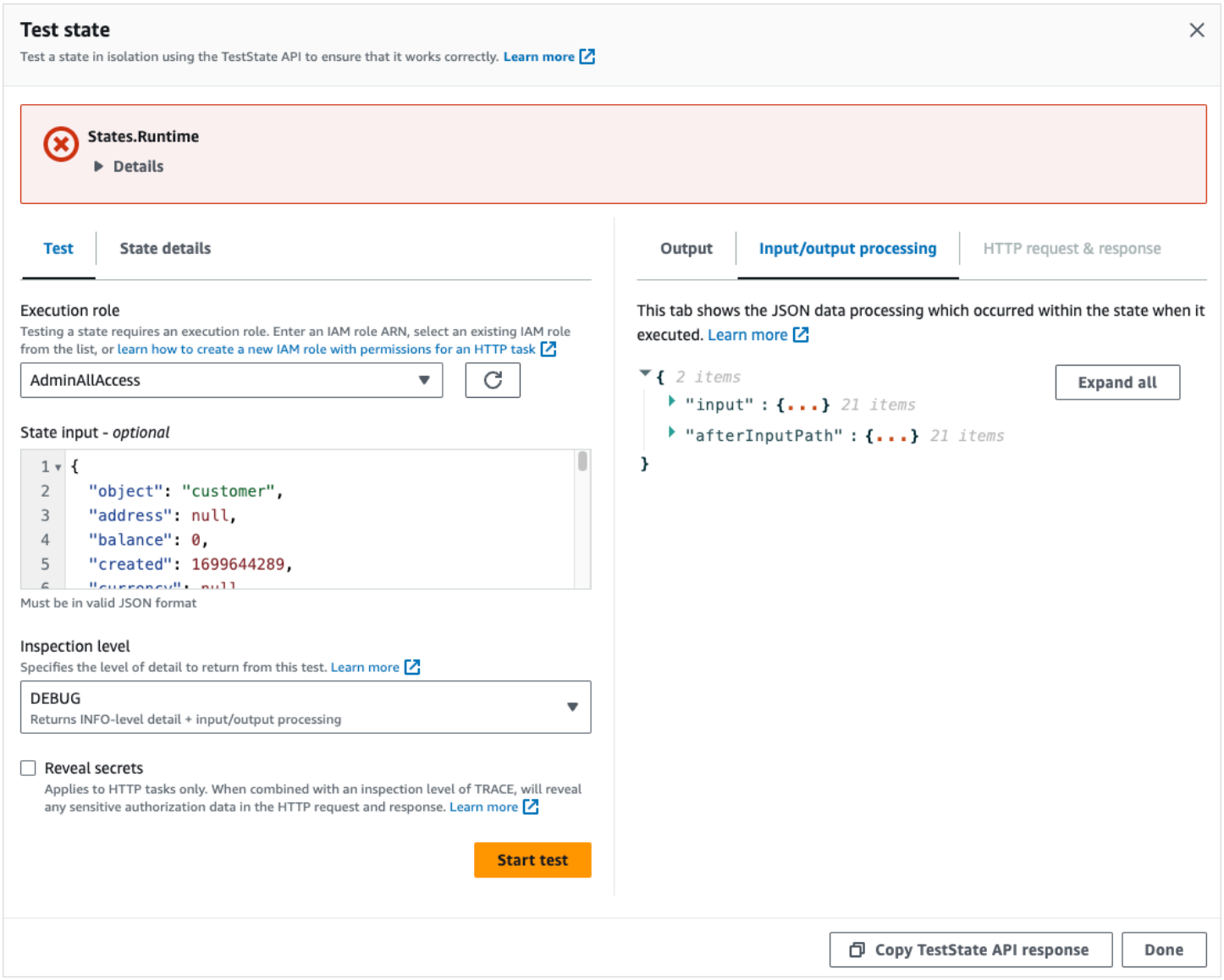
Beispiel für einen Test mit einer erfolgreichen DEBUG-Ebene

Die folgende Abbildung zeigt einen Test auf einen erfolgreichen Status „Bestanden“. Die Überprüfungsstufe für diesen Zustand ist auf DEBUG gesetzt. Die Registerkarte Eingabe-/Ausgabeverarbeitung in der folgenden Abbildung zeigt das Ergebnis der Anwendung von `Parameters` auf die für diesen Status bereitgestellte Eingabe.



Beispiel für einen Test mit einer DEBUG-Ebene, die fehlschlägt

Die folgende Abbildung zeigt einen Test, der für einen Aufgabenstatus fehlgeschlagen ist, wenn die Überprüfungsstufe auf DEBUG gesetzt ist. Die Registerkarte Eingabe-/Ausgabeverarbeitung in der folgenden Abbildung zeigt das Ergebnis der Eingabe- und Ausgabedatenverarbeitung für den Status bis zum Zeitpunkt ihres Fehlers.



TRACE-inspectionLevel

Step Functions stellt die TRACE-Ebene bereit, um eine HTTP-Aufgabe zu testen. Diese Stufe gibt Informationen über die HTTP-Anforderung zurück, die Step Functions sendet, und die Antwort, die eine Drittanbieter-API zurückgibt. Die Antwort kann Informationen wie Header und Anforderungstext enthalten. Darüber hinaus können Sie die Statusausgabe und das Ergebnis der Eingabe- und Ausgabedatenverarbeitung auf dieser Ebene anzeigen.

Wenn der Test fehlschlägt, zeigt diese Stufe die Fehlerausgabe an.

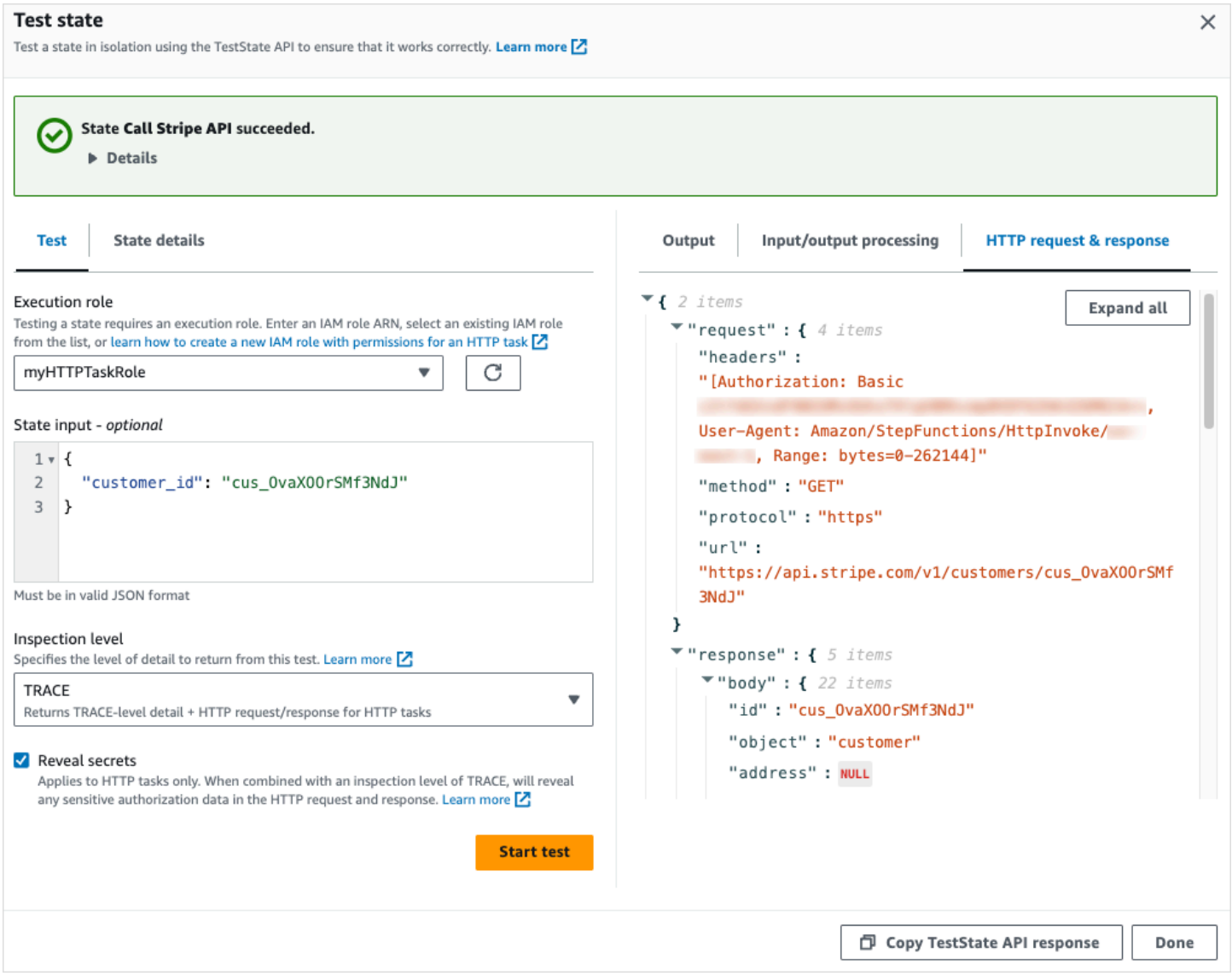
Diese Stufe gilt nur für HTTP Task. Step Functions gibt einen Fehler aus, wenn Sie diese Stufe für andere Statustypen verwenden.

Wenn Sie die Überprüfungsstufe auf TRACE setzen, können Sie auch Geheimnisse anzeigen, die in der EventBridge Verbindung enthalten sind. Dazu müssen Sie den `revealSecrets` Parameter `true` in der TestState API auf setzen. Darüber hinaus müssen Sie sicherstellen, dass der IAM Benutzer, der die TestState API aufruft, über die Berechtigung für die `states:RevealSecrets` Aktion verfügt. Ein Beispiel für eine IAM Richtlinie, die die `states:RevealSecrets` Berechtigung festlegt, finden Sie unter IAM -Berechtigungen für die Verwendung der TestState API. Ohne diese Berechtigung Step Functions löst einen Fehler aufgrund Zugriffsverweigerung aus.

Wenn Sie den `revealSecrets` Parameter auf setzen`false`, Step Functions lässt alle Secrets in den HTTP-Anforderungs- und Antwortdaten weg.

Beispiel für einen Test mit TRACE-Stufe, die erfolgreich ist

Die folgende Abbildung zeigt einen Test für eine erfolgreiche HTTP-Aufgabe. Die Überprüfungsstufe für diesen Zustand ist auf TRACE gesetzt. Die Registerkarte HTTP-Anfrage und -Antwort in der folgenden Abbildung zeigt das Ergebnis des API-Aufrufs eines Drittanbieters.



IAM -Berechtigungen für die Verwendung der TestState API

Der IAM Benutzer, der die `TestState` API aufruft, muss über Berechtigungen zum Ausführen der `iam:PassRole` Aktionen `states:TestState` und verfügen. Wenn Sie den Parameter revealSecrets auf setzen`true`, müssen Sie außerdem sicherstellen, dass der IAM Benutzer über Berechtigungen zum Ausführen der `states:RevealSecrets` Aktion verfügt. Ohne diese Berechtigung Step Functions löst einen Fehler aufgrund Zugriffsverweigerung aus.

Sie müssen auch sicherstellen, dass Ihre Ausführungsrolle die erforderlichen IAM Berechtigungen für die Ressourcen enthält, auf die Ihr Status zugreift. Informationen zu den Berechtigungen, die ein Status benötigt, finden Sie unter Verwalten von Ausführungsrollen.

Im folgenden IAM Richtlinienbeispiel werden die `states:RevealSecrets` Berechtigungen `states:TestStateiam:PassRole`, und festgelegt.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:TestState",
        "states:RevealSecrets",
        "iam:PassRole"
      ],
      "Resource": "*"
    }
  ]
}
```

Testen eines Zustands (Konsole)

Sie können einen Status in der Konsole testen und den Statusausgabe- oder Eingabe- und Ausgabedatenverarbeitungsablauf überprüfen. Für eine HTTP-Aufgabe können Sie die unformatierte HTTP-Anfrage und -Antwort testen.

So testen Sie einen Status

1. Öffnen Sie die Step-Functions-Konsole .
2. Wählen Sie Zustandsautomat erstellen, um mit der Erstellung eines Zustandsautomaten zu beginnen, oder wählen Sie einen vorhandenen Zustandsautomaten aus.
3. Wählen Sie in der Entwurfsmodus von Workflow Studio einen Status aus, den Sie testen möchten.
4. Wählen Sie Teststatus im Inspector Bereich von Workflow Studio aus.
5. Gehen Sie im Dialogfeld Teststatus wie folgt vor:

    a. Wählen Sie für Ausführungsrolle eine Ausführungsrolle aus, um den Status zu testen. Stellen Sie sicher, dass Sie über die erforderlichen IAM Berechtigungen für den Status verfügen, den Sie testen möchten.

    b. (Optional) Geben Sie alle JSON-Eingaben an, die Ihr ausgewählter Status für den Test benötigt.

c. Wählen Sie für Überprüfungsstufe eine der folgenden Optionen basierend auf den Werten aus, die Sie anzeigen möchten:

- INFO – Zeigt die Statusausgabe auf der Registerkarte Ausgabe an, wenn der Test erfolgreich ist. Wenn der Test fehlschlägt, zeigt INFO die Fehlerausgabe an, die den Fehlernamen und eine detaillierte Erklärung der Ursache für diesen Fehler enthält. Standardmäßig Step Functions legt die Überprüfungsstufe auf INFO fest, wenn Sie keine Stufe auswählen.
- DEBUG – Zeigt die Zustandsausgabe und das Ergebnis der Eingabe- und Ausgabedatenverarbeitung an, wenn der Test erfolgreich ist. Wenn der Test fehlschlägt, zeigt DEBUG die Fehlerausgabe an, die den Fehlernamen und eine detaillierte Erklärung der Ursache für diesen Fehler enthält.
- TRACE – Zeigt die unformatierte HTTP-Anforderung und -Antwort an und ist nützlich für die Überprüfung von Headern, Abfrageparametern und anderen API-spezifischen Details. Diese Option ist nur für die HTTP-Aufgabe verfügbar.

  Optional können Sie wählen, ob Sie Geheimnisse anzeigen auswählen möchten. In Kombination mit TRACE können Sie mit dieser Einstellung die sensiblen Daten sehen, die die EventBridge Verbindung einfügt, z. B. API-Schlüssel. Die IAM Benutzeridentität, die Sie für den Zugriff auf die Konsole verwenden, muss über die Berechtigung zum Ausführen der `states:RevealSecrets` Aktion verfügen. Ohne diese Berechtigung Step Functions gibt einen Fehler aufgrund Zugriffsverweigerung aus, wenn Sie den Test starten. Ein Beispiel für eine -IAMRichtlinie, die die `states:RevealSecrets` Berechtigung festlegt, finden Sie unter IAM -Berechtigungen für die Verwendung der TestState API.

d. Wählen Sie Test starten aus.

Testen eines Zustands mit AWS CLI

Sie können einen unterstützten Status mithilfe der TestState API in der testenAWS CLI. Diese API akzeptiert die Definition eines Zustands und führt sie aus.

Für jeden Status können Sie den Detailumfang angeben, den Sie in den Testergebnissen anzeigen möchten. Diese Details enthalten zusätzliche Informationen zur Ausführung des Status, einschließlich des Ergebnisses der Eingabe- und Ausgabedatenverarbeitung sowie HTTP-Anfrage- und Antwortinformationen. Die folgenden Beispiele zeigen die verschiedenen Inspektionsstufen, die

Sie für die TestState API angeben können. Denken Sie daran, den *kursiv gedruckten* Text durch Ihre ressourcenspezifischen Informationen zu ersetzen.

Dieser Abschnitt enthält die folgenden Beispiele, die beschreiben, wie Sie die verschiedenen Inspektionsstufen verwenden können, die in der Step Functions bietetAWS CLI:

- Verwenden von INFO inspectionLevel
- Verwenden von DEBUG inspectionLevel
- Verwenden von TRACE inspectionLevel
- Verwenden des Hilfsprogramms jq in AWS CLI zum Filtern und Drucken der HTTP-Antwort, die die TestState API zurückgibt

Beispiel 1: Verwenden von INFO inspectionLevel zum Testen eines Choice-Status

Um einen Status mit der `INFO` inspectionLevel in der zu testen AWS CLI, führen Sie den `test-state` Befehl aus, wie im folgenden Beispiel gezeigt.

```
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}'
```

In diesem Beispiel wird ein Choice-Status verwendet, um den Ausführungspfad für den Status basierend auf der von Ihnen angegebenen numerischen Eingabe zu bestimmen. Standardmäßig Step Functions setzt die `inspectionLevel` auf , `INFO` wenn Sie keine Ebene festlegen.

Step Functions gibt die folgende Ausgabe zurück.

```
{
    "output": "{\"number\": 2}",
    "nextState": "Equals 2",
    "status": "SUCCEEDED"
}
```

Beispiel 2: Verwenden von DEBUG inspectionLevel zum Debuggen der Eingabe- und Ausgabedatenverarbeitung im Status „Bestanden“

Um einen Status mit der DEBUG inspectionLevel in der zu testen AWS CLI, führen Sie den `test-state` Befehl aus, wie im folgenden Beispiel gezeigt.

```
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
```

In diesem Beispiel wird ein -PassZustand verwendet, um zu zeigen, wie Eingabe-JSON-Daten mithilfe der Eingabe- und Ausgabedatenverarbeitungsfilter Step Functions filtert und manipuliert. In diesem Beispiel werden die folgenden Filter verwendet: `InputPath`, `ParameterResultPath`, und `OutputPath`.

Step Functions gibt die folgende Ausgabe zurück.

```
{
    "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"
}
```

Beispiel 3: Verwenden von TRACE inspectionLevel und revealSecrets zur Überprüfung der HTTP-Anfrage, die an eine Drittanbieter-API gesendet wird

Um eine HTTP-Aufgabe mit dem TRACE inspectionLevel zusammen mit dem Parameter revealSecrets in der zu testen AWS CLI, führen Sie den `test-state` Befehl aus, wie im folgenden Beispiel gezeigt.

```
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
```

In diesem Beispiel wird getestet, ob die HTTP-Aufgabe die angegebene Drittanbieter-API aufruft`https://httpbin.org/`. Außerdem werden die HTTP-Anforderungs- und Antwortdaten für den API-Aufruf angezeigt.

```
{
    "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"
}
```

Beispiel 4: Verwenden des jq-Hilfsprogramms zum Filtern und Drucken der Antwort, die die TestState API zurückgibt

Die TestState API gibt JSON-Daten als Escape-Zeichenfolgen in ihrer Antwort zurück. Das folgende AWS CLI Beispiel erweitert Beispiel 3 und verwendet das `jq` Dienstprogramm , um die HTTP-Antwort, die die TestState API zurückgibt, in einem für Menschen lesbaren Format zu filtern und zu drucken. Informationen zu `jq` und dessen Installationsanweisungen finden Sie unter jq auf 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'
```

Das folgende Beispiel zeigt die Ausgabe in einem für Menschen lesbaren Format.

```
{
```

```
  "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"
}
```

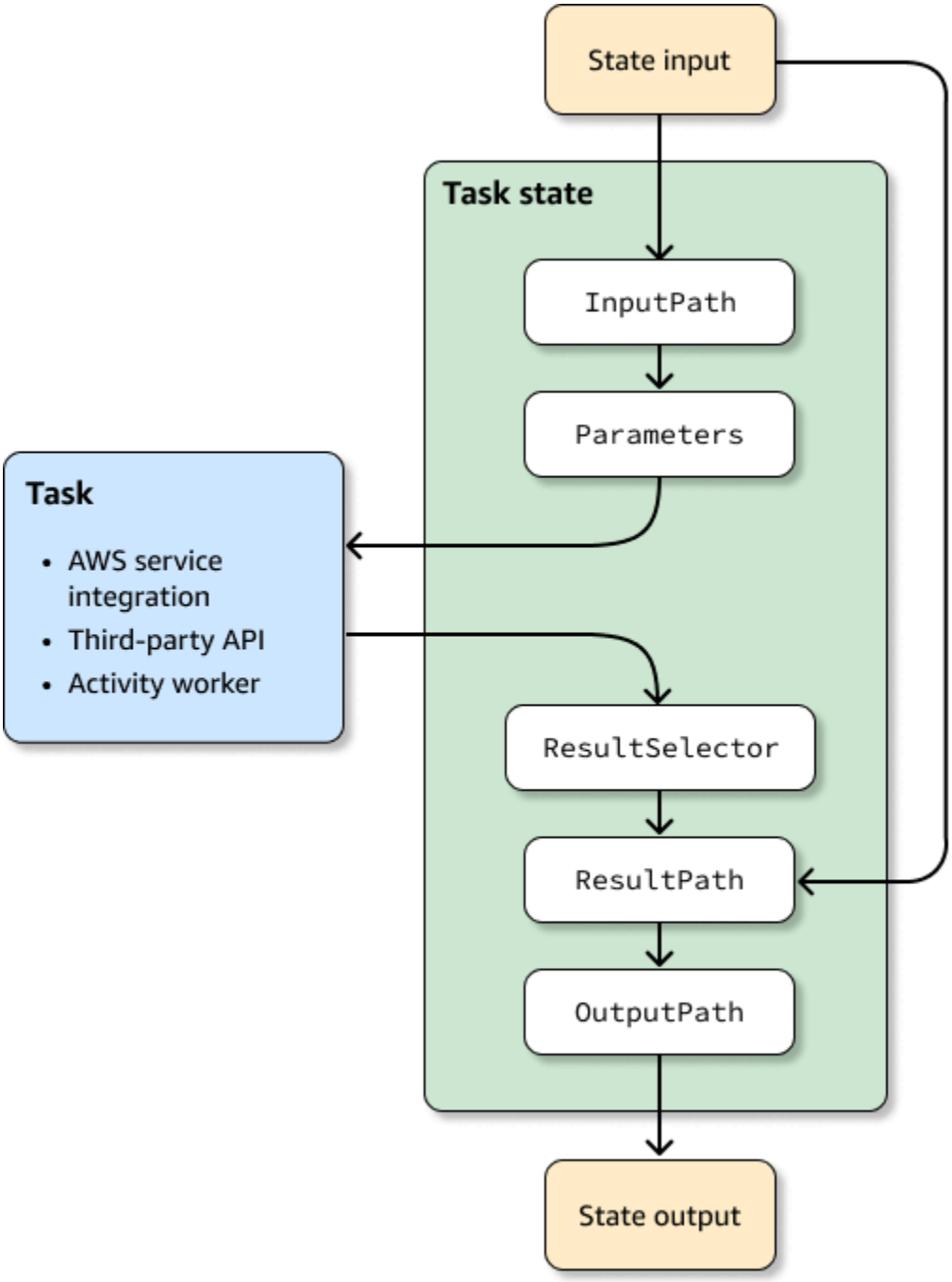

Testen und Debuggen des Eingabe- und Ausgabedatenflusses

Die `TestState` API ist hilfreich, um die Daten zu testen und zu debuggen, die durch Ihren Workflow fließen. Dieser Abschnitt enthält einige wichtige Konzepte und erklärt, wie Sie die TestState für diesen Zweck verwenden.

Die wichtigsten Konzepte

In wird Step Functionsder Prozess des Filterns und Bearbeitens von JSON-Daten während des Durchlaufens der Zustände in Ihrem Zustandsautomaten als Eingabe- und Ausgabeverarbeitung bezeichnet. Weitere Information zur Funktionsweise finden Sie unter Eingabe- und Ausgabeverarbeitung in Step Functions.

Alle Zustandstypen in (Amazon States LanguageASL) (Aufgabe, Parallel, Zuordnung, Pass, Warten, Auswahl, Erfolgreich und Fehlgeschlagen) teilen sich eine Reihe gemeinsamer Felder zum Filtern und Bearbeiten der JSON-Daten, die sie durchlaufen. Diese Felder sind: InputPath, Parameter, ResultSelectorResultPath, und OutputPath. Die Unterstützung für jedes Feld variiert je nach Status . Zur Laufzeit Step Functions wendet jedes Feld in einer bestimmten Reihenfolge an. Das folgende Diagramm zeigt die Reihenfolge, in der diese Felder auf die Daten innerhalb eines Aufgabenstatus angewendet werden:



In der folgenden Liste wird die Reihenfolge der Anwendung der im Diagramm gezeigten Eingabe- und Ausgabeverarbeitungsfelder beschrieben.

1. Die Zustandseingabe ist die JSON-Daten, die aus einem früheren Zustand an den aktuellen Zustand übergeben wurden.
2. InputPath filtert einen Teil der Rohstatuseingabe.
3. Parameter konfiguriert den Satz von Werten, die an die Aufgabe übergeben werden sollen.

4. Die Aufgabe führt Arbeit aus und gibt ein Ergebnis zurück.
5. ResultSelector wählt einen Satz von Werten aus, die aus dem Aufgabenergebnis beibehalten werden sollen.
6. ResultPath kombiniert das Ergebnis mit der Rohstatuseingabe oder ersetzt das Ergebnis durch es.
7. OutputPath filtert einen Teil der Ausgabe, der an den nächsten Status übergeben werden soll.
8. Die Zustandsausgabe ist die JSON-Daten, die vom aktuellen Status an den nächsten Status übergeben werden.

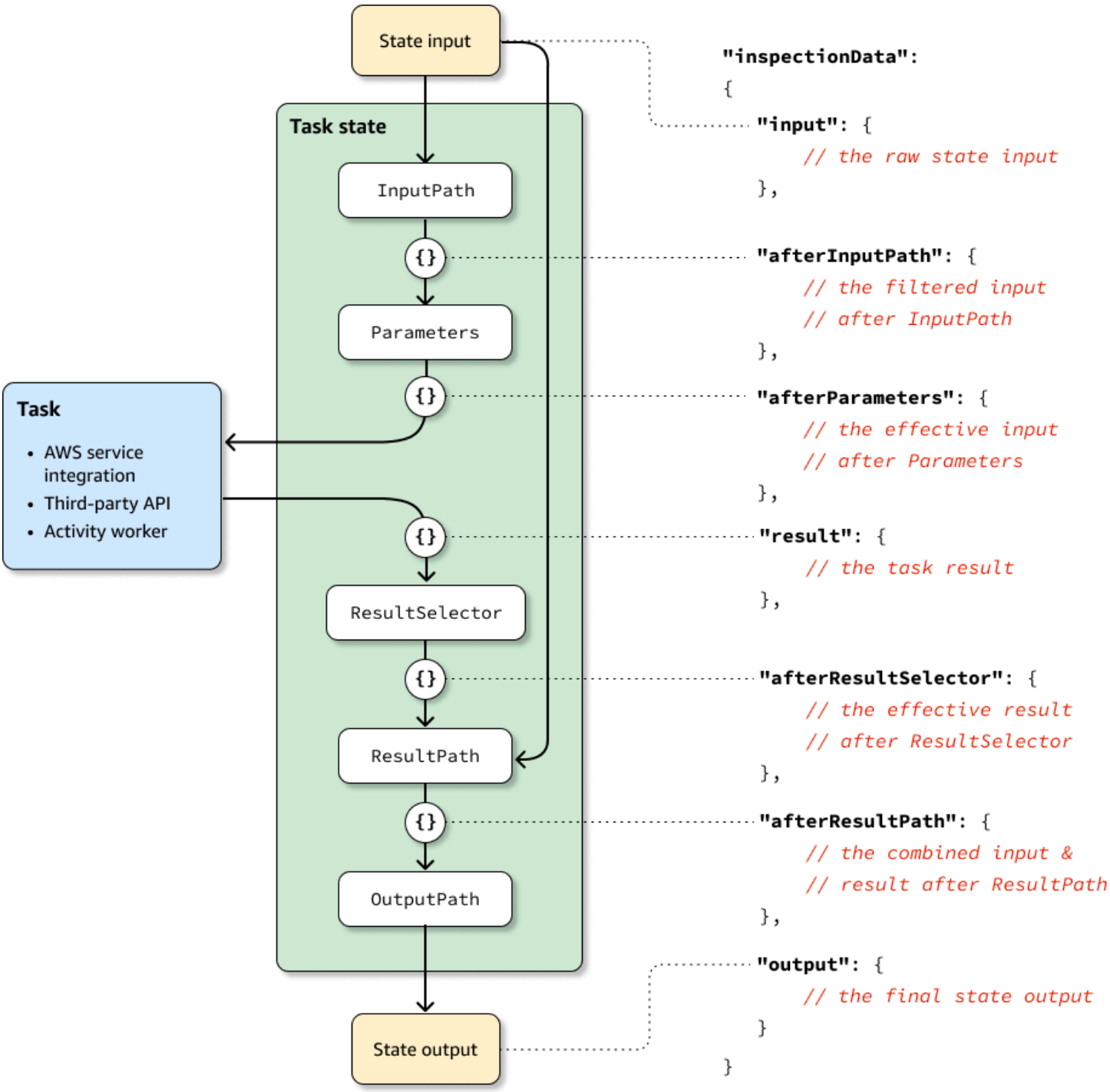
Diese Eingabe- und Ausgabeverarbeitungsfelder sind optional. Wenn Sie keines dieser Felder in Ihrer Statusdefinition verwenden, verbraucht die Aufgabe die Rohstatuseingabe und gibt das Aufgabenergebnis als Statusausgabe zurück.

Verwenden von TestState zur Überprüfung der Eingabe- und Ausgabeverarbeitung

Wenn Sie die `TestState` API aufrufen und den `inspectionLevel` Parameter auf setzen`DEBUG`, enthält die API-Antwort ein Objekt namens `inspectionData`. Dieses Objekt enthält Felder, mit denen Sie überprüfen können, wie Daten beim Ausführen innerhalb des -Status gefiltert oder manipuliert wurden. Das folgende Beispiel zeigt das -`inspectionData`Objekt für einen Aufgabenstatus.

```
"inspectionData":   {
  "input": string,
  "afterInputPath": string,
  "afterParameters": string,
  "result": string,
  "afterResultSelector": string,
  "afterResultPath": string,
  "output": string
}
```

In diesem Beispiel zeigt jedes Feld, das das `after` Präfix enthält, die Daten an, nachdem ein bestimmtes Feld angewendet wurde. Beispielsweise `afterInputPath` zeigt die Auswirkung der Anwendung des `InputPath` Felds zum Filtern der Rohstatuseingabe. Das folgende Diagramm ordnet jedes ASL-Definitionsfeld dem entsprechenden Feld im -`inspectionData`Objekt zu:



Beispiele für die Verwendung der TestState -API zum Debuggen der Eingabe- und Ausgabeverarbeitung finden Sie im Folgenden:

- Testen eines Zustands mit der DEBUG-Inspektionsstufe in der Step Functions Konsole
- Testen eines Zustands mit der DEBUG-Inspektionsebene in der AWS CLI

Zustandsmaschinen lokal testen

AWSStep Functions Local ist eine herunterladbare Version von Step Functions, mit der Sie Anwendungen mithilfe einer Version von Step Functions entwickeln und testen können, die in Ihrer eigenen Entwicklungsumgebung ausgeführt wird. Die lokale Version von Step Functions kann AWS Lambda Funktionen aufrufen, AWS sowohl in als auch während der lokalen Ausführung. Sie können auch andere unterstützte AWS-Services koordinieren.

> **Note**
>
> Step Functions Local verwendet Dummy-Konten, um zu funktionieren.

Während Sie Step Functions Local ausführen, können Sie eine der folgenden Methoden verwenden, um Service-Integrationen aufzurufen:

- Konfiguration lokaler Endpunkte für AWS Lambda und andere Dienste. Informationen zu den unterstützten Endpunkten finden Sie unter. Konfigurationsoptionen für Step Functions lokal einrichten
- Direktes Aufrufen eines AWS Dienstes von Step Functions Local aus.
- Die Antwort von Serviceintegrationen verspotten. Hinweise zur Verwendung von nachgemachten Serviceintegrationen finden Sie unter. Verwenden von Mocked-Service-Integrationen

AWSStep Functions Local ist als JAR-Paket oder als eigenständiges Docker-Image verfügbar, das unter Microsoft Windows, Linux, macOS und anderen Plattformen läuft, die Java oder Docker unterstützen.

> **Warning**
>
> Die herunterladbare Version von AWS Step Functions soll nur zum Testen verwendet werden und sollte niemals zur Verarbeitung vertraulicher Informationen verwendet werden.

> **Tip**
>
> Stellen Sie sicher, dass Sie Step Functions Local Version 1.12.0 oder höher verwenden, um alle systemeigenen Funktionen in Ihre Workflows integrieren zu können.

In den folgenden Themen wird beschrieben, wie Sie Step Functions Local mithilfe von Docker und JAR-Datei einrichten und Step Functions Local ausführen könnenAWS Lambda, um mit AWS Serverless Application Model (AWS SAM) CLI Local oder anderen unterstützten Diensten zu arbeiten.

Themen

Step Functions lokal (herunterladbare Version) und Docker einrichten

Mit dem Docker-Image Step Functions Local können Sie schnell mit Step Functions Local beginnen, indem Sie ein Docker-Image mit allen erforderlichen Abhängigkeiten verwenden. Das Docker-Image ermöglicht es Ihnen, Step Functions Local in Ihre containerisierten Builds und als Teil Ihrer kontinuierlichen Integrationstests aufzunehmen.

Das Docker-Image für Step Functions Local finden Sie unter https://hub.docker.com/r/amazon/ aws-stepfunctions-local, oder geben Sie den folgenden `pull` Docker-Befehl ein.

```
docker pull amazon/aws-stepfunctions-local
```

Führen Sie den folgenden Docker-Befehl aus, um die herunterladbare Version von Step Functions auf `run` Docker zu starten

```
docker run -p 8083:8083 amazon/aws-stepfunctions-local
```

Um mit AWS Lambda oder anderen unterstützten Diensten zu interagieren, müssen Sie zuerst Ihre Anmeldeinformationen und andere Konfigurationsoptionen konfigurieren. Weitere Informationen finden Sie unter den folgenden Themen:

- Konfigurationsoptionen für Step Functions lokal einrichten
- Anmeldeinformationen und Konfiguration für Docker

Step Functions lokal einrichten (herunterladbare Version) — Java-Version

Die herunterladbare Version von AWS Step Functions wird als ausführbare JAR-Datei und als Docker-Image bereitgestellt. Die Java-Anwendung kann auf Windows-, Linux-, macOS- und anderen Plattformen, die Java unterstützen, ausgeführt werden. Zusätzlich zu Java müssen Sie die AWS Command Line Interface (AWS CLI) installieren. Informationen zur Installation und Konfiguration von finden Sie im AWS Command Line InterfaceBenutzerhandbuch. AWS CLI

So richten Sie Step Functions auf Ihrem Computer ein und führen sie aus

1. Laden Sie Step Functions über die folgenden Links herunter.

Download-Links	Prüfsumme
.tar.gz	.tar.gz.md5
.zip	.zip.md5

2. Extrahieren Sie die `.zip`-Datei.
3. Testen Sie den Download und lesen Sie die Versionsinformationen.

```
$ java -jar StepFunctionsLocal.jar -v
Step Function Local
Version: 1.0.0
Build: 2019-01-21
```

4. (Optional) Zeigen Sie eine Liste der verfügbaren Befehle an.

```
$ java -jar StepFunctionsLocal.jar -h
```

5. Um Step Functions auf Ihrem Computer zu starten, öffnen Sie eine Befehlszeile, navigieren Sie zu dem Verzeichnis, in das Sie extrahiert haben`StepFunctionsLocal.jar`, und geben Sie den folgenden Befehl ein.

```
java -jar StepFunctionsLocal.jar
```

6. Verwenden Sie den `--endpoint-url` Parameter, um auf lokal ausgeführte Step-Funktionen zuzugreifen. Mit den würden Sie AWS CLI beispielsweise die Befehle Step Functions wie folgt angeben:

```
aws stepfunctions --endpoint-url http://localhost:8083 command
```

> **Note**
>
> Standardmäßig verwendet Step Functions Local ein lokales Testkonto und Anmeldeinformationen, und die AWS Region ist auf USA Ost (Nord-Virginia) eingestellt. Um Step Functions Local mit AWS Lambda oder anderen unterstützten Diensten verwenden zu können, müssen Sie Ihre Anmeldeinformationen und Region konfigurieren.
> Wenn Sie Express-Workflows mit Step Functions Local verwenden, wird der Ausführungsverlauf in einer Protokolldatei gespeichert. Es wird nicht in CloudWatch Logs protokolliert. Der Protokolldateipfad basiert auf dem ARN der CloudWatch Logs-Protokollgruppe, der beim Erstellen des lokalen Zustandsmaschinen bereitgestellt wurde. Die Protokolldatei wird `/aws/states/log-group-name/${execution_arn}.log` relativ zu dem Speicherort gespeichert, an dem Sie Step Functions Local ausführen. Wenn der Ausführungs-ARN beispielsweise folgenden Namen hat:
>
> ```
> arn:aws:states:us-east-1:123456789012:express:test:example-ExpressLogGroup-
> wJalrXUtnFEMI
> ```
>
> hat die Protokolldatei den Namen:
>
> ```
> aws/states/log-group-name/arn:aws:states:us-
> east-1:123456789012:express:test:example-ExpressLogGroup-wJalrXUtnFEMI.log
> ```

Konfigurationsoptionen für Step Functions lokal einrichten

Wenn Sie AWS Step Functions Local mithilfe der JAR-Datei starten, können Sie die Konfigurationsoptionen mithilfe von AWS Command Line Interface (AWS CLI) festlegen oder indem Sie sie in die Systemumgebung aufnehmen. Für Docker müssen Sie diese Optionen in einer Datei angeben, auf die Sie beim Start von Step Functions Local verweisen.

Konfigurationsoptionen

Wenn Sie den Step Functions Local-Container so konfigurieren, dass er einen Override-Endpunkt wie Lambda-Endpunkt und Batch-Endpunkt verwendet, und diesen Endpunkt aufrufen, verwendet

Step Functions Local die von Ihnen angegebenen Anmeldeinformationen nicht. Das Festlegen dieser Endpunkt-Overrides ist optional.

Option	Befehlszeile	Umgebung
Account	-account, --aws-account	AWS_ACCOUNT_ID
Region	-region, --aws-region	AWS_DEFAULT_REGION
Wartezeit-Skalierung	-waitTimeScale, --wait-time-scale	WAIT_TIME_SCALE
Lambda-Endpunkt	-Lambda-Endpunkt, --lambda-Endpunkt	LAMBDA_ENDPOINT
Stapelendpunkt	-Batch-Endpunkt, --batch-Endpunkt	BATCH_ENDPOINT
DynamoDB-Endpunkt	-DynamodBendPoint, --dynamodb-endpoint	DYNAMODB_ENDPOINT
ECS-Endpunkt	-ecsEndpoint, --ecs-endpoint	ECS_ENDPOINT
Glue-Endpunkt	-glue-Endpunkt, --glue-endpoint	GLUE_ENDPOINT
SageMaker Endpunkt	-sageMakerEndpoint, --sagemaker-endpoint	SAGE_MAKER_ENDPOINT
SQS-Endpunkt	-sqSendpoint, --sqs-endpoint	SQS_ENDPOINT
SNS-Endpunkt	-snSendPoint, --sns-endpoint	SNS_ENDPOINT
Step Functions Endpunkt	-stepFunctionsEndpoint, --step-functions-endpoint	STEP_FUNCTIONS_ENDPOINT

Anmeldeinformationen und Konfiguration für Docker

Um Step Functions Local für Docker zu konfigurieren, erstellen Sie die folgende Datei:`aws-stepfunctions-local-credentials.txt`.

Diese Datei enthält Ihre Anmeldeinformationen und andere Konfigurationsoptionen. Folgendes kann beim Erstellen der `aws-stepfunctions-local-credentials.txt` Datei als Vorlage verwendet werden.

```
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
```

Nachdem Sie Ihre Anmeldeinformationen und Konfigurationsoptionen unter konfiguriert habenaws-stepfunctions-local-credentials.txt, starten Sie Step Functions mit dem folgenden Befehl.

```
docker run -p 8083:8083 --env-file aws-stepfunctions-local-credentials.txt amazon/aws-stepfunctions-local
```

> **Note**
>
> Es wird empfohlen, den speziellen DNS-Namen zu verwendenhost.docker.internal, der in die interne IP-Adresse aufgelöst wird, die der Host verwendet, z. B. `http://host.docker.internal:8000` Weitere Informationen finden Sie in der Docker-Dokumentation für Mac und Windows unter Netzwerkfunktionen in Docker Desktop für Mac bzw. Netzwerkfunktionen in Docker Desktop für Windows.

Step Functions lokal auf Ihrem Computer ausführen

Verwenden Sie die lokale Version von Step Functions, um Zustandsmaschinen auf Ihrem Computer zu konfigurieren, zu entwickeln und zu testen.

Führen Sie eine HelloWorld Zustandsmaschine lokal aus

Nachdem Sie Step Functions lokal mit AWS Command Line Interface (AWS CLI) ausgeführt haben, können Sie eine State-Machine-Ausführung starten.

1. Erstellen Sie einen Zustandsmaschine aus dem, AWS CLI indem Sie die Zustandsmaschinen-Definition umgehen.

```
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**
>
> Das `role-arn` wird nicht für Step Functions Local verwendet, aber Sie müssen es mit der richtigen Syntax einbinden. Sie können den Amazon-Ressourcennamen (ARN) aus dem vorherigen Beispiel verwenden.

Wenn Sie die Zustandsmaschine erfolgreich erstellt haben, antwortet Step Functions mit dem Erstellungsdatum und dem State-Machine-ARN.

```
{
    "creationDate": 1548454198.202,
    "stateMachineArn": "arn:aws:states:us-
east-1:123456789012:stateMachine:HelloWorld"
}
```

2. Starten Sie eine Ausführung mithilfe des ARN des Zustandsautomaten, den Sie erstellt haben.

```
aws stepfunctions --endpoint-url http://localhost:8083 start-execution --state-
machine-arn arn:aws:states:us-east-1:123456789012:stateMachine:HelloWorld
```

Step Functions Lokal mit AWS SAM CLI Local

Sie können die lokale Version von Step Functions mit einer lokalen Version von verwendenAWS Lambda. Zur Konfiguration müssen Sie AWS SAM installieren und konfigurieren.

Informationen zum Konfigurieren und Ausführen von AWS SAM finden Sie unter den folgenden Themen:

- Einrichten von AWS SAM
- Starten von AWS SAM CLI Local

Wenn Lambda auf Ihrem lokalen System läuft, können Sie Step Functions Local starten. Starten Sie Step Functions Local aus dem Verzeichnis, in das Sie Ihre lokalen Step Functions Functions-JAR-Dateien extrahiert haben, und verwenden Sie den `--lambda-endpoint` Parameter, um den lokalen Lambda-Endpunkt zu konfigurieren.

```
java -jar StepFunctionsLocal.jar --lambda-endpoint http://127.0.0.1:3001 command
```

Weitere Hinweise zum Ausführen von Step Functions Local mit AWS Lambda finden Sie unterStep-Funktionen und AWS SAM CLI Local testen.

Step-Funktionen und AWS SAM CLI Local testen

Wenn Sie beide AWS Step Functions auf Ihrem lokalen Computer AWS Lambda ausführen, können Sie Ihre State-Maschine und Lambda-Funktionen testen, ohne Ihren Code darauf bereitstellen zu müssen. AWS

Weitere Informationen finden Sie unter den folgenden Themen:

- Zustandsmaschinen lokal testen
- Einrichten von AWS SAM

Themen

- Schritt 1: Einrichten von AWS SAM
- Schritt 2: Testen von AWS SAM CLI Local
- Schritt 3: Starten von AWS SAM CLI Local
- Schritt 4: Starten Sie Step Functions Local

- Schritt 5: Erstellen eines Zustandsautomaten, der auf die AWS SAM CLI Local-Funktion verweist
- Schritt 6: Die Ausführung des lokalen Zustandsautomaten starten

Schritt 1: Einrichten von AWS SAM

AWS Serverless Application Model (AWS SAM) CLI Local erfordert, dass die AWS Command Line Interface, AWS SAM und Docker installiert sind.

1. Installieren der AWS SAM-CLI.

   > **Note**
   >
   > Bevor Sie die AWS SAM CLI installieren können, müssen Sie die AWS CLI und Docker installieren. Weitere Informationen finden Sie in den Voraussetzungen für die Installation der AWS SAM CLI.

2. Gehen Sie die AWS SAM Schnellstart-Dokumentation durch. Beachten Sie die Schritte zum Ausführen folgender Aufgaben:

   1. Initialisieren der Anwendung
   2. Lokales Testen der Anwendung

   Dadurch wird ein `sam-app` Verzeichnis erstellt und eine Umgebung erstellt, die eine Python-basierte Hello World Lambda-Funktion enthält.

Schritt 2: Testen von AWS SAM CLI Local

Nachdem Sie die Hello World Lambda-Funktion installiert AWS SAM und erstellt haben, können Sie die Funktion testen. Geben Sie den folgenden Befehl im Verzeichnis `sam-app` ein:

```
sam local start-api
```

Dadurch wird eine lokale Instanz Ihrer Lambda-Funktion gestartet. Sie sollten eine Ausgabe sehen, die der folgenden ähnelt:

```
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)
```

Öffnen Sie einen Browser und geben Sie Folgendes ein:

```
http://127.0.0.1:3000/hello
```

Dadurch wird eine Antwort ausgegeben, die der folgenden ähnelt:

```
{"message": "hello world", "location": "72.21.198.66"}
```

Geben Sie STRG+C ein, um die Lambda-API zu beenden.

Schritt 3: Starten von AWS SAM CLI Local

Nachdem Sie nun getestet haben, ob die Funktion funktioniert, starten Sie AWS SAM CLI Local. Geben Sie den folgenden Befehl im Verzeichnis `sam-app` ein:

```
sam local start-lambda
```

Dadurch wird AWS SAM CLI Local gestartet und der zu verwendende Endpunkt bereitgestellt, ähnlich der folgenden Ausgabe:

```
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)
```

Schritt 4: Starten Sie Step Functions Local

JAR-Datei

Wenn Sie die `.jar` Dateiversion von Step Functions Local verwenden, starten Sie Step Functions und geben Sie den Lambda-Endpunkt an. Geben Sie in dem Verzeichnis, in dem Sie die `.jar` Dateien extrahiert haben, den folgenden Befehl ein:

```
java -jar StepFunctionsLocal.jar --lambda-endpoint http://localhost:3001
```

Wenn Step Functions Local gestartet wird, überprüft es die Umgebung und dann die in Ihrer `~/.aws/credentials` Datei konfigurierten Anmeldeinformationen. Standardmäßig verwendet es zunächst eine fiktive Benutzer-ID und wird als aufgeführt. `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

Wenn Sie die Docker-Version von Step Functions Local verwenden, starten Sie Step Functions mit dem folgenden Befehl:

```
docker run -p 8083:8083 amazon/aws-stepfunctions-local
```

Hinweise zur Installation der Docker-Version von Step Functions finden Sie unterStep Functions lokal (herunterladbare Version) und Docker einrichten.

> **Note**
>
> Sie können den Endpunkt über die Befehlszeile oder durch Festlegen von Umgebungsvariablen angeben, wenn Sie Step Functions von der `.jar` Datei aus starten. Für die Docker-Version müssen Sie die Endpunkte und Anmeldeinformationen in einer Textdatei angeben. Siehe Konfigurationsoptionen für Step Functions lokal einrichten.

Schritt 5: Erstellen eines Zustandsautomaten, der auf die AWS SAM CLI Local-Funktion verweist

Sobald Step Functions Local ausgeführt wird, erstellen Sie eine State-Maschine, die auf `HelloWorldFunction` die verweist, in Schritt 1: Einrichten von AWS SAM der Sie initialisiert haben.

```
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"
```

Dadurch wird eine State-Maschine erstellt und ein Amazon Resource Name (ARN) bereitgestellt, mit dem Sie eine Ausführung starten können.

```
{
    "creationDate": 1548805711.403,
    "stateMachineArn": "arn:aws:states:us-east-1:123456789012:stateMachine:HelloWorld"
}
```

Schritt 6: Die Ausführung des lokalen Zustandsautomaten starten

Sobald Sie eine State Machine erstellt haben, starten Sie eine Ausführung. Sie müssen den Endpunkt und den ARN der Statusmaschine referenzieren, wenn Sie den folgenden **`aws stepfunctions`** Befehl verwenden:

```
aws stepfunctions --endpoint http://localhost:8083 start-execution --state-machine
 arn:aws:states:us-east-1:123456789012:stateMachine:HelloWorld --name test
```

Dadurch wird eine Ausführung mit dem Namen `test` Ihrer `HelloWorld` State-Maschine gestartet.

```
{
    "startDate": 1548810641.52,
    "executionArn": "arn:aws:states:us-east-1:123456789012:execution:HelloWorld:test"
}
```

Jetzt, da Step Functions lokal läuft, können Sie mit dem Programm interagierenAWS CLI. Verwenden Sie beispielsweise den folgenden Befehl, um Informationen zu dieser Ausführung zu erhalten:

```
aws stepfunctions --endpoint http://localhost:8083 describe-execution --execution-arn
 arn:aws:states:us-east-1:123456789012:execution:HelloWorld:test
```

Der Aufruf `describe-execution` zur Ausführung liefert vollständigere Details, ähnlich wie bei der folgenden Ausgabe:

```
{
    "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": "{}"
}
```

Verwenden von Mocked-Service-Integrationen

In Step Functions Local können Sie die Ausführungspfade Ihrer Zustandsautomaten testen, ohne tatsächlich integrierte Services aufzurufen, indem Sie simulierte Serviceintegrationen verwenden. Um Ihre Zustandsautomaten für die Verwendung von simulierten Serviceintegrationen zu konfigurieren, erstellen Sie eine Pseudokonfigurationsdatei. In dieser Datei definieren Sie die gewünschte Ausgabe Ihrer Serviceintegrationen als simulierte Antworten und die Ausführungen, die Ihre simulierten Antworten verwenden, um einen Ausführungspfad als Testfälle zu simulieren.

Durch die Bereitstellung der Mock-Konfigurationsdatei für Step Functions Local können Sie Serviceintegrationsaufrufe testen, indem Sie Zustandsmaschinen ausführen, die die in den Testfällen angegebenen Mocked-Antworten verwenden, anstatt tatsächliche Serviceintegrationsaufrufe zu tätigen.

> **Note**
>
> Wenn Sie keine simulierten Serviceintegrationsantworten in der Mock-Konfigurationsdatei angeben, ruft Step Functions Local die AWS Serviceintegration mit dem Endpunkt auf, den Sie bei der Einrichtung von Step Functions Local konfiguriert haben. Informationen

zum Konfigurieren von Endpunkten für Step Functions Local finden Sie unter Konfigurationsoptionen für Step Functions lokal einrichten.

Themen

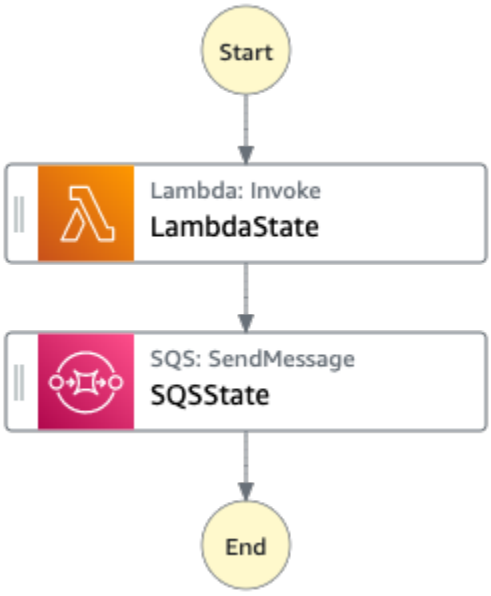
Wichtige Konzepte in diesem Thema

In diesem Thema werden mehrere Konzepte verwendet, die in der folgenden Liste definiert sind:

- Mocked Service Integrations – Bezieht sich auf Aufgabenstatus, die so konfiguriert sind, dass sie simulierte Antworten verwenden, anstatt tatsächliche Serviceaufrufe durchzuführen.
- Pseudoantworten – Bezieht sich auf Pseudodaten, für deren Verwendung Aufgabenstatus konfiguriert werden können.
- Testfälle – Bezieht sich auf Zustandsautomatenausführungen, die für die Verwendung von simulierten Serviceintegrationen konfiguriert sind.
- Mock-Konfigurationsdatei – Bezieht sich auf die Mock-Konfigurationsdatei, die JSON enthält, das simulierte Serviceintegrationen, simulierte Antworten und Testfälle definiert.

Schritt 1: Angeben von Mocked-Service-Integrationen in einer Mock-Konfigurationsdatei

Sie können Step Functions AWS SDK und optimierte Serviceintegrationen mit Step Functions Local testen. Die folgende Abbildung zeigt den Zustandsautomaten, der auf der Registerkarte Definition des Zustandsautomaten definiert ist:



Dazu müssen Sie eine Mock-Konfigurationsdatei erstellen, die Abschnitte enthält, wie in definiertEinführung in die Struktur der Mock-Konfiguration.

1. Erstellen Sie eine Datei mit dem Namen `MockConfigFile.json`, um Tests mit simulierten Serviceintegrationen zu konfigurieren.

   Das folgende Beispiel zeigt eine Mock-Konfigurationsdatei, die auf einen Zustandsautomaten mit zwei definierten Zuständen namens `LambdaState` und verweistSQSState.

   Mock configuration file example

   Im Folgenden finden Sie ein Beispiel für eine Mock-Konfigurationsdatei, die zeigt, wie Antworten aus dem Aufrufen einer Lambda-Funktion und dem Senden einer Nachricht an Amazon SQS nachgeahmt werden. In diesem Beispiel enthält der `LambdaSQSIntegration` Zustandsautomat drei Testfälle mit dem Namen `HappyPath`, und `RetryPath`, `HybridPath` die die `Task` Zustände mit dem Namen `LambdaState` und nachahmenSQSState. Diese Zustände verwenden die `MockedLambdaSuccess`, `MockedSQSSuccess`und `MockedLambdaRetry` simulierten Service-Antworten. Diese simulierten Service-Antworten sind im -`MockedResponses`Abschnitt der -Datei definiert.

   ```
   {
     "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

Im Folgenden finden Sie ein Beispiel für eine Zustandsautomatendefinition namens `LambdaSQSIntegration`, die zwei Status von Serviceintegrationsaufgaben mit dem Namen `LambdaState` und definiert`SQSState`. `LambdaState` enthält eine Wiederholungsrichtlinie, die auf basiert`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
    }
  }
}
```

Sie können die Definition des `LambdaSQSIntegration` Zustandsautomaten, auf die in der Mock-Konfigurationsdatei verwiesen wird, mit einem der folgenden Testfälle ausführen:

- `HappyPath` – Dieser Test ahmt die Ausgabe von `LambdaState` und `SQSState` mit `MockedLambdaSuccess` `MockedSQSSuccess` bzw. nach.
  - Der `LambdaState` gibt den folgenden Wert zurück:

    ```
    "0":{
      "Return":{
        "StatusCode":200,
        "Payload":{
          "StatusCode":200,
          "body":"Hello from Lambda!"
        }
      }
    }
    ```

  - Der `SQSState` gibt den folgenden Wert zurück:

```
"0":{
  "Return":{
    "MD5OfMessageBody":"3bcb6e8e-7h85-4375-b0bc-1a59812c6e51",
    "MessageId":"3bcb6e8e-8b51-4375-b0bc-1a59812c6e51"
  }
}
```

- `RetryPath` – Dieser Test ahmt die Ausgabe von `LambdaState` und `SQSState` mit `MockedLambdaRetry` `MockedSQSSuccess` bzw. nach. Darüber hinaus `LambdaState` ist so konfiguriert, dass vier Wiederholungsversuche durchgeführt werden. Die simulierten Antworten für diese Versuche werden im `MockedLambdaRetry` Status definiert und indiziert.
  - Der erste Versuch endet mit einem Aufgabenfehler, der eine Ursache und eine Fehlermeldung enthält, wie im folgenden Beispiel gezeigt:

```
"0":{
  "Throw": {
    "Error": "Lambda.ResourceNotReadyException",
    "Cause": "Lambda resource is not ready."
  }
}
```

  - Die ersten und zweiten Wiederholungsversuche enden mit einem Aufgabenfehler, der eine Ursache und eine Fehlermeldung enthält, wie im folgenden Beispiel gezeigt:

```
"1-2":{
  "Throw": {
    "Error": "Lambda.TimeoutException",
    "Cause": "Lambda timed out."
  }
}
```

  - Der dritte Wiederholungsversuch endet mit einem Aufgabenerfolg, der den Abschnitt Statusergebnis aus Nutzlast in der simulierten Lambda-Antwort enthält.

```
"3":{
  "Return": {
    "StatusCode": 200,
    "Payload": {
      "StatusCode": 200,
      "body": "Hello from Lambda!"
```

```
    }
  }
}
```

> **Note**
>
> - Bei Zuständen mit einer Wiederholungsrichtlinie erschöpft Step Functions Local die in der Richtlinie festgelegten Wiederholungsversuche, bis es eine Erfolgsantwort erhält. Das bedeutet, dass Sie Mocks für Wiederholungsversuche mit aufeinanderfolgenden Versuchszahlen kennzeichnen müssen und alle Wiederholungsversuche abdecken sollten, bevor Sie eine Erfolgsantwort zurückgeben.
> - Wenn Sie keine simulierte Antwort für einen bestimmten Wiederholungsversuch angeben, z. B. „3“ wiederholen, schlägt die Ausführung des Zustandsautomaten fehl.

- `HybridPath` – Dieser Test ahmt die Ausgabe von nach`LambdaState`. Nachdem erfolgreich `LambdaState` ausgeführt wurde und simulierte Daten als Antwort empfängt, `SQSState` führt einen tatsächlichen Serviceaufruf der in der Produktion angegebenen Ressource durch.

Informationen zum Starten von Testausführungen mit simulierten Serviceintegrationen finden Sie unter Schritt 3: Ausführen von Mocked-Service-Integrationstests.

2. Stellen Sie sicher, dass die Struktur der simulierten Antworten der Struktur der tatsächlichen Serviceantworten entspricht, die Sie erhalten, wenn Sie integrierte Serviceaufrufe tätigen. Informationen zu den Strukturanforderungen für simulierte Antworten finden Sie unter Konfiguration von simulierten Serviceintegrationen.

   In der vorherigen Beispiel-Mock-Konfigurationsdatei `MockedLambdaRetry` entsprechen die in definierten `MockedLambdaSuccess` und simulierten Antworten der Struktur der tatsächlichen Antworten, die vom Aufruf von zurückgegeben werden`HelloFromLambda`.

   > **Important**
   >
   > AWS -Serviceantworten können in der Struktur zwischen verschiedenen Services variieren. Step Functions Local überprüft nicht, ob simulierte Antwortstrukturen den tatsächlichen Service-Antwortstrukturen entsprechen. Sie müssen sicherstellen, dass Ihre simulierten Antworten den tatsächlichen Antworten entsprechen, bevor Sie sie

testen. Um die Struktur der Service-Antworten zu überprüfen, können Sie entweder die tatsächlichen Service-Aufrufe mit Step Functions durchführen oder die Dokumentation für diese Services anzeigen.

Schritt 2: Bereitstellen der Mock-Konfigurationsdatei für Step Functions Local

Sie können die Mock-Konfigurationsdatei auf eine der folgenden Arten für Step Functions Local bereitstellen:

Docker

> **Note**
>
> Wenn Sie die Docker-Version von Step Functions Local verwenden, können Sie die Mock-Konfigurationsdatei nur mit einer Umgebungsvariablen bereitstellen. Darüber hinaus müssen Sie die Mock-Konfigurationsdatei beim ersten Serverstart auf dem lokalen Container von Step Functions mounten.

Mounten Sie die Mock-Konfigurationsdatei in einem beliebigen Verzeichnis im lokalen Container von Step Functions. Legen Sie dann eine Umgebungsvariable mit dem Namen fest`SFN_MOCK_CONFIG`, die den Pfad zur Mock-Konfigurationsdatei im Container enthält. Mit dieser Methode kann die Mock-Konfigurationsdatei benannt werden, solange die Umgebungsvariable den Dateipfad und den Namen enthält.

Der folgende Befehl zeigt das Format zum Starten des Docker-Images.

```
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
```

Im folgenden Beispiel wird der Befehl verwendet, um das Docker-Image zu starten.

```
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

Verwenden Sie eine der folgenden Möglichkeiten, die Mock-Konfigurationsdatei für Step Functions Local bereitzustellen:

- Platzieren Sie die Mock-Konfigurationsdatei im selben Verzeichnis wie `Step FunctionsLocal.jar`. Wenn Sie diese Methode verwenden, müssen Sie die Mock-Konfigurationsdatei benennen`MockConfigFile.json`.
- Legen Sie in der Sitzung mit Step Functions Local eine Umgebungsvariable mit dem Namen `SFN_MOCK_CONFIG`auf den vollständigen Pfad der Mock-Konfigurationsdatei fest. Mit dieser Methode kann die Mock-Konfigurationsdatei benannt werden, solange die Umgebungsvariable ihren Dateipfad und Namen enthält. Im folgenden Beispiel ist die `SFN_MOCK_CONFIG` Variable so eingestellt, dass sie auf eine Mock-Konfigurationsdatei namens verweist`EnvSpecifiedMockConfig.json`, die sich im `/home/workspace` Verzeichnis befindet.

  ```
  export SFN_MOCK_CONFIG="/home/workspace/EnvSpecifiedMockConfig.json"
  ```

> **Note**
>
> - Wenn Sie die Umgebungsvariable nicht `SFN_MOCK_CONFIG` für Step Functions Local bereitstellen, versucht sie standardmäßig, eine Mock-Konfigurationsdatei mit dem Namen `MockConfigFile.json` in dem Verzeichnis zu lesen, aus dem Sie Step Functions Local gestartet haben.
> - Wenn Sie die Mock-Konfigurationsdatei im selben Verzeichnis wie platzieren `Step FunctionsLocal.jar` und die Umgebungsvariable festlegen`SFN_MOCK_CONFIG`, liest Step Functions Local die durch die Umgebungsvariable angegebene Datei.

Schritt 3: Ausführen von Mocked-Service-Integrationstests

Nachdem Sie eine Mock-Konfigurationsdatei für Step Functions Local erstellt und bereitgestellt haben, führen Sie den in der Mock-Konfigurationsdatei konfigurierten Zustandsautomaten mithilfe von

Mocked-Service-Integrationen aus. Überprüfen Sie dann die Ausführungsergebnisse mit einer API-Aktion.

1. Erstellen Sie einen Zustandsautomaten basierend auf der zuvor erwähnten Definition in der Mock-Konfigurationsdatei .

```
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. Führen Sie den Zustandsautomaten mit simulierten Serviceintegrationen aus.

   Um die Mock-Konfigurationsdatei zu verwenden, führen Sie einen `StartExecution` API-Aufruf auf einem Zustandsautomaten durch, der in der Mock-Konfigurationsdatei konfiguriert ist. Hängen Sie dazu das Suffix #*`test_name`*an den von verwendeten Zustandsautomaten-ARN an`StartExecution`. *`test_name`* ist ein Testfall, der für den Zustandsautomaten in derselben Mock-Konfigurationsdatei konfiguriert ist.

   Der folgende Befehl ist ein Beispiel, das den `LambdaSQSIntegration` Zustandsautomaten und die Mock-Konfiguration verwendet. In diesem Beispiel wird der `LambdaSQSIntegration` Zustandsautomat mit dem in definierten `HappyPath` Test ausgeführtSchritt 1: Angeben von Mocked-Service-Integrationen in einer Mock-Konfigurationsdatei. Der `HappyPath` Test enthält die Konfiguration für die Ausführung, um simulierte Serviceintegrationsaufrufe zu verarbeiten, die `LambdaState` und -`SQSState`Zustände mit den `MockedSQSSuccess` simulierten Serviceantworten `MockedLambdaSuccess` und durchführen.

```
aws stepfunctions start-execution \
    --endpoint http://localhost:8083 \
    --name executionWithHappyPathMockedServices \
```

```
    --state-machine arn:aws:states:us-
east-1:123456789012:stateMachine:LambdaSQSIntegration#HappyPath
```

3. Zeigen Sie die Ausführungsantwort des Zustandsautomaten an.

   Die Antwort auf den Aufruf `StartExecution` mit einem simulierten Serviceintegrationstest entspricht der Antwort auf `StartExecution` den normalen Aufruf von , der den Ausführungs-ARN und das Startdatum zurückgibt.

   Im Folgenden finden Sie eine Beispielantwort auf den Aufruf `StartExecution` von mit dem simulierten Serviceintegrationstest:

```
{
   "startDate":"2022-01-28T15:03:16.981000-05:00",
   "executionArn":"arn:aws:states:us-
east-1:123456789012:execution:LambdaSQSIntegration:executionWithHappyPathMockedServices"
}
```

4. Überprüfen Sie die Ergebnisse der Ausführung`ListExecutions`, indem Sie einen `DescribeExecution`-, - oder `GetExecutionHistory`-API-Aufruf durchführen.

```
aws stepfunctions get-execution-history \
    --endpoint http://localhost:8083 \
    --execution-arn arn:aws:states:us-
east-1:123456789012:execution:LambdaSQSIntegration:executionWithHappyPathMockedServices
```

   Das folgende Beispiel zeigt Teile einer Antwort auf den Aufruf `GetExecutionHistory` mit dem Ausführungs-ARN aus der Beispielantwort in Schritt 2. In diesem Beispiel `SQSState` ist die Ausgabe von `LambdaState` und die Mock-Daten, die in `MockedLambdaSuccess` und `MockedSQSSuccess` in der Mock-Konfigurationsdatei definiert sind. Darüber hinaus werden die simulierten Daten auf die gleiche Weise verwendet, wie Daten verwendet werden, die durch die Durchführung tatsächlicher Serviceintegrationsaufrufe zurückgegeben werden. Außerdem `LambdaState` wird in diesem Beispiel die Ausgabe von `SQSState` als Eingabe an übergeben.

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

Konfigurationsdatei für Mocked Service-Integrationen

Um Mock-Service-Integrationen verwenden zu können, müssen Sie zunächst eine Scheinkonfigurationsdatei `MockConfigFile.json` mit dem Namen erstellen, die Ihre Scheinkonfigurationen enthält. Stellen Sie dann Step Functions Local die Mock-Konfigurationsdatei zur Verfügung. Diese Konfigurationsdatei definiert Testfälle, die Scheinzustände enthalten, die simulierte Antworten auf die Serviceintegration verwenden. Der folgende Abschnitt enthält Informationen zur Struktur der Scheinkonfiguration, einschließlich der Mock-Status und der simulierten Antworten:

Themen

- Einführung in die Struktur der Mock-Konfiguration
- Konfiguration von simulierten Serviceintegrationen

Einführung in die Struktur der Mock-Konfiguration

Eine Scheinkonfiguration ist ein JSON-Objekt, das die folgenden Felder auf oberster Ebene enthält:

- `StateMachines`- Die Felder dieses Objekts stellen State-Maschinen dar, die für die Verwendung nachgemachter Dienstintegrationen konfiguriert sind.
- `MockedResponse`- Die Felder dieses Objekts stellen vorgetäuschte Antworten auf Service-Integrationsaufrufe dar.

Das Folgende ist ein Beispiel für eine Scheinkonfigurationsdatei, die eine `StateMachine` Definition und enthält`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!"
          }
        }
      }
    }
  }
}
```

Referenz zum Feld „Scheinkonfiguration“

In den folgenden Abschnitten werden die Objektfelder der obersten Ebene erläutert, die Sie in Ihrer Scheinkonfiguration definieren müssen.

- StateMachines
- MockedResponses

StateMachines

Das `StateMachines` Objekt definiert, welche State-Maschinen simulierte Serviceintegrationen verwenden werden. Die Konfiguration für jede Zustandsmaschine wird als übergeordnetes Feld von `StateMachines` dargestellt. Der Feldname ist der Name der Zustandsmaschine und Wert ist ein

Objekt, das ein einzelnes Feld mit dem Namen enthält`TestCases`, dessen Felder Testfälle dieser Zustandsmaschine darstellen.

Die folgende Syntax zeigt eine State Machine mit zwei Testfällen:

```
"MyStateMachine": {
  "TestCases": {
    "HappyPath": {
      ...
    },
    "SadPath": {
      ...
    }
  }
```

TestCases

Die Felder von `TestCases` stellen einzelne Testfälle für die Staatsmaschine dar. Der Name jedes Testfalls muss pro Zustandsmaschine eindeutig sein, und der Wert jedes Testfalls ist ein Objekt, das eine simulierte Antwort angibt, die für Aufgabenzustände in der Zustandsmaschine verwendet werden soll.

Das folgende Beispiel für a `TestCase` verknüpft zwei `Task` Staaten mit zwei`MockedResponses`:

```
"HappyPath": {
  "SomeTaskState": "SomeMockedResponse",
  "AnotherTaskState": "AnotherMockedResponse"
}
```

MockedResponses

`MockedResponses`ist ein Objekt, das mehrere simulierte Antwortobjekte mit eindeutigen Feldnamen enthält. Ein simuliertes Antwortobjekt definiert die erfolgreiche Ergebnis- oder Fehlerausgabe für jeden Aufruf eines simulierten Task-Status. Sie geben die Aufrufnummer mithilfe einzelner ganzzahliger Zeichenketten wie „0“, „1“, „2“ und „3“ oder eines inklusiven Bereichs von Ganzzahlen wie „0-1“, „2-3“ an.

Wenn Sie eine Aufgabe simulieren, müssen Sie für jeden Aufruf eine simulierte Antwort angeben. Eine Antwort muss ein einzelnes Feld enthalten, das den Namen hat `Return` oder `Throw` dessen Wert die Ergebnis- oder Fehlerausgabe des simulierten Task-Aufrufs ist. Wenn Sie keine simulierte Antwort angeben, schlägt die Ausführung der State Machine fehl.

Das Folgende ist ein Beispiel für `Return` Objekte `MockedResponse` mit `Throw` und. In diesem Beispiel wird beim ersten dreimaligen Ausführen der Zustandsmaschine die in angegebene Antwort zurückgegeben, und beim vierten Mal, wenn die Zustandsmaschine ausgeführt `"0-2"` wird, `"3"` wird die in angegebene Antwort zurückgegeben.

```
"SomeMockedResponse": {
  "0-2": {
    "Throw": {
      ...
    }
  },
  "3": {
    "Return": {
      ...
    }
  }
}
```

> **Note**
>
> Wenn Sie einen `Map` Bundesstaat verwenden und vorhersehbare Reaktionen für diesen `Map` Bundesstaat sicherstellen möchten, setzen Sie den Wert von `maxConcurrency` auf 1. Wenn Sie einen Wert größer als 1 festlegen, führt Step Functions Local mehrere Iterationen gleichzeitig aus, wodurch die allgemeine Ausführungsreihenfolge der Zustände zwischen den Iterationen unvorhersehbar ist. Dies kann außerdem dazu führen, dass Step Functions Local von einer Ausführung zur nächsten unterschiedliche simulierte Antworten für Iterationszustände verwendet.

Ergebnis

`Return`wird als Feld der `MockedResponse` Objekte dargestellt. Es gibt das erfolgreiche Ergebnis eines simulierten Task-Status an.

Das Folgende ist ein Beispiel für ein `Return` Objekt, das eine simulierte Antwort für den Aufruf `Invoke`einer Lambda-Funktion enthält:

```
"Return": {
  "StatusCode": 200,
  "Payload": {
```

```
    "StatusCode": 200,
    "body": "Hello from Lambda!"
  }
}
```

Wirf

`Throw`wird als Feld der `MockedResponse` Objekte dargestellt. Es gibt die Fehlerausgabe einer fehlgeschlagenen Aufgabe an. Der Wert von `Throw` muss ein Objekt sein, das `Cause` Felder `Error` und Felder mit Zeichenfolgenwerten enthält. Darüber hinaus `MockConfigFile.json` muss der Zeichenfolgenwert, den Sie im `Error` Feld in der angeben, mit den Fehlern übereinstimmen, die in den `Catch` Abschnitten `Retry` und auf Ihrem State-Computer behandelt werden.

Das Folgende ist ein Beispiel für ein `Throw` Objekt, das eine simulierte Antwort für den Aufruf `Invoke`einer Lambda-Funktion enthält:

```
"Throw": {
  "Error": "Lambda.TimeoutException",
  "Cause": "Lambda timed out."
}
```

Konfiguration von simulierten Serviceintegrationen

Mit Step Functions Local können Sie jede Dienstintegration simulieren. Step Functions Local erzwingt jedoch nicht, dass die Mocks mit den echten APIs identisch sind. Eine nachgeahmte Aufgabe ruft niemals den Dienstendpunkt auf. Wenn Sie keine simulierte Antwort angeben, versucht ein Task, die Dienstendpunkte aufzurufen. Darüber hinaus generiert Step Functions Local automatisch ein Task-Token, wenn Sie eine Aufgabe mit dem simulieren `.waitForTaskToken`.

Bewährte Methoden für Step Functions

Die folgenden bewährten Methoden zum Implementieren von AWS Step Functions-Workflows können Sie bei der Optimierung der Leistung Ihrer Implementierungen unterstützen.

Themen

Verwenden Sie Timeouts, um festgefahrene Ausführungen zu vermeiden

Standardmäßig gibt die Sprache von Amazon States keine Timeouts für State-Machine-Definitionen an. Ohne ein explizites Timeout verlassen sich Step Functions oft ausschließlich auf die Antwort eines Aktivitätsmitarbeiters, um zu wissen, dass eine Aufgabe abgeschlossen ist. Wenn etwas schief geht und das `TimeoutSeconds` Feld nicht für den `Task` Status `Activity` oder angegeben ist, bleibt eine Ausführung hängen und wartet auf eine Antwort, die niemals kommen wird.

Um diese Situation zu vermeiden, geben Sie ein angemessenes Timeout an, wenn Sie eine `Task` Zustandsmaschine erstellen. Beispiele:

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

Wenn Sie einen Callback mit einem Task-Token verwenden (. waitForTaskToken), empfehlen wir, Heartbeats zu verwenden und das `HeartbeatSeconds` Feld zu Ihrer `Task` Statusdefinition

hinzuzufügen. Sie können einen Wert festlegen`HeartbeatSeconds`, der unter dem Zeitlimit für die Aufgabe liegt. Wenn Ihr Workflow also mit einem Heartbeat-Fehler fehlschlägt, wissen Sie, dass es am Aufgabenfehler liegt und nicht daran, dass die Ausführung der Aufgabe viel Zeit in Anspruch nimmt.

```
{
  "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
    }
  }
}
```

Weitere Informationen finden Sie Status der Aufgabe in der Sprachdokumentation von Amazon States.

> **Note**
>
> Sie können mithilfe des `TimeoutSeconds` Felds in Ihrer Sprachdefinition für die Amazon-Staaten ein Timeout für Ihren State Machine festlegen. Weitere Informationen finden Sie unter Struktur der Zustandsmaschine.

Verwenden Sie Amazon S3 S3-ARNs, anstatt große Nutzlasten weiterzuleiten

Ausführungen, die große Datennutzlasten zwischen Zuständen übergeben, können beendet werden. Wenn die Daten, die Sie zwischen Bundesstaaten übertragen, auf über 256 KB anwachsen könnten, verwenden Sie Amazon Simple Storage Service (Amazon S3), um die Daten zu speichern, und analysieren Sie den Amazon-Ressourcennamen (ARN) des Buckets im `Payload` Parameter, um den

Bucket-Namen und den Schlüsselwert abzurufen. Alternativ passen Sie Ihre Implementierung an, sodass Sie in Ihren Ausführungen kleinere Nutzlasten übergeben.

Im folgenden Beispiel übergibt eine Zustandsmaschine Eingaben an eine AWS Lambda Funktion, die eine JSON-Datei in einem Amazon S3 S3-Bucket verarbeitet. Nachdem Sie diese Zustandsmaschine ausgeführt haben, liest die Lambda-Funktion den Inhalt der JSON-Datei und gibt den Dateiinhalt als Ausgabe zurück.

So erstellen Sie die Lambda-Funktion:

Die folgende Lambda-Funktion mit dem Namen *`pass-large-payload`* liest den Inhalt einer JSON-Datei, die in einem bestimmten Amazon S3 S3-Bucket gespeichert ist.

> **Note**
>
> Nachdem Sie diese Lambda-Funktion erstellt haben, stellen Sie sicher, dass Sie ihrer IAM-Rolle die entsprechende Berechtigung zum Lesen aus einem Amazon S3 S3-Bucket erteilen. Fügen Sie beispielsweise die ReadOnlyAccessAmazonS3-Berechtigung der Rolle der Lambda-Funktion hinzu.

```
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
```

Erstellen Sie die Zustandsmaschine

Die folgende Zustandsmaschine ruft die Lambda-Funktion auf, die Sie zuvor erstellt haben.

```
{
   "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
      }
   }
}
```

Anstatt eine große Datenmenge in der Eingabe zu übergeben, könnten Sie diese Daten in einem Amazon S3 S3-Bucket speichern und den Amazon-Ressourcennamen (ARN) des Buckets im `Payload` Parameter übergeben, um den Bucket-Namen und den Schlüsselwert zu erhalten. Ihre Lambda-Funktion kann dann diesen ARN verwenden, um direkt auf die Daten zuzugreifen. Im Folgenden finden Sie eine Beispieleingabe für die State-Machine-Ausführung, bei der die Daten `data.json` in einem Amazon S3 S3-Bucket mit dem Namen gespeichert werden*large-payload-json*.

```
{
  "key": "data.json",
  "bucket": "arn:aws:s3:::large-payload-json"
}
```

Vermeiden Sie es, das historische Kontingent zu erreichen

AWS Step Functionshat ein festes Kontingent von 25.000 Einträgen im Verlauf der Ausführungsereignisse. Wenn eine Ausführung 24.999 Ereignisse erreicht, wartet sie auf das nächste Ereignis.

- Wenn die Ereignisnummer 25.000 lautet`ExecutionSucceeded`, wird die Ausführung erfolgreich abgeschlossen.
- Wenn die Ereignisnummer 25.000 nicht angegeben ist`ExecutionSucceeded`, wird das `ExecutionFailed` Ereignis protokolliert und die Ausführung der Zustandsmaschine schlägt fehl, da das Verlaufslimit erreicht ist

Um zu verhindern, dass dieses Kontingent für lang andauernde Ausführungen erreicht wird, können Sie eine der folgenden Problemumgehungen ausprobieren:

- Verwenden Sie den Status Map im Modus Verteilt. In diesem Modus führt der `Map` Status jede Iteration als untergeordnete Workflow-Ausführung aus, wodurch eine hohe Parallelität von bis zu 10.000 parallel untergeordneten Workflow-Ausführungen ermöglicht wird. Jede untergeordnete Workflow-Ausführung hat ihren eigenen Ausführungsverlauf, der von dem des übergeordneten Workflows getrennt ist.
- Starten Sie eine neue State-Machine-Ausführung direkt vom `Task` Status einer laufenden Ausführung aus. Um solche verschachtelten Workflow-Ausführungen zu starten, verwenden Sie die `StartExecution` API-Aktion von Step Functions in der übergeordneten Zustandsmaschine zusammen mit den erforderlichen Parametern. Weitere Informationen zur Verwendung verschachtelter Workflows finden Sie unter Starten von Workflow-Ausführungen über einen Aufgabenstatus oder Verwenden einer Step Functions Functions-API-Aktion, um ein neues Ausführungs-Tutorial fortzusetzen.

  > **Tip**
  >
  > Ein Beispiel für einen verschachtelten Workflow finden Sie AWS-Konto in Modul 13 — Verschachtelte Express-Workflows.

- Implementieren Sie ein Muster, das eine AWS Lambda Funktion verwendet, die eine neue Ausführung Ihrer Zustandsmaschine starten kann, um die laufende Arbeit auf mehrere Workflow-Ausführungen aufzuteilen. Weitere Informationen finden Sie im Tutorial Verwenden einer Lambda-Funktion, um eine neue Ausführung fortzusetzen.

Lambda-Serviceausnahmen behandeln

In AWS Lambda können gelegentlich vorübergehende Servicefehler auftreten. In diesem Fall führt der Aufruf von Lambda zu einem 500-Fehler, z. B.`ClientExecutionTimeoutException`, `ServiceException`AWSLambdaException, oder. `SdkClientException` Es hat sich bewährt, diese Ausnahmen in Ihrer Zustandsmaschine proaktiv zu behandeln, um Ihre Lambda-Funktion `Retry` aufzurufen oder um den Fehler zu `Catch` beheben.

Lambda-Fehler werden als `Lambda.`*`ErrorName`* gemeldet. Um einen Lambda-Serviceausnahmefehler erneut zu versuchen, können Sie den folgenden `Retry` Code verwenden.

```
"Retry": [ {
   "ErrorEquals": [ "Lambda.ClientExecutionTimeoutException",
 "Lambda.ServiceException", "Lambda.AWSLambdaException", "Lambda.SdkClientException"],
   "IntervalSeconds": 2,
   "MaxAttempts": 6,
   "BackoffRate": 2
} ]
```

> **Note**
>
> Unbehandelte Fehler in Lambda werden wie `Lambda.Unknown` in der Fehlerausgabe gemeldet. Dazu gehören out-of-memory Fehler und Funktions-Timeouts. Sie können nach, oder abgleichen `Lambda.UnknownStates.ALL`, `States.TaskFailed` um diese Fehler zu behandeln. Wenn Lambda die maximale Anzahl von Aufrufen erreicht, lautet der Fehler. `Lambda.TooManyRequestsException` Weitere Informationen zu Lambda-Funktionsfehlern finden Sie unter Fehlerbehandlung und automatische Wiederholungen im AWS LambdaEntwicklerhandbuch.

Weitere Informationen finden Sie hier:

- Wiederholter Versuch nach einem Fehler
- Behandlung von Fehlerbedingungen mithilfe einer Step Functions Functions-Zustandsmaschine
- Fehler beim Aufrufen von Lambda

Vermeiden Sie Latenz bei der Abfrage von Aktivitätsaufgaben

Die `GetActivityTask`-API ist so konzipiert, dass sie ein `taskToken` einmalig unterstützt. Wenn ein `taskToken` während der Kommunikation mit einem Aktivitäts-Worker verloren geht, können mehrere `GetActivityTask`-Anfragen für 60 Sekunden blockiert werden, die auf eine Antwort warten, bis ein Timeout für `GetActivityTask` stattfindet.

Wenn Sie nur eine kleine Anzahl von Abrufen haben, die auf eine Antwort warten, ist es möglich, dass alle Anforderungen hinter der blockierten Anforderung auflaufen und anhalten. Wenn Sie jedoch eine große Anzahl ausstehender Umfragen für jede Aktivität haben (Amazon Resource Name, ARN), und ein Teil Ihrer Anfragen in der Warteschleife stecken bleibt, können noch viele weitere Anfragen erhalten `taskToken` und mit der Bearbeitung beginnen.

Für Produktionssysteme empfehlen wir mindestens 100 offene Abrufe pro Aktivitäts-ARN zu jedem Zeitpunkt. Wenn ein Abruf blockiert wird, und einen Teil dieser Abrufe dahinter auflaufen, gibt es nach wie vor viele weitere Anfragen, die ein `taskToken` erhalten, um weiterzuarbeiten, während die `GetActivityTask`-Anfrage blockiert ist.

So vermeiden Sie diese Art von Latenzproblemen beim Abrufen von Aufgaben

- Implementieren Sie Ihre Poller als separate Threads aus der Arbeit in Ihrer Aktivitäts-Worker-Implementierung.
- Verwenden Sie mindestens 100 offene Abrufe pro Aktivitäts-ARN zu jedem Zeitpunkt.

> **Note**
>
> Eine Skalierung auf 100 offene Abrufe pro ARN kann kostspielig sein. Beispielsweise ist das Abfragen von 100 Lambda-Funktionen pro ARN 100-mal teurer als eine einzelne Lambda-Funktion mit 100 Abfrage-Threads. Um sowohl die Latenz zu reduzieren als auch die Kosten zu minimieren, verwenden Sie eine Sprache mit asynchronem E/A und implementieren pro Worker mehrere Abfrage-Threads. Ein Beispiel für einen Aktivitäts-Worker, in dem die Poller-Threads unabhängig von der Arbeitsthreads sind, finden Sie unter Beispiel für einen Aktivitäts-Worker in Ruby.

Weitere Informationen zu Aktivitäten und Aktivitäts-Workern finden Sie unter Aktivitäten.

Auswählen von Standard- oder Express-Workflows

AWS Step Functions bietet Standard-Workflows als Standard-Workflowtyp mit der Option, Express-Workflows auszuwählen.

Sie können Standard-Workflows wählen, wenn Sie langlebige, langlebige und überprüfbare Workflows benötigen, oder Express-Workflows für umfangreiche Workloads zur Ereignisverarbeitung. Die Ausführungen des Zustandsautomaten verhalten sich abhängig von dem von Ihnen ausgewählten `Type` jeweils anders. Sobald der Zustandsautomat erstellt wurde, kann der von Ihnen ausgewählte `Type` nicht mehr geändert werden.

- Ausführliche Informationen zu den Unterschieden zwischen Standard- und Express-Workflows finden Sie unter. Standard- und Express-Workflows
- Informationen zur Kostenoptimierung bei der Erstellung serverloser Workflows mithilfe von Step Functions finden Sie unterKostenoptimierung mit Express Workflows.

Größenbeschränkungen der Amazon CloudWatch Logs-Ressourcenrichtlinie

CloudWatch Die Ressourcenrichtlinien für Logs sind auf 5120 Zeichen begrenzt. Wenn CloudWatch Logs feststellt, dass sich eine Richtlinie dieser Größenbeschränkung nähert, werden automatisch Protokollgruppen aktiviert, die mit `/aws/vendedlogs/` beginnen.

Wenn Sie eine Zustandsmaschine mit aktivierter Protokollierung erstellen, muss Step Functions Ihre CloudWatch Logs-Ressourcenrichtlinie mit der von Ihnen angegebenen Protokollgruppe aktualisieren. Um zu verhindern, dass die Größenbeschränkung für die CloudWatch Logs-Ressourcenrichtlinie erreicht wird, stellen Sie Ihren CloudWatch Logs-Protokollgruppennamen das Präfix voran`/aws/vendedlogs/`. Wenn Sie in der Step Functions Functions-Konsole eine Protokollgruppe erstellen, wird den Namen der Protokollgruppen ein Präfix `/aws/vendedlogs/states` vorangestellt. Weitere Informationen finden Sie unter Aktivieren der Protokollierung für bestimmte AWS Dienste.

Wenn Sie nicht auf die CloudWatch Protokolle zugreifen können, finden Sie Unable to access the CloudWatch Logs weitere Informationen unter.

Verwendung AWS Step Functions mit anderen Diensten

Erfahren Sie, wie Sie andere APIs koordinieren AWS-Services oder APIs von Drittanbietern mit aufrufen. AWS Step Functions

Themen

Rufen Sie andere Dienste AWS an

Mit AWS Serviceintegrationen können Sie API-Aktionen aufrufen und Ausführungen direkt von Ihrem Workflow aus koordinieren. Sie können die AWS SDK-Integrationen von Step Functions verwenden, um jeden der über zweihundert AWS Dienste direkt von Ihrer Zustandsmaschine aus aufzurufen, sodass Sie Zugriff auf über neuntausend API-Aktionen haben. Oder Sie können die optimierten Integrationen von Step Functions verwenden, von denen jede angepasst wurde, um spezielle Funktionen für Ihren Arbeitsablauf bereitzustellen. Einige API-Aktionen sind in beiden Integrationstypen verfügbar. Wenn möglich, empfehlen wir die Verwendung der optimierten Integration.

Sie koordinieren diese Dienste direkt von einem `Task` Bundesstaat aus in der Sprache der Amazonas-Staaten. Mit Step Functions können Sie beispielsweise andere Dienste aufrufen, um:

- Rufen Sie eine AWS Lambda Funktion auf.
- Führen Sie einen AWS Batch Job aus und führen Sie dann basierend auf den Ergebnissen verschiedene Aktionen aus.
- Fügen Sie einen Artikel von Amazon DynamoDB ein oder rufen Sie ihn ab.
- Führen Sie eine Amazon Elastic Container Service (Amazon ECS) -Aufgabe aus und warten Sie, bis sie abgeschlossen ist.

- Veröffentlichen Sie zu einem Thema in Amazon Simple Notification Service (Amazon SNS).
- Senden Sie eine Nachricht in Amazon Simple Queue Service (Amazon SQS).
- Verwalte einen Job für Amazon AWS Glue oder Amazon SageMaker.
- Erstellen Sie Workflows für die Ausführung von Amazon EMR-Jobs.
- Starten Sie eine AWS Step Functions Workflow-Ausführung.

Optimierte Integrationen

Optimierte Integrationen wurden von Step Functions angepasst, um spezielle Funktionen für einen Workflow-Kontext bereitzustellen. Lambda `Invoke` konvertiert beispielsweise seine API-Ausgabe von einem maskierten JSON in ein JSON-Objekt. AWS Batch`SubmitJob`ermöglicht es Ihnen, die Ausführung anzuhalten, bis der Job abgeschlossen ist. Die ersten optimierten Integrationen wurden 2018 veröffentlicht, und inzwischen gibt es über fünfzig APIs.

AWS SDK-Integrationen

AWS SDK-Integrationen funktionieren genau wie ein standardmäßiger API-Aufruf, der das AWS SDK verwendet. Sie bieten die Möglichkeit, über neuntausend APIs für die mehr als zweihundert AWS Dienste direkt von Ihrer State Machine-Definition aus aufzurufen.

Unterstützung von Integrationsmustern

Standard-Workflows und Express-Workflows unterstützen dieselben Integrationen, aber nicht dieselben Integrationsmuster.

- Die Unterstützung von optimierten Integrationsmustern ist für jede Integration unterschiedlich.
- Express-Workflows unterstützen Run a Job (.sync) oder Wait for Callback () nicht. waitForTaskToken).
- Weitere Informationen finden Sie unter Standard- und Express-Workflows.

Standard Workflows

Unterstützte Serviceintegrationen

	Service	Request Response (Antwort anfordern)	Run a Job (Auftrag ausführen) (`.sync`)	Wait for Callback (Auf Rückruf warten) (`.waitForTaskToken`)
Optimierte Integrationen	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	✓		✓

	Service	Request Response (Antwort anfordern)	Run a Job (Auftrag ausführen) (`.sync`)	Wait for Callback (Auf Rückruf warten) (`.waitForTaskToken`)
	AWS Elemental MediaConvert	✓	✓	
	Amazon SageMaker	✓	✓	
	Amazon SNS	✓		✓
	Amazon SQS	✓		✓
	AWS Step Functions	✓	✓	✓
AWS SDK-Integrationen	Über zweihundert	✓		✓

Express Workflows

Unterstützte Serviceintegrationen

	Service	Request Response (Antwort anfordern)	Run a Job (Auftrag ausführen) (`.sync`)	Wait for Callback (Auf Rückruf warten) (`.waitForTaskToken`)
Optimierte	Amazon API Gateway	✓		
	Amazon Athena	✓		

	Service	Request Response (Antwort anfordern)	Run a Job (Auftrag ausführen) (`.sync`)	Wait for Callback (Auf Rückruf warten) (`.waitForTaskToken`)
Integrationen	AWS Batch	✓		
	Amazon Bedrock	✓		
	AWS CodeBuild	✓		
	Amazon DynamoDB	✓		
	Amazon ECS/Fargate	✓		
	Amazon EKS	✓		
	Amazon EMR	✓		
	Amazon EMR on EKS	✓		
	Amazon EMR Serverless	✓		
	Amazon EventBridge	✓		
	AWS Glue	✓		
	AWS Glue DataBrew	✓		
	AWS Lambda	✓		
	AWS Elemental MediaConvert	✓		
	Amazon SageMaker	✓		
	Amazon SNS	✓		
	Amazon SQS	✓		

	Service	Request Response (Antwort anfordern)	Run a Job (Auftrag ausführen) (`.sync`)	Wait for Callback (Auf Rückruf warten) (`.waitForTaskToken`)
	AWS Step Functions	✓		
AWS SDK-Integrationen	Über zweihundert	✓		

Kontoübergreifender Zugriff

Step Functions bietet kontoübergreifenden Zugriff auf Ressourcen, die AWS-Konten in Ihren Workflows unterschiedlich konfiguriert sind. Mithilfe der Step Functions Functions-Dienstintegrationen können Sie jede kontoübergreifende AWS Ressource aufrufen, auch wenn diese AWS-Service keine ressourcenbasierten Richtlinien oder kontenübergreifenden Aufrufe unterstützt.

Weitere Informationen finden Sie unter Zugreifen auf Ressourcen AWS-Konten in anderen Workflows.

AWS SDK-Serviceintegrationen

AWS Step Functions lässt sich in integrieren AWS-Services, sodass Sie die API-Aktionen der einzelnen Dienste von Ihrem Workflow aus aufrufen können. Sie können die AWS SDK-Integrationen von Step Functions verwenden, um fast alle API-Aktionen von Ihrer AWS-Service Zustandsmaschine aus aufzurufen. Sie können auch die optimierten Integrationen von Step Functions verwenden, von denen jede so angepasst wurde, dass sie spezielle Funktionen für Ihren Workflow bieten.

Einige Dienste oder SDKs sind möglicherweise nicht als AWS SDK-Integrationen verfügbar, weder vorübergehend noch dauerhaft. Bei kürzlich veröffentlichten Diensten sind SDK-Interaktionen möglicherweise erst nach einem späteren Update verfügbar. Für einige Dienste ist eine benutzerdefinierte Konfiguration erforderlich, z. B. die Angabe eines kundenspezifischen Endpunkts,

der für eine optimierte Integration möglicherweise besser geeignet ist. Andere SDKs sind für die Verwendung in einem Workflow ungeeignet, z. B. solche für das Streamen von Audio oder Video. Schließlich können einige Dienste zurückgehalten werden, bis sie bestimmte interne Validierungen bestanden haben, die von Step Functions durchgeführt werden.

> **Tip**
>
> Ein Beispiel für einen Workflow, der AWS SDK-Integrationen für Sie verwendet AWS-Konto, finden Sie unter Modul 9 — AWS SDK-Dienstintegrationen in The Workshop. AWS Step Functions

Themen

Verwenden von AWS SDK-Dienstintegrationen

Um AWS SDK-Integrationen zu verwenden, geben Sie den Dienstnamen und den API-Aufruf sowie optional ein Dienstintegrationsmuster an.

> **Note**
>
> - Parameter in Step Functions werden in ausgedrückt PascalCase, auch wenn sich die native Service-API in CamelCase befindet. Sie könnten beispielsweise die Step Functions API-Aktion verwenden `startSyncExecution` und ihren Parameter als angeben. `StateMachineArn`
> - Für API-Aktionen, die Aufzählungsparameter akzeptieren, wie z. B. die `DescribeLaunchTemplateVersions` API-Aktion für Amazon EC2, geben Sie eine Pluralversion des Parameternamens an. Geben Sie beispielsweise `Filters` für den `Filter.N` Parameter der API-Aktion an. `DescribeLaunchTemplateVersions`

Sie können AWS SDK-Dienste direkt aus der Sprache der Amazon States im `Resource` Feld eines Aufgabenstatus aufrufen. Verwenden Sie dazu die folgende Syntax:

`arn:aws:states:::aws-sdk:`*`serviceName`*`:`*`apiAction`*`.`*`[serviceIntegrationPattern]`*

Für Amazon EC2 könnten Sie beispielsweise verwenden`arn:aws:states:::aws-sdk:ec2:describeInstances`. Dies würde die Ausgabe zurückgeben, wie sie für den Amazon EC2 EC2-API-Aufruf DescribeInstances definiert wurde.

Wenn bei einer AWS SDK-Integration ein Fehler auftritt, besteht das resultierende Fehlerfeld aus dem Servicenamen und dem Fehlernamen, getrennt durch einen Punkt:. *`ServiceName`*`.`*`ErrorName`* Sowohl der Dienstname als auch der Fehlername werden in Pascal-Groß- und Kleinschreibung eingegeben. Sie können den Dienstnamen auch in Kleinbuchstaben im Feld Ressource des Aufgabenstatus sehen. Die Namen potenzieller Fehler finden Sie in der API-Referenzdokumentation des Zieldienstes.

Sie könnten beispielsweise die `arn:aws:states:::aws-sdk:acmpca:deleteCertificateAuthority` AWS SDK-Integration verwenden. Die AWS Private Certificate Authority API-Referenz gibt an, dass die `DeleteCertificateAuthority` API-Aktion `ResourceNotFoundException` beispielsweise zu einem führen kann. Um diesen Fehler zu behandeln, würden Sie daher den Fehler `AcmPca.ResourceNotFoundException` in den Retriern oder Catchern Ihres Aufgabenstatus angeben. Weitere Informationen zur Fehlerbehandlung in Step Functions finden Sie unterFehlerbehandlung in Step Functions.

Step Functions kann IAM-Richtlinien für AWS SDK-Integrationen nicht automatisch generieren. Nachdem Sie Ihre Zustandsmaschine erstellt haben, müssen Sie zur IAM-Konsole navigieren und Ihre Rollenrichtlinien konfigurieren. Weitere Informationen finden Sie unter IAM-Richtlinien für integrierte Dienste.

Im Sammeln Sie Amazon S3 S3-Bucket-Informationen mithilfe von AWS SDK-Serviceintegrationen Tutorial finden Sie ein Beispiel für die Verwendung von AWS SDK-Integrationen.

Unterstützte AWS SDK-Dienstintegrationen

In der folgenden Tabelle sind die von Step Functions unterstützten AWS SDK-Dienstintegrationen aufgeführt. In der Spalte mit den *`Task`*Statusressourcen ist die Syntax für den Aufruf einer bestimmten API-Aktion aufgeführt, wenn die Integration für den in der Spalte Dienstname auf der linken Seite angegebenen Dienst verwendet wird. Die Spalte Unterstütztes Datum enthält Informationen zu den Terminen, an denen die Serviceintegration unterstützt wurde. Darüber

hinaus werden in der Spalte Ausnahmepräfix auf der rechten Seite die Ausnahmepräfixe für jede Serviceintegration aufgeführt. Diese Ausnahmepräfixe sind in den Ausnahmen enthalten, die generiert werden, wenn Sie fälschlicherweise eine AWS SDK-Dienstintegration mit Step Functions durchführen.

> **Note**
>
> - Dienste, die mit*** gekennzeichnet sind, verfügen über API-Aktionen, die derzeit nicht von Step Functions unterstützt werden. Informationen zu den Aktionen, die für einen Dienst nicht unterstützt werden, finden Sie in der Nicht unterstützte API-Aktionen für unterstützte Dienste Tabelle.
> - Informationen zu den Aktualisierungen, die bei jedem Start vorgenommen werden, um die Unterstützung für AWS SDK-Integrationen zu erweitern, finden Sie unterÄnderungsprotokoll für unterstützte AWS SDK-Integrationen.

> **Important**
>
> Die Unterstützung für API-Aktionen wird vierteljährlich veröffentlicht. Aktualisierungen bereits unterstützter Aktionen, wie z. B. neue Parameter, sind möglicherweise nicht sofort verfügbar.

Service-Name	**Task**staatliche Ressource	Datum unterstützt	Präfix für Ausnahmen
AWS AppFabric	`arn:aws:states:::aws-sdk:appfabric: [apiAction]`	18. Januar 2024	AppFabric
B2B Data Interchange	`arn:aws:states:::aws-sdk:b2bi: [apiAction]`	18. Januar 2024	B2Bi
AWS Data Exports	`arn:aws:states:::aws-sdk:bcmdataexports: [apiAction]`	18. Januar 2024	BcmDataExports

Service-Name	**Task**staatliche Ressource	Datum unterstützt	Präfix für Ausnahmen
Amazon Bedrock	`arn:aws:states:::aws-sdk:bedrock:` *[apiAction]*	18. Januar 2024	Bedrock
Amazon Bedrock Agents	`arn:aws:states:::aws-sdk:bedrockagent:` *[apiAction]*	18. Januar 2024	BedrockAgent
Amazon Bedrock Runtime Agents	`arn:aws:states:::aws-sdk:bedrockagentruntime:` *[apiAction]*	18. Januar 2024	BedrockAgentRuntime
Amazon Bedrock Runtime	`arn:aws:states:::aws-sdk:bedrockruntime:` *[apiAction]*	18. Januar 2024	BedrockRuntime
AWS Clean Rooms	`arn:aws:states:::aws-sdk:cleanroomsml:` *[apiAction]*	18. Januar 2024	CleanRoomsMl
Amazon CloudFront KeyValueStore	`arn:aws:states:::aws-sdk:cloudfrontkeyvaluestore:` *[apiAction]*	18. Januar 2024	CloudFrontKeyValueStore
CodeGuru Security	`arn:aws:states:::aws-sdk:codegurusecurity:` *[apiAction]*	18. Januar 2024	CodeGuruSecurity
AWS Cost Optimization Hub	`arn:aws:states:::aws-sdk:costoptimizationhub:` *[apiAction]*	18. Januar 2024	CostOptimizationHub

Service-Name	**Task**staatliche Ressource	Datum unterstützt	Präfix für Ausnahmen
Amazon DataZone	`arn:aws:states:::aws-sdk:datazone:` *`[apiAction]`*	18. Januar 2024	DataZone
Amazon EKS Auth	`arn:aws:states:::aws-sdk:eksauth:` *`[apiAction]`*	18. Januar 2024	EksAuth
AWS Entity Resolution	`arn:aws:states:::aws-sdk:entityresolution:` *`[apiAction]`*	18. Januar 2024	EntityResolution
Kostenloses AWS-Kontingent	`arn:aws:states:::aws-sdk:freetier:` *`[apiAction]`*	18. Januar 2024	FreeTier
Amazon Inspector Scan	`arn:aws:states:::aws-sdk:inspectorscan:` *`[apiAction]`*	18. Januar 2024	InspectorScan
AWS Launch Wizard	`arn:aws:states:::aws-sdk:launchwizard:` *`[apiAction]`*	18. Januar 2024	LaunchWizard
Amazon Managed Blockchain Query	`arn:aws:states:::aws-sdk:managedblockchainquery:` *`[apiAction]`*	18. Januar 2024	ManagedBlockchainQuery
AWS Elemental MediaPackage V2	`arn:aws:states:::aws-sdk:mediapackagev2:` *`[apiAction]`*	18. Januar 2024	MediaPackageV2

Service-Name	**Task**staatliche Ressource	Datum unterstützt	Präfix für Ausnahmen
AWS HealthImaging	`arn:aws:states:::aws-sdk:medicalimaging: [apiAction]`	18. Januar 2024	MedicalImaging
Network Manager	`arn:aws:states:::aws-sdk:networkmanager: [apiAction]`	18. Januar 2024	NetworkManager
AWS Payment Cryptography	`arn:aws:states:::aws-sdk:paymentcryptography: [apiAction]`	18. Januar 2024	PaymentCryptography
AWS Payment Cryptography Data	`arn:aws:states:::aws-sdk:paymentcryptographydata: [apiAction]`	18. Januar 2024	PaymentCryptographyData
AWS Private CA Connector for Active Directory	`arn:aws:states:::aws-sdk:pcaconnectorad: [apiAction]`	18. Januar 2024	PcaConnectorAd
Amazon Q Business	`arn:aws:states:::aws-sdk:qbusiness: [apiAction]`	18. Januar 2024	QBusiness
Amazon Q Connect	`arn:aws:states:::aws-sdk:qconnect: [apiAction]`	18. Januar 2024	QConnect
AWS re:Post	`arn:aws:states:::aws-sdk:repostspace: [apiAction]`	18. Januar 2024	Repostspace

Service-Name	**Task**staatliche Ressource	Datum unterstützt	Präfix für Ausnahmen
Amazon Timestream Query	`arn:aws:states:::aws-sdk:timestreamquery:` *`[apiAction]`*	18. Januar 2024	TimestreamQuery
Amazon Timestream Write	`arn:aws:states:::aws-sdk:timestreamwrite:` *`[apiAction]`*	18. Januar 2024	TimestreamWrite
Trusted Advisor	`arn:aws:states:::aws-sdk:trustedadvisor:` *`[apiAction]`*	18. Januar 2024	TrustedAdvisor
Verified Permissions	`arn:aws:states:::aws-sdk:verifiedpermissions:` *`[apiAction]`*	18. Januar 2024	VerifiedPermissions
Amazon WorkSpaces Thin Client	`arn:aws:states:::aws-sdk:workspacesthinclient:` *`[apiAction]`*	18. Januar 2024	WorkSpacesThinClient
AWS CloudTrail Data	`arn:aws:states:::aws-sdk:cloudtraildata:` *`[apiAction]`*	16. Juni 2023	CloudTrailData
Amazon CloudWatch Internet Monitor	`arn:aws:states:::aws-sdk:internetmonitor:` *`[apiAction]`*	16. Juni 2023	InternetMonitor
Amazon Interactive Video Service RealTime	`arn:aws:states:::aws-sdk:ivsrealtime:` *`[apiAction]`*	16. Juni 2023	IvsRealTime

Service-Name	**Task**staatliche Ressource	Datum unterstützt	Präfix für Ausnahmen
AWS IoT TwinMaker	`arn:aws:states:::aws-sdk:iottwinmaker:` *`[apiAction]`*	16. Juni 2023	IoTTwinMaker
Amazon OpenSearch Ingestion	`arn:aws:states:::aws-sdk:osis:` *`[apiAction]`*	16. Juni 2023	Osis
AWS Telco Network Builder	`arn:aws:states:::aws-sdk:` **`tnb`**`:`*`[apiAction]`*	16. Juni 2023	Tnb
Amazon VPC Lattice	`arn:aws:states:::aws-sdk:vpclattice:` *`[apiAction]`*	16. Juni 2023	VpcLattice
Amazon Chime Media Pipelines	`arn:aws:states:::aws-sdk:chimesdkmediapipelines:` *`[apiAction]`*	17. Februar 2023	ChimeSdkMediaPipelines
Amazon Chime Voice	`arn:aws:states:::aws-sdk:chimesdkvoice:` *`[apiAction]`*	17. Februar 2023	ChimeSdkVoice
Amazon CodeCatalyst	`arn:aws:states:::aws-sdk:codecatalyst:` *`[apiAction]`*	17. Februar 2023	CodeCatalyst
Amazon Connect Cases	`arn:aws:states:::aws-sdk:connectcases:` *`[apiAction]`*	17. Februar 2023	ConnectCases

Service-Name	**Task**staatliche Ressource	Datum unterstützt	Präfix für Ausnahmen
Amazon DocumentDB Elastic Clusters	`arn:aws:states:::aws-sdk:docdbelastic: [apiAction]`	17. Februar 2023	DocDbElastic
Amazon EMR Serverless	`arn:aws:states:::aws-sdk:emrserverless: [apiAction]`	17. Februar 2023	EmrServerless
Amazon IVS Chat	`arn:aws:states:::aws-sdk:ivs: [apiAction]`	17. Februar 2023	Ivs
Amazon Kendra Intelligent Ranking	`arn:aws:states:::aws-sdk:kendraranking: [apiAction]`	17. Februar 2023	KendraRanking
AWS HealthOmics	`arn:aws:states:::aws-sdk:omics: [apiAction]`	17. Februar 2023	Omics
Amazon Redshift Serverless	`arn:aws:states:::aws-sdk:redshiftserverless: [apiAction]`	17. Februar 2023	RedshiftServerless
Amazon Security Lake	`arn:aws:states:::aws-sdk:securitylake: [apiAction]`	17. Februar 2023	SecurityLake
AWS Backup Storage	`arn:aws:states:::aws-sdk:backupstorage: [apiAction]`	17. Februar 2023	BackupStorage
AWS Clean Rooms	`arn:aws:states:::aws-sdk:cleanrooms: [apiAction]`	17. Februar 2023	CleanRooms

Service-Name	**Task**staatliche Ressource	Datum unterstützt	Präfix für Ausnahmen
AWS Control Tower	`arn:aws:states:::aws-sdk:controltower: [apiAction]`	17. Februar 2023	ControlTower
AWS Health	`arn:aws:states:::aws-sdk:health: [apiAction]`	17. Februar 2023	Health
AWS IoT FleetWise	`arn:aws:states:::aws-sdk:iotfleetwise: [apiAction]`	17. Februar 2023	IotFleetWise
AWS Mainframe Modernization	`arn:aws:states:::aws-sdk:m2: [apiAction]`	17. Februar 2023	M2
AWS Migration Hub Orchestrator	`arn:aws:states:::aws-sdk:migrationhuborchestrator: [apiAction]`	17. Februar 2023	Migration HubOrchestrator
AWS Private 5G	`arn:aws:states:::aws-sdk:privatenetworks: [apiAction]`	17. Februar 2023	PrivateNetworks
AWS Ressourcen Explorer	`arn:aws:states:::aws-sdk:resourceexplorer2: [apiAction]`	17. Februar 2023	ResourceExplorer2
AWS SimSpace Weaver	`arn:aws:states:::aws-sdk:simspaceweaver: [apiAction]`	17. Februar 2023	SimSpaceWeaver

Service-Name	**Task**staatliche Ressource	Datum unterstützt	Präfix für Ausnahmen
AWS Support App	`arn:aws:states:::aws-sdk:supportapp:` *[apiAction]*	17. Februar 2023	SupportApp
CloudWatch Observability Access Manager	`arn:aws:states:::aws-sdk:oam:` *[apiAction]*	17. Februar 2023	Oam
EventBridge Pipes	`arn:aws:states:::aws-sdk:pipes:` *[apiAction]*	17. Februar 2023	Pipes
EventBridge Scheduler	`arn:aws:states:::aws-sdk:scheduler:` *[apiAction]*	17. Februar 2023	Scheduler
IAM Roles Anywhere	`arn:aws:states:::aws-sdk:rolesanywhere:` *[apiAction]*	17. Februar 2023	RolesAnywhere
Kinesis Video WebRTC Storage	`arn:aws:states:::aws-sdk:kinesisvideowebrtcstorage:` *[apiAction]*	17. Februar 2023	KinesisVideoWebRtcStorage
License Manager Linux Subscriptions	`arn:aws:states:::aws-sdk:licensemanagerlinuxsubscriptions:` *[apiAction]*	17. Februar 2023	LicenseManagerLinuxSubscriptions
License Manager User Subscriptions	`arn:aws:states:::aws-sdk:licensemanagerusersubscriptions:` *[apiAction]*	17. Februar 2023	LicenseManagerUserSubscriptions

Service-Name	**Task**staatliche Ressource	Datum unterstützt	Präfix für Ausnahmen
OpenSearch Serverless	`arn:aws:states:::aws-sdk:opensearchserverless:` *[apiAction]*	17. Februar 2023	OpenSearchServerless
Route 53 ARC Zonal Shift	`arn:aws:states:::aws-sdk:arczonalshift:` *[apiAction]*	17. Februar 2023	ArcZonalShift
SageMaker Geospatial	`arn:aws:states:::aws-sdk:sagemakergeospatial:` *[apiAction]*	17. Februar 2023	SageMaker Geospatial
SageMaker Metrics	`arn:aws:states:::aws-sdk:sagemakermetrics:` *[apiAction]*	17. Februar 2023	SageMakerMetrics
Systems Manager for SAP	`arn:aws:states:::aws-sdk:ssmsap:` *[apiAction]*	17. Februar 2023	SsmSap
AWS Account Management	`arn:aws:states:::aws-sdk:account:` *[apiAction]*	19. April 2022	Account
AWS Amplify	`arn:aws:states:::aws-sdk:amplify:` *[apiAction]*	30. September 2021	Amplify
AWS App Mesh	`arn:aws:states:::aws-sdk:appmesh:` *[apiAction]*	30. September 2021	AppMesh

Service-Name	**Task**staatliche Ressource	Datum unterstützt	Präfix für Ausnahmen
AWS App Runner	`arn:aws:states:::aws-sdk:apprunner: [apiAction]`	30. September 2021	AppRunner
AWS AppConfig	`arn:aws:states:::aws-sdk:appconfig: [apiAction]`	30. September 2021	AppConfig
AWS AppConfig Data	`arn:aws:states:::aws-sdk:appconfigdata: [apiAction]`	19. April 2022	AppConfigData
AWS AppSync	`arn:aws:states:::aws-sdk:appsync: [apiAction]`	30. September 2021	AppSync
AWS Application Discovery Service	`arn:aws:states:::aws-sdk:applicationdiscovery: [apiAction]` ***	30. September 2021	ApplicationDiscovery
AWS Application Migration Service	`arn:aws:states:::aws-sdk:mgn: [apiAction]`	30. September 2021	Mgn
AWS Audit Manager	`arn:aws:states:::aws-sdk:auditmanager: [apiAction]`	30. September 2021	AuditManager
AWS Auto Scaling Plans	`arn:aws:states:::aws-sdk:autoscalingplans: [apiAction]`	30. September 2021	AutoScalingPlans

Service-Name	**Task**staatliche Ressource	Datum unterstützt	Präfix für Ausnahmen
AWS Backup	`arn:aws:states:::aws-sdk:backup:` *[apiAction]*	30. September 2021	Backup
AWS Backup gateway	`arn:aws:states:::aws-sdk:backupgateway:` *[apiAction]*	19. April 2022	BackupGateway
AWS Batch	`arn:aws:states:::aws-sdk:batch:` *[apiAction]*	30. September 2021	Batch
AWS Billing Conductor	`arn:aws:states:::aws-sdk:billingconductor:` *[apiAction]*	26. Juli 2022	Billingconductor
AWS Budgets	`arn:aws:states:::aws-sdk:budgets:` *[apiAction]*	30. September 2021	Budgets
AWS Certificate Manager	`arn:aws:states:::aws-sdk:acm:` *[apiAction]*	30. September 2021	Acm
AWS Private Certificate Authority	`arn:aws:states:::aws-sdk:acmpca:` *[apiAction]*	30. September 2021	AcmPca
AWS Cloud Map	`arn:aws:states:::aws-sdk:servicediscovery:` *[apiAction]*	30. September 2021	ServiceDiscovery
AWS Cloud9	`arn:aws:states:::aws-sdk:cloud9:` *[apiAction]*	30. September 2021	Cloud9

Service-Name	**Task**staatliche Ressource	Datum unterstützt	Präfix für Ausnahmen
AWS CloudFormation	`arn:aws:states:::aws-sdk:cloudformation: [apiAction]`	30. September 2021	CloudFormation
AWS CloudHSM	`arn:aws:states:::aws-sdk:cloudhsm: [apiAction]`	30. September 2021	CloudHsm
AWS CloudHSM	`arn:aws:states:::aws-sdk:cloudhsmv2: [apiAction]`	30. September 2021	CloudHsmV2
AWS CloudTrail	`arn:aws:states:::aws-sdk:cloudtrail: [apiAction]`	30. September 2021	CloudTrail
AWS Cloud Control	`arn:aws:states:::aws-sdk:cloudcontrol: [apiAction]`	19. April 2022	CloudControl
AWS CodeBuild	`arn:aws:states:::aws-sdk:codebuild: [apiAction]`	30. September 2021	CodeBuild
AWS CodeCommit	`arn:aws:states:::aws-sdk:codecommit: [apiAction]`	30. September 2021	CodeCommit
AWS CodeDeploy	`arn:aws:states:::aws-sdk:codedeploy: [apiAction]` *** —	30. September 2021	CodeDeploy
AWS CodePipeline	`arn:aws:states:::aws-sdk:codepipeline: [apiAction]`	30. September 2021	CodePipeline

Service-Name	**Task**staatliche Ressource	Datum unterstützt	Präfix für Ausnahmen
AWS CodeStar	`arn:aws:states:::aws-sdk:codestar: [apiAction]`	30. September 2021	CodeStar
AWS CodeStar	`arn:aws:states:::aws-sdk:codestarnotifications: [apiAction]`	30. September 2021	CodestarNotifications
AWS CodeStar	`arn:aws:states:::aws-sdk:codestarconnections: [apiAction]`	30. September 2021	CodeStarConnections
AWS Compute Optimizer	`arn:aws:states:::aws-sdk:computeoptimizer: [apiAction]`	30. September 2021	ComputeOptimizer
AWS Config	`arn:aws:states:::aws-sdk:config: [apiAction]`	30. September 2021	Config
AWS Cost Explorer Service	`arn:aws:states:::aws-sdk:costexplorer: [apiAction]`	30. September 2021	CostExplorer
AWS Cost and Usage Report	`arn:aws:states:::aws-sdk:costandusagereport: [apiAction]`	30. September 2021	CostAndUsageReport
AWS Data Exchange	`arn:aws:states:::aws-sdk:dataexchange: [apiAction]`	30. September 2021	DataExchange

Service-Name	**Task**staatliche Ressource	Datum unterstützt	Präfix für Ausnahmen
AWS Data Pipeline	`arn:aws:states:::aws-sdk:datapipeline:` *[apiAction]*	30. September 2021	DataPipeline
AWS DataSync	`arn:aws:states:::aws-sdk:datasync:` *[apiAction]*	30. September 2021	DataSync
AWS Database Migration Service	`arn:aws:states:::aws-sdk:databasemigration:` *[apiAction]*	30. September 2021	DatabaseMigration
AWS Device Farm	`arn:aws:states:::aws-sdk:devicefarm:` *[apiAction]*	30. September 2021	DeviceFarm
AWS Direct Connect	`arn:aws:states:::aws-sdk:directconnect:` *[apiAction]* ***	30. September 2021	DirectConnect
AWS Directory Service	`arn:aws:states:::aws-sdk:directory:` *[apiAction]*	30. September 2021	Directory
AWS EC2 Instance Connect	`arn:aws:states:::aws-sdk:ec2instanceconnect:` *[apiAction]*	30. September 2021	Ec2InstanceConnect
AWS Elastic Beanstalk	`arn:aws:states:::aws-sdk:elasticbeanstalk:` *[apiAction]*	30. September 2021	ElasticBeanstalk
AWS Elemental MediaLive	`arn:aws:states:::aws-sdk:medialive:` *[apiAction]*	30. September 2021	MediaLive

Service-Name	**Task**staatliche Ressource	Datum unterstützt	Präfix für Ausnahmen
AWS Elemental MediaPackage	`arn:aws:states:::aws-sdk:mediapackage: [apiAction]` *** —	30. September 2021	MediaPackage
AWS Elemental MediaPackage VOD	`arn:aws:states:::aws-sdk:mediapackagevod: [apiAction]`	30. September 2021	MediaPackageVod
AWS Elemental MediaStore	`arn:aws:states:::aws-sdk:mediastore: [apiAction]`	30. September 2021	MediaStore
AWS Fault Injection Service	`arn:aws:states:::aws-sdk:fis: [apiAction]`	30. September 2021	Fis
AWS Firewall Manager	`arn:aws:states:::aws-sdk:fms: [apiAction]`	30. September 2021	Fms
AWS Glue	`arn:aws:states:::aws-sdk:glue: [apiAction]`	30. September 2021	Glue
AWS Glue DataBrew	`arn:aws:states:::aws-sdk:databrew: [apiAction]`	30. September 2021	DataBrew
AWS IoT Greengrass	`arn:aws:states:::aws-sdk:greengrass: [apiAction]`	30. September 2021	Greengrass
AWS Ground Station	`arn:aws:states:::aws-sdk:groundstation: [apiAction]`	30. September 2021	GroundStation

Service-Name	**Task**staatliche Ressource	Datum unterstützt	Präfix für Ausnahmen
AWS Identity and Access Management	`arn:aws:states:::aws-sdk:iam:` *`[apiAction]`*	30. September 2021	iam
AWS IoT	`arn:aws:states:::aws-sdk:iot:` *`[apiAction]`* ***	30. September 2021	iot
AWS IoT 1-Click	`arn:aws:states:::aws-sdk:iot1clickprojects:` *`[apiAction]`*	30. September 2021	iot1ClickProjects
AWS IoT Analytics	`arn:aws:states:::aws-sdk:iotanalytics:` *`[apiAction]`*	30. September 2021	IoTAnalytics
AWS IoT Core Device Advisor	`arn:aws:states:::aws-sdk:iotdeviceadvisor:` *`[apiAction]`* ***	30. September 2021	IotDeviceAdvisor
AWS IoT Events	`arn:aws:states:::aws-sdk:ioteventsdata:` *`[apiAction]`*	30. September 2021	IotEvents
AWS IoT Events-Daten	`arn:aws:states:::aws-sdk:ioteventsdata:` *`[apiAction]`*	30. September 2021	IotEventsData
AWS IoT Fleet Hub	`arn:aws:states:::aws-sdk:iotfleethub:` *`[apiAction]`*	30. September 2021	IoTFleetHub
AWS IoT Greengrass Version 2	`arn:aws:states:::aws-sdk:greengrassv2:` *`[apiAction]`*	30. September 2021	GreengrassV2

Service-Name	**Task**staatliche Ressource	Datum unterstützt	Präfix für Ausnahmen
AWS IoT-Auftragsdaten Plane	`arn:aws:states:::aws-sdk:iotjobsdataplane: [apiAction]`	30. September 2021	IotJobsDataPlane
AWS IoT Secure Tunneling	`arn:aws:states:::aws-sdk:iotsecuretunneling: [apiAction]`	30. September 2021	IoTSecure Tunneling
AWS IoT SiteWise	`arn:aws:states:::aws-sdk:iotsitewise: [apiAction]`	30. September 2021	IoTSiteWise
AWS IoT Wireless	`arn:aws:states:::aws-sdk:iotwireless: [apiAction]`	30. September 2021	IotWireless
AWS Key Management Service	`arn:aws:states:::aws-sdk:kms: [apiAction]`	30. September 2021	Kms
AWS Lake Formation	`arn:aws:states:::aws-sdk:lakeformation: [apiAction]`	30. September 2021	LakeFormation
AWS Lambda	`arn:aws:states:::aws-sdk:lambda: [apiAction]` *** —	30. September 2021	Lambda
AWS License Manager	`arn:aws:states:::aws-sdk:licensemanager: [apiAction]`	30. September 2021	LicenseManager
AWS Marketplace	`arn:aws:states:::aws-sdk:marketplacecatalog: [apiAction]`	30. September 2021	MarketplaceCatalog

Service-Name	**Task**staatliche Ressource	Datum unterstützt	Präfix für Ausnahmen
AWS Marketplace Commerce Analytics	`arn:aws:states:::aws-sdk:marketplacecommerceanalytics:` *`[apiAction]`*	30. September 2021	MarketplaceCommerceAnalytics
AWS Marketplace Entitlement Service	`arn:aws:states:::aws-sdk:marketplaceentitlement:` *`[apiAction]`*	30. September 2021	MarketplaceEntitlement
AWS Elemental MediaTailor	`arn:aws:states:::aws-sdk:mediatailor:` *`[apiAction]`*	30. September 2021	MediaTailor
AWS Migration Hub	`arn:aws:states:::aws-sdk:migrationhub:` *`[apiAction]`*	30. September 2021	MigrationHub
AWS Migration Hub Config	`arn:aws:states:::aws-sdk:migrationhubconfig:` *`[apiAction]`*	30. September 2021	MigrationHubConfig
Strategieempfehlungen für den AWS Migration Hub	`arn:aws:states:::aws-sdk:migrationhubstrategy:` *`[apiAction]`*	19. April 2022	MigrationHubStrategy
AWS Mobile	`arn:aws:states:::aws-sdk:mobile:` *`[apiAction]`*	30. September 2021	
AWS Network Firewall	`arn:aws:states:::aws-sdk:networkfirewall:` *`[apiAction]`*	30. September 2021	NetworkFirewall

Service-Name	**Task**staatliche Ressource	Datum unterstützt	Präfix für Ausnahmen
AWS OpsWorks	`arn:aws:states:::aws-sdk:opsworks: [apiAction]`	30. September 2021	OpsWorks
AWS OpsWorks CM	`arn:aws:states:::aws-sdk:opsworkscm: [apiAction]`	30. September 2021	OpsWorksCm
AWS Organizations	`arn:aws:states:::aws-sdk:organizations: [apiAction]`	30. September 2021	Organizations
AWS Outposts	`arn:aws:states:::aws-sdk:outposts: [apiAction]`	30. September 2021	Outposts
AWS Panorama	`arn:aws:states:::aws-sdk:panorama: [apiAction]`	19. April 2022	Panorama
Amazon Relational Database Service Performance Insights	`arn:aws:states:::aws-sdk:pi: [apiAction]`	30. September 2021	Pi
AWS-Preisliste	`arn:aws:states:::aws-sdk:pricing: [apiAction]`	30. September 2021	Pricing
Amazon Relational Database Service	`arn:aws:states:::aws-sdk:rdsdata: [apiAction]` ***	30. September 2021	RdsData

Service-Name	**Task**staatliche Ressource	Datum unterstützt	Präfix für Ausnahmen
AWS Resilience Hub	`arn:aws:states:::aws-sdk:resiliencehub: [apiAction]`	19. April 2022	Resiliencehub
AWS Resource Access Manager	`arn:aws:states:::aws-sdk:ram: [apiAction]`	30. September 2021	Ram
AWS Resource Groups	`arn:aws:states:::aws-sdk:resourcegroups: [apiAction]`	30. September 2021	ResourceGroups
AWS Resource Groups Tagging API	`arn:aws:states:::aws-sdk:resourcegroupstaggingapi: [apiAction]`	30. September 2021	ResourceGroupsTaggingApi
AWS RoboMaker	`arn:aws:states:::aws-sdk:robomaker: [apiAction]`	30. September 2021	RoboMaker
AWS IAM Identity Center	`arn:aws:states:::aws-sdk:identitystore: [apiAction]`	30. September 2021	Identitystore
IAM Identity Center OIDC	`arn:aws:states:::aws-sdk:ssooidc: [apiAction]`	30. September 2021	SsoOidc
AWS Secrets Manager	`arn:aws:states:::aws-sdk:secretsmanager: [apiAction]`	30. September 2021	SecretsManager

Service-Name	**Task**staatliche Ressource	Datum unterstützt	Präfix für Ausnahmen
AWS Security Token Service	`arn:aws:states:::aws-sdk:sts:` *`[apiAction]`* ***	30. September 2021	Sts
AWS Security Hub	`arn:aws:states:::aws-sdk:securityhub:` *`[apiAction]`*	30. September 2021	SecurityHub
AWS Server Migration Service	`arn:aws:states:::aws-sdk:sms:` *`[apiAction]`*	30. September 2021	Sms
AWS Service Catalog	`arn:aws:states:::aws-sdk:servicecatalog:` *`[apiAction]`*	30. September 2021	ServiceCatalog
AWS Service Catalog AppRegistry	`arn:aws:states:::aws-sdk:servicecatalogappregistry:` *`[apiAction]`*	30. September 2021	ServiceCatalogAppRegistry
AWS Shield	`arn:aws:states:::aws-sdk:shield:` *`[apiAction]`* ***	30. September 2021	Shield
AWS Signer	`arn:aws:states:::aws-sdk:signer:` *`[apiAction]`*	30. September 2021	Signer
IAM Identity Center	`arn:aws:states:::aws-sdk:sso:` *`[apiAction]`*	30. September 2021	Sso

Service-Name	**Task**staatliche Ressource	Datum unterstützt	Präfix für Ausnahmen
IAM Identity Center Admin	`arn:aws:states:::aws-sdk:ssoadmin: [apiAction]`	30. September 2021	SsoAdmin
AWS Step Functions	`arn:aws:states:::aws-sdk:sfn: [apiAction]`	30. September 2021	Sfn
AWS Storage Gateway	`arn:aws:states:::aws-sdk:storagegateway: [apiAction]`	30. September 2021	StorageGateway
AWS Support	`arn:aws:states:::aws-sdk:support: [apiAction]`	30. September 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. September 2021	Transfer
AWS WAF	`arn:aws:states:::aws-sdk:waf: [apiAction]`	30. September 2021	Waf
AWS WAF Regional	`arn:aws:states:::aws-sdk:wafregional: [apiAction]`	30. September 2021	WafRegional
AWS WAFV2	`arn:aws:states:::aws-sdk:wafv2: [apiAction]`	30. September 2021	Wafv2

Service-Name	**Task**staatliche Ressource	Datum unterstützt	Präfix für Ausnahmen
AWS Well-Architected Tool	`arn:aws:states:::aws-sdk:wellarchitected: [apiAction]`	30. September 2021	WellArchitected
AWS X-Ray	`arn:aws:states:::aws-sdk:xray: [apiAction]`	30. September 2021	XRay
AWS Marketplace Metering Service	`arn:aws:states:::aws-sdk:marketplacemetering: [apiAction]`	30. September 2021	MarketplaceMetering
AWS Serverless Application Repository	`arn:aws:states:::aws-sdk:serverlessapplicationrepository: [apiAction]`	30. September 2021	ServerlessApplicationRepository
AWS Identity and Access Management Access Analyzer	`arn:aws:states:::aws-sdk:accessanalyzer: [apiAction]`	30. September 2021	AccessAnalyzer
Alexa for Business	`arn:aws:states:::aws-sdk:alexaforbusiness: [apiAction]`	30. September 2021	AlexaForBusiness
Amazon API Gateway	`arn:aws:states:::aws-sdk:apigateway: [apiAction]`	30. September 2021	ApiGateway
Amazon API Gateway	`arn:aws:states:::aws-sdk:apigatewayv2: [apiAction]`	30. September 2021	ApiGatewayV2

Service-Name	**Task**staatliche Ressource	Datum unterstützt	Präfix für Ausnahmen
Amazon AppIntegrations	`arn:aws:states:::aws-sdk:appintegrations: [apiAction]`	30. September 2021	AppIntegrations
Amazon AppStream 2.0	`arn:aws:states:::aws-sdk:appstream: [apiAction]`	30. September 2021	AppStream
Amazon AppFlow	`arn:aws:states:::aws-sdk:appflow: [apiAction]`	30. September 2021	Appflow
Amazon Athena	`arn:aws:states:::aws-sdk:athena: [apiAction]`	30. September 2021	Athena
Amazon Augmented AI	`arn:aws:states:::aws-sdk:sagemakera2iruntime: [apiAction]`	30. September 2021	SageMaker A2IRuntime
Amazon Braket	`arn:aws:states:::aws-sdk:braket: [apiAction]`	30. September 2021	Braket
Amazon Chime	`arn:aws:states:::aws-sdk:chime: [apiAction]`	30. September 2021	Chime
Amazon Chime Meetings	`arn:aws:states:::aws-sdk:chimesdkmeetings: [apiAction]`	19. April 2022	ChimeSdkMeetings

Service-Name	**Task**staatliche Ressource	Datum unterstützt	Präfix für Ausnahmen
Amazon Cloud Directory	`arn:aws:states:::aws-sdk:clouddirectory: [apiAction]`	30. September 2021	CloudDirectory
Amazon CloudFront	`arn:aws:states:::aws-sdk:cloudfront: [apiAction]`	30. September 2021	CloudFront
Amazon CloudSearch	`arn:aws:states:::aws-sdk:cloudsearch: [apiAction]`	30. September 2021	CloudSearch
Amazon CloudWatch	`arn:aws:states:::aws-sdk:cloudwatch: [apiAction]`	30. September 2021	CloudWatch
Amazon CloudWatch Application Insights	`arn:aws:states:::aws-sdk:applicationinsights: [apiAction]`	30. September 2021	ApplicationInsights
CloudWatch Evidently	`arn:aws:states:::aws-sdk:evidently: [apiAction]`	19. April 2022	Evidently
Amazon CloudWatch Logs	`arn:aws:states:::aws-sdk:cloudwatchlogs: [apiAction]`	30. September 2021	CloudWatchLogs
Amazon CloudWatch RUM	`arn:aws:states:::aws-sdk:rum: [apiAction]`	19. April 2022	Rum

Service-Name	**Task**staatliche Ressource	Datum unterstützt	Präfix für Ausnahmen
Amazon CloudWatch Synthetics	`arn:aws:states:::aws-sdk:synthetics:` *`[apiAction]`*	30. September 2021	Synthetics
Amazon CodeGuru Profiler	`arn:aws:states:::aws-sdk:codeguruprofiler:` *`[apiAction]`*	30. September 2021	CodeGuruProfiler
Amazon CodeGuru Reviewer	`arn:aws:states:::aws-sdk:codegurureviewer:` *`[apiAction]`*	30. September 2021	CodeGuruReviewer
Amazon Cognito	`arn:aws:states:::aws-sdk:cognitoidentity:` *`[apiAction]`*	30. September 2021	CognitoIdentity
Amazon Cognito Identity Provider	`arn:aws:states:::aws-sdk:cognitoidentityprovider:` *`[apiAction]`*	30. September 2021	CognitoIdentityProvider
Amazon Cognito Sync	`arn:aws:states:::aws-sdk:cognitosync:` *`[apiAction]`*	30. September 2021	CognitoSync
Amazon Comprehend	`arn:aws:states:::aws-sdk:comprehend:` *`[apiAction]`*	30. September 2021	Comprehend
Amazon Comprehend Medical	`arn:aws:states:::aws-sdk:comprehendmedical:` *`[apiAction]`* *** —	30. September 2021	ComprehendMedical

Service-Name	**Task**staatliche Ressource	Datum unterstützt	Präfix für Ausnahmen
Amazon Connect Contact Lens	`arn:aws:states:::aws-sdk:connectcontactlens: [apiAction]`	30. September 2021	ConnectContactLens
Amazon Connect Participant Service	`arn:aws:states:::aws-sdk:connectparticipant: [apiAction]`	30. September 2021	ConnectParticipant
Amazon Connect	`arn:aws:states:::aws-sdk:connect: [apiAction]`	30. September 2021	Connect
Amazon Connect Voice ID	`arn:aws:states:::aws-sdk:voiceid: [apiAction]`	19. April 2022	VoiceId
Amazon Connect Wisdom	`arn:aws:states:::aws-sdk:wisdom: [apiAction]`	19. April 2022	Wisdom
Amazon Data Lifecycle Manager	`arn:aws:states:::aws-sdk:dlm: [apiAction]`	30. September 2021	Dlm
Amazon Detective	`arn:aws:states:::aws-sdk:detective: [apiAction]`	30. September 2021	Detective
Amazon DevOps Guru	`arn:aws:states:::aws-sdk:devopsguru: [apiAction]`	30. September 2021	DevOpsGuru

Service-Name	**Task**staatliche Ressource	Datum unterstützt	Präfix für Ausnahmen
Amazon DocumentDB (with MongoDB compatibility)	`arn:aws:states:::aws-sdk:docdb: [apiAction]`	30. September 2021	DocDb
Amazon DynamoDB	`arn:aws:states:::aws-sdk:dynamodb: [apiAction]`	30. September 2021	DynamoDb
Amazon DynamoDB Streams	`arn:aws:states:::aws-sdk:dynamodbstreams: [apiAction]`	30. September 2021	DynamoDbStreams
Amazon EC2 Container Registry	`arn:aws:states:::aws-sdk:ecr: [apiAction]`	30. September 2021	Ecr
Amazon EC2 Container Service	`arn:aws:states:::aws-sdk:ecs: [apiAction]`	30. September 2021	Ecs
Amazon EC2 Systems Manager	`arn:aws:states:::aws-sdk:ssm: [apiAction]`	30. September 2021	Ssm
Amazon EMR	`arn:aws:states:::aws-sdk:emrcontainers: [apiAction]` ***	30. September 2021	EmrContainers
Amazon ElastiCache	`arn:aws:states:::aws-sdk:elasticache: [apiAction]`	30. September 2021	ElastiCache

Service-Name	**Task**staatliche Ressource	Datum unterstützt	Präfix für Ausnahmen
Amazon Elastic Inference	`arn:aws:states:::aws-sdk:elasticinference: [apiAction]`	30. September 2021	ElasticInference
Amazon Elastic Block Store	`arn:aws:states:::aws-sdk:ebs: [apiAction]`	30. September 2021	Ebs
Amazon Elastic Compute Cloud	`arn:aws:states:::aws-sdk:ec2: [apiAction]`	30. September 2021	Ec2
Amazon Elastic Container Registry Public	`arn:aws:states:::aws-sdk:ecrpublic: [apiAction]`	30. September 2021	EcrPublic
Amazon Elastic File System	`arn:aws:states:::aws-sdk:efs: [apiAction]` ***	30. September 2021	Efs
Amazon Elastic Kubernetes Service	`arn:aws:states:::aws-sdk:eks: [apiAction]`	30. September 2021	Eks
Amazon EMR	`arn:aws:states:::aws-sdk:emr: [apiAction]`	30. September 2021	Emr
Amazon Elastic Transcoder	`arn:aws:states:::aws-sdk:elastictranscoder: [apiAction]` ***	30. September 2021	ElasticTranscoder
Amazon OpenSearch Service	`arn:aws:states:::aws-sdk:elasticsearch: [apiAction]`	30. September 2021	Elasticsearch

Service-Name	**Task**staatliche Ressource	Datum unterstützt	Präfix für Ausnahmen
Amazon OpenSearch Service	`arn:aws:states:::aws-sdk:opensearch:` *`[apiAction]`*	19. April 2022	OpenSearch
Amazon EventBridge	`arn:aws:states:::aws-sdk:eventbridge:` *`[apiAction]`*	30. September 2021	EventBridge
Amazon FSx	`arn:aws:states:::aws-sdk:fsx:` *`[apiAction]`*	30. September 2021	FSx
Amazon Forecast Query	`arn:aws:states:::aws-sdk:forecastquery:` *`[apiAction]`*	30. September 2021	Forecastquery
Amazon Forecast Service	`arn:aws:states:::aws-sdk:forecast:` *`[apiAction]`*	30. September 2021	Forecast
Amazon Fraud Detector	`arn:aws:states:::aws-sdk:frauddetector:` *`[apiAction]`*	30. September 2021	FraudDetector
Amazon GameLift	`arn:aws:states:::aws-sdk:gamelift:` *`[apiAction]`*	30. September 2021	Amazon GameLift
Amazon GameSparks	`arn:aws:states:::aws-sdk:gamesparks:` *`[apiAction]`*	27. Juli 2022	GameSparks
Amazon S3 Glacier	`arn:aws:states:::aws-sdk:glacier:` *`[apiAction]`*	30. September 2021	Glacier

Service-Name	**Task**staatliche Ressource	Datum unterstützt	Präfix für Ausnahmen
Amazon GuardDuty	`arn:aws:states:::aws-sdk:guardduty:` *`[apiAction]`*	30. September 2021	GuardDuty
AWS HealthLake	`arn:aws:states:::aws-sdk:healthlake:` *`[apiAction]`*	30. September 2021	HealthLake
Amazon Honeycode	`arn:aws:states:::aws-sdk:honeycode:` *`[apiAction]`*	30. September 2021	Honeycode
Amazon Inspector	`arn:aws:states:::aws-sdk:inspector:` *`[apiAction]`*	30. September 2021	Inspector
Amazon Inspector V2	`arn:aws:states:::aws-sdk:inspector2:` *`[apiAction]`*	19. April 2022	Inspector2
Amazon Interactive Video Service	`arn:aws:states:::aws-sdk:ivs:` *`[apiAction]`*	30. September 2021	Ivs
Amazon Kendra	`arn:aws:states:::aws-sdk:kendra:` *`[apiAction]`*	30. September 2021	Kendra
Amazon Kinesis	`arn:aws:states:::aws-sdk:kinesis:` *`[apiAction]`* *** —	30. September 2021	Kinesis
Amazon Kinesis Analytics	`arn:aws:states:::aws-sdk:kinesisanalytics:` *`[apiAction]`*	30. September 2021	KinesisAnalytics

Service-Name	**Task**staatliche Ressource	Datum unterstützt	Präfix für Ausnahmen
Amazon Kinesis Analytics V2	`arn:aws:states:::aws-sdk:kinesisanalyticsv2: [apiAction]`	30. September 2021	KinesisAnalyticsV2
Amazon Kinesis Firehose	`arn:aws:states:::aws-sdk:firehose: [apiAction]`	30. September 2021	Firehose
Amazon Kinesis Video Signaling Channels	`arn:aws:states:::aws-sdk:kinesisvideosignaling: [apiAction]`	30. September 2021	KinesisVideoSignaling
Amazon Kinesis Video Streams	`arn:aws:states:::aws-sdk:kinesisvideo: [apiAction]`	30. September 2021	KinesisVideo
Amazon Kinesis Video Streams Archived Media	`arn:aws:states:::aws-sdk:kinesisvideoarchivedmedia: [apiAction]`	30. September 2021	KinesisVideoArchivedMedia
Amazon Kinesis video stream	`arn:aws:states:::aws-sdk:kinesisvideomedia: [apiAction]`	30. September 2021	KinesisVideoMedia
Amazon Lex Model Building Service	`arn:aws:states:::aws-sdk:lexmodelbuilding: [apiAction]`	30. September 2021	LexModelBuilding
Amazon Lex Model Building Service V2	`arn:aws:states:::aws-sdk:lexmodelsv2: [apiAction]`	30. September 2021	LexModelsV2

Service-Name	**Task**staatliche Ressource	Datum unterstützt	Präfix für Ausnahmen
Amazon Lex	`arn:aws:states:::aws-sdk:lexruntime:` *[apiAction]*	30. September 2021	LexRuntime
Amazon Lex Runtime V2	`arn:aws:states:::aws-sdk:lexruntimev2:` *[apiAction]* *** —	30. September 2021	LexRuntimeV2
Amazon Lightsail	`arn:aws:states:::aws-sdk:lightsail:` *[apiAction]*	30. September 2021	Lightsail
Amazon Location Service	`arn:aws:states:::aws-sdk:location:` *[apiAction]*	30. September 2021	Location
Amazon Lookout for Equipment	`arn:aws::states:::aws-sdk:lookoutequipment:` *[apiAction]*	30. September 2021	LookoutEquipment
Amazon Lookout for Metrics	`arn:aws:states:::aws-sdk:lookoutmetrics:` *[apiAction]*	30. September 2021	LookoutMetrics
Amazon Lookout for Vision	`arn:aws:states:::aws-sdk:lookoutvision:` *[apiAction]*	30. September 2021	LookoutVision
Amazon MQ	`arn:aws:states:::aws-sdk:mq:` *[apiAction]*	30. September 2021	Mq
Amazon Macie	`arn:aws:states:::aws-sdk:macie:` *[apiAction]*	30. September 2021	

Service-Name	**Task**staatliche Ressource	Datum unterstützt	Präfix für Ausnahmen
Amazon Macie 2	`arn:aws:states:::aws-sdk:macie2: [apiAction]`	30. September 2021	Macie2
Amazon Managed Blockchain	`arn:aws:states:::aws-sdk:managedblockchain: [apiAction]`	30. September 2021	ManagedBlockchain
Amazon Managed Grafana	`arn:aws:states:::aws-sdk:grafana: [apiAction]`	19. April 2022	Grafana
Amazon Managed Service for Prometheus	`arn:aws:states:::aws-sdk:amp: [apiAction]`	30. September 2021	Amp
Amazon Managed Streaming for Apache Kafka	`arn:aws:states:::aws-sdk:kafka: [apiAction]`	30. September 2021	Kafka
Amazon MSK Connect	`arn:aws:states:::aws-sdk:kafkaconnect: [apiAction]`	19. April 2022	KafkaConnect
Amazon Managed Workflows for Apache Airflow	`arn:aws:states:::aws-sdk:mwaa: [apiAction]`	30. September 2021	Mwaa
Amazon Mechanical Turk	`arn:aws:states:::aws-sdk:mturk: [apiAction]`	30. September 2021	MTurk
Amazon MemoryDB for Redis	`arn:aws:states:::aws-sdk:memorydb: [apiAction]`	19. April 2022	MemoryDB

Service-Name	**Task**staatliche Ressource	Datum unterstützt	Präfix für Ausnahmen
Amazon Nimble Studio	`arn:aws:states:::aws-sdk:nimble: [apiAction]`	30. September 2021	Nimble
Amazon Personalize	`arn:aws:states:::aws-sdk:personalize: [apiAction]`	30. September 2021	Personalize
Amazon Personalize Events	`arn:aws:states:::aws-sdk:personalizeevents: [apiAction]`	30. September 2021	PersonalizeEvents
Amazon Personalize Runtime	`arn:aws:states:::aws-sdk:personalizeruntime: [apiAction]`	30. September 2021	PersonalizeRuntime
Amazon Pinpoint	`arn:aws:states:::aws-sdk:pinpoint: [apiAction]`	30. September 2021	Pinpoint
Amazon Pinpoint Email Service	`arn:aws:states:::aws-sdk:pinpointemail: [apiAction]`	30. September 2021	PinpointEmail
Amazon Pinpoint SMS and Voice Service	`arn:aws:states:::aws-sdk:pinpointsmsvoice: [apiAction]`	30. September 2021	PinpointSmsVoice
Amazon Pinpoint SMS and Voice V2 Service	`arn:aws:states:::aws-sdk:pinpointsmsvoicev2: [apiAction]`	27. Juli 2022	PinpointSmsVoiceV2
Amazon Polly	`arn:aws:states:::aws-sdk:polly: [apiAction]`	30. September 2021	Polly

Service-Name	**Task**staatliche Ressource	Datum unterstützt	Präfix für Ausnahmen
Amazon QLDB	`arn:aws:states:::aws-sdk:qldb: [apiAction]`	30. September 2021	Qldb
Amazon QLDB Session	`arn:aws:states:::aws-sdk:qldbsession: [apiAction]`	30. September 2021	QldbSession
Amazon QuickSight	`arn:aws:states:::aws-sdk:quicksight: [apiAction]`	30. September 2021	QuickSight
Amazon Redshift	`arn:aws:states:::aws-sdk:redshift: [apiAction]`	30. September 2021	Redshift
Amazon Redshift Data API	`arn:aws:states:::aws-sdk:redshiftdata: [apiAction]`	30. September 2021	RedshiftData
Amazon Rekognition	`arn:aws:states:::aws-sdk:rekognition: [apiAction]`	30. September 2021	Rekognition
Amazon Relational Database Service	`arn:aws:states:::aws-sdk:rds: [apiAction]`	30. September 2021	Rds
Amazon Route 53	`arn:aws:states:::aws-sdk:route53: [apiAction]`	30. September 2021	Route53

Service-Name	**Task**staatliche Ressource	Datum unterstützt	Präfix für Ausnahmen
Amazon Route 53 Recovery Control Config	`arn:aws:states:::aws-sdk:route53recoverycontrolconfig:` *`[apiAction]`*	30. September 2021	Route53RecoveryControlConfig
Amazon Route 53 Domains	`arn:aws:states:::aws-sdk:route53domains:` *`[apiAction]`*	30. September 2021	Route53Domains
Amazon Route 53 Resolver	`arn:aws:states:::aws-sdk:route53resolver:` *`[apiAction]`*	30. September 2021	Route53Resolver
Amazon S3 on Outposts	`arn:aws:states:::aws-sdk:s3outposts:` *`[apiAction]`*	30. September 2021	S3Outposts
Amazon SageMaker Runtime Feature Store Runtime	`arn:aws:states:::aws-sdk:sagemakerfeaturestoreruntime:` *`[apiAction]`*	30. September 2021	SageMakerRuntimeFeatureStoreRuntime
Amazon SageMaker Runtime Runtime	`arn:aws:states:::aws-sdk:sagemakerruntime:` *`[apiAction]`*	30. September 2021	SageMakerRuntimeRuntime
Amazon SageMaker	`arn:aws:states:::aws-sdk:sagemaker:` *`[apiAction]`*	30. September 2021	SageMakerRuntime
Amazon SageMaker Edge Manager	`arn:aws:states:::aws-sdk:sagemakeredge:` *`[apiAction]`*	30. September 2021	SagemakerEdge

Service-Name	**Task**staatliche Ressource	Datum unterstützt	Präfix für Ausnahmen
Amazon Simple Email Service	`arn:aws:states:::aws-sdk:ses:` *`[apiAction]`*	30. September 2021	Ses
Amazon Simple Email Service V2	`arn:aws:states:::aws-sdk:sesv2:` *`[apiAction]`*	30. September 2021	SesV2
Amazon Simple Notification Service	`arn:aws:states:::aws-sdk:sns:` *`[apiAction]`*	30. September 2021	Sns
Amazon Simple Queue Service	`arn:aws:states:::aws-sdk:sqs:` *`[apiAction]`*	30. September 2021	Sqs
Amazon Simple Storage Service	`arn:aws:states:::aws-sdk:s3:` *`[apiAction]`* ***	30. September 2021	S3
Amazon Simple Workflow Service	`arn:aws:states:::aws-sdk:swf:` *`[apiAction]`*	30. September 2021	Swf
Amazon Textract	`arn:aws:states:::aws-sdk:textract:` *`[apiAction]`*	30. September 2021	Textract
Amazon Transcribe	`arn:aws:states:::aws-sdk:transcribe:` *`[apiAction]`*	30. September 2021	Transcribe
Amazon Translate	`arn:aws:states:::aws-sdk:translate:` *`[apiAction]`*	30. September 2021	Translate

Service-Name	**Task**staatliche Ressource	Datum unterstützt	Präfix für Ausnahmen
Amazon WorkDocs	`arn:aws:states:::aws-sdk:workdocs:` *`[apiAction]`*	30. September 2021	WorkDocs
Amazon WorkMail	`arn:aws:states:::aws-sdk:workmail:` *`[apiAction]`*	30. September 2021	WorkMail
Amazon WorkMail Message Flow	`arn:aws:states:::aws-sdk:workmailmessageflow:` *`[apiAction]`*	30. September 2021	WorkMailMessageFlow
Amazon WorkSpaces	`arn:aws:states:::aws-sdk:workspaces:` *`[apiAction]`*	30. September 2021	WorkSpaces
Amazon WorkSpaces Web	`arn:aws:states:::aws-sdk:workspacesweb:` *`[apiAction]`*	19. April 2022	WorkSpacesWeb
Amplify	`arn:aws:states:::aws-sdk:amplifybackend:` *`[apiAction]`*	30. September 2021	AmplifyBackend
Amplify UI Builder	`arn:aws:states:::aws-sdk:amplifyuibuilder:` *`[apiAction]`*	19. April 2022	AmplifyUiBuilder
Application Auto Scaling	`arn:aws:states:::aws-sdk:applicationautoscaling:` *`[apiAction]`*	30. September 2021	ApplicationAutoScaling

Service-Name	**Task**staatliche Ressource	Datum unterstützt	Präfix für Ausnahmen
Amazon EC2 Auto Scaling	`arn:aws:states:::aws-sdk:autoscaling: [apiAction]`	30. September 2021	Auto Scaling
CodeArtifact	`arn:aws:states:::aws-sdk:codeartifact: [apiAction]`	30. September 2021	Codeartifact
DynamoDB Accelerator	`arn:aws:states:::aws-sdk:dax: [apiAction]`	30. September 2021	Dax
EC2 Image Builder	`arn:aws:states:::aws-sdk:imagebuilder: [apiAction]`	30. September 2021	Imagebuilder
AWS Elastic Disaster Recovery	`arn:aws:states:::aws-sdk:drs: [apiAction]`	19. April 2022	Drs
Elastic Load Balancing	`arn:aws:states:::aws-sdk:elasticloadbalancing: [apiAction]`	30. September 2021	ElasticLoadBalancing
Elastic Load Balancing V2	`arn:aws:states:::aws-sdk:elasticloadbalancingv2: [apiAction]`	30. September 2021	ElasticLoadBalancingV2
MediaConnect	`arn:aws:states:::aws-sdk:mediaconnect: [apiAction]`	30. September 2021	MediaConnect

Service-Name	**Task**staatliche Ressource	Datum unterstützt	Präfix für Ausnahmen
Amazon S3 Control	`arn:aws:states:::aws-sdk:s3control:` *`[apiAction]`* ***	30. September 2021	S3Control
Recycle Bin for Amazon EBS	`arn:aws:states:::aws-sdk:rbin:` *`[apiAction]`*	19. April 2022	Rbin
Savings Plans	`arn:aws:states:::aws-sdk:savingsplans:` *`[apiAction]`*	30. September 2021	Savingsplans
Amazon EventBridge Schema Registry	`arn:aws:states:::aws-sdk:schemas:` *`[apiAction]`*	30. September 2021	Schemas
Service Quotas	`arn:aws:states:::aws-sdk:servicequotas:` *`[apiAction]`*	30. September 2021	ServiceQuotas
AWS Snowball	`arn:aws:states:::aws-sdk:snowball:` *`[apiAction]`*	30. September 2021	Snowball

Nicht unterstützte API-Aktionen für unterstützte Dienste

In der folgenden Tabelle sind die nicht unterstützten API-Aktionen für AWS SDK-Dienstintegrationen aufgeführt. Die rechte Spalte enthält die API-Aktionen, die derzeit für den in der linken Spalte aufgeführten Dienst nicht unterstützt werden.

Service-Name	Nicht unterstützte API-Aktion
AWS Application Discovery Service	• `DescribeExportConfigurations`

Service-Name	Nicht unterstützte API-Aktion
	• `ExportConfigurations`
Amazon Bedrock	• `InvokeModelWithResponseStream`
Agenten für Amazon Bedrock Runtime	• `InvokeAgent`
AWS CodeDeploy	• `BatchGetDeploymentInstances` • `GetDeploymentInstance` • `ListDeploymentInstances` • `SkipWaitTimeForInstanceTermination`
Amazon Comprehend Medical	• `DetectEntities`
AWS Direct Connect	• `AllocateConnectionOnInterconnect` • `DescribeConnectionLoa` • `DescribeConnectionsOnInterconnect` • `DescribeInterconnectLoa`
Amazon Elastic File System	• `CreateTags`
Amazon Elastic Transcoder	• `TestRole`
Amazon EMR	• `DescribeJobFlows`
AWS IoT	• `AttachPrincipalPolicy` • `ListPrincipalPolicies` • `DetachPrincipalPolicy` • `ListPolicyPrincipals` • `DetachPrincipalPolicy`
AWS IoT Wichtiger Geräteberater	• `ListTestCases`
Amazon Kinesis	• `SubscribeToShard`

Service-Name	Nicht unterstützte API-Aktion
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 S3-Steuerung	• `SelectObjectContent`
AWS Shield	• `DeleteSubscription`
AWS Security Token Service	• `AssumeRole` • `AssumeRoleWithSAML` • `AssumeRoleWithWebIdentity`

Veraltete AWS SDK-Serviceintegrationen

Die folgenden AWS SDK-Dienstintegrationen sind jetzt veraltet:

- AWS Mobil
- Amazon Macie
- AWS IoT RoboRunner

Optimierte Integrationen für Step Functions

Die folgenden Themen umfassen die unterstützten APIs, Parameter und die Anforderungs-/Antwortsyntax in der Sprache der Amazon-Staaten für die Koordination anderer Dienste. AWS Die Themen enthalten auch Beispielcode. Sie können Optimized Integrations Services direkt aus der Sprache der Amazon States im `Resource` Feld eines `Task` Bundesstaates aufrufen.

Sie können drei Service-Integrationsmuster verwenden:

- Antwort anfordern (Standard) — Warten Sie auf die HTTP-Antwort und wechseln Sie dann zum nächsten Bundesstaat
- Job ausführen (`.sync`) — warte, bis der Job abgeschlossen ist
- Warte auf Callback (`.waitForTaskToken`) — unterbricht einen Workflow, bis ein Task-Token zurückgegeben wird

Standard-Workflows und Express-Workflows unterstützen dieselben Integrationen, aber nicht dieselben Integrationsmuster.

- Die Unterstützung von optimierten Integrationsmustern ist für jede Integration unterschiedlich.
- Express-Workflows unterstützen Run a Job (.sync) oder Wait for Callback () nicht. waitForTaskToken).
- Weitere Informationen finden Sie unter Standard- und Express-Workflows.

Standard Workflows

Unterstützte Serviceintegrationen

	Service	Request Response (Antwort anfordern)	Run a Job (Auftrag ausführen) (**`.sync`**)	Wait for Callback (Auf Rückruf warten) (**`.waitForTaskToken`**)
Optimierte Integrationen	Amazon API Gateway	✓		✓
	Amazon Athena	✓	✓	
	AWS Batch	✓	✓	
	Amazon Bedrock	✓	✓	✓
	AWS CodeBuild	✓	✓	
	Amazon DynamoDB	✓		

	Service	Request Response (Antwort anfordern)	Run a Job (Auftrag ausführen) (`.sync`)	Wait for Callback (Auf Rückruf warten) (`.waitForTaskToken`)
	Amazon ECS/Fargate	✓	✓	✓
	Amazon EKS	✓	✓	✓
	Amazon EMR	✓	✓	
	Amazon EMR on EKS	✓	✓	
	Amazon EMR Serverless	✓	✓	
	Amazon EventBridge	✓		✓
	AWS Glue	✓	✓	
	AWS Glue DataBrew	✓	✓	
	AWS Lambda	✓		✓
	AWS Elemental MediaConvert	✓	✓	
	Amazon SageMaker	✓	✓	
	Amazon SNS	✓		✓
	Amazon SQS	✓		✓
	AWS Step Functions	✓	✓	✓
AWS SDK-Integrationen	Über zweihundert	✓		✓

Express Workflows

Unterstützte Serviceintegrationen

	Service	Request Response (Antwort anfordern)	Run a Job (Auftrag ausführen) (`.sync`)	Wait for Callback (Auf Rückruf warten) (`.waitForTaskToken`)
Optimierte Integrationen	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	✓		

	Service	Request Response (Antwort anfordern)	Run a Job (Auftrag ausführen) (`.sync`)	Wait for Callback (Auf Rückruf warten) (`.waitForTaskToken`)
	AWS Elemental MediaConvert	✓		
	Amazon SageMaker	✓		
	Amazon SNS	✓		
	Amazon SQS	✓		
	AWS Step Functions	✓		
AWS SDK-Integrationen	Über zweihundert	✓		

API Gateway mit Step-Funktionen aufrufen

Step Functions kann bestimmte AWS Dienste direkt von Amazon States Language (ASL) aus steuern. Weitere Informationen hierzu finden Sie unter Arbeiten mit anderen -Services und Parameter an eine Service-API übergeben.

Wie sich die optimierte API Gateway-Integration von der API Gateway AWS SDK-Integration unterscheidet

- `apigateway:invoke:`hat kein Äquivalent in der AWS SDK-Dienstintegration. Stattdessen ruft der Optimized API Gateway Gateway-Dienst Ihren API Gateway Gateway-Endpunkt direkt auf.

Sie verwenden Amazon API Gateway, um HTTP- und REST-APIs zu erstellen, zu veröffentlichen, zu verwalten und zu überwachen. Für die Integration mit API Gateway definieren Sie in Step Functions einen `Task` Status, der direkt einen API-Gateway-HTTP- oder API-Gateway-REST-Endpunkt aufruft, ohne Code zu schreiben oder sich auf eine andere Infrastruktur zu verlassen. Eine `Task` Statusdefinition enthält alle erforderlichen Informationen für den API-Aufruf. Sie können auch verschiedene Autorisierungsmethoden auswählen.

Unterstützung von API-Gateway-Funktionen

Die Step Functions API Gateway Gateway-Integration unterstützt einige, aber nicht alle API Gateway Gateway-Funktionen. Eine detailliertere Liste der unterstützten Funktionen finden Sie im Folgenden.

- Wird sowohl von den Step Functions API Gateway Gateway-REST-API- als auch von den API Gateway Gateway-HTTP-API-Integrationen unterstützt:
  - Autorisierer: IAM (mit Signature Version 4), No Auth, Lambda Authorizers (auf Anforderungsparametern und Token-basiert mit benutzerdefiniertem Header)
  - API-Typen: Regional
  - API-Verwaltung: API-Gateway-API-Domänennamen, API-Phase, Pfad, Abfrageparameter, Anfragetext
- Unterstützt von der Step Functions API Gateway HTTP-API-Integration. Die STEP Functions API Gateway REST-API-Integration, die die Option für Edge-optimierte APIs bietet, wird nicht unterstützt.
- Wird von der Step Functions API Gateway Gateway-Integration nicht unterstützt:
  - Autorisierer: Amazon Cognito, Native Open ID Connect/OAuth 2.0, Autorisierungsheader für tokenbasierte Lambda-Autorisierer
  - API-Typen: Privat
  - API-Verwaltung: Benutzerdefinierte Domainnamen

Weitere Informationen zu API Gateway und seinen HTTP- und REST-APIs finden Sie im Folgenden.

- Die Amazon API Gateway Gateway-Konzeptseite.
- Auswahl zwischen HTTP-APIs und REST-APIs im API Gateway Gateway-Entwicklerhandbuch.

Anforderungsformat

Wenn Sie Ihre `Task` Statusdefinition erstellen, validiert Step Functions die Parameter, erstellt die für den Aufruf erforderliche URL und ruft dann die API auf. Die Antwort umfasst den HTTP-Statuscode, die Header und den Antworttext. Das Anforderungsformat hat sowohl erforderliche als auch optionale Parameter.

Erforderliche Anforderungsparameter

- `ApiEndpoint`
  - Typ: `String`
  - Der Hostname einer API-Gateway-URL. Das Format ist *`<API ID>`*`.execute-api.`*`<region>`*`.amazonaws.com`.

    Die API-ID darf nur eine Kombination der folgenden alphanumerischen Zeichen enthalten: `0123456789abcdefghijklmnopqrstuvwxyz`
- `Method`
  - Typ: `Enum`
  - Die HTTP-Methode, die eine der folgenden sein muss:
    - `GET`
    - `POST`
    - `PUT`
    - `DELETE`
    - `PATCH`
    - `HEAD`
    - `OPTIONS`

Optionale Anforderungsparameter

- `Headers`
  - Typ: `JSON`
  - HTTP-Header ermöglichen eine Liste von Werten, die demselben Schlüssel zugeordnet sind.
- `Stage`
  - Typ: `String`

  - Der Name der Phase, in der die API in API Gateway bereitgestellt wird. Es ist optional für jede HTTP-API, die den `$default` Stage verwendet.
- `Path`
  - Typ: `String`
  - Pfadparameter, die nach dem API-Endpunkt angehängt werden.
- `QueryParameters`
  - Typ: `JSON`
  - Abfragezeichenfolgen erlauben nur eine Liste von Werten, die demselben Schlüssel zugeordnet sind.
- `RequestBody`
  - Typ: `JSON` oder `String`.
  - Der Hauptteil der HTTP-Anfrage. Sein Typ kann entweder ein `JSON` Objekt oder sein`String`. `RequestBody`wird nur für `PATCHPOST`, und `PUT` HTTP-Methoden unterstützt.
- `AllowNullValues`
  - Typ: `BOOLEAN` — Standardwert: `false`
  - Mit der Standardeinstellung werden Nullwerte im Status der Anforderungseingabe nicht an Ihre API gesendet. Im folgenden Beispiel wird das `category` Feld nicht in die Anfrage aufgenommen, es sei denn, es `AllowNullValues` ist `true` in Ihrer State-Machine-Definition auf festgelegt.

```
{
    "NewPet": {
        "type": "turtle",
        "price": 123,
        "category": null
    }
}
```

> **Note**
>
> Standardmäßig werden Felder mit Nullwerten im Anforderungseingabestatus nicht an Ihre API gesendet. Sie können das Senden von Nullwerten an Ihre API erzwingen, indem Sie `true` in Ihrer State-Machine-Definition die Einstellung `AllowNullValues` auf setzen.

- `AuthType`
  - Typ: `JSON`
  - Die Authentifizierungsmethode. Die Standardmethode ist`NO_AUTH`. Die zulässigen Werte lauten:
    - `NO_AUTH`
    - `IAM_ROLE`
    - `RESOURCE_POLICY`

    Weitere Informationen finden Sie unter Authentifizierung und Autorisierung.

> **Note**
>
> Aus Sicherheitsgründen sind die folgenden HTTP-Header-Schlüssel derzeit nicht zulässig:
>
> - Alles, dem ein`X-Forwarded`, `X-Amz` oder `X-Amzn` vorangestellt ist.
> - `Authorization`
> - `Connection`
> - `Content-md5`
> - `Expect`
> - `Host`
> - `Max-Forwards`
> - `Proxy-Authenticate`
> - `Server`
> - `TE`
> - `Transfer-Encoding`
> - `Trailer`
> - `Upgrade`
> - `Via`
> - `Www-Authenticate`

Das folgende Codebeispiel zeigt, wie API Gateway mithilfe von Step Functions aufgerufen wird.


```
{
```

```
    "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"
    }
}
```

Authentifizierung und Autorisierung

Sie können die folgenden Authentifizierungsmethoden verwenden:

- Keine Autorisierung: Rufen Sie die API direkt ohne Autorisierungsmethode auf.
- IAM-Rolle: Bei dieser Methode übernimmt Step Functions die Rolle der Zustandsmaschine, signiert die Anfrage mit Signature Version 4 (Sigv4) und ruft dann die API auf.
- Ressourcenrichtlinie: Step Functions authentifiziert die Anfrage und ruft dann die API auf. Sie müssen der API eine Ressourcenrichtlinie anhängen, die Folgendes spezifiziert:
  1. Die Zustandsmaschine, die API Gateway aufruft.

     > **Important**
     >
     > Sie müssen Ihren Zustandsmaschine angeben, um den Zugriff darauf zu beschränken. Wenn Sie dies nicht tun, wird jeder Zustandsmaschine, die ihre API-Gateway-Anfrage mit der Ressourcenrichtlinien-Authentifizierung für Ihre API authentifiziert, Zugriff gewährt.

  2. That Step Functions ist der Dienst, der API Gateway aufruft:`"Service": "states.amazonaws.com"`.
  3. Die Ressource, auf die Sie zugreifen möchten, einschließlich:

- Die *Region*.
- Die *Konto-ID* in der angegebenen Region.
- *Die API-ID.*
- *Der Bühnenname.*
- Das *HTTP-VERB* (Methode).
- *Der Ressourcenpfadspezifizierer.*

Ein Beispiel für eine Ressourcenrichtlinie finden Sie unter IAM-Richtlinien für Step Functions und API Gateway.

Weitere Informationen zum Ressourcenformat finden Sie unter Ressourcenformat der Berechtigungen für die Ausführung von APIs in API Gateway im API Gateway Developer Guide.

> **Note**
>
> Ressourcenrichtlinien werden nur für die REST-API unterstützt.

Muster der Serviceintegration

Die API Gateway Gateway-Integration unterstützt zwei Dienstintegrationsmuster:

- Request Response (Antwort anfordern), was das Standard-Integrationsmuster ist. Dadurch kann Step Functions unmittelbar nach Erhalt einer HTTP-Antwort mit dem nächsten Schritt fortfahren.
- Warten auf einen Callback mit dem Aufgabentoken(`.waitForTaskToken`), wodurch gewartet wird, bis ein Task-Token mit einer Nutzlast zurückgegeben wird. Um das `.waitForTaskToken` Muster zu verwenden, fügen Sie .wait ForTaskToken an das Ende des Felds Resource Ihrer Aufgabendefinition an, wie im folgenden Beispiel gezeigt:

```
{
    "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"
    }
}
```

Ausgabeformat

Die folgenden Ausgabeparameter sind verfügbar:

Name	Typ	Beschreibung
`ResponseBody`	`JSON` oder `String`	Der Antworttext des API-Aufrufs.
`Headers`	`JSON`	Die Antwort-Header.
`StatusCode`	`Integer`	Den HTTP-Statuscode der Antwort.
`StatusText`	`String`	Der Statustext der Antwort.

Ein Beispiel für eine Antwort:

```
{
    "ResponseBody": {
        "myBills": []
    },
    "Headers": {
        "key": ["value1", "value2"]
    },
    "StatusCode": 200,
    "StatusText": "OK"
}
```

Fehlerbehandlung

Wenn ein Fehler auftritt, `cause` wird ein `error` und wie folgt zurückgegeben:

- Wenn der HTTP-Statuscode verfügbar ist, wird der Fehler im folgenden Format zurückgegebenApiGateway.*<HTTP Status Code>*.
- Wenn der HTTP-Statuscode nicht verfügbar ist, wird der Fehler im Format zurückgegebenApiGateway.*<Exception>*.

In beiden Fällen `cause` wird das als Zeichenfolge zurückgegeben.

Das folgende Beispiel zeigt eine Antwort, bei der ein Fehler aufgetreten ist:

```
{
    "error": "ApiGateway.403",
    "cause": "{\"message\":\"Missing Authentication Token\"}"
}
```

> **Note**
>
> Der Statuscode 2XX bedeutet Erfolg, und es wird kein Fehler zurückgegeben. Alle anderen Statuscodes oder ausgelösten Ausnahmen führen zu einem Fehler.

Weitere Informationen finden Sie unter:

- Die Konzepte von Amazon API Gateway im API Gateway Developer Guide.
- IAM-Richtlinien für Amazon API Gateway
- Ein Beispielprojekt, das zeigt, wie Rufen Sie API Gateway auf

Die Konzepte von Amazon API Gateway im API Gateway Developer Guide.

Rufen Sie Athena mit Step Functions auf

Step Functions kann bestimmte AWS Dienste direkt von Amazon States Language (ASL) aus steuern. Weitere Informationen hierzu finden Sie unter Arbeiten mit anderen -Services und Parameter an eine Service-API übergeben.

Wie sich die optimierte Athena-Integration von der Athena AWS SDK-Integration unterscheidet

- Das Ausführen einer Aufgabe (.sync) Integrationsmuster wird unterstützt.
- Es gibt keine Optimierungen für das Request Response (Antwort anfordern) Integrationsmuster.
- Das Warten auf einen Callback mit dem Aufgabentoken Integrationsmuster wird nicht unterstützt.

Die AWS Step Functions Serviceintegration mit Amazon Athena ermöglicht es Ihnen, Step Functions zu verwenden, um die Abfrageausführung zu starten und zu beenden und Abfrageergebnisse abzurufen. Mithilfe von Step Functions können Sie Ad-hoc-Datenabfragen oder geplante Datenabfragen ausführen und Ergebnisse abrufen, die auf Ihre S3-Datenseen abzielen. Athena ist Serverless, weshalb auch keine Infrastruktur eingerichtet oder verwaltet werden muss – und Sie zahlen nur für tatsächlich ausgeführte Abfragen.

Für die Integration AWS Step Functions mit Amazon Athena verwenden Sie die bereitgestellten Athena-Serviceintegrations-APIs.

Die Service-Integration-APIs sind dieselben wie die entsprechenden Athena-APIs. Nicht alle APIs unterstützen alle Integrationsmuster, wie in der folgenden Tabelle dargestellt wird:

API	Request Response (Antwort anfordern)	Ausführen einer Aufgabe (.sync)
`StartQueryExecution`	✓	✓
`StopQueryExecution`	✓	
`GetQueryExecution`	✓	
`GetQueryResults`	✓	

Unterstützte Amazon Athena Athena-APIs:

> **Note**
>
> In Step Functions gibt es ein Kontingent für die maximale Eingabe- oder Ergebnisdatengröße für eine Aufgabe. Dadurch sind Sie auf 256 KB an Daten als UTF-8-kodierte Zeichenfolge beschränkt, wenn Sie Daten an einen anderen Dienst senden oder von einem anderen Dienst empfangen. Siehe Kontingente im Zusammenhang mit der Ausführung von Zustandsmaschinen.

- `StartQueryExecution`
  - Erforderliche Syntax
  - Unterstützte Parameter:
    - `ClientRequestToken`
    - `ExecutionParameters`
    - `QueryExecutionContext`
    - `QueryString`
    - `ResultConfiguration`
    - `WorkGroup`
  - `Response syntax`
- `StopQueryExecution`
  - Erforderliche Syntax
  - Unterstützte Parameter:
    - `QueryExecutionId`
- `GetQueryExecution`
  - Erforderliche Syntax
  - Unterstützte Parameter:
    - `QueryExecutionId`
  - `Response syntax`
- `GetQueryResults`
  - Erforderliche Syntax
  - Unterstützte Parameter:
    - `MaxResults`

    - NextToken
    - QueryExecutionId
- Response syntax

Das Folgende beinhaltet einen Task-Status, der eine Athena-Abfrage startet.

```
"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"
}
```

Informationen zur Konfiguration von IAM Berechtigungen bei der Verwendung Step Functions mit anderen AWS Diensten finden Sie unterIAM-Richtlinien für integrierte Dienste.

AWS Batch Mit Step Functions verwalten

Step Functions kann bestimmte AWS Dienste direkt von Amazon States Language (ASL) aus steuern. Weitere Informationen hierzu finden Sie unter Arbeiten mit anderen -Services und Parameter an eine Service-API übergeben.

> Wie unterscheidet sich die optimierte AWS Batch Integration von der AWS BatchAWS SDK-Integration
>
> - Das Ausführen einer Aufgabe (.sync) Integrationsmuster ist verfügbar.
>
> Beachten Sie, dass es keine Optimierungen für die Request Response (Antwort anfordern) Warten auf einen Callback mit dem Aufgabentoken Integrationsmuster gibt.

Unterstützte AWS Batch APIs:

- SubmitJob
  - Erforderliche Syntax
  - Unterstützte Parameter:
    - ArrayProperties
    - ContainerOverrides
    - DependsOn
    - JobDefinition
    - JobName
    - JobQueue
    - Parameters
    - RetryStrategy
    - Timeout
    - Tags
  - Antwortsyntax

> **Die Parameter in Step Functions werden ausgedrückt in PascalCase**
>
> Auch wenn sich die native Service-API in CamelCase befindet, z. B. die API-Aktion`startSyncExecution`, geben Sie Parameter in an PascalCase, z. B.:. `StateMachineArn`

Das Folgende beinhaltet einen `Task` Status, der einen AWS Batch Job einreicht und darauf wartet, dass er abgeschlossen ist.

```
{
  "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
    }
  }
}
```

Informationen zur Konfiguration von IAM Berechtigungen bei der Verwendung Step Functions mit anderen AWS Diensten finden Sie unter. IAM-Richtlinien für integrierte Dienste

Aufrufen von Amazon Bedrock mit Step Functions

Step Functions kann bestimmte AWS Dienste direkt von Amazon States Language (ASL) aus steuern. Weitere Informationen hierzu finden Sie unter Arbeiten mit anderen -Services und Parameter an eine Service-API übergeben.

Themen

- Amazon BedrockAPIs zur Serviceintegration
- Definition des Aufgabenstatus für die Amazon Bedrock Integration

Amazon BedrockAPIs zur Serviceintegration

Für die AWS Step Functions Integration Amazon Bedrock können Sie die folgenden APIs verwenden. Diese APIs ähneln den entsprechenden Amazon Bedrock APIs, weisen jedoch einige Unterschiede in den übergebenen Anforderungsfeldern auf.

Die Unterschiede zwischen den einzelnen Service-Integrations-APIs und den entsprechenden Amazon Bedrock-APIs werden in der folgenden Tabelle beschrieben.

Amazon BedrockAPIs für die Serviceintegration und entsprechende Amazon Bedrock APIs

Amazon BedrockAPI zur Serviceintegration	Entsprechende Amazon Bedrock API	Unterschiede
InvokeModel Ruft das angegebene Amazon Bedrock Modell auf, um mithilfe der Eingabe, die Sie im Hauptteil der Anfrage angeben, eine Inferenz auszuführen. Sie verwenden `InvokeModel` es, um Inferenzen für Textmodelle, Bildmodelle und Einbettungsmodelle auszuführen.	InvokeModel	Der Anfragetext der Amazon Bedrock Service-Integrationsns-API enthält die folgenden zusätzlichen Parameter. • `Body`— Spezifiziert Eingabedaten in dem Format, das im Inhaltstyp-Anforderungsheader angegeben ist. `Body`enthält spezifische Parameter für das Zielmodell. Wenn Sie die `InvokeModel` API verwenden, müssen Sie den `Body` Parameter angeben. Step Functionsvalidiert die von Ihnen eingegebene Eingabe nicht`Body`. Wenn Sie angeben, dass Sie die Amazon Bedrock optimierte Integration `Body` verwenden, können Sie eine Nutzlast von bis zu 256 KB angeben. Wenn Ihre Nutzlast 256 KB überschreitet, empfehlen wir die Verwendung. `Input` • `Input`— Gibt die Quelle an, aus der die Eingabedaten abgerufen werden sollen.

Amazon BedrockAPI zur Serviceintegration	Entsprechende Amazon Bedrock API	Unterschiede
		Dieses optionale Feld ist spezifisch für die Amazon Bedrock optimierte Integration mitStep Functions. In diesem Feld können Sie eine angeben`S3Uri`. Sie können entweder `Body` in den Parametern oder`Input`, aber nicht beide angeben. Wenn Sie `Input` ohne Angabe angeben`ContentType` , wird der Inhaltstyp der Eingabedatenquelle zum Wert für`ContentType` . • `Output`— Gibt das Ziel an, in das die API-Antwort geschrieben wird. Dieses optionale Feld ist spezifisch für die Amazon Bedrock optimierte Integration mitStep Functions. In diesem Feld können Sie eine angeben`S3Uri`. Wenn Sie dieses Feld angeben, wird der API-Antworttext durch einen Verweis auf den Amazon S3 Speicherort der ursprünglichen Ausgabe ersetzt.

Amazon BedrockAPI zur Serviceintegration	Entsprechende Amazon Bedrock API	Unterschiede
		Das folgende Beispiel zeigt die Syntax für die InvokeModel API zur Amazon Bedrock Integration. `{ "ModelId": String, // required "Accept": String, // default: application/json "ContentType": String, // default: application/json "Input": { // not from Bedrock API "S3Uri": String }, "Output": { // not from Bedrock API "S3Uri": String } }`
CreateModelCustomizationJob Erstellt einen Feinabstimmungsauftrag zur Anpassung eines Basismodells.	CreateModelCustomizationJob	None

Amazon BedrockAPI zur Serviceintegration	Entsprechende Amazon Bedrock API	Unterschiede
CreateModelCustomizationJob .sync Erstellt einen Feinabsti mmungsauftrag zur Anpassung eines Basismode lls.	CreateModelCustomizationJob	None

Informationen zur Konfiguration von IAM Berechtigungen bei der Verwendung Step Functions mit anderen AWS Diensten finden Sie unterIAM-Richtlinien für integrierte Dienste.

Definition des Aufgabenstatus für die Amazon Bedrock Integration

Die folgende Aufgabenstatusdefinition zeigt, wie Sie sich Amazon Bedrock in Ihre Zustandsmaschinen integrieren können. Dieses Beispiel zeigt einen Task-Status, der das vollständige Ergebnis eines durch den Pfad, `result_one` angegebenen Modellaufrufs extrahiert. Dies basiert auf Inferenzparametern für Fundamentmodelle. In diesem Beispiel wird das Large Language Model (LLM) von Cohere Command verwendet.

```
{
  "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**
>
> Ein Beispiel für eine Zustandsmaschine, die in Ihren Computer integriert werden kann AWS-Konto, Amazon Bedrock finden Sie unter. Führen Sie AI-Prompt-Chaining durch mit Amazon Bedrock

Rufen Sie AWS CodeBuild mit Step Functions auf

Step Functions kann bestimmte AWS Dienste direkt von Amazon States Language (ASL) aus steuern. Weitere Informationen hierzu finden Sie unter Arbeiten mit anderen -Services und Parameter an eine Service-API übergeben.

> **Wie unterscheidet sich die optimierte CodeBuild Integration von der CodeBuild AWS SDK-Integration**
>
> - Das Ausführen einer Aufgabe (.sync) Integrationsmuster wird unterstützt.
> - Nachdem Sie `StopBuild` oder aufgerufen haben`StopBuildBatch`, kann der Build oder der Build-Batch erst gelöscht werden, wenn einige interne Arbeiten abgeschlossen sind, CodeBuild um den Status des Builds oder der Builds abzuschließen. Wenn Sie versuchen, `BatchDeleteBuilds` oder `DeleteBuildBatch` während dieses Zeitraums zu verwenden, wird der Build oder der Build-Batch möglicherweise nicht gelöscht. Die optimierten Serviceintegrationen `DeleteBuildBatch` bieten einen internen Wiederholungsversuch, um den Anwendungsfall des Löschens unmittelbar nach dem Beenden zu vereinfachen. `BatchDeleteBuilds`

Die AWS Step Functions Serviceintegration mit AWS CodeBuild ermöglicht es Ihnen, Step Functions zu verwenden, um Builds auszulösen, zu stoppen und zu verwalten und Build-Berichte gemeinsam zu nutzen. Mit Step Functions können Sie Pipelines für die kontinuierliche Integration entwerfen und ausführen, um Ihre Softwareänderungen für Anwendungen zu validieren.

Nicht alle APIs unterstützen alle Integrationsmuster, wie in der folgenden Tabelle dargestellt wird:

API	Request Response (Antwort anfordern)	Ausführen einer Aufgabe (.sync)
StartBuild	✓	✓
StopBuild	✓	
BatchDeleteBuilds	✓	
BatchGetReports	✓	
StartBuildBatch	✓	✓
StopBuildBatch	✓	
RetryBuildBatch	✓	✓
DeleteBuildBatch	✓	

Parameter in Step Functions werden ausgedrückt in PascalCase

Auch wenn sich die native Service-API in CamelCase befindet, z. B. die API-Aktion`startSyncExecution`, geben Sie Parameter in an PascalCase, z. B.:. `StateMachineArn`

Unterstützte CodeBuild APIs und Syntax:

- `StartBuild`
  - Erforderliche Syntax
  - Unterstützte Parameter:
    - `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

- Antwortsyntax

- StopBuild
  - Erforderliche Syntax
  - Unterstützte Parameter:
    - Id
  - Antwortsyntax

- BatchDeleteBuilds
  - Erforderliche Syntax
  - Unterstützte Parameter:
    - Ids
  - Antwortsyntax
- BatchGetReports
  - Erforderliche Syntax
  - Unterstützte Parameter:
    - ReportArns
  - Antwortsyntax
- StartBuildBatch
  - Erforderliche Syntax
  - Unterstützte Parameter:
    - 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

- Antwortsyntax

- StopBuildBatch
  - Erforderliche Syntax
  - Unterstützte Parameter:
    - Id
  - Antwortsyntax
- RetryBuildBatch
  - Erforderliche Syntax
  - Unterstützte Parameter:
    - Id
    - IdempotencyToken
    - RetryType
  - Antwortsyntax
- DeleteBuildBatch
  - Erforderliche Syntax
  - Unterstützte Parameter:

- `Id`
- Antwortsyntax

> **Note**
>
> Sie können den `JSONPath`-`(..)`Operator für rekursiven Abstieg für `BatchDeleteBuilds` verwenden. Dies gibt ein Array zurück und ermöglicht es Ihnen, das `Arn`-Feld von `StartBuild` in einen `Ids`-Pluralparameter umzuwandeln, wie im folgenden Beispiel gezeigt.
>
> ```
> "BatchDeleteBuilds": {
>     "Type": "Task",
>     "Resource": "arn:aws:states:::codebuild:batchDeleteBuilds",
>     "Parameters": {
>         "Ids.$": "$.Build..Arn"
>     },
>     "Next": "MyNextState"
> },
> ```

Informationen zur Konfiguration von IAM Berechtigungen bei der Verwendung Step Functions mit anderen AWS Diensten finden Sie unterIAM-Richtlinien für integrierte Dienste.

Rufen Sie DynamoDB-APIs mit Step Functions auf

Step Functions kann bestimmte AWS Dienste direkt von Amazon States Language (ASL) aus steuern. Weitere Informationen hierzu finden Sie unter Arbeiten mit anderen -Services und Parameter an eine Service-API übergeben.

> **Note**
>
> In Step Functions gibt es ein Kontingent für die maximale Eingabe- oder Ergebnisdatengröße für eine Aufgabe. Dadurch sind Sie auf 256 KB an Daten als UTF-8-kodierte Zeichenfolge beschränkt, wenn Sie Daten an einen anderen Dienst senden oder von einem anderen Dienst empfangen. Siehe Kontingente im Zusammenhang mit der Ausführung von Zustandsmaschinen.

Wie sich die optimierte DynamoDB-Integration von der AWS DynamoDB-SDK-Integration unterscheidet

- Es gibt keine Optimierung für das Integrationsmuster. Request Response (Antwort anfordern)
- Das Warten auf einen Callback mit dem Aufgabentoken Integrationsmuster wird nicht unterstützt.
- Nur `DeleteItem`API-Aktionen `GetItemPutItem`, `UpdateItem`, und sind im Rahmen der optimierten Integration verfügbar. Andere API-Aktionen, z. B. `CreateTable`sind über die AWS DynamoDB-SDK-Integration verfügbar.

Unterstützte Amazon DynamoDB DynamoDB-APIs und -Syntax:

- `GetItem`
  - Erforderliche Syntax
  - Unterstützte Parameter:
    - `Key`
    - `TableName`
    - `AttributesToGet`
    - `ConsistentRead`
    - `ExpressionAttributeNames`
    - `ProjectionExpression`
    - `ReturnConsumedCapacity`
  - Antwortsyntax
- `PutItem`
  - Erforderliche Syntax
  - Unterstützte Parameter:
    - `Item`
    - `TableName`
    - `ConditionalOperator`
    - `ConditionExpression`

    - Expected
    - ExpressionAttributeNames
    - ExpressionAttributeValues
    - ReturnConsumedCapacity
    - ReturnItemCollectionMetrics
    - ReturnValues
  - Antwortsyntax
- DeleteItem
  - Erforderliche Syntax
  - Unterstützte Parameter:
    - Key
    - TableName
    - ConditionalOperator
    - ConditionExpression
    - Expected
    - ExpressionAttributeNames
    - ExpressionAttributeValues
    - ReturnConsumedCapacity
    - ReturnItemCollectionMetrics
    - ReturnValues
  - Antwortsyntax
- UpdateItem
  - Erforderliche Syntax
  - Unterstützte Parameter:
    - Key
    - TableName
    - AttributeUpdates
    - ConditionalOperator
    - ConditionExpression
    - Expected

- `ExpressionAttributeNames`
- `ExpressionAttributeValues`
- `ReturnConsumedCapacity`
- `ReturnItemCollectionMetrics`
- `ReturnValues`
- `UpdateExpression`

- Antwortsyntax

> **Parameter in Step Functions werden ausgedrückt in PascalCase**
>
> Auch wenn sich die native Service-API in CamelCase befindet, z. B. die API-Aktion`startSyncExecution`, geben Sie Parameter in an PascalCase, z. B.:. `StateMachineArn`

Der folgende `Task` Status ruft eine Nachricht von DynamoDB ab.

```
"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"
},
```

In einem funktionierenden Beispiel sehen Sie diesen Status im Beispielprojekt Datensätze übertragen (Lambda,DynamoDB,Amazon SQS).

Informationen zur Konfiguration von IAM Berechtigungen bei der Verwendung Step Functions mit anderen AWS Diensten finden Sie unter. IAM-Richtlinien für integrierte Dienste

Amazon ECS- oder Fargate-Aufgaben mit Step Functions verwalten

Step Functions kann bestimmte AWS Dienste direkt von Amazon States Language (ASL) aus steuern. Weitere Informationen hierzu finden Sie unter Arbeiten mit anderen -Services und Parameter an eine Service-API übergeben.

> Wie sich die optimierte Amazon ECS/Fargate-Integration von der Amazon ECS- oder Fargate SDK-Integration unterscheidet AWS
>
> - Das Ausführen einer Aufgabe (.sync) Integrationsmuster wird unterstützt.
> - `ecs:runTask`kann eine HTTP 200-Antwort zurückgeben, hat aber ein nicht leeres `Failures` Feld wie folgt:
>   - Antwort anfordern: Geben Sie die Antwort zurück und scheitern Sie nicht an der Aufgabe. Das ist dasselbe wie keine Optimierung.
>   - Einen Job- oder Task-Token ausführen: Wenn ein nicht leeres `Failures` Feld gefunden wird, schlägt die Aufgabe fehl und es wird ein `AmazonECS.Unknown` Fehler angezeigt.

Unterstützte Amazon ECS/Fargate-APIs und -Syntax:

> Die Parameter in werden ausgedrückt in Step Functions PascalCase
>
> Auch wenn sich die native Service-API in CamelCase befindet, z. B. die API-Aktion`startSyncExecution`, geben Sie Parameter in an PascalCase, z. B.:. `StateMachineArn`

- `RunTask` startet eine neue Aufgabe mit der angegebenen Aufgabendefinition.
  - Erforderliche Syntax
  - Unterstützte Parameter:
    - `Cluster`
    - `Group`
    - `LaunchType`
    - `NetworkConfiguration`
    - `Overrides`

- `PlacementConstraints`
- `PlacementStrategy`
- `PlatformVersion`
- `PropagateTags`
- `TaskDefinition`
- `EnableExecuteCommand`

- Antwortsyntax

Daten an eine Amazon ECS-Aufgabe übergeben

Step Functions kann bestimmte AWS Dienste direkt von Amazon States Language (ASL) aus steuern. Weitere Informationen hierzu finden Sie unter Arbeiten mit anderen -Services und Parameter an eine Service-API übergeben.

Sie können `overrides` damit den Standardbefehl für einen Container überschreiben und Eingaben an Ihre Amazon ECS-Aufgaben übergeben. Siehe `ContainerOverride`. In diesem Beispiel haben wir JsonPath Werte `Task` von der Eingabe an den `Task` Status übergeben.

Im Folgenden wird ein `Task` Status beschrieben, der eine Amazon ECS-Aufgabe ausführt und darauf wartet, dass sie abgeschlossen ist.

```
{
 "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
    }
  }
}
```

Die `"Command.$": "$.commands"`-Zeile in `ContainerOverrides` gibt die Befehle von der Zustandseingabe an den Container weiter.

Für das vorherige Beispiel wird jeder der Befehle als Container-Überschreibung übergeben, wenn die Eingabe für die Ausführung wie folgt lautet.

```
{
  "commands": [
    "test command 1",
    "test command 2",
    "test command 3"
  ]
}
```

Das Folgende beinhaltet einen `Task` Status, der eine Amazon ECS-Aufgabe ausführt und dann darauf wartet, dass das Task-Token zurückgegeben wird. Siehe Warten auf einen Callback mit dem Aufgabentoken.

```
{
   "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
  }
 }
}
```

Informationen zur Konfiguration von IAM Berechtigungen bei der Verwendung Step Functions mit anderen AWS Diensten finden Sie unterIAM-Richtlinien für integrierte Dienste.

Rufen Sie Amazon EKS mit Step Functions auf

Step Functions kann bestimmte AWS Dienste direkt von Amazon States Language (ASL) aus steuern. Weitere Informationen hierzu finden Sie unter Arbeiten mit anderen -Services und Parameter an eine Service-API übergeben.

> Wie sich die optimierte Amazon EKS-Integration von der Amazon EKS AWS SDK-Integration unterscheidet
>
> - Das Ausführen einer Aufgabe (.sync) Integrationsmuster wird unterstützt.
> - Es gibt keine Optimierungen für das Request Response (Antwort anfordern) Integrationsmuster.
> - Das Warten auf einen Callback mit dem Aufgabentoken Integrationsmuster wird nicht unterstützt.

Informationen zur Konfiguration von IAM Berechtigungen bei der Verwendung Step Functions mit anderen AWS Diensten finden Sie unterIAM-Richtlinien für integrierte Dienste.

Step Functions bietet zwei Arten von Service-Integrations-APIs für die Integration mit Amazon Elastic Kubernetes Service. Mit einem können Sie die Amazon EKS-APIs verwenden, um einen Amazon EKS-Cluster zu erstellen und zu verwalten. Mit der anderen können Sie mithilfe der Kubernetes-API mit Ihrem Cluster interagieren und Jobs als Teil des Workflows Ihrer Anwendung ausführen. Sie können die Kubernetes-API-Integrationen mit Amazon EKS-Clustern verwenden, die mit Step Functions erstellt wurden, mit Amazon EKS-Clustern, die mit dem eksctl-Tool oder der Amazon

EKS-Konsole erstellt wurden, oder mit ähnlichen Methoden. Weitere Informationen finden Sie unter Erstellen eines Amazon EKS-Clusters im Amazon EKS-Benutzerhandbuch.

> **Note**
>
> Die Step Functions EKS-Integration unterstützt nur Kubernetes-APIs mit öffentlichem Endpunktzugriff. Standardmäßig haben API-Serverendpunkte für EKS-Cluster öffentlichen Zugriff. Weitere Informationen finden Sie unter Amazon EKS-Cluster-Endpunktzugriffskontrolle im Amazon EKS-Benutzerhandbuch.

Step Functions beendet einen Amazon EKS-Cluster nicht automatisch, wenn die Ausführung gestoppt wird. Wenn Ihr State Machine stoppt, bevor Ihr Amazon EKS-Cluster beendet wurde, läuft Ihr Cluster möglicherweise auf unbestimmte Zeit weiter und es können zusätzliche Gebühren anfallen. Um dies zu vermeiden, stellen Sie sicher, dass jeder Amazon EKS-Cluster, den Sie erstellen, ordnungsgemäß beendet wird. Weitere Informationen finden Sie hier:

- Löschen eines Clusters im Amazon EKS-Benutzerhandbuch.
- Ausführen einer Aufgabe (.sync)in Muster der Serviceintegration.

> **Note**
>
> In Step Functions gibt es ein Kontingent für die maximale Eingabe- oder Ergebnisdatengröße für eine Aufgabe. Dadurch sind Sie auf 256 KB an Daten als UTF-8-kodierte Zeichenfolge beschränkt, wenn Sie Daten an einen anderen Dienst senden oder von einem anderen Dienst empfangen. Siehe Kontingente im Zusammenhang mit der Ausführung von Zustandsmaschinen.

Kubernetes-API-Integrationen

Step Functions unterstützt die folgenden Kubernetes-APIs:

RunJob

Die `eks:runJob` Service-Integration ermöglicht es Ihnen, einen Job auf Ihrem Amazon EKS-Cluster auszuführen. Bei `eks:runJob.sync` dieser Variante können Sie warten, bis der Job abgeschlossen ist, und optional Protokolle abrufen.

Ihr Kubernetes-API-Server muss der von Ihrem Zustandsmaschine verwendeten IAM-Rolle Berechtigungen gewähren. Weitere Informationen finden Sie unter Berechtigungen.

Für das Run a Job (`.sync`) -Muster wird der Status des Jobs durch Abfragen bestimmt. Step Functions fragt zunächst mit einer Geschwindigkeit von ungefähr 1 Umfrage pro Minute ab. Diese Rate verlangsamt sich schließlich auf etwa 1 Umfrage alle 5 Minuten. Wenn Sie häufigere Abfragen benötigen oder mehr Kontrolle über die Abfragestrategie benötigen, können Sie die `eks:call` Integration verwenden, um den Status des Jobs abzufragen.

Die `eks:runJob` Integration ist spezifisch für `batch/v1` Kubernetes Jobs. Weitere Informationen finden Sie unter Jobs in der Kubernetes-Dokumentation. Wenn Sie andere Kubernetes-Ressourcen, einschließlich benutzerdefinierter Ressourcen, verwalten möchten, verwenden Sie die Serviceintegration. `eks:call` Sie können Step Functions verwenden, um Abfrageschleifen zu erstellen, wie im the section called “Umfrage zum Jobstatus (Lambda, AWS Batch) ” Beispielprojekt gezeigt.

Zu den unterstützten Parametern gehören:

- `ClusterName`: Der Name des Amazon EKS-Clusters, den Sie aufrufen möchten.
  - `Type`: `String`
  - Erforderlich: Ja
- `CertificateAuthority`: Die Base64-codierten Zertifikatsdaten, die für die Kommunikation mit Ihrem Cluster erforderlich sind. Sie können diesen Wert von der Amazon EKS-Konsole oder mithilfe der Amazon DescribeClusterEKS-API abrufen.
  - `Type`: `String`
  - Erforderlich: Ja
- `Endpoint`: Die Endpunkt-URL für Ihren Kubernetes-API-Server. Sie können diesen Wert von der Amazon EKS-Konsole oder mithilfe der Amazon DescribeClusterEKS-API abrufen.
  - `Type`: `String`
  - Erforderlich: Ja
- `Namespace`: Der Namespace, in dem der Job ausgeführt werden soll. Wenn nicht angegeben, wird der Namespace `default` verwendet.
  - `Type`: `String`
  - Erforderlich: nein
- `Job`: Die Definition des Kubernetes-Jobs. Siehe Jobs in der Kubernetes-Dokumentation.

  - Type: `JSON` oder `String`
  - Erforderlich: Ja
- `LogOptions`: Eine Reihe von Optionen zur Steuerung des optionalen Abrufs von Protokollen. Gilt nur, wenn das Dienstintegrationsmuster Run a Job (.sync) verwendet wird, um auf den Abschluss des Jobs zu warten.
  - Type: `JSON`
  - Erforderlich: nein
  - Protokolle sind in der Antwort unter dem Schlüssel `logs` enthalten. Der Job kann mehrere Pods mit jeweils mehreren Containern enthalten.

```
{
  ...
  "logs": {
    "pods": {
      "pod1": {
        "containers": {
          "container1": {
            "log": <log>
          },
          ...
        }
      },
      ...
    }
  }
```

  - Der Protokollabruf erfolgt nach bestem Wissen und Gewissen. Wenn beim Abrufen eines Protokolls ein Fehler auftritt, werden anstelle des `log` Felds die Felder und angezeigt. `error cause`
- `LogOptions.RetrieveLogs`: Aktiviert den Protokollabruf nach Abschluss des Jobs. Standardmäßig werden Protokolle nicht abgerufen.
  - Type: `Boolean`
  - Erforderlich: nein
- `LogOptions.RawLogs`: Wenn auf true gesetzt `RawLogs` ist, werden Logs als unformatierte Zeichenketten zurückgegeben, ohne dass versucht wird, sie in JSON zu parsen. Standardmäßig werden Logs nach Möglichkeit in JSON deserialisiert. In einigen Fällen kann eine solche Analyse

zu unerwünschten Änderungen führen, z. B. zur Einschränkung der Genauigkeit von Zahlen mit vielen Ziffern.

- `Type`: `Boolean`
- Erforderlich: nein

- `LogOptions.LogParameters`: Die Read Log API der Kubernetes-API unterstützt Abfrageparameter zur Steuerung des Protokollabrufs. Sie können beispielsweise `tailLines` oder verwenden, `limitBytes` um die Größe der abgerufenen Protokolle zu begrenzen und dabei das Datengrößenkontingent von Step Functions einzuhalten. Weitere Informationen finden Sie im Abschnitt „Protokoll lesen“ der Kubernetes-API-Referenz.
  - `Type:` Karte von `String` `List of Strings`
  - Erforderlich: nein
  - Beispiel:

```
"LogParameters": {
  "tailLines": [ "6" ]
}
```

Das folgende Beispiel enthält einen `Task` Status, der einen Job ausführt, auf seinen Abschluss wartet und dann die Logs des Jobs abruft:

```
{
  "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`

Die `eks:call` Serviceintegration ermöglicht es Ihnen, die Kubernetes-API zu verwenden, um Kubernetes-Ressourcenobjekte über einen Kubernetes-API-Endpunkt zu lesen und zu schreiben.

Ihr Kubernetes-API-Server muss der von Ihrem Zustandsmaschine verwendeten IAM-Rolle Berechtigungen gewähren. Weitere Informationen finden Sie unter Berechtigungen.

Weitere Informationen zu den verfügbaren Vorgängen finden Sie in der Kubernetes-API-Referenz.

Zu den unterstützten Parametern für gehören`Call`:

- `ClusterName`: Der Name des Amazon EKS-Clusters, den Sie aufrufen möchten.

  - Type: Zeichenfolge
  - Erforderlich: Ja
- `CertificateAuthority`: Die Base64-codierten Zertifikatsdaten, die für die Kommunikation mit Ihrem Cluster erforderlich sind. Sie können diesen Wert von der Amazon EKS-Konsole oder mithilfe der Amazon DescribeClusterEKS-API abrufen.
  - Type: `String`
  - Erforderlich: Ja
- `Endpoint`: Die Endpunkt-URL für Ihren Kubernetes-API-Server. Sie finden diesen Wert in der Amazon EKS-Konsole oder mithilfe der Amazon DescribeCluster EKS-API.
  - Type: `String`
  - Erforderlich: Ja
- `Method`: Die HTTP-Methode Ihrer Anfrage. Eins von: `GET`, `POST`, `PUT`, `DELETE`, `HEAD` oder von `PATCH`.
  - Type: `String`
  - Erforderlich: Ja
- `Path`: Der HTTP-Pfad des Kubernetes-REST-API-Vorgangs.
  - Type: `String`
  - Erforderlich: Ja
- `QueryParameters`: Die HTTP-Abfrageparameter des Kubernetes-REST-API-Vorgangs.
  - Type: Karte von bis `String` `List of Strings`
  - Erforderlich: Nein
  - Beispiel:

```
"QueryParameters": {
  "labelSelector": [ "job-name=example-job" ]
}
```

- `RequestBody`: Der HTTP-Nachrichtentext des Kubernetes-REST-API-Vorgangs.
  - Type: `JSON` oder `String`
  - Erforderlich: Nein

Im Folgenden finden Sie einen `Task` Status, der verwendet wird, `eks:call` um die Pods aufzulisten, die zu dem Job gehören. `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
    }
  }
}
```

Der folgende Status enthält einen `Task` Status, der `eks:call` zum Löschen des Auftrags verwendet wird`example-job`, und legt den Status fest, `propagationPolicy` um sicherzustellen, dass die Pods des Auftrags ebenfalls gelöscht werden.

```
{
  "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
    }
  }
}
```

Unterstützte Amazon EKS-APIs

Zu den unterstützten Amazon EKS-APIs und -Syntax gehören:

- `CreateCluster`
  - Erforderliche Syntax
  - Antwortsyntax

    Wenn ein Amazon EKS-Cluster mithilfe der `eks:createCluster` Service-Integration erstellt wird, wird die IAM-Rolle als Administrator (mit system:masters-Berechtigungen) zur Kubernetes-RBAC-Autorisierungstabelle hinzugefügt. Anfänglich kann nur diese IAM-Entität den Kubernetes-API-Server aufrufen. Weitere Informationen finden Sie hier:

    - Verwaltung von Benutzern oder IAM-Rollen für Ihren Cluster im Amazon EKS-Benutzerhandbuch
    - Der Abschnitt Berechtigungen

    Amazon EKS verwendet serviceverknüpfte Rollen, die die Berechtigungen enthalten, die Amazon EKS benötigt, um andere Services in Ihrem Namen aufzurufen. Wenn diese serviceverknüpften Rollen noch nicht in Ihrem Konto vorhanden sind, müssen Sie die `iam:CreateServiceLinkedRole` Berechtigung zu der von Step Functions verwendeten IAM-Rolle hinzufügen. Weitere Informationen finden Sie unter Verwenden von serviceverknüpften Rollen im Amazon EKS-Benutzerhandbuch.

    Die von Step Functions verwendete IAM-Rolle muss über `iam:PassRole` Berechtigungen verfügen, um die Cluster-IAM-Rolle an Amazon EKS weiterzugeben. Weitere Informationen finden Sie unter Amazon EKS-Cluster-IAM-Rolle im Amazon EKS-Benutzerhandbuch.
- `DeleteCluster`
  - Erforderliche Syntax
  - Antwortsyntax

Sie müssen alle Fargate-Profile oder Knotengruppen löschen, bevor Sie einen Cluster löschen.

- `CreateFargateProfile`
  - Erforderliche Syntax
  - Antwortsyntax

  Amazon EKS verwendet serviceverknüpfte Rollen, die die Berechtigungen enthalten, die Amazon EKS benötigt, um andere Services in Ihrem Namen aufzurufen. Wenn diese serviceverknüpften Rollen noch nicht in Ihrem Konto vorhanden sind, müssen Sie die `iam:CreateServiceLinkedRole` Berechtigung zu der von Step Functions verwendeten IAM-Rolle hinzufügen. Weitere Informationen finden Sie unter Verwenden von serviceverknüpften Rollen im Amazon EKS-Benutzerhandbuch.

  Amazon EKS on Fargate ist möglicherweise nicht in allen Regionen verfügbar. Informationen zur regionalen Verfügbarkeit finden Sie im Abschnitt über Fargate im Amazon EKS-Benutzerhandbuch.

  Die von Step Functions verwendete IAM-Rolle muss über `iam:PassRole` Berechtigungen verfügen, um die IAM-Rolle für die Pod-Ausführung an Amazon EKS weiterzugeben. Weitere Informationen finden Sie unter Pod-Ausführungsrolle im Amazon EKS-Benutzerhandbuch.
- `DeleteFargateProfile`
  - Erforderliche Syntax
  - Antwortsyntax
- `CreateNodegroup`
  - Erforderliche Syntax
  - Antwortsyntax

  Amazon EKS verwendet eine serviceverknüpfte Rolle, die die Berechtigungen enthält, die Amazon EKS benötigt, um andere Dienste in Ihrem Namen aufzurufen. Wenn diese serviceverknüpften Rollen noch nicht in Ihrem Konto vorhanden sind, müssen Sie die `iam:CreateServiceLinkedRole` Berechtigung zu der von Step Functions verwendeten IAM-Rolle hinzufügen. Weitere Informationen finden Sie unter Verwenden von serviceverknüpften Rollen im Amazon EKS-Benutzerhandbuch.

  Die von Step Functions verwendete IAM-Rolle muss über `iam:PassRole` Berechtigungen verfügen, um die Knoten-IAM-Rolle an Amazon EKS weiterzugeben. Weitere Informationen finden Sie unter Verwenden von serviceverknüpften Rollen im Amazon EKS-Benutzerhandbuch.

- `DeleteNodegroup`
  - Erforderliche Syntax
  - Antwortsyntax

Das Folgende beinhaltet einen`Task`, der einen Amazon EKS-Cluster erstellt.

```
{
  "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
    }
  }
}
```

Im Folgenden wird ein `Task` Status beschrieben, der einen Amazon EKS-Cluster löscht.

```
{
  "StartAt": "DeleteCluster.sync",
  "States": {
    "DeleteCluster.sync": {
      "Type": "Task",
      "Resource": "arn:aws:states:::eks:deleteCluster.sync",
      "Parameters": {
        "Name": "MyCluster"
      },
      "End": true
    }
  }
```

```
}
```

Im Folgenden wird ein `Task` Bundesstaat beschrieben, der ein Fargate-Profil erstellt.

```
{
  "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
    }
  }
}
```

Das Folgende beinhaltet einen `Task` Status, der ein Fargate-Profil löscht.

```
{
  "StartAt": "DeleteFargateProfile.sync",
  "States": {
    "DeleteFargateProfile.sync": {
      "Type": "Task",
      "Resource": "arn:aws:states:::eks:deleteFargateProfile.sync",
      "Parameters": {
        "ClusterName": "MyCluster",
        "FargateProfileName": "MyFargateProfile"
      },
      "End": true
    }
  }
}
```

Im Folgenden wird ein `Task` Status beschrieben, der eine Knotengruppe erstellt.

```
{
  "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
    }
  }
}
```

Das Folgende beinhaltet einen `Task` Status, der eine Knotengruppe löscht.

```
{
  "StartAt": "DeleteNodegroup.sync",
  "States": {
    "DeleteNodegroup.sync": {
      "Type": "Task",
      "Resource": "arn:aws:states:::eks:deleteNodegroup.sync",
      "Parameters": {
        "ClusterName": "MyCluster",
        "NodegroupName": "MyNodegroup"
      },
      "End": true
    }
  }
}
```

Berechtigungen

Wenn ein Amazon EKS-Cluster mithilfe der `eks:createCluster` Serviceintegration erstellt wird, wird die IAM-Rolle als Administrator mit Berechtigungen zur Kubernetes-RBAC-Autorisierungstabelle hinzugefügt. `system:masters` Anfänglich kann nur diese IAM-Entität den Kubernetes-API-Server aufrufen. Beispielsweise können Sie kubectl nicht verwenden, um mit Ihrem Kubernetes-API-Server

zu interagieren, es sei denn, Sie übernehmen dieselbe Rolle wie Ihre Step Functions Functions-Zustandsmaschine oder wenn Sie Kubernetes so konfigurieren, dass es zusätzlichen IAM-Entitäten Berechtigungen erteilt. Weitere Informationen finden Sie unter Verwalten von Benutzern oder IAM-Rollen für Ihren Cluster im Amazon EKS-Benutzerhandbuch.

Sie können Berechtigungen für zusätzliche IAM-Entitäten wie Benutzer oder Rollen hinzufügen, indem Sie sie dem Namespace `aws-auth` `ConfigMap` in the kube-system hinzufügen. Wenn Sie Ihren Cluster aus Step Functions erstellen, verwenden Sie die `eks:call` Serviceintegration.

Im Folgenden finden Sie einen `Task` Status, der eine `aws-auth` `ConfigMap` IAM-Rolle `arn:aws:iam::123456789012:role/my-role` erstellt `arn:aws:iam::123456789012:user/my-user` und dem Benutzer eine `system:masters` entsprechende Berechtigung erteilt.

```
{
  "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**
>
> Möglicherweise wird der ARN für eine IAM-Rolle in einem Format angezeigt, das den Pfad /service-role/ enthält, z. B. `arn:aws:iam::123456789012:role/service-role/my-role` Dieses Pfadtoken für die Dienstrolle sollte nicht enthalten sein, wenn die Rolle in aufgeführt wird. `aws-auth`

Wenn Ihr Cluster zum ersten Mal erstellt `aws-auth` `ConfigMap` wird, ist er nicht vorhanden, wird aber automatisch hinzugefügt, wenn Sie ein Fargate-Profil erstellen. Sie können den aktuellen Wert von abrufen`aws-auth`, die zusätzlichen Berechtigungen und PUT eine neue Version hinzufügen. Es ist normalerweise einfacher, es `aws-auth` vor dem Fargate-Profil zu erstellen.

Wenn Ihr Cluster außerhalb von Step Functions erstellt wurde, können Sie kubectl so konfigurieren, dass es mit Ihrem Kubernetes-API-Server kommuniziert. Erstellen Sie dann einen neuen `aws-auth` `ConfigMap` mit `kubectl apply -f aws-auth.yaml` oder bearbeiten Sie einen, der bereits vorhanden ist, mit. `kubectl edit -n kube-system configmap/aws-auth` Weitere Informationen finden Sie hier:

- Erstellen Sie im Amazon EKS-Benutzerhandbuch eine kubeconfig für Amazon EKS.
- Verwaltung von Benutzern oder IAM-Rollen für Ihren Cluster im Amazon EKS-Benutzerhandbuch.

Wenn Ihre IAM-Rolle nicht über ausreichende Berechtigungen in Kubernetes verfügt, schlagen die `eks:call` oder `eks:runJob` Service-Integrationen mit dem folgenden Fehler fehl:

```
Error:
EKS.401

Cause:
{
  "ResponseBody": {
    "kind": "Status",
    "apiVersion": "v1",
    "metadata": {},
    "status": "Failure",
    "message": "Unauthorized",
    "reason": "Unauthorized",
```

```
    "code": 401
  },
  "StatusCode": 401,
  "StatusText": "Unauthorized"
}
```

Rufen Sie Amazon EMR mit Step Functions auf

Step Functions kann bestimmte AWS Dienste direkt von Amazon States Language (ASL) aus steuern. Weitere Informationen hierzu finden Sie unter Arbeiten mit anderen -Services und Parameter an eine Service-API übergeben.

> **Wie sich die optimierte Amazon EMR-Integration von der Amazon EMR AWS SDK-Integration unterscheidet**
>
> Die optimierte Amazon EMR-Serviceintegration verfügt über einen benutzerdefinierten Satz von APIs, die die zugrunde liegenden Amazon EMR-APIs umfassen, die unten beschrieben werden. Aus diesem Grund unterscheidet es sich erheblich von der Amazon EMR AWS SDK-Serviceintegration. Darüber hinaus wird das Ausführen einer Aufgabe (.sync) Integrationsmuster unterstützt.

Für die Integration AWS Step Functions mit Amazon EMR verwenden Sie die bereitgestellten Amazon EMR-Serviceintegrations-APIs. Die Service-Integrations-APIs ähneln den entsprechenden Amazon EMR-APIs, mit einigen Unterschieden in den Feldern, die übergeben werden, und in den Antworten, die zurückgegeben werden.

Step Functions beendet einen Amazon EMR-Cluster nicht automatisch, wenn die Ausführung gestoppt wird. Wenn Ihr State Machine stoppt, bevor Ihr Amazon EMR-Cluster beendet wurde, läuft Ihr Cluster möglicherweise auf unbestimmte Zeit weiter und es können zusätzliche Gebühren anfallen. Um dies zu vermeiden, stellen Sie sicher, dass jeder Amazon EMR-Cluster, den Sie erstellen, ordnungsgemäß beendet wird. Weitere Informationen finden Sie hier:

- Steuern Sie die Clusterbeendigung im Amazon EMR-Benutzerhandbuch.
- Der Ausführen einer Aufgabe (.sync) Abschnitt Serviceintegrationsmuster.

> **Note**
>
> Ab sofort `emr-5.28.0` können Sie den Parameter `StepConcurrencyLevel` beim Erstellen eines Clusters angeben, damit mehrere Schritte parallel auf einem einzelnen Cluster ausgeführt werden können. Sie können die Step-Funktionen `Map` und `Parallel` -Status verwenden, um Arbeiten parallel an den Cluster einzureichen.

Die Verfügbarkeit der Amazon EMR-Serviceintegration hängt von der Verfügbarkeit der Amazon EMR-APIs ab. Informationen zu Einschränkungen in bestimmten Regionen finden Sie in der Amazon EMR-Dokumentation.

> **Note**
>
> Für die Integration mit Amazon EMR verfügt Step Functions über eine fest codierte Jobabfragefrequenz von 60 Sekunden für die ersten 10 Minuten und danach 300 Sekunden.

In der folgenden Tabelle werden die Unterschiede zwischen den einzelnen Service-Integrations-APIs und der entsprechenden Amazon EMR-API beschrieben.

Amazon EMR Service Integration APIs und entsprechende Amazon EMR APIs

API für die Amazon EMR-Serviceintegration	Entsprechende EMR-API	Unterschiede
createCluster Erstellt und startet einen Cluster (Auftragsverlauf). Amazon EMR ist direkt mit einer speziellen Art von IAM-Rolle verknüpft, die als serviceverknüpfte Rolle bezeichnet wird. Damit `createCluster` und `createCluster.sync` funktionieren, müssen Sie die	runJobFlow	`createCluster` verwendet dieselbe Anfragesyntax wie runJobFlow, mit Ausnahme der folgenden: • Das Feld `Instances.KeepJobFlowAliveWhenNoSteps` ist obligatorisch und muss den booleschen Wert TRUE aufweisen.

API für die Amazon EMR-Serviceintegration	Entsprechende EMR-API	Unterschiede

erforderlichen Berechtigungen zum Erstellen der serviceverknüpften Rolle `AWSServiceRoleForEMRCleanup` konfiguriert haben. Weitere Informationen dazu, einschließlich einer Erklärung, die Sie zu Ihrer IAM-Berechtigungsrichtlinie hinzufügen können, finden Sie unter Verwenden der serviceverknüpften Rolle für Amazon EMR.		• Das Feld `Steps` ist nicht zulässig.
		• Das Feld `Instances.InstanceFleets[index].Name` sollte angegeben werden und muss eindeutig sein, wenn die optionale Connector-API `modifyInstanceFleetByName` verwendet wird.
		• Das Feld `Instances.InstanceGroups[index].Name` sollte angegeben werden und muss eindeutig sein, wenn die optionale `modifyInstanceGroupByName` - API verwendet wird.
		
		Die Antwort lautet:
		
		`{ "ClusterId": "string" }`
		
		Amazon EMR verwendet Folgendes:
		
		`{ "JobFlowId": "string" }`

API für die Amazon EMR-Serviceintegration	Entsprechende EMR-API	Unterschiede
createCluster.sync Erstellt und startet einen Cluster (Auftragsverlauf).	runJobFlow	Entspricht `createCluster` , wartet aber darauf, dass der Cluster den Zustand `WAITING` erreicht.

setClusterTerminationSchutz
 	
Sperrt einen Cluster (Auftragsverlauf), sodass die EC2-Instances im Cluster nicht durch Benutzereingriffe, einen API-Aufruf oder durch einen Fehler im Auftragsverlauf beendet werden können.
	setTerminationProtection	Anforderung verwendet Folgendes:
		
		`{ "ClusterId": "string" }`
		
		Amazon EMR verwendet Folgendes:
		
		`{ "JobFlowIds": ["string"] }`

terminateCluster
 	
Beendet einen Cluster (Auftragsverlauf).
	terminateJobFlows	Anforderung verwendet Folgendes:
		
		`{ "ClusterId": "string" }`
		
		Amazon EMR verwendet Folgendes:
		
		`{ "JobFlowIds": ["string"] }`

API für die Amazon EMR-Serviceintegration	Entsprechende EMR-API	Unterschiede
terminateCluster.sync Beendet einen Cluster (Auftragsverlauf).	terminateJobFlows	Entspricht `terminateCluster` , wartet aber auf den Abschluss des Clusters.

API für die Amazon EMR-Serviceintegration	Entsprechende EMR-API	Unterschiede

addStep
 	
Fügt einem ausgeführten Cluster einen neuen Schritt hinzu.
 	
Optional können Sie den `ExecutionRoleArn` Parameter auch angeben, während Sie diese API verwenden.
	addJobFlowSchritte	Die Anfrage verwendet den Schlüssel`"ClusterId"` . Amazon EMR verwendet `"JobFlowId"` . Anforderung verwendet einen einzelnen Schritt.
		
		`{ "Step": <"StepConfig object"> }`
		
		Amazon EMR verwendet Folgendes:
		
		`{ "Steps": [<StepConfig objects>] }`
		
		Die Antwort lautet:
		
		`{ "StepId": "string" }`
		
		Amazon EMR gibt Folgendes zurück:
		
		`{ "StepIds": [<strings>] }`

API für die Amazon EMR-Serviceintegration	Entsprechende EMR-API	Unterschiede
addStep.sync Fügt einem ausgeführten Cluster einen neuen Schritt hinzu. Optional können Sie den `ExecutionRoleArn` Parameter auch angeben, während Sie diese API verwenden.	addJobFlowSchritte	Wie `addStep`, wartet aber auf den Abschluss des Schrittes.

API für die Amazon EMR-Serviceintegration	Entsprechende EMR-API	Unterschiede

cancelStep
 	
Bricht einen ausstehenden Schritt in einem laufenden Cluster ab.
	cancelSteps	Anforderung verwendet Folgendes:
		
		`{ "StepId": "string" }`
		
		Amazon EMR verwendet Folgendes:
		
		`{ "StepIds": [<strings>] }`
		
		Die Antwort lautet:
		
		`{ "CancelStepsInfo": <CancelStepsInfo object> }`
		
		Amazon EMR verwendet Folgendes:
		
		`{ "CancelStepsInfoList": [<CancelStepsInfo objects>] }`

API für die Amazon EMR-Serviceintegration	Entsprechende EMR-API	Unterschiede
modifyInstanceFleetByName Ändert die Ziel-On-Demand- und Ziel-Spot-Kapazitäten für die Instance-Flotte mit dem angegebenen `InstanceFleetName` .	modifyInstanceFleet	Die Anforderung entspricht der für `modifyInstanceFleet` , mit Ausnahme von Folgendem: • Das Feld `Instance.InstanceFleetId` ist nicht zulässig. • Zur Laufzeit wird die `InstanceFleetId` automatisch anhand der Serviceintegration bestimmt, indem `ListInstanceFleets` aufgerufen und das Ergebnis analysiert wird.

API für die Amazon EMR-Serviceintegration	Entsprechende EMR-API	Unterschiede

modifyInstanceGroupByName
 	
Ändert die Anzahl der Knoten und Konfigurationseinstellungen einer Instance-Gruppe.
	modifyInstanceGroups	Anforderung lautet:
		
		`{ "ClusterId": "string", "InstanceGroup": <InstanceGroupModifyConfig object> }`
		
		Amazon EMR verwendet eine Liste:
		
		`{ "ClusterId": ["string"], "InstanceGroups": [<InstanceGroupModifyConfig objects>] }`
		
		Innerhalb des Objekts `InstanceGroupModifyConfig` ist das Feld `InstanceGroupId` nicht zulässig.
		
		Das neue Feld `InstanceGroupName` wurde hinzugefügt. Zur Laufzeit wird die `InstanceGroupId` automatisch anhand der Serviceintegration bestimmt, indem `ListInstanceGroups` aufgerufen und das Ergebnis analysiert wird.

Im Folgenden finden Sie einen `Task`-Zustand, der einen Cluster erstellt.

```
"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
```

```
}
```

Im Folgenden finden Sie einen `Task`-Zustand, der den Kündigungsschutz ermöglicht.

```
"Enable_Termination_Protection": {
    "Type": "Task",
    "Resource": "arn:aws:states:::elasticmapreduce:setClusterTerminationProtection",
    "Parameters": {
        "ClusterId.$": "$.ClusterId",
        "TerminationProtected": true
    },
    "End": true
}
```

Im Folgenden finden Sie einen `Task`-Zustand, der einen Schritt an einen Cluster sendet.

```
"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
```

```
}
```

Im Folgenden finden Sie einen `Task`-Zustand, der einen Schritt abbricht.

```
"Cancel_Step_One": {
    "Type": "Task",
    "Resource": "arn:aws:states:::elasticmapreduce:cancelStep",
    "Parameters": {
        "ClusterId.$": "$.ClusterId",
        "StepId.$": "$.AddStepsResult.StepId"
    },
    "End": true
}
```

Im Folgenden finden Sie einen `Task`-Zustand, der einen Cluster beendet.

```
"Terminate_Cluster": {
    "Type": "Task",
    "Resource": "arn:aws:states:::elasticmapreduce:terminateCluster.sync",
    "Parameters": {
        "ClusterId.$": "$.ClusterId"
    },
    "End": true
}
```

Im Folgenden finden Sie einen `Task`-Zustand, der einen Cluster für eine Instance-Gruppe nach oben oder unten skaliert.

```
"ModifyInstanceGroupByName": {
    "Type": "Task",
    "Resource": "arn:aws:states:::elasticmapreduce:modifyInstanceGroupByName",
    "Parameters": {
        "ClusterId": "j-1234567890123",
        "InstanceGroupName": "MyCoreGroup",
        "InstanceGroup": {
            "InstanceCount": 8
        }
    },
    "End": true
}
```

Im Folgenden finden Sie einen `Task`-Zustand, der einen Cluster für eine Instance-Flotte nach oben oder unten skaliert.

```
"ModifyInstanceFleetByName": {
    "Type": "Task",
    "Resource": "arn:aws:states:::elasticmapreduce:modifyInstanceFleetByName",
    "Parameters": {
        "ClusterId": "j-1234567890123",
        "InstanceFleetName": "MyCoreFleet",
        "InstanceFleet": {
            "TargetOnDemandCapacity": 8,
            "TargetSpotCapacity": 0
        }
    },
    "End": true
}
```

Informationen zur Konfiguration von IAM Berechtigungen bei der Verwendung Step Functions mit anderen AWS Diensten finden Sie unterIAM-Richtlinien für integrierte Dienste.

Rufen Sie Amazon EMR auf EKS an mit AWS Step Functions

Step Functions kann bestimmte AWS Dienste direkt von Amazon States Language (ASL) aus steuern. Weitere Informationen hierzu finden Sie unter Arbeiten mit anderen -Services und Parameter an eine Service-API übergeben.

Wie sich die optimierte Amazon EMR on EKS-Integration von der Amazon EMR on EKS AWS SDK-Integration unterscheidet

- Das Ausführen einer Aufgabe (.sync) Integrationsmuster wird unterstützt.
- Es gibt keine Optimierungen für das Request Response (Antwort anfordern) Integrationsmuster.
- Das Warten auf einen Callback mit dem Aufgabentoken Integrationsmuster wird nicht unterstützt.

> **Note**
>
> Für die Integration mit Amazon EMR verfügt Step Functions über eine fest codierte Jobabfragefrequenz von 60 Sekunden für die ersten 10 Minuten und danach 300 Sekunden.

Verwenden Sie für die Integration AWS Step Functions mit Amazon EMR on EKS die Service-Integrations-APIs von Amazon EMR on EKS. Die Service-Integrations-APIs sind dieselben wie die entsprechenden Amazon EMR on EKS-APIs, aber nicht alle APIs unterstützen alle Integrationsmuster, wie in der folgenden Tabelle dargestellt.

API	Antwort anfordern	Führen Sie einen Job aus (.sync)
CreateVirtualCluster	✓	
DeleteVirtualCluster	✓	✓
StartJobRun	✓	✓

Unterstützte Amazon EMR auf EKS-APIs:

> **Note**
>
> In Step Functions gibt es ein Kontingent für die maximale Eingabe- oder Ergebnisdatengröße für eine Aufgabe. Dadurch sind Sie auf 256 KB an Daten als UTF-8-kodierte Zeichenfolge beschränkt, wenn Sie Daten an einen anderen Dienst senden oder von einem anderen Dienst empfangen. Siehe Kontingente im Zusammenhang mit der Ausführung von Zustandsmaschinen.

- `CreateVirtualCluster`
  - Erforderliche Syntax
  - Unterstützte Parameter
  - Antwortsyntax
- `DeleteVirtualCluster`

  - Erforderliche Syntax
  - Unterstützte Parameter
  - Antwortsyntax
- `StartJobRun`
  - Erforderliche Syntax
  - Unterstützte Parameter
  - Antwortsyntax

Im Folgenden wird ein `Task` Status beschrieben, der einen virtuellen Cluster erstellt.

```
"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
}
```

Im Folgenden wird ein `Task` Status beschrieben, der einen Auftrag an einen virtuellen Cluster weiterleitet und darauf wartet, dass er abgeschlossen ist.

```
"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
}
```

Der folgende Status beinhaltet einen `Task` Status, der einen virtuellen Cluster löscht und darauf wartet, dass der Löschvorgang abgeschlossen ist.

```
"Delete_Virtual_Cluster": {
  "Type": "Task",
  "Resource": "arn:aws:states:::emr-containers:deleteVirtualCluster.sync",
  "Parameters": {
    "Id.$": "$.VirtualClusterId"
  },
  "End": true
}
```

Informationen zur Konfiguration von IAM Berechtigungen bei der Verwendung Step Functions mit anderen AWS Diensten finden Sie unter. IAM-Richtlinien für integrierte Dienste

Aufrufen von Amazon EMR Serverless mit Step Functions

Step Functions kann bestimmte AWS Dienste direkt von Amazon States Language (ASL) aus steuern. Weitere Informationen hierzu finden Sie unter Arbeiten mit anderen -Services und Parameter an eine Service-API übergeben.

Wie unterscheidet sich die optimierte EMR Serverless Integration von der EMR Serverless AWS SDK-Integration

- Die optimierte EMR Serverless Serviceintegration verfügt über einen benutzerdefinierten Satz von APIs, die die zugrunde liegenden EMR Serverless APIs umfassen. Aufgrund dieser Anpassung unterscheidet sich die optimierte EMR Serverless Integration erheblich von der EMR Serverless AWS SDK-Serviceintegration. Darüber hinaus unterstützt die optimierte EMR Serverless Integration das Ausführen einer Aufgabe (.sync) Integrationsmuster.
- Das Warten auf einen Callback mit dem Aufgabentoken Integrationsmuster wird nicht unterstützt.

In diesem Thema

EMR ServerlessAPIs zur Serviceintegration

Für die AWS Step Functions Integration EMR Serverless können Sie die folgenden sechs EMR Serverless Serviceintegrations-APIs verwenden. Diese APIs zur Serviceintegration ähneln den entsprechenden EMR Serverless APIs, mit einigen Unterschieden in den Feldern, die übergeben werden, und in den Antworten, die zurückgegeben werden.

Die Unterschiede zwischen den einzelnen Service-Integrations-APIs und den entsprechenden EMR Serverless-APIs werden in der folgenden Tabelle beschrieben.

EMR ServerlessAPIs für die Serviceintegration und entsprechende EMR Serverless APIs

EMR ServerlessAPI zur Serviceintegration	Entsprechende EMR Serverless API	Unterschiede
Anwendung erstellen Erstellt eine Anwendung EMR Serverlessist mit einem eindeutigen Rollentyp verknüpft, der als dienstbez ogene IAM Rolle bezeichne t wird. Damit `createApp lication` und `createApp lication.sync` funktioni eren, müssen Sie die erforderlichen Berechtig ungen zum Erstellen der serviceverknüpften Rolle `AWS ServiceRoleForAmaz onEMRServerless` konfiguriert haben. Weitere Informationen dazu, einschlie ßlich einer Erklärung, die Sie zu Ihrer IAM Berechtig ungsrichtlinie hinzufügen können, finden Sie unter Verwenden von dienstbez	CreateApplication	None

EMR ServerlessAPI zur Serviceintegration	Entsprechende EMR Serverless API	Unterschiede
ogenen Rollen für. EMR Serverless		
CreateApplication.Sync Erstellt eine Anwendung	CreateApplication	Keine Unterschiede zwischen den Anfragen und Antworten der API und der EMR Serverless EMR Serverles s Service-Integrations-API. CreateApplication.Sync wartet jedoch darauf, dass die Anwendung den Status erreicht. `CREATED`
Anwendung starten Startet eine angegebene Anwendung und initialisiert die Anfangskapazität der Anwendung, falls sie konfiguri ert ist.	StartApplication	Die EMR Serverless API-Antwort enthält keine Daten, aber die API-Antwort zur EMR Serverless Serviceintegration enthält die folgenden Daten. `{ "ApplicationId": "string" }`

EMR ServerlessAPI zur Serviceintegration	Entsprechende EMR Serverless API	Unterschiede
Starten Sie Application.Sync Startet eine angegebene Anwendung und initialisiert die Anfangskapazität, falls konfiguriert.	StartApplication	Die EMR Serverless API-Antwort enthält keine Daten, aber die API-Antwort für die EMR Serverless Servicein tegration enthält die folgenden Daten. `{ "ApplicationId": "string" }` Außerdem wartet StartAppl ication.Sync darauf, dass die Anwendung den Status erreicht. `STARTED`
Anwendung beenden Stoppt eine angegebene Anwendung und gibt die Anfangskapazität frei, falls sie konfiguriert ist. Alle geplanten und laufenden Jobs müssen abgeschlossen oder abgebrochen werden, bevor eine Anwendung gestoppt werden kann.	StopApplication	Die EMR Serverless API-Antwort enthält keine Daten, aber die API-Antwort zur EMR Serverless Serviceintegration enthält die folgenden Daten. `{ "ApplicationId": "string" }`

EMR ServerlessAPI zur Serviceintegration	Entsprechende EMR Serverless API	Unterschiede
StopApplication.Sync Stoppt eine angegebene Anwendung und gibt die Anfangskapazität frei, falls sie konfiguriert ist. Alle geplanten und laufenden Jobs müssen abgeschlossen oder abgebrochen werden, bevor eine Anwendung gestoppt werden kann.	StopApplication	Die EMR Serverless API-Antwort enthält keine Daten, aber die API-Antwort zur EMR Serverless Serviceintegration enthält die folgenden Daten. `{ "ApplicationId": "string" }` Außerdem wartet StopApplication.Sync darauf, dass die Anwendung den Status erreicht. `STOPPED`
Anwendung löschen Löscht eine Anwendung. Eine Anwendung muss sich im `CREATED` Status `STOPPED` oder befinden, um gelöscht zu werden.	DeleteApplication	Die EMR Serverless API-Antwort enthält keine Daten, aber die API-Antwort zur EMR Serverless Serviceintegration enthält die folgenden Daten. `{ "ApplicationId": "string" }`

EMR ServerlessAPI zur Serviceintegration	Entsprechende EMR Serverless API	Unterschiede
DeleteApplication.Sync Löscht eine Anwendung. Eine Anwendung muss sich im `CREATED` Status `STOPPED` oder befinden, um gelöscht zu werden.	DeleteApplication	Die EMR Serverless API-Antwort enthält keine Daten, aber die API-Antwort zur EMR Serverless Serviceintegration enthält die folgenden Daten. `{ "ApplicationId": "string" }` Außerdem wartet StopApplication.Sync darauf, dass die Anwendung den Status erreicht. `TERMINATED`
startJobRun Startet eine Auftragsausführung.	StartJobRun	None
startJobRun.sync Startet einen Joblauf.	StartJobRun	Keine Unterschiede zwischen den Anfragen und Antworten der EMR Serverless API und der EMR Serverless Service-Integrations-API. startJobRun.sync wartet jedoch darauf, dass die Anwendung den Status erreicht. `SUCCESS`
cancelJobRun Bricht die Ausführung eines Jobs ab.	CancelJobRun	None

EMR ServerlessAPI zur Serviceintegration	Entsprechende EMR Serverless API	Unterschiede
cancelJobRun.sync Bricht einen Joblauf ab.	CancelJobRun	Keine Unterschiede zwischen den Anfragen und Antworten der EMR Serverless API und der EMR Serverless Service-Integrations-API. cancelJobRun.sync wartet jedoch darauf, dass die Anwendung den Status erreicht. `CANCELLED`

Anwendungsfälle für die serverlose EMR-Integration

Für die optimierte EMR Serverless Serviceintegration empfehlen wir, dass Sie eine einzelne Anwendung erstellen und diese Anwendung dann verwenden, um mehrere Jobs auszuführen. In einer einzigen Zustandsmaschine können Sie beispielsweise mehrere startJobRunAnfragen einschließen, die alle dieselbe Anwendung verwenden. Die folgenden Status der Aufgabe Statusbeispiele zeigen Anwendungsfälle für die Integration EMR Serverless von APIsStep Functions. Informationen zu anderen Anwendungsfällen von EMR Serverless finden Sie unter Was ist Amazon EMR Serverless.

> **Tip**
>
> Ein Beispiel EMR Serverless für eine Zustandsmaschine, die in die Ausführung mehrerer Jobs integriert ist AWS-Konto, finden Sie unterEinen EMR Serverless Job ausführen.

- Erstellen einer Anwendung
- Starten Sie eine Anwendung
- Stoppen Sie eine Anwendung
- Löschen einer Anwendung
- Starten Sie einen Job in einer Anwendung
- Stornieren Sie einen Job in einer Anwendung

Informationen zur Konfiguration von IAM Berechtigungen bei der Verwendung Step Functions mit anderen AWS Diensten finden Sie unterIAM-Richtlinien für integrierte Dienste.

Ersetzen Sie in den Beispielen, die in den folgenden Anwendungsfällen gezeigt werden, den *kursiv gedruckten* Text durch Ihre ressourcenspezifischen Informationen. Ersetzen Sie ihn beispielsweise *yourApplicationId*durch die ID Ihrer EMR Serverless Anwendung, z. B. `00yv7iv71inak893`

Erstellen einer Anwendung

Im folgenden Beispiel für einen Task-Status wird eine Anwendung mithilfe der CreateApplication.Sync-Dienstintegrations-API erstellt.

```
"Create_Application": {
    "Type": "Task",
    "Resource": "arn:aws:states:::emr-serverless:createApplication.sync",
    "Parameters": {
        "Name": "MyApplication",
        "ReleaseLabel": "emr-6.9.0",
        "Type": "SPARK"
    },
    "End": true
}
```

Starten Sie eine Anwendung

Im folgenden Beispiel für einen Task-Status wird eine Anwendung mithilfe der Dienstintegrations-API StartApplication.Sync gestartet.

```
"Start_Application": {
    "Type": "Task",
    "Resource": "arn:aws:states:::emr-serverless:startApplication.sync",
    "Parameters": {
        "ApplicationId": "yourApplicationId"
    },
    "End": true
}
```

Stoppen Sie eine Anwendung

Im folgenden Beispiel mit einem Task-Status wird eine Anwendung mithilfe der Service-Integrations-API StopApplication.Sync gestoppt.

```
"Stop_Application": {
    "Type": "Task",
    "Resource": "arn:aws:states:::emr-serverless:stopApplication.sync",
    "Parameters": {
        "ApplicationId": "yourApplicationId"
    },
    "End": true
}
```

Löschen einer Anwendung

Im folgenden Beispiel für einen Task-Status wird eine Anwendung mithilfe der Serviceintegrations-API DeleteApplication.Sync gelöscht.

```
"Delete_Application": {
    "Type": "Task",
    "Resource": "arn:aws:states:::emr-serverless:deleteApplication.sync",
    "Parameters": {
        "ApplicationId": "yourApplicationId"
    },
    "End": true
}
```

Starten Sie einen Job in einer Anwendung

Im folgenden Beispiel für den Aufgabenstatus wird ein Job in einer Anwendung mithilfe der Dienstintegrations-API startJobRun.sync gestartet.

```
"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
}
```

Stornieren Sie einen Job in einer Anwendung

Im folgenden Beispiel für den Aufgabenstatus wird ein Job in einer Anwendung mithilfe der Dienstintegrations-API für cancelJobRun.sync storniert.

```
"Cancel_Job": {
    "Type": "Task",
    "Resource": "arn:aws:states:::emr-serverless:cancelJobRun.sync",
    "Parameters": {
        "ApplicationId.$": "$.ApplicationId",
        "JobRunId.$": "$.JobRunId"
    },
    "End": true
}
```

Rufen Sie EventBridge mit Step Functions auf

Step Functions kann bestimmte AWS Dienste direkt von Amazon States Language (ASL) aus steuern. Weitere Informationen hierzu finden Sie unter Arbeiten mit anderen -Services und Parameter an eine Service-API übergeben.

> Wie unterscheidet sich die optimierte EventBridge Integration von der EventBridge AWS SDK-Integration
>
> - Der Ausführungs-ARN und der State-Machine-ARN werden jeweils `PutEventsRequestEntry` automatisch an das jeweilige `Resources` Feld angehängt.
> - Wenn die Antwort von einen Wert ungleich Null `PutEvents` `FailedEntryCount` enthält, schlägt der `Task` Status mit dem Fehler fehl. `EventBridge.FailedEntry`

Hinweise zur Konfiguration von IAM Berechtigungen bei der Verwendung Step Functions mit anderen AWS Diensten finden Sie unterIAM-Richtlinien für integrierte Dienste.

Step Functions bietet eine Service-Integrations-API für die Integration mit Amazon EventBridge. Auf diese Weise können Sie ereignisgesteuerte Anwendungen erstellen, indem Sie benutzerdefinierte Ereignisse direkt aus den Workflows von Step Functions senden.

Um die `PutEvents` API verwenden zu können, müssen Sie in Ihrem Konto eine EventBridge Regel erstellen, die dem spezifischen Muster der Ereignisse entspricht, die Sie senden werden. So können Sie beispielsweise:

- Erstellen Sie in Ihrem Konto eine Lambda-Funktion, die ein Ereignis empfängt und ausgibt, das einer EventBridge Regel entspricht.
- Erstellen Sie in Ihrem Konto auf dem Standard-Event-Bus eine EventBridge Regel, die einem bestimmten Ereignismuster entspricht und auf die Lambda-Funktion abzielt.

Weitere Informationen finden Sie hier:

- Hinzufügen von EventBridge Amazon-Veranstaltungen mit PutEvents im EventBridge Benutzerhandbuch.
- Warten auf einen Callback mit dem Aufgabentokenin Muster der Serviceintegration.

> **Note**
>
> In Step Functions gibt es ein Kontingent für die maximale Eingabe- oder Ergebnisdatengröße für eine Aufgabe. Dadurch sind Sie auf 256 KB an Daten als UTF-8-kodierte Zeichenfolge beschränkt, wenn Sie Daten an einen anderen Dienst senden oder von einem anderen Dienst empfangen. Siehe Kontingente im Zusammenhang mit der Ausführung von Zustandsmaschinen.

Unterstützte API EventBridge

Zu den unterstützten EventBridge APIs und Syntax gehören:

- `PutEvents`
  - Erforderliche Syntax
  - Unterstützte Parameter:
    - `Entries`

- Antwortsyntax

Folgendes beinhaltet einen`Task`, der ein benutzerdefiniertes Ereignis sendet:

```
{
  "Type": "Task",
  "Resource": "arn:aws:states:::events:putEvents",
  "Parameters": {
    "Entries": [
      {
        "Detail": {
          "Message": "MyMessage"
        },
        "DetailType": "MyDetailType",
        "EventBusName": "MyEventBus",
        "Source": "my.source"
      }
    ]
  },
  "End": true
}
```

Fehlerbehandlung

Die `PutEvents` API akzeptiert ein Array von Einträgen als Eingabe und gibt dann ein Array von Ergebniseinträgen zurück. Solange die `PutEvents` Aktion erfolgreich war, `PutEvents` wird eine HTTP 200-Antwort zurückgegeben, auch wenn ein oder mehrere Einträge fehlgeschlagen sind. `PutEvents`gibt die Anzahl der fehlgeschlagenen Einträge im `FailedEntryCount` Feld zurück.

Step Functions prüft, ob der größer als Null `FailedEntryCount` ist. Wenn es größer als Null ist, schlägt Step Functions den Status mit dem Fehler fehl`EventBridge.FailedEntry`. Auf diese Weise können Sie die integrierte Fehlerbehandlung von Step Functions für Aufgabenstatus verwenden, um fehlgeschlagene Eingaben abzufangen oder erneut zu versuchen, anstatt einen zusätzlichen Status verwenden zu müssen, um die Antwort `FailedEntryCount` anhand der Antwort zu analysieren.

> **Note**
>
> Wenn Sie Idempotenz implementiert haben und sicher alle Einträge erneut versuchen können, können Sie die Wiederholungslogik von Step Functions verwenden. Step Functions

entfernt erfolgreiche Einträge nicht aus dem `PutEvents` Eingabearray, bevor es erneut versucht wird. Stattdessen versucht es erneut mit dem ursprünglichen Array von Einträgen.

AWS Glue Jobs mit Step-Funktionen verwalten

Step Functions kann bestimmte AWS Dienste direkt von Amazon States Language (ASL) aus steuern. Weitere Informationen hierzu finden Sie unter Arbeiten mit anderen -Services und Parameter an eine Service-API übergeben.

Wie unterscheidet sich die optimierte AWS Glue Integration von der AWS GlueAWS SDK-Integration

- Das Ausführen einer Aufgabe (.sync) Integrationsmuster ist verfügbar.
- Das `JobName` Feld wird aus der Anfrage extrahiert und in die Antwort eingefügt, die normalerweise nur enthält`JobRunID`.

Unterstützte AWS Glue API:

- `StartJobRun`

Die Parameter in Step Functions werden ausgedrückt in PascalCase

Auch wenn sich die native Service-API in CamelCase befindet, z. B. die API-Aktion`startSyncExecution`, geben Sie Parameter in an PascalCase, z. B.:. `StateMachineArn`

Das Folgende beinhaltet einen `Task` Status, der einen AWS Glue Job startet.

```
"Glue StartJobRun": {
      "Type": "Task",
      "Resource": "arn:aws:states:::glue:startJobRun.sync",
      "Parameters": {
        "JobName": "GlueJob-JTrRO5l98qMG"
      },
```

```
      "Next": "ValidateOutput"
    },
```

Informationen zur Konfiguration von IAM Berechtigungen bei der Verwendung Step Functions mit anderen AWS Diensten finden Sie unterIAM-Richtlinien für integrierte Dienste.

AWS Glue DataBrew Jobs mit Step-Funktionen verwalten

Step Functions kann bestimmte AWS Dienste direkt von Amazon States Language (ASL) aus steuern. Weitere Informationen hierzu finden Sie unter Arbeiten mit anderen -Services und Parameter an eine Service-API übergeben.

Sie können die DataBrew Integration verwenden, um Schritte zur Datenbereinigung und Datennormalisierung in Ihre Analyse- und Machine-Learning-Workflows aufzunehmen.

Unterstützte DataBrew API:

- `StartJobRun`

Im Folgenden wird ein `Task` Status beschrieben, der einen Anforderungs- und Antwortauftrag DataBrew startet.

```
"DataBrew StartJobRun": {
            "Type": "Task",
            "Resource": "arn:aws:states:::databrew:startJobRun",
            "Parameters": {
               "Name": "sample-proj-job-1"
            },
            "Next": "NEXT_STATE"
          },
```

Im Folgenden wird ein `Task` Status beschrieben, der einen DataBrew Synchronisierungsauftrag startet.

```
"DataBrew StartJobRun": {
            "Type": "Task",
            "Resource": "arn:aws:states:::databrew:startJobRun.sync",
            "Parameters": {
               "Name": "sample-proj-job-1"
```

```
        },
        "Next": "NEXT_STATE"
      },
```

Informationen zur Konfiguration von IAM Berechtigungen bei der Verwendung Step Functions mit anderen AWS Diensten finden Sie unterIAM-Richtlinien für integrierte Dienste.

Lambda mit Step-Funktionen aufrufen

Step Functions kann bestimmte AWS Dienste direkt von Amazon States Language (ASL) aus steuern. Weitere Informationen hierzu finden Sie unter Arbeiten mit anderen -Services und Parameter an eine Service-API übergeben.

> Wie sich die optimierte Lambda-Integration von der AWS Lambda-SDK-Integration unterscheidet
>
> - Das `Payload` Feld der Antwort wird von maskiertem Json nach Json analysiert.
> - Wenn die Antwort das Feld enthält `FunctionError` oder innerhalb der Lambda-Funktion eine Ausnahme ausgelöst wird, schlägt die Aufgabe fehl.

Weitere Informationen zum Verwalten von Zustandseingaben, -ausgaben und -ergebnissen finden Sie unter Eingabe- und Ausgabeverarbeitung in Step Functions.

Unterstützte AWS Lambda APIs:

- `Invoke`
  - Anforderungssyntax
  - Unterstützte Parameter
    - `ClientContext`
    - `FunctionName`
    - `InvocationType`
    - `Qualifier`
    - `Payload`
  - Antwortsyntax

> **Die Parameter in Step Functions werden ausgedrückt in PascalCase**
>
> Auch wenn sich die native Service-API in CamelCase befindet, z. B. die API-Aktion`startSyncExecution`, geben Sie Parameter in an PascalCase, z. B.:. `StateMachineArn`

Das Folgende beinhaltet einen `Task` Zustand, der eine Lambda-Funktion aufruft.

```
{
   "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
      }
   }
}
```

Das Folgende enthält einen `Task`-Zustand zur Implementierung des Callback- Service-Integrationsmusters.

```
{
   "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
```

```
      }
   }
}
```

Wenn Sie eine Lambda-Funktion aufrufen, wartet die Ausführung, bis die Funktion abgeschlossen ist. Wenn Sie die Lambda-Funktion mit einer Callback-Aufgabe aufrufen, beginnt das Heartbeat-Timeout erst zu zählen, wenn die Ausführung der Lambda-Funktion abgeschlossen und ein Ergebnis zurückgegeben wurde. Solange die Lambda-Funktion ausgeführt wird, wird das Heartbeat-Timeout nicht erzwungen.

Es ist auch möglich, Lambda asynchron mithilfe des `InvocationType` Parameters aufzurufen, wie im folgenden Beispiel zu sehen ist:

> **Note**
>
> Bei asynchronen Aufrufen von Lambda-Funktionen beginnt die Heartbeat-Timeout-Periode sofort.

```
{

  "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
    }
  }
}
```

Wenn das `Task` Ergebnis zurückgegeben wird, ist die Funktionsausgabe in einem Metadatenwörterbuch verschachtelt. Beispielsweise:

```
{
```

```
   "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
}
```

Alternativ können Sie eine Lambda-Funktion aufrufen, indem Sie einen Funktions-ARN direkt im Feld „Resource“ angeben. Wenn Sie eine Lambda-Funktion auf diese Weise aufrufen, können Sie sie nicht angeben `.waitForTaskToken`, und das Aufgabenergebnis enthält nur die Funktionsausgabe.

```
{
   "StartAt":"CallFunction",
   "States":{
      "CallFunction": {
         "Type":"Task",
         "Resource":"arn:aws:lambda:us-east-1:123456789012:function:HelloFunction",
         "End": true
      }
   }
}
```

Sie können eine bestimmte Lambda-Funktionsversion oder einen bestimmten Alias aufrufen, indem Sie diese Optionen im ARN im `Resource` Feld angeben. In der Lambda-Dokumentation finden Sie Folgendes:

- AWS Lambda -Versioning
- AWS Lambda Aliase

Informationen zur Konfiguration von IAM Berechtigungen bei der Verwendung Step Functions mit anderen AWS Diensten finden Sie unterIAM-Richtlinien für integrierte Dienste.

AWS Elemental MediaConvert Mit Step Functions verwalten

> **Experimentieren Sie mit Step Functions und MediaConvert**
>
> Erfahren Sie, wie Sie die MediaConvert optimierte Integration in einem Workflow einsetzen können, der SMTPE-Farbbalken unbekannter Länge am Anfang eines Videoclips erkennt und entfernt. Lesen Sie den Blogbeitrag vom 12. April 2024: Low-Code-Workflows mit AWS Elemental MediaConvert

Step Functions kann bestimmte AWS Dienste direkt von Amazon States Language (ASL) aus steuern. Weitere Informationen hierzu finden Sie unter Arbeiten mit anderen -Services und Parameter an eine Service-API übergeben.

> **Wie unterscheidet sich die optimierte Integration von der AWS Standard-SDK-Integration**
>
> - Das Ausführen einer Aufgabe (.sync) Integrationsmuster ist verfügbar.
> - Keine Optimierungen Request Response (Antwort anfordern) oder Warten auf einen Callback mit dem Aufgabentoken Integrationsmuster.

Unterstützte MediaConvert APIs:

- `CreateJob`
  - Erforderliche Syntax
  - Unterstützte Parameter:
    - `Role` (Erforderlich)
    - `Settings` (Erforderlich)
    - `CreateJobRequest` (Optional)
  - Antwortsyntax — siehe CreateJobResponse Schema

Im Folgenden wird ein `Task` Status beschrieben, der einen MediaConvert Job weiterleitet und darauf wartet, dass er abgeschlossen ist.

```
{
    "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
        }
    }
}
```

Informationen zur Konfiguration von IAM Berechtigungen bei der Verwendung von Step Functions with finden Sie MediaConvert unter. IAM-Richtlinien für AWS Elemental MediaConvert

> **Die Parameter in Step Functions werden ausgedrückt in PascalCase**
>
> Auch wenn sich die native Service-API in CamelCase befindet, z. B. die API-Aktion`startSyncExecution`, geben Sie Parameter in an PascalCase, z. B.:. `StateMachineArn`

SageMaker Mit Step Functions verwalten

Step Functions kann bestimmte AWS Dienste direkt von Amazon States Language (ASL) aus steuern. Weitere Informationen hierzu finden Sie unter Arbeiten mit anderen -Services und Parameter an eine Service-API übergeben.

Wie unterscheidet sich die optimierte SageMaker Integration von der SageMaker AWS SDK-Integration

- Das Ausführen einer Aufgabe (.sync) Integrationsmuster wird unterstützt.
- Es gibt keine Optimierungen für das Request Response (Antwort anfordern) Integrationsmuster.
- Das Warten auf einen Callback mit dem Aufgabentoken Integrationsmuster wird nicht unterstützt.

Unterstützte SageMaker APIs und Syntax:

- `CreateEndpoint`
  - Erforderliche Syntax
  - Unterstützte Parameter:
    - `EndpointConfigName`
    - `EndpointName`
    - `Tags`
  - Antwortsyntax
- `CreateEndpointConfig`
  - Erforderliche Syntax
  - Unterstützte Parameter:
    - `EndpointConfigName`
    - `KmsKeyId`
    - `ProductionVariants`
    - `Tags`
  - Antwortsyntax
- `CreateHyperParameterTuningJob`

Note

Diese API-Aktion unterstützt das `.sync` Integrationsmuster.

- Erforderliche Syntax
- Unterstützte Parameter:
  - HyperParameterTuningJobConfig
  - HyperParameterTuningJobName
  - Tags
  - TrainingJobDefinition
  - WarmStartConfig
- Antwortsyntax

- CreateLabelingJob

> **Note**
>
> Diese API-Aktion unterstützt das .sync Integrationsmuster.

  - Erforderliche Syntax
  - Unterstützte Parameter:
    - HumanTaskConfig
    - InputConfig
    - LabelAttributeName
    - LabelCategoryConfigS3Uri
    - LabelingJobAlgorithmsConfig
    - LabelingJobName
    - OutputConfig
    - RoleArn
    - StoppingConditions
    - Tags
  - Antwortsyntax
- CreateModel
  - Erforderliche Syntax
  - Unterstützte Parameter:

    - Containers
    - EnableNetworkIsolation
    - ExecutionRoleArn
    - ModelName
    - PrimaryContainer
    - Tags
    - VpcConfig
- CreateProcessingJob

> **Note**
>
> Diese API-Aktion unterstützt das `.sync` Integrationsmuster.

    - Erforderliche Syntax
    - Unterstützte Parameter:
        - AppSpecification
        - Environment
        - ExperimentConfig
        - NetworkConfig
        - ProcessingInputs
        - ProcessingJobName
        - ProcessingOutputConfig
        - ProcessingResources
        - RoleArn
        - StoppingCondition
        - Tags
    - Antwortsyntax
- CreateTrainingJob

> **Note**
>
> Diese API-Aktion unterstützt das `.sync` Integrationsmuster.

- Erforderliche Syntax
- Unterstützte Parameter:
  - `AlgorithmSpecification`
  - `HyperParameters`
  - `InputDataConfig`
  - `OutputDataConfig`
  - `ResourceConfig`
  - `RoleArn`
  - `StoppingCondition`
  - `Tags`
  - `TrainingJobName`
  - `VpcConfig`
- Antwortsyntax

- `CreateTransformJob`

> **Note**
>
> Diese API-Aktion unterstützt das `.sync` Integrationsmuster.

> **Note**
>
> AWS Step Functions erstellt nicht automatisch eine Richtlinie für`CreateTransformJob`. Der erstellten Rolle muss eine Inline-Richtlinie zugewiesen werden. Weitere Informationen finden Sie in diesem Beispiel für eine IAM-Richtlinie:`CreateTrainingJob`.

- Erforderliche Syntax

- Unterstützte Parameter:
  - BatchStrategy
  - Environment
  - MaxConcurrentTransforms
  - MaxPayloadInMB
  - ModelName
  - Tags
  - TransformInput
  - TransformJobName
  - TransformOutput
  - TransformResources
- Antwortsyntax
- UpdateEndpoint
  - Erforderliche Syntax
  - Unterstützte Parameter:
    - EndpointConfigName
    - EndpointName
  - Antwortsyntax

SageMaker Beispiel für einen Transform-Job

Im Folgenden finden Sie einen `Task` Status, der einen SageMaker Amazon-Transformationsauftrag erstellt und den Amazon S3-Standort für `DataSource` und angibt`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 Beispiel für einen Ausbildungsberuf

Im Folgenden finden Sie einen `Task` Status, in dem ein SageMaker Amazon-Schulungsjob erstellt wird.

```
{
   "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 Beispiel für einen Labeling-Job

Im Folgenden finden Sie einen `Task` Status, der einen SageMaker Amazon-Labeling-Job erstellt.

```
{
  "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 Beispiel für einen Verarbeitungsjob

Im Folgenden wird ein `Task` Status beschrieben, der einen SageMaker Amazon-Verarbeitungsauftrag erstellt.

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

Informationen zur Konfiguration von IAM Berechtigungen bei der Verwendung Step Functions mit anderen AWS Diensten finden Sie unterIAM-Richtlinien für integrierte Dienste.

Rufen Sie Amazon SNS mit Step Functions auf

Step Functions kann bestimmte AWS Dienste direkt von Amazon States Language (ASL) aus steuern. Weitere Informationen hierzu finden Sie unter Arbeiten mit anderen -Services und Parameter an eine Service-API übergeben.

Wie sich die optimierte Amazon SNS-Integration von der Amazon SNS AWS SDK-Integration unterscheidet

Es gibt keine Optimierungen für die IntegrationsmusterRequest Response (Antwort anfordern). Warten auf einen Callback mit dem Aufgabentoken

Unterstützte Amazon SNS SNS-APIs:

Note

In Step Functions gibt es ein Kontingent für die maximale Eingabe- oder Ergebnisdatengröße für eine Aufgabe. Dadurch sind Sie auf 256 KB an Daten als UTF-8-kodierte Zeichenfolge beschränkt, wenn Sie Daten an einen anderen Dienst senden oder von einem anderen Dienst empfangen. Siehe Kontingente im Zusammenhang mit der Ausführung von Zustandsmaschinen.

- `Publish`
  - Erforderliche Syntax
  - Unterstützte Parameter
    - `Message`
    - `MessageAttributes`
    - `MessageStructure`
    - `PhoneNumber`
    - `Subject`
    - `TargetArn`
    - `TopicArn`
  - Antwortsyntax

> **Parameter in werden ausgedrückt in Step Functions PascalCase**
>
> Auch wenn sich die native Service-API in CamelCase befindet, z. B. die API-Aktion`startSyncExecution`, geben Sie Parameter in an PascalCase, z. B.:. `StateMachineArn`

Im Folgenden finden Sie einen `Task` Status, der zu einem Amazon Simple Notification Service (Amazon SNS) -Thema veröffentlicht.

```
{
 "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
    }
  }
}
```

Übergeben dynamischer Werte. Sie können das obige Beispiel ändern, um ein Attribut aus dieser JSON-Payload dynamisch zu übergeben:

```
{
"input": {
  "message": "Hello world"
```

```
},
"SNSDetails": {
  "attribute1": "some value",
  "attribute2": "some other value",
}
}
```

Hängen Sie das `.$` an das Feld an`StringValue`:

```
"MessageAttributes": {
  "my_attribute_no_1": {
      "DataType": "String",
      "StringValue.$": "$.SNSDetails.attribute1"
  },
  "my_attribute_no_2": {
      "DataType": "String",
      "StringValue.$": "$.SNSDetails.attribute2"
  }
```

Im Folgenden wird ein `Task` Status beschrieben, der in einem Amazon SNS SNS-Thema veröffentlicht und dann darauf wartet, dass das Task-Token zurückgegeben wird. Siehe Warten auf einen Callback mit dem Aufgabentoken.

```
{
   "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
      }
   }
}
```

Informationen zur Konfiguration von IAM Berechtigungen bei der Verwendung Step Functions mit anderen AWS Diensten finden Sie unter. IAM-Richtlinien für integrierte Dienste

Rufen Sie Amazon SQS mit Step Functions auf

Step Functions kann bestimmte AWS Dienste direkt von Amazon States Language (ASL) aus steuern. Weitere Informationen hierzu finden Sie unter Arbeiten mit anderen -Services und Parameter an eine Service-API übergeben.

> Wie sich die optimierte Amazon SQS-Integration von der Amazon SQS AWS SDK-Integration unterscheidet
>
> Es gibt keine Optimierungen für die IntegrationsmusterRequest Response (Antwort anfordern). Warten auf einen Callback mit dem Aufgabentoken

Unterstützte Amazon SQS SQS-APIs:

> Note
>
> In Step Functions gibt es ein Kontingent für die maximale Eingabe- oder Ergebnisdatengröße für eine Aufgabe. Dadurch sind Sie auf 256 KB an Daten als UTF-8-kodierte Zeichenfolge beschränkt, wenn Sie Daten an einen anderen Dienst senden oder von einem anderen Dienst empfangen. Siehe Kontingente im Zusammenhang mit der Ausführung von Zustandsmaschinen.

- `SendMessage`

  Unterstützte Parameter:

  - `DelaySeconds`
  - `MessageAttribute`
  - `MessageBody`
  - `MessageDeduplicationId`
  - `MessageGroupId`
  - `QueueUrl`
- `Response syntax`

> **Parameter in werden ausgedrückt in Step Functions PascalCase**
>
> Auch wenn sich die native Service-API in CamelCase befindet, z. B. die API-Aktion`startSyncExecution`, geben Sie Parameter in an PascalCase, z. B.:. `StateMachineArn`

Im Folgenden wird ein `Task` Status beschrieben, der eine Amazon Simple Queue Service (Amazon SQS) -Nachricht sendet.

```
{
 "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
    }
  }
}
```

Im Folgenden wird ein `Task` Status beschrieben, der in einer Amazon SQS SQS-Warteschlange veröffentlicht und dann darauf wartet, dass das Task-Token zurückgegeben wird. Siehe Warten auf einen Callback mit dem Aufgabentoken.

```
{
   "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
      }
   }
}
```

Weitere Informationen zum Empfangen von Nachrichten in Amazon SQS finden Sie unter Empfangen und Löschen Ihrer Nachricht im Amazon Simple Queue Service Developer Guide.

Informationen zur Konfiguration von IAM Berechtigungen bei der Verwendung Step Functions mit anderen AWS Diensten finden Sie unterIAM-Richtlinien für integrierte Dienste.

Managen Sie AWS Step Functions Ausführungen als integrierten Service

Step Functions lässt sich in seine eigene API als Service-Integration integrieren. Auf diese Weise kann Step Functions eine neue Ausführung einer Zustandsmaschine direkt aus dem Aufgabenstatus einer laufenden Ausführung starten. Verwenden Sie beim Erstellen neuer Workflows verschachtelte Workflow-Ausführungen, um die Komplexität Ihrer Haupt-Workflows zu reduzieren und gängige Prozesse wiederzuverwenden.

Wie sich die Optimized Step Functions-Integration von der Step Functions AWS SDK-Integration unterscheidet

- Das Ausführen einer Aufgabe (.sync) Integrationsmuster ist verfügbar.

Beachten Sie, dass es keine Optimierungen für die Request Response (Antwort anfordern) Warten auf einen Callback mit dem Aufgabentoken Integrationsmuster gibt.

Weitere Informationen finden Sie hier:

- Starten von Ausführungen über eine Aufgabe
- Arbeiten mit anderen -Services
- Parameter an eine Service-API übergeben

Unterstützte Step Functions Functions-APIs und Syntax:

- `StartExecution`
  - Anforderungssyntax
  - Unterstützte Parameter
    - `Input`
    - `Name`
    - `StateMachineArn`
  - Antwortsyntax

Das folgende Beispiel enthält einen `Task`-Status, der die Ausführung eines anderen Zustandsautomaten startet und wartet, bis sie abgeschlossen ist.

```
{
   "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
}
```

Das folgende Beispiel enthält einen `Task`-Status, der die Ausführung eines anderen Zustandsautomaten startet.

```
{
   "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
}
```

Das Folgende enthält einen `Task`-Zustand zur Implementierung des Callback- Service-Integrationsmusters.

```
{
   "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
}
```

Um eine verschachtelte Workflow-Ausführung mit der übergeordneten Ausführung zu verknüpfen, von der sie gestartet wurde, übergeben Sie einen speziell benannten Parameter, der die Ausführungs-ID enthält, die aus dem Context-Objekt abgerufen wurde. Wenn Sie eine verschachtelte Ausführung starten, verwenden Sie einen Parameter namens `AWS_STEP_FUNCTIONS_STARTED_BY_EXECUTION_ID`. Übergeben Sie die Ausführungs-ID, indem Sie `.$` an den Parameternamen anhängen und mit `$$.Execution.Id` auf die ID im Context-Objekt mit verweisen. Weitere Informationen finden Sie unter Zugriff auf das Kontext-Objekt.

```
{
   "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
}
```

Verschachtelte Zustandsautomaten geben Folgendes zurück:

Ressource	Output
Startexecution.sync	String
Startexecution.sync:2	JSON

Beide warten, bis der verschachtelte Zustandsautomat fertig ist, aber sie geben verschiedene `Output`-Formate zurück. Wenn Sie beispielsweise eine Lambda-Funktion erstellen, die das Objekt zurückgibt`{ "MyKey": "MyValue" }`, würden Sie die folgenden Antworten erhalten:

Für startExecution.sync:

```
{
   <other fields>
   "Output": "{ \"MyKey\": \"MyValue\" }"
}
```

Für startExecution.sync:2:

```
{
   <other fields>
   "Output": {
      "MyKey": "MyValue"
   }
}
```

Konfiguration von IAM-Berechtigungen für verschachtelte Zustandsmaschinen

Ein übergeordneter Zustandsmaschine bestimmt anhand von Abfragen und Ereignissen, ob ein untergeordneter Zustandsmaschine die Ausführung abgeschlossen hat. Für Abfragen ist eine Genehmigung erforderlich, `states:DescribeExecution` während für Ereignisse, die EventBridge an Step Functions gesendet werden, Berechtigungen für `events:PutTargetsevents:PutRule`, und `events:DescribeRule` erforderlich sind. Wenn diese Berechtigungen in Ihrer IAM-Rolle fehlen, kann es zu einer Verzögerung kommen, bis ein übergeordneter Zustandsmaschine erfährt, dass die Ausführung des untergeordneten Zustandsmaschinen abgeschlossen ist.

Verwenden Sie für einen Zustandsmaschine, der die Ausführung eines einzelnen verschachtelten Workflows erfordert`StartExecution`, eine IAM-Richtlinie, die die Berechtigungen auf diesen Zustandsmaschine beschränkt.

Weitere Informationen finden Sie unter IAM-Berechtigungen für Step Functions.

Rufen Sie APIs von Drittanbietern auf

Eine HTTP-Aufgabe ist eine Art von Status der Aufgabe Status, mit dem Sie jede öffentliche API eines Drittanbieters wie Salesforce und Stripe in Ihren Workflows aufrufen können. Um eine Drittanbieter-API aufzurufen, verwenden Sie den Aufgabenstatus mit der `arn:aws:states:::http:invoke` Ressource. Geben Sie dann die Konfigurationsdetails des API-Endpunkts an, z. B. die API-URL, die Methode, die Sie verwenden möchten, und Authentifizierungsdetails.

Wenn Sie Workflow Studio verwenden, um Ihren Zustandsmaschine zu erstellen, der eine HTTP-Task enthält, generiert Workflow Studio automatisch eine Ausführungsrolle mit IAM Richtlinien für die HTTP-Task. Weitere Informationen finden Sie unter Rolle zum Testen von HTTP-Aufgaben in Workflow Studio.

Themen

- Definition der HTTP-Aufgabe
- HTTP-Aufgabenfelder
- Authentifizierung für eine HTTP-Aufgabe
- EventBridgeVerbindungs- und HTTP-Aufgabendefinitionsdaten zusammenführen
- URL-Kodierung auf den Anfragetext anwenden
- IAM-Berechtigungen zum Ausführen einer HTTP-Aufgabe

- Beispiel für eine HTTP-Aufgabe
- Testen einer HTTP-Aufgabe
- Antworten auf HTTP-Aufgaben werden nicht unterstützt

Definition der HTTP-Aufgabe

Die ASL-Definition stellt eine HTTP-Aufgabe mit `http:invoke` Ressource dar. Die folgende HTTP-Aufgabendefinition ruft eine Stripe-API auf, die eine Liste aller Kunden zurückgibt.

```
"Call third-party API": {
  "Type": "Task",
  "Resource": "arn:aws:states:::http:invoke",
  "Parameters": {
    "ApiEndpoint": "https://api.stripe.com/v1/customers",
    "Authentication": {
      "ConnectionArn": "arn:aws:events:us-east-2:123456789012:connection/
Stripe/81210c42-8af1-456b-9c4a-6ff02fc664ac"
    },
    "Method": "GET"
  },
  "End": true
}
```

HTTP-Aufgabenfelder

Eine HTTP-Aufgabe umfasst die folgenden Felder in ihrer Definition.

`Resource` (Erforderlich)

Um einen Aufgabentyp anzugeben, geben Sie dessen ARN in das `Resource` Feld ein. Für eine HTTP-Aufgabe geben Sie das `Resource` Feld wie folgt an.

```
"Resource": "arn:aws:states:::http:invoke"
```

`Parameters` (Erforderlich)

Enthält die `ConnectionArn` Felder `ApiEndpoint``Method`, und, die Informationen über die Drittanbieter-API enthalten, die Sie aufrufen möchten. `Parameters`enthält auch optionale Felder wie `Headers` und`QueryParameters`.

Sie können eine Kombination aus statischem JSON und JsonPathSyntax wie `Parameters` im `Parameters` Feld angeben. Weitere Informationen finden Sie unter Parameter an eine Service-API übergeben.

`ApiEndpoint`(Erforderlich)

Gibt die URL der Drittanbieter-API an, die Sie aufrufen möchten. Verwenden Sie das Feld, um Abfrageparameter an die URL anzuhängen. `QueryParameters` Das folgende Beispiel zeigt, wie Sie eine Stripe-API aufrufen können, um die Liste aller Kunden abzurufen.

```
"ApiEndpoint":"https://api.stripe.com/v1/customers"
```

Sie können mithilfe der JsonPathSyntax auch einen Referenzpfad angeben, um den JSON-Knoten auszuwählen, der die API-URL eines Drittanbieters enthält. Angenommen, Sie möchten eine der APIs von Stripe mit einer bestimmten Kunden-ID aufrufen. Stellen Sie sich vor, Sie haben die folgende Statuseingabe bereitgestellt.

```
{
    "customer_id": "1234567890",
    "name": "John Doe"
}
```

Um die Details dieser Kunden-ID mithilfe einer Stripe-API abzurufen, geben Sie die `ApiEndpoint` wie im folgenden Beispiel gezeigt an. In diesem Beispiel werden eine systeminterne Funktion und ein Referenzpfad verwendet.

```
"ApiEndpoint.$":"States.Format('https://api.stripe.com/v1/customers/{}',
 $.customer_id)"
```

Zur Laufzeit löst Step Functions den Wert von `ApiEndpoint` wie folgt auf.

```
https://api.stripe.com/v1/customers/1234567890
```

`Method`(Erforderlich)

Gibt die HTTP-Methode an, die Sie für den Aufruf einer Drittanbieter-API verwenden möchten. Sie können in Ihrer HTTP-Aufgabe eine der folgenden Methoden angeben: GET, POST, PUT, DELETE, PATCH, OPTIONS oder HEAD.

Um beispielsweise die GET-Methode zu verwenden, geben Sie das `Method` Feld wie folgt an.

```
"Method": "GET"
```

Sie können auch einen Referenzpfad verwenden, um die Methode zur Laufzeit anzugeben. z. B. **`"Method.$": "$.myHTTPMethod"`**.

`Authentication`(Erforderlich)

Enthält das `ConnectionArn` Feld, das angibt, wie ein API-Aufruf eines Drittanbieters authentifiziert werden soll. Step Functionsunterstützt die Authentifizierung für ein bestimmtes Objekt `ApiEndpoint` mithilfe der Verbindungsressource vonAmazon EventBridge.

`ConnectionArn`(Erforderlich)

Gibt den EventBridge Verbindungs-ARN an.

Ein HTTP-Task erfordert eine EventBridge Verbindung, die die Authentifizierungsdaten eines API-Anbieters sicher verwaltet. Eine Verbindung gibt den Autorisierungstyp und die Anmeldeinformationen an, die für die Autorisierung einer Drittanbieter-API verwendet werden sollen. Durch die Verwendung einer Verbindung können Sie verhindern, dass Geheimnisse wie API-Schlüssel fest in Ihre Zustandsmaschinen-Definition integriert werden. In einer Verbindung können Sie auch `RequestBody` Parameter `HeadersQueryParameters`, und angeben.

Wenn Sie eine EventBridge Verbindung herstellen, geben Sie Ihre Authentifizierungsdetails an. Weitere Informationen darüber, wie die Authentifizierung für eine HTTP-Aufgabe funktioniert, finden Sie unterAuthentifizierung für eine HTTP-Aufgabe.

Das folgende Beispiel zeigt, wie Sie das `Authentication` Feld in Ihrer HTTP-Aufgabendefinition angeben können.

```
"Authentication": {
  "ConnectionArn": "arn:aws:events:us-east-2:123456789012:connection/
Stripe/81210c42-8af1-456b-9c4a-6ff02fc664ac"
}
```

`Headers` (Optional)

Stellt zusätzlichen Kontext und Metadaten für den API-Endpunkt bereit. Sie können Header als Zeichenfolge oder JSON-Array angeben.

Sie können Header in der EventBridge Verbindung und das `Headers` Feld in einer HTTP-Task angeben. Wir empfehlen, dass Sie in dem `Headers` Feld keine Authentifizierungsdetails

für Ihre API-Anbieter angeben. Wir empfehlen Ihnen, diese Details in Ihre EventBridge Verbindung aufzunehmen.

Step Functionsfügt die Header, die Sie in der EventBridge Verbindung angeben, zu den Headern hinzu, die Sie in der HTTP-Aufgabendefinition angeben. Wenn dieselben Header-Schlüssel in der Definition und Verbindung vorhanden sind, Step Functions verwendet die entsprechenden Werte, die in der EventBridge Verbindung für diese Header angegeben wurden. Weitere Hinweise zur Durchführung der Step Functions Datenzusammenführung finden Sie unter. EventBridgeVerbindungs- und HTTP-Aufgabendefinitionsdaten zusammenführen

Das folgende Beispiel spezifiziert einen Header, der in einen API-Aufruf eines Drittanbieters aufgenommen wird:`content-type`.

```
"Headers": {
  "content-type": "application/json"
}
```

Sie können auch einen Referenzpfad verwenden, um die Header zur Laufzeit anzugeben. z. B. **`"Headers.$": "$.myHTTPHeaders"`**.

Step Functionslegt die `User-Agent` `Host` HeaderRange, und fest. Step Functionslegt den Wert des `Host` Headers basierend auf der API fest, die Sie aufrufen. Im Folgenden finden Sie ein Beispiel für diese Header.

```
User-Agent: Amazon|StepFunctions|HttpInvoke|us-east-1,
Range: bytes=0-262144,
Host: api.stripe.com
```

Sie können die folgenden Header nicht in Ihrer HTTP-Aufgabendefinition verwenden. Wenn Sie diese Header verwenden, schlägt die HTTP-Aufgabe mit dem `States.Runtime` Fehler fehl.

- A-IM
- Accept-Charset
- Accept-Datetime
- Accept-Encoding
- Cache-Control

- Verbindung
- Content-Encoding
- Inhalt-MD5
- Datum
- Expect
- Forwarded
- Aus
- Host
- HTTP2-Settings
- If-Match
- If-Modified-Since
- If-None-Match
- If-Range
- If-Unmodified-Since
- Max-Forwards
- Urspung
- Pragma
- Proxy-Authorization
- Referer
- Server
- TE
- Trailer
- Transfer-Encoding
- Upgrade
- Via
- Warnung
- x-weitergeleiteten-*
- x-amz-*
- x-amzn-*

`QueryParameters` (Optional)

Fügt Schlüssel-Wert-Paare am Ende einer API-URL ein. Sie können Abfrageparameter als Zeichenfolge, JSON-Array oder JSON-Objekt angeben. Step FunctionsBeim Aufrufen einer Drittanbieter-API werden Abfrageparameter automatisch URL-kodiert.

Angenommen, Sie möchten die Stripe-API aufrufen, um nach Kunden zu suchen, die ihre Transaktionen in US-Dollar (USD) tätigen. Stellen Sie sich vor, Sie haben Folgendes `QueryParameters` als Statuseingabe angegeben.

```
"QueryParameters": {
  "currency": "usd"
}
```

Zur Laufzeit hängt Step Functions die wie folgt `QueryParameters` an die API-URL an.

```
https://api.stripe.com/v1/customers/search?currency=usd
```

Sie können auch einen Referenzpfad verwenden, um die Abfrageparameter zur Laufzeit anzugeben. z. B. **`"QueryParameters.$": "$.myQueryParameters"`**.

Wenn Sie in Ihrer EventBridge Verbindung Abfrageparameter angegeben haben, Step Functions fügt diese Abfrageparameter den Abfrageparametern hinzu, die Sie in der HTTP-Aufgabendefinition angeben. Wenn dieselben Abfrageparameterschlüssel in der Definition und Verbindung vorhanden sind, Step Functions verwendet die entsprechenden Werte, die in der EventBridge Verbindung für diese Header angegeben wurden. Weitere Hinweise zur Durchführung der Step Functions Datenzusammenführung finden Sie unter. EventBridgeVerbindungs- und HTTP-Aufgabendefinitionsdaten zusammenführen

`Transform` (Optional)

Enthält die `RequestEncodingOptions` Felder `RequestBodyEncoding` und. Step FunctionsSendet den Anforderungstext standardmäßig als JSON-Daten an einen API-Endpunkt.

Wenn Ihr API-Anbieter `form-urlencoded` Anfragetexte akzeptiert, verwenden Sie das `Transform` Feld, um die URL-Kodierung für die Anfragetexte anzugeben. Sie müssen den `content-type` Header auch als angeben. `application/x-www-form-urlencoded` Step Functionsdann wird Ihr Anfragetext automatisch URL-kodiert.

`RequestBodyEncoding`

Gibt die URL-Kodierung Ihres Anfragetexts an. Sie können einen dieser Werte angeben: `NONE` oder. `URL_ENCODED`

- `NONE`— Der Hauptteil der HTTP-Anfrage wird das serialisierte JSON des `RequestBody` Felds sein. Dies ist der Standardwert.
- `URL_ENCODED`— Der Hauptteil der HTTP-Anfrage besteht aus den URL-codierten Formulardaten des Felds. `RequestBody`

`RequestEncodingOptions`

Bestimmt die Kodierungsoption, die für Arrays in Ihrem Anfragetext verwendet werden soll, wenn Sie diese Option festlegen. `RequestBodyEncoding` `URL_ENCODED`

Step Functionsunterstützt die folgenden Array-Kodierungsoptionen. Weitere Informationen zu diesen Optionen und ihren Beispielen finden Sie unterURL-Kodierung auf den Anfragetext anwenden.

- `INDICES`— Kodiert Arrays mithilfe des Indexwerts von Array-Elementen. Step FunctionsVerwendet standardmäßig diese Kodierungsoption.
- `REPEAT`— Wiederholt einen Schlüssel für jedes Element in einem Array.
- `COMMAS`— Kodiert alle Werte in einem Schlüssel als kommagetrennte Werteliste.
- `BRACKETS`— Wiederholt einen Schlüssel für jedes Element in einem Array und hängt eine Klammer [] an den Schlüssel an, um anzuzeigen, dass es sich um ein Array handelt.

Im folgenden Beispiel wird die URL-Kodierung für die Hauptdaten der Anfrage festgelegt. Außerdem wird angegeben, dass die `COMMAS` Kodierungsoption für Arrays im Anforderungstext verwendet werden soll.

```
"Transform": {
  "RequestBodyEncoding": "URL_ENCODED",
  "RequestEncodingOptions": {
    "ArrayFormat": "COMMAS"
  }
}
```

`RequestBody` (Optional)

Akzeptiert JSON-Daten, die Sie in der Statuseingabe angeben. `RequestBody`In können Sie eine Kombination aus statischem JSON und JsonPathSyntax angeben. Angenommen, Sie geben die folgende Statuseingabe an:

```
{
    "CardNumber": "1234567890",
    "ExpiryDate": "09/25"
}
```

Um diese Werte von `CardNumber` und zur Laufzeit `ExpiryDate` in Ihrem Anfragetext zu verwenden, können Sie die folgenden JSON-Daten in Ihrem Anfragetext angeben.

```
"RequestBody": {
  "Card": {
    "Number.$": "$.CardNumber",
    "Expiry.$": "$.ExpiryDate",
    "Name": "John Doe",
    "Address": "123 Any Street, Any Town, USA"
  }
}
```

Wenn die Drittanbieter-API, die Sie aufrufen möchten, `form-urlencoded` Anfragetexte benötigt, müssen Sie die URL-Kodierung für Ihre Anfragetextdaten angeben. Weitere Informationen finden Sie unter URL-Kodierung auf den Anfragetext anwenden.

Authentifizierung für eine HTTP-Aufgabe

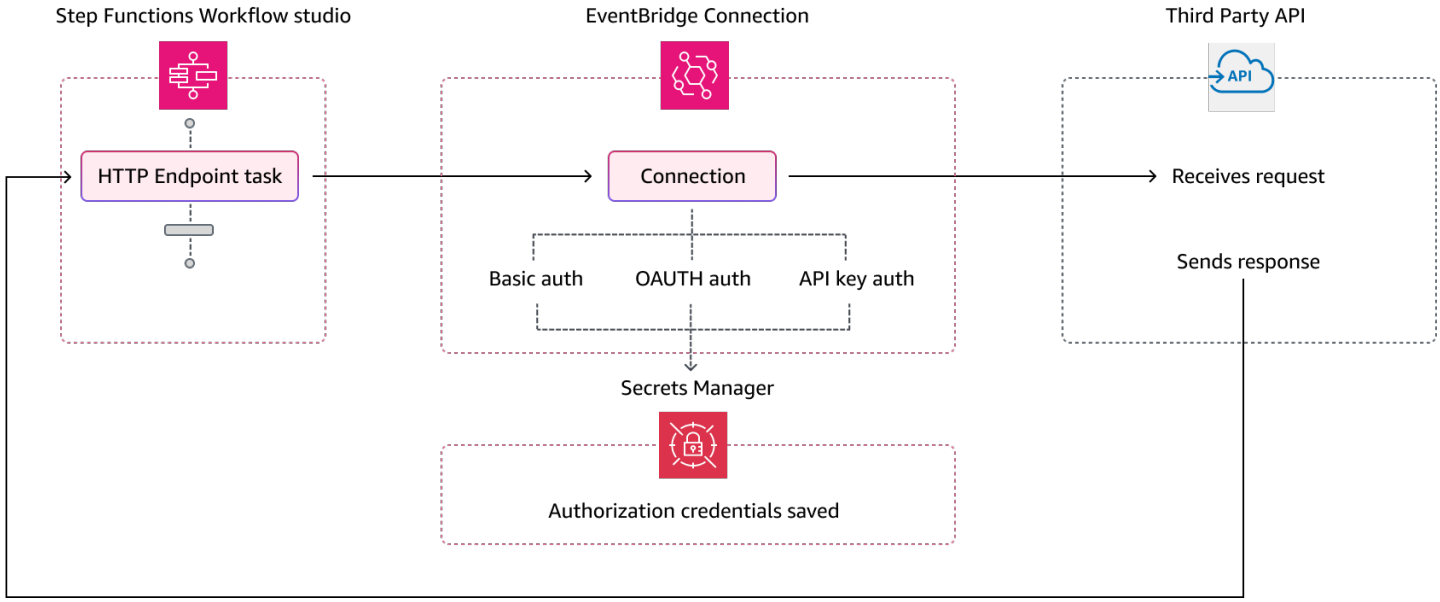
Ein HTTP-Task erfordert eine EventBridge Verbindung, die die Authentifizierungsdaten eines API-Anbieters sicher verwaltet. Eine Verbindung gibt den Autorisierungstyp und die Anmeldeinformationen an, die für die Autorisierung einer Drittanbieter-API verwendet werden sollen. Durch die Verwendung einer Verbindung können Sie verhindern, dass Geheimnisse wie API-Schlüssel fest in Ihre Zustandsmaschinen-Definition integriert werden. Eine EventBridge Verbindung unterstützt die Autorisierungsschemata Basic, OAuth und API Key.

Wenn Sie eine EventBridge Verbindung herstellen, geben Sie Ihre Authentifizierungsdetails an. Sie können auch die Header-, Text- und Abfrageparameter angeben, die für die Autorisierung mit einer

API erforderlich sind. Sie müssen den Verbindungs-ARN in jede HTTP-Aufgabe aufnehmen, die eine Drittanbieter-API aufruft.

Wenn Sie eine Verbindung herstellen und Autorisierungsparameter hinzufügen, EventBridge erstellt ein Secret in AWS Secrets Manager. In diesem Secret EventBridge werden die Verbindungs- und Autorisierungsparameter in verschlüsselter Form gespeichert. Um erfolgreich eine Verbindung herzustellen oder zu aktualisieren, müssen Sie eine Verbindung verwenden, AWS-Konto die über die Berechtigung zur Verwendung von Secrets Manager verfügt. Weitere Informationen zu den IAM Berechtigungen, die Ihr State-Machine für den Zugriff auf eine EventBridge Verbindung benötigt, finden Sie unterIAM-Berechtigungen zum Ausführen einer HTTP-Aufgabe.

Die folgende Abbildung zeigt, wie die Authentifizierung für API-Aufrufe von Drittanbietern über eine EventBridge Verbindung Step Functions gehandhabt wird.



EventBridgeVerbindungs- und HTTP-Aufgabendefinitionsdaten zusammenführen

Wenn Sie eine HTTP-Task aufrufen, können Sie Daten in Ihrer EventBridge Verbindung und Ihrer HTTP-Aufgabendefinition angeben. Zu diesen Daten gehören `HeadersQueryParameters`, und `RequestBody` Parameter. Vor dem Aufrufen einer Drittanbieter-API führt Step Functions den Hauptteil der Anfrage in allen Fällen mit den Parametern des Verbindungstexts zusammen, es sei denn, Ihr Anfragetext ist eine Zeichenfolge und die Parameter für den Verbindungstext sind nicht leer. In diesem Fall schlägt der HTTP-Task mit dem `States.Runtime` Fehler fehl.

Wenn in der HTTP-Task-Definition und der EventBridge Verbindung doppelte Schlüssel angegeben sind, werden die Werte in der HTTP-Task mit den Werten in der Verbindung Step Functions überschrieben.

In der folgenden Liste wird beschrieben, wie Daten Step Functions zusammengeführt werden, bevor eine Drittanbieter-API aufgerufen wird:

- Header — Step Functions Fügt alle Header, die Sie in der Verbindung angegeben haben, zu den Headern im `Headers` Feld der HTTP-Aufgabe hinzu. Wenn ein Konflikt zwischen den Header-Schlüsseln besteht, Step Functions verwendet die in der Verbindung angegebenen Werte für diese Header. Wenn Sie beispielsweise den `content-type` Header sowohl in der HTTP-Aufgabendefinition als auch in der EventBridge Verbindung angegeben haben, Step Functions verwendet es den in der Verbindung angegebenen `content-type` Header-Wert.
- Abfrageparameter — Step Functions Fügt alle Abfrageparameter, die Sie in der Verbindung angegeben haben, zu den Abfrageparametern im `QueryParameters` Feld des HTTP-Tasks hinzu. Wenn ein Konflikt zwischen den Abfrageparameterschlüsseln besteht, Step Functions verwendet die in der Verbindung angegebenen Werte für diese Abfrageparameter. Wenn Sie beispielsweise den `maxItems` Abfrageparameter sowohl in der HTTP-Aufgabendefinition als auch in der EventBridge Verbindung angegeben haben, Step Functions wird der in der Verbindung angegebene `maxItems` Abfrageparameterwert verwendet.
- Body-Parameter
  - Step Functionsfügt alle in der Verbindung angegebenen Werte des Anforderungstexts zum Anforderungstext im `RequestBody` Feld der HTTP-Task hinzu. Wenn ein Konflikt zwischen den Schlüsseln des Anforderungstexts besteht, Step Functions verwendet die in der Verbindung angegebenen Werte für den Anforderungstext. Nehmen wir beispielsweise an, dass Sie sowohl in der `RequestBody` HTTP-Aufgabendefinition als auch in der EventBridge Verbindung ein `Mode` Feld angegeben haben. Step Functionsverwendet den `Mode` Feldwert, den Sie in der Verbindung angegeben haben.
  - Wenn Sie den Anforderungstext als Zeichenfolge statt als JSON-Objekt angeben und die EventBridge Verbindung auch den Anforderungstext enthält, Step Functions kann der an diesen beiden Stellen angegebene Anforderungstext nicht zusammengeführt werden. Die HTTP-Aufgabe schlägt mit dem `States.Runtime` Fehler fehl.

  Step Functionswendet alle Transformationen an und serialisiert den Anforderungstext, nachdem die Zusammenführung des Anforderungstexts abgeschlossen ist.

Im folgenden Beispiel werden die `RequestBody` Felder `HeadersQueryParameters`, und sowohl in der HTTP-Aufgabe als auch in der HTTP-Verbindung festgelegt. EventBridge

Definition der HTTP-Aufgabe

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

EventBridgeVerbindung

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

In diesem Beispiel werden doppelte Schlüssel in der HTTP-Aufgabe und der EventBridge Verbindung angegeben. Step FunctionsÜberschreibt daher die Werte in der HTTP-Task mit den Werten in der Verbindung. Der folgende Codeausschnitt zeigt die HTTP-Anfrage, die Step Functions an die Drittanbieter-API gesendet wird.

```
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"}
```

URL-Kodierung auf den Anfragetext anwenden

Step FunctionsSendet den Anforderungstext standardmäßig als JSON-Daten an einen API-Endpunkt. Wenn Ihr API-Drittanbieter `form-urlencoded` Anfragetexte benötigt, müssen Sie die URL-Kodierung für die Anfragetexte angeben. Step Functionsdann wird der Anfragetext automatisch auf der Grundlage der von Ihnen ausgewählten URL-Kodierungsoption URL-codiert.

Sie geben die URL-Kodierung mithilfe des Felds an. `Transform` Dieses Feld enthält das `RequestBodyEncoding` Feld, das angibt, ob Sie die URL-Kodierung für Ihre Anfragetexte anwenden möchten oder nicht. Wenn Sie das `RequestBodyEncoding` Feld angeben, wird Ihr JSON-Anforderungstext in den Anforderungstext Step Functions `form-urlencoded` konvertiert, bevor die Drittanbieter-API aufgerufen wird. Sie müssen auch den `content-type` Header angeben, `application/x-www-form-urlencoded` da APIs, die URL-kodierte Daten akzeptieren, den Header erwarten. `content-type`

Step FunctionsStellt die folgenden Array-Kodierungsoptionen zur Verfügung, um Arrays in Ihrem Anfragetext zu kodieren.

- `INDICES`— Wiederholt einen Schlüssel für jedes Element in einem Array und hängt eine Klammer [] an den Schlüssel an, um anzuzeigen, dass es sich um ein Array handelt. Diese Klammer enthält den Index des Array-Elements. Durch Hinzufügen des Indexes können Sie die Reihenfolge der Array-Elemente festlegen. Step FunctionsVerwendet standardmäßig diese Kodierungsoption.

  Zum Beispiel, wenn Ihr Anfragetext das folgende Array enthält.

  ```
  {"array": ["a","b","c","d"]}
  ```

  Step Functionskodiert dieses Array in die folgende Zeichenfolge.

  ```
  array[0]=a&array[1]=b&array[2]=c&array[3]=d
  ```

- `REPEAT`— Wiederholt einen Schlüssel für jedes Element in einem Array.

  Zum Beispiel, wenn Ihr Anfragetext das folgende Array enthält.

  ```
  {"array": ["a","b","c","d"]}
  ```

  Step Functionskodiert dieses Array in die folgende Zeichenfolge.

```
array=a&array=b&array=c&array=d
```

- `COMMAS`— Kodiert alle Werte in einem Schlüssel als kommagetrennte Werteliste.

  Zum Beispiel, wenn Ihr Anfragetext das folgende Array enthält.

  ```
  {"array": ["a","b","c","d"]}
  ```

  Step Functionskodiert dieses Array in die folgende Zeichenfolge.

  ```
  array=a,b,c,d
  ```

- `BRACKETS`— Wiederholt einen Schlüssel für jedes Element in einem Array und hängt eine Klammer [] an den Schlüssel an, um anzuzeigen, dass es sich um ein Array handelt.

  Zum Beispiel, wenn Ihr Anfragetext das folgende Array enthält.

  ```
  {"array": ["a","b","c","d"]}
  ```

  Step Functionskodiert dieses Array in die folgende Zeichenfolge.

  ```
  array[]=a&array[]=b&array[]=c&array[]=d
  ```

IAM-Berechtigungen zum Ausführen einer HTTP-Aufgabe

Ihre State-Machine-Ausführungsrolle muss über die `secretsmanager:DescribeSecret` Berechtigungen`states:InvokeHTTPEndpoint`,`events:RetrieveConnectionCredentials`, und verfügen`secretsmanager:GetSecretValue`, damit eine HTTP-Task eine Drittanbieter-API aufrufen kann. Das folgende Beispiel für eine IAM-Richtlinie gewährt Ihrer State-Machine-Rolle die geringsten Rechte, die für den Aufruf von Stripe-APIs erforderlich sind. Diese IAM-Richtlinie gewährt der State-Machine-Rolle auch die Erlaubnis, auf eine bestimmte EventBridge Verbindung zuzugreifen, einschließlich des Geheimnisses für diese Verbindung, das in Secrets Manager gespeichert ist.

```
{
    "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/*"
        }
    ]
}
```

Beispiel für eine HTTP-Aufgabe

Die folgende State-Machine-Definition zeigt einen HTTP-Task`Headers`, der die `RequestBody` Parameter `QueryParametersTransform`,, und enthält. Der HTTP-Task ruft eine Stripe-API, https://api.stripe.com/v1/invoices, auf, um eine Rechnung zu generieren. Die HTTP-Aufgabe spezifiziert mithilfe der Kodierungsoption auch die URL-Kodierung für den `INDICES` Anfragetext.

Stellen Sie sicher, dass Sie eine EventBridge Verbindung hergestellt haben. Das folgende Beispiel zeigt eine Verbindung, die mit dem Basic-Authentifizierungstyp erstellt wurde.

```
{
    "Type": "BASIC",
    "AuthParameters": {
        "BasicAuthParameters": {
            "Password": "myPassword",
            "Username": "myUsername"
         },
    }
}
```

Denken Sie daran, den *kursiv geschriebenen* Text durch Ihre ressourcenspezifischen Informationen zu ersetzen.

```
{
  "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
    }
  }
}
```

Um diesen Zustandsmaschine auszuführen, geben Sie die Kunden-ID als Eingabe ein, wie im folgenden Beispiel gezeigt:

```
{
    "customer_id": "1234567890"
}
```

Das folgende Beispiel zeigt die HTTP-Anfrage, die Step Functions an die Stripe-API gesendet wird.

```
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
```

Testen einer HTTP-Aufgabe

Sie können die TestStateAPI über die Konsole, das SDK oder das verwenden, AWS CLI um eine HTTP-Task zu testen. Das folgende Verfahren beschreibt, wie die TestState API in der Step Functions Konsole verwendet wird. Sie können die API-Anfrage, die Antwort und die Authentifizierungsdetails iterativ testen, bis Ihre HTTP-Aufgabe wie erwartet funktioniert.

Testen Sie den Status einer HTTP-Task in der Konsole Step Functions

1. Öffnen Sie die Step Functions Functions-Konsole.
2. Wählen Sie Create State Machine, um mit der Erstellung einer State-Machine zu beginnen, oder wählen Sie eine bestehende State-Machine aus, die eine HTTP-Task enthält.

   Lesen Sie Schritt 4, wenn Sie die Aufgabe in einer vorhandenen Zustandsmaschine testen.
3. Konfigurieren Sie in Workflow Studio eine HTTP-Aufgabe visuell. Entwurfsmodus Oder wählen Sie den Codemodus, um die State-Machine-Definition aus Ihrer lokalen Entwicklungsumgebung zu kopieren und einzufügen.
4. Wählen Sie im Entwurfsmodus im Inspector Bedienfeld von Workflow Studio die Option Teststatus aus.
5. Gehen Sie im Dialogfeld Teststatus wie folgt vor:

   a. Wählen Sie unter Ausführungsrolle eine Ausführungsrolle aus, um den Status zu testen. Wenn Sie nicht über eine Rolle mit ausreichenden Berechtigungen für eine HTTP-Aufgabe verfügen, finden Sie weitere Informationen unter Rolle zum Testen von HTTP-Aufgaben in Workflow Studio So erstellen Sie eine Rolle.

b. (Optional) Geben Sie alle JSON-Eingaben an, die Ihr ausgewählter Bundesstaat für den Test benötigt.

c. Behalten Sie für Inspection Level die Standardauswahl INFO bei. Diese Stufe zeigt Ihnen den Status des API-Aufrufs und die Statusausgabe. Dies ist nützlich, um die API-Antwort schnell zu überprüfen.

d. Wählen Sie Test starten.

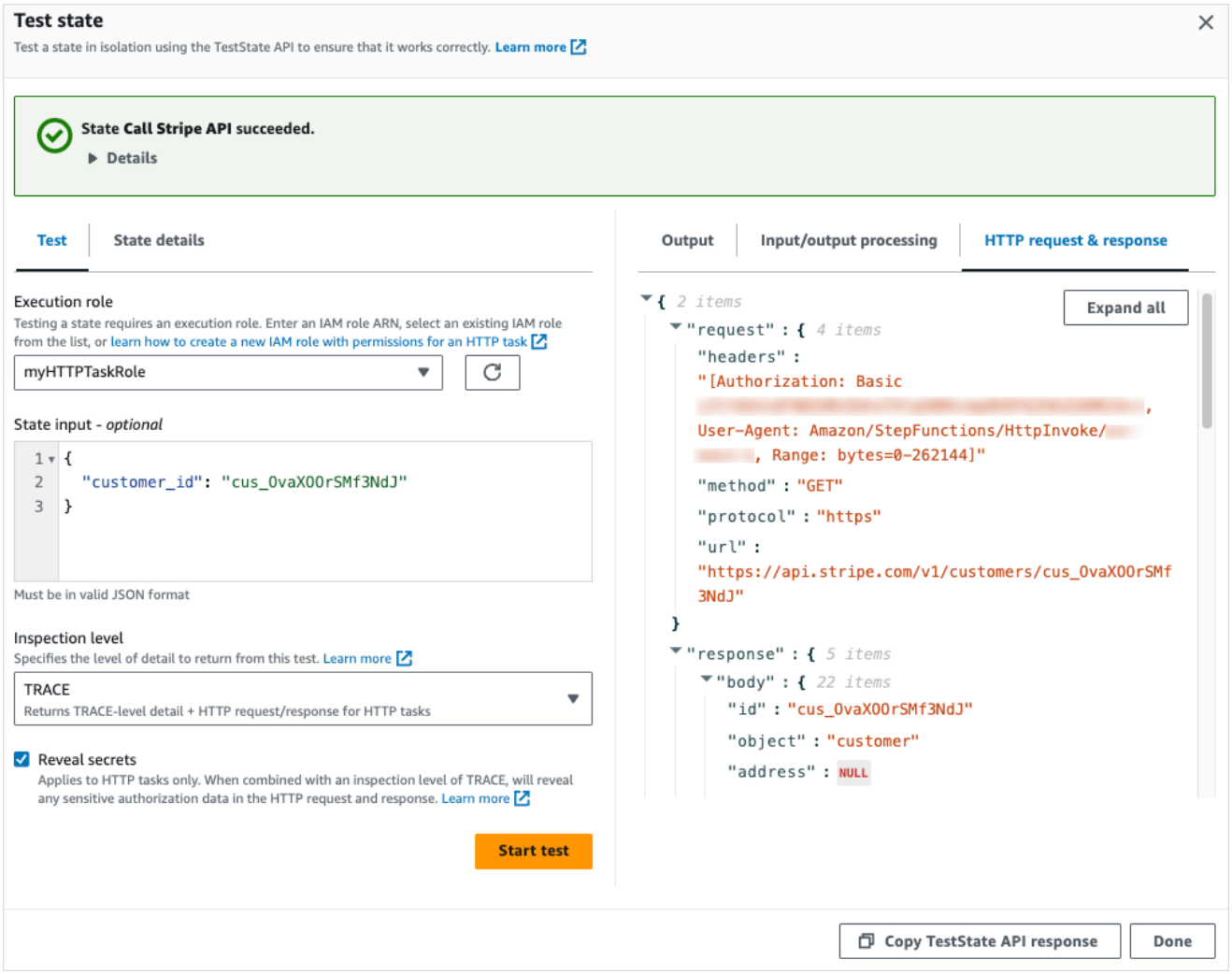
e. Wenn der Test erfolgreich ist, wird die Statusausgabe auf der rechten Seite des Dialogfelds Teststatus angezeigt. Schlägt der Test fehl, wird ein Fehler angezeigt.

   Auf der Registerkarte Statusdetails des Dialogfelds können Sie die Statusdefinition und einen Link zu Ihrer EventBridgeVerbindung sehen.

f. Ändern Sie die Inspektionsebene auf TRACE. Diese Stufe zeigt Ihnen die unverarbeitete HTTP-Anfrage und -Antwort und ist nützlich, um Header, Abfrageparameter und andere API-spezifische Details zu überprüfen.

g. Aktivieren Sie das Kontrollkästchen Geheimnisse enthüllen. In Kombination mit TRACE können Sie mit dieser Einstellung die vertraulichen Daten sehen, die die EventBridge Verbindung einfügt, z. B. API-Schlüssel. Die IAM Benutzeridentität, die Sie für den Zugriff auf die Konsole verwenden, muss berechtigt sein, die `states:RevealSecrets` Aktion auszuführen. Ohne diese Berechtigung wird beim Starten des Tests die Fehlermeldung „Zugriff verweigert“ ausgegeben. Step Functions Ein Beispiel für eine IAM Richtlinie, die die `states:RevealSecrets` Berechtigung festlegt, finden Sie unterIAM -Berechtigungen für die Verwendung der TestState API.

   Die folgende Abbildung zeigt einen erfolgreichen Test für eine HTTP-Aufgabe. Die Inspektionsebene für diesen Status ist auf TRACE festgelegt. Die Registerkarte „HTTP-Anfrage und Antwort“ in der folgenden Abbildung zeigt das Ergebnis des API-Aufrufs eines Drittanbieters.



h. Wählen Sie Test starten.

i. Wenn der Test erfolgreich ist, können Sie Ihre HTTP-Details auf der Registerkarte HTTP-Anfrage und -Antwort sehen.

Antworten auf HTTP-Aufgaben werden nicht unterstützt

Eine HTTP-Aufgabe schlägt mit dem `States.Runtime` Fehler fehl, wenn eine der folgenden Bedingungen für die zurückgegebene Antwort zutrifft:

- Die Antwort enthält einen Inhaltstyp-Header von`application/octet-stream`, `image/*``video/*`, oder. `audio/*`
- Die Antwort kann nicht als gültige Zeichenfolge gelesen werden. Zum Beispiel Binär- oder Bilddaten.

Muster der Serviceintegration

AWS Step Functions lässt sich direkt in Dienste in der Sprache der Amazon-Staaten integrieren. Sie können diese AWS -Services mithilfe dreier verschiedener Serviceintegrationsmuster verwalten:

- Rufen Sie einen Dienst auf und lassen Sie Step Functions sofort nach Erhalt einer HTTP-Antwort zum nächsten Status übergehen.
- Rufen Sie einen Service auf und lassen Sie Step Functions warten, bis ein Job abgeschlossen ist.
- Rufen Sie einen Dienst mit einem Task-Token auf und lassen Sie Step Functions warten, bis dieses Token mit einer Nutzlast zurückgegeben wird.

Jedes dieser Serviceintegrationsmuster wird dadurch gesteuert, wie Sie im `"Resource"`-Feld Ihrer Aufgabendefinition eine URI erstellen.

Möglichkeiten zum Aufrufen eines integrierten Service

- Request Response (Antwort anfordern)
- Ausführen einer Aufgabe (.sync)
- Warten auf einen Callback mit dem Aufgabentoken

Hinweise zur Konfiguration AWS Identity and Access Management (IAM) für integrierte Dienste finden Sie unter. IAM-Richtlinien für integrierte Dienste

Request Response (Antwort anfordern)

Wenn Sie in der `"Resource"` Zeichenfolge Ihres Aufgabenstatus einen Dienst angeben und nur die Ressource angeben, wartet Step Functions auf eine HTTP-Antwort und geht dann zum nächsten Status über. Step Functions wartet nicht darauf, dass ein Job abgeschlossen ist.

Das folgende Beispiel zeigt, wie Sie ein Amazon SNS SNS-Thema veröffentlichen können.

```
"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"
```

```
}
```

Dieses Beispiel verweist auf die Publish-API von Amazon SNS. Der Workflow fährt nach dem Aufruf der `Publish`-API mit dem nächsten Zustand fort.

> **Tip**
>
> Informationen zur Implementierung eines Beispiel-Workflows, der das Request Response Service-Integrationsmuster für Ihren verwendet AWS-Konto, finden Sie unter Modul 2 — Request Response of The AWS Step Functions Workshop.

Ausführen einer Aufgabe (.sync)

Bei integrierten Services wie AWS Batch Amazon ECS kann Step Functions warten, bis eine Anfrage abgeschlossen ist, bevor sie zum nächsten Status übergehen. Damit Step Functions warten kann, geben Sie das `"Resource"` Feld in Ihrer Aufgabenstatusdefinition mit dem `.sync` Suffix an, das hinter dem Ressourcen-URI angehängt wird.

Wenn Sie beispielsweise einen AWS Batch Job einreichen, verwenden Sie das `"Resource"` Feld in der State-Machine-Definition, wie in diesem Beispiel gezeigt.

```
"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"
}
```

Wenn der `.sync` Teil an die Ressource Amazon Resource Name (ARN) angehängt ist, bedeutet dies, dass Step Functions auf den Abschluss des Jobs wartet. Nach dem Aufrufen von AWS Batch `submitJob` wird der Workflow unterbrochen. Wenn der Job abgeschlossen ist, wechselt Step Functions zum nächsten Status. Weitere Informationen finden Sie im AWS Batch Beispielprojekt:Einen Batch-Job verwalten (AWS Batch,Amazon SNS).

Wenn eine Aufgabe, die dieses Dienstintegrationsmuster (`.sync`) verwendet, abgebrochen wird und Step Functions die Aufgabe nicht stornieren kann, fallen möglicherweise zusätzliche Gebühren für den integrierten Service an. Eine Aufgabe kann abgebrochen werden, wenn:

- Die Ausführung der Zustandsmaschine ist gestoppt.
- Ein anderer Zweig eines Parallel-Zustands schlägt mit einem nicht abgefangenen Fehler fehl.
- Eine Iteration eines Map-Status schlägt mit einem nicht erfassten Fehler fehl.

Step Functions versucht nach besten Kräften, die Aufgabe abzubrechen. Wenn beispielsweise eine Step Functions `states:startExecution.sync` Functions-Aufgabe abgebrochen wird, ruft sie die Step Functions `StopExecution` Functions-API-Aktion auf. Es ist jedoch möglich, dass Step Functions die Aufgabe nicht abbrechen kann. Zu den Gründen hierfür gehören, sind aber nicht beschränkt auf:

- Ihrer IAM-Ausführungsrolle fehlt die Berechtigung, den entsprechenden API-Aufruf durchzuführen.
- Ein vorübergehender Serviceausfall ist aufgetreten.

Wenn Sie das `.sync` Dienstintegrationsmuster verwenden, verwendet Step Functions Polling, das Ihre zugewiesenen Kontingente und Ereignisse nutzt, um den Status eines Jobs zu überwachen. Für `.sync` Aufrufe innerhalb desselben Kontos verwendet Step Functions EventBridge Ereignisse und fragt die APIs ab, die Sie im Status angeben. `Task` Für kontoübergreifende `.sync` Aufrufe verwendet Step Functions nur Polling. Beispielsweise führt Step Functions Abfragen auf der DescribeExecutionAPI durch und verwendet Ihr zugewiesenes Kontingent. `states:StartExecution.sync`

> **Tip**
>
> Informationen zur Bereitstellung eines Beispielworkflows, der das Service-Integrationsmuster Run a Job (.sync) für Ihren verwendet AWS-Konto, finden Sie unter Modul 3 — Run a Job (.sync) von The AWS Step Functions Workshop.

Für eine Liste der integrierten Services, die das Warten auf den Abschluss eines Auftrags unterstützen (`.sync`) vgl. Optimierte Integrationen für Step Functions. .

> **Note**
>
> Für Serviceintegrationen, die das `.sync` Muster verwenden, sind zusätzliche IAM-Berechtigungen erforderlich. Weitere Informationen finden Sie unter IAM-Richtlinien für integrierte Dienste.

In einigen Fällen möchten Sie vielleicht, dass Step Functions Ihren Arbeitsablauf fortsetzt, bevor der Job vollständig abgeschlossen ist. Sie können dies auf die gleiche Weise erreichen wie bei der Verwendung des Warten auf einen Callback mit dem Aufgabentoken Service Integration Patterns. Übergeben Sie dazu ein Aufgabentoken an Ihren Job und geben Sie es dann mithilfe eines `SendTaskSuccess`oder `SendTaskFailure`API-Aufrufs zurück. Step Functions verwendet die Daten, die Sie in diesem Aufruf angegeben haben, um die Aufgabe abzuschließen, die Überwachung des Jobs zu beenden und den Workflow fortzusetzen.

Warten auf einen Callback mit dem Aufgabentoken

Callback-Aufgaben bieten eine Möglichkeit, einen Workflow anhalten, bis ein Aufgabentoken zurückgegeben wird. Eine Aufgabe auf eine menschliche Genehmigung warten, mit einer Drittpartei integriert werden oder ältere Systeme aufrufen müssen. Für Aufgaben wie diese können Sie Step Functions anhalten, bis die Workflow-Ausführung das Servicekontingent von einem Jahr erreicht hat (siehe,Kontingente im Zusammenhang mit staatlicher Drosselung), und warten, bis ein externer Prozess oder Workflow abgeschlossen ist. In diesen Situationen können Sie mit Step Functions ein Task-Token an die AWS SDK-Dienstintegrationen und auch an einige optimierte Dienstintegrationen übergeben. Die Aufgabe wird angehalten, bis sie dieses Aufgabentoken mit einem `SendTaskSuccess`- oder `SendTaskFailure`- Aufruf zurückerhält.

Wenn bei einem `Task` Status, der das Callback-Task-Token verwendet, eine Zeitüberschreitung eintritt, wird ein neues zufälliges Token generiert. Sie können vom Kontextobjekt aus auf die Task-Token zugreifen.

> **Note**
>
> Ein Task-Token muss mindestens ein Zeichen enthalten und darf 1024 Zeichen nicht überschreiten.

Für die Verwendung `.waitForTaskToken` mit einer AWS SDK-Integration muss die von Ihnen verwendete API über ein Parameterfeld verfügen, in dem das Task-Token platziert werden kann.

> **Note**
>
> Sie müssen Aufgabentokens von Prinzipalen innerhalb desselben AWS Kontos übergeben. Die Token funktionieren nicht, wenn Sie sie von Schulleitern mit einem anderen AWS Konto senden.

> **Tip**
>
> Informationen zur Bereitstellung eines Beispielworkflows, der ein Integrationsmuster für den Callback-Task-Token-Service für Ihren verwendet AWS-Konto, finden Sie unter Modul 4 — Warten Sie auf einen Rückruf mit dem Task-Token von The Workshop. AWS Step Functions

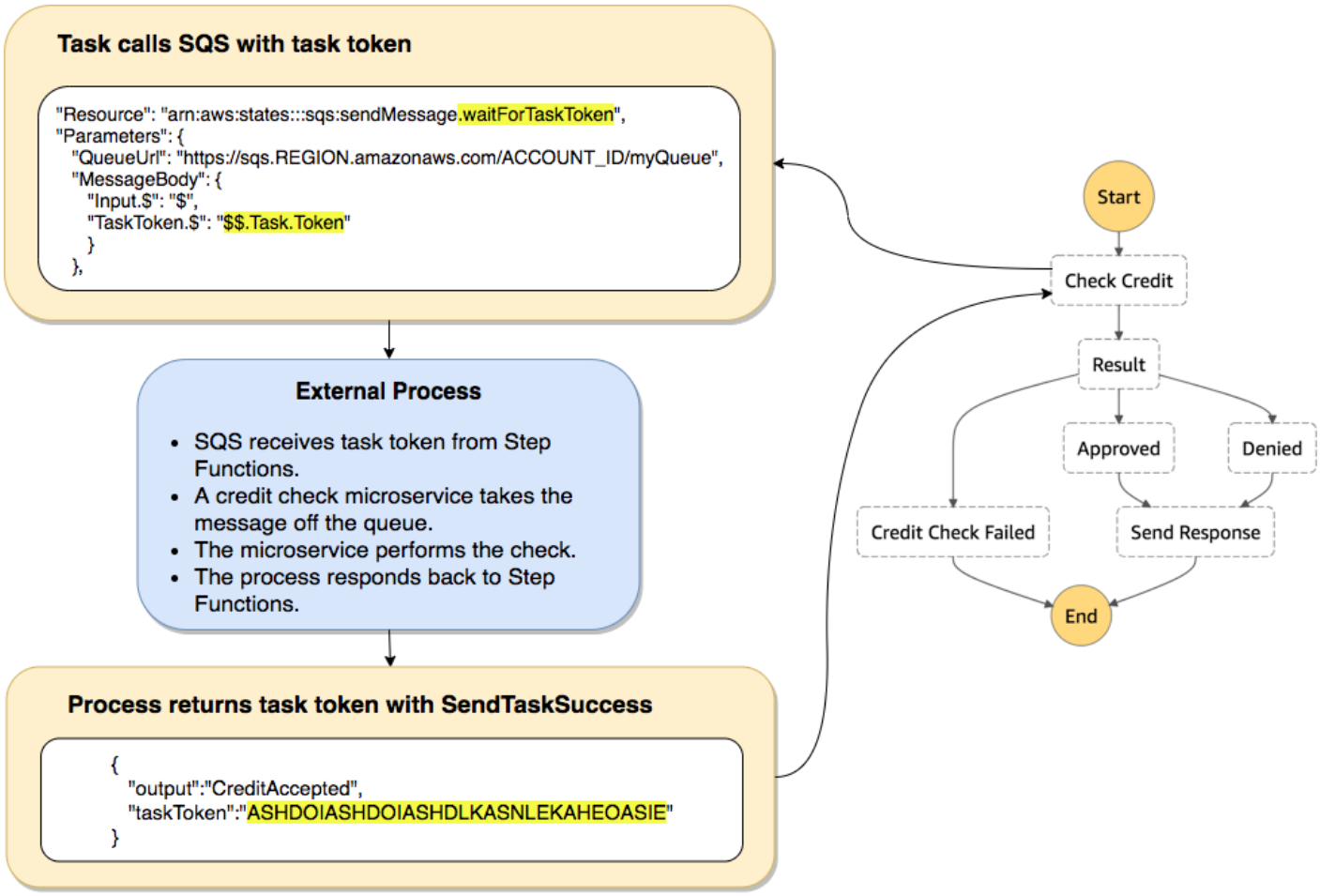
Eine Liste der integrierten Services, die das Warten auf ein Aufgabentoken unterstützen (`.waitForTaskToken`) finden Sie unter Optimierte Integrationen für Step Functions.

Themen

- Aufgabentoken-Beispiel
- Abrufen eines Tokens von einem Kontextobjekt
- Konfigurieren einer Heartbeat-Zeitüberschreitung für eine wartende Aufgabe

Aufgabentoken-Beispiel

In diesem Beispiel muss ein Step Functions Functions-Workflow in einen externen Microservice integriert werden, um eine Bonitätsprüfung als Teil eines Genehmigungsworkflows durchzuführen. Step Functions veröffentlicht eine Amazon SQS SQS-Nachricht, die ein Task-Token als Teil der Nachricht enthält. Ein externes System ist in Amazon SQS integriert und zieht die Nachricht aus der Warteschlange. Wenn das abgeschlossen ist, werden das Ergebnis und das ursprüngliche Task-Token zurückgegeben. Step Functions setzt dann seinen Arbeitsablauf fort.



Das `"Resource"` Feld der Aufgabendefinition, das auf Amazon SQS verweist, ist `.waitForTaskToken` am Ende angehängt.

```
"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"
}
```

Dadurch wird Step Functions angewiesen, eine Pause einzulegen und auf das Task-Token zu warten. Wenn Sie eine Ressource mit `.waitForTaskToken`, abgeben, können Sie im `"Parameters"`-Feld Ihrer Zustandsdefinition mit einer speziellen Pfadbezeichnung (`$$.Task.Token`) auf das Aufgabentoken zugreifen. Der anfängliche `$$.` legt fest, dass der Pfad auf das Kontextobjekt zugreift und das Aufgabentoken für die aktuelle Aufgabe in einer laufenden Ausführung empfängt.

Wenn der Vorgang abgeschlossen ist, ruft der externe Service `SendTaskSuccess` oder `SendTaskFailure` mit `taskToken` auf. Erst danach wird der Workflow mit dem nächsten Zustand fortgesetzt.

> **Note**
>
> Wie zu vermeiden ist, dass unbegrenzt lange gewartet wird, wenn ein Prozess das Aufgabentoken nicht zusammen mit `SendTaskSuccess` oder `SendTaskFailure` sendet, finden Sie unter Konfigurieren einer Heartbeat-Zeitüberschreitung für eine wartende Aufgabe.

Abrufen eines Tokens von einem Kontextobjekt

Das Kontext-Objekt ist ein internes JSON-Objekt mit Informationen zu Ihrer Ausführung. Wie die Zustandsausgabe ist es über einen Pfad aus dem Feld `"Parameters"` während einer Ausführung zugänglich. Wenn aus einer Aufgabendefinition darauf zugegriffen wird, enthält es Informationen über die spezifische Ausführung, einschließlich des Aufgabentokens.

```
{
    "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"
    }
}
```

Sie können auf das Aufgabentoken mit einem speziellen Pfad innerhalb des `"Parameters"`-Felds Ihrer Aufgabendefinition zugreifen. Um auf die Eingabe oder das Kontextobjekt zuzugreifen, geben Sie zuerst an, dass der Parameter ein Pfad ist, indem Sie dem Parameternamen ein `.$` anhängen. Das folgende Beispiel gibt Knoten sowohl über das Eingabe- und das Kontext-Objekt in einer `"Parameters"`-Spezifikation an.

```
"Parameters": {
   "Input.$": "$",
   "TaskToken.$": "$$.Task.Token"
 },
```

In beiden Fällen wird Step Functions durch Anhängen `.$` an den Parameternamen angewiesen, einen Pfad zu erwarten. Im ersten Fall ist `"$"` ein Pfad, der die gesamte Eingabe enthält. Im zweiten Fall gibt `$$.` an, dass der Pfad auf das Kontext-Objekt zugreift, und `$$.Task.Token` setzt den Parameter auf den Wert des Aufgabentokens in dem Aufgabentoken einer laufenden Ausführung.

Im Amazon SQS SQS-Beispiel weist das `.waitForTaskToken` `"Resource"` Feld Step Functions an, auf die Rückgabe des Aufgabentokens zu warten. Der `"TaskToken.$": "`**`$$.Task.Token`**`"` Parameter übergibt dieses Token als Teil der Amazon SQS SQS-Nachricht.

```
"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"
}
```

Weitere Informationen über das Kontext-Objekt finden Sie unter Context-Objekt im Abschnitt Verarbeitung von Eingabe und Ausgabe in diesem Handbuch.

Konfigurieren einer Heartbeat-Zeitüberschreitung für eine wartende Aufgabe

Eine Aufgabe, die auf einen Aufgabentoken wartet, wartet, bis die Ausführung das einjährige Servicekontingent erreicht (siehe Kontingente im Zusammenhang mit staatlicher Drosselung). Um zu verhindern, dass Ausführungen hängen bleiben, können Sie eine Heartbeat-Zeitüberschreitung in Ihrer Zustandsautomatendefinition konfigurieren. Verwenden Sie das `HeartbeatSeconds`-Feld, um das Timeout-Intervall anzugeben.

```
{
  "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
    }
  }
}
```

In dieser Zustandsmaschinendefinition sendet eine Aufgabe eine Nachricht an Amazon SQS und wartet darauf, dass ein externer Prozess mit dem bereitgestellten Task-Token zurückruft. Das `"HeartbeatSeconds": 600`-Feld legt das Intervall der Heartbeat-Zeitüberschreitung Intervall auf 10 Minuten fest. Die Aufgabe wartet auf die Rückgabe des Aufgabentokens mit einer der folgenden API-Aktionen:

- `SendTaskSuccess`
- `SendTaskFailure`
- `SendTaskHeartbeat`

Wenn die wartende Aufgabe innerhalb dieser 10 Minuten kein gültiges Aufgabentoken erhält, schlägt die Aufgabe mit dem Fehlernamen `States.Timeout` fehl.

Weitere Informationen finden Sie im Beispielprojekt Beispiel für ein Rückrufmuster (Amazon SQS, Amazon SNS, Lambda) für eine Callback-Aufgabe.

Parameter an eine Service-API übergeben

Über das `Parameters`-Feld in einem `Task`-Status können Sie steuern, welche Parameter an eine Service-API übergeben werden.

Innerhalb des `Parameters` Felds müssen Sie die Pluralform der Array-Parameter in einer API-Aktion verwenden. Wenn Sie beispielsweise das Feld Filter der `DescribeSnapshots` API-Aktion für die Integration mit Amazon EC2 verwenden, müssen Sie das Feld als `Filters` definieren. Wenn Sie die Pluralform nicht verwenden, gibt Step Functions den folgenden Fehler zurück:

```
The field Filter is not supported by Step Functions.
```

Übergeben Sie statisches JSON als Parameter

Sie können ein JSON-Objekt direkt in die Definition Ihres Zustandsautomaten einschließen, um es als Parameter an eine Ressource zu übergeben.

Um beispielsweise den Parameter `RetryStrategy` für die `SubmitJob`-API für AWS Batch festzulegen, können Sie Folgendes in Ihre Parameter einschließen.

```
"RetryStrategy": {
  "attempts": 5
}
```

Sie können auch mehrere Parameter mit statischem JSON übergeben. Als vollständigeres Beispiel finden Sie im Folgenden die `Parameters` Felder `Resource` und die Spezifikation einer Aufgabe, die zu einem Amazon SNS-Thema mit dem Namen *`myTopic`* veröffentlicht wird.

```
"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"
      }
    }
  },
```

Übergeben Sie die Zustandseingabe als Parameter mithilfe von Pfaden

Sie können Teile der Statuseingabe als Parameter übergeben, indem Sie Pfade verwenden. Ein Pfad ist eine Zeichenfolge, die mit beginnt und verwendet wird$, um Komponenten innerhalb von JSON-Text zu identifizieren. Die Pfade von Step Functions verwenden JsonPathSyntax.

Um anzugeben, dass ein Parameter einen Pfad verwendet, beenden Sie den Parameternamen mit`.$`. Wenn Ihre Statuseingabe beispielsweise Text in einem Knoten mit dem Namen enthält`message`, können Sie diesen Text als Parameter mithilfe eines Pfads übergeben.

Beachten Sie die folgende Zustandseingabe:

```
{
  "comment": "A message in the state input",
  "input": {
    "message": "foo",
    "otherInfo": "bar"
  },
  "data": "example"
}
```

Um den Wert des `message` als Parameter benannten Knotens zu übergeben, geben Sie die folgende Syntax an:

```
"Parameters": {"myMessage.$": "$.input.message"},
```

Step Functions übergibt dann den Wert `foo` als Parameter.

Weitere Informationen zur Verwendung von Parametern in Step Functions finden Sie im Folgenden:

- Verarbeitung von Eingabe und Ausgabe
- InputPath, Parameter und ResultSelector

Übergeben von Kontext-Knoten als Parameter

Zusätzlich zu statischem Inhalt und Knoten aus der Zustandseingabe können Sie Knoten aus dem Kontext-Objekt als Parameter übergeben. Das Kontext-Objekt besteht aus dynamischen JSON-Daten, die während der Ausführung eines Zustandsautomaten vorhanden sind. Es enthält unter anderem Informationen über Ihren Zustandsautomaten und die aktuelle Ausführung. Sie können auf das Kontext-Objekt mithilfe eines Pfads im `"Parameters"`-Feld der Definition eines Zustands zugreifen.

Weitere Informationen über das Kontext-Objekt sowie den Zugriff auf die Daten aus einem `"Parameters"`-Feld finden Sie unter:

- Context-Objekt
- Zugriff auf das Kontext-Objekt
- Abrufen eines Tokens von einem Kontextobjekt

Änderungsprotokoll für unterstützte AWS SDK-Integrationen

In der folgenden Tabelle wird zusammengefasst, wann Dienste ursprünglich in Step Functions integriert wurden und wann ihre Integrations-API zuletzt aktualisiert wurde. Einzelheiten zur Verwendung von Integrationen finden Sie unter. AWS SDK-Serviceintegrationen

> **Important**
>
> Die Unterstützung von API-Aktionen wird vierteljährlich veröffentlicht. Aktualisierungen bereits unterstützter Aktionen, wie z. B. neue Parameter, sind möglicherweise nicht sofort verfügbar.

Service	Erste Unterstützung	Aktualisiert	
AWS AppFabric	18. Januar 2024		
B2B Data Interchange	18. Januar 2024		
AWS Data Exports	18. Januar 2024		
Amazon Bedrock	18. Januar 2024		

Service	Erste Unterstützung	Aktualisiert	
Amazon Bedrock Agents	18. Januar 2024		
Amazon Bedrock Runtime Agents	18. Januar 2024		
Amazon Bedrock Runtime	18. Januar 2024		
Amazon CloudFront KeyValueStore	18. Januar 2024		
Amazon CodeGuru Security	18. Januar 2024		
AWS Cost Optimization Hub	18. Januar 2024		
Amazon DataZone	18. Januar 2024		
Amazon EKS Auth	18. Januar 2024		
AWS Entity Resolution	18. Januar 2024		
Kostenloses AWS-Kontingent	18. Januar 2024		
Amazon Inspector Scan	18. Januar 2024		
AWS Launch Wizard	18. Januar 2024		
Amazon Managed Blockchain Query	18. Januar 2024		
AWS Elemental MediaPackage V2	18. Januar 2024		

Service	Erste Unterstützung	Aktualisiert	
AWS HealthImaging	18. Januar 2024		
Network Manager	18. Januar 2024		
AWS Payment Cryptography	18. Januar 2024		
AWS Payment Cryptography Data	18. Januar 2024		
AWS Private CA Connector for Active Directory	18. Januar 2024		
Amazon Q Business	18. Januar 2024		
Amazon Q Connect	18. Januar 2024		
AWS re:Post	18. Januar 2024		
Amazon Timestream Query	18. Januar 2024		
Amazon Timestream Write	18. Januar 2024		
Trusted Advisor	18. Januar 2024		
Verified Permissions	18. Januar 2024		
Amazon WorkSpaces Thin Client	18. Januar 2024		
AWS CloudTrail Data	16. Juni 2023		
Amazon CloudWatch Internet Monitor	16. Juni 2023		

Service	Erste Unterstützung	Aktualisiert	
Amazon Interacti ve Video Service RealTime	16. Juni 2023		
AWS IoT TwinMaker	16. Juni 2023		
Amazon OpenSearch Ingestion	16. Juni 2023		
AWS Telco Network Builder	16. Juni 2023		
Amazon VPC Lattice	16. Juni 2023		
AWS Backup Storage	17. Februar 2023		
Amazon Chime Media Pipelines	17. Februar 2023		
Amazon Chime Voice	17. Februar 2023		
AWS Clean Rooms	17. Februar 2023	18. Januar 2024	
Amazon CodeCatalyst	17. Februar 2023		
Amazon Connect Cases	17. Februar 2023		
AWS Control Tower	17. Februar 2023		
Amazon DocumentDB Elastic Clusters	17. Februar 2023		
Amazon EMR Serverless	17. Februar 2023		
Amazon IVS Chat	17. Februar 2023		

Service	Erste Unterstützung	Aktualisiert	
Amazon Kendra Intelligent Ranking	17. Februar 2023		
AWS HealthOmics	17. Februar 2023		
Amazon Redshift Serverless	17. Februar 2023		
Amazon Security Lake	17. Februar 2023		
AWS Health	17. Februar 2023		
AWS IoT FleetWise	17. Februar 2023		
AWS IoT RoboRunner	17. Februar 2023		
AWS Mainframe Modernization	17. Februar 2023		
AWS Migration Hub Orchestrator	17. Februar 2023		
AWS Private 5G	17. Februar 2023		
AWS Ressourcen Explorer	17. Februar 2023		
AWS SimSpace Weaver	17. Februar 2023		
AWS Support App	17. Februar 2023		
CloudWatch Observability Access Manager	17. Februar 2023		
EventBridge Pipes	17. Februar 2023		

Service	Erste Unterstützung	Aktualisiert	
EventBridge Scheduler	17. Februar 2023		
IAM Roles Anywhere	17. Februar 2023		
Kinesis Video WebRTC Storage	17. Februar 2023		
License Manager Linux Subscriptions	17. Februar 2023		
License Manager User Subscriptions	17. Februar 2023		
OpenSearch Serverless	17. Februar 2023		
Route 53 ARC Zonal Shift	17. Februar 2023		
SageMaker Geospatial	17. Februar 2023		
SageMaker Metrics	17. Februar 2023		
Systems Manager for SAP	17. Februar 2023		
AWS Account Management	19. April 2022		
AWS Amplify	30. September 2021		
AWS App Mesh	30. September 2021		
AWS App Runner	30. September 2021	17. Februar 2023	
AWS AppConfig	30. September 2021		

Service	Erste Unterstützung	Aktualisiert	
AWS AppConfig Data	19. April 2022		
AWS AppSync	30. September 2021	17. Februar 2023	
AWS Application Discovery Service	30. September 2021		
AWS Application Migration Service	30. September 2021		
AWS Audit Manager	30. September 2021		
AWS Auto Scaling Plans	30. September 2021		
AWS Backup	30. September 2021	17. Februar 2023	
AWS Backup gateway	19. April 2022	17. Februar 2023	
AWS Batch	30. September 2021	17. Februar 2023	
AWS Billing Conductor	26. Juli 2022	17. Februar 2023	
AWS Budgets	30. September 2021		
AWS Certificate Manager	30. September 2021		
AWS Private Certificate Authority	30. September 2021		
AWS Cloud Map	30. September 2021		
AWS Cloud9	30. September 2021		
AWS CloudFormation	30. September 2021	17. Februar 2023	
AWS CloudHSM	30. September 2021		

Service	Erste Unterstützung	Aktualisiert	
AWS CloudHSM	30. September 2021		
AWS CloudTrail	30. September 2021	17. Februar 2023	
AWS Cloud Control	19. April 2022		
AWS CodeBuild	30. September 2021		
AWS CodeCommit	30. September 2021	17. Februar 2023	
AWS CodeDeploy	30. September 2021		
AWS CodePipeline	30. September 2021		
AWS CodeStar	30. September 2021		
AWS CodeStar	30. September 2021		
AWS CodeStar	30. September 2021		
AWS Compute Optimizer	30. September 2021	17. Februar 2023	
AWS Config	30. September 2021	26. Juli 2022	
AWS Cost Explorer Service	30. September 2021	17. Februar 2023	
AWS Cost and Usage Report	30. September 2021		
AWS Data Exchange	30. September 2021	26. Juli 2022	
AWS Data Pipeline	30. September 2021		
AWS DataSync	30. September 2021	26. Juli 2022	
AWS Database Migration Service	30. September 2021		

Service	Erste Unterstützung	Aktualisiert	
AWS Device Farm	30. September 2021		
AWS Direct Connect	30. September 2021		
AWS Directory Service	30. September 2021	17. Februar 2023	
AWS EC2 Instance Connect	30. September 2021		
AWS Elastic Beanstalk	30. September 2021		
AWS Elemental MediaLive	30. September 2021		
AWS Elemental MediaPackage	30. September 2021		
AWS Elemental MediaPackage VOD	30. September 2021		
AWS Elemental MediaStore	30. September 2021		
AWS Fault Injection Service	30. September 2021		
AWS Firewall Manager	30. September 2021	17. Februar 2023	
AWS Glue	30. September 2021	17. Februar 2023	
AWS Glue DataBrew	30. September 2021		
AWS IoT Greengrass	30. September 2021		
AWS Ground Station	30. September 2021	17. Februar 2023	

Service	Erste Unterstützung	Aktualisiert	
AWS Identity and Access Management	30. September 2021		
AWS IoT	30. September 2021	17. Februar 2023	
AWS IoT 1-Click	30. September 2021		
AWS IoT Analytics	30. September 2021		
AWS IoT Core Device Advisor	30. September 2021		
AWS IoT Events	30. September 2021		
AWS IoT Events-Da ten	30. September 2021		
AWS IoT Fleet Hub	30. September 2021		
AWS IoT Greengrass Version 2	30. September 2021		
AWS IoT Jobs Data Plane	30. September 2021		
AWS IoT Secure Tunneling	30. September 2021		
AWS IoT SiteWise	30. September 2021	17. Februar 2023	
AWS IoT Things Graph	30. September 2021		
AWS IoT Wireless	30. September 2021	17. Februar 2023	
AWS Key Management Service	30. September 2021	26. Juli 2022	
AWS Lake Formation	30. September 2021	17. Februar 2023	

Service	Erste Unterstützung	Aktualisiert	
AWS Lambda	30. September 2021	17. Februar 2023	
AWS License Manager	30. September 2021	17. Februar 2023	
AWS Marketplace	30. September 2021	17. Februar 2023	
AWS Marketplace Commerce Analytics	30. September 2021		
AWS Marketplace Entitlement Service	30. September 2021		
AWS Elemental MediaTailor	30. September 2021	26. Juli 2022	
AWS Migration Hub	30. September 2021		
AWS Migration Hub Config	30. September 2021		
Strategieempfehlungen für den AWS Migration Hub	19. April 2022	17. Februar 2023	
AWS Mobile	30. September 2021		
AWS Network Firewall	30. September 2021		
AWS OpsWorks	30. September 2021		
AWS OpsWorks CM	30. September 2021		
AWS Organizations	30. September 2021	17. Februar 2023	
AWS Outposts	30. September 2021		
AWS Panorama	19. April 2022	17. Februar 2023	

Service	Erste Unterstützung	Aktualisiert	
Amazon Relationa l Database Service Performance Insights	30. September 2021		
AWS-Preisliste	30. September 2021		
Amazon Relational Database Service	30. September 2021		
AWS Resilience Hub	19. April 2022		
AWS Resource Access Manager	30. September 2021		
AWS Resource Groups	30. September 2021	17. Februar 2023	
AWS Resource Groups Tagging API	30. September 2021		
AWS RoboMaker	30. September 2021		
AWS IAM Identity Center	30. September 2021	17. Februar 2023	
AWS SSO OIDC	30. September 2021		
AWS Secrets Manager	30. September 2021		
AWS Security Token Service	30. September 2021		
AWS Security Hub	30. September 2021		
AWS Server Migration Service	30. September 2021		

Service	Erste Unterstützung	Aktualisiert	
AWS Service Catalog	30. September 2021		
AWS Service Catalog AppRegistry	30. September 2021	17. Februar 2023	
AWS Shield	30. September 2021		
AWS Signer	30. September 2021		
AWS IAM Identity Center	30. September 2021		
AWS IAM Identity Center Admin	30. September 2021		
AWS Step Functions	30. September 2021	17. Februar 2023	
AWS Storage Gateway	30. September 2021		
AWS Support	30. September 2021		
AWS Transfer Family	30. September 2021	17. Februar 2023	
AWS WAF	30. September 2021		
AWS WAF Regional	30. September 2021		
AWS WAFV2	30. September 2021		
AWS Well-Architected Tool	30. September 2021	17. Februar 2023	
AWS X-Ray	30. September 2021	17. Februar 2023	
AWS Marketplace Metering Service	30. September 2021		

Service	Erste Unterstützung	Aktualisiert	
AWS Serverless Application Repository	30. September 2021		
AWS Identity and Access Management Access Analyzer	30. September 2021		
Alexa for Business	30. September 2021		
Amazon API Gateway	30. September 2021	17. Februar 2023	
Amazon API Gateway	30. September 2021		
Amazon AppIntegrations	30. September 2021		
Amazon AppStream 2.0	30. September 2021		
Amazon AppFlow	30. September 2021	17. Februar 2023	
Amazon Athena	30. September 2021	17. Februar 2023	
Amazon Augmented AI	30. September 2021		
Amazon Braket	30. September 2021		
Amazon Chime	30. September 2021		
Amazon Chime Meetings	19. April 2022	17. Februar 2023	
Amazon Cloud Directory	30. September 2021		
Amazon CloudFront	30. September 2021	17. Februar 2023	
Amazon CloudSearch	30. September 2021		

Service	Erste Unterstützung	Aktualisiert	
Amazon CloudWatch	30. September 2021	17. Februar 2023	
Amazon CloudWatch Application Insights	30. September 2021		
CloudWatch Evidently	19. April 2022		
Amazon CloudWatch Logs	30. September 2021		
Amazon CloudWatch RUM	19. April 2022	17. Februar 2023	
Amazon CloudWatch Synthetics	30. September 2021		
Amazon CodeGuru Profiler	30. September 2021		
Amazon CodeGuru Reviewer	30. September 2021		
Amazon Cognito	30. September 2021		
Amazon Cognito Identity Provider	30. September 2021		
Amazon Cognito Sync	30. September 2021		
Amazon Comprehend	30. September 2021	17. Februar 2023	
Amazon Comprehend Medical	30. September 2021		
Amazon Connect Contact Lens	30. September 2021		

Service	Erste Unterstützung	Aktualisiert	
Amazon Connect Participant Service	30. September 2021		
Amazon Connect	30. September 2021	17. Februar 2023	
Amazon Connect Voice ID	19. April 2022		
Amazon Connect Wisdom	19. April 2022		
Amazon Data Lifecycle Manager	30. September 2021		
Amazon Detective	30. September 2021		
Amazon DevOps Guru	30. September 2021	26. Juli 2022	
Amazon DocumentDB (with MongoDB compatibility)	30. September 2021		
Amazon DynamoDB	30. September 2021	17. Februar 2023	
Amazon DynamoDB Streams	30. September 2021		
Amazon EC2 Container Registry	30. September 2021		
Amazon EC2 Container Service	30. September 2021	17. Februar 2023	
Amazon EC2 Systems Manager	30. September 2021	17. Februar 2023	
Amazon EMR	30. September 2021	17. Februar 2023	

Service	Erste Unterstützung	Aktualisiert	
Amazon ElastiCache	30. September 2021		
Amazon Elastic Inference	30. September 2021		
Amazon Elastic Block Store	30. September 2021		
Amazon Elastic Compute Cloud	30. September 2021	17. Februar 2023	
Amazon Elastic Container Registry Public	30. September 2021		
Amazon Elastic File System	30. September 2021		
Amazon Elastic Kubernetes Service	30. September 2021	17. Februar 2023	
Amazon EMR	30. September 2021		
Amazon Elastic Transcoder	30. September 2021		
Amazon OpenSearch Service	30. September 2021	17. Februar 2023	
Amazon OpenSearch Service	19. April 2022	17. Februar 2023	
Amazon EventBridge	30. September 2021	17. Februar 2023	
Amazon FSx	30. September 2021	17. Februar 2023	
Amazon Forecast Query	30. September 2021	17. Februar 2023	

Service	Erste Unterstützung	Aktualisiert	
Amazon Forecast Service	30. September 2021	17. Februar 2023	
Amazon Fraud Detector	30. September 2021		
Amazon GameLift	30. September 2021	17. Februar 2023	
Amazon GameSparks	26. Juli 2022		
Amazon S3 Glacier	30. September 2021		
Amazon GuardDuty	30. September 2021		
AWS HealthLake	30. September 2021		
Amazon Honeycode	30. September 2021		
Amazon Inspector	30. September 2021		
Amazon Inspector V2	19. April 2022		
Amazon Interactive Video Service	30. September 2021		
Amazon Kendra	30. September 2021		
Amazon Kinesis	30. September 2021		
Amazon Kinesis Analytics	30. September 2021		
Amazon Kinesis Analytics V2	30. September 2021		
Amazon Kinesis Firehose	30. September 2021		

Service	Erste Unterstützung	Aktualisiert	
Amazon Kinesis Video Signaling Channels	30. September 2021		
Amazon Kinesis Video Streams	30. September 2021	17. Februar 2023	
Amazon Kinesis Video Streams Archived Media	30. September 2021		
Amazon Kinesis video stream	30. September 2021		
Amazon Lex Model Building Service	30. September 2021		
Amazon Lex Model Building Service V2	30. September 2021	17. Februar 2023	
Amazon Lex	30. September 2021		
Amazon Lex Runtime V2	30. September 2021		
Amazon Lightsail	30. September 2021	17. Februar 2023	
Amazon Location Service	30. September 2021	17. Februar 2023	
Amazon Lookout for Equipment	30. September 2021		
Amazon Lookout for Metrics	30. September 2021	17. Februar 2023	
Amazon Lookout for Vision	30. September 2021		

Service	Erste Unterstützung	Aktualisiert	
Amazon MQ	30. September 2021		
Amazon Macie	30. September 2021		
Amazon Macie 2	30. September 2021	17. Februar 2023	
Amazon Managed Blockchain	30. September 2021	17. Februar 2023	
Amazon Managed Grafana	19. April 2022	17. Februar 2023	
Amazon Managed Service for Prometheus	30. September 2021	17. Februar 2023	
Amazon Managed Streaming for Apache Kafka	30. September 2021	17. Februar 2023	
Amazon Managed Streaming for Apache Kafkax	19. April 2022		
Amazon Managed Workflows for Apache Airflow	30. September 2021		
Amazon Mechanical Turk	30. September 2021		
Amazon MemoryDB for Redis	19. April 2022	17. Februar 2023	
Amazon Nimble Studio	30. September 2021		
Amazon Personalize	30. September 2021	17. Februar 2023	

Service	Erste Unterstützung	Aktualisiert	
Amazon Personalize Events	30. September 2021		
Amazon Personalize Runtime	30. September 2021		
Amazon Pinpoint	30. September 2021		
Amazon Pinpoint Email Service	30. September 2021		
Amazon Pinpoint SMS and Voice Service	30. September 2021		
Amazon Pinpoint SMS and Voice V2 Service	26. Juli 2022		
Amazon Polly	30. September 2021		
Amazon QLDB	30. September 2021		
Amazon QLDB Session	30. September 2021		
Amazon QuickSight	30. September 2021	17. Februar 2023	
Amazon Redshift	30. September 2021		
Amazon Redshift Data API	30. September 2021		
Amazon Rekognition	30. September 2021	17. Februar 2023	
Amazon Relational Database Service	30. September 2021	17. Februar 2023	

Service	Erste Unterstützung	Aktualisiert	
Amazon Route 53	30. September 2021		
Amazon Route 53 Recovery Control Config	30. September 2021	26. Juli 2022	
Amazon Route 53 Domains	30. September 2021	17. Februar 2023	
Amazon Route 53 Resolver	30. September 2021		
Amazon S3 on Outposts	30. September 2021	26. Juli 2022	
Amazon SageMaker Runtime Feature Store Runtime	30. September 2021		
Amazon SageMaker Runtime Runtime	30. September 2021		
Amazon SageMaker	30. September 2021	17. Februar 2023	
Amazon SageMaker Edge Manager	30. September 2021		
Amazon Simple Email Service	30. September 2021		
Amazon Simple Email Service V2	30. September 2021	17. Februar 2023	
Amazon Simple Notification Service	30. September 2021	17. Februar 2023	
Amazon Simple Queue Service	30. September 2021	17. Februar 2023	

Service	Erste Unterstützung	Aktualisiert	
Amazon Simple Storage Service	30. September 2021	17. Februar 2023	
Amazon Simple Workflow Service	30. September 2021		
Amazon Textract	30. September 2021	17. Februar 2023	
Amazon Transcribe	30. September 2021		
Amazon Translate	30. September 2021	17. Februar 2023	
Amazon WorkDocs	30. September 2021	17. Februar 2023	
Amazon WorkMail	30. September 2021	17. Februar 2023	
Amazon WorkMail Message Flow	30. September 2021		
Amazon WorkSpaces	30. September 2021	17. Februar 2023	
Amazon WorkSpaces Web	19. April 2022	17. Februar 2023	
Amplify	30. September 2021		
Amplify UI Builder	19. April 2022	17. Februar 2023	
Application Auto Scaling	30. September 2021		
Amazon EC2 Auto Scaling	30. September 2021	17. Februar 2023	
CodeArtifact	30. September 2021		
DynamoDB Accelerator	30. September 2021		

Service	Erste Unterstützung	Aktualisiert	
EC2 Image Builder	30. September 2021		
AWS Elastic Disaster Recovery	19. April 2022	17. Februar 2023	
Elastic Load Balancing	30. September 2021		
Elastic Load Balancing V2	30. September 2021		
MediaConnect	30. September 2021		
Amazon S3 Control	30. September 2021	17. Februar 2023	
Recycle Bin for Amazon EBS	19. April 2022	17. Februar 2023	
Savings Plans	30. September 2021		
Amazon EventBridge Schema Registry	30. September 2021		
Service Quotas	30. September 2021		
AWS Snowball	30. September 2021		

Beispielprojekte für Step Functions

In der AWS Step FunctionsKonsole können Sie eine der folgenden Startvorlagen auswählen, um Zustandsmaschinen auf Ihrem Computer bereitzustellenAWS-Konten. Bei diesen Startvorlagen handelt es sich um ready-to-run Beispielprojekte, die automatisch den Workflow-Prototyp und die Workflow-Definition sowie alle zugehörigen AWS Ressourcen für das Projekt erstellen.

Sie können diese Beispielprojekte verwenden, um sie unverändert bereitzustellen und auszuführen, oder Sie können die Workflow-Prototypen verwenden, um darauf aufzubauen. Wenn Sie auf diesen Projekten aufbauen, erstellt Step Functions den Workflow-Prototyp, stellt jedoch nicht die in der Workflow-Definition aufgeführten Ressourcen bereit.

Wenn Sie die Beispielprojekte bereitstellen, stellen sie eine voll funktionsfähige Zustandsmaschine bereit und erstellen die zugehörigen Ressourcen, damit die Zustandsmaschine ausgeführt werden kann. Wenn Sie ein Beispielprojekt erstellen, erstellt Step Functions die zugehörigen Ressourcen, AWS CloudFormation auf die von der Zustandsmaschine verwiesen wird.

Themen

- Beispiel für selektives Checkpointing (Express-Workflows)
- Ein AWS CodeBuild Projekt erstellen (CodeBuild, Amazon SNS)
- Daten vorverarbeiten und ein Modell für maschinelles Lernen trainieren
- Beispiel für eine Lambda-Orchestrierung
- Eine Athena-Abfrage starten
- Führen Sie mehrere Abfragen aus (Amazon Athena, Amazon SNS)
- Große Datensätze abfragen (Amazon Athena, Amazon S3 AWS Glue, Amazon SNS)
- Daten auf dem neuesten Stand halten (Amazon Athena, Amazon S3, AWS Glue)
- Einen Amazon EKS-Cluster verwalten
- Rufen Sie API Gateway auf
- Rufen Sie mithilfe der API Gateway Gateway-Integration einen auf Fargate laufenden Microservice auf
- Senden Sie ein benutzerdefiniertes Ereignis an EventBridge
- Synchrone Express-Workflows aufrufen
- Führen Sie ETL/ELT-Workflows mit Amazon Redshift aus (Lambda, Amazon Redshift Data API)
- Verwenden Step Functions und AWS Batch mit Fehlerbehandlung
- Einen AWS Batch Job ausfindig machen
- AWS Batch mit Lambda
- Führen Sie AI-Prompt-Chaining durch mit Amazon Bedrock

Einen Batch-Job verwalten (AWS Batch,Amazon SNS)

Dieses Beispielprojekt zeigt, wie Sie einen AWS Batch-Auftrag übermitteln und anschließend eine Amazon SNS-Benachrichtigung senden, die darüber informiert, ob der Auftrag erfolgreich abgeschlossen wurde oder fehlgeschlagen ist. Mit der Bereitstellung dieses Beispielprojekts werden ein AWS Step Functions-Zustandsautomat, ein AWS Batch-Auftrag und ein Amazon SNS-Thema erstellt.

In diesem Projekt verwendet Step Functions einen Zustandsautomaten, um den AWS Batch-Auftrag synchron aufzurufen. Anschließend wartet Step Functions darauf, dass der Auftrag erfolgreich abgeschlossen wird oder fehlschlägt, und sendet ein Amazon SNS-Thema mit einer Meldung, ob der Auftrag erfolgreich abgeschlossen wurde oder fehlgeschlagen ist.

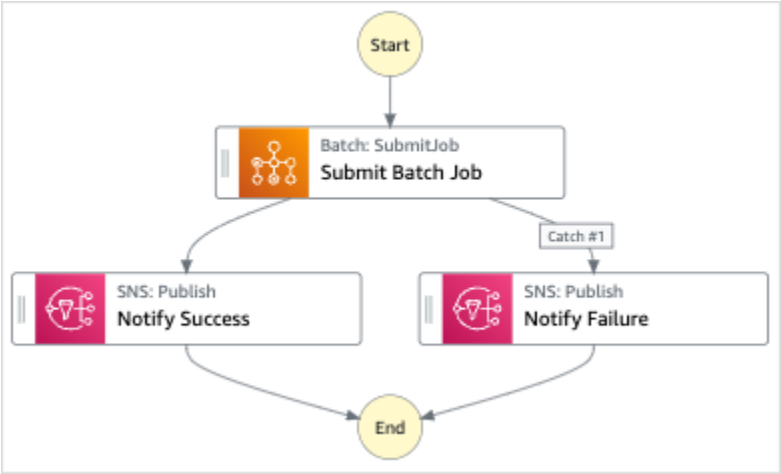
Schritt 1: Erstellen Sie den Zustandsmaschine und stellen Sie Ressourcen bereit

1. Öffnen Sie die Step Functions Functions-Konsole und wählen Sie Create State Machine.
2. Geben Sie **`Manage a batch job`** etwas in das Suchfeld ein und wählen Sie dann aus den zurückgegebenen Suchergebnissen die Option Batch-Job verwalten aus.
3. Wählen Sie Next (Weiter), um fortzufahren.
4. Step Functions listet die Funktionen auf, die in dem von Ihnen ausgewählten Beispielprojekt AWS-Services verwendet wurden. Außerdem wird ein Workflow-Diagramm für das Beispielprojekt angezeigt. Stellen Sie dieses Projekt für Ihr Projekt bereit AWS-Konto oder verwenden Sie es als Ausgangspunkt für die Erstellung Ihrer eigenen Projekte. Je nachdem, wie Sie vorgehen möchten, wählen Sie „Demo ausführen“ oder „Darauf aufbauen“.

   In diesem Beispielprojekt werden die folgenden Ressourcen bereitgestellt:

   - Ein Job AWS Batch
   - Amazon-SNS-Thema
   - Eine AWS Step Functions Staatsmaschine
   - Verwandte AWS Identity and Access Management Rollen (IAM)

   Die folgende Abbildung zeigt das Workflow-Diagramm für das Beispielprojekt „Einen Batch-Job verwalten“:

5. Wählen Sie Vorlage verwenden, um mit Ihrer Auswahl fortzufahren.
6. Führen Sie eine der folgenden Aktionen aus:

- Wenn Sie „Darauf aufbauen“ ausgewählt haben, erstellt Step Functions den Workflow-Prototyp für das von Ihnen ausgewählte Beispielprojekt. Step Functions stellt die in der Workflow-Definition aufgeführten Ressourcen nicht bereit.

  Ziehen Sie in Workflow Studio Status per Drag-and-Drop aus demEntwurfsmodus, Bundesstaaten-Browser um mit der Erstellung Ihres Workflow-Prototyps fortzufahren. Oder wechseln Sie zu demCodemodus, der einen integrierten Code-Editor bietet, der VS Code ähnelt, um die Amazon States Language (ASL-) Definition Ihrer Zustandsmaschine in der Step Functions Functions-Konsole zu aktualisieren. Weitere Informationen zur Verwendung von Workflow Studio zum Erstellen Ihrer Zustandsmaschinen finden Sie unterVerwenden von Workflow Studio.

  > **Important**
  >
  > Denken Sie daran, den Platzhalter Amazon Resource Name (ARN) für die im Beispielprojekt verwendeten Ressourcen zu aktualisieren, bevor Sie Ihren Workflow ausführen.

- Wenn Sie Eine Demo ausführen ausgewählt haben, erstellt Step Functions ein schreibgeschütztes Beispielprojekt, das eine AWS CloudFormation Vorlage verwendet, um die in dieser Vorlage aufgeführten AWS Ressourcen für Ihr Projekt bereitzustellen. AWS-Konto

  > **Tip**
  >
  > Um die State-Machine-Definition des Beispielprojekts anzuzeigen, wählen Sie Code.

  Wenn Sie bereit sind, wählen Sie Deploy and run aus, um das Beispielprojekt bereitzustellen und die Ressourcen zu erstellen.

  Es kann bis zu 10 Minuten dauern, bis diese Ressourcen und die zugehörigen IAM-Berechtigungen erstellt sind. Während der Bereitstellung Ihrer Ressourcen können Sie den CloudFormation Stack-ID-Link öffnen, um zu sehen, welche Ressourcen bereitgestellt werden.

  Nachdem alle Ressourcen im Beispielprojekt erstellt wurden, wird das neue Beispielprojekt auf der Seite State Machines aufgeführt.

> **Important**
>
> Für jeden in der CloudFormation Vorlage verwendeten Dienst können Standardgebühren anfallen.

Schritt 2: Führen Sie die Zustandsmaschine aus

1. Wählen Sie auf der Seite State Machines Ihr Beispielprojekt aus.
2. Wählen Sie auf der Seite mit dem Beispielprojekt die Option Ausführung starten aus.
3. Gehen Sie im Dialogfeld Ausführung starten wie folgt vor:

   1. (Optional) Um Ihre Ausführung zu identifizieren, können Sie im Feld Name einen Namen dafür angeben. Standardmäßig generiert Step Functions automatisch einen eindeutigen Ausführungsnamen.

      > **Note**
      >
      > Mit Step Functions können Sie Namen für Zustandsmaschinen, Ausführungen und Aktivitäten sowie Beschriftungen erstellen, die Nicht-ASCII-Zeichen enthalten. Diese Nicht-ASCII-Namen funktionieren nicht mit Amazon. CloudWatch Um sicherzustellen, dass Sie CloudWatch Messwerte verfolgen können, wählen Sie einen Namen, der nur ASCII-Zeichen verwendet.

   2. (Optional) Geben Sie in das Eingabefeld Eingabewerte im JSON-Format ein, um Ihren Workflow auszuführen.

      Wenn Sie sich dafür entschieden haben, eine Demo auszuführen, müssen Sie keine Ausführungseingaben angeben.

      > **Note**
      >
      > Wenn das von Ihnen bereitgestellte Demo-Projekt vorab aufgefüllte Eingabedaten für die Ausführung enthält, verwenden Sie diese Eingabe, um die Zustandsmaschine auszuführen.

   3. Wählen Sie Start execution (Ausführung starten) aus.

4. Die Step Functions Functions-Konsole leitet Sie zu einer Seite weiter, die mit Ihrer Ausführungs-ID betitelt ist. Diese Seite wird als Seite mit den Ausführungsdetails bezeichnet. Auf dieser Seite können Sie die Ausführungsergebnisse im Verlauf der Ausführung oder nach deren Abschluss überprüfen.

   Um die Ausführungsergebnisse zu überprüfen, wählen Sie in der Diagrammansicht einzelne Status aus und wählen Sie dann die einzelnen Registerkarten im Schrittdetails Bereich, um die Details der einzelnen Status, einschließlich Eingabe, Ausgabe und Definition, anzuzeigen. Einzelheiten zu den Ausführungsinformationen, die Sie auf der Seite mit den Ausführungsdetails einsehen können, finden Sie unterSeite mit Ausführungsdetails – Schnittstellenübersicht.

Code des Zustandsautomaten aus diesem Beispiel

Die Zustandsmaschine in diesem Beispielprojekt lässt sich AWS Batch in Amazon SNS integrieren, indem Parameter direkt an diese Ressourcen übergeben werden.

Sehen Sie sich diese Beispiel-Zustandsmaschine an, um zu sehen, wie Step Functions AWS Batch und Amazon SNS steuern, indem Sie eine Verbindung zum Amazon-Ressourcennamen (ARN) im `Resource` Feld herstellen und `Parameters` an die Service-API übergeben.

Weitere Informationen darüber, wie AWS Step Functions Sie andere AWS Dienste steuern können, finden Sie unterVerwendung AWS Step Functions mit anderen Diensten.

```
{
  "Comment": "An example of the Amazon States Language for notification on an AWS Batch
 job completion",
  "StartAt": "Submit Batch Job",
  "TimeoutSeconds": 3600,
  "States": {
    "Submit Batch Job": {
      "Type": "Task",
      "Resource": "arn:aws:states:::batch:submitJob.sync",
      "Parameters": {
        "JobName": "BatchJobNotification",
        "JobQueue": "arn:aws:batch:us-east-1:123456789012:job-queue/
BatchJobQueue-7049d367474b4dd",
        "JobDefinition": "arn:aws:batch:us-east-1:123456789012:job-definition/
BatchJobDefinition-74d55ec34c4643c:1"
      },
```

```
      "Next": "Notify Success",
      "Catch": [
          {
            "ErrorEquals": [ "States.ALL" ],
            "Next": "Notify Failure"
          }
      ]
    },
    "Notify Success": {
      "Type": "Task",
      "Resource": "arn:aws:states:::sns:publish",
      "Parameters": {
        "Message": "Batch job submitted through Step Functions succeeded",
        "TopicArn": "arn:aws:sns:us-east-1:123456789012:batchjobnotificatiointemplate-
SNSTopic-1J757CVBQ2KHM"
      },
      "End": true
    },
    "Notify Failure": {
      "Type": "Task",
      "Resource": "arn:aws:states:::sns:publish",
      "Parameters": {
        "Message": "Batch job submitted through Step Functions failed",
        "TopicArn": "arn:aws:sns:us-east-1:123456789012:batchjobnotificatiointemplate-
SNSTopic-1J757CVBQ2KHM"
      },
      "End": true
    }
  }
}
```

IAM-Beispiel

Diese vom Beispielprojekt generierte Beispielrichtlinie AWS Identity and Access Management (IAM) beinhaltet die geringsten Rechte, die für die Ausführung der Zustandsmaschine und der zugehörigen Ressourcen erforderlich sind. Wir empfehlen, dass Sie nur die Berechtigungen in Ihre IAM-Richtlinien aufnehmen, die erforderlich sind.

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

Informationen zur Konfiguration von IAM bei der Verwendung von Step Functions mit anderen AWS Diensten finden Sie unterIAM-Richtlinien für integrierte Dienste.

Eine Container-Aufgabe verwalten (Amazon ECS,Amazon SNS)

Dieses Beispielprojekt zeigt, wie eine AWS Fargate Aufgabe ausgeführt und anschließend eine Amazon SNS Benachrichtigung gesendet wird, je nachdem, ob die Aufgabe erfolgreich war oder nicht. Durch die Bereitstellung dieses Beispielprojekts werden ein AWS Step Functions Zustandsmaschine, ein Fargate Cluster und ein Amazon SNS Thema erstellt.

In diesem Projekt wird eine Zustandsmaschine Step Functions verwendet, um die Fargate Aufgabe synchron aufzurufen. Anschließend wartet Step Functions darauf, dass die Aufgabe erfolgreich abgeschlossen wird oder fehlschlägt, und sendet ein Amazon SNS-Thema mit einer Meldung, ob der Auftrag erfolgreich abgeschlossen wurde oder fehlgeschlagen ist.

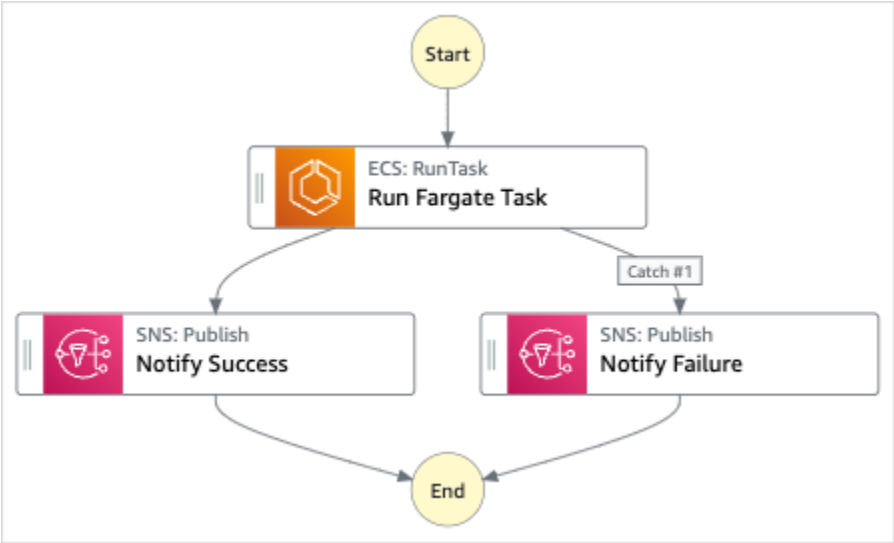
Schritt 1: Erstellen Sie die Zustandsmaschine und stellen Sie Ressourcen bereit

1. Öffnen Sie die Step Functions Functions-Konsole und wählen Sie Create State Machine.
2. Geben Sie **`Manage a container task`** etwas in das Suchfeld ein und wählen Sie dann aus den zurückgegebenen Suchergebnissen die Option Container-Aufgabe verwalten aus.
3. Wählen Sie Next (Weiter), um fortzufahren.
4. Step Functions listet die Funktionen auf, die in dem von Ihnen ausgewählten Beispielprojekt AWS-Services verwendet wurden. Außerdem wird ein Workflow-Diagramm für das Beispielprojekt angezeigt. Stellen Sie dieses Projekt für Ihr Projekt bereit AWS-Konto oder verwenden Sie es als Ausgangspunkt für die Erstellung Ihrer eigenen Projekte. Je nachdem, wie Sie vorgehen möchten, wählen Sie „Demo ausführen“ oder „Darauf aufbauen“.

   In diesem Beispielprojekt werden die folgenden Ressourcen bereitgestellt:

   - Ein Cluster AWS Fargate
   - Amazon-SNS-Thema
   - Eine AWS Step Functions Zustandsmaschine
   - Verwandte AWS Identity and Access Management Rollen (IAM)

   Die folgende Abbildung zeigt das Workflow-Diagramm für das Beispielprojekt „Container verwalten“:



5. Wählen Sie Vorlage verwenden, um mit Ihrer Auswahl fortzufahren.
6. Führen Sie eine der folgenden Aktionen aus:

   - Wenn Sie „Darauf aufbauen“ ausgewählt haben, erstellt Step Functions den Workflow-Prototyp für das von Ihnen ausgewählte Beispielprojekt. Step Functions stellt die in der Workflow-Definition aufgeführten Ressourcen nicht bereit.

     Ziehen Sie in Workflow Studio Status per Drag-and-Drop aus demEntwurfsmodus, Bundesstaaten-Browser um mit der Erstellung Ihres Workflow-Prototyps fortzufahren. Oder wechseln Sie zu demCodemodus, der einen integrierten Code-Editor bietet, der VS Code ähnelt, um die Amazon States Language (ASL-) Definition Ihrer Zustandsmaschine in der Step Functions Functions-Konsole zu aktualisieren. Weitere Informationen zur Verwendung von Workflow Studio zum Erstellen Ihrer Zustandsmaschinen finden Sie unterVerwenden von Workflow Studio.

     > **Important**
     >
     > Denken Sie daran, den Platzhalter Amazon Resource Name (ARN) für die im Beispielprojekt verwendeten Ressourcen zu aktualisieren, bevor Sie Ihren Workflow ausführen.

   - Wenn Sie Eine Demo ausführen ausgewählt haben, erstellt Step Functions ein schreibgeschütztes Beispielprojekt, das eine AWS CloudFormation Vorlage verwendet, um die in dieser Vorlage aufgeführten AWS Ressourcen für Ihr Projekt bereitzustellen. AWS-Konto

> **Tip**
>
> Um die State-Machine-Definition des Beispielprojekts anzuzeigen, wählen Sie Code.

Wenn Sie bereit sind, wählen Sie Deploy and run aus, um das Beispielprojekt bereitzustellen und die Ressourcen zu erstellen.

Es kann bis zu 10 Minuten dauern, bis diese Ressourcen und die zugehörigen IAM-Berechtigungen erstellt sind. Während der Bereitstellung Ihrer Ressourcen können Sie den CloudFormation Stack-ID-Link öffnen, um zu sehen, welche Ressourcen bereitgestellt werden.

Nachdem alle Ressourcen im Beispielprojekt erstellt wurden, wird das neue Beispielprojekt auf der Seite State Machines aufgeführt.

> **Important**
>
> Für jeden in der CloudFormation Vorlage verwendeten Dienst können Standardgebühren anfallen.

Schritt 2: Führen Sie die Zustandsmaschine aus

1. Wählen Sie auf der Seite State Machines Ihr Beispielprojekt aus.
2. Wählen Sie auf der Seite mit dem Beispielprojekt die Option Ausführung starten aus.
3. Gehen Sie im Dialogfeld Ausführung starten wie folgt vor:

    1. (Optional) Um Ihre Ausführung zu identifizieren, können Sie im Feld Name einen Namen dafür angeben. Standardmäßig generiert Step Functions automatisch einen eindeutigen Ausführungsnamen.

        > **Note**
        >
        > Mit Step Functions können Sie Namen für Zustandsmaschinen, Ausführungen und Aktivitäten sowie Beschriftungen erstellen, die Nicht-ASCII-Zeichen enthalten. Diese Nicht-ASCII-Namen funktionieren nicht mit Amazon. CloudWatch Um sicherzustellen,

dass Sie CloudWatch Messwerte verfolgen können, wählen Sie einen Namen, der nur ASCII-Zeichen verwendet.

2. (Optional) Geben Sie in das Eingabefeld Eingabewerte im JSON-Format ein, um Ihren Workflow auszuführen.

   Wenn Sie sich dafür entschieden haben, eine Demo auszuführen, müssen Sie keine Ausführungseingaben angeben.

   > **Note**
   >
   > Wenn das von Ihnen bereitgestellte Demo-Projekt vorab aufgefüllte Eingabedaten für die Ausführung enthält, verwenden Sie diese Eingabe, um die Zustandsmaschine auszuführen.

3. Wählen Sie Start execution (Ausführung starten) aus.
4. Die Step Functions Functions-Konsole leitet Sie zu einer Seite weiter, die mit Ihrer Ausführungs-ID betitelt ist. Diese Seite wird als Seite mit den Ausführungsdetails bezeichnet. Auf dieser Seite können Sie die Ausführungsergebnisse im Verlauf der Ausführung oder nach deren Abschluss überprüfen.

   Um die Ausführungsergebnisse zu überprüfen, wählen Sie in der Diagrammansicht einzelne Status aus und wählen Sie dann die einzelnen Registerkarten im Schrittdetails Bereich, um die Details der einzelnen Status, einschließlich Eingabe, Ausgabe und Definition, anzuzeigen. Einzelheiten zu den Ausführungsinformationen, die Sie auf der Seite mit den Ausführungsdetails einsehen können, finden Sie unterSeite mit Ausführungsdetails – Schnittstellenübersicht.

Code des Zustandsautomaten aus diesem Beispiel

Die Zustandsmaschine in diesem Beispielprojekt lässt sich AWS Fargate in Amazon SNS integrieren, indem Parameter direkt an diese Ressourcen übergeben werden. Sehen Sie sich diese Beispiel-Zustandsmaschine an, um zu sehen, wie Step Functions eine Zustandsmaschine verwendet, um die Fargate-Aufgabe synchron aufzurufen, auf den Erfolg oder Misserfolg der Aufgabe wartet und ein Amazon SNS SNS-Thema mit einer Meldung darüber sendet, ob der Job erfolgreich war oder nicht.

Weitere Informationen darüber, wie Sie andere AWS Dienste steuern AWS Step Functions können, finden Sie unter. Verwendung AWS Step Functions mit anderen Diensten

```
{
  "Comment": "An example of the Amazon States Language for notification on an AWS
 Fargate task completion",
  "StartAt": "Run Fargate Task",
  "TimeoutSeconds": 3600,
  "States": {
    "Run Fargate Task": {
      "Type": "Task",
      "Resource": "arn:aws:states:::ecs:runTask.sync",
      "Parameters": {
        "LaunchType": "FARGATE",
        "Cluster": "arn:aws:ecs:ap-northeast-1:123456789012:cluster/
FargateTaskNotification-ECSCluster-VHLR20IF9IMP",
        "TaskDefinition": "arn:aws:ecs:ap-northeast-1:123456789012:task-definition/
FargateTaskNotification-ECSTaskDefinition-13YOJT8Z2LY5Q:1",
        "NetworkConfiguration": {
          "AwsvpcConfiguration": {
            "Subnets": [
              "subnet-07e1ad3abcfce6758",
              "subnet-04782e7f34ae3efdb"
            ],
            "AssignPublicIp": "ENABLED"
          }
        }
      },
      "Next": "Notify Success",
      "Catch": [
          {
            "ErrorEquals": [ "States.ALL" ],
            "Next": "Notify Failure"
          }
      ]
    },
    "Notify Success": {
      "Type": "Task",
      "Resource": "arn:aws:states:::sns:publish",
      "Parameters": {
        "Message": "AWS Fargate Task started by Step Functions succeeded",
        "TopicArn": "arn:aws:sns:ap-northeast-1:123456789012:FargateTaskNotification-
SNSTopic-1XYW5YD5V0M7C"
      },
      "End": true
    },
```

```
    "Notify Failure": {
      "Type": "Task",
      "Resource": "arn:aws:states:::sns:publish",
      "Parameters": {
        "Message": "AWS Fargate Task started by Step Functions failed",
        "TopicArn": "arn:aws:sns:ap-northeast-1:123456789012:FargateTaskNotification-
SNSTopic-1XYW5YD5V0M7C"
      },
      "End": true
    }
  }
}
```

IAM-Beispiel

Diese vom Beispielprojekt generierte Beispielrichtlinie AWS Identity and Access Management (IAM) beinhaltet die geringsten Rechte, die für die Ausführung der Zustandsmaschine und der zugehörigen Ressourcen erforderlich sind. Es hat sich bewährt, nur die Berechtigungen in Ihre IAM-Richtlinien aufzunehmen, die erforderlich sind.

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

Informationen zur Konfiguration von IAM bei der Verwendung von Step Functions mit anderen AWS Diensten finden Sie unterIAM-Richtlinien für integrierte Dienste.

Datensätze übertragen (Lambda,DynamoDB,Amazon SQS)

Dieses Beispielprojekt zeigt, wie iterativ Elemente aus einer Amazon DynamoDB Tabelle gelesen und diese Elemente mithilfe eines Step Functions Zustandsautomaten an eine Amazon SQS Warteschlange gesendet werden. Durch die Bereitstellung dieses Beispielprojekts werden ein Step Functions Zustandsmaschine, eine DynamoDB Tabelle, eine AWS Lambda Funktion und eine Amazon SQS Warteschlange erstellt.

In diesem Projekt wird die Lambda Funktion Step Functions verwendet, um die DynamoDB Tabelle zu füllen. Die Zustandsmaschine verwendet außerdem eine `for` Schleife, um jeden der Einträge zu lesen, und sendet dann jeden Eintrag in eine Amazon SQS Warteschlange.

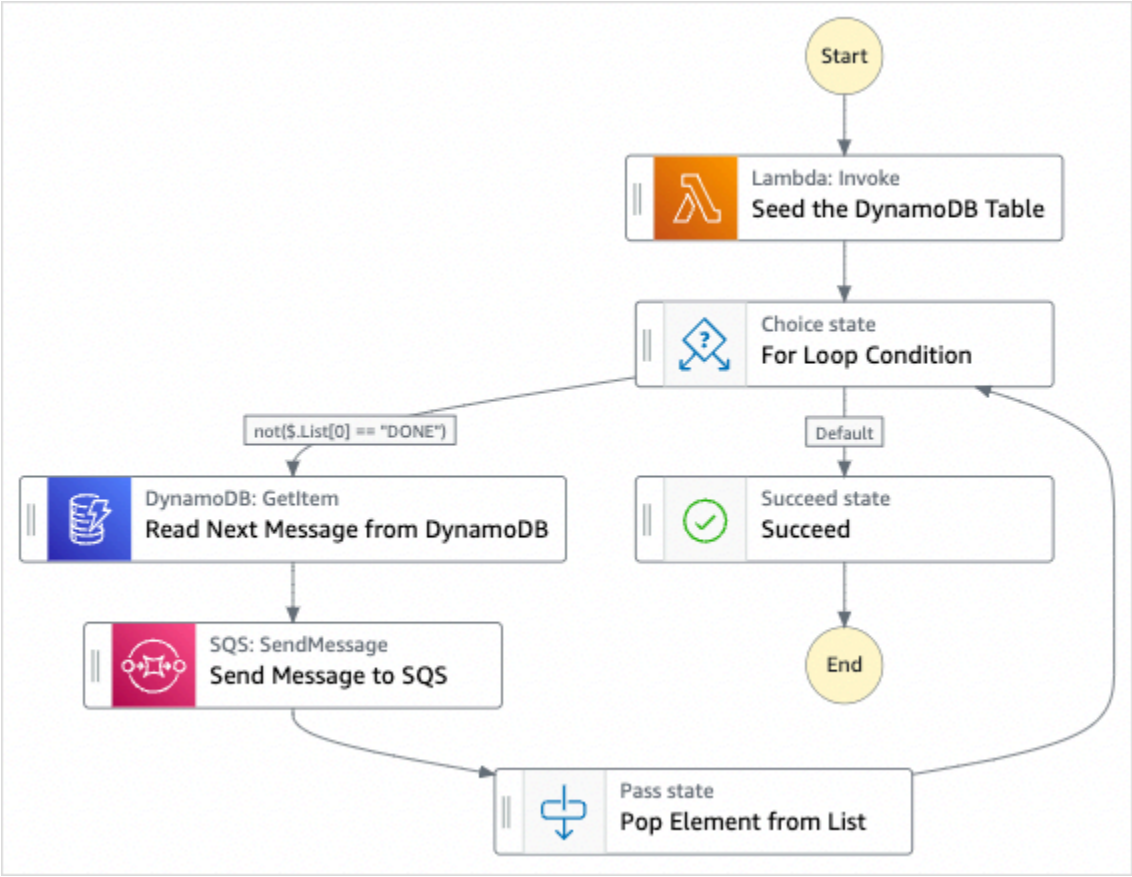
Schritt 1: Erstellen Sie die Zustandsmaschine und stellen Sie Ressourcen bereit

1. Öffnen Sie die Step Functions Functions-Konsole und wählen Sie Create State Machine.
2. Geben Sie etwas **`Transfer data records`** in das Suchfeld ein und wählen Sie dann Datensätze aus den zurückgegebenen Suchergebnissen übertragen aus.
3. Wählen Sie Next (Weiter), um fortzufahren.
4. Step Functions listet die Funktionen auf, die in dem von Ihnen ausgewählten Beispielprojekt AWS-Services verwendet wurden. Außerdem wird ein Workflow-Diagramm für das Beispielprojekt angezeigt. Stellen Sie dieses Projekt für Ihr Projekt bereit AWS-Konto oder verwenden Sie es als Ausgangspunkt für die Erstellung Ihrer eigenen Projekte. Je nachdem, wie Sie vorgehen möchten, wählen Sie „Demo ausführen“ oder „Darauf aufbauen“.

   In diesem Beispielprojekt werden die folgenden Ressourcen bereitgestellt:

   - Eine Lambda-Funktion zum Seeding der DynamoDB-Tabelle
   - Eine Amazon-SQS-Warteschlange
   - Eine DynamoDB-Tabelle
   - AWS Step Functions Eine Zustandsmaschine
   - Verwandte AWS Identity and Access Management Rollen (IAM)

   Die folgende Abbildung zeigt das Workflow-Diagramm für das Beispielprojekt Datensätze übertragen:



5. Wählen Sie Vorlage verwenden, um mit Ihrer Auswahl fortzufahren.
6. Führen Sie eine der folgenden Aktionen aus:

   - Wenn Sie „Darauf aufbauen“ ausgewählt haben, erstellt Step Functions den Workflow-Prototyp für das von Ihnen ausgewählte Beispielprojekt. Step Functions stellt die in der Workflow-Definition aufgeführten Ressourcen nicht bereit.

     Ziehen Sie in Workflow Studio Status per Drag-and-Drop aus demEntwurfsmodus, Bundesstaaten-Browser um mit der Erstellung Ihres Workflow-Prototyps fortzufahren. Oder wechseln Sie zu demCodemodus, der einen integrierten Code-Editor bietet, der VS Code ähnelt, um die Amazon States Language (ASL-) Definition Ihrer Zustandsmaschine in der Step Functions Functions-Konsole zu aktualisieren. Weitere Informationen zur Verwendung von Workflow Studio zum Erstellen Ihrer Zustandsmaschinen finden Sie unterVerwenden von Workflow Studio.

> **Important**
>
> Denken Sie daran, den Platzhalter Amazon Resource Name (ARN) für die im Beispielprojekt verwendeten Ressourcen zu aktualisieren, bevor Sie Ihren Workflow ausführen.

- Wenn Sie Eine Demo ausführen ausgewählt haben, erstellt Step Functions ein schreibgeschütztes Beispielprojekt, das eine AWS CloudFormation Vorlage verwendet, um die in dieser Vorlage aufgeführten AWS Ressourcen für Ihr Projekt bereitzustellen. AWS-Konto

  > **Tip**
  >
  > Um die State-Machine-Definition des Beispielprojekts anzuzeigen, wählen Sie Code.

  Wenn Sie bereit sind, wählen Sie Deploy and run aus, um das Beispielprojekt bereitzustellen und die Ressourcen zu erstellen.

  Es kann bis zu 10 Minuten dauern, bis diese Ressourcen und die zugehörigen IAM-Berechtigungen erstellt sind. Während der Bereitstellung Ihrer Ressourcen können Sie den CloudFormation Stack-ID-Link öffnen, um zu sehen, welche Ressourcen bereitgestellt werden.

  Nachdem alle Ressourcen im Beispielprojekt erstellt wurden, wird das neue Beispielprojekt auf der Seite State Machines aufgeführt.

  > **Important**
  >
  > Für jeden in der CloudFormation Vorlage verwendeten Dienst können Standardgebühren anfallen.

Schritt 2: Führen Sie die Zustandsmaschine aus

1. Wählen Sie auf der Seite State Machines Ihr Beispielprojekt aus.
2. Wählen Sie auf der Seite mit dem Beispielprojekt die Option Ausführung starten aus.
3. Gehen Sie im Dialogfeld Ausführung starten wie folgt vor:

1. (Optional) Um Ihre Ausführung zu identifizieren, können Sie im Feld Name einen Namen dafür angeben. Standardmäßig generiert Step Functions automatisch einen eindeutigen Ausführungsnamen.

   > **Note**
   >
   > Mit Step Functions können Sie Namen für Zustandsmaschinen, Ausführungen und Aktivitäten sowie Beschriftungen erstellen, die Nicht-ASCII-Zeichen enthalten. Diese Nicht-ASCII-Namen funktionieren nicht mit Amazon. CloudWatch Um sicherzustellen, dass Sie CloudWatch Messwerte verfolgen können, wählen Sie einen Namen, der nur ASCII-Zeichen verwendet.

2. (Optional) Geben Sie in das Eingabefeld Eingabewerte im JSON-Format ein, um Ihren Workflow auszuführen.

   Wenn Sie sich dafür entschieden haben, eine Demo auszuführen, müssen Sie keine Ausführungseingaben angeben.

   > **Note**
   >
   > Wenn das von Ihnen bereitgestellte Demo-Projekt vorab aufgefüllte Eingabedaten für die Ausführung enthält, verwenden Sie diese Eingabe, um die Zustandsmaschine auszuführen.

3. Wählen Sie Start execution (Ausführung starten) aus.
4. Die Step Functions Functions-Konsole leitet Sie zu einer Seite weiter, die mit Ihrer Ausführungs-ID betitelt ist. Diese Seite wird als Seite mit den Ausführungsdetails bezeichnet. Auf dieser Seite können Sie die Ausführungsergebnisse im Verlauf der Ausführung oder nach deren Abschluss überprüfen.

   Um die Ausführungsergebnisse zu überprüfen, wählen Sie in der Diagrammansicht einzelne Status aus und wählen Sie dann die einzelnen Registerkarten im Schrittdetails Bereich, um die Details der einzelnen Status, einschließlich Eingabe, Ausgabe und Definition, anzuzeigen. Einzelheiten zu den Ausführungsinformationen, die Sie auf der Seite mit den Ausführungsdetails einsehen können, finden Sie unterSeite mit Ausführungsdetails – Schnittstellenübersicht.

Code des Zustandsautomaten aus diesem Beispiel

Die Zustandsmaschine in diesem Beispielprojekt lässt sich in DynamoDB und Amazon SQS integrieren, indem Parameter direkt an diese Ressourcen übergeben werden.

Sehen Sie sich diese Beispiel-Zustandsmaschine an, um zu sehen, wie Step Functions DynamoDB und Amazon SQS steuert, indem es eine Verbindung zum Amazon-Ressourcennamen (ARN) im `Resource` Feld herstellt und `Parameters` an die Service-API weiterleitet.

Weitere Informationen darüber, wie Sie andere AWS Dienste steuern AWS Step Functions können, finden Sie unter. Verwendung AWS Step Functions mit anderen Diensten

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

Weitere Informationen zum Übergeben von Parametern und Verwalten von Ergebnissen finden Sie im Folgenden:

- Parameter an eine Service-API übergeben
- ResultPath

IAM-Beispiel

Diese vom Beispielprojekt generierte Beispielrichtlinie AWS Identity and Access Management (IAM) beinhaltet die geringsten Rechte, die für die Ausführung der Zustandsmaschine und der zugehörigen

Ressourcen erforderlich sind. Es hat sich bewährt, nur die Berechtigungen in Ihre IAM-Richtlinien aufzunehmen, die erforderlich sind.

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

Informationen zur Konfiguration von IAM bei der Verwendung von Step Functions mit anderen AWS Diensten finden Sie unterIAM-Richtlinien für integrierte Dienste.

Umfrage zum Jobstatus (Lambda, AWS Batch)

In diesem Beispielprojekt wird ein AWS Batch Job-Poller erstellt. Es implementiert eine AWS Step Functions Zustandsmaschine, die verwendet wird, AWS Lambda um eine `Wait` Statusschleife zu erstellen, die einen AWS Batch Job überprüft.

In diesem Beispielprojekt werden alle Ressourcen so erstellt und konfiguriert, dass Ihr Step Functions Functions-Workflow einen AWS Batch Job weiterleitet und wartet, bis dieser Job abgeschlossen ist, bevor er erfolgreich beendet wird.

> **Note**
>
> Sie können dieses Muster auch implementieren, ohne eine Lambda-Funktion zu verwenden. Hinweise zur AWS Batch direkten Steuerung finden Sie unterVerwendung AWS Step Functions mit anderen Diensten.

Dieses Beispielprojekt erstellt die Zustandsmaschine, zwei Lambda-Funktionen und eine AWS Batch Warteschlange und konfiguriert die zugehörigen IAM-Berechtigungen.

Weitere Informationen darüber, wie Sie andere AWS Dienste steuern AWS Step Functions können, finden Sie unter. Verwendung AWS Step Functions mit anderen Diensten

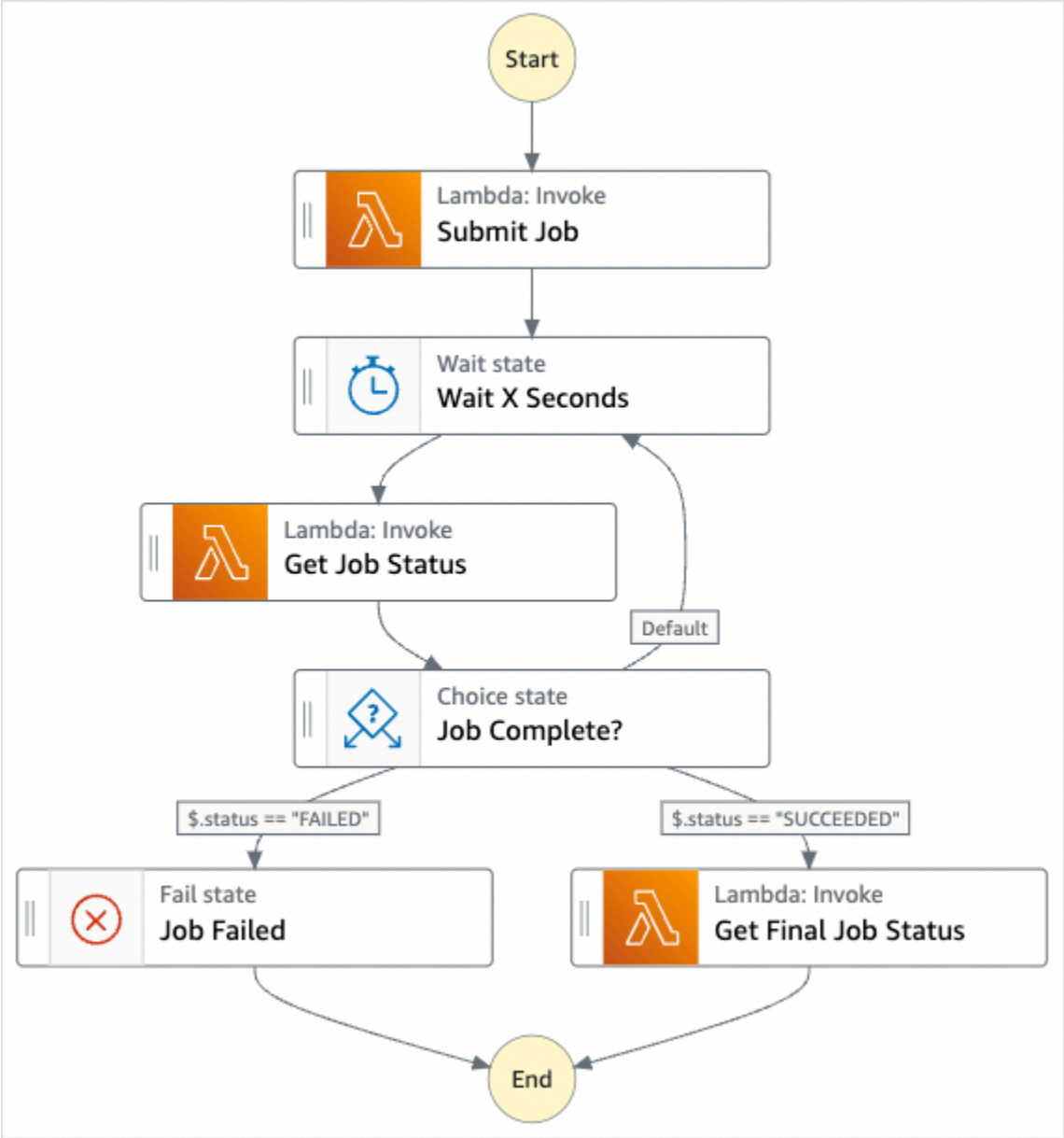
Schritt 1: Erstellen Sie den Zustandsmaschine und stellen Sie Ressourcen bereit

1. Öffnen Sie die Step Functions Functions-Konsole und wählen Sie Create State Machine.
2. Geben Sie **`Job Poller`** etwas in das Suchfeld ein, und wählen Sie dann Job Poller aus den zurückgegebenen Suchergebnissen aus.
3. Wählen Sie Next (Weiter), um fortzufahren.
4. Step Functions listet die Funktionen auf, die in dem von Ihnen ausgewählten Beispielprojekt AWS-Services verwendet wurden. Außerdem wird ein Workflow-Diagramm für das Beispielprojekt angezeigt. Stellen Sie dieses Projekt für Ihr Projekt bereit AWS-Konto oder verwenden Sie es als Ausgangspunkt für die Erstellung Ihrer eigenen Projekte. Je nachdem, wie Sie vorgehen möchten, wählen Sie „Demo ausführen“ oder „Darauf aufbauen“.

   In diesem Beispielprojekt werden die folgenden Ressourcen bereitgestellt:

- Drei Lambda-Funktionen zum Senden eines AWS Batch Jobs, zum Abrufen des aktuellen Status des übermittelten AWS Batch Jobs und zum endgültigen Abschlussstatus des Jobs.
- Ein Job AWS Batch
- Eine AWS Step Functions Staatsmaschine
- Verwandte AWS Identity and Access Management Rollen (IAM)

Die folgende Abbildung zeigt das Workflow-Diagramm für das Job Poller-Beispielprojekt:



5. Wählen Sie Vorlage verwenden, um mit Ihrer Auswahl fortzufahren.
6. Führen Sie eine der folgenden Aktionen aus:

- Wenn Sie „Darauf aufbauen“ ausgewählt haben, erstellt Step Functions den Workflow-Prototyp für das von Ihnen ausgewählte Beispielprojekt. Step Functions stellt die in der Workflow-Definition aufgeführten Ressourcen nicht bereit.

  Ziehen Sie in Workflow Studio Status per Drag-and-Drop aus demEntwurfsmodus, Bundesstaaten-Browser um mit der Erstellung Ihres Workflow-Prototyps fortzufahren. Oder wechseln Sie zu demCodemodus, der einen integrierten Code-Editor bietet, der VS Code ähnelt, um die Amazon States Language (ASL-) Definition Ihrer Zustandsmaschine in der Step Functions Functions-Konsole zu aktualisieren. Weitere Informationen zur Verwendung von Workflow Studio zum Erstellen Ihrer Zustandsmaschinen finden Sie unterVerwenden von Workflow Studio.

  > **Important**
  >
  > Denken Sie daran, den Platzhalter Amazon Resource Name (ARN) für die im Beispielprojekt verwendeten Ressourcen zu aktualisieren, bevor Sie Ihren Workflow ausführen.

- Wenn Sie Eine Demo ausführen ausgewählt haben, erstellt Step Functions ein schreibgeschütztes Beispielprojekt, das eine AWS CloudFormation Vorlage verwendet, um die in dieser Vorlage aufgeführten AWS Ressourcen für Ihr Projekt bereitzustellen. AWS-Konto

  > **Tip**
  >
  > Um die State-Machine-Definition des Beispielprojekts anzuzeigen, wählen Sie Code.

  Wenn Sie bereit sind, wählen Sie Deploy and run aus, um das Beispielprojekt bereitzustellen und die Ressourcen zu erstellen.

  Es kann bis zu 10 Minuten dauern, bis diese Ressourcen und die zugehörigen IAM-Berechtigungen erstellt sind. Während der Bereitstellung Ihrer Ressourcen können Sie den CloudFormation Stack-ID-Link öffnen, um zu sehen, welche Ressourcen bereitgestellt werden.

  Nachdem alle Ressourcen im Beispielprojekt erstellt wurden, wird das neue Beispielprojekt auf der Seite State Machines aufgeführt.

> **Important**
>
> Für jeden in der CloudFormation Vorlage verwendeten Dienst können Standardgebühren anfallen.

Schritt 2: Führen Sie die Zustandsmaschine aus

Nachdem alle Ressourcen bereitgestellt und bereitgestellt wurden, wird das Dialogfeld Ausführung starten mit einer Beispieleingabe angezeigt, die der folgenden ähnelt.

```
{
    "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` weist den `Wait`-Zustand an, alle 60 Sekunden eine Schleife auszuführen.

- Gehen Sie im Dialogfeld Ausführung starten wie folgt vor:

  1. (Optional) Um Ihre Ausführung zu identifizieren, können Sie im Feld Name einen Namen dafür angeben. Standardmäßig generiert Step Functions automatisch einen eindeutigen Ausführungsnamen.

     > **Note**
     >
     > Mit Step Functions können Sie Namen für Zustandsmaschinen, Ausführungen und Aktivitäten sowie Beschriftungen erstellen, die Nicht-ASCII-Zeichen enthalten. Diese Nicht-ASCII-Namen funktionieren nicht mit Amazon. CloudWatch Um sicherzustellen,

dass Sie CloudWatch Messwerte verfolgen können, wählen Sie einen Namen, der nur ASCII-Zeichen verwendet.

2. (Optional) Geben Sie in das Eingabefeld Eingabewerte im JSON-Format ein, um Ihren Workflow auszuführen.

   Wenn Sie sich dafür entschieden haben, eine Demo auszuführen, müssen Sie keine Ausführungseingaben angeben.

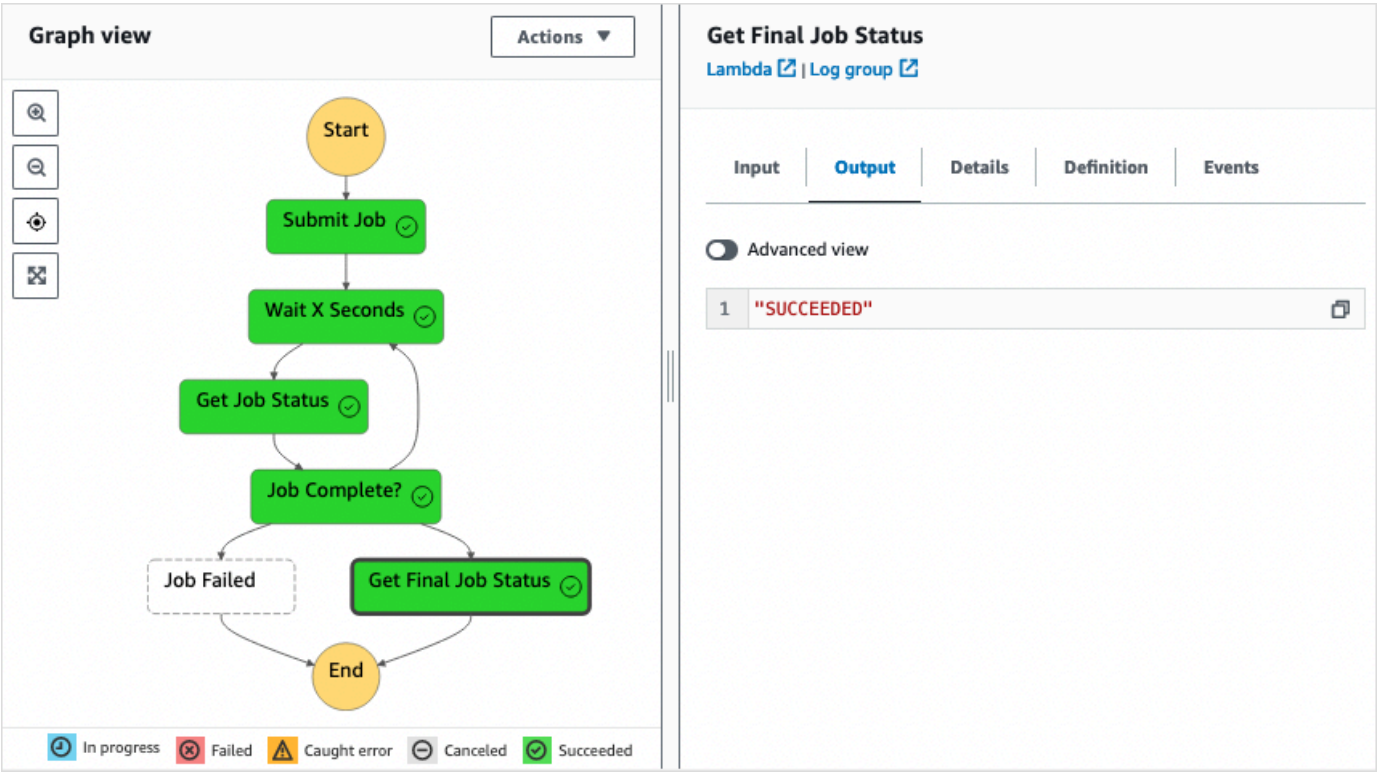
   > **Note**
   >
   > Wenn das von Ihnen bereitgestellte Demo-Projekt vorab aufgefüllte Eingabedaten für die Ausführung enthält, verwenden Sie diese Eingabe, um die Zustandsmaschine auszuführen.

3. Wählen Sie Start execution (Ausführung starten) aus.
4. Die Step Functions Functions-Konsole leitet Sie zu einer Seite weiter, die mit Ihrer Ausführungs-ID betitelt ist. Diese Seite wird als Seite mit den Ausführungsdetails bezeichnet. Auf dieser Seite können Sie die Ausführungsergebnisse im Verlauf der Ausführung oder nach deren Abschluss überprüfen.

   Um die Ausführungsergebnisse zu überprüfen, wählen Sie in der Diagrammansicht einzelne Status aus und wählen Sie dann die einzelnen Registerkarten im Schrittdetails Bereich, um die Details der einzelnen Status, einschließlich Eingabe, Ausgabe und Definition, anzuzeigen. Einzelheiten zu den Ausführungsinformationen, die Sie auf der Seite mit den Ausführungsdetails einsehen können, finden Sie unterSeite mit Ausführungsdetails – Schnittstellenübersicht.

Um beispielsweise den sich ändernden Status Ihres AWS Batch Jobs und die Loop-Ergebnisse Ihrer Ausführung anzuzeigen, wählen Sie die Registerkarte Ausgabe.

Die folgende Abbildung zeigt das Diagramm zum Ausführungsstatus in der Diagrammansicht. Es zeigt auch die Ausführungsausgabe für den ausgewählten Schritt auf der Registerkarte Ausgabe.



Code des Zustandsautomaten aus diesem Beispiel

Die Zustandsmaschine in diesem Beispielprojekt lässt sich integrieren AWS Lambda , um einen AWS Batch Job zu senden. Sehen Sie sich diese Beispiel-Zustandsmaschine an, um zu sehen, wie Step Functions Lambda und AWS Batch steuert.

Weitere Informationen darüber, wie AWS Step Functions Sie andere AWS Dienste steuern können, finden Sie unterVerwendung AWS Step Functions mit anderen Diensten.

```
{
    "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
      }
```

```
    }
  }
```

Aufgabentimer (Lambda, Amazon SNS)

Dieses Beispielprojekt erstellt einen Aufgabentimer. Es implementiert eine AWS Step Functions Zustandsmaschine, die einen `Wait` Status implementiert, und verwendet eine AWS Lambda Funktion, die eine Amazon Simple Notification Service (Amazon SNS) -Benachrichtigung sendet. Ein Wait-Zustand ist ein Zustandstyp, der auf einen Auslöser wartet, um eine einzelne Arbeitseinheit auszuführen.

> **Note**
>
> Dieses Beispielprojekt implementiert eine AWS Lambda Funktion zum Senden einer Amazon Simple Notification Service (Amazon SNS) -Benachrichtigung. Sie können eine Amazon SNS SNS-Benachrichtigung auch direkt aus der Sprache der Amazon-Staaten senden. Siehe Verwendung AWS Step Functions mit anderen Diensten.

Dieses Beispielprojekt erstellt die Zustandsmaschine, eine Lambda-Funktion und ein Amazon SNS SNS-Thema und konfiguriert die entsprechenden AWS Identity and Access Management (IAM-) Berechtigungen. Weitere Informationen zu den Ressourcen, die mit dem Beispielprojekt Task Timer (Aufgabentimer) erstellt werden, finden Sie im Folgenden:

Weitere Informationen darüber, wie Sie andere AWS Dienste steuern AWS Step Functions können, finden Sie unter. Verwendung AWS Step Functions mit anderen Diensten

- AWS CloudFormation Benutzerhandbuch
- Entwicklerhandbuch zu Amazon Simple Notification Service
- AWS Lambda Entwicklerhandbuch
- IAM – Handbuch Erste Schritte

Schritt 1: Erstellen Sie den Zustandsmaschine und stellen Sie Ressourcen bereit

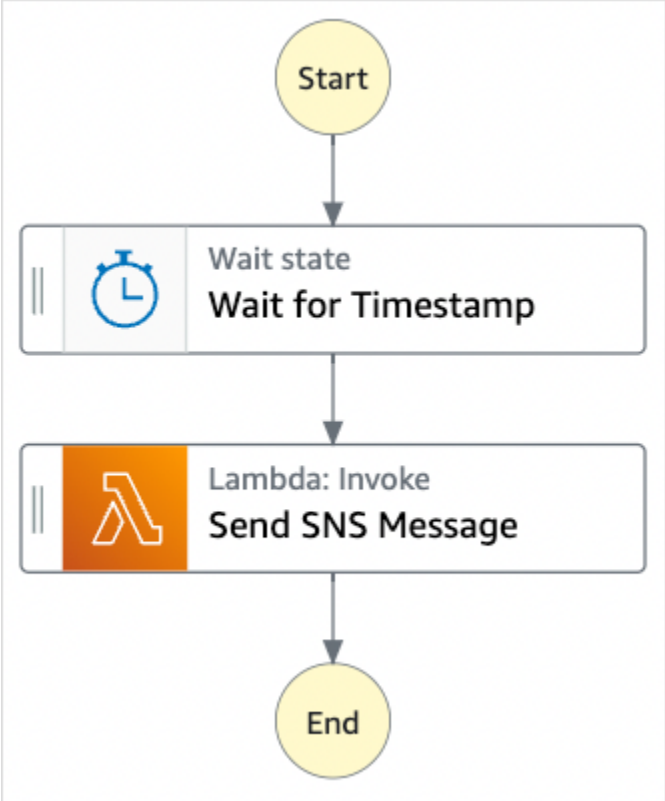
1. Öffnen Sie die Step Functions Functions-Konsole und wählen Sie Create State Machine.

2. Geben Sie **`Task Timer`** etwas in das Suchfeld ein und wählen Sie dann Task Timer aus den zurückgegebenen Suchergebnissen aus.
3. Wählen Sie Next (Weiter), um fortzufahren.
4. Step Functions listet die Funktionen auf, die in dem von Ihnen ausgewählten Beispielprojekt AWS-Services verwendet wurden. Außerdem wird ein Workflow-Diagramm für das Beispielprojekt angezeigt. Stellen Sie dieses Projekt für Sie bereit AWS-Konto oder verwenden Sie es als Ausgangspunkt für die Erstellung Ihrer eigenen Projekte. Je nachdem, wie Sie vorgehen möchten, wählen Sie „Demo ausführen“ oder „Darauf aufbauen“.

   Dieses Beispielprojekt stellt die folgenden Ressourcen bereit:

   - eine Lambda-Funktion, die eine Amazon SNS SNS-Benachrichtigung sendet.
   - Eine AWS Step Functions Zustandsmaschine
   - Verwandte AWS Identity and Access Management Rollen (IAM)

   Die folgende Abbildung zeigt das Workflow-Diagramm für das Task Timer-Beispielprojekt:



5. Wählen Sie Vorlage verwenden, um mit Ihrer Auswahl fortzufahren.
6. Führen Sie eine der folgenden Aktionen aus:

- Wenn Sie „Darauf aufbauen“ ausgewählt haben, erstellt Step Functions den Workflow-Prototyp für das von Ihnen ausgewählte Beispielprojekt. Step Functions stellt die in der Workflow-Definition aufgeführten Ressourcen nicht bereit.

  Ziehen Sie in Workflow Studio Status per Drag-and-Drop aus demEntwurfsmodus, Bundesstaaten-Browser um mit der Erstellung Ihres Workflow-Prototyps fortzufahren. Oder wechseln Sie zu demCodemodus, der einen integrierten Code-Editor bietet, der VS Code ähnelt, um die Amazon States Language (ASL-) Definition Ihrer Zustandsmaschine in der Step Functions Functions-Konsole zu aktualisieren. Weitere Informationen zur Verwendung von Workflow Studio zum Erstellen Ihrer Zustandsmaschinen finden Sie unterVerwenden von Workflow Studio.

  > **Important**
  >
  > Denken Sie daran, den Platzhalter Amazon Resource Name (ARN) für die im Beispielprojekt verwendeten Ressourcen zu aktualisieren, bevor Sie Ihren Workflow ausführen.

- Wenn Sie Eine Demo ausführen ausgewählt haben, erstellt Step Functions ein schreibgeschütztes Beispielprojekt, das eine AWS CloudFormation Vorlage verwendet, um die in dieser Vorlage aufgeführten AWS Ressourcen für Ihr Projekt bereitzustellen. AWS-Konto

  > **Tip**
  >
  > Um die State-Machine-Definition des Beispielprojekts anzuzeigen, wählen Sie Code.

  Wenn Sie bereit sind, wählen Sie Deploy and run aus, um das Beispielprojekt bereitzustellen und die Ressourcen zu erstellen.

  Es kann bis zu 10 Minuten dauern, bis diese Ressourcen und die zugehörigen IAM-Berechtigungen erstellt sind. Während der Bereitstellung Ihrer Ressourcen können Sie den CloudFormation Stack-ID-Link öffnen, um zu sehen, welche Ressourcen bereitgestellt werden.

  Nachdem alle Ressourcen im Beispielprojekt erstellt wurden, wird das neue Beispielprojekt auf der Seite State Machines aufgeführt.

> **Important**
>
> Für jeden in der CloudFormation Vorlage verwendeten Dienst können Standardgebühren anfallen.

Schritt 2: Führen Sie die Zustandsmaschine aus

Nachdem alle Ressourcen bereitgestellt und bereitgestellt wurden, wird das Dialogfeld Ausführung starten mit einer Beispieleingabe angezeigt, die der folgenden ähnelt.

```
{
    "jobName": "my-job",{
  "topic": "arn:aws:sns:us-east-2:123456789012:StepFunctionsSample-TaskTimercc68840e-c3d3-42a8-911e-821b7ce248e5-SNSTopic-44UjcFxzhACT",
  "message": "HelloWorld",
  "timer_seconds": 10
}
```

- Gehen Sie im Dialogfeld Ausführung starten wie folgt vor:

  1. (Optional) Um Ihre Ausführung zu identifizieren, können Sie im Feld Name einen Namen dafür angeben. Standardmäßig generiert Step Functions automatisch einen eindeutigen Ausführungsnamen.

     > **Note**
     >
     > Mit Step Functions können Sie Namen für Zustandsmaschinen, Ausführungen und Aktivitäten sowie Labels erstellen, die Nicht-ASCII-Zeichen enthalten. Diese Nicht-ASCII-Namen funktionieren nicht mit Amazon. CloudWatch Um sicherzustellen, dass Sie CloudWatch Messwerte verfolgen können, wählen Sie einen Namen, der nur ASCII-Zeichen verwendet.

  2. (Optional) Geben Sie in das Eingabefeld Eingabewerte im JSON-Format ein, um Ihren Workflow auszuführen.

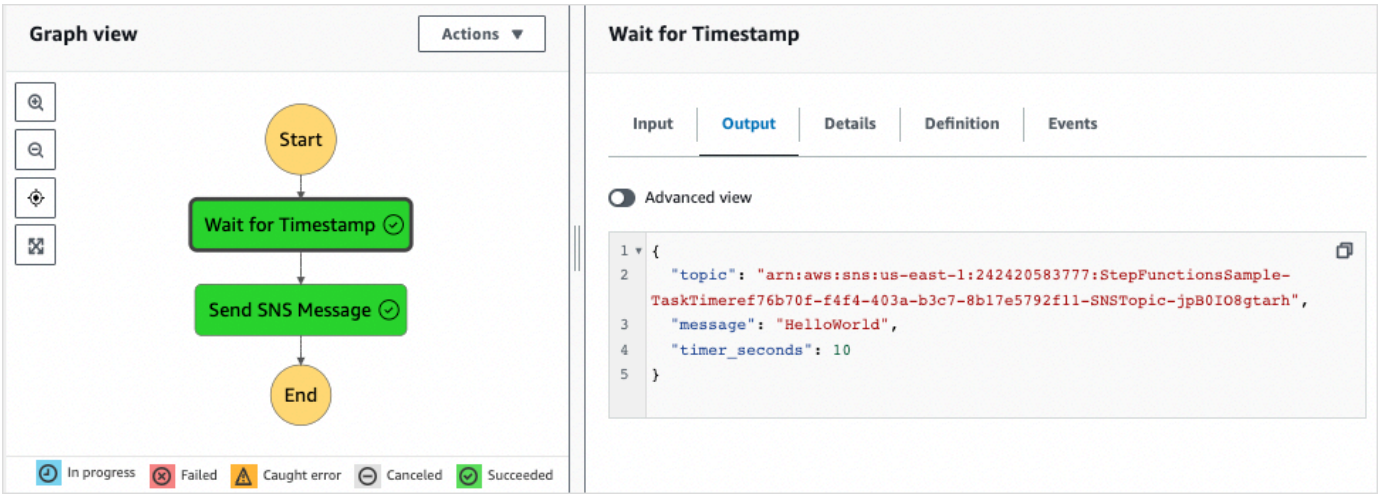
     Wenn Sie sich dafür entschieden haben, eine Demo auszuführen, müssen Sie keine Ausführungseingaben angeben.

> **Note**
>
> Wenn das von Ihnen bereitgestellte Demo-Projekt vorab aufgefüllte Eingabedaten für die Ausführung enthält, verwenden Sie diese Eingabe, um die Zustandsmaschine auszuführen.

3. Wählen Sie Start execution (Ausführung starten) aus.
4. Die Step Functions Functions-Konsole leitet Sie zu einer Seite weiter, die mit Ihrer Ausführungs-ID betitelt ist. Diese Seite wird als Seite mit den Ausführungsdetails bezeichnet. Auf dieser Seite können Sie die Ausführungsergebnisse im Verlauf oder nach Abschluss der Ausführung überprüfen.

   Um die Ausführungsergebnisse zu überprüfen, wählen Sie in der Diagrammansicht einzelne Status und dann die einzelnen Registerkarten im Schrittdetails Bereich aus, um die Details der einzelnen Status, einschließlich Eingabe, Ausgabe und Definition, anzuzeigen. Einzelheiten zu den Ausführungsinformationen, die Sie auf der Seite mit den Ausführungsdetails einsehen können, finden Sie unterSeite mit Ausführungsdetails – Schnittstellenübersicht.

Die folgende Abbildung zeigt beispielsweise die Ausgabe des ausgewählten Schritts Warten auf Zeitstempel. Die Ausgabe dieses Schritts wird als Eingabe an den Schritt SNS-Nachricht senden übergeben.



Beispiel für ein Rückrufmuster (Amazon SQS, Amazon SNS, Lambda)

Dieses Beispielprojekt zeigt, wie Sie während einer Aufgabe eine AWS Step Functions Pause einlegen und darauf warten können, dass ein externer Prozess ein Aufgaben-Token zurückgibt, das beim Start der Aufgabe generiert wurde.

Wenn dieses Beispielprojekt bereitgestellt wird und eine Ausführung gestartet wird, werden die folgenden Schritte ausgeführt:

1. Step Functions leitet eine Nachricht, die ein Aufgaben-Token enthält, an eine Amazon Simple Queue Service (Amazon SQS) -Warteschlange weiter.
2. Step Functions hält dann an und wartet darauf, dass das Token zurückgegeben wird.
3. Die Amazon SQS SQS-Warteschlange löst eine AWS Lambda Funktion aus, die `SendTaskSuccess`mit demselben Task-Token aufgerufen wird.
4. Wenn der Aufgabentoken empfangen wird, wird der Workflow fortgesetzt.
5. Die `"Notify Success"` Aufgabe veröffentlicht eine Amazon Simple Notification Service (Amazon SNS) -Meldung, dass der Rückruf empfangen wurde.

Informationen zur Implementierung des Callback-Musters in Step Functions finden Sie unterWarten auf einen Callback mit dem Aufgabentoken.

Weitere Informationen darüber, wie AWS Step Functions Sie andere AWS Dienste steuern können, finden Sie unterVerwendung AWS Step Functions mit anderen Diensten.

Schritt 1: Erstellen Sie den Zustandsmaschine und stellen Sie Ressourcen bereit

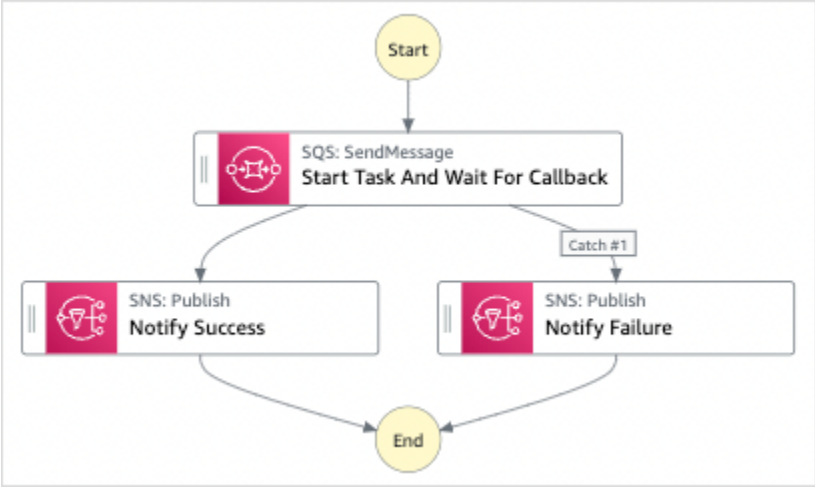
1. Öffnen Sie die Step Functions Functions-Konsole und wählen Sie Create State Machine.
2. Geben Sie **`Callback pattern example`** etwas in das Suchfeld ein und wählen Sie dann aus den zurückgegebenen Suchergebnissen ein Beispiel für ein Callback-Muster aus.
3. Wählen Sie Next (Weiter), um fortzufahren.
4. Step Functions listet die Funktionen auf, die in dem von Ihnen ausgewählten Beispielprojekt AWS-Services verwendet wurden. Außerdem wird ein Workflow-Diagramm für das Beispielprojekt angezeigt. Stellen Sie dieses Projekt für Sie bereit AWS-Konto oder verwenden

Sie es als Ausgangspunkt für die Erstellung Ihrer eigenen Projekte. Je nachdem, wie Sie vorgehen möchten, wählen Sie „Demo ausführen“ oder „Darauf aufbauen“.

Dieses Beispielprojekt stellt die folgenden Ressourcen bereit:

- Eine Amazon SQS SQS-Nachrichtenwarteschlange.
- Eine Lambda-Funktion, die die API-Aktion SendTaskSuccessStep Functions aufruft.
- Ein Amazon SNS SNS-Thema, das über den Erfolg oder Misserfolg einer Aufgabe informiert und angibt, ob der Workflow fortgesetzt werden kann oder nicht.
- Eine AWS Step Functions Zustandsmaschine
- Verwandte AWS Identity and Access Management Rollen (IAM)

Die folgende Abbildung zeigt das Workflow-Diagramm für das Beispielprojekt mit einem Callback-Muster:



5. Wählen Sie Vorlage verwenden, um mit Ihrer Auswahl fortzufahren.
6. Führen Sie eine der folgenden Aktionen aus:

   - Wenn Sie „Darauf aufbauen“ ausgewählt haben, erstellt Step Functions den Workflow-Prototyp für das von Ihnen ausgewählte Beispielprojekt. Step Functions stellt die in der Workflow-Definition aufgeführten Ressourcen nicht bereit.

     Ziehen Sie in Workflow Studio Status per Drag-and-Drop aus demEntwurfsmodus, Bundesstaaten-Browser um mit der Erstellung Ihres Workflow-Prototyps fortzufahren. Oder wechseln Sie zu demCodemodus, der einen integrierten Code-Editor bietet, der VS Code ähnelt, um die Amazon States Language (ASL-) Definition Ihrer Zustandsmaschine in der Step Functions Functions-Konsole zu aktualisieren. Weitere Informationen zur Verwendung

von Workflow Studio zum Erstellen Ihrer Zustandsmaschinen finden Sie unterVerwenden von Workflow Studio.

> **Important**
>
> Denken Sie daran, den Platzhalter Amazon Resource Name (ARN) für die im Beispielprojekt verwendeten Ressourcen zu aktualisieren, bevor Sie Ihren Workflow ausführen.

- Wenn Sie Eine Demo ausführen ausgewählt haben, erstellt Step Functions ein schreibgeschütztes Beispielprojekt, das eine AWS CloudFormation Vorlage verwendet, um die in dieser Vorlage aufgeführten AWS Ressourcen für Ihr Projekt bereitzustellen. AWS-Konto

  > **Tip**
  >
  > Um die State-Machine-Definition des Beispielprojekts anzuzeigen, wählen Sie Code.

  Wenn Sie bereit sind, wählen Sie Deploy and run aus, um das Beispielprojekt bereitzustellen und die Ressourcen zu erstellen.

  Es kann bis zu 10 Minuten dauern, bis diese Ressourcen und die zugehörigen IAM-Berechtigungen erstellt sind. Während der Bereitstellung Ihrer Ressourcen können Sie den CloudFormation Stack-ID-Link öffnen, um zu sehen, welche Ressourcen bereitgestellt werden.

  Nachdem alle Ressourcen im Beispielprojekt erstellt wurden, wird das neue Beispielprojekt auf der Seite State Machines aufgeführt.

  > **Important**
  >
  > Für jeden in der CloudFormation Vorlage verwendeten Dienst können Standardgebühren anfallen.

Schritt 2: Führen Sie die Zustandsmaschine aus

1. Wählen Sie auf der Seite State Machines Ihr Beispielprojekt aus.
2. Wählen Sie auf der Seite mit dem Beispielprojekt die Option Ausführung starten aus.

3. Gehen Sie im Dialogfeld Ausführung starten wie folgt vor:

   1. (Optional) Um Ihre Ausführung zu identifizieren, können Sie im Feld Name einen Namen dafür angeben. Standardmäßig generiert Step Functions automatisch einen eindeutigen Ausführungsnamen.

      > **Note**
      >
      > Mit Step Functions können Sie Namen für Zustandsmaschinen, Ausführungen und Aktivitäten sowie Labels erstellen, die Nicht-ASCII-Zeichen enthalten. Diese Nicht-ASCII-Namen funktionieren nicht mit Amazon. CloudWatch Um sicherzustellen, dass Sie CloudWatch Messwerte verfolgen können, wählen Sie einen Namen, der nur ASCII-Zeichen verwendet.

   2. (Optional) Geben Sie in das Eingabefeld Eingabewerte im JSON-Format ein, um Ihren Workflow auszuführen.

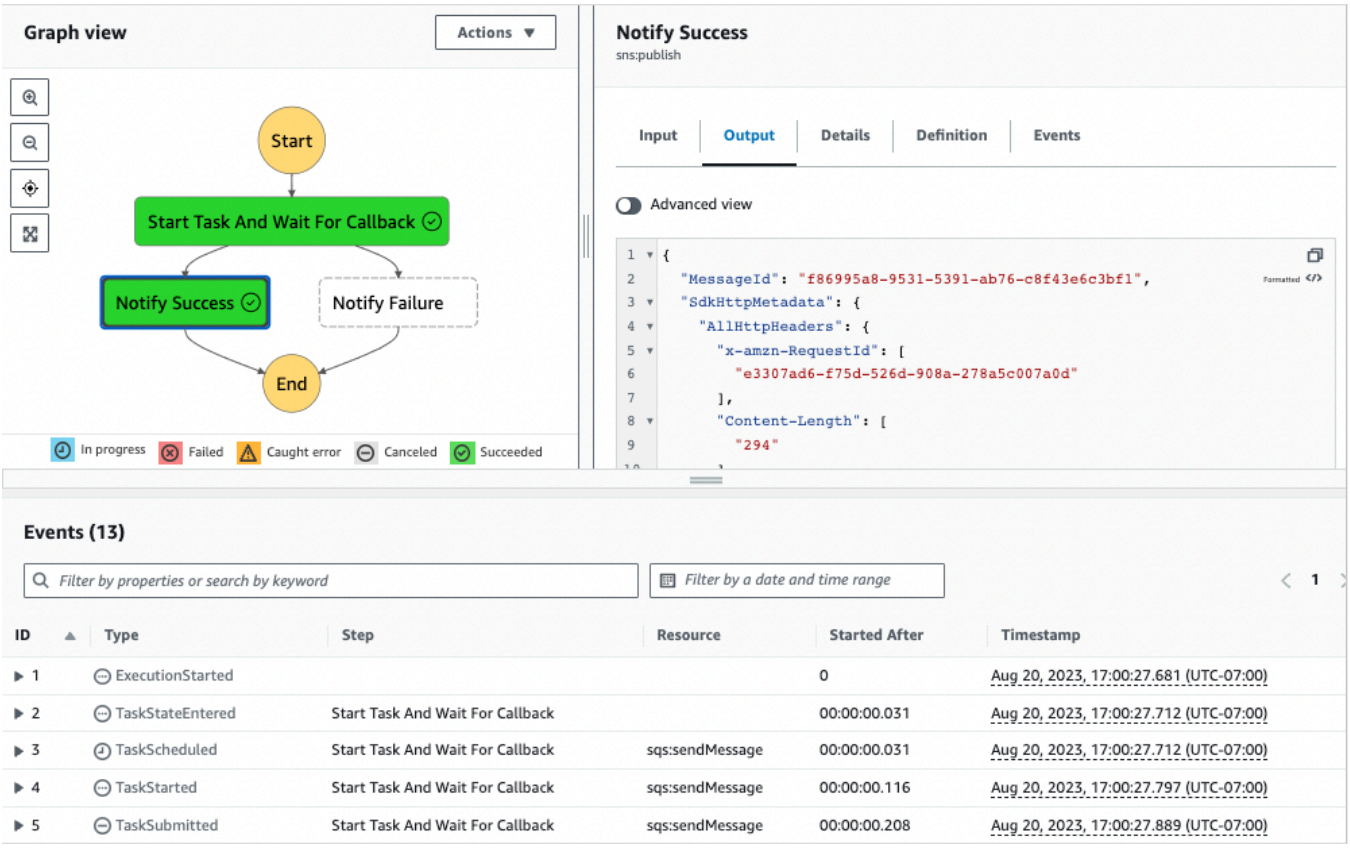
      Wenn Sie sich dafür entschieden haben, eine Demo auszuführen, müssen Sie keine Ausführungseingaben angeben.

      > **Note**
      >
      > Wenn das von Ihnen bereitgestellte Demo-Projekt vorab aufgefüllte Eingabedaten für die Ausführung enthält, verwenden Sie diese Eingabe, um die Zustandsmaschine auszuführen.

   3. Wählen Sie Start execution (Ausführung starten) aus.
   4. Die Step Functions Functions-Konsole leitet Sie zu einer Seite weiter, die mit Ihrer Ausführungs-ID betitelt ist. Diese Seite wird als Seite mit den Ausführungsdetails bezeichnet. Auf dieser Seite können Sie die Ausführungsergebnisse im Verlauf oder nach Abschluss der Ausführung überprüfen.

      Um die Ausführungsergebnisse zu überprüfen, wählen Sie in der Diagrammansicht einzelne Status und dann die einzelnen Registerkarten im Schrittdetails Bereich aus, um die Details der einzelnen Status, einschließlich Eingabe, Ausgabe und Definition, anzuzeigen. Einzelheiten zu den Ausführungsinformationen, die Sie auf der Seite mit den Ausführungsdetails einsehen können, finden Sie unterSeite mit Ausführungsdetails – Schnittstellenübersicht.

Um beispielsweise zu überprüfen, wie Step Functions den Workflow durchlief und einen Rückruf von Amazon SQS erhalten hat, überprüfen Sie die Einträge in der Tabelle Ereignisse. Die folgende Abbildung zeigt die Ausführungsausgabe für den Schritt „Erfolg benachrichtigen“. Außerdem werden die ersten fünf Ereignisse aus dem Verlauf der Ausführungsereignisse angezeigt. Erweitern Sie jedes Ereignis, um weitere Details zu diesem Ereignis anzuzeigen.



Beispiel für einen Lambda-Callback

Um zu sehen, wie die Komponenten dieses Beispielprojekts zusammenarbeiten, sehen Sie sich die Ressourcen an, die in Ihrem AWS Konto bereitgestellt wurden. Hier ist zum Beispiel die Lambda-Funktion, die Step Functions mit dem Task-Token aufruft.

```
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);
        });
    }
};
```

Einen Amazon EMR-Job verwalten

Dieses Beispielprojekt demonstriert Amazon EMR und die AWS Step Functions Integration.

Es zeigt, wie Sie einen Amazon EMR-Cluster erstellen, mehrere Schritte hinzufügen und diese ausführen und dann den Cluster beenden.

> **Important**
>
> Amazon EMR hat kein kostenloses Preiskontingent. Bei der Ausführung des Beispielprojekts fallen Kosten an. Preisinformationen finden Sie auf der Amazon EMR-Preisseite. Die Verfügbarkeit der Amazon EMR-Serviceintegration hängt von der Verfügbarkeit der Amazon EMR-APIs ab. Aus diesem Grund funktioniert dieses Beispielprojekt in einigen AWS Regionen möglicherweise nicht richtig. Informationen zu Einschränkungen in speziellen Regionen finden Sie in der Amazon EMR-Dokumentation.

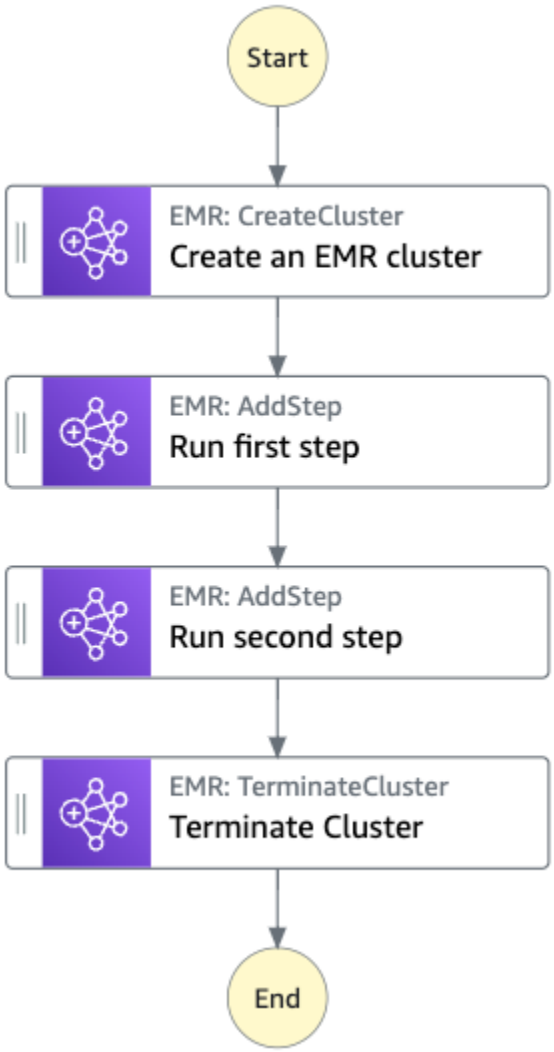
Schritt 1: Erstellen Sie den State Machine und stellen Sie Ressourcen bereit

1. Öffnen Sie die Step Functions Functions-Konsole und wählen Sie Create State Machine.
2. Geben Sie **`Manage an EMR job`** etwas in das Suchfeld ein, und wählen Sie dann aus den zurückgegebenen Suchergebnissen die Option EMR-Job verwalten aus.
3. Wählen Sie Next (Weiter), um fortzufahren.
4. Step Functions listet die Funktionen auf, die in dem von Ihnen ausgewählten Beispielprojekt AWS-Services verwendet wurden. Außerdem wird ein Workflow-Diagramm für das Beispielprojekt angezeigt. Stellen Sie dieses Projekt für Ihr Projekt bereit AWS-Konto oder verwenden Sie es als Ausgangspunkt für die Erstellung Ihrer eigenen Projekte. Je nachdem, wie Sie vorgehen möchten, wählen Sie „Demo ausführen“ oder „Darauf aufbauen“.

   In diesem Beispielprojekt werden die folgenden Ressourcen bereitgestellt:

   - Einen Amazon S3-Bucket
   - Ein Cluster Amazon EMR
   - Eine AWS Step Functions Zustandsmaschine
   - Verwandte AWS Identity and Access Management Rollen (IAM)

   Die folgende Abbildung zeigt das Workflow-Diagramm für das Beispielprojekt Manage an EMR:



5. Wählen Sie Vorlage verwenden, um mit Ihrer Auswahl fortzufahren.
6. Führen Sie eine der folgenden Aktionen aus:

   - Wenn Sie „Darauf aufbauen“ ausgewählt haben, erstellt Step Functions den Workflow-Prototyp für das von Ihnen ausgewählte Beispielprojekt. Step Functions stellt die in der Workflow-Definition aufgeführten Ressourcen nicht bereit.

     Ziehen Sie in Workflow Studio Status per Drag-and-Drop aus demEntwurfsmodus, Bundesstaaten-Browser um mit der Erstellung Ihres Workflow-Prototyps fortzufahren. Oder wechseln Sie zu demCodemodus, der einen integrierten Code-Editor bietet, der VS Code ähnelt, um die Amazon States Language (ASL-) Definition Ihrer Zustandsmaschine in der Step Functions Functions-Konsole zu aktualisieren. Weitere Informationen zur Verwendung von Workflow Studio zum Erstellen Ihrer Zustandsmaschinen finden Sie unterVerwenden von Workflow Studio.

> **Important**
>
> Denken Sie daran, den Platzhalter Amazon Resource Name (ARN) für die im Beispielprojekt verwendeten Ressourcen zu aktualisieren, bevor Sie Ihren Workflow ausführen.

- Wenn Sie Eine Demo ausführen ausgewählt haben, erstellt Step Functions ein schreibgeschütztes Beispielprojekt, das eine AWS CloudFormation Vorlage verwendet, um die in dieser Vorlage aufgeführten AWS Ressourcen für Ihr Projekt bereitzustellen. AWS-Konto

  > **Tip**
  >
  > Um die State-Machine-Definition des Beispielprojekts anzuzeigen, wählen Sie Code.

  Wenn Sie bereit sind, wählen Sie Deploy and run aus, um das Beispielprojekt bereitzustellen und die Ressourcen zu erstellen.

  Es kann bis zu 10 Minuten dauern, bis diese Ressourcen und die zugehörigen IAM-Berechtigungen erstellt sind. Während der Bereitstellung Ihrer Ressourcen können Sie den CloudFormation Stack-ID-Link öffnen, um zu sehen, welche Ressourcen bereitgestellt werden.

  Nachdem alle Ressourcen im Beispielprojekt erstellt wurden, wird das neue Beispielprojekt auf der Seite State Machines aufgeführt.

  > **Important**
  >
  > Für jeden in der CloudFormation Vorlage verwendeten Dienst können Standardgebühren anfallen.

Schritt 2: Führen Sie die Zustandsmaschine aus

1. Wählen Sie auf der Seite State Machines Ihr Beispielprojekt aus.
2. Wählen Sie auf der Seite mit dem Beispielprojekt die Option Ausführung starten aus.
3. Gehen Sie im Dialogfeld Ausführung starten wie folgt vor:

1. (Optional) Um Ihre Ausführung zu identifizieren, können Sie im Feld Name einen Namen dafür angeben. Standardmäßig generiert Step Functions automatisch einen eindeutigen Ausführungsnamen.

   > **Note**
   >
   > Mit Step Functions können Sie Namen für Zustandsmaschinen, Ausführungen und Aktivitäten sowie Beschriftungen erstellen, die Nicht-ASCII-Zeichen enthalten. Diese Nicht-ASCII-Namen funktionieren nicht mit Amazon. CloudWatch Um sicherzustellen, dass Sie CloudWatch Messwerte verfolgen können, wählen Sie einen Namen, der nur ASCII-Zeichen verwendet.

2. (Optional) Geben Sie in das Eingabefeld Eingabewerte im JSON-Format ein, um Ihren Workflow auszuführen.

   Wenn Sie sich dafür entschieden haben, eine Demo auszuführen, müssen Sie keine Ausführungseingaben angeben.

   > **Note**
   >
   > Wenn das von Ihnen bereitgestellte Demo-Projekt vorab aufgefüllte Eingabedaten für die Ausführung enthält, verwenden Sie diese Eingabe, um die Zustandsmaschine auszuführen.

3. Wählen Sie Start execution (Ausführung starten) aus.
4. Die Step Functions Functions-Konsole leitet Sie zu einer Seite weiter, die mit Ihrer Ausführungs-ID betitelt ist. Diese Seite wird als Seite mit den Ausführungsdetails bezeichnet. Auf dieser Seite können Sie die Ausführungsergebnisse im Verlauf der Ausführung oder nach deren Abschluss überprüfen.

   Um die Ausführungsergebnisse zu überprüfen, wählen Sie in der Diagrammansicht einzelne Status aus und wählen Sie dann die einzelnen Registerkarten im Schrittdetails Bereich, um die Details der einzelnen Status, einschließlich Eingabe, Ausgabe und Definition, anzuzeigen. Einzelheiten zu den Ausführungsinformationen, die Sie auf der Seite mit den Ausführungsdetails einsehen können, finden Sie unterSeite mit Ausführungsdetails – Schnittstellenübersicht.

Code des Zustandsautomaten aus diesem Beispiel

Die Zustandsmaschine in diesem Beispielprojekt lässt sich in Amazon EMR integrieren, indem Parameter direkt an diese Ressourcen übergeben werden. Sehen Sie sich diese Beispiel-Zustandsmaschine an, um zu sehen, wie Step Functions eine Zustandsmaschine verwendet, um die Amazon EMR-Aufgabe synchron aufzurufen, darauf wartet, dass die Aufgabe erfolgreich ist oder fehlschlägt, und den Cluster beendet.

Weitere Informationen darüber, wie Sie andere AWS Dienste steuern AWS Step Functions können, finden Sie unter. Verwendung AWS Step Functions mit anderen Diensten

```
{
  "Comment": "An example of the Amazon States Language for running jobs on Amazon EMR",
  "StartAt": "Create an EMR cluster",
  "States": {
    "Create an EMR cluster": {
      "Type": "Task",
      "Resource": "arn:<PARTITION>:states:::elasticmapreduce:createCluster.sync",
      "Parameters": {
        "Name": "ExampleCluster",
        "VisibleToAllUsers": true,
        "ReleaseLabel": "emr-5.26.0",
        "Applications": [
          { "Name": "Hive" }
        ],
        "ServiceRole": "<EMR_SERVICE_ROLE>",
        "JobFlowRole": "<EMR_EC2_INSTANCE_PROFILE>",
        "LogUri": "s3://<EMR_LOG_S3_BUCKET>/logs/",
        "Instances": {
          "KeepJobFlowAliveWhenNoSteps": true,
          "InstanceFleets": [
            {
              "Name": "MyMasterFleet",
              "InstanceFleetType": "MASTER",
              "TargetOnDemandCapacity": 1,
              "InstanceTypeConfigs": [
                {
                  "InstanceType": "m5.xlarge"
                }
              ]
            },
            {
```

```
            "Name": "MyCoreFleet",
            "InstanceFleetType": "CORE",
            "TargetOnDemandCapacity": 1,
            "InstanceTypeConfigs": [
              {
                "InstanceType": "m5.xlarge"
              }
            ]
          }
        ]
      }
    },
    "ResultPath": "$.cluster",
    "Next": "Run first step"
  },
  "Run first step": {
    "Type": "Task",
    "Resource": "arn:<PARTITION>:states:::elasticmapreduce:addStep.sync",
    "Parameters": {
      "ClusterId.$": "$.cluster.ClusterId",
      "Step": {
        "Name": "My first EMR step",
        "ActionOnFailure": "CONTINUE",
        "HadoopJarStep": {
          "Jar": "command-runner.jar",
          "Args": ["<COMMAND_ARGUMENTS>"]
        }
      }
    },
    "Retry" : [
      {
        "ErrorEquals": [ "States.ALL" ],
        "IntervalSeconds": 1,
        "MaxAttempts": 3,
        "BackoffRate": 2.0
      }
    ],
    "ResultPath": "$.firstStep",
    "Next": "Run second step"
  },
  "Run second step": {
    "Type": "Task",
    "Resource": "arn:<PARTITION>:states:::elasticmapreduce:addStep.sync",
    "Parameters": {
```

```
        "ClusterId.$": "$.cluster.ClusterId",
        "Step": {
          "Name": "My second EMR step",
          "ActionOnFailure": "CONTINUE",
          "HadoopJarStep": {
            "Jar": "command-runner.jar",
            "Args": ["<COMMAND_ARGUMENTS>"]
          }
        }
      },
      "Retry" : [
        {
          "ErrorEquals": [ "States.ALL" ],
          "IntervalSeconds": 1,
          "MaxAttempts": 3,
          "BackoffRate": 2.0
        }
      ],
      "ResultPath": "$.secondStep",
      "Next": "Terminate Cluster"
    },
    "Terminate Cluster": {
      "Type": "Task",
      "Resource": "arn:<PARTITION>:states:::elasticmapreduce:terminateCluster",
      "Parameters": {
        "ClusterId.$": "$.cluster.ClusterId"
      },
      "End": true
    }
  }
}
```

IAM-Beispiel

Diese vom Beispielprojekt generierte Beispielrichtlinie AWS Identity and Access Management (IAM) beinhaltet die geringsten Rechte, die für die Ausführung der Zustandsmaschine und der zugehörigen Ressourcen erforderlich sind. Es hat sich bewährt, nur die Berechtigungen in Ihre IAM-Richtlinien aufzunehmen, die erforderlich sind.

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

Die folgende Richtlinie stellt sicher, dass `addStep` über ausreichende Berechtigungen verfügt.

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

Informationen zur Konfiguration von IAM bei der Verwendung von Step Functions mit anderen AWS Diensten finden Sie unterIAM-Richtlinien für integrierte Dienste.

Einen EMR Serverless Job ausführen

Dieses Beispielprojekt zeigt, wie Sie eine EMR Serverless Anwendung erstellen und starten können. Dieses Projekt zeigt auch, wie Sie mehrere Jobs innerhalb dieser Anwendung ausführen können.

In diesem Beispielprojekt werden die Zustandsmaschine und die unterstützenden AWS Ressourcen erstellt und die zugehörigen IAM-Berechtigungen konfiguriert. Erkunden Sie dieses Beispielprojekt, um zu erfahren, wie EMR Serverless Jobs mithilfe von Step Functions Zustandsautomaten ausgeführt werden, oder verwenden Sie es als Ausgangspunkt für Ihre eigenen Projekte.

> **Important**
>
> EMR Serverless hat kein kostenloses Preiskontingent. Bei der Ausführung des Beispielprojekts fallen Kosten an. Preisinformationen finden Sie auf der Seite Amazon EMR Serverless-Preisliste.
> Darüber hinaus hängt die Verfügbarkeit der EMR Serverless Serviceintegration von der Verfügbarkeit von EMR Serverless APIs ab. Aus diesem Grund funktioniert dieses

Beispielprojekt möglicherweise nicht richtig oder ist in einigen Fällen nicht verfügbar AWS-Regionen. Informationen zur Verfügbarkeit von EMR Serverless in finden Sie im Thema Weitere Überlegungen AWS-Regionen.

AWS CloudFormation-Vorlage und zusätzliche Ressourcen

Sie verwenden eine CloudFormation Vorlage, um dieses Beispielprojekt bereitzustellen. Diese Vorlage erstellt die folgenden Ressourcen in Ihrem AWS-Konto:

- Eine Step Functions Zustandsmaschine.
- Ausführungsrolle für die Zustandsmaschine. Diese Rolle gewährt die Berechtigungen, die Ihre Zustandsmaschine für den Zugriff auf andere AWS-Services Ressourcen wie die EMR Serverless CreateApplicationAktion benötigt.
- Jobausführungsrolle fürEMR Serverless. Diese Rolle gewährt die Berechtigungen, die ein EMR Serverless Jobrun annehmen kann, wenn er in Ihrem Namen andere Dienste aufruft.

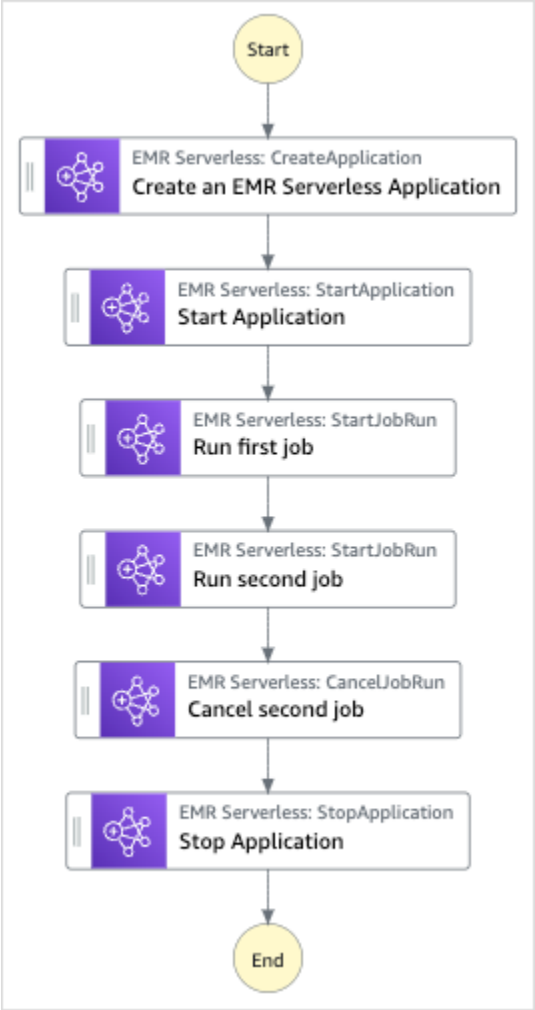
Schritt 1: Erstellen Sie den Zustandsmaschine und stellen Sie Ressourcen bereit

1. Öffnen Sie die Step Functions Functions-Konsole und wählen Sie Create State Machine.
2. Geben Sie **`EMR Serverless`** etwas in das Suchfeld ein und wählen Sie dann aus den zurückgegebenen Suchergebnissen die Option EMR Serverless Job ausführen aus.
3. Wählen Sie Next (Weiter), um fortzufahren.
4. Step Functions listet die Funktionen auf, die in dem von Ihnen ausgewählten Beispielprojekt AWS-Services verwendet wurden. Außerdem wird ein Workflow-Diagramm für das Beispielprojekt angezeigt. Stellen Sie dieses Projekt für Ihr Projekt bereit AWS-Konto oder verwenden Sie es als Ausgangspunkt für die Erstellung Ihrer eigenen Projekte. Je nachdem, wie Sie vorgehen möchten, wählen Sie „Demo ausführen“ oder „Darauf aufbauen“.

   In diesem Beispielprojekt werden die folgenden Ressourcen bereitgestellt:

   - Ein Step Functions-Zustandsautomat
   - Zugehörige AWS Identity and Access Management (IAM)-Rollen

Die folgende Abbildung zeigt das Workflow-Diagramm für das Beispielprojekt EMR ServerlessJob ausführen:



5. Wählen Sie Vorlage verwenden, um mit Ihrer Auswahl fortzufahren.
6. Führen Sie eine der folgenden Aktionen aus:

   - Wenn Sie „Darauf aufbauen“ ausgewählt haben, erstellt Step Functions den Workflow-Prototyp für das von Ihnen ausgewählte Beispielprojekt. Step Functions stellt die in der Workflow-Definition aufgeführten Ressourcen nicht bereit.

     Ziehen Sie in Workflow Studio Status per Drag-and-Drop aus demEntwurfsmodus, Bundesstaaten-Browser um mit der Erstellung Ihres Workflow-Prototyps fortzufahren. Oder wechseln Sie zu demCodemodus, der einen integrierten Code-Editor bietet, der VS Code ähnelt, um die Amazon States Language (ASL-) Definition Ihrer Zustandsmaschine in der Step Functions Functions-Konsole zu aktualisieren. Weitere Informationen zur Verwendung

von Workflow Studio zum Erstellen Ihrer Zustandsmaschinen finden Sie unterVerwenden von Workflow Studio.

> **Important**
>
> Denken Sie daran, den Platzhalter Amazon Resource Name (ARN) für die im Beispielprojekt verwendeten Ressourcen zu aktualisieren, bevor Sie Ihren Workflow ausführen.

- Wenn Sie Eine Demo ausführen ausgewählt haben, erstellt Step Functions ein schreibgeschütztes Beispielprojekt, das eine AWS CloudFormation Vorlage verwendet, um die in dieser Vorlage aufgeführten AWS Ressourcen für Ihr Projekt bereitzustellen. AWS-Konto

  > **Tip**
  >
  > Um die State-Machine-Definition des Beispielprojekts anzuzeigen, wählen Sie Code.

  Wenn Sie bereit sind, wählen Sie Deploy and run aus, um das Beispielprojekt bereitzustellen und die Ressourcen zu erstellen.

  Es kann bis zu 10 Minuten dauern, bis diese Ressourcen und die zugehörigen IAM-Berechtigungen erstellt sind. Während der Bereitstellung Ihrer Ressourcen können Sie den CloudFormation Stack-ID-Link öffnen, um zu sehen, welche Ressourcen bereitgestellt werden.

  Nachdem alle Ressourcen im Beispielprojekt erstellt wurden, wird das neue Beispielprojekt auf der Seite State Machines aufgeführt.

  > **Important**
  >
  > Für jeden in der CloudFormation Vorlage verwendeten Dienst können Standardgebühren anfallen.

Schritt 2: Führen Sie die Zustandsmaschine aus

1. Wählen Sie auf der Seite State Machines Ihr Beispielprojekt aus.
2. Wählen Sie auf der Seite mit dem Beispielprojekt die Option Ausführung starten aus.

3. Gehen Sie im Dialogfeld Ausführung starten wie folgt vor:

   1. (Optional) Um Ihre Ausführung zu identifizieren, können Sie im Feld Name einen Namen dafür angeben. Standardmäßig generiert Step Functions automatisch einen eindeutigen Ausführungsnamen.

      > **Note**
      >
      > Mit Step Functions können Sie Namen für Zustandsmaschinen, Ausführungen, Aktivitäten und Labels erstellen, die Nicht-ASCII-Zeichen enthalten. Diese Nicht-ASCII-Namen funktionieren nicht mit Amazon. CloudWatch Um sicherzustellen, dass Sie CloudWatch Messwerte verfolgen können, wählen Sie einen Namen, der nur ASCII-Zeichen verwendet.

   2. (Optional) Geben Sie in das Eingabefeld Eingabewerte im JSON-Format ein, um Ihren Workflow auszuführen.

      Wenn Sie sich dafür entschieden haben, eine Demo auszuführen, müssen Sie keine Ausführungseingaben angeben.

   3. Wählen Sie Start execution (Ausführung starten) aus.
   4. Die Step Functions Functions-Konsole leitet Sie zu einer Seite weiter, die mit Ihrer Ausführungs-ID betitelt ist. Diese Seite wird als Seite mit den Ausführungsdetails bezeichnet. Auf dieser Seite können Sie die Ausführungsergebnisse im Verlauf der Ausführung oder nach deren Abschluss überprüfen.

      Um die Ausführungsergebnisse zu überprüfen, wählen Sie in der Diagrammansicht einzelne Status aus und wählen Sie dann die einzelnen Registerkarten im Schrittdetails Bereich, um die Details der einzelnen Status, einschließlich Eingabe, Ausgabe und Definition, anzuzeigen. Einzelheiten zu den Ausführungsinformationen, die Sie auf der Seite mit den Ausführungsdetails einsehen können, finden Sie unterSeite mit Ausführungsdetails – Schnittstellenübersicht.

Einen Workflow innerhalb eines Workflows starten (Step Functions, Lambda)

Dieses Beispielprojekt zeigt, wie eine AWS Step Functions Zustandsmaschine verwendet wird, um andere State-Machine-Ausführungen zu starten. Hinweise zum Starten von State-Machine-

Ausführungen von einem anderen Zustandsmaschine aus finden Sie unter. Starten von Workflow-Ausführungen über einen Aufgabenstatus

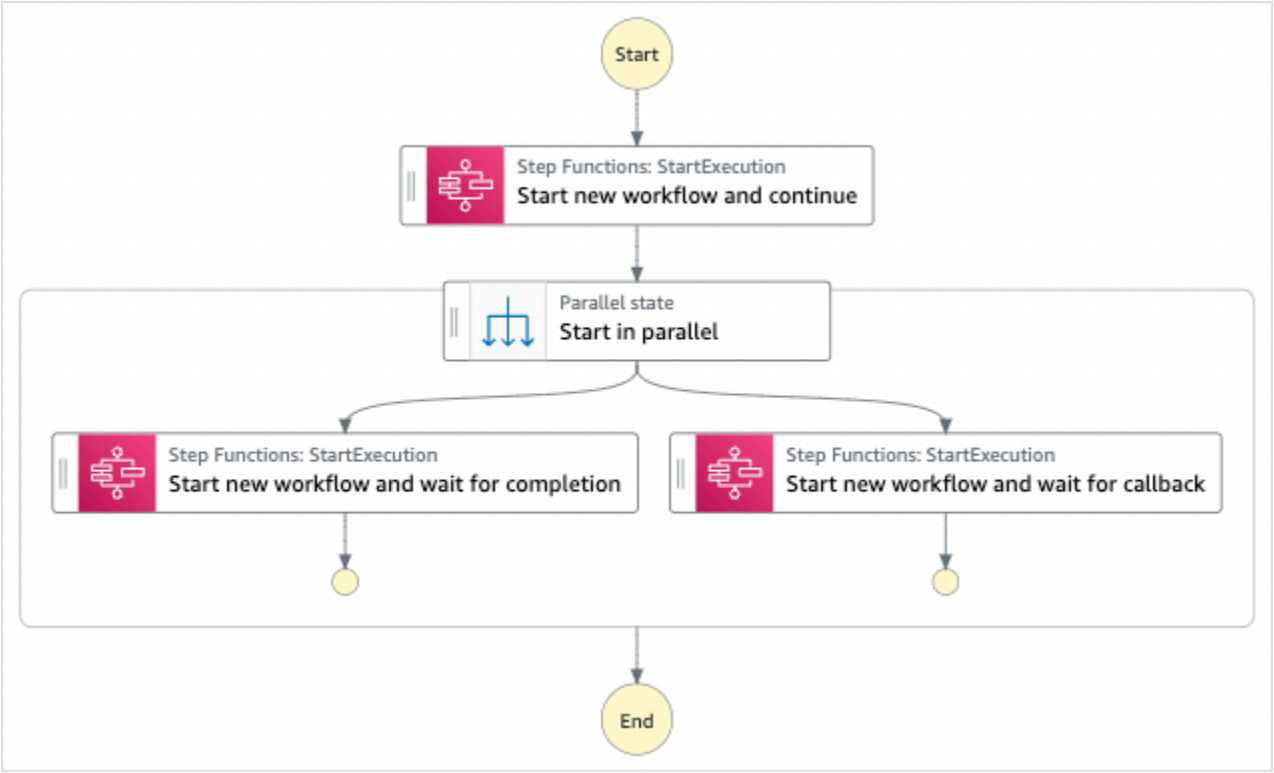
Schritt 1: Erstellen Sie den Zustandsmaschine und stellen Sie Ressourcen bereit

1. Öffnen Sie die Step Functions Functions-Konsole und wählen Sie Create State Machine.
2. Geben Sie **`Start a workflow within a workflow`** etwas in das Suchfeld ein, und wählen Sie dann aus den zurückgegebenen Suchergebnissen die Option Einen Workflow innerhalb eines Workflows starten aus.
3. Wählen Sie Next (Weiter), um fortzufahren.
4. Step Functions listet die Funktionen auf, die in dem von Ihnen ausgewählten Beispielprojekt AWS-Services verwendet wurden. Außerdem wird ein Workflow-Diagramm für das Beispielprojekt angezeigt. Stellen Sie dieses Projekt für Ihr Projekt bereit AWS-Konto oder verwenden Sie es als Ausgangspunkt für die Erstellung Ihrer eigenen Projekte. Je nachdem, wie Sie vorgehen möchten, wählen Sie „Demo ausführen“ oder „Darauf aufbauen“.

   In diesem Beispielprojekt werden die folgenden Ressourcen bereitgestellt:

   - Eine zusätzliche Zustandsmaschine. Die Ausführung dieser Zustandsmaschine wird von der Zustandsmaschine gestartet, die Sie ausführen.
   - Eine Callback-Lambda-Funktion. Diese Funktion wird in der zusätzlichen Zustandsmaschine verwendet, um den Callback-Mechanismus zu implementieren.
   - Eine AWS Step Functions Zustandsmaschine
   - Verwandte AWS Identity and Access Management Rollen (IAM)

   Die folgende Abbildung zeigt das Workflow-Diagramm für das Beispielprojekt „Einen Workflow innerhalb eines Workflows starten“:



5. Wählen Sie Vorlage verwenden, um mit Ihrer Auswahl fortzufahren.
6. Führen Sie eine der folgenden Aktionen aus:

   - Wenn Sie „Darauf aufbauen“ ausgewählt haben, erstellt Step Functions den Workflow-Prototyp für das von Ihnen ausgewählte Beispielprojekt. Step Functions stellt die in der Workflow-Definition aufgeführten Ressourcen nicht bereit.

     Ziehen Sie in Workflow Studio Status per Drag-and-Drop aus demEntwurfsmodus, Bundesstaaten-Browser um mit der Erstellung Ihres Workflow-Prototyps fortzufahren. Oder wechseln Sie zu demCodemodus, der einen integrierten Code-Editor bietet, der VS Code ähnelt, um die Amazon States Language (ASL-) Definition Ihrer Zustandsmaschine in der Step Functions Functions-Konsole zu aktualisieren. Weitere Informationen zur Verwendung von Workflow Studio zum Erstellen Ihrer Zustandsmaschinen finden Sie unterVerwenden von Workflow Studio.

     > **Important**
     >
     > Denken Sie daran, den Platzhalter Amazon Resource Name (ARN) für die im Beispielprojekt verwendeten Ressourcen zu aktualisieren, bevor Sie Ihren Workflow ausführen.

- Wenn Sie Eine Demo ausführen ausgewählt haben, erstellt Step Functions ein schreibgeschütztes Beispielprojekt, das eine AWS CloudFormation Vorlage verwendet, um die in dieser Vorlage aufgeführten AWS Ressourcen für Ihr Projekt bereitzustellen. AWS-Konto

  > **Tip**
  >
  > Um die State-Machine-Definition des Beispielprojekts anzuzeigen, wählen Sie Code.

  Wenn Sie bereit sind, wählen Sie Deploy and run aus, um das Beispielprojekt bereitzustellen und die Ressourcen zu erstellen.

  Es kann bis zu 10 Minuten dauern, bis diese Ressourcen und die zugehörigen IAM-Berechtigungen erstellt sind. Während der Bereitstellung Ihrer Ressourcen können Sie den CloudFormation Stack-ID-Link öffnen, um zu sehen, welche Ressourcen bereitgestellt werden.

  Nachdem alle Ressourcen im Beispielprojekt erstellt wurden, wird das neue Beispielprojekt auf der Seite State Machines aufgeführt.

  > **Important**
  >
  > Für jeden in der CloudFormation Vorlage verwendeten Dienst können Standardgebühren anfallen.

Schritt 2: Führen Sie die Zustandsmaschine aus

1. Wählen Sie auf der Seite State Machines Ihr Beispielprojekt aus.
2. Wählen Sie auf der Seite mit dem Beispielprojekt die Option Ausführung starten aus.
3. Gehen Sie im Dialogfeld Ausführung starten wie folgt vor:

   1. (Optional) Um Ihre Ausführung zu identifizieren, können Sie im Feld Name einen Namen dafür angeben. Standardmäßig generiert Step Functions automatisch einen eindeutigen Ausführungsnamen.

> **Note**
>
> Mit Step Functions können Sie Namen für Zustandsmaschinen, Ausführungen und Aktivitäten sowie Beschriftungen erstellen, die Nicht-ASCII-Zeichen enthalten. Diese Nicht-ASCII-Namen funktionieren nicht mit Amazon. CloudWatch Um sicherzustellen, dass Sie CloudWatch Messwerte verfolgen können, wählen Sie einen Namen, der nur ASCII-Zeichen verwendet.

2. (Optional) Geben Sie in das Eingabefeld Eingabewerte im JSON-Format ein, um Ihren Workflow auszuführen.

   Wenn Sie sich dafür entschieden haben, eine Demo auszuführen, müssen Sie keine Ausführungseingaben angeben.

   > **Note**
   >
   > Wenn das von Ihnen bereitgestellte Demo-Projekt vorab aufgefüllte Eingabedaten für die Ausführung enthält, verwenden Sie diese Eingabe, um die Zustandsmaschine auszuführen.

3. Wählen Sie Start execution (Ausführung starten) aus.
4. Die Step Functions Functions-Konsole leitet Sie zu einer Seite weiter, die mit Ihrer Ausführungs-ID betitelt ist. Diese Seite wird als Seite mit den Ausführungsdetails bezeichnet. Auf dieser Seite können Sie die Ausführungsergebnisse im Verlauf der Ausführung oder nach deren Abschluss überprüfen.

   Um die Ausführungsergebnisse zu überprüfen, wählen Sie in der Diagrammansicht einzelne Status aus und wählen Sie dann die einzelnen Registerkarten im Schrittdetails Bereich, um die Details der einzelnen Status, einschließlich Eingabe, Ausgabe und Definition, anzuzeigen. Einzelheiten zu den Ausführungsinformationen, die Sie auf der Seite mit den Ausführungsdetails einsehen können, finden Sie unterSeite mit Ausführungsdetails – Schnittstellenübersicht.

Code des Zustandsautomaten aus diesem Beispiel

Die Zustandsmaschine in diesem Beispielprojekt integriert eine weitere Zustandsmaschine und AWS Lambda übergibt Parameter direkt an diese Ressourcen.

Sehen Sie sich diese Beispiel-Zustandsmaschine an, um zu sehen, wie Step Functions die `StartExecution` API-Aktion für die andere Zustandsmaschine aufruft. Es startet zwei Instances des anderen Zustandsautomaten parallel: eine mit dem Muster Ausführen einer Aufgabe (.sync) und eine mit dem Muster Warten auf einen Callback mit dem Aufgabentoken.

Weitere Informationen darüber, wie AWS Step Functions Sie andere AWS Dienste steuern können, finden Sie unterVerwendung AWS Step Functions mit anderen Diensten.

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

Informationen zur Konfiguration von IAM bei der Verwendung von Step Functions mit anderen AWS Diensten finden Sie unterIAM-Richtlinien für integrierte Dienste.

Dynamisches Verarbeiten von Daten mit einem Map-Status

Dieses Beispielprojekt demonstriert die dynamische Parallelität mithilfe eines Zuordnung-Zustands.

In diesem Projekt verwendet Step Functions eine AWS Lambda Funktion, um Nachrichten aus einer Amazon SQS SQS-Warteschlange abzurufen und ein JSON-Array dieser Nachrichten an einen `Map` Status weiterzuleiten. Für jede Nachricht in der Warteschlange schreibt die Zustandsmaschine die Nachricht in DynamoDB, ruft die andere Lambda-Funktion auf, um die Nachricht aus Amazon SQS zu entfernen, und veröffentlicht die Nachricht dann im Amazon SNS SNS-Thema.

Weitere Informationen zu `Map` Status- und Step Functions Functions-Dienstintegrationen finden Sie im Folgenden:

- Zuordnung
- Verwendung AWS Step Functions mit anderen Diensten

Schritt 1: Erstellen Sie die Zustandsmaschine und stellen Sie Ressourcen bereit

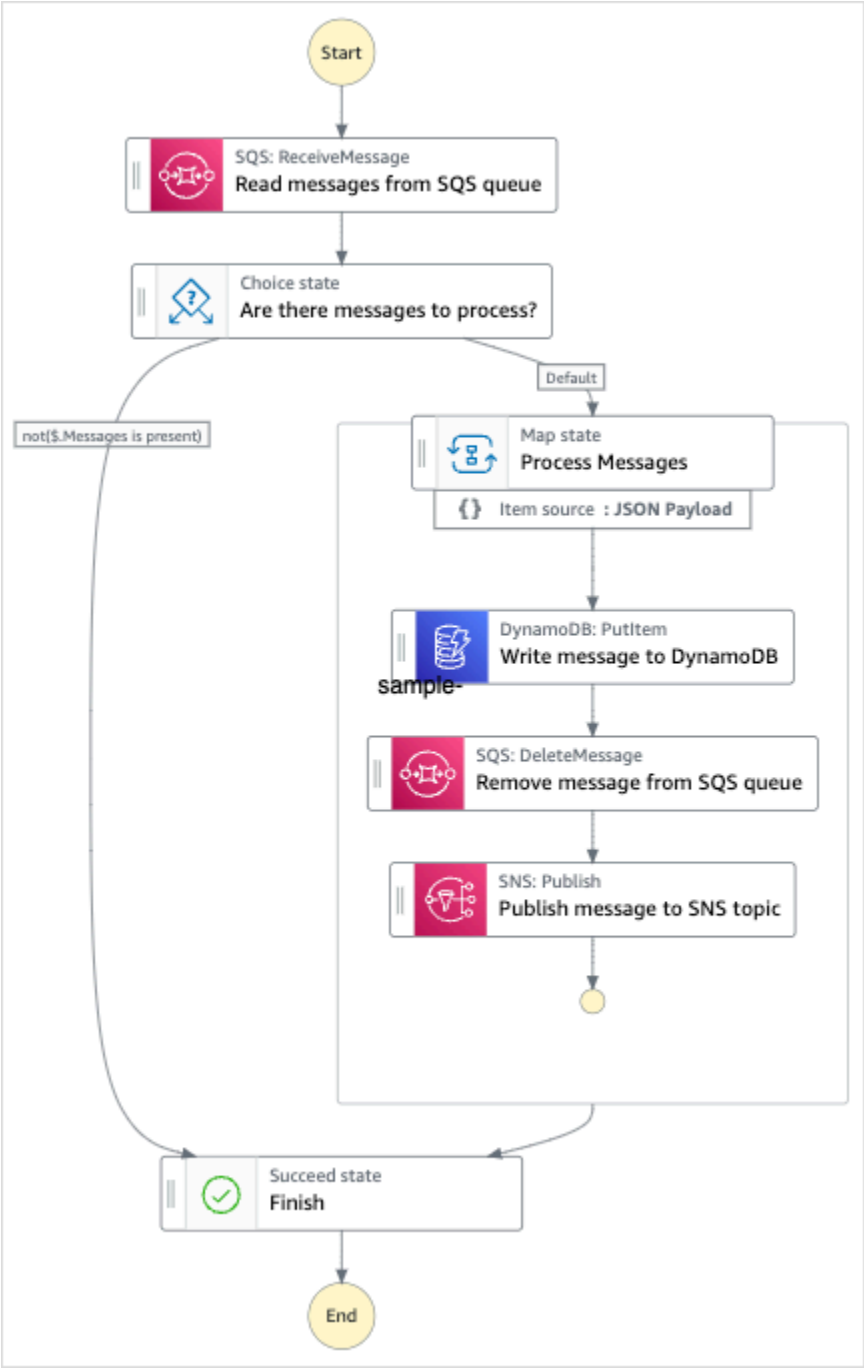
1. Öffnen Sie die Step Functions Functions-Konsole und wählen Sie Create State Machine.
2. Geben Sie **`Dynamically process data with a Map state`** etwas in das Suchfeld ein und wählen Sie dann aus den zurückgegebenen Suchergebnissen die Option Daten dynamisch verarbeiten mit einem Zuordnungsstatus aus.
3. Wählen Sie Next (Weiter), um fortzufahren.
4. Step Functions listet die Funktionen auf, die in dem von Ihnen ausgewählten Beispielprojekt AWS-Services verwendet wurden. Außerdem wird ein Workflow-Diagramm für das Beispielprojekt angezeigt. Stellen Sie dieses Projekt für Ihr Projekt bereit AWS-Konto oder verwenden Sie es als Ausgangspunkt für die Erstellung Ihrer eigenen Projekte. Je nachdem, wie Sie vorgehen möchten, wählen Sie „Demo ausführen“ oder „Darauf aufbauen“.

   In diesem Beispielprojekt werden die folgenden Ressourcen bereitgestellt:

   - Eine Amazon SQS SQS-Warteschlange, aus der der Map-Status Nachrichten iterativ liest und entfernt.
   - Eine DynamoDB-Tabelle, in die der Map-Status iterativ Nachrichten schreibt.

- Ein Amazon SNS SNS-Thema, zu dem Step Functions die Nachrichten veröffentlicht, die es aus der Amazon SQS SQS-Warteschlange liest.
- Zwei Funktionen AWS Lambda
- Eine AWS Step Functions Staatsmaschine
- Verwandte AWS Identity and Access Management Rollen (IAM)

Die folgende Abbildung zeigt das Workflow-Diagramm für das Beispielprojekt Dynamisch verarbeiten mit einem Map-Status:



5. Wählen Sie Vorlage verwenden, um mit Ihrer Auswahl fortzufahren.
6. Führen Sie eine der folgenden Aktionen aus:

   - Wenn Sie „Darauf aufbauen“ ausgewählt haben, erstellt Step Functions den Workflow-Prototyp für das von Ihnen ausgewählte Beispielprojekt. Step Functions stellt die in der Workflow-Definition aufgeführten Ressourcen nicht bereit.

Ziehen Sie in Workflow Studio Status per Drag-and-Drop aus demEntwurfsmodus, Bundesstaaten-Browser um mit der Erstellung Ihres Workflow-Prototyps fortzufahren. Oder wechseln Sie zu demCodemodus, der einen integrierten Code-Editor bietet, der VS Code ähnelt, um die Amazon States Language (ASL-) Definition Ihrer Zustandsmaschine in der Step Functions Functions-Konsole zu aktualisieren. Weitere Informationen zur Verwendung von Workflow Studio zum Erstellen Ihrer Zustandsmaschinen finden Sie unterVerwenden von Workflow Studio.

> **Important**
>
> Denken Sie daran, den Platzhalter Amazon Resource Name (ARN) für die im Beispielprojekt verwendeten Ressourcen zu aktualisieren, bevor Sie Ihren Workflow ausführen.

- Wenn Sie Eine Demo ausführen ausgewählt haben, erstellt Step Functions ein schreibgeschütztes Beispielprojekt, das eine AWS CloudFormation Vorlage verwendet, um die in dieser Vorlage aufgelisteten AWS Ressourcen für Ihr Projekt bereitzustellen. AWS-Konto

> **Tip**
>
> Um die State-Machine-Definition des Beispielprojekts anzuzeigen, wählen Sie Code.

Wenn Sie bereit sind, wählen Sie Deploy and run aus, um das Beispielprojekt bereitzustellen und die Ressourcen zu erstellen.

Es kann bis zu 10 Minuten dauern, bis diese Ressourcen und die zugehörigen IAM-Berechtigungen erstellt sind. Während der Bereitstellung Ihrer Ressourcen können Sie den CloudFormation Stack-ID-Link öffnen, um zu sehen, welche Ressourcen bereitgestellt werden.

Nachdem alle Ressourcen im Beispielprojekt erstellt wurden, wird das neue Beispielprojekt auf der Seite State Machines aufgeführt.

> **Important**
>
> Für jeden in der CloudFormation Vorlage verwendeten Dienst können Standardgebühren anfallen.

Nachdem die Ressourcen des Beispielprojekts bereitgestellt wurden, müssen Sie Elemente zur Amazon SQS-Warteschlange hinzufügen und das Amazon SNS-Thema abonnieren, bevor Sie die Zustandsmaschine ausführen.

Schritt 2: Abonnieren Sie das Amazon SNS SNS-Thema

1. Öffnen Sie die Amazon-SNS-Konsole.
2. Wählen Sie Topics (Themen) und dann das Thema aus, das vom `Map`-Zustands-Beispielprojekt erstellt wurde.

   Der Name wird ähnlich sein wie MapSampleProj-snsTopic-1cqo4hq3ir1kn.
3. Wählen Sie Create subscription (Abonnement erstellen) aus.

   Die Seite Create subscription (Abonnement erstellen) wird angezeigt und enthält den Topic ARN (Thema-ARN) für das Thema.
4. Wählen Sie unter Protocol (Protokoll) die Option Email (E-Mail) aus.
5. Geben Sie unter Endpoint (Endpunkt) eine E-Mail-Adresse ein, um das Thema zu abonnieren.
6. Wählen Sie Create subscription (Abonnement erstellen) aus.

   > **Note**
   >
   > Sie müssen das Abonnement in Ihrer E-Mail bestätigen, bevor es aktiv ist.
7. Öffnen Sie die E-Mail Subscription Confirmation (Abonnementbestätigung) im zugehörigen Konto und dann die URL Confirm subscription (Abonnement bestätigen).

   Die Seite Subscription confirmed! (Abonnement bestätigen!) wird angezeigt.

Schritt 3: Nachrichten zur Amazon SQS SQS-Warteschlange hinzufügen

1. Öffnen Sie die Amazon-SQS-Konsole.
2. Wählen Sie die Warteschlange aus, die vom `Map`-Zustandsbeispiel-Projekt erstellt wurde.

   Der Name wird ähnlich sein wie MapSampleProj-sqsqueue-1udic9vzdorN7.
3. Wählen Sie Nachrichten senden und empfangen.
4. Geben Sie auf der Seite Nachrichten senden und empfangen eine Nachricht ein und wählen Sie Nachricht senden aus.

5. Geben Sie eine weitere Nachricht ein und wählen Sie Nachricht senden. Geben Sie weitere Nachrichten ein, bis Sie mehrere in der Amazon SQS SQS-Warteschlange haben.

Schritt 4: Führen Sie die Zustandsmaschine aus

> **Note**
>
> Die Warteschlangen in Amazon SNS sind irgendwann konsistent. Um die besten Ergebnisse zu erzielen, warten Sie einige Minuten zwischen dem Füllen Ihrer Warteschlange und dem Ausführen einer Ausführung Ihres Zustandsautomaten.

1. Wählen Sie auf der Seite State Machines Ihr Beispielprojekt aus.
2. Wählen Sie auf der Seite mit dem Beispielprojekt die Option Ausführung starten aus.
3. Gehen Sie im Dialogfeld Ausführung starten wie folgt vor:

   1. (Optional) Um Ihre Ausführung zu identifizieren, können Sie im Feld Name einen Namen dafür angeben. Standardmäßig generiert Step Functions automatisch einen eindeutigen Ausführungsnamen.

      > **Note**
      >
      > Mit Step Functions können Sie Namen für Zustandsmaschinen, Ausführungen und Aktivitäten sowie Beschriftungen erstellen, die Nicht-ASCII-Zeichen enthalten. Diese Nicht-ASCII-Namen funktionieren nicht mit Amazon. CloudWatch Um sicherzustellen, dass Sie CloudWatch Messwerte verfolgen können, wählen Sie einen Namen, der nur ASCII-Zeichen verwendet.

   2. (Optional) Geben Sie in das Eingabefeld Eingabewerte im JSON-Format ein, um Ihren Workflow auszuführen.

      Wenn Sie sich dafür entschieden haben, eine Demo auszuführen, müssen Sie keine Ausführungseingaben angeben.

> **Note**
>
> Wenn das von Ihnen bereitgestellte Demo-Projekt vorab aufgefüllte Eingabedaten für die Ausführung enthält, verwenden Sie diese Eingabe, um die Zustandsmaschine auszuführen.

3. Wählen Sie Start execution (Ausführung starten) aus.
4. Die Step Functions Functions-Konsole leitet Sie zu einer Seite weiter, die mit Ihrer Ausführungs-ID betitelt ist. Diese Seite wird als Seite mit den Ausführungsdetails bezeichnet. Auf dieser Seite können Sie die Ausführungsergebnisse im Verlauf der Ausführung oder nach deren Abschluss überprüfen.

   Um die Ausführungsergebnisse zu überprüfen, wählen Sie in der Diagrammansicht einzelne Status aus und wählen Sie dann die einzelnen Registerkarten im Schrittdetails Bereich, um die Details der einzelnen Status, einschließlich Eingabe, Ausgabe und Definition, anzuzeigen. Einzelheiten zu den Ausführungsinformationen, die Sie auf der Seite mit den Ausführungsdetails einsehen können, finden Sie unterSeite mit Ausführungsdetails – Schnittstellenübersicht.

Beispiel für einen State-Machine-Code

Die Zustandsmaschine in diesem Beispielprojekt lässt sich in Amazon SQS, Amazon SNS und Lambda integrieren, indem Parameter direkt an diese Ressourcen übergeben werden.

Sehen Sie sich diese Beispiel-Zustandsmaschine an, um zu sehen, wie Step Functions Lambda, DynamoDB und Amazon SNS steuert, indem es eine Verbindung zum Amazon-Ressourcennamen (ARN) im `Resource` Feld herstellt und `Parameters` an die Service-API weiterleitet.

Weitere Informationen darüber, wie Sie andere AWS Dienste steuern AWS Step Functions können, finden Sie unter. Verwendung AWS Step Functions mit anderen Diensten

```
{
  "Comment": "An example of the Amazon States Language for reading messages from an SQS
 queue and iteratively processing each message.",
  "StartAt": "Read messages from SQS Queue",
  "States": {
    "Read messages from SQS Queue": {
      "Type": "Task",
```

```
      "Resource": "arn:aws:states:::lambda:invoke",
      "OutputPath": "$.Payload",
      "Parameters": {
        "FunctionName": "MapSampleProj-ReadFromSQSQueueLambda-1MY3M63RMJVA9"
      },
      "Next": "Are there messages to process?"
    },
    "Are there messages to process?": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$",
          "StringEquals": "No messages",
          "Next": "Finish"
        }
      ],
      "Default": "Process messages"
    },
    "Process messages": {
      "Type": "Map",
      "Next": "Finish",
      "ItemsPath": "$",
      "Parameters": {
        "MessageNumber.$": "$$.Map.Item.Index",
        "MessageDetails.$": "$$.Map.Item.Value"
      },
      "Iterator": {
        "StartAt": "Write message to DynamoDB",
        "States": {
          "Write message to DynamoDB": {
            "Type": "Task",
            "Resource": "arn:aws:states:::dynamodb:putItem",
            "ResultPath": null,
            "Parameters": {
              "TableName": "MapSampleProj-DDBTable-YJDJ1MKIN6C5",
              "ReturnConsumedCapacity": "TOTAL",
              "Item": {
                "MessageId": {
                  "S.$": "$.MessageDetails.MessageId"
                },
                "Body": {
                  "S.$": "$.MessageDetails.Body"
                }
              }
```

```
            },
            "Next": "Remove message from SQS queue"
          },
          "Remove message from SQS queue": {
            "Type": "Task",
            "Resource": "arn:aws:states:::lambda:invoke",
            "InputPath": "$.MessageDetails",
            "ResultPath": null,
            "Parameters": {
              "FunctionName": "MapSampleProj-DeleteFromSQSQueueLambda-198J2839ZO5K2",
              "Payload": {
                "ReceiptHandle.$": "$.ReceiptHandle"
              }
            },
            "Next": "Publish message to SNS topic"
          },
          "Publish message to SNS topic": {
            "Type": "Task",
            "Resource": "arn:aws:states:::sns:publish",
            "InputPath": "$.MessageDetails",
            "Parameters": {
              "Subject": "Message from Step Functions!",
              "Message.$": "$.Body",
              "TopicArn": "arn:aws:sns:us-east-1:012345678910:MapSampleProj-
SNSTopic-1CQO4HQ3IR1KN"
            },
            "End": true
          }
        }
      }
    },
    "Finish": {
      "Type": "Succeed"
    }
  }
}
```

IAM-Beispiel

Diese vom Beispielprojekt generierte Beispielrichtlinie AWS Identity and Access Management (IAM) beinhaltet die geringsten Rechte, die für die Ausführung der Zustandsmaschine und der zugehörigen Ressourcen erforderlich sind. Wir empfehlen, dass Sie nur die Berechtigungen in Ihre IAM-Richtlinien aufnehmen, die erforderlich sind.

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

Informationen zur Konfiguration von IAM bei der Verwendung von Step Functions mit anderen AWS Diensten finden Sie unterIAM-Richtlinien für integrierte Dienste.

Verarbeiten einer CSV-Datei mit Distributed Map

Dieses Beispielprojekt zeigt, wie Sie den Status Verteilte Zuordnung verwenden können, um mehr als 10 000 Zeilen einer CSV-Datei zu iterieren, die mit einer -LambdaFunktion generiert wird. Die CSV-Datei enthält Versandinformationen zu Kundenaufträgen und wird in einem Amazon S3-Bucket gespeichert. Die verteilte Zuordnung iteriert über einen Stapel von 10 Zeilen in der CSV-Datei zur Datenanalyse.

Die verteilte Karte enthält eine Lambda Funktion, um verzögerte Bestellungen zu erkennen. Die verteilte Zuordnung enthält auch eine Inline-Zuweisung, um die verzögerten Bestellungen in einem Batch zu verarbeiten, und gibt diese verzögerten Bestellungen in einem Array zurück. Für jede verzögerte Reihenfolge sendet die Inline-Zuweisung eine Nachricht an eine Amazon SQS-Warteschlange. Schließlich speichert dieses Beispielprojekt die Ergebnisse von Map Run in einem anderen Amazon S3-Bucket in Ihrem AWS-Konto.

Mit Distributed Map können Sie bis zu 10.000 parallele untergeordnete Workflow-Ausführungen gleichzeitig ausführen. In diesem Beispielprojekt ist die maximale Parallelität von Distributed Map auf 1 000 festgelegt, was sie auf 1 000 parallele untergeordnete Workflow-Ausführungen begrenzt.

Dieses Beispielprojekt erstellt den Zustandsautomaten und die unterstützenden AWS Ressourcen und konfiguriert die zugehörigen IAM-Berechtigungen. Erkunden Sie dieses Beispielprojekt, um mehr über die Verwendung der verteilten Karte zur Orchestrierung großer, paralleler Workloads zu erfahren, oder verwenden Sie sie als Ausgangspunkt für Ihre eigenen Projekte.

AWS CloudFormation -Vorlage und zusätzliche Ressourcen

Sie verwenden eine CloudFormation Vorlage, um dieses Beispielprojekt bereitzustellen. Diese Vorlage erstellt die folgenden Ressourcen in Ihrem AWS-Konto:

- Ein Step-Functions-Zustandsautomat.
- Ausführungsrolle für den Zustandsautomaten. Diese Rolle gewährt die Berechtigungen, die Ihr Zustandsautomat für den Zugriff auf andere AWS-Services und Ressourcen benötigt, wie z. B. die Invoke-Aktion der Lambda-Funktion.
- Eine Lambda-Funktion namens `CSVGeneratorFunction`, die eine CSV-Datei generiert, die die Details der Kundenbestellung enthält.
- Ausführungsrolle für die Lambda-Funktion des CSV-Generators. Diese Rolle gewährt der Funktion die Berechtigung, auf andere zuzugreifen AWS-Services.

- Ein Amazon S3-Eingabe-Bucket zum Speichern der generierten CSV-Datei.
- Eine Lambda-Funktion zur verzögerten Bestellerkennung, die die CSV-Dateidaten analysiert und verzögerte Bestellungen erkennt.
- Ausführungsrolle für die Lambda-Funktion mit verzögerter Reihenfolge. Diese Rolle gewährt der Funktion die Berechtigung, auf andere zuzugreifen AWS-Services.
- Ein Amazon S3-Ausgabe-Bucket zum Speichern der Analyseergebnisse der Kundenaufträge.
- Eine Amazon SQS-Warteschlange, an die Step Functions Nachrichten für jede verzögerte Reihenfolge sendet. Diese Nachrichten enthalten die IDs der Kunden und ihrer Bestellungen.
- Eine CloudWatch Protokollgruppe, die Informationen über den Ausführungsverlauf des Zustandsautomaten speichert.

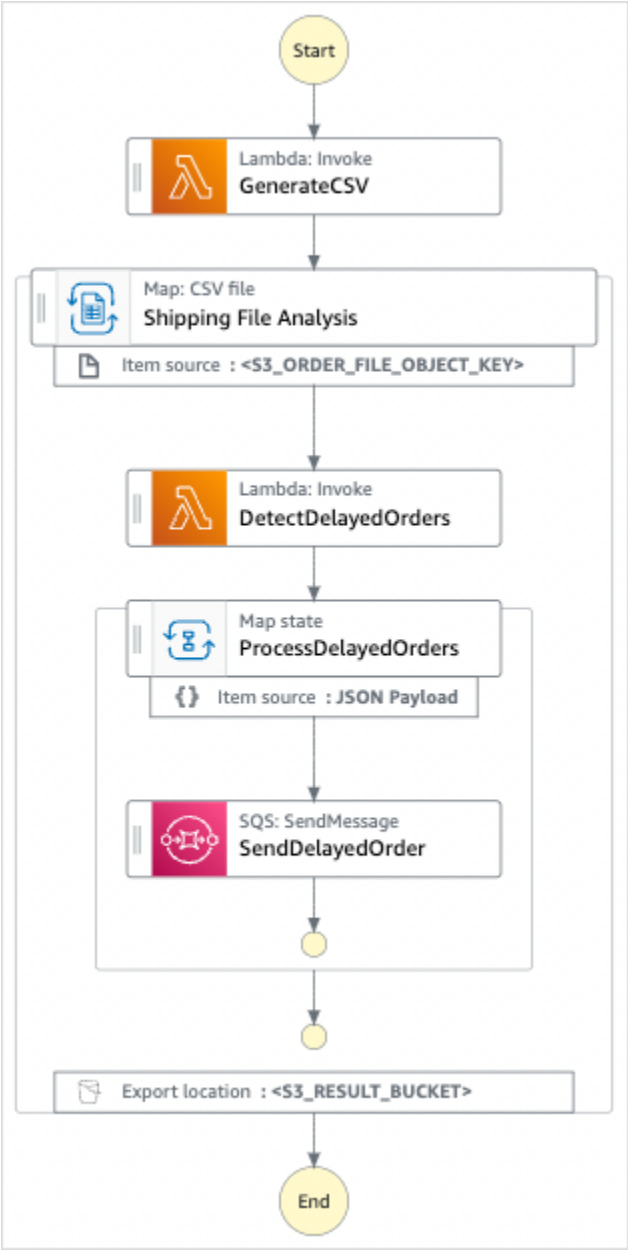
> **Important**
>
> Für jeden Service fallen Standardgebühren an.

Schritt 1: Erstellen des Zustandsautomaten und Bereitstellen von Ressourcen

1. Öffnen Sie die Step-Functions-Konsole und wählen Sie Zustandsautomaten erstellen aus.
2. Geben Sie **`Distributed Map to process a CSV file in S3`** in das Suchfeld ein und wählen Sie dann Distributed Map aus, um eine CSV-Datei in S3 aus den zurückgegebenen Suchergebnissen zu verarbeiten.
3. Wählen Sie Next (Weiter), um fortzufahren.
4. Step Functions listet die im ausgewählten Beispielprojekt AWS-Services verwendeten auf. Es zeigt auch ein Workflow-Diagramm für das Beispielprojekt. Stellen Sie dieses Projekt in Ihrem bereit AWS-Konto oder verwenden Sie es als Ausgangspunkt für die Erstellung Ihrer eigenen Projekte. Wählen Sie je nachdem, wie Sie fortfahren möchten, Demo ausführen oder darauf aufbauen aus.

   Informationen zu den Ressourcen, die für dieses Beispielprojekt erstellt werden, finden Sie unter AWS CloudFormation -Vorlage und zusätzliche Ressourcen.

   Die folgende Abbildung zeigt das Workflow-Diagramm für die verteilte Zuordnung zur Verarbeitung einer CSV-Datei im S3-Beispielprojekt:



5. Wählen Sie Vorlage verwenden, um mit Ihrer Auswahl fortzufahren.
6. Führen Sie eine der folgenden Aktionen aus:

   - Wenn Sie darauf aufbauen ausgewählt haben, erstellt Step Functions den Workflow-Prototyp für das ausgewählte Beispielprojekt. Step Functions stellt die in der Workflow-Definition aufgeführten Ressourcen nicht bereit.

     Ziehen Sie in Workflow Studio den Status aus dem Entwurfsmodus, Bundesstaaten-Browser um mit der Erstellung Ihres Workflow-Projekts fortzufahren. Oder wechseln Sie zu Codemodus, das einen integrierten Code-Editor ähnlich wie VS Code zum Aktualisieren der

Amazon States Language (ASL)-Definition Ihres Zustandsautomaten in der Step Functions-Konsole bietet. Weitere Informationen zur Verwendung von Workflow Studio zum Erstellen Ihrer Zustandsautomaten finden Sie unter Verwenden von Workflow Studio.

> **Important**
>
> Denken Sie daran, den Platzhalter Amazon-Ressourcennamen (ARN) für die im Beispielprojekt verwendeten Ressourcen zu aktualisieren, bevor Sie Ihren Workflow ausführen.

- Wenn Sie Demo ausführen ausgewählt haben, erstellt Step Functions ein schreibgeschütztes Beispielprojekt, das eine - AWS CloudFormation Vorlage verwendet, um die in dieser Vorlage aufgeführten AWS Ressourcen in Ihrem bereitzustellen AWS-Konto.

  > **Tip**
  >
  > Um die Definition des Zustandsautomaten des Beispielprojekts anzuzeigen, wählen Sie Code .

  Wenn Sie bereit sind, wählen Sie Bereitstellen und Ausführen aus, um das Beispielprojekt bereitzustellen und die Ressourcen zu erstellen.

  Es kann bis zu 10 Minuten dauern, bis diese Ressourcen und zugehörige IAM-Berechtigungen erstellt werden. Während Ihre Ressourcen bereitgestellt werden, können Sie den Link CloudFormation Stack-ID öffnen, um zu sehen, welche Ressourcen bereitgestellt werden.

  Nachdem alle Ressourcen im Beispielprojekt erstellt wurden, wird das neue Beispielprojekt auf der Seite Zustandsautomaten aufgeführt.

  > **Important**
  >
  > Für jeden in der CloudFormation Vorlage verwendeten Service können Standardgebühren anfallen.

Schritt 2: Ausführen des Zustandsautomaten

Nachdem alle Ressourcen bereitgestellt und bereitgestellt wurden, können Sie den Zustandsautomaten ausführen.

1. Wählen Sie auf der Seite Zustandsautomaten Ihr Beispielprojekt aus.
2. Wählen Sie auf der Beispielprojektseite Ausführung starten aus.
3. Gehen Sie im Dialogfeld Ausführung starten wie folgt vor:

   a. (Optional) Geben Sie Eingabewerte im JSON-Format ein, um Ihr Beispielprojekt auszuführen.

      Wenn Sie eine Demo ausführen ausgewählt haben, müssen Sie keine Ausführungseingabe angeben.

      > **Note**
      >
      > Wenn das bereitgestellte Demo-Projekt vorab ausgefüllte Eingabedaten für die Ausführung enthält, verwenden Sie diese Eingabe, um den Zustandsautomaten auszuführen.

   b. Wählen Sie Start execution (Ausführung starten) aus.

   c. (Optional) Die Step-Functions-Konsole leitet Sie zu einer Seite mit dem Titel mit Ihrer Ausführungs-ID weiter. Diese Seite wird als Ausführungsdetails-Seite bezeichnet. Auf dieser Seite können Sie die Ausführungsergebnisse überprüfen, während die Ausführung fortschreitet oder nachdem sie abgeschlossen sind.

      Nachdem die Ausführung abgeschlossen ist, wählen Sie in der Diagrammansicht einzelne Status und dann die einzelnen Registerkarten im Schrittdetails Bereich aus, um die Details der einzelnen Status anzuzeigen, einschließlich Eingabe, Ausgabe und Definition.

      - Weitere Informationen zu den Ausführungsinformationen, die Sie auf der Seite Ausführungsdetails anzeigen können, finden Sie unter Seite mit Ausführungsdetails – Schnittstellenübersicht.
      - Weitere Informationen zum Anzeigen der Ausführung eines Distributed Map-Status in der Konsole finden Sie unter Untersuchen der Kartenausführung.

d. (Optional) Überprüfen Sie die Ausführungsergebnisse, die in den Amazon S3-Bucket exportiert wurden. Zu diesen Ergebnissen gehören Daten wie Ausführungseingabe und -ausgabe, ARN und Ausführungsstatus. Weitere Informationen finden Sie unter ResultWriter.

Verarbeiten von Daten in einem Amazon S3-Bucket mit verteilter Zuordnung

Dieses Beispielprojekt zeigt, wie Sie den Status Verteilte Karte verwenden können, um große Daten zu verarbeiten, z. B. historische Wetterdaten zu analysieren und die Wetterzone zu identifizieren, bei der die durchschnittliche Temperatur jeden Monat am Arzt am höchsten ist. Die Wetterdaten werden in über 12.000 CSV-Dateien aufgezeichnet, die wiederum in einem Amazon S3-Bucket gespeichert werden.

Dieses Beispielprojekt enthält zwei Distributed Map-Status mit dem Namen Distributed S3 copy NOA Data und ProcessNOAAData . Verteilte S3-Kopier-NOA-Daten iterieren über die CSV-Dateien in einem öffentlichen Amazon S3-Bucket mit dem Namen noaa-gsod-pds und kopieren sie in einen Amazon S3-Bucket in Ihrem AWS-Konto. ProcessNOAAData iteriert über die kopierten Dateien und enthält eine Lambda-Funktion, die die Temperaturanalyse durchführt.

Das Beispielprojekt überprüft zunächst den Inhalt des Amazon S3-Buckets mit einem Aufruf der -ListObjectsV2-API-Aktion. Basierend auf der Anzahl der Schlüssel, die als Reaktion auf diesen Aufruf zurückgegeben wurden, trifft das Beispielprojekt eine der folgenden Entscheidungen:

- Wenn die Schlüsselanzahl größer oder gleich 1 ist, wechselt das Projekt in den Status ProcessNOAAData. Dieser Status der verteilten Karte enthält eine Lambda Funktion namens TemperatureFunction, die die Wetterstation findet, die jeden Monat die höchste durchschnittliche Temperatur hatte. Diese Funktion gibt ein Wörterbuch mit `year-month` als Schlüssel und ein Wörterbuch zurück, das Informationen über die Wetterstation als Wert enthält.
- Wenn die zurückgegebene Schlüsselanzahl 1 nicht überschreitet, listet der Status Verteilte S3-Kopie NOA-Daten alle Objekte aus dem öffentlichen Bucket auf noaa-gsod-pds und kopiert die einzelnen Objekte iterativ in Batches von 100 in einen anderen Bucket in Ihrem Konto. Eine Inline-Zuweisung führt das iterative Kopieren der Objekte durch.

  Nachdem alle Objekte kopiert wurden, wechselt das Projekt zur Verarbeitung der Wetterdaten in den Status ProcessNOAAData.

Das Beispielprojekt wechselt schließlich zu einer Reducer-LambdaFunktion, die eine endgültige Aggregation der von der TemperatureFunction Funktion zurückgegebenen Ergebnisse durchführt und die Ergebnisse in eine -Amazon DynamoDBTabelle schreibt.

Mit Distributed Map können Sie bis zu 10.000 parallele untergeordnete Workflow-Ausführungen gleichzeitig ausführen. In diesem Beispielprojekt ist die maximale Parallelität von ProcessNOAAData Distributed Map auf 3 000 festgelegt, was sie auf 3 000 parallele untergeordnete Workflow-Ausführungen beschränkt.

Dieses Beispielprojekt erstellt den Zustandsautomaten und die unterstützenden AWS Ressourcen und konfiguriert die zugehörigen IAM-Berechtigungen. Erkunden Sie dieses Beispielprojekt, um mehr über die Verwendung der verteilten Karte zur Orchestrierung großer, paralleler Workloads zu erfahren, oder verwenden Sie sie als Ausgangspunkt für Ihre eigenen Projekte.

> **Important**
>
> Dieses Beispielprojekt ist nur in der Region USA Ost (Nord-Virginia) verfügbar.

AWS CloudFormation -Vorlage und zusätzliche Ressourcen

Sie verwenden eine CloudFormation Vorlage, um dieses Beispielprojekt bereitzustellen. Diese Vorlage erstellt die folgenden Ressourcen in Ihrem AWS-Konto:

- Ein Step-Functions-Zustandsautomat.
- Ausführungsrolle für den Zustandsautomaten. Diese Rolle gewährt die Berechtigungen, die Ihr Zustandsautomat für den Zugriff auf andere AWS-Services und Ressourcen benötigt, wie z. B. die Invoke-Aktion der Lambda-Funktion.
- Ein Amazon S3-Bucket mit dem Namen `NOAADataBucket`. Dieser Bucket enthält die CSV-Dateien mit Wetterdaten.
- Eine Lambda-Funktion namens `ReducerFunction`, die eine endgültige Aggregation der Wetterdaten durchführt und die Ergebnisse in eine Amazon-DynamoDB-Tabelle schreibt.
- Ausführungsrolle für die Reducer-Lambda-Funktion. Diese Rolle gewährt der Funktion die Berechtigung, auf andere zuzugreifen AWS-Services.
- Ein Amazon S3-Ausgabe-Bucket mit dem Namen `ResultsBucket` , um die Wetteranalyseergebnisse zu speichern.

- Eine DynamoDB-Tabelle mit dem Namen `ResultsDynamoDBTable`, die die vom zurückgegebenen Ergebnisse enthält`ReducerFunction`.
- Eine Lambda-Funktion mit dem Namen `TemperatureFunction`, die die höchste monatliche durchschnittliche Temperatur findet.
- Ausführungsrolle für die Lambda-Funktion. Diese Rolle gewährt der Funktion die Berechtigung, auf andere zuzugreifen AWS-Services.
- Eine CloudWatch Protokollgruppe, die Informationen über den Ausführungsverlauf des Zustandsautomaten speichert.

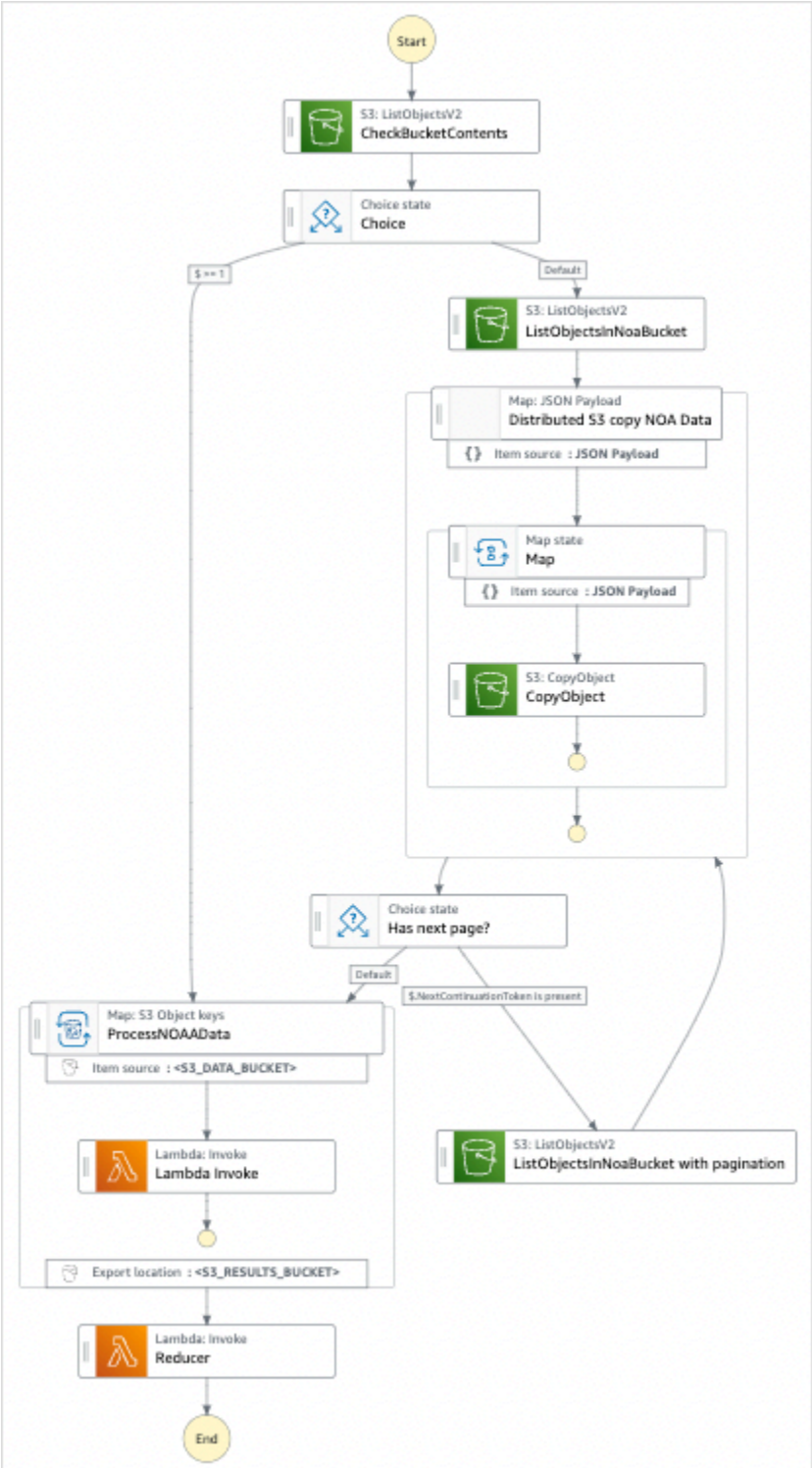
> **Important**
>
> Für jeden Service fallen Standardgebühren an.

Schritt 1: Erstellen des Zustandsautomaten und Bereitstellen von Ressourcen

1. Öffnen Sie die Step-Functions-Konsole und wählen Sie Zustandsautomaten erstellen aus.
2. Geben Sie **`Distributed Map to process files in S3`** in das Suchfeld ein und wählen Sie dann Distributed Map aus, um Dateien in S3 aus den zurückgegebenen Suchergebnissen zu verarbeiten.
3. Wählen Sie Next (Weiter), um fortzufahren.
4. Step Functions listet die im ausgewählten Beispielprojekt AWS-Services verwendeten auf. Es zeigt auch ein Workflow-Diagramm für das Beispielprojekt. Stellen Sie dieses Projekt in Ihrem bereit AWS-Konto oder verwenden Sie es als Ausgangspunkt für die Erstellung Ihrer eigenen Projekte. Wählen Sie je nachdem, wie Sie fortfahren möchten, Demo ausführen oder darauf aufbauen aus.

   Informationen zu den Ressourcen, die für dieses Beispielprojekt erstellt werden, finden Sie unter AWS CloudFormation -Vorlage und zusätzliche Ressourcen.

   Die folgende Abbildung zeigt das Workflow-Diagramm für die verteilte Karte zur Verarbeitung von Dateien im S3-Beispielprojekt:



5. Wählen Sie Vorlage verwenden, um mit Ihrer Auswahl fortzufahren.
6. Führen Sie eine der folgenden Aktionen aus:

- Wenn Sie darauf aufbauen ausgewählt haben, erstellt Step Functions den Workflow-Prototyp für das ausgewählte Beispielprojekt. Step Functions stellt die in der Workflow-Definition aufgeführten Ressourcen nicht bereit.

  Ziehen Sie in Workflow Studio den Status aus dem Entwurfsmodus, Bundesstaaten-Browser um mit der Erstellung Ihres Workflow-Projekts fortzufahren. Oder wechseln Sie zu Codemodus, das einen integrierten Code-Editor ähnlich wie VS Code zum Aktualisieren der Amazon States Language (ASL)-Definition Ihres Zustandsautomaten in der Step Functions-Konsole bietet. Weitere Informationen zur Verwendung von Workflow Studio zum Erstellen Ihrer Zustandsautomaten finden Sie unter Verwenden von Workflow Studio.

  > **Important**
  >
  > Denken Sie daran, den Amazon-Ressourcennamen (ARN) des Platzhalters für die im Beispielprojekt verwendeten Ressourcen zu aktualisieren, bevor Sie Ihren Workflow ausführen.

- Wenn Sie Demo ausführen ausgewählt haben, erstellt Step Functions ein schreibgeschütztes Beispielprojekt, das eine - AWS CloudFormation Vorlage verwendet, um die in dieser Vorlage aufgeführten AWS Ressourcen in Ihrem bereitzustellen AWS-Konto.

  > **Tip**
  >
  > Um die Definition des Zustandsautomaten des Beispielprojekts anzuzeigen, wählen Sie Code .

  Wenn Sie bereit sind, wählen Sie Bereitstellen und Ausführen aus, um das Beispielprojekt bereitzustellen und die Ressourcen zu erstellen.

  Es kann bis zu 10 Minuten dauern, bis diese Ressourcen und zugehörige IAM-Berechtigungen erstellt werden. Während Ihre Ressourcen bereitgestellt werden, können Sie den Link CloudFormation Stack-ID öffnen, um zu sehen, welche Ressourcen bereitgestellt werden.

  Nachdem alle Ressourcen im Beispielprojekt erstellt wurden, wird das neue Beispielprojekt auf der Seite Zustandsautomaten aufgeführt.

> **Important**
>
> Für jeden in der CloudFormation Vorlage verwendeten Service können Standardgebühren anfallen.

Schritt 2: Ausführen des Zustandsautomaten

Nachdem alle Ressourcen bereitgestellt und bereitgestellt wurden, können Sie den Zustandsautomaten ausführen.

1. Wählen Sie auf der Seite Zustandsautomaten Ihr Beispielprojekt aus.
2. Wählen Sie auf der Beispielprojektseite Ausführung starten aus.
3. Gehen Sie im Dialogfeld Ausführung starten wie folgt vor:

   a. (Optional) Geben Sie Eingabewerte im JSON-Format ein, um Ihr Beispielprojekt auszuführen.

   Wenn Sie eine Demo ausführen ausgewählt haben, müssen Sie keine Ausführungseingabe angeben.

   > **Note**
   >
   > Wenn das bereitgestellte Demo-Projekt vorab ausgefüllte Eingabedaten für die Ausführung enthält, verwenden Sie diese Eingabe, um den Zustandsautomaten auszuführen.

   b. Wählen Sie Start execution (Ausführung starten) aus.

   c. (Optional) Die Step-Functions-Konsole leitet Sie zu einer Seite mit dem Titel mit Ihrer Ausführungs-ID weiter. Diese Seite wird als Ausführungsdetails-Seite bezeichnet. Auf dieser Seite können Sie die Ausführungsergebnisse überprüfen, während die Ausführung fortschreitet oder nachdem sie abgeschlossen sind.

   Nachdem die Ausführung abgeschlossen ist, wählen Sie in der Diagrammansicht einzelne Status und dann die einzelnen Registerkarten im Schrittdetails Bereich aus, um die Details jedes Status anzuzeigen, einschließlich Eingabe, Ausgabe und Definition.

- Weitere Informationen zu den Ausführungsinformationen, die Sie auf der Seite Ausführungsdetails anzeigen können, finden Sie unter Seite mit Ausführungsdetails – Schnittstellenübersicht.
- Weitere Informationen zum Anzeigen der Ausführung eines Distributed Map-Status in der Konsole finden Sie unter Untersuchen der Kartenausführung.

d. (Optional) Überprüfen Sie die Ausführungsergebnisse, die in den Amazon S3-Bucket exportiert wurden. Zu diesen Ergebnissen gehören Daten wie Ausführungseingabe und -ausgabe, ARN und Ausführungsstatus. Weitere Informationen finden Sie unter ResultWriter.

Schulen eines Machine Learning-Modells

Dieses Beispielprojekt zeigt, wie ein Modell für maschinelles Lernen verwendet SageMaker und AWS Step Functions trainiert wird und wie ein Testdatensatz stapelweise transformiert wird.

In diesem Projekt verwendet Step Functions eine Lambda-Funktion, um einen Amazon S3 S3-Bucket mit einem Testdatensatz zu versorgen. Anschließend trainiert es ein Modell für maschinelles Lernen und führt mithilfe der SageMaker Serviceintegration eine Batch-Transformation durch.

Weitere Informationen zu SageMaker Step Functions Functions-Dienstintegrationen finden Sie im Folgenden:

- Verwendung AWS Step Functions mit anderen Diensten
- SageMaker Mit Step Functions verwalten

> **Note**
>
> Für dieses Beispielprojekt können Gebühren anfallen.
> Für neue AWS Benutzer ist ein kostenloses Nutzungskontingent verfügbar. Im Rahmen dieses Kontingents sind die Services bis zu einem bestimmten Nutzungsumfang kostenlos. Weitere Informationen zu den AWS Kosten und dem kostenlosen Kontingent finden Sie unter SageMaker Preise.

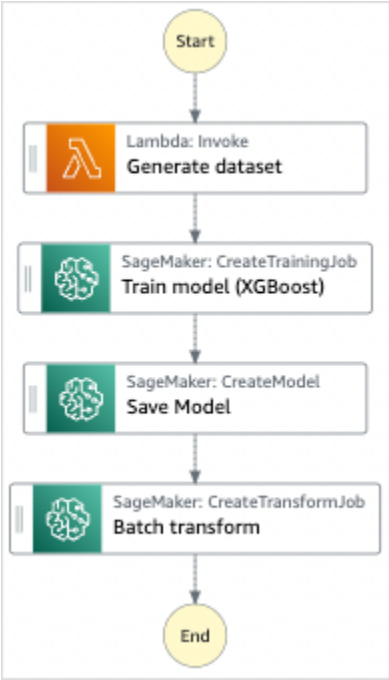
Schritt 1: Erstellen Sie den Zustandsmaschine und stellen Sie Ressourcen bereit

1. Öffnen Sie die Step Functions Functions-Konsole und wählen Sie Create State Machine.
2. Geben Sie **`Train a machine learning model`** etwas in das Suchfeld ein und wählen Sie dann aus den zurückgegebenen Suchergebnissen die Option Modell für maschinelles Lernen trainieren aus.
3. Wählen Sie Next (Weiter), um fortzufahren.
4. Step Functions listet die Funktionen auf, die in dem von Ihnen ausgewählten Beispielprojekt AWS-Services verwendet wurden. Außerdem wird ein Workflow-Diagramm für das Beispielprojekt angezeigt. Stellen Sie dieses Projekt für Ihr Projekt bereit AWS-Konto oder verwenden Sie es als Ausgangspunkt für die Erstellung Ihrer eigenen Projekte. Je nachdem, wie Sie vorgehen möchten, wählen Sie „Demo ausführen“ oder „Darauf aufbauen“.

   In diesem Beispielprojekt werden die folgenden Ressourcen bereitgestellt:

   - Eine Funktion AWS Lambda
   - Ein Amazon Simple Storage Service (Amazon S3)-Bucket
   - Eine AWS Step Functions Staatsmaschine
   - Verwandte AWS Identity and Access Management Rollen (IAM)

   Die folgende Abbildung zeigt das Workflow-Diagramm für das Beispielprojekt Train a machine learning model:



5. Wählen Sie Vorlage verwenden, um mit Ihrer Auswahl fortzufahren.
6. Führen Sie eine der folgenden Aktionen aus:

   - Wenn Sie „Darauf aufbauen“ ausgewählt haben, erstellt Step Functions den Workflow-Prototyp für das von Ihnen ausgewählte Beispielprojekt. Step Functions stellt die in der Workflow-Definition aufgeführten Ressourcen nicht bereit.

     Ziehen Sie in Workflow Studio Status per Drag-and-Drop aus demEntwurfsmodus, Bundesstaaten-Browser um mit der Erstellung Ihres Workflow-Prototyps fortzufahren. Oder wechseln Sie zu demCodemodus, der einen integrierten Code-Editor bietet, der VS Code ähnelt, um die Amazon States Language (ASL-) Definition Ihrer Zustandsmaschine in der Step Functions Functions-Konsole zu aktualisieren. Weitere Informationen zur Verwendung von Workflow Studio zum Erstellen Ihrer Zustandsmaschinen finden Sie unterVerwenden von Workflow Studio.

     > **Important**
     >
     > Denken Sie daran, den Platzhalter Amazon Resource Name (ARN) für die im Beispielprojekt verwendeten Ressourcen zu aktualisieren, bevor Sie Ihren Workflow ausführen.

- Wenn Sie Eine Demo ausführen ausgewählt haben, erstellt Step Functions ein schreibgeschütztes Beispielprojekt, das eine AWS CloudFormation Vorlage verwendet, um die in dieser Vorlage aufgeführten AWS Ressourcen für Ihr Projekt bereitzustellen. AWS-Konto

  > **Tip**
  >
  > Um die State-Machine-Definition des Beispielprojekts anzuzeigen, wählen Sie Code.

  Wenn Sie bereit sind, wählen Sie Deploy and run aus, um das Beispielprojekt bereitzustellen und die Ressourcen zu erstellen.

  Es kann bis zu 10 Minuten dauern, bis diese Ressourcen und die zugehörigen IAM-Berechtigungen erstellt sind. Während der Bereitstellung Ihrer Ressourcen können Sie den CloudFormation Stack-ID-Link öffnen, um zu sehen, welche Ressourcen bereitgestellt werden.

  Nachdem alle Ressourcen im Beispielprojekt erstellt wurden, wird das neue Beispielprojekt auf der Seite State Machines aufgeführt.

  > **Important**
  >
  > Für jeden in der CloudFormation Vorlage verwendeten Dienst können Standardgebühren anfallen.

Schritt 2: Führen Sie die Zustandsmaschine aus

1. Wählen Sie auf der Seite State Machines Ihr Beispielprojekt aus.
2. Wählen Sie auf der Seite mit dem Beispielprojekt die Option Ausführung starten aus.
3. Gehen Sie im Dialogfeld Ausführung starten wie folgt vor:

   1. (Optional) Um Ihre Ausführung zu identifizieren, können Sie im Feld Name einen Namen dafür angeben. Standardmäßig generiert Step Functions automatisch einen eindeutigen Ausführungsnamen.

> **Note**
>
> Mit Step Functions können Sie Namen für Zustandsmaschinen, Ausführungen und Aktivitäten sowie Beschriftungen erstellen, die Nicht-ASCII-Zeichen enthalten. Diese Nicht-ASCII-Namen funktionieren nicht mit Amazon. CloudWatch Um sicherzustellen, dass Sie CloudWatch Messwerte verfolgen können, wählen Sie einen Namen, der nur ASCII-Zeichen verwendet.

2. (Optional) Geben Sie in das Eingabefeld Eingabewerte im JSON-Format ein, um Ihren Workflow auszuführen.

   Wenn Sie sich dafür entschieden haben, eine Demo auszuführen, müssen Sie keine Ausführungseingaben angeben.

   > **Note**
   >
   > Wenn das von Ihnen bereitgestellte Demo-Projekt vorab aufgefüllte Eingabedaten für die Ausführung enthält, verwenden Sie diese Eingabe, um die Zustandsmaschine auszuführen.

3. Wählen Sie Start execution (Ausführung starten) aus.
4. Die Step Functions Functions-Konsole leitet Sie zu einer Seite weiter, die mit Ihrer Ausführungs-ID betitelt ist. Diese Seite wird als Seite mit den Ausführungsdetails bezeichnet. Auf dieser Seite können Sie die Ausführungsergebnisse im Verlauf der Ausführung oder nach deren Abschluss überprüfen.

   Um die Ausführungsergebnisse zu überprüfen, wählen Sie in der Diagrammansicht einzelne Status aus und wählen Sie dann die einzelnen Registerkarten im Schrittdetails Bereich, um die Details der einzelnen Status, einschließlich Eingabe, Ausgabe und Definition, anzuzeigen. Einzelheiten zu den Ausführungsinformationen, die Sie auf der Seite mit den Ausführungsdetails einsehen können, finden Sie unterSeite mit Ausführungsdetails – Schnittstellenübersicht.

Code des Zustandsautomaten aus diesem Beispiel

Die Zustandsmaschine in diesem Beispielprojekt integriert sich mit SageMaker und AWS Lambda durch direkte Übergabe von Parametern an diese Ressourcen und verwendet einen Amazon S3 S3-Bucket für die Trainingsdatenquelle und -ausgabe.

Sehen Sie sich diese Beispiel-Zustandsmaschine an, um zu sehen, wie Step Functions Lambda und SageMaker steuert.

Weitere Informationen darüber, wie AWS Step Functions Sie andere AWS Dienste steuern können, finden Sie unterVerwendung AWS Step Functions mit anderen Diensten.

```
{
  "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
    }
  }
}
```

Informationen zur Konfiguration von IAM bei der Verwendung von Step Functions mit anderen AWS Diensten finden Sie unterIAM-Richtlinien für integrierte Dienste.

IAM-Beispiel

Diese vom Beispielprojekt generierten Beispielrichtlinien AWS Identity and Access Management (IAM) beinhalten die geringsten Rechte, die für die Ausführung der Zustandsmaschine und der zugehörigen Ressourcen erforderlich sind. Wir empfehlen, dass Sie nur die Berechtigungen in Ihre IAM-Richtlinien aufnehmen, die erforderlich sind.

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

Die folgende Richtlinie ermöglicht es der Lambda-Funktion, den Amazon S3 S3-Bucket mit Beispieldaten zu versorgen.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::trainandbatchtransform-s3bucket-1jn1le6gadwfz/*",
            "Effect": "Allow"
        }
    ]
}
```

Informationen zur Konfiguration von IAM bei der Verwendung von Step Functions mit anderen AWS Diensten finden Sie unterIAM-Richtlinien für integrierte Dienste.

Optimieren eines Machine Learning-Modells

In diesem Beispielprojekt wird gezeigt SageMaker , wie Sie die Hyperparameter eines Modells für maschinelles Lernen optimieren und einen Testdatensatz stapelweise transformieren können.

In diesem Projekt verwendet Step Functions eine Lambda-Funktion, um einen Amazon S3 S3-Bucket mit einem Testdatensatz zu versorgen. Anschließend erstellt es mithilfe der SageMakerServiceintegration einen Hyperparameter-Tuning-Job. Anschließend extrahiert es

mithilfe einer Lambda-Funktion den Datenpfad, speichert das Optimierungsmodell, extrahiert den Modellnamen und führt dann einen Batch-Transformationsjob aus, um Inferenzen durchzuführen. SageMaker

Weitere Informationen zu SageMaker Step Functions Functions-Dienstintegrationen finden Sie im Folgenden:

- Verwendung AWS Step Functions mit anderen Diensten
- SageMaker Mit Step Functions verwalten

> **Note**
>
> Für dieses Beispielprojekt können Gebühren anfallen.
> Für neue AWS Benutzer ist ein kostenloses Nutzungskontingent verfügbar. Im Rahmen dieses Kontingents sind die Services bis zu einem bestimmten Nutzungsumfang kostenlos. Weitere Informationen zu den AWS Kosten und dem kostenlosen Kontingent finden Sie unter SageMakerPreise.

Schritt 1: Erstellen Sie den Zustandsmaschine und stellen Sie Ressourcen bereit

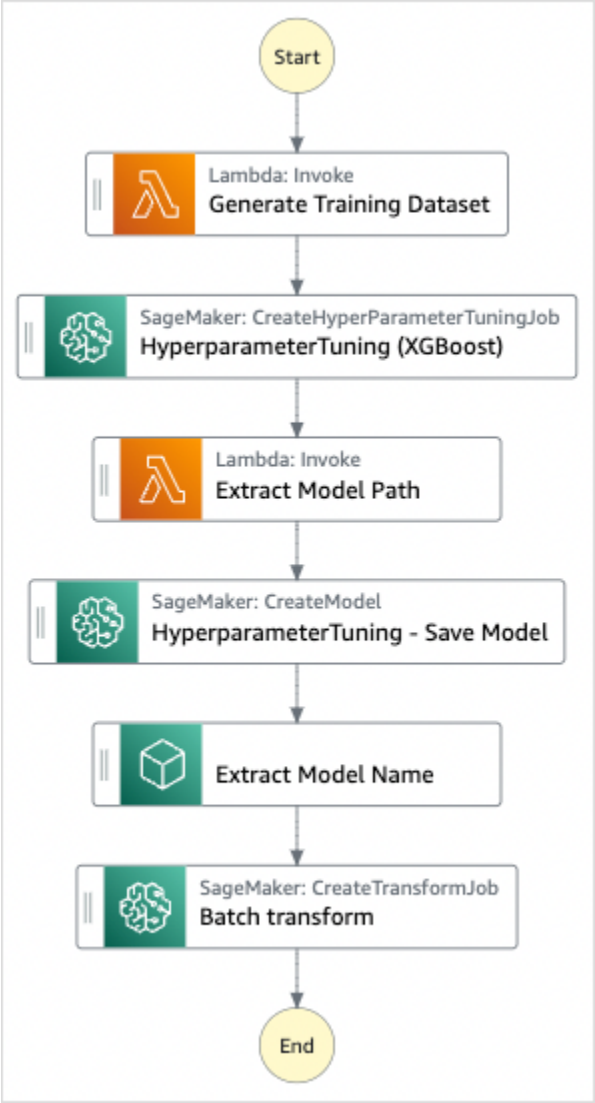
1. Öffnen Sie die Step Functions Functions-Konsole und wählen Sie Create State Machine.
2. Geben Sie **`Tune a machine learning model`** etwas in das Suchfeld ein und wählen Sie dann aus den zurückgegebenen Suchergebnissen die Option Modell für maschinelles Lernen optimieren aus.
3. Wählen Sie Next (Weiter), um fortzufahren.
4. Step Functions listet die Funktionen auf, die in dem von Ihnen ausgewählten Beispielprojekt AWS-Services verwendet wurden. Außerdem wird ein Workflow-Diagramm für das Beispielprojekt angezeigt. Stellen Sie dieses Projekt für Ihr Projekt bereit AWS-Konto oder verwenden Sie es als Ausgangspunkt für die Erstellung Ihrer eigenen Projekte. Je nachdem, wie Sie vorgehen möchten, wählen Sie „Demo ausführen“ oder „Darauf aufbauen“.

   In diesem Beispielprojekt werden die folgenden Ressourcen bereitgestellt:

   - Drei Funktionen AWS Lambda

- Ein Amazon Simple Storage Service (Amazon S3)-Bucket
- Eine AWS Step Functions Staatsmaschine
- Verwandte AWS Identity and Access Management Rollen (IAM)

Die folgende Abbildung zeigt das Workflow-Diagramm für das Beispielprojekt „Modell für maschinelles Lernen optimieren“:



5. Wählen Sie Vorlage verwenden, um mit Ihrer Auswahl fortzufahren.
6. Führen Sie eine der folgenden Aktionen aus:

   - Wenn Sie „Darauf aufbauen“ ausgewählt haben, erstellt Step Functions den Workflow-Prototyp für das von Ihnen ausgewählte Beispielprojekt. Step Functions stellt die in der Workflow-Definition aufgeführten Ressourcen nicht bereit.

Ziehen Sie in Workflow Studio Status per Drag-and-Drop aus demEntwurfsmodus, Bundesstaaten-Browser um mit der Erstellung Ihres Workflow-Prototyps fortzufahren. Oder wechseln Sie zu demCodemodus, der einen integrierten Code-Editor bietet, der VS Code ähnelt, um die Amazon States Language (ASL-) Definition Ihrer Zustandsmaschine in der Step Functions Functions-Konsole zu aktualisieren. Weitere Informationen zur Verwendung von Workflow Studio zum Erstellen Ihrer Zustandsmaschinen finden Sie unterVerwenden von Workflow Studio.

> **Important**
>
> Denken Sie daran, den Platzhalter Amazon Resource Name (ARN) für die im Beispielprojekt verwendeten Ressourcen zu aktualisieren, bevor Sie Ihren Workflow ausführen.

- Wenn Sie Eine Demo ausführen ausgewählt haben, erstellt Step Functions ein schreibgeschütztes Beispielprojekt, das eine AWS CloudFormation Vorlage verwendet, um die in dieser Vorlage aufgeführten AWS Ressourcen für Ihr Projekt bereitzustellen. AWS-Konto

> **Tip**
>
> Um die State-Machine-Definition des Beispielprojekts anzuzeigen, wählen Sie Code.

Wenn Sie bereit sind, wählen Sie Deploy and run aus, um das Beispielprojekt bereitzustellen und die Ressourcen zu erstellen.

Es kann bis zu 10 Minuten dauern, bis diese Ressourcen und die zugehörigen IAM-Berechtigungen erstellt sind. Während der Bereitstellung Ihrer Ressourcen können Sie den CloudFormation Stack-ID-Link öffnen, um zu sehen, welche Ressourcen bereitgestellt werden.

Nachdem alle Ressourcen im Beispielprojekt erstellt wurden, wird das neue Beispielprojekt auf der Seite State Machines aufgeführt.

> **Important**
>
> Für jeden in der CloudFormation Vorlage verwendeten Dienst können Standardgebühren anfallen.

Schritt 2: Führen Sie die Zustandsmaschine aus

1. Wählen Sie auf der Seite State Machines Ihr Beispielprojekt aus.
2. Wählen Sie auf der Seite mit dem Beispielprojekt die Option Ausführung starten aus.
3. Gehen Sie im Dialogfeld Ausführung starten wie folgt vor:

    1. (Optional) Um Ihre Ausführung zu identifizieren, können Sie im Feld Name einen Namen dafür angeben. Standardmäßig generiert Step Functions automatisch einen eindeutigen Ausführungsnamen.

        > **Note**
        >
        > Mit Step Functions können Sie Namen für Zustandsmaschinen, Ausführungen und Aktivitäten sowie Beschriftungen erstellen, die Nicht-ASCII-Zeichen enthalten. Diese Nicht-ASCII-Namen funktionieren nicht mit Amazon. CloudWatch Um sicherzustellen, dass Sie CloudWatch Messwerte verfolgen können, wählen Sie einen Namen, der nur ASCII-Zeichen verwendet.

    2. (Optional) Geben Sie in das Eingabefeld Eingabewerte im JSON-Format ein, um Ihren Workflow auszuführen.

        Wenn Sie sich dafür entschieden haben, eine Demo auszuführen, müssen Sie keine Ausführungseingaben angeben.

        > **Note**
        >
        > Wenn das von Ihnen bereitgestellte Demo-Projekt vorab aufgefüllte Eingabedaten für die Ausführung enthält, verwenden Sie diese Eingabe, um die Zustandsmaschine auszuführen.

    3. Wählen Sie Start execution (Ausführung starten) aus.
    4. Die Step Functions Functions-Konsole leitet Sie zu einer Seite weiter, die mit Ihrer Ausführungs-ID betitelt ist. Diese Seite wird als Seite mit den Ausführungsdetails bezeichnet. Auf dieser Seite können Sie die Ausführungsergebnisse im Verlauf oder nach Abschluss der Ausführung überprüfen.

        Um die Ausführungsergebnisse zu überprüfen, wählen Sie in der Diagrammansicht einzelne Status aus und wählen Sie dann die einzelnen Registerkarten im Schrittdetails Bereich,

um die Details der einzelnen Status, einschließlich Eingabe, Ausgabe und Definition, anzuzeigen. Einzelheiten zu den Ausführungsinformationen, die Sie auf der Seite mit den Ausführungsdetails einsehen können, finden Sie unterSeite mit Ausführungsdetails – Schnittstellenübersicht.

Code des Zustandsautomaten aus diesem Beispiel

Die Zustandsmaschine in diesem Beispielprojekt integriert sich mit SageMaker und AWS Lambda durch direkte Übergabe von Parametern an diese Ressourcen und verwendet einen Amazon S3 S3-Bucket für die Trainingsdatenquelle und -ausgabe.

Sehen Sie sich diese Beispiel-Zustandsmaschine an, um zu sehen, wie Step Functions Lambda und SageMaker steuert.

Weitere Informationen darüber, wie AWS Step Functions Sie andere AWS Dienste steuern können, finden Sie unterVerwendung AWS Step Functions mit anderen Diensten.

```
{
    "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
        }
    }
}
```

Informationen zur Konfiguration von IAM bei der Verwendung von Step Functions mit anderen AWS Diensten finden Sie unterIAM-Richtlinien für integrierte Dienste.

IAM-Beispiele

Diese vom Beispielprojekt generierten Beispielrichtlinien AWS Identity and Access Management (IAM) beinhalten die geringsten Rechte, die für die Ausführung der Zustandsmaschine und der zugehörigen Ressourcen erforderlich sind. Wir empfehlen, dass Sie nur die Berechtigungen in Ihre IAM-Richtlinien aufnehmen, die erforderlich sind.

Die folgende IAM-Richtlinie ist an die Zustandsmaschine angehängt und ermöglicht der Ausführung der Zustandsmaschine den Zugriff auf die erforderlichen SageMaker Lambda- und Amazon S3 S3-Ressourcen.

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

Auf die folgende IAM-Richtlinie wird in den `HyperparameterTuning` Feldern `TrainingJobDefinition` und des Bundesstaates verwiesen. `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"
        }
    ]
}
```

Die folgende IAM-Richtlinie ermöglicht es der Lambda-Funktion, den Amazon S3 S3-Bucket mit Beispieldaten zu versorgen.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:PutObject"
```

```
            ],
            "Resource": "arn:aws:s3:::stepfunctionssample-sagemak-
bucketformodelanddata-80fblmdlcs9f/*",
            "Effect": "Allow"
        }
    ]
}
```

Informationen zur Konfiguration von IAM bei der Verwendung von Step Functions mit anderen AWS Diensten finden Sie unterIAM-Richtlinien für integrierte Dienste.

Verarbeiten von Nachrichten mit hohem Volumen aus Amazon SQS (Express Workflows)

Dieses Beispielprojekt zeigt, wie Sie einen AWS Step Functions Express-Workflow verwenden, um Nachrichten oder Daten aus einer Ereignisquelle mit hohem Volumen zu verarbeiten, z. B. Amazon Simple Queue Service (Amazon SQS). Da Express-Workflows mit sehr hoher Rate gestartet werden können, eignen sie sich ideal für die Verarbeitung von hochvolumigen Ereignissen oder für Streamingdaten-Workloads.

Im Folgenden finden Sie zwei häufig verwendete Methoden für die Ausführung Ihres Zustandsautomaten über eine Ereignisquelle:

- Konfigurieren Sie eine Amazon- CloudWatch Events-Regel, um eine Ausführung des Zustandsautomaten zu starten, wenn die Ereignisquelle ein Ereignis ausgibt. Weitere Informationen finden Sie unter Erstellen einer CloudWatch Ereignisregel, die bei einem Ereignis ausgelöst wird.
- Ordnen Sie die Ereignisquelle einer Lambda-Funktion zu und schreiben Sie Funktionscode zur Ausführung Ihres Zustandsautomaten. Die AWS Lambda Funktion wird jedes Mal aufgerufen, wenn Ihre Ereignisquelle ein Ereignis ausgibt, und startet dann eine Zustandsautomaten-Ausführung. Weitere Informationen finden Sie unter Verwenden von AWS Lambda mit Amazon SQS.

Dieses Beispielprojekt verwendet die zweite Methode, um eine Ausführung jedes Mal zu starten, wenn die Amazon SQS-Warteschlange eine Nachricht sendet. Sie können eine ähnliche Konfiguration verwenden, um die Ausführung von Express Workflows aus anderen Ereignisquellen wie Amazon Simple Storage Service (Amazon S3), Amazon DynamoDB und Amazon Kinesis auszulösen.

Weitere Informationen zu Express-Workflows- und Step-Functions-Serviceintegrationen finden Sie im Folgenden:

- Standard- und Express-Workflows
- Verwendung AWS Step Functions mit anderen Diensten
- Kontingente

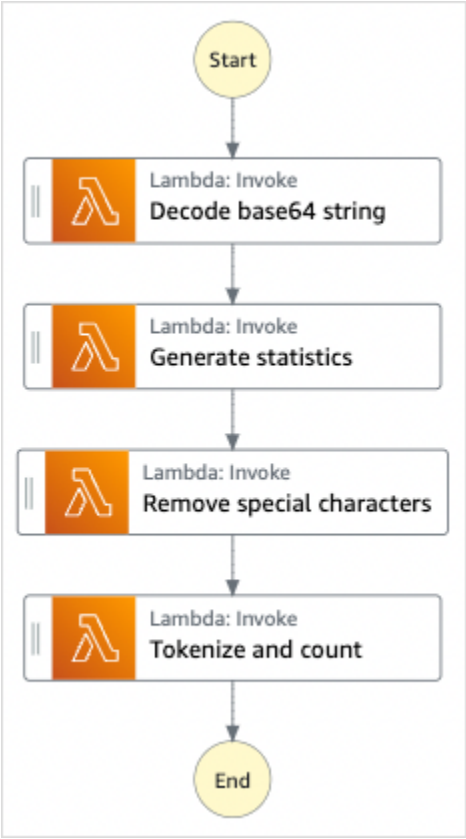
Schritt 1: Erstellen des Zustandsautomaten und Bereitstellen von Ressourcen

1. Öffnen Sie die Step-Functions-Konsole und wählen Sie Zustandsautomaten erstellen aus.
2. Geben Sie **`Process high-volume messages from SQS`** in das Suchfeld ein und wählen Sie dann in den zurückgegebenen Suchergebnissen die Option Nachrichten mit hohem Volumen aus SQS verarbeiten aus.
3. Wählen Sie Next (Weiter), um fortzufahren.
4. Step Functions listet die im ausgewählten Beispielprojekt AWS-Services verwendeten auf. Es zeigt auch ein Workflow-Diagramm für das Beispielprojekt. Stellen Sie dieses Projekt in Ihrem bereit AWS-Konto oder verwenden Sie es als Ausgangspunkt für die Erstellung Ihrer eigenen Projekte. Wählen Sie je nachdem, wie Sie fortfahren möchten, Demo ausführen oder darauf aufbauen aus.

   Dieses Beispielprojekt stellt die folgenden Ressourcen bereit:

   - Vier Lambda-Funktionen
   - Eine Amazon-SQS-Warteschlange
   - Ein - AWS Step Functions Zustandsautomat
   - Verwandte AWS Identity and Access Management (IAM)-Rollen

   Die folgende Abbildung zeigt das Workflow-Diagramm für das Beispielprojekt Verarbeiten von Nachrichten mit hohem Volumen aus SQS:



5. Wählen Sie Vorlage verwenden, um mit Ihrer Auswahl fortzufahren.
6. Führen Sie eine der folgenden Aktionen aus:

   - Wenn Sie darauf aufbauen ausgewählt haben, erstellt Step Functions den Workflow-Prototyp für das ausgewählte Beispielprojekt. Step Functions stellt die in der Workflow-Definition aufgeführten Ressourcen nicht bereit.

     Ziehen Sie in Workflow Studio den Status aus dem Entwurfsmodus, Bundesstaaten-Browser um mit der Erstellung Ihres Workflow-Projekts fortzufahren. Oder wechseln Sie zu Codemodus, das einen integrierten Code-Editor ähnlich VS Code zum Aktualisieren der Amazon States Language (ASL)-Definition Ihres Zustandsautomaten in der Step Functions-Konsole bietet. Weitere Informationen zur Verwendung von Workflow Studio zum Erstellen Ihrer Zustandsautomaten finden Sie unter Verwenden von Workflow Studio.

> **Important**
>
> Denken Sie daran, den Amazon-Ressourcennamen (ARN) des Platzhalters für die im Beispielprojekt verwendeten Ressourcen zu aktualisieren, bevor Sie Ihren Workflow ausführen.

- Wenn Sie Demo ausführen ausgewählt haben, erstellt Step Functions ein schreibgeschütztes Beispielprojekt, das eine - AWS CloudFormation Vorlage verwendet, um die in dieser Vorlage aufgeführten AWS Ressourcen in Ihrem bereitzustellen AWS-Konto.

  > **Tip**
  >
  > Um die Definition des Zustandsautomaten des Beispielprojekts anzuzeigen, wählen Sie Code .

  Wenn Sie bereit sind, wählen Sie Bereitstellen und Ausführen aus, um das Beispielprojekt bereitzustellen und die Ressourcen zu erstellen.

  Es kann bis zu 10 Minuten dauern, bis diese Ressourcen und zugehörige IAM-Berechtigungen erstellt werden. Während Ihre Ressourcen bereitgestellt werden, können Sie den Link CloudFormation Stack-ID öffnen, um zu sehen, welche Ressourcen bereitgestellt werden.

  Nachdem alle Ressourcen im Beispielprojekt erstellt wurden, wird das neue Beispielprojekt auf der Seite Zustandsautomaten aufgeführt.

  > **Important**
  >
  > Für jeden in der CloudFormation Vorlage verwendeten Service können Standardgebühren anfallen.

Schritt 2: Auslösen der Ausführung des Zustandsautomaten

1. Öffnen Sie die Amazon-SQS-Konsole.
2. Wählen Sie die Warteschlange aus, die vom Beispielprojekt erstellt wurde.

Der Name lautet in etwa Example-SQSQueue-wJalrXUtnFEMI.

3. Wählen Sie in der Liste Queue Actions (Warteschlangenaktionen) die Option Send a Message (Eine Nachricht senden) aus.
4. Verwenden Sie die Schaltfläche zum Kopieren, um die folgende Nachricht zu kopieren. Geben Sie sie dann im Fenster Send a Message (Eine Nachricht senden) ein und wählen Sie Send Message (Nachricht senden) aus.

> **Note**
>
> In dieser Beispielmeldung wurde die Zeile `input:` mit Zeilenumbrüchen formatiert, um sie an die Seite anzupassen. Verwenden Sie die Schaltfläche „Copy (Kopieren)“ oder stellen Sie anderweitig sicher, dass sie als einzelne Zeile ohne Umbrüche eingegeben wird.

```
{
      "input":
 "QW5kIGxpa2UgdGhlIGJhc2VsZXNzIGZhYnJpYyBvZiB0aGlzIHZpc2lvbiwgVGhlIGNsb3VkLWNhcHBlZCB0b3dlc

  91cyBwYWxhY2VzLCBUaGUgc29sZW1uIHRlbXBsZXMsIHRoZSBncmVhdCBnbG9iZSBpdHNlbGbigJQgWWVhLCBhbGw

  ZXJpdOKAlHNoYWxsIGRpc3NvbHZlLCBBbmQgbGlrZSB0aGlzIGluc3Vic3RhbnRpYWwgcGFnZWFudCBmYWRlZCwgT

  FjayBiZWhpbmQuIFdlIGFyZSBzdWNoIHN0dWZmIEFzIGRyZWFtcyBhcmUgbWFkZSBvbiwgYW5kIG91ciBsaXR0bGU

  ZGVkIHdpdGggYSBzbGVlcC4gU2lyLCBJIGFtIHZleGVkLiBCZWFyIHdpdGggbXkgd2Vha25lc3MuIE15IG9sZCBic

  xlZC4gQmUgbm90IGRpc3R1cmJlZCB3aXRoIG15IGluZmlybWl0eS4gSWYgeW91IGJlIHBsZWFzZWQsIHJldGlyZSB

  QW5kIHRoZXJlIHJlcG9zZS4gQSB0dXJuIG9yIHR3byBJ4oCZbGwgd2FsayBUbyBzdGlsbCBteSBzdWF0aW5nIG1pb
}
```

5. Klicken Sie auf Schließen.
6. Öffnen Sie die Step-Functions-Konsole .
7. Rufen Sie Ihre Amazon- CloudWatch Logs-Protokollgruppe auf und überprüfen Sie die Protokolle. Der Name der Protokollgruppe sieht wie example-ExpressLogGroup-wJalrXUtnFEMI aus.

Beispiel für Lambda-Funktionscode

Im Folgenden finden Sie Lambda-Funktionscode, der zeigt, wie die initiierende Lambda-Funktion eine Zustandsautomaten-Ausführung mit dem AWS SDK startet.

```
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
    )
```

Code des Zustandsautomaten aus diesem Beispiel

Der Express-Workflow in diesem Beispielprojekt besteht aus einer Reihe von Lambda-Funktionen für die Textverarbeitung.

Weitere Informationen darüber, wie andere - AWS Services steuern AWS Step Functions kann, finden Sie unter Verwendung AWS Step Functions mit anderen Diensten.

```
{
  "Comment": "An example of using Express workflows to run text processing for each
 message sent from an SQS queue.",
  "StartAt": "Decode base64 string",
  "States": {
    "Decode base64 string": {
      "Type": "Task",
      "Resource": "arn:<PARTITION>:states:::lambda:invoke",
      "OutputPath": "$.Payload",
      "Parameters": {
        "FunctionName": "<BASE64_DECODER_LAMBDA_FUNCTION_NAME>",
        "Payload.$": "$"
      },
      "Next": "Generate statistics"
    },
    "Generate statistics": {
      "Type": "Task",
      "Resource": "arn:<PARTITION>:states:::lambda:invoke",
```

```
      "OutputPath": "$.Payload",
      "Parameters": {
        "FunctionName": "<TEXT_STATS_GENERATING_LAMBDA_FUNCTION_NAME>",
        "Payload.$": "$"
      },
      "Next": "Remove special characters"
    },
    "Remove special characters": {
      "Type": "Task",
      "Resource": "arn:<PARTITION>:states:::lambda:invoke",
      "OutputPath": "$.Payload",
      "Parameters": {
        "FunctionName": "<STRING_CLEANING_LAMBDA_FUNCTION_NAME>",
        "Payload.$": "$"
      },
      "Next": "Tokenize and count"
    },
    "Tokenize and count": {
      "Type": "Task",
      "Resource": "arn:<PARTITION>:states:::lambda:invoke",
      "OutputPath": "$.Payload",
      "Parameters": {
        "FunctionName": "<TOKENIZING_AND_WORD_COUNTING_LAMBDA_FUNCTION_NAME>",
        "Payload.$": "$"
      },
      "End": true
    }
  }
}
```

IAM-Beispiel

Diese vom Beispielprojekt generierte AWS Identity and Access Management (IAM)-Richtlinie enthält die geringsten Berechtigungen, die zum Ausführen des Zustandsautomaten und der zugehörigen Ressourcen erforderlich sind. Wir empfehlen Ihnen, nur die Berechtigungen einzubeziehen, die in Ihren IAM-Richtlinien erforderlich sind.

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

Die folgende Richtlinie stellt sicher, dass ausreichende Berechtigungen für CloudWatch Protokolle vorhanden sind.

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

Informationen zum Konfigurieren von IAM bei der Verwendung von Step Functions mit anderen - AWS Services finden Sie unter IAM-Richtlinien für integrierte Dienste.

Beispiel für selektives Checkpointing (Express-Workflows)

Dieses Beispielprojekt veranschaulicht, wie Standard- und Express-Workflows kombiniert werden, indem ein Pseudo-E-Commerce-Workflow ausgeführt wird, der selektives Checkpointing durchführt. Durch die Bereitstellung dieses Beispielprojekts werden eine Standard-Workflow-Zustandsmaschine, eine verschachtelte Express Workflow-Zustandsmaschine, eine AWS Lambda Funktion, eine Amazon Simple Queue Service (Amazon SQS) -Warteschlange und ein Amazon Simple Notification Service (Amazon SNS) -Thema erstellt.

Weitere Informationen zu Express-Workflows, verschachtelten Workflows und Step Functions Functions-Dienstintegrationen finden Sie im Folgenden:

- Standard- und Express-Workflows
- Starten von Workflow-Ausführungen über einen Aufgabenstatus
- Verwendung AWS Step Functions mit anderen Diensten

Schritt 1: Den State Machine erstellen und Ressourcen bereitstellen

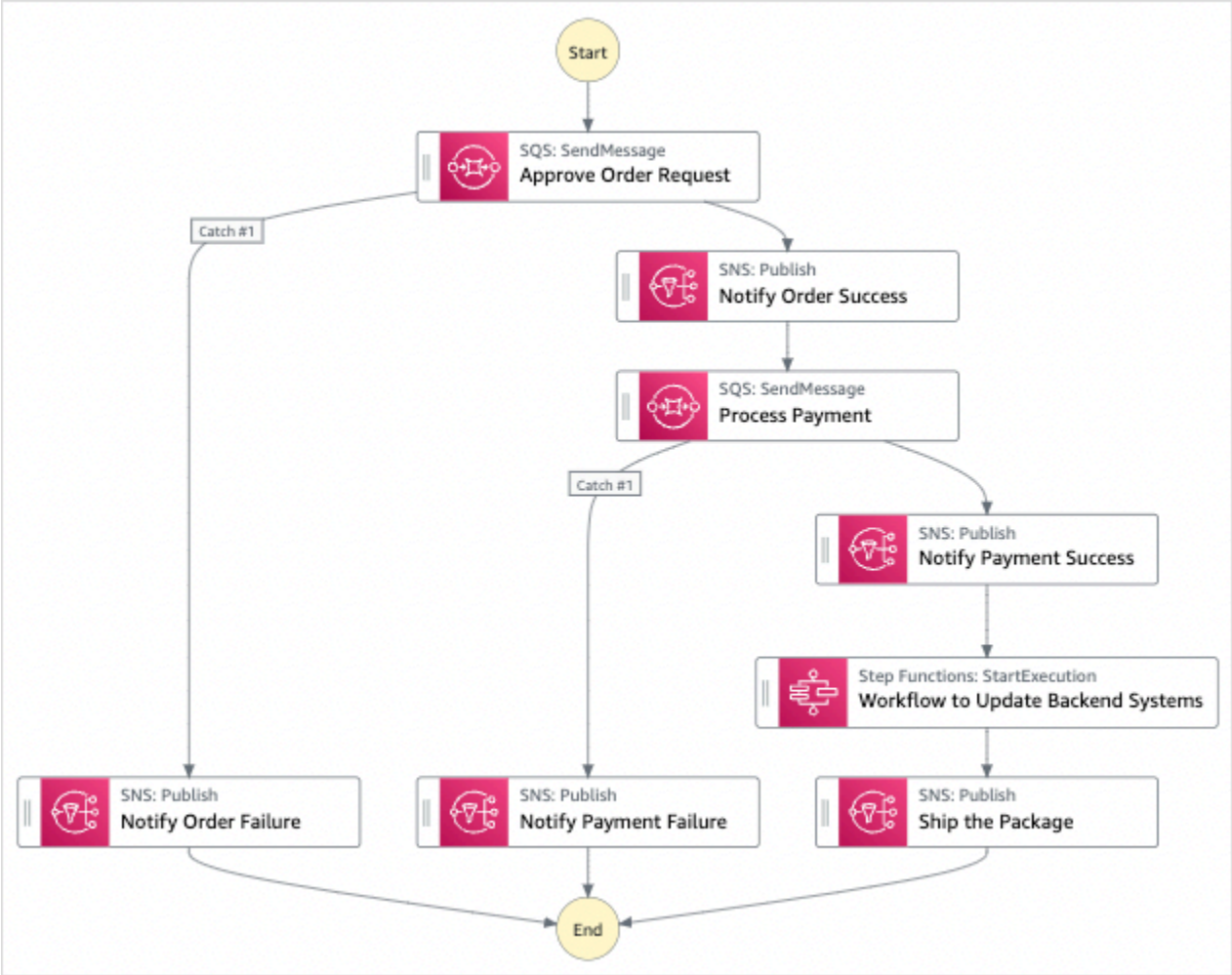
1. Öffnen Sie die Step Functions Functions-Konsole und wählen Sie Create State Machine.
2. Geben Sie **`Selective checkpointing example`** etwas in das Suchfeld ein und wählen Sie dann aus den zurückgegebenen Suchergebnissen die Option Beispiel für selektives Checkpointing aus.
3. Wählen Sie Next (Weiter), um fortzufahren.
4. Step Functions listet die Funktionen auf, die in dem von Ihnen ausgewählten Beispielprojekt AWS-Services verwendet wurden. Außerdem wird ein Workflow-Diagramm für das Beispielprojekt angezeigt. Stellen Sie dieses Projekt für Sie bereit AWS-Konto oder verwenden Sie es als Ausgangspunkt für die Erstellung Ihrer eigenen Projekte. Je nachdem, wie Sie vorgehen möchten, wählen Sie „Demo ausführen“ oder „Darauf aufbauen“.

   Dieses Beispielprojekt stellt die folgenden Ressourcen bereit:

   - Eine Funktion AWS Lambda
   - Eine Amazon-SQS-Warteschlange
   - Amazon-SNS-Thema
   - Eine AWS Step Functions Zustandsmaschine vom Typ Standard
   - Eine Step Functions Functions-Zustandsmaschine vom Typ Express

- Verwandte AWS Identity and Access Management Rollen (IAM)

Die folgende Abbildung zeigt das Workflow-Diagramm für das Beispielprojekt Selective Checkpointing:



5. Wählen Sie Vorlage verwenden, um mit Ihrer Auswahl fortzufahren.
6. Führen Sie eine der folgenden Aktionen aus:

- Wenn Sie „Darauf aufbauen“ ausgewählt haben, erstellt Step Functions den Workflow-Prototyp für das von Ihnen ausgewählte Beispielprojekt. Step Functions stellt die in der Workflow-Definition aufgeführten Ressourcen nicht bereit.

  Ziehen Sie in Workflow Studio Status per Drag-and-Drop aus demEntwurfsmodus, Bundesstaaten-Browser um mit der Erstellung Ihres Workflow-Prototyps fortzufahren. Oder wechseln Sie zu demCodemodus, der einen integrierten Code-Editor bietet, der VS Code ähnelt, um die Amazon States Language (ASL-) Definition Ihrer Zustandsmaschine in der

Step Functions Functions-Konsole zu aktualisieren. Weitere Informationen zur Verwendung von Workflow Studio zum Erstellen Ihrer Zustandsmaschinen finden Sie unterVerwenden von Workflow Studio.

> **Important**
>
> Denken Sie daran, den Platzhalter Amazon Resource Name (ARN) für die im Beispielprojekt verwendeten Ressourcen zu aktualisieren, bevor Sie Ihren Workflow ausführen.

- Wenn Sie Eine Demo ausführen ausgewählt haben, erstellt Step Functions ein schreibgeschütztes Beispielprojekt, das eine AWS CloudFormation Vorlage verwendet, um die in dieser Vorlage aufgeführten AWS Ressourcen für Ihr Projekt bereitzustellen. AWS-Konto

  > **Tip**
  >
  > Um die State-Machine-Definition des Beispielprojekts anzuzeigen, wählen Sie Code.

  Wenn Sie bereit sind, wählen Sie Deploy and run aus, um das Beispielprojekt bereitzustellen und die Ressourcen zu erstellen.

  Es kann bis zu 10 Minuten dauern, bis diese Ressourcen und die zugehörigen IAM-Berechtigungen erstellt sind. Während der Bereitstellung Ihrer Ressourcen können Sie den CloudFormation Stack-ID-Link öffnen, um zu sehen, welche Ressourcen bereitgestellt werden.

  Nachdem alle Ressourcen im Beispielprojekt erstellt wurden, wird das neue Beispielprojekt auf der Seite State Machines aufgeführt.

  > **Important**
  >
  > Für jeden in der CloudFormation Vorlage verwendeten Dienst können Standardgebühren anfallen.

Gehen Sie folgendermaßen vor, nachdem die Ressourcen des Beispielprojekts bereitgestellt wurden.

Schritt 2: Führen Sie die Zustandsmaschine aus

1. Wählen Sie auf der Seite State Machines Ihr Beispielprojekt aus.
2. Wählen Sie auf der Seite mit dem Beispielprojekt die Option Ausführung starten aus.
3. Gehen Sie im Dialogfeld Ausführung starten wie folgt vor:

    1. (Optional) Um Ihre Ausführung zu identifizieren, können Sie im Feld Name einen Namen dafür angeben. Standardmäßig generiert Step Functions automatisch einen eindeutigen Ausführungsnamen.

        > **Note**
        >
        > Mit Step Functions können Sie Namen für Zustandsmaschinen, Ausführungen und Aktivitäten sowie Labels erstellen, die Nicht-ASCII-Zeichen enthalten. Diese Nicht-ASCII-Namen funktionieren nicht mit Amazon. CloudWatch Um sicherzustellen, dass Sie CloudWatch Messwerte verfolgen können, wählen Sie einen Namen, der nur ASCII-Zeichen verwendet.

    2. (Optional) Geben Sie in das Eingabefeld Eingabewerte im JSON-Format ein, um Ihren Workflow auszuführen.

        Wenn Sie sich dafür entschieden haben, eine Demo auszuführen, müssen Sie keine Ausführungseingaben angeben.

        > **Note**
        >
        > Wenn das von Ihnen bereitgestellte Demo-Projekt vorab aufgefüllte Eingabedaten für die Ausführung enthält, verwenden Sie diese Eingabe, um die Zustandsmaschine auszuführen.

    3. Wählen Sie Start execution (Ausführung starten) aus.
    4. Die Step Functions Functions-Konsole leitet Sie zu einer Seite weiter, die mit Ihrer Ausführungs-ID betitelt ist. Diese Seite wird als Seite mit den Ausführungsdetails bezeichnet. Auf dieser Seite können Sie die Ausführungsergebnisse im Verlauf oder nach Abschluss der Ausführung überprüfen.

        Um die Ausführungsergebnisse zu überprüfen, wählen Sie in der Diagrammansicht einzelne Status und dann die einzelnen Registerkarten im Schrittdetails Bereich aus, um die Details der

einzelnen Status, einschließlich Eingabe, Ausgabe und Definition, anzuzeigen. Einzelheiten zu den Ausführungsinformationen, die Sie auf der Seite mit den Ausführungsdetails einsehen können, finden Sie unterSeite mit Ausführungsdetails – Schnittstellenübersicht.

4. Gehen Sie zu Ihrer Protokollgruppe CloudWatch Logs und überprüfen Sie die Logs. Der Name der Protokollgruppe wird wie Beispiel- ExpressLogGroup -wjalrxUTnFemi aussehen.

Zustandsautomaten-Beispielcode für den übergeordneten Zustandsautomaten (Standard-Workflows)

Die Zustandsmaschine in diesem Beispielprojekt ist in Amazon SQS-, Amazon SNS- und Step Functions Express-Workflows integriert.

Sehen Sie sich diese Beispiel-Zustandsmaschine an, um zu sehen, wie Step Functions Eingaben von Amazon SQS und Amazon SNS verarbeitet und dann eine verschachtelte Express Workflow-Zustandsmaschine verwendet, um Backend-Systeme zu aktualisieren.

Weitere Informationen darüber, wie Sie andere AWS Dienste steuern AWS Step Functions können, finden Sie unter. Verwendung AWS Step Functions mit anderen Diensten

```
{
    "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
        }
    }
```

```
}
```

Beispiel für eine IAM-Rolle für den Parent State Machine

Diese vom Beispielprojekt generierten Beispielrichtlinien AWS Identity and Access Management (IAM) beinhalten die geringsten Rechte, die für die Ausführung des Zustandsmaschinen und der zugehörigen Ressourcen erforderlich sind. Wir empfehlen, dass Sie nur die Berechtigungen in Ihre IAM-Richtlinien aufnehmen, die erforderlich sind.

Amazon SNS SNS-Richtlinie:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "sns:Publish"
            ],
            "Resource": "arn:aws:sns:us-east-1:123456789012:Checkpoint-SNSTopic-
wJalrXUtnFEMI",
            "Effect": "Allow"
        }
    ]
}
```

Amazon SQS SQS-Richtlinie:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "sqs:SendMessage"
            ],
            "Resource": "arn:aws:sqs:us-east-1:123456789012:Checkpoint-SQSQueue-
je7MtGbClwBF",
            "Effect": "Allow"
        }
    ]
}
```

Ausführungsrichtlinie für Zustände:

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

Zustandsautomaten-Beispielcode für den verschachtelten Zustandsautomaten (Express-Workflows)

Der Zustandsautomat in diesem Beispielprojekt aktualisiert Backend-Informationen, wenn er vom übergeordneten Zustandsautomaten aufgerufen wird.

Sehen Sie sich diese Beispiel-Zustandsmaschine an, um zu sehen, wie Step Functions die verschiedenen Komponenten der simulierten E-Commerce-Backend-Systeme aktualisiert.

Weitere Informationen darüber, wie AWS Step Functions Sie andere AWS Dienste steuern können, finden Sie unterVerwendung AWS Step Functions mit anderen Diensten.


```
{
  "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
    }
  }
}
```

Beispiel für eine IAM-Rolle für Child State Machine

Diese vom Beispielprojekt generierte Beispielrichtlinie AWS Identity and Access Management (IAM) beinhaltet die geringsten Rechte, die für die Ausführung der Zustandsmaschine und der zugehörigen Ressourcen erforderlich sind. Wir empfehlen, dass Sie nur die Berechtigungen in Ihre IAM-Richtlinien aufnehmen, die erforderlich sind.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": [
```

```
                "arn:aws:lambda:us-east-1:123456789012:function:Example-
UpdateDatabaseLambdaFunction-wJalrXUtnFEMI"
            ],
            "Effect": "Allow"
        }
    ]
}
```

Die folgende Richtlinie stellt sicher, dass genügend Berechtigungen für CloudWatch Logs vorhanden sind.

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

Informationen zur Konfiguration von IAM bei der Verwendung von Step Functions mit anderen AWS Diensten finden Sie unterIAM-Richtlinien für integrierte Dienste.

Ein AWS CodeBuild Projekt erstellen (CodeBuild, Amazon SNS)

Dieses Beispielprojekt zeigt, wie Sie AWS Step Functions ein AWS CodeBuild Projekt erstellen, Tests ausführen und anschließend eine Amazon SNS SNS-Benachrichtigung senden können.

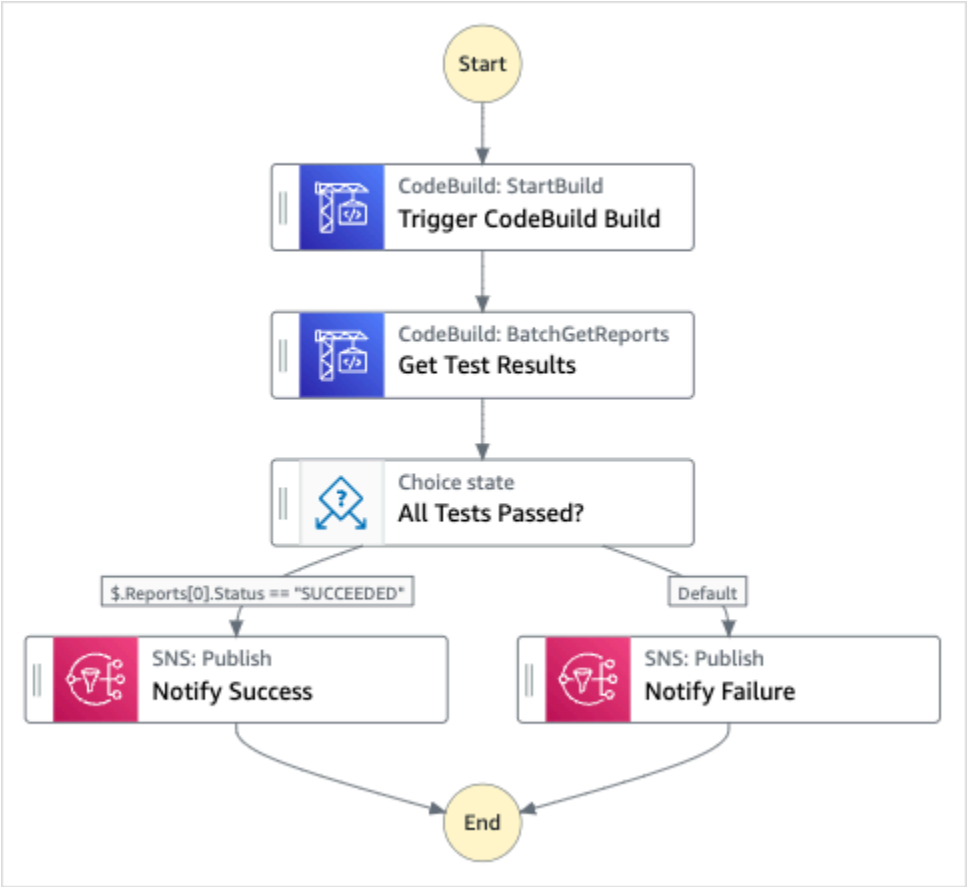
Schritt 1: Erstellen Sie die Zustandsmaschine und stellen Sie Ressourcen bereit

1. Öffnen Sie die Step Functions Functions-Konsole und wählen Sie Create State Machine.
2. Geben Sie **`Start a CodeBuild build`** etwas in das Suchfeld ein und wählen Sie dann aus den zurückgegebenen Suchergebnissen die Option CodeBuild Build starten aus.
3. Wählen Sie Next (Weiter), um fortzufahren.
4. Step Functions listet die Funktionen auf, die in dem von Ihnen ausgewählten Beispielprojekt AWS-Services verwendet wurden. Außerdem wird ein Workflow-Diagramm für das Beispielprojekt angezeigt. Stellen Sie dieses Projekt für Ihr Projekt bereit AWS-Konto oder verwenden Sie es als Ausgangspunkt für die Erstellung Ihrer eigenen Projekte. Je nachdem, wie Sie vorgehen möchten, wählen Sie „Demo ausführen“ oder „Darauf aufbauen“.

   In diesem Beispielprojekt werden die folgenden Ressourcen bereitgestellt:

   - Ein AWS CodeBuild Build
   - Amazon-SNS-Thema
   - Eine AWS Step Functions Staatsmaschine
   - Verwandte AWS Identity and Access Management Rollen (IAM)

   Die folgende Abbildung zeigt das Workflow-Diagramm für das Beispielprojekt Start a CodeBuild build:



5. Wählen Sie Vorlage verwenden, um mit Ihrer Auswahl fortzufahren.
6. Führen Sie eine der folgenden Aktionen aus:

   - Wenn Sie „Darauf aufbauen“ ausgewählt haben, erstellt Step Functions den Workflow-Prototyp für das von Ihnen ausgewählte Beispielprojekt. Step Functions stellt die in der Workflow-Definition aufgeführten Ressourcen nicht bereit.

     Ziehen Sie in Workflow Studio Status per Drag-and-Drop aus demEntwurfsmodus, Bundesstaaten-Browser um mit der Erstellung Ihres Workflow-Prototyps fortzufahren. Oder wechseln Sie zu demCodemodus, der einen integrierten Code-Editor bietet, der VS Code ähnelt, um die Amazon States Language (ASL-) Definition Ihrer Zustandsmaschine in der Step Functions Functions-Konsole zu aktualisieren. Weitere Informationen zur Verwendung von Workflow Studio zum Erstellen Ihrer Zustandsmaschinen finden Sie unterVerwenden von Workflow Studio.

> **Important**
>
> Denken Sie daran, den Platzhalter Amazon Resource Name (ARN) für die im Beispielprojekt verwendeten Ressourcen zu aktualisieren, bevor Sie Ihren Workflow ausführen.

- Wenn Sie Eine Demo ausführen ausgewählt haben, erstellt Step Functions ein schreibgeschütztes Beispielprojekt, das eine AWS CloudFormation Vorlage verwendet, um die in dieser Vorlage aufgeführten AWS Ressourcen für Ihr Projekt bereitzustellen. AWS-Konto

  > **Tip**
  >
  > Um die State-Machine-Definition des Beispielprojekts anzuzeigen, wählen Sie Code.

  Wenn Sie bereit sind, wählen Sie Deploy and run aus, um das Beispielprojekt bereitzustellen und die Ressourcen zu erstellen.

  Es kann bis zu 10 Minuten dauern, bis diese Ressourcen und die zugehörigen IAM-Berechtigungen erstellt sind. Während der Bereitstellung Ihrer Ressourcen können Sie den CloudFormation Stack-ID-Link öffnen, um zu sehen, welche Ressourcen bereitgestellt werden.

  Nachdem alle Ressourcen im Beispielprojekt erstellt wurden, wird das neue Beispielprojekt auf der Seite State Machines aufgeführt.

  > **Important**
  >
  > Für jeden in der CloudFormation Vorlage verwendeten Dienst können Standardgebühren anfallen.

Schritt 2: Führen Sie die Zustandsmaschine aus

1. Wählen Sie auf der Seite State Machines Ihr Beispielprojekt aus.
2. Wählen Sie auf der Seite mit dem Beispielprojekt die Option Ausführung starten aus.
3. Gehen Sie im Dialogfeld Ausführung starten wie folgt vor:

1. (Optional) Um Ihre Ausführung zu identifizieren, können Sie im Feld Name einen Namen dafür angeben. Standardmäßig generiert Step Functions automatisch einen eindeutigen Ausführungsnamen.

   > **Note**
   >
   > Mit Step Functions können Sie Namen für Zustandsmaschinen, Ausführungen und Aktivitäten sowie Beschriftungen erstellen, die Nicht-ASCII-Zeichen enthalten. Diese Nicht-ASCII-Namen funktionieren nicht mit Amazon. CloudWatch Um sicherzustellen, dass Sie CloudWatch Messwerte verfolgen können, wählen Sie einen Namen, der nur ASCII-Zeichen verwendet.

2. (Optional) Geben Sie in das Eingabefeld Eingabewerte im JSON-Format ein, um Ihren Workflow auszuführen.

   Wenn Sie sich dafür entschieden haben, eine Demo auszuführen, müssen Sie keine Ausführungseingaben angeben.

   > **Note**
   >
   > Wenn das von Ihnen bereitgestellte Demo-Projekt vorab aufgefüllte Eingabedaten für die Ausführung enthält, verwenden Sie diese Eingabe, um die Zustandsmaschine auszuführen.

3. Wählen Sie Start execution (Ausführung starten) aus.
4. Die Step Functions Functions-Konsole leitet Sie zu einer Seite weiter, die mit Ihrer Ausführungs-ID betitelt ist. Diese Seite wird als Seite mit den Ausführungsdetails bezeichnet. Auf dieser Seite können Sie die Ausführungsergebnisse im Verlauf der Ausführung oder nach deren Abschluss überprüfen.

   Um die Ausführungsergebnisse zu überprüfen, wählen Sie in der Diagrammansicht einzelne Status aus und wählen Sie dann die einzelnen Registerkarten im Schrittdetails Bereich, um die Details der einzelnen Status, einschließlich Eingabe, Ausgabe und Definition, anzuzeigen. Einzelheiten zu den Ausführungsinformationen, die Sie auf der Seite mit den Ausführungsdetails einsehen können, finden Sie unterSeite mit Ausführungsdetails – Schnittstellenübersicht.

Code des Zustandsautomaten aus diesem Beispiel

Die Zustandsmaschine in diesem Beispielprojekt lässt sich CodeBuild in Amazon SNS integrieren.

Sehen Sie sich diese Beispiel-Zustandsmaschine an, um zu sehen, wie Step Functions eine Zustandsmaschine verwendet, um ein CodeBuild Projekt zu erstellen, und anschließend ein Amazon SNS SNS-Thema mit einer Meldung darüber sendet, ob der Job erfolgreich war oder nicht.

Weitere Informationen darüber, wie Step Functions andere AWS Dienste steuern kann, finden Sie unterVerwendung AWS Step Functions mit anderen Diensten.

```
{
  "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
    }
  }
}
```

Informationen zur Konfiguration von IAM bei der Verwendung von Step Functions mit anderen AWS Diensten finden Sie unterIAM-Richtlinien für integrierte Dienste.

Daten vorverarbeiten und ein Modell für maschinelles Lernen trainieren

Dieses Beispielprojekt zeigt, wie Daten verwendet SageMaker und AWS Step Functions vorverarbeitet und ein Modell für maschinelles Lernen trainiert werden.

In diesem Projekt verwendet Step Functions eine Lambda-Funktion, um einen Amazon S3 S3-Bucket mit einem Testdatensatz und einem Python-Skript für die Datenverarbeitung zu versorgen. Anschließend trainiert es ein Modell für maschinelles Lernen und führt mithilfe der SageMaker Serviceintegration eine Batch-Transformation durch.

Weitere Informationen zu SageMaker Step Functions Functions-Dienstintegrationen finden Sie im Folgenden:

- Verwendung AWS Step Functions mit anderen Diensten

- SageMaker Mit Step Functions verwalten

> **Note**
>
> Für dieses Beispielprojekt können Gebühren anfallen.
> Für neue AWS Benutzer ist ein kostenloses Nutzungskontingent verfügbar. Im Rahmen dieses Kontingents sind die Services bis zu einem bestimmten Nutzungsumfang kostenlos. Weitere Informationen zu den AWS Kosten und dem kostenlosen Kontingent finden Sie unter SageMaker Preise.

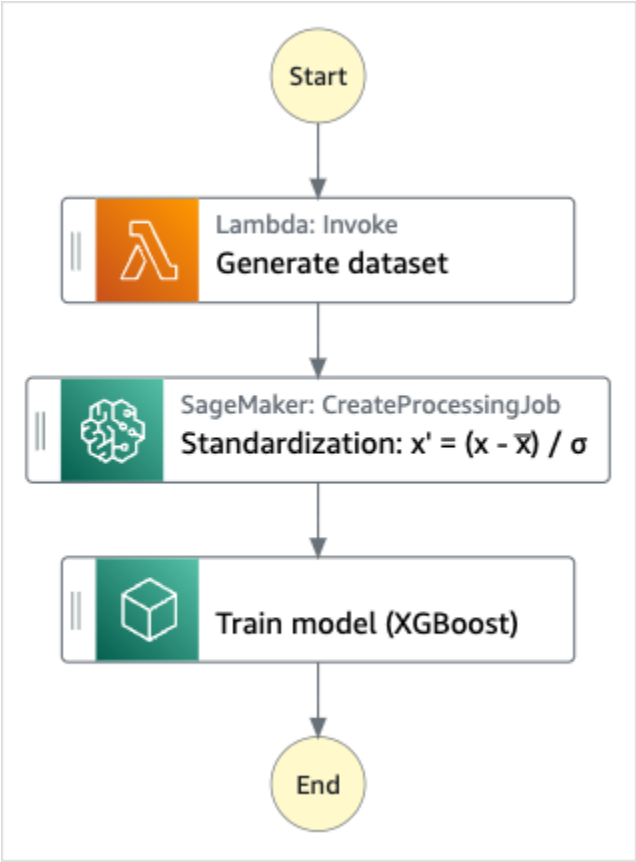
Schritt 1: Erstellen Sie den Zustandsmaschine und stellen Sie Ressourcen bereit

1. Öffnen Sie die Step Functions Functions-Konsole und wählen Sie Create State Machine.
2. Geben Sie **`Preprocess data and train a machine learning model`** etwas in das Suchfeld ein und wählen Sie dann Daten vorverarbeiten und anhand der zurückgegebenen Suchergebnisse ein Modell für maschinelles Lernen trainieren aus.
3. Wählen Sie Next (Weiter), um fortzufahren.
4. Step Functions listet die Funktionen auf, die in dem von Ihnen ausgewählten Beispielprojekt AWS-Services verwendet wurden. Außerdem wird ein Workflow-Diagramm für das Beispielprojekt angezeigt. Stellen Sie dieses Projekt für Ihr Projekt bereit AWS-Konto oder verwenden Sie es als Ausgangspunkt für die Erstellung Ihrer eigenen Projekte. Je nachdem, wie Sie vorgehen möchten, wählen Sie „Demo ausführen“ oder „Darauf aufbauen“.

   In diesem Beispielprojekt werden die folgenden Ressourcen bereitgestellt:

   - Eine Funktion AWS Lambda
   - Ein Amazon-S3-Bucket
   - Eine AWS Step Functions Staatsmaschine
   - Verwandte AWS Identity and Access Management Rollen (IAM)

   Die folgende Abbildung zeigt das Workflow-Diagramm für das Beispielprojekt „Daten vorverarbeiten und ein Modell für maschinelles Lernen trainieren“:



5. Wählen Sie Vorlage verwenden, um mit Ihrer Auswahl fortzufahren.
6. Führen Sie eine der folgenden Aktionen aus:

   - Wenn Sie „Darauf aufbauen“ ausgewählt haben, erstellt Step Functions den Workflow-Prototyp für das von Ihnen ausgewählte Beispielprojekt. Step Functions stellt die in der Workflow-Definition aufgeführten Ressourcen nicht bereit.

     Ziehen Sie in Workflow Studio Status per Drag-and-Drop aus demEntwurfsmodus, Bundesstaaten-Browser um mit der Erstellung Ihres Workflow-Prototyps fortzufahren. Oder wechseln Sie zu demCodemodus, der einen integrierten Code-Editor bietet, der VS Code ähnelt, um die Amazon States Language (ASL-) Definition Ihrer Zustandsmaschine in der Step Functions Functions-Konsole zu aktualisieren. Weitere Informationen zur Verwendung von Workflow Studio zum Erstellen Ihrer Zustandsmaschinen finden Sie unterVerwenden von Workflow Studio.

> **Important**
>
> Denken Sie daran, den Platzhalter Amazon Resource Name (ARN) für die im Beispielprojekt verwendeten Ressourcen zu aktualisieren, bevor Sie Ihren Workflow ausführen.

- Wenn Sie Eine Demo ausführen ausgewählt haben, erstellt Step Functions ein schreibgeschütztes Beispielprojekt, das eine AWS CloudFormation Vorlage verwendet, um die in dieser Vorlage aufgeführten AWS Ressourcen für Ihr Projekt bereitzustellen. AWS-Konto

  > **Tip**
  >
  > Um die State-Machine-Definition des Beispielprojekts anzuzeigen, wählen Sie Code.

  Wenn Sie bereit sind, wählen Sie Deploy and run aus, um das Beispielprojekt bereitzustellen und die Ressourcen zu erstellen.

  Es kann bis zu 10 Minuten dauern, bis diese Ressourcen und die zugehörigen IAM-Berechtigungen erstellt sind. Während der Bereitstellung Ihrer Ressourcen können Sie den CloudFormation Stack-ID-Link öffnen, um zu sehen, welche Ressourcen bereitgestellt werden.

  Nachdem alle Ressourcen im Beispielprojekt erstellt wurden, wird das neue Beispielprojekt auf der Seite State Machines aufgeführt.

  > **Important**
  >
  > Für jeden in der CloudFormation Vorlage verwendeten Dienst können Standardgebühren anfallen.

Schritt 2: Führen Sie die Zustandsmaschine aus

1. Wählen Sie auf der Seite State Machines Ihr Beispielprojekt aus.
2. Wählen Sie auf der Seite mit dem Beispielprojekt die Option Ausführung starten aus.
3. Gehen Sie im Dialogfeld Ausführung starten wie folgt vor:

1. (Optional) Um Ihre Ausführung zu identifizieren, können Sie im Feld Name einen Namen dafür angeben. Standardmäßig generiert Step Functions automatisch einen eindeutigen Ausführungsnamen.

   > **Note**
   >
   > Mit Step Functions können Sie Namen für Zustandsmaschinen, Ausführungen und Aktivitäten sowie Beschriftungen erstellen, die Nicht-ASCII-Zeichen enthalten. Diese Nicht-ASCII-Namen funktionieren nicht mit Amazon. CloudWatch Um sicherzustellen, dass Sie CloudWatch Messwerte verfolgen können, wählen Sie einen Namen, der nur ASCII-Zeichen verwendet.

2. (Optional) Geben Sie in das Eingabefeld Eingabewerte im JSON-Format ein, um Ihren Workflow auszuführen.

   Wenn Sie sich dafür entschieden haben, eine Demo auszuführen, müssen Sie keine Ausführungseingaben angeben.

   > **Note**
   >
   > Wenn das von Ihnen bereitgestellte Demo-Projekt vorab aufgefüllte Eingabedaten für die Ausführung enthält, verwenden Sie diese Eingabe, um die Zustandsmaschine auszuführen.

3. Wählen Sie Start execution (Ausführung starten) aus.
4. Die Step Functions Functions-Konsole leitet Sie zu einer Seite weiter, die mit Ihrer Ausführungs-ID betitelt ist. Diese Seite wird als Seite mit den Ausführungsdetails bezeichnet. Auf dieser Seite können Sie die Ausführungsergebnisse im Verlauf der Ausführung oder nach deren Abschluss überprüfen.

   Um die Ausführungsergebnisse zu überprüfen, wählen Sie in der Diagrammansicht einzelne Status aus und wählen Sie dann die einzelnen Registerkarten im Schrittdetails Bereich, um die Details der einzelnen Status, einschließlich Eingabe, Ausgabe und Definition, anzuzeigen. Einzelheiten zu den Ausführungsinformationen, die Sie auf der Seite mit den Ausführungsdetails einsehen können, finden Sie unterSeite mit Ausführungsdetails – Schnittstellenübersicht.

Code des Zustandsautomaten aus diesem Beispiel

Die Zustandsmaschine in diesem Beispielprojekt integriert sich mit SageMaker und AWS Lambda durch direkte Übergabe von Parametern an diese Ressourcen und verwendet einen Amazon S3 S3-Bucket für die Trainingsdatenquelle und -ausgabe.

Sehen Sie sich diese Beispiel-Zustandsmaschine an, um zu sehen, wie Step Functions Lambda und SageMaker steuert.

Weitere Informationen darüber, wie AWS Step Functions Sie andere AWS Dienste steuern können, finden Sie unterVerwendung AWS Step Functions mit anderen Diensten.

```
{
  "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
    }
  }
}
```

Informationen zur Konfiguration von IAM bei der Verwendung von Step Functions mit anderen AWS Diensten finden Sie unterIAM-Richtlinien für integrierte Dienste.

IAM-Beispiel

Diese vom Beispielprojekt generierten Beispielrichtlinien AWS Identity and Access Management (IAM) beinhalten die geringsten Rechte, die für die Ausführung der Zustandsmaschine und der zugehörigen Ressourcen erforderlich sind. Wir empfehlen, dass Sie nur die Berechtigungen in Ihre IAM-Richtlinien aufnehmen, die erforderlich sind.

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

Die folgende Richtlinie ermöglicht es der Lambda-Funktion, den Amazon S3 S3-Bucket mit Beispieldaten zu versorgen.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::featuretransform-
bucketforcodeanddata-1jn1le6gadwfz/*",
            "Effect": "Allow"
        }
    ]
}
```

Informationen zur Konfiguration von IAM bei der Verwendung von Step Functions mit anderen AWS Diensten finden Sie unterIAM-Richtlinien für integrierte Dienste.

Beispiel für eine Lambda-Orchestrierung

Dieses Beispielprojekt zeigt, wie AWS Lambda Funktionen in Step Functions Functions-Zustandsmaschinen integriert werden können.

In diesem Projekt verwendet Step Functions Lambda-Funktionen, um einen Aktienkurs zu überprüfen und eine Kauf- oder Verkaufsempfehlung zu ermitteln. Der Benutzer erhält dann diese Empfehlung und kann wählen, ob er die Aktie kaufen oder verkaufen möchte. Das Ergebnis des Handels wird unter Verwendung eines SNS-Themas zurückgegeben.

Weitere Informationen zu Step Functions Functions-Dienstintegrationen finden Sie im Folgenden:

- Verwendung AWS Step Functions mit anderen Diensten
- IAM-Richtlinien für:
  - IAM-Richtlinien für AWS Lambda
  - IAM-Richtlinien für Amazon SQS
  - IAM-Richtlinien für Amazon SNS

> **Note**
>
> Für dieses Beispielprojekt können Gebühren anfallen.

Für neue AWS Benutzer ist ein kostenloses Nutzungskontingent verfügbar. Im Rahmen dieses Kontingents sind die Services bis zu einem bestimmten Nutzungsumfang kostenlos. Weitere Informationen zu den AWS Kosten und dem kostenlosen Kontingent finden Sie unter Preise.

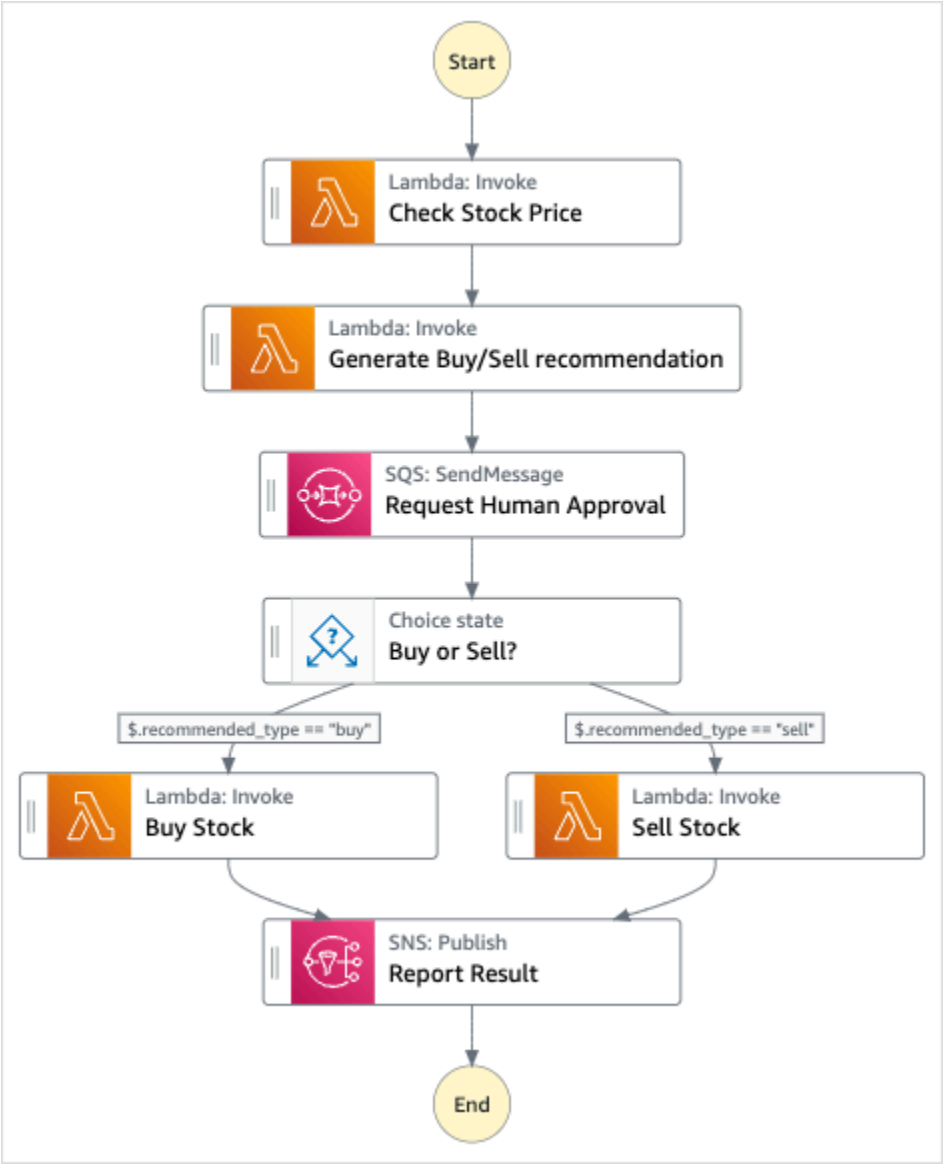
Schritt 1: Erstellen Sie den Zustandsmaschine und stellen Sie Ressourcen bereit

1. Öffnen Sie die Step Functions Functions-Konsole und wählen Sie Create State Machine.
2. Geben Sie **`Orchestrate Lambda functions`** etwas in das Suchfeld ein und wählen Sie dann aus den zurückgegebenen Suchergebnissen die Option LambdaFunktionen orchestrieren aus.
3. Wählen Sie Next (Weiter), um fortzufahren.
4. Step Functions listet die Funktionen auf, die in dem von Ihnen ausgewählten Beispielprojekt AWS-Services verwendet wurden. Außerdem wird ein Workflow-Diagramm für das Beispielprojekt angezeigt. Stellen Sie dieses Projekt für Sie bereit AWS-Konto oder verwenden Sie es als Ausgangspunkt für die Erstellung Ihrer eigenen Projekte. Je nachdem, wie Sie vorgehen möchten, wählen Sie „Demo ausführen“ oder „Darauf aufbauen“.

   Dieses Beispielprojekt stellt die folgenden Ressourcen bereit:

   - Fünf Funktionen Lambda
   - Eine Amazon Simple Queue Service-Warteschlange
   - Ein Amazon Simple Notification Service-Thema
   - Ein AWS Step Functions-Zustandsautomat
   - Zugehörige AWS Identity and Access Management (IAM)-Rollen

   Die folgende Abbildung zeigt das Workflow-Diagramm für das Beispielprojekt LambdaOrchestrate-Funktionen:



5. Wählen Sie Vorlage verwenden, um mit Ihrer Auswahl fortzufahren.
6. Führen Sie eine der folgenden Aktionen aus:

   - Wenn Sie „Darauf aufbauen“ ausgewählt haben, erstellt Step Functions den Workflow-Prototyp für das von Ihnen ausgewählte Beispielprojekt. Step Functions stellt die in der Workflow-Definition aufgeführten Ressourcen nicht bereit.

     Ziehen Sie in Workflow Studio Status per Drag-and-Drop aus demEntwurfsmodus, Bundesstaaten-Browser um mit der Erstellung Ihres Workflow-Prototyps fortzufahren. Oder wechseln Sie zu demCodemodus, der einen integrierten Code-Editor bietet, der VS Code ähnelt, um die Amazon States Language (ASL-) Definition Ihrer Zustandsmaschine in der Step Functions Functions-Konsole zu aktualisieren. Weitere Informationen zur Verwendung

von Workflow Studio zum Erstellen Ihrer Zustandsmaschinen finden Sie unterVerwenden von Workflow Studio.

> **Important**
>
> Denken Sie daran, den Platzhalter Amazon Resource Name (ARN) für die im Beispielprojekt verwendeten Ressourcen zu aktualisieren, bevor Sie Ihren Workflow ausführen.

- Wenn Sie Eine Demo ausführen ausgewählt haben, erstellt Step Functions ein schreibgeschütztes Beispielprojekt, das eine AWS CloudFormation Vorlage verwendet, um die in dieser Vorlage aufgeführten AWS Ressourcen für Ihr Projekt bereitzustellen. AWS-Konto

  > **Tip**
  >
  > Um die State-Machine-Definition des Beispielprojekts anzuzeigen, wählen Sie Code.

  Wenn Sie bereit sind, wählen Sie Deploy and run aus, um das Beispielprojekt bereitzustellen und die Ressourcen zu erstellen.

  Es kann bis zu 10 Minuten dauern, bis diese Ressourcen und die zugehörigen IAM-Berechtigungen erstellt sind. Während der Bereitstellung Ihrer Ressourcen können Sie den CloudFormation Stack-ID-Link öffnen, um zu sehen, welche Ressourcen bereitgestellt werden.

  Nachdem alle Ressourcen im Beispielprojekt erstellt wurden, wird das neue Beispielprojekt auf der Seite State Machines aufgeführt.

  > **Important**
  >
  > Für jeden in der CloudFormation Vorlage verwendeten Dienst können Standardgebühren anfallen.

Schritt 2: Führen Sie die Zustandsmaschine aus

Nachdem alle Ressourcen bereitgestellt und bereitgestellt wurden, wird das Dialogfeld Ausführung starten angezeigt.

1. Wählen Sie auf der Seite State Machines Ihr Beispielprojekt aus.
2. Wählen Sie auf der Seite mit dem Beispielprojekt die Option Ausführung starten aus.
3. Gehen Sie im Dialogfeld Ausführung starten wie folgt vor:

   1. (Optional) Um Ihre Ausführung zu identifizieren, können Sie im Feld Name einen Namen dafür angeben. Standardmäßig generiert Step Functions automatisch einen eindeutigen Ausführungsnamen.

      > **Note**
      >
      > Mit Step Functions können Sie Namen für Zustandsmaschinen, Ausführungen und Aktivitäten sowie Beschriftungen erstellen, die Nicht-ASCII-Zeichen enthalten. Diese Nicht-ASCII-Namen funktionieren nicht mit Amazon. CloudWatch Um sicherzustellen, dass Sie CloudWatch Messwerte verfolgen können, wählen Sie einen Namen, der nur ASCII-Zeichen verwendet.

   2. (Optional) Geben Sie in das Eingabefeld Eingabewerte im JSON-Format ein, um Ihren Workflow auszuführen.

      Wenn Sie sich dafür entschieden haben, eine Demo auszuführen, müssen Sie keine Ausführungseingaben angeben.

      > **Note**
      >
      > Wenn das von Ihnen bereitgestellte Demo-Projekt vorab aufgefüllte Eingabedaten für die Ausführung enthält, verwenden Sie diese Eingabe, um die Zustandsmaschine auszuführen.

   3. Wählen Sie Start execution (Ausführung starten) aus.
   4. Die Step Functions Functions-Konsole leitet Sie zu einer Seite weiter, die mit Ihrer Ausführungs-ID betitelt ist. Diese Seite wird als Seite mit den Ausführungsdetails bezeichnet. Auf dieser Seite können Sie die Ausführungsergebnisse im Verlauf oder nach Abschluss der Ausführung überprüfen.

      Um die Ausführungsergebnisse zu überprüfen, wählen Sie in der Diagrammansicht einzelne Status aus und wählen Sie dann die einzelnen Registerkarten im Schrittdetails Bereich, um die Details der einzelnen Status, einschließlich Eingabe, Ausgabe und Definition, anzuzeigen. Einzelheiten zu den Ausführungsinformationen, die Sie auf der Seite mit

den Ausführungsdetails einsehen können, finden Sie unterSeite mit Ausführungsdetails – Schnittstellenübersicht.

Informationen zur Zustandsmaschine und ihrer Ausführung

Die Zustandsmaschine in diesem Beispielprojekt integriert sich in, AWS Lambda indem sie Parameter direkt an diese Ressourcen übergibt, verwendet eine Amazon SQS SQS-Warteschlange, um die Anfrage zur Genehmigung durch einen Mitarbeiter zu verwalten, und verwendet ein Amazon SNS SNS-Thema, um die Ergebnisse der Abfrage zurückzugeben.

Eine Step Functions Functions-Ausführung empfängt einen JSON-Text als Eingabe und übergibt diese Eingabe an den ersten Status im Workflow. Einzelne Staaten erhalten JSON-Daten als Eingabe und geben JSON-Daten normalerweise als Ausgabe an den nächsten Status weiter. In diesem Beispielprojekt wird die Ausgabe jedes Schritts als Eingabe an den nächsten Schritt im Workflow übergeben. Beispielsweise erhält der Schritt „Kauf-/Verkaufsempfehlung generieren“ die Ausgabe des Schritts „Aktienkurs prüfen“ als Eingabe. Außerdem wird die Ausgabe des Schritts „Kauf-/Verkaufsempfehlung generieren“ als Eingabe an den nächsten Schritt, „Genehmigung durch einen Mitarbeiter anfordern“, übergeben, der einem menschlichen Genehmigungsschritt nachempfunden ist.

> **Note**
>
> Um die von einem Schritt zurückgegebene Ausgabe und die an einen Schritt weitergeleiteten Eingaben anzuzeigen, öffnen Sie die Seite mit den Ausführungsdetails für Ihre Workflow-Ausführung. Sehen Sie sich im Abschnitt Schrittdetails die Eingabe und Ausgabe für jeden Schritt an, den Sie im Anzeigemodus auswählen.

Um einen menschlichen Genehmigungsschritt zu implementieren, unterbrechen Sie in der Regel die Workflow-Ausführung, bis ein Task-Token zurückgegeben wird. In diesem Beispielprojekt wird eine Nachricht an eine Amazon SQS SQS-Warteschlange übergeben, die als Auslöser für die Lambda-Funktion verwendet wird, die für die Verarbeitung der Callback-Funktionalität definiert ist. Die Nachricht enthält ein Task-Token und die vom vorherigen Schritt zurückgegebene Ausgabe. Die Lambda-Funktion wird mit der Nutzlast der Nachricht aufgerufen. Die Workflow-Ausführung wird angehalten, bis sie das Task-Token mit einem API-Aufruf zurückerhält. `SendTaskSuccess` Weitere Informationen zu Task-Token finden Sie unter. Warten auf einen Callback mit dem Aufgabentoken

Der folgende Code für die `StepFunctionsSample-HelloLambda-ApproveSqsLambda` Funktion zeigt, wie sie so definiert ist, dass alle Aufgaben, die von der Amazon SQS-Warteschlange über die Step Functions Functions-Zustandsmaschine eingereicht wurden, automatisch genehmigt werden.

Beispiel für einen Lambda-Funktionscode zur Handhabung der Callback-Funktionalität und zur Rückgabe des Task-Tokens

```
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);
      });
  }
};
```

Sehen Sie sich diese Beispiel-Zustandsmaschine an, um zu sehen, wie Step Functions Lambda und Amazon SQS steuert.

Weitere Informationen darüber, wie Sie andere AWS Dienste steuern AWS Step Functions können, finden Sie unter. Verwendung AWS Step Functions mit anderen Diensten

```
{
    "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
        }
    }
}
```

Informationen zur Konfiguration von IAM bei der Verwendung von Step Functions mit anderen AWS Diensten finden Sie unterIAM-Richtlinien für integrierte Dienste.

IAM-Beispiele

Diese vom Beispielprojekt generierten Beispielrichtlinien AWS Identity and Access Management (IAM) beinhalten die geringsten Rechte, die für die Ausführung der Zustandsmaschine und der zugehörigen Ressourcen erforderlich sind. Wir empfehlen, dass Sie nur die Berechtigungen in Ihre IAM-Richtlinien aufnehmen, die erforderlich sind.

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

Informationen zur Konfiguration von IAM bei der Verwendung von Step Functions mit anderen AWS Diensten finden Sie unterIAM-Richtlinien für integrierte Dienste.

Eine Athena-Abfrage starten

Dieses Beispielprojekt, das auf Standard-Workflows basiert, zeigt, wie Sie Step Functions und Amazon Athena verwenden, um eine Athena-Abfrage zu starten und eine Benachrichtigung mit Abfrageergebnissen zu senden.

In diesem Projekt verwendet Step Functions Lambda-Funktionen und einen AWS Glue Crawler, um eine Reihe von Beispieldaten zu generieren. Anschließend führt es eine Abfrage mithilfe der Athena-Serviceintegration durch und gibt die Ergebnisse unter Verwendung eines SNS-Themas zurück.

Weitere Informationen zu den Service-Integrationen von Athena und Step Functions finden Sie im Folgenden:

- Verwendung AWS Step Functions mit anderen Diensten
- Rufen Sie Athena mit Step Functions auf

> **Note**
>
> Für dieses Beispielprojekt können Gebühren anfallen.
> Für neue AWS Benutzer ist ein kostenloses Nutzungskontingent verfügbar. Im Rahmen dieses Kontingents sind die Services bis zu einem bestimmten Nutzungsumfang kostenlos. Weitere Informationen zu den AWS Kosten und dem kostenlosen Kontingent finden Sie unter Athena-Preise.

Schritt 1: Erstellen Sie die Zustandsmaschine und stellen Sie Ressourcen bereit

1. Öffnen Sie die Step Functions Functions-Konsole und wählen Sie Create State Machine.
2. Geben Sie **`Start an Athena query`** etwas in das Suchfeld ein, und wählen Sie dann aus den zurückgegebenen Suchergebnissen die Option Athena Abfrage starten aus.
3. Wählen Sie Next (Weiter), um fortzufahren.
4. Step Functions listet die Funktionen auf, die in dem von Ihnen ausgewählten Beispielprojekt AWS-Services verwendet wurden. Außerdem wird ein Workflow-Diagramm für das

Beispielprojekt angezeigt. Stellen Sie dieses Projekt für Ihr Projekt bereit AWS-Konto oder verwenden Sie es als Ausgangspunkt für die Erstellung Ihrer eigenen Projekte. Je nachdem, wie Sie vorgehen möchten, wählen Sie „Demo ausführen“ oder „Darauf aufbauen“.

In diesem Beispielprojekt werden die folgenden Ressourcen bereitgestellt:

- Eine Abfrage Amazon Athena
- Ein AWS-Glue-Crawler
- Ein Amazon SNS-Thema
- Ein AWS Step Functions-Zustandsautomat
- Zugehörige AWS Identity and Access Management (IAM)-Rollen

Die folgende Abbildung zeigt das Workflow-Diagramm für das Beispielprojekt Eine Athena Abfrage starten:

5. Wählen Sie Vorlage verwenden, um mit Ihrer Auswahl fortzufahren.
6. Führen Sie eine der folgenden Aktionen aus:

   - Wenn Sie „Darauf aufbauen“ ausgewählt haben, erstellt Step Functions den Workflow-Prototyp für das von Ihnen ausgewählte Beispielprojekt. Step Functions stellt die in der Workflow-Definition aufgeführten Ressourcen nicht bereit.

     Ziehen Sie in Workflow Studio Status per Drag-and-Drop aus demEntwurfsmodus, Bundesstaaten-Browser um mit der Erstellung Ihres Workflow-Prototyps fortzufahren. Oder wechseln Sie zu demCodemodus, der einen integrierten Code-Editor bietet, der VS Code ähnelt, um die Amazon States Language (ASL-) Definition Ihrer Zustandsmaschine in der Step Functions Functions-Konsole zu aktualisieren. Weitere Informationen zur Verwendung von Workflow Studio zum Erstellen Ihrer Zustandsmaschinen finden Sie unterVerwenden von Workflow Studio.

     > **Important**
     >
     > Denken Sie daran, den Platzhalter Amazon Resource Name (ARN) für die im Beispielprojekt verwendeten Ressourcen zu aktualisieren, bevor Sie Ihren Workflow ausführen.

   - Wenn Sie Eine Demo ausführen ausgewählt haben, erstellt Step Functions ein schreibgeschütztes Beispielprojekt, das eine AWS CloudFormation Vorlage verwendet, um die in dieser Vorlage aufgeführten AWS Ressourcen für Ihr Projekt bereitzustellen. AWS-Konto

     > **Tip**
     >
     > Um die State-Machine-Definition des Beispielprojekts anzuzeigen, wählen Sie Code.

     Wenn Sie bereit sind, wählen Sie Deploy and run aus, um das Beispielprojekt bereitzustellen und die Ressourcen zu erstellen.

     Es kann bis zu 10 Minuten dauern, bis diese Ressourcen und die zugehörigen IAM-Berechtigungen erstellt sind. Während der Bereitstellung Ihrer Ressourcen können Sie den CloudFormation Stack-ID-Link öffnen, um zu sehen, welche Ressourcen bereitgestellt werden.

Nachdem alle Ressourcen im Beispielprojekt erstellt wurden, wird das neue Beispielprojekt auf der Seite State Machines aufgeführt.

> **Important**
>
> Für jeden in der CloudFormation Vorlage verwendeten Dienst können Standardgebühren anfallen.

Schritt 2: Führen Sie die Zustandsmaschine aus

1. Wählen Sie auf der Seite State Machines Ihr Beispielprojekt aus.
2. Wählen Sie auf der Seite mit dem Beispielprojekt die Option Ausführung starten aus.
3. Gehen Sie im Dialogfeld Ausführung starten wie folgt vor:

   1. (Optional) Um Ihre Ausführung zu identifizieren, können Sie im Feld Name einen Namen dafür angeben. Standardmäßig generiert Step Functions automatisch einen eindeutigen Ausführungsnamen.

      > **Note**
      >
      > Mit Step Functions können Sie Namen für Zustandsmaschinen, Ausführungen und Aktivitäten sowie Beschriftungen erstellen, die Nicht-ASCII-Zeichen enthalten. Diese Nicht-ASCII-Namen funktionieren nicht mit Amazon. CloudWatch Um sicherzustellen, dass Sie CloudWatch Messwerte verfolgen können, wählen Sie einen Namen, der nur ASCII-Zeichen verwendet.

   2. (Optional) Geben Sie in das Eingabefeld Eingabewerte im JSON-Format ein, um Ihren Workflow auszuführen.

      Wenn Sie sich dafür entschieden haben, eine Demo auszuführen, müssen Sie keine Ausführungseingaben angeben.

> **Note**
>
> Wenn das von Ihnen bereitgestellte Demo-Projekt vorab aufgefüllte Eingabedaten für die Ausführung enthält, verwenden Sie diese Eingabe, um die Zustandsmaschine auszuführen.

3. Wählen Sie Start execution (Ausführung starten) aus.
4. Die Step Functions Functions-Konsole leitet Sie zu einer Seite weiter, die mit Ihrer Ausführungs-ID betitelt ist. Diese Seite wird als Seite mit den Ausführungsdetails bezeichnet. Auf dieser Seite können Sie die Ausführungsergebnisse im Verlauf oder nach Abschluss der Ausführung überprüfen.

   Um die Ausführungsergebnisse zu überprüfen, wählen Sie in der Diagrammansicht einzelne Status aus und wählen Sie dann die einzelnen Registerkarten im Schrittdetails Bereich, um die Details der einzelnen Status, einschließlich Eingabe, Ausgabe und Definition, anzuzeigen. Einzelheiten zu den Ausführungsinformationen, die Sie auf der Seite mit den Ausführungsdetails einsehen können, finden Sie unterSeite mit Ausführungsdetails – Schnittstellenübersicht.

Code des Zustandsautomaten aus diesem Beispiel

Die Zustandsmaschine in diesem Beispielprojekt lässt sich in Athena integrieren, AWS Lambda indem Parameter direkt an diese Ressourcen übergeben werden, und verwendet ein SNS-Thema, um die Ergebnisse der Abfrage zurückzugeben.

Sehen Sie sich diese Beispiel-Zustandsmaschine an, um zu sehen, wie Step Functions Lambda und Athena steuert.

Weitere Informationen darüber, wie Sie andere AWS Dienste steuern AWS Step Functions können, finden Sie unter. Verwendung AWS Step Functions mit anderen Diensten

```
{
  "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
    }
  }
}
```

Informationen zur Konfiguration von IAM bei der Verwendung von Step Functions mit anderen AWS Diensten finden Sie unterIAM-Richtlinien für integrierte Dienste.

IAM-Beispiel

Diese vom Beispielprojekt generierten Beispielrichtlinien AWS Identity and Access Management (IAM) beinhalten die geringsten Rechte, die für die Ausführung der Zustandsmaschine und der zugehörigen Ressourcen erforderlich sind. Wir empfehlen, dass Sie nur die Berechtigungen in Ihre IAM-Richtlinien aufnehmen, die erforderlich sind.

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

Informationen zur Konfiguration von IAM bei der Verwendung von Step Functions mit anderen AWS Diensten finden Sie unterIAM-Richtlinien für integrierte Dienste.

Führen Sie mehrere Abfragen aus (Amazon Athena, Amazon SNS)

Dieses Beispielprojekt zeigt, wie Athena-Abfragen nacheinander und dann parallel ausgeführt, Fehler behandelt und dann eine Amazon SNS SNS-Benachrichtigung gesendet werden, je nachdem, ob die Abfragen erfolgreich waren oder nicht.

In diesem Projekt verwendet Step Functions eine Zustandsmaschine, um Athena-Abfragen synchron auszuführen. Nachdem die Abfrageergebnisse zurückgegeben wurden, wechseln Sie in den parallelen Status, in dem zwei Athena-Abfragen parallel ausgeführt werden. Es wartet dann darauf, dass der Job erfolgreich ist oder nicht, und sendet ein Amazon SNS SNS-Thema mit einer Nachricht darüber, ob der Job erfolgreich war oder nicht.

Schritt 1: Erstellen Sie die Zustandsmaschine und stellen Sie Ressourcen bereit

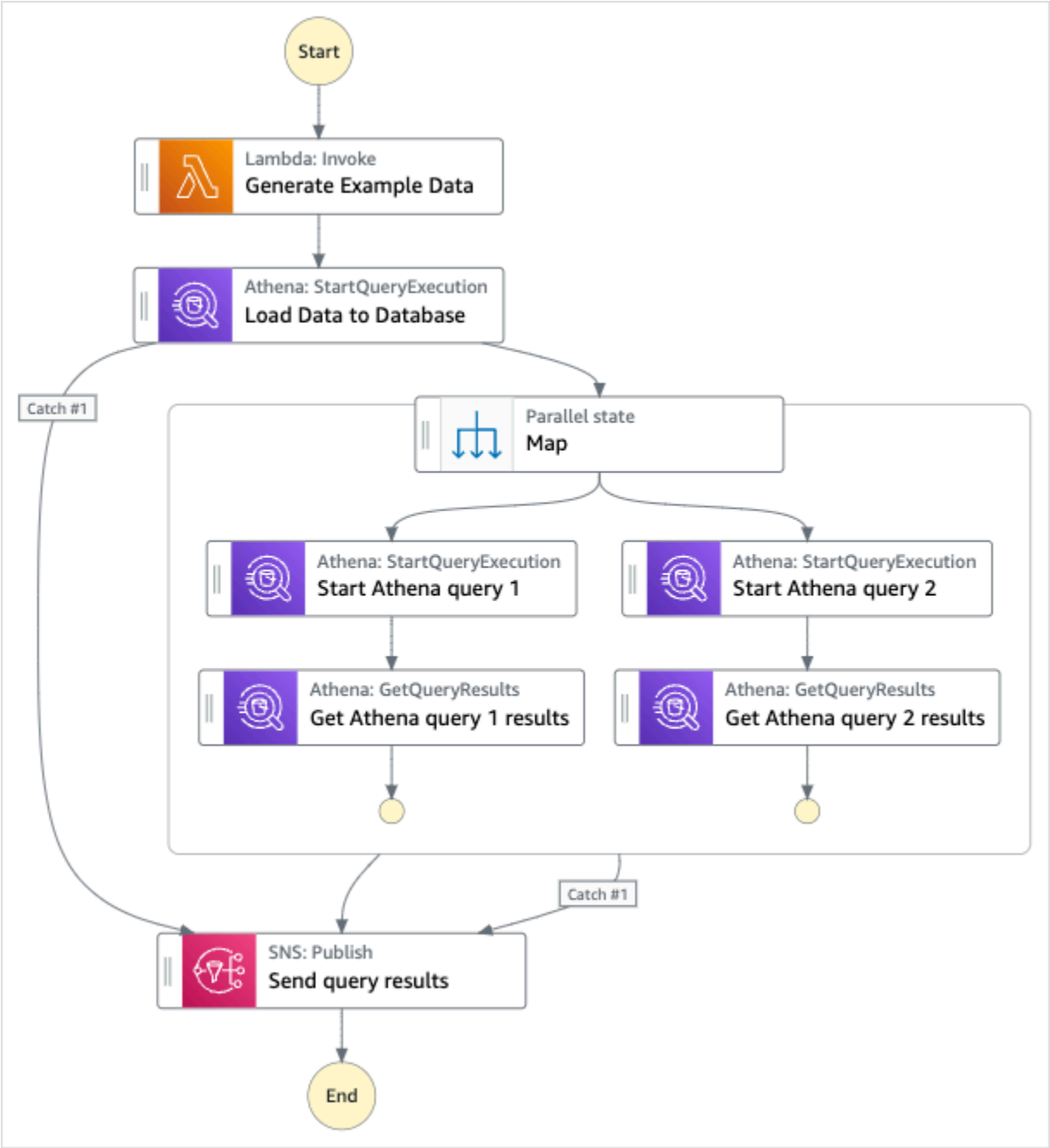
1. Öffnen Sie die Step Functions Functions-Konsole und wählen Sie Create State Machine.
2. Geben Sie **`Execute multiple queries`** etwas in das Suchfeld ein und wählen Sie dann aus den zurückgegebenen Suchergebnissen die Option Mehrere Abfragen ausführen aus.
3. Wählen Sie Next (Weiter), um fortzufahren.
4. Step Functions listet die Funktionen auf, die in dem von Ihnen ausgewählten Beispielprojekt AWS-Services verwendet wurden. Außerdem wird ein Workflow-Diagramm für das Beispielprojekt angezeigt. Stellen Sie dieses Projekt für Ihr Projekt bereit AWS-Konto oder verwenden Sie es als Ausgangspunkt für die Erstellung Ihrer eigenen Projekte. Je nachdem, wie Sie vorgehen möchten, wählen Sie „Demo ausführen“ oder „Darauf aufbauen“.

   In diesem Beispielprojekt werden die folgenden Ressourcen bereitgestellt:

   - Amazon AthenaAbfragen
   - Ein Amazon SNS-Thema

- Ein AWS Step Functions-Zustandsautomat
- Zugehörige AWS Identity and Access Management (IAM)-Rollen

Die folgende Abbildung zeigt das Workflow-Diagramm für das Beispielprojekt Mehrere Abfragen ausführen:



5. Wählen Sie Vorlage verwenden, um mit Ihrer Auswahl fortzufahren.
6. Führen Sie eine der folgenden Aktionen aus:

- Wenn Sie „Darauf aufbauen“ ausgewählt haben, erstellt Step Functions den Workflow-Prototyp für das von Ihnen ausgewählte Beispielprojekt. Step Functions stellt die in der Workflow-Definition aufgeführten Ressourcen nicht bereit.

  Ziehen Sie in Workflow Studio Status per Drag-and-Drop aus demEntwurfsmodus, Bundesstaaten-Browser um mit der Erstellung Ihres Workflow-Prototyps fortzufahren. Oder wechseln Sie zu demCodemodus, der einen integrierten Code-Editor bietet, der VS Code ähnelt, um die Amazon States Language (ASL-) Definition Ihrer Zustandsmaschine in der Step Functions Functions-Konsole zu aktualisieren. Weitere Informationen zur Verwendung von Workflow Studio zum Erstellen Ihrer Zustandsmaschinen finden Sie unterVerwenden von Workflow Studio.

  > **Important**
  >
  > Denken Sie daran, den Platzhalter Amazon Resource Name (ARN) für die im Beispielprojekt verwendeten Ressourcen zu aktualisieren, bevor Sie Ihren Workflow ausführen.

- Wenn Sie Eine Demo ausführen ausgewählt haben, erstellt Step Functions ein schreibgeschütztes Beispielprojekt, das eine AWS CloudFormation Vorlage verwendet, um die in dieser Vorlage aufgeführten AWS Ressourcen für Ihr Projekt bereitzustellen. AWS-Konto

  > **Tip**
  >
  > Um die State-Machine-Definition des Beispielprojekts anzuzeigen, wählen Sie Code.

  Wenn Sie bereit sind, wählen Sie Deploy and run aus, um das Beispielprojekt bereitzustellen und die Ressourcen zu erstellen.

  Es kann bis zu 10 Minuten dauern, bis diese Ressourcen und die zugehörigen IAM-Berechtigungen erstellt sind. Während der Bereitstellung Ihrer Ressourcen können Sie den CloudFormation Stack-ID-Link öffnen, um zu sehen, welche Ressourcen bereitgestellt werden.

  Nachdem alle Ressourcen im Beispielprojekt erstellt wurden, wird das neue Beispielprojekt auf der Seite State Machines aufgeführt.

> **Important**
>
> Für jeden in der CloudFormation Vorlage verwendeten Dienst können Standardgebühren anfallen.

Schritt 2: Führen Sie die Zustandsmaschine aus

1. Wählen Sie auf der Seite State Machines Ihr Beispielprojekt aus.
2. Wählen Sie auf der Seite mit dem Beispielprojekt die Option Ausführung starten aus.
3. Gehen Sie im Dialogfeld Ausführung starten wie folgt vor:

    1. (Optional) Um Ihre Ausführung zu identifizieren, können Sie im Feld Name einen Namen dafür angeben. Standardmäßig generiert Step Functions automatisch einen eindeutigen Ausführungsnamen.

        > **Note**
        >
        > Mit Step Functions können Sie Namen für Zustandsmaschinen, Ausführungen, Aktivitäten und Labels erstellen, die Nicht-ASCII-Zeichen enthalten. Diese Nicht-ASCII-Namen funktionieren nicht mit Amazon. CloudWatch Um sicherzustellen, dass Sie CloudWatch Messwerte verfolgen können, wählen Sie einen Namen, der nur ASCII-Zeichen verwendet.

    2. (Optional) Geben Sie in das Eingabefeld Eingabewerte im JSON-Format ein, um Ihren Workflow auszuführen.

        Wenn Sie sich dafür entschieden haben, eine Demo auszuführen, müssen Sie keine Ausführungseingaben angeben.

    3. Wählen Sie Start execution (Ausführung starten) aus.
    4. Die Step Functions Functions-Konsole leitet Sie zu einer Seite weiter, die mit Ihrer Ausführungs-ID betitelt ist. Diese Seite wird als Seite mit den Ausführungsdetails bezeichnet. Auf dieser Seite können Sie die Ausführungsergebnisse im Verlauf der Ausführung oder nach deren Abschluss überprüfen.

Um die Ausführungsergebnisse zu überprüfen, wählen Sie in der Diagrammansicht einzelne Status aus und wählen Sie dann die einzelnen Registerkarten im Schrittdetails Bereich, um die Details der einzelnen Status, einschließlich Eingabe, Ausgabe und Definition, anzuzeigen. Einzelheiten zu den Ausführungsinformationen, die Sie auf der Seite mit den Ausführungsdetails einsehen können, finden Sie unterSeite mit Ausführungsdetails – Schnittstellenübersicht.

Code des Zustandsautomaten aus diesem Beispiel

Die Zustandsmaschine in diesem Beispielprojekt lässt sich in Amazon Athena und Amazon SNS integrieren, indem Parameter direkt an diese Ressourcen übergeben werden.

Sehen Sie sich diese Beispiel-Zustandsmaschine an, um zu sehen, wie Step Functions Amazon Athena und Amazon SNS steuert, indem es eine Verbindung zum Amazon-Ressourcennamen (ARN) im `Resource` Feld herstellt und `Parameters` an die Service-API weiterleitet.

Weitere Informationen darüber, wie Sie andere AWS Dienste steuern AWS Step Functions können, finden Sie unter. Verwendung AWS Step Functions mit anderen Diensten

```
{
    "Comment": "An example of using Athena to execute queries in sequence and parallel,
 with error handling and notifications.",
    "StartAt": "Generate Example Data",
    "States": {
      "Generate Example Data": {
        "Type": "Task",
        "Resource": "arn:aws:states:::lambda:invoke",
        "OutputPath": "$.Payload",
        "Parameters": {
          "FunctionName": "<ATHENA_FUNCTION_NAME>"
        },
        "Next": "Load Data to Database"
      },
      "Load Data to Database": {
        "Type": "Task",
        "Resource": "arn:aws:states:::athena:startQueryExecution.sync",
        "Parameters": {
          "QueryString": "<ATHENA_QUERYSTRING>",
          "WorkGroup": "<ATHENA_WORKGROUP>"
        },
```

```
        "Catch": [
          {
            "ErrorEquals": [
              "States.ALL"
            ],
            "Next": "Send query results"
          }
        ],
        "Next": "Map"
      },
      "Map": {
        "Type": "Parallel",
        "ResultSelector": {
          "Query1Result.$": "$[0].ResultSet.Rows",
          "Query2Result.$": "$[1].ResultSet.Rows"
        },
        "Catch": [
          {
            "ErrorEquals": [
              "States.ALL"
            ],
            "Next": "Send query results"
          }
        ],
        "Branches": [
          {
            "StartAt": "Start Athena query 1",
            "States": {
              "Start Athena query 1": {
                "Type": "Task",
                "Resource": "arn:aws:states:::athena:startQueryExecution.sync",
                "Parameters": {
                  "QueryString": "<ATHENA_QUERYSTRING>",
                  "WorkGroup": "<ATHENA_WORKGROUP>"
                },
                "Next": "Get Athena query 1 results"
              },
              "Get Athena query 1 results": {
                "Type": "Task",
                "Resource": "arn:aws:states:::athena:getQueryResults",
                "Parameters": {
                  "QueryExecutionId.$": "$.QueryExecution.QueryExecutionId"
                },
                "End": true
```

```
            }
          }
        },
        {
          "StartAt": "Start Athena query 2",
          "States": {
            "Start Athena query 2": {
              "Type": "Task",
              "Resource": "arn:aws:states:::athena:startQueryExecution.sync",
              "Parameters": {
                  "QueryString": "<ATHENA_QUERYSTRING>",
                  "WorkGroup": "<ATHENA_WORKGROUP>"
              },
              "Next": "Get Athena query 2 results"
            },
            "Get Athena query 2 results": {
              "Type": "Task",
              "Resource": "arn:aws:states:::athena:getQueryResults",
              "Parameters": {
                "QueryExecutionId.$": "$.QueryExecution.QueryExecutionId"
              },
              "End": true
            }
          }
        }
      ],
      "Next": "Send query results"
    },
    "Send query results": {
      "Type": "Task",
      "Resource": "arn:aws:states:::sns:publish",
      "Parameters": {
        "Message.$": "$",
        "TopicArn": "<SNS_TOPIC_ARN>"
      },
      "End": true
    }
  }
}
```

IAM-Beispiele

Diese vom Beispielprojekt generierte Beispielrichtlinie AWS Identity and Access Management (IAM) beinhaltet die geringsten Rechte, die für die Ausführung der Zustandsmaschine und der zugehörigen Ressourcen erforderlich sind. Wir empfehlen, dass Sie nur die Berechtigungen in Ihre IAM-Richtlinien aufnehmen, die erforderlich sind.

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

SNS Publish

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

Informationen zur Konfiguration von IAM bei der Verwendung von Step Functions mit anderen AWS Diensten finden Sie unterIAM-Richtlinien für integrierte Dienste.

Große Datensätze abfragen (Amazon Athena, Amazon S3 AWS Glue, Amazon SNS)

Dieses Beispielprojekt zeigt, wie Sie einen großen Datensatz in Amazon S3 aufnehmen und über AWS Glue Crawler partitionieren und dann Amazon Athena Athena-Abfragen für diese Partition ausführen.

In diesem Projekt ruft die Step Functions Functions-Zustandsmaschine einen AWS Glue Crawler auf, der einen großen Datensatz in Amazon S3 partitioniert. Sobald der AWS Glue Crawler eine Erfolgsmeldung zurückgibt, führt der Workflow Athena-Abfragen für diese Partition aus. Sobald die Abfrageausführung erfolgreich abgeschlossen wurde, wird eine Amazon SNS SNS-Benachrichtigung an ein Amazon SNS SNS-Thema gesendet.

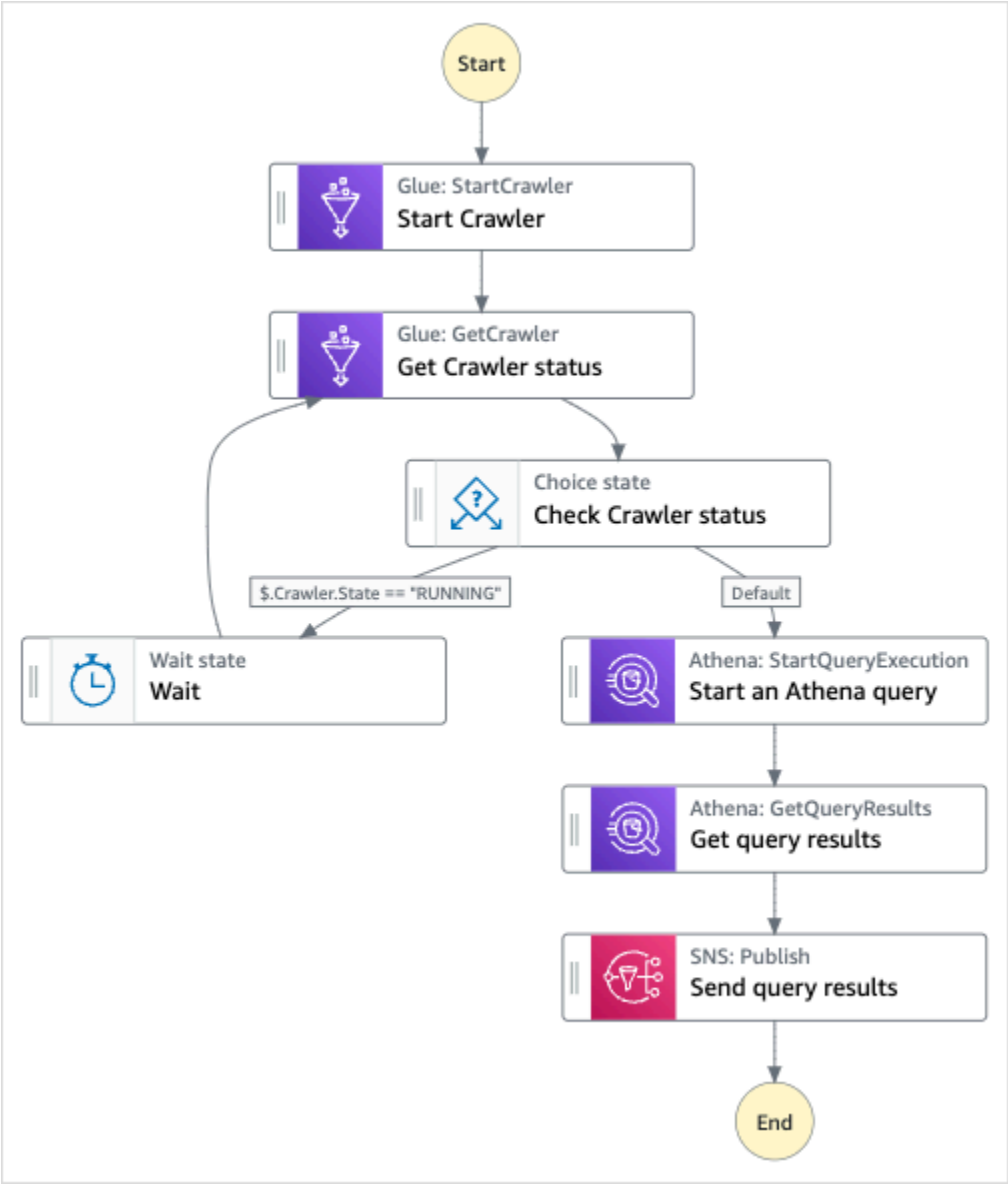
Schritt 1: Erstellen Sie die Zustandsmaschine und stellen Sie Ressourcen bereit

1. Öffnen Sie die Step Functions Functions-Konsole und wählen Sie Create State Machine.
2. Geben Sie **`Query large datasets`** etwas in das Suchfeld ein, und wählen Sie dann Große Datensätze aus den zurückgegebenen Suchergebnissen abfragen aus.
3. Wählen Sie Next (Weiter), um fortzufahren.
4. Step Functions listet die Funktionen auf, die in dem von Ihnen ausgewählten Beispielprojekt AWS-Services verwendet wurden. Außerdem wird ein Workflow-Diagramm für das Beispielprojekt angezeigt. Stellen Sie dieses Projekt für Sie bereit AWS-Konto oder verwenden Sie es als Ausgangspunkt für die Erstellung Ihrer eigenen Projekte. Je nachdem, wie Sie vorgehen möchten, wählen Sie „Demo ausführen“ oder „Darauf aufbauen“.

   Dieses Beispielprojekt stellt die folgenden Ressourcen bereit:

   - Einen Amazon S3-Bucket
   - Ein AWS-Glue-Crawler
   - Ein Amazon SNS-Thema
   - Ein AWS Step Functions-Zustandsautomat
   - Zugehörige AWS Identity and Access Management (IAM)-Rollen

   Die folgende Abbildung zeigt das Workflow-Diagramm für das Beispielprojekt Query large datasets:



5. Wählen Sie Vorlage verwenden, um mit Ihrer Auswahl fortzufahren.
6. Führen Sie eine der folgenden Aktionen aus:

   - Wenn Sie „Darauf aufbauen“ ausgewählt haben, erstellt Step Functions den Workflow-Prototyp für das von Ihnen ausgewählte Beispielprojekt. Step Functions stellt die in der Workflow-Definition aufgeführten Ressourcen nicht bereit.

     Ziehen Sie in Workflow Studio Status per Drag-and-Drop aus demEntwurfsmodus, Bundesstaaten-Browser um mit der Erstellung Ihres Workflow-Prototyps fortzufahren. Oder wechseln Sie zu demCodemodus, der einen integrierten Code-Editor bietet, der VS Code ähnelt, um die Amazon States Language (ASL-) Definition Ihrer Zustandsmaschine in der Step Functions Functions-Konsole zu aktualisieren. Weitere Informationen zur Verwendung

von Workflow Studio zum Erstellen Ihrer Zustandsmaschinen finden Sie unterVerwenden von Workflow Studio.

> **Important**
>
> Denken Sie daran, den Platzhalter Amazon Resource Name (ARN) für die im Beispielprojekt verwendeten Ressourcen zu aktualisieren, bevor Sie Ihren Workflow ausführen.

- Wenn Sie Eine Demo ausführen ausgewählt haben, erstellt Step Functions ein schreibgeschütztes Beispielprojekt, das eine AWS CloudFormation Vorlage verwendet, um die in dieser Vorlage aufgeführten AWS Ressourcen für Ihr Projekt bereitzustellen. AWS-Konto

  > **Tip**
  >
  > Um die State-Machine-Definition des Beispielprojekts anzuzeigen, wählen Sie Code.

  Wenn Sie bereit sind, wählen Sie Deploy and run aus, um das Beispielprojekt bereitzustellen und die Ressourcen zu erstellen.

  Es kann bis zu 10 Minuten dauern, bis diese Ressourcen und die zugehörigen IAM-Berechtigungen erstellt sind. Während der Bereitstellung Ihrer Ressourcen können Sie den CloudFormation Stack-ID-Link öffnen, um zu sehen, welche Ressourcen bereitgestellt werden.

  Nachdem alle Ressourcen im Beispielprojekt erstellt wurden, wird das neue Beispielprojekt auf der Seite State Machines aufgeführt.

  > **Important**
  >
  > Für jeden in der CloudFormation Vorlage verwendeten Dienst können Standardgebühren anfallen.

Schritt 2: Führen Sie die Zustandsmaschine aus

1. Wählen Sie auf der Seite State Machines Ihr Beispielprojekt aus.
2. Wählen Sie auf der Seite mit dem Beispielprojekt die Option Ausführung starten aus.

3. Gehen Sie im Dialogfeld Ausführung starten wie folgt vor:

    1. (Optional) Um Ihre Ausführung zu identifizieren, können Sie im Feld Name einen Namen dafür angeben. Standardmäßig generiert Step Functions automatisch einen eindeutigen Ausführungsnamen.

        > **Note**
        >
        > Mit Step Functions können Sie Namen für Zustandsmaschinen, Ausführungen, Aktivitäten und Labels erstellen, die Nicht-ASCII-Zeichen enthalten. Diese Nicht-ASCII-Namen funktionieren nicht mit Amazon. CloudWatch Um sicherzustellen, dass Sie CloudWatch Messwerte verfolgen können, wählen Sie einen Namen, der nur ASCII-Zeichen verwendet.

    2. (Optional) Geben Sie in das Eingabefeld Eingabewerte im JSON-Format ein, um Ihren Workflow auszuführen.

        Wenn Sie sich dafür entschieden haben, eine Demo auszuführen, müssen Sie keine Ausführungseingaben angeben.

    3. Wählen Sie Start execution (Ausführung starten) aus.
    4. Die Step Functions Functions-Konsole leitet Sie zu einer Seite weiter, die mit Ihrer Ausführungs-ID betitelt ist. Diese Seite wird als Seite mit den Ausführungsdetails bezeichnet. Auf dieser Seite können Sie die Ausführungsergebnisse im Verlauf oder nach Abschluss der Ausführung überprüfen.

        Um die Ausführungsergebnisse zu überprüfen, wählen Sie in der Diagrammansicht einzelne Status und dann die einzelnen Registerkarten im Schrittdetails Bereich aus, um die Details der einzelnen Status, einschließlich Eingabe, Ausgabe und Definition, anzuzeigen. Einzelheiten zu den Ausführungsinformationen, die Sie auf der Seite mit den Ausführungsdetails einsehen können, finden Sie unterSeite mit Ausführungsdetails – Schnittstellenübersicht.

Code des Zustandsautomaten aus diesem Beispiel

Die Zustandsmaschine in diesem Beispielprojekt lässt sich in Amazon S3 AWS Glue, Amazon Athena und Amazon SNS integrieren, indem Parameter direkt an diese Ressourcen übergeben werden.

Sehen Sie sich diese Beispiel-Zustandsmaschine an, um zu sehen, wie Step Functions Amazon S3 AWS Glue, Amazon Athena und Amazon SNS steuert, indem es eine Verbindung zum Amazon-

Ressourcennamen (ARN) im `Resource` Feld herstellt und `Parameters` an die Service-API weiterleitet.

Weitere Informationen darüber, wie Sie andere AWS Dienste steuern AWS Step Functions können, finden Sie unter. Verwendung AWS Step Functions mit anderen Diensten

```
{
    "Comment": "An example demonstrates how to ingest a large data set in Amazon S3 and
 partition it through aws Glue Crawlers, then execute Amazon Athena queries against
 that partition.",
   "StartAt": "Start Crawler",
   "States": {
     "Start Crawler": {
       "Type": "Task",
       "Next": "Get Crawler status",
       "Parameters": {
         "Name": "<GLUE_CRAWLER_NAME>"
       },
       "Resource": "arn:aws:states:::aws-sdk:glue:startCrawler"
     },
     "Get Crawler status": {
       "Type": "Task",
       "Parameters": {
         "Name": "<GLUE_CRAWLER_NAME>"
       },
       "Resource": "arn:aws:arn:aws:states:::aws-sdk:glue:getCrawler",
       "Next": "Check Crawler status"
     },
     "Check Crawler status": {
       "Type": "Choice",
       "Choices": [
         {
           "Variable": "$.Crawler.State",
           "StringEquals": "RUNNING",
           "Next": "Wait"
         }
       ],
       "Default": "Start an Athena query"
     },
     "Wait": {
       "Type": "Wait",
       "Seconds": 30,
       "Next": "Get Crawler status"
```

```
      },
      "Start an Athena query": {
        "Resource": "arn:aws:states:::athena:startQueryExecution.sync",
        "Parameters": {
            "QueryString": "<ATHENA_QUERYSTRING>",
            "WorkGroup": "<ATHENA_WORKGROUP>"
        },
        "Type": "Task",
        "Next": "Get query results"
      },
      "Get query results": {
        "Resource": "arn:aws:states:::athena:getQueryResults",
        "Parameters": {
          "QueryExecutionId.$": "$.QueryExecution.QueryExecutionId"
        },
        "Type": "Task",
        "Next": "Send query results"
      },
      "Send query results": {
        "Resource": "arn:aws:states:::sns:publish",
        "Parameters": {
          "TopicArn": "<SNS_TOPIC_ARN>",
          "Message": {
            "Input.$": "$.ResultSet.Rows"
          }
        },
        "Type": "Task",
        "End": true
      }
    }
  }
```

IAM-Beispiele

Diese vom Beispielprojekt generierten Beispielrichtlinien AWS Identity and Access Management (IAM) beinhalten die geringsten Rechte, die für die Ausführung der Zustandsmaschine und der zugehörigen Ressourcen erforderlich sind. Wir empfehlen, dass Sie nur die Berechtigungen in Ihre IAM-Richtlinien aufnehmen, die erforderlich sind.

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": [
                "*"
            ]
        }
    ]
}
```

SNS Publish

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

Informationen zur Konfiguration von IAM bei der Verwendung von Step Functions mit anderen AWS Diensten finden Sie unterIAM-Richtlinien für integrierte Dienste.

Daten auf dem neuesten Stand halten (Amazon Athena, Amazon S3, AWS Glue)

Dieses Beispielprojekt zeigt, wie Sie eine Zieltabelle abfragen, um aktuelle Daten mit AWS Glue Catalog abzurufen, und sie dann mithilfe von Amazon Athena mit neuen Daten aus anderen Quellen aktualisieren.

In diesem Projekt ruft die Step Functions Functions-Zustandsmaschine AWS Glue Catalog auf, um zu überprüfen, ob eine Zieltabelle in einem Amazon S3 S3-Bucket vorhanden ist. Wenn keine Tabelle gefunden wird, wird eine neue Tabelle erstellt. Step FunctionsFührt dann eine Athena-Abfrage aus, um der Zieltabelle Zeilen aus einer anderen Datenquelle hinzuzufügen: zuerst wird die Zieltabelle abgefragt, um das neueste Datum zu erhalten, dann wird die Quelltabelle nach neueren Daten abgefragt und in die Zieltabelle eingefügt.

Schritt 1: Erstellen Sie die Zustandsmaschine und stellen Sie Ressourcen bereit

1. Öffnen Sie die Step Functions Functions-Konsole und wählen Sie Create State Machine.
2. Geben Sie etwas **`Keep data up to date`** in das Suchfeld ein und wählen Sie dann Daten aus den zurückgegebenen Suchergebnissen auf dem neuesten Stand halten aus.
3. Wählen Sie Next (Weiter), um fortzufahren.
4. Step Functions listet die Funktionen auf, die in dem von Ihnen ausgewählten Beispielprojekt AWS-Services verwendet wurden. Außerdem wird ein Workflow-Diagramm für das Beispielprojekt angezeigt. Stellen Sie dieses Projekt für Ihr Projekt bereit AWS-Konto oder verwenden Sie es als Ausgangspunkt für die Erstellung Ihrer eigenen Projekte. Je nachdem, wie Sie vorgehen möchten, wählen Sie „Demo ausführen“ oder „Darauf aufbauen“.

   In diesem Beispielprojekt werden die folgenden Ressourcen bereitgestellt:

   - Einen Amazon S3-Bucket
   - Amazon AthenaAbfragen
   - Ein AWS Glue Data Catalog Anruf
   - Ein AWS Step Functions-Zustandsautomat
   - Zugehörige AWS Identity and Access Management (IAM)-Rollen

Die folgende Abbildung zeigt das Workflow-Diagramm für das Beispielprojekt Keep data up to date:

5. Wählen Sie Vorlage verwenden, um mit Ihrer Auswahl fortzufahren.
6. Führen Sie eine der folgenden Aktionen aus:

   - Wenn Sie „Darauf aufbauen“ ausgewählt haben, erstellt Step Functions den Workflow-Prototyp für das von Ihnen ausgewählte Beispielprojekt. Step Functions stellt die in der Workflow-Definition aufgeführten Ressourcen nicht bereit.

     Ziehen Sie in Workflow Studio Status per Drag-and-Drop aus demEntwurfsmodus, Bundesstaaten-Browser um mit der Erstellung Ihres Workflow-Prototyps fortzufahren. Oder wechseln Sie zu demCodemodus, der einen integrierten Code-Editor bietet, der VS Code ähnelt, um die Amazon States Language (ASL-) Definition Ihrer Zustandsmaschine in der Step Functions Functions-Konsole zu aktualisieren. Weitere Informationen zur Verwendung von Workflow Studio zum Erstellen Ihrer Zustandsmaschinen finden Sie unterVerwenden von Workflow Studio.

> **Important**
>
> Denken Sie daran, den Platzhalter Amazon Resource Name (ARN) für die im Beispielprojekt verwendeten Ressourcen zu aktualisieren, bevor Sie Ihren Workflow ausführen.

- Wenn Sie Eine Demo ausführen ausgewählt haben, erstellt Step Functions ein schreibgeschütztes Beispielprojekt, das eine AWS CloudFormation Vorlage verwendet, um die in dieser Vorlage aufgeführten AWS Ressourcen für Ihr Projekt bereitzustellen. AWS-Konto

  > **Tip**
  >
  > Um die State-Machine-Definition des Beispielprojekts anzuzeigen, wählen Sie Code.

  Wenn Sie bereit sind, wählen Sie Deploy and run aus, um das Beispielprojekt bereitzustellen und die Ressourcen zu erstellen.

  Es kann bis zu 10 Minuten dauern, bis diese Ressourcen und die zugehörigen IAM-Berechtigungen erstellt sind. Während der Bereitstellung Ihrer Ressourcen können Sie den CloudFormation Stack-ID-Link öffnen, um zu sehen, welche Ressourcen bereitgestellt werden.

  Nachdem alle Ressourcen im Beispielprojekt erstellt wurden, wird das neue Beispielprojekt auf der Seite State Machines aufgeführt.

  > **Important**
  >
  > Für jeden in der CloudFormation Vorlage verwendeten Dienst können Standardgebühren anfallen.

Schritt 2: Führen Sie die Zustandsmaschine aus

1. Wählen Sie auf der Seite State Machines Ihr Beispielprojekt aus.
2. Wählen Sie auf der Seite mit dem Beispielprojekt die Option Ausführung starten aus.
3. Gehen Sie im Dialogfeld Ausführung starten wie folgt vor:

1. (Optional) Um Ihre Ausführung zu identifizieren, können Sie im Feld Name einen Namen dafür angeben. Standardmäßig generiert Step Functions automatisch einen eindeutigen Ausführungsnamen.

   > **Note**
   >
   > Mit Step Functions können Sie Namen für Zustandsmaschinen, Ausführungen, Aktivitäten und Labels erstellen, die Nicht-ASCII-Zeichen enthalten. Diese Nicht-ASCII-Namen funktionieren nicht mit Amazon. CloudWatch Um sicherzustellen, dass Sie CloudWatch Messwerte verfolgen können, wählen Sie einen Namen, der nur ASCII-Zeichen verwendet.

2. (Optional) Geben Sie in das Eingabefeld Eingabewerte im JSON-Format ein, um Ihren Workflow auszuführen.

   Wenn Sie sich dafür entschieden haben, eine Demo auszuführen, müssen Sie keine Ausführungseingaben angeben.

3. Wählen Sie Start execution (Ausführung starten) aus.

4. Die Step Functions Functions-Konsole leitet Sie zu einer Seite weiter, die mit Ihrer Ausführungs-ID betitelt ist. Diese Seite wird als Seite mit den Ausführungsdetails bezeichnet. Auf dieser Seite können Sie die Ausführungsergebnisse im Verlauf der Ausführung oder nach deren Abschluss überprüfen.

   Um die Ausführungsergebnisse zu überprüfen, wählen Sie in der Diagrammansicht einzelne Status aus und wählen Sie dann die einzelnen Registerkarten im Schrittdetails Bereich, um die Details der einzelnen Status, einschließlich Eingabe, Ausgabe und Definition, anzuzeigen. Einzelheiten zu den Ausführungsinformationen, die Sie auf der Seite mit den Ausführungsdetails einsehen können, finden Sie unterSeite mit Ausführungsdetails – Schnittstellenübersicht.

Code des Zustandsautomaten aus diesem Beispiel

Die Zustandsmaschine in diesem Beispielprojekt lässt sich in Amazon S3 und Amazon Athena integrieren AWS Glue, indem Parameter direkt an diese Ressourcen übergeben werden.

Sehen Sie sich diese Beispiel-Zustandsmaschine an, um zu sehen AWS Glue, wie Step Functions Amazon S3 und Amazon Athena steuert, indem es eine Verbindung zum Amazon-Ressourcennamen (ARN) im `Resource` Feld herstellt und `Parameters` an die Service-API weiterleitet.

Weitere Informationen darüber, wie AWS Step Functions Sie andere AWS Dienste steuern können, finden Sie unter[Verwendung AWS Step Functions mit anderen Diensten](#).

```
{
    "Comment": "An example demonstrates how to use Athena to query a target table to
 get current data, then update it with new data from other sources.",
    "StartAt": "Get Target Table",
    "States": {
      "Get Target Table": {
        "Type": "Task",
        "Parameters": {
          "DatabaseName": "<GLUE_DATABASE_NAME>",
          "Name": "target"
        },
        "Catch": [
          {
            "ErrorEquals": [
              "Glue.EntityNotFoundException"
            ],
            "Next": "Create Target Table"
          }
        ],
        "Resource": "arn:aws:states:::aws-sdk:glue:getTable",
        "Next": "Update Target Table"
      },
      "Create Target Table": {
        "Resource": "arn:aws:states:::athena:startQueryExecution.sync",
        "Parameters": {
            "QueryString": "<ATHENA_QUERYSTRING>",
            "WorkGroup": "<ATHENA_WORKGROUP>"
        },
        "Type": "Task",
        "Next": "Update Target Table"
      },
      "Update Target Table": {
        "Resource": "arn:aws:states:::athena:startQueryExecution.sync",
        "Parameters": {
            "QueryString": "<ATHENA_QUERYSTRING>",
            "WorkGroup": "<ATHENA_WORKGROUP>"
```

```
        },
        "Type": "Task",
        "End": true
      }
    }
  }
```

IAM-Beispiel

Diese vom Beispielprojekt generierte Beispielrichtlinie AWS Identity and Access Management (IAM) beinhaltet die geringsten Rechte, die für die Ausführung der Zustandsmaschine und der zugehörigen Ressourcen erforderlich sind. Wir empfehlen, dass Sie nur die Berechtigungen in Ihre IAM-Richtlinien aufnehmen, die erforderlich sind.

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": [
                "*"
```

```
            ]
        }
    ]
}
```

Informationen zur Konfiguration von IAM bei der Verwendung von Step Functions mit anderen AWS Diensten finden Sie unterIAM-Richtlinien für integrierte Dienste.

Einen Amazon EKS-Cluster verwalten

Dieses Beispielprojekt zeigt, wie Sie Step Functions und Amazon Elastic Kubernetes Service verwenden, um einen Amazon EKS-Cluster mit einer Knotengruppe zu erstellen, einen Job auf Amazon EKS auszuführen und anschließend die Ausgabe zu untersuchen. Wenn der Vorgang abgeschlossen ist, werden die Knotengruppen und der Amazon EKS-Cluster entfernt.

Weitere Informationen zu Step Functions- und Step Functions Functions-Dienstintegrationen finden Sie im Folgenden:

- Verwendung AWS Step Functions mit anderen Diensten
- Rufen Sie Amazon EKS mit Step Functions auf

> **Note**
>
> Für dieses Beispielprojekt können Gebühren anfallen.
> Für neue AWS Benutzer ist ein kostenloses Nutzungskontingent verfügbar. Im Rahmen dieses Kontingents sind die Services bis zu einem bestimmten Nutzungsumfang kostenlos. Weitere Informationen zu den AWS Kosten und dem kostenlosen Kontingent finden Sie unter Amazon EKS-Preise.

Schritt 1: Erstellen Sie die Zustandsmaschine und stellen Sie Ressourcen bereit

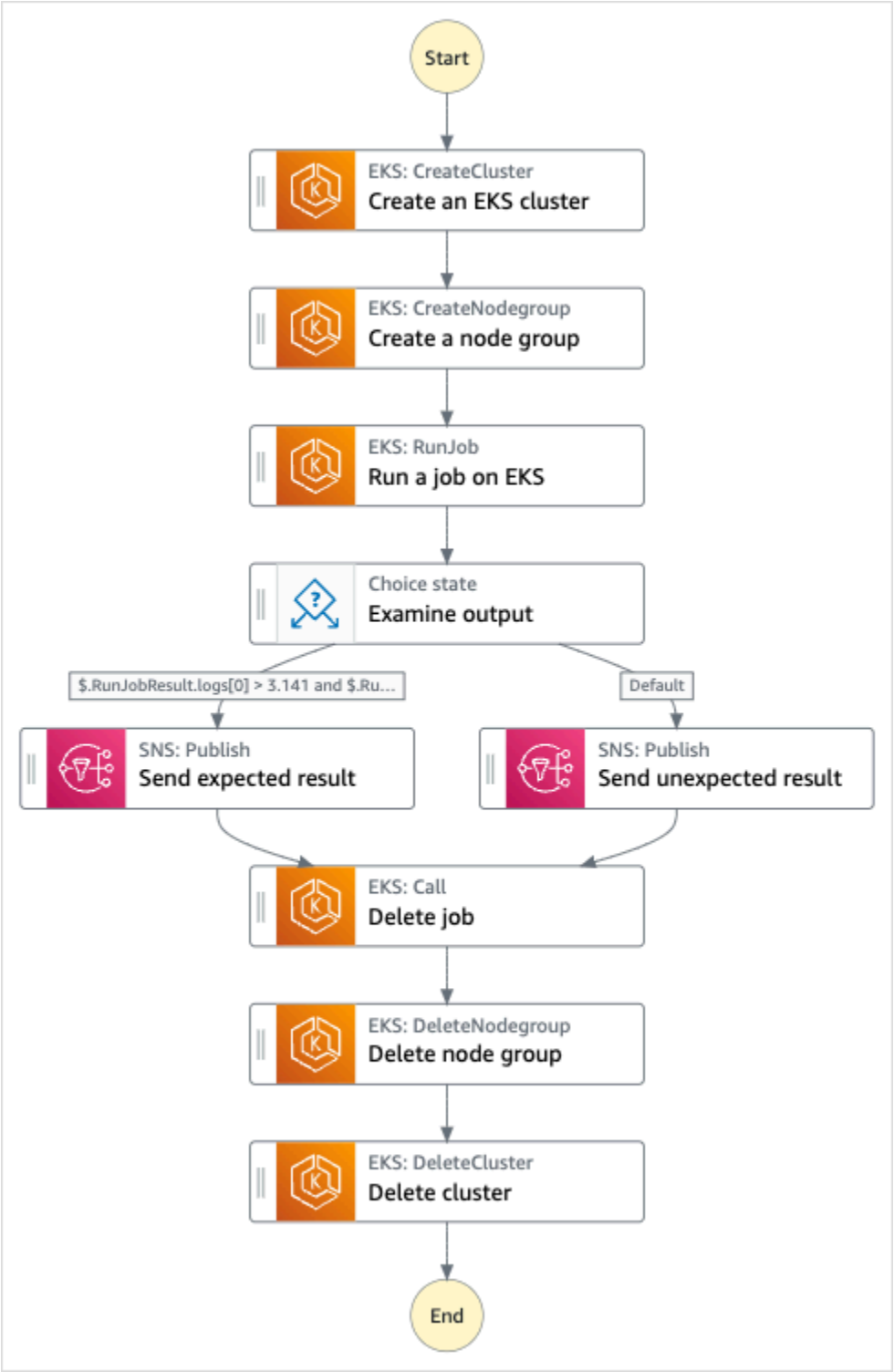
1. Öffnen Sie die Step Functions Functions-Konsole und wählen Sie Create State Machine.
2. Geben Sie **`Manage an EKS cluster`** etwas in das Suchfeld ein und wählen Sie dann aus den zurückgegebenen Suchergebnissen die Option Einen EKS-Cluster verwalten aus.

3. Wählen Sie Next (Weiter), um fortzufahren.
4. Step Functions listet die Funktionen auf, die in dem von Ihnen ausgewählten Beispielprojekt AWS-Services verwendet wurden. Außerdem wird ein Workflow-Diagramm für das Beispielprojekt angezeigt. Stellen Sie dieses Projekt für Sie bereit AWS-Konto oder verwenden Sie es als Ausgangspunkt für die Erstellung Ihrer eigenen Projekte. Je nachdem, wie Sie vorgehen möchten, wählen Sie „Demo ausführen“ oder „Darauf aufbauen“.

   In diesem Beispielprojekt werden die folgenden Ressourcen bereitgestellt:

   - Ein Cluster Amazon Elastic Kubernetes Service
   - Ein Amazon SNS-Thema
   - Ein AWS Step Functions-Zustandsautomat
   - Zugehörige AWS Identity and Access Management (IAM)-Rollen

   Die folgende Abbildung zeigt das Workflow-Diagramm für das Beispielprojekt Manage an EKS cluster:



5. Wählen Sie Vorlage verwenden, um mit Ihrer Auswahl fortzufahren.
6. Führen Sie eine der folgenden Aktionen aus:

- Wenn Sie „Darauf aufbauen“ ausgewählt haben, erstellt Step Functions den Workflow-Prototyp für das von Ihnen ausgewählte Beispielprojekt. Step Functions stellt die in der Workflow-Definition aufgeführten Ressourcen nicht bereit.

  Ziehen Sie in Workflow Studio Status per Drag-and-Drop aus demEntwurfsmodus, Bundesstaaten-Browser um mit der Erstellung Ihres Workflow-Prototyps fortzufahren. Oder wechseln Sie zu demCodemodus, der einen integrierten Code-Editor bietet, der VS Code ähnelt, um die Amazon States Language (ASL-) Definition Ihrer Zustandsmaschine in der Step Functions Functions-Konsole zu aktualisieren. Weitere Informationen zur Verwendung von Workflow Studio zum Erstellen Ihrer Zustandsmaschinen finden Sie unterVerwenden von Workflow Studio.

  > **Important**
  >
  > Denken Sie daran, den Platzhalter Amazon Resource Name (ARN) für die im Beispielprojekt verwendeten Ressourcen zu aktualisieren, bevor Sie Ihren Workflow ausführen.

- Wenn Sie Eine Demo ausführen ausgewählt haben, erstellt Step Functions ein schreibgeschütztes Beispielprojekt, das eine AWS CloudFormation Vorlage verwendet, um die in dieser Vorlage aufgeführten AWS Ressourcen für Ihr Projekt bereitzustellen. AWS-Konto

  > **Tip**
  >
  > Um die State-Machine-Definition des Beispielprojekts anzuzeigen, wählen Sie Code.

  Wenn Sie bereit sind, wählen Sie Deploy and run aus, um das Beispielprojekt bereitzustellen und die Ressourcen zu erstellen.

  Es kann bis zu 10 Minuten dauern, bis diese Ressourcen und die zugehörigen IAM-Berechtigungen erstellt sind. Während der Bereitstellung Ihrer Ressourcen können Sie den CloudFormation Stack-ID-Link öffnen, um zu sehen, welche Ressourcen bereitgestellt werden.

  Nachdem alle Ressourcen im Beispielprojekt erstellt wurden, wird das neue Beispielprojekt auf der Seite State Machines aufgeführt.

> **Important**
>
> Für jeden in der CloudFormation Vorlage verwendeten Dienst können Standardgebühren anfallen.

Schritt 2: Führen Sie die Zustandsmaschine aus

1. Wählen Sie auf der Seite State Machines Ihr Beispielprojekt aus.
2. Wählen Sie auf der Seite mit dem Beispielprojekt die Option Ausführung starten aus.
3. Gehen Sie im Dialogfeld Ausführung starten wie folgt vor:

   1. (Optional) Um Ihre Ausführung zu identifizieren, können Sie im Feld Name einen Namen dafür angeben. Standardmäßig generiert Step Functions automatisch einen eindeutigen Ausführungsnamen.

      > **Note**
      >
      > Mit Step Functions können Sie Namen für Zustandsmaschinen, Ausführungen und Aktivitäten sowie Beschriftungen erstellen, die Nicht-ASCII-Zeichen enthalten. Diese Nicht-ASCII-Namen funktionieren nicht mit Amazon. CloudWatch Um sicherzustellen, dass Sie CloudWatch Messwerte verfolgen können, wählen Sie einen Namen, der nur ASCII-Zeichen verwendet.

   2. (Optional) Geben Sie in das Eingabefeld Eingabewerte im JSON-Format ein, um Ihren Workflow auszuführen.

      Wenn Sie sich dafür entschieden haben, eine Demo auszuführen, müssen Sie keine Ausführungseingaben angeben.

      > **Note**
      >
      > Wenn das von Ihnen bereitgestellte Demo-Projekt vorab aufgefüllte Eingabedaten für die Ausführung enthält, verwenden Sie diese Eingabe, um die Zustandsmaschine auszuführen.

   3. Wählen Sie Start execution (Ausführung starten) aus.

4. Die Step Functions Functions-Konsole leitet Sie zu einer Seite weiter, die mit Ihrer Ausführungs-ID betitelt ist. Diese Seite wird als Seite mit den Ausführungsdetails bezeichnet. Auf dieser Seite können Sie die Ausführungsergebnisse im Verlauf der Ausführung oder nach deren Abschluss überprüfen.

   Um die Ausführungsergebnisse zu überprüfen, wählen Sie in der Diagrammansicht einzelne Status und dann die einzelnen Registerkarten im Schrittdetails Bereich aus, um die Details der einzelnen Status, einschließlich Eingabe, Ausgabe und Definition, anzuzeigen. Einzelheiten zu den Ausführungsinformationen, die Sie auf der Seite mit den Ausführungsdetails einsehen können, finden Sie unterSeite mit Ausführungsdetails – Schnittstellenübersicht.

Code des Zustandsautomaten aus diesem Beispiel

Die Zustandsmaschine in diesem Beispielprojekt integriert sich in Amazon EKS, indem sie einen Amazon EKS-Cluster und eine Knotengruppe erstellt, und verwendet ein SNS-Thema, um Ergebnisse zurückzugeben.

Sehen Sie sich diese Beispiel-Zustandsmaschine an, um zu sehen, wie Step Functions Amazon EKS-Cluster und -Knotengruppen verwaltet.

Weitere Informationen darüber, wie AWS Step Functions Sie andere AWS Dienste steuern können, finden Sie unterVerwendung AWS Step Functions mit anderen Diensten.

```
{
  "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
    }
  }
}
```

Informationen zur Konfiguration von IAM bei der Verwendung von Step Functions mit anderen AWS Diensten finden Sie unterIAM-Richtlinien für integrierte Dienste.

IAM-Beispiel

Diese vom Beispielprojekt generierten Beispielrichtlinien AWS Identity and Access Management (IAM) beinhalten die geringsten Rechte, die für die Ausführung der Zustandsmaschine und der zugehörigen Ressourcen erforderlich sind. Wir empfehlen, dass Sie nur die Berechtigungen in Ihre IAM-Richtlinien aufnehmen, die erforderlich sind.

```
{
```

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

Informationen zur Konfiguration von IAM bei der Verwendung von Step Functions mit anderen AWS Diensten finden Sie unterIAM-Richtlinien für integrierte Dienste.

Rufen Sie API Gateway auf

Dieses Beispielprojekt zeigt, wie mit Step Functions ein API Gateway aufgerufen wird, und überprüft, ob der Aufruf erfolgreich war.

Weitere Informationen zu API Gateway- und Step Functions Functions-Dienstintegrationen finden Sie im Folgenden:

- Verwendung AWS Step Functions mit anderen Diensten
- API Gateway mit Step-Funktionen aufrufen

Schritt 1: Erstellen Sie die Zustandsmaschine und stellen Sie Ressourcen bereit

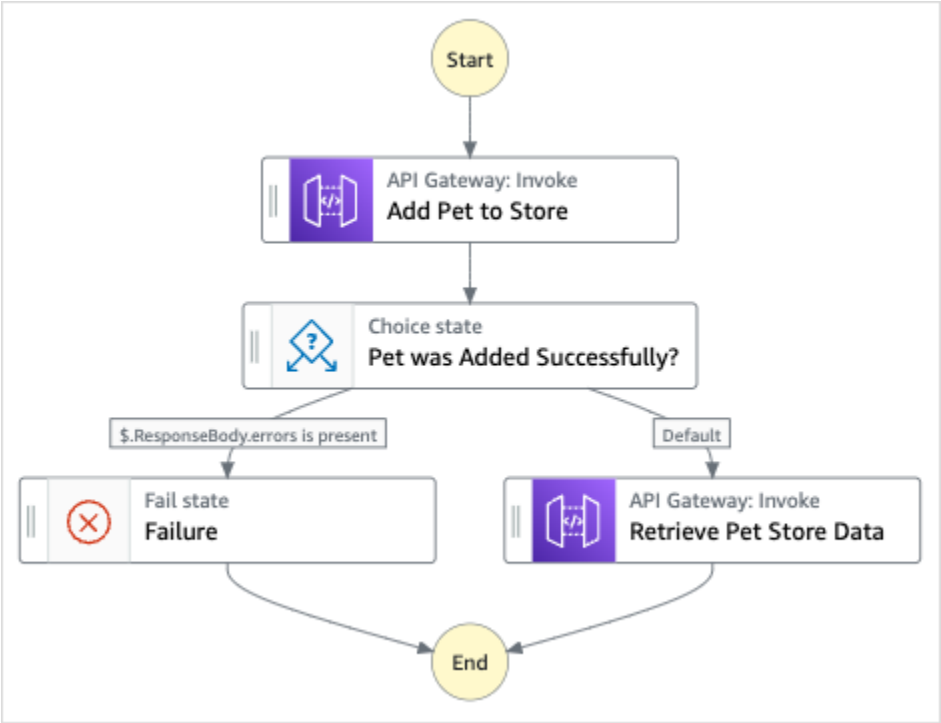
1. Öffnen Sie die Step Functions Functions-Konsole und wählen Sie Create State Machine.
2. Geben Sie etwas **`Make a call to API Gateway`** in das Suchfeld ein, und wählen Sie dann API Gateway aus den zurückgegebenen Suchergebnissen die Option Anruf tätigen aus.
3. Wählen Sie Next (Weiter), um fortzufahren.
4. Step Functions listet die Funktionen auf, die in dem von Ihnen ausgewählten Beispielprojekt AWS-Services verwendet wurden. Außerdem wird ein Workflow-Diagramm für das Beispielprojekt angezeigt. Stellen Sie dieses Projekt für Ihr Projekt bereit AWS-Konto oder verwenden Sie es als Ausgangspunkt für die Erstellung Ihrer eigenen Projekte. Je nachdem, wie Sie vorgehen möchten, wählen Sie „Demo ausführen“ oder „Darauf aufbauen“.

   In diesem Beispielprojekt werden die folgenden Ressourcen bereitgestellt:

   - Eine Amazon API Gateway REST-API, die von der Zustandsmaschine aufgerufen wird.

- Ein AWS Step Functions-Zustandsautomat
- Zugehörige AWS Identity and Access Management (IAM)-Rollen

Die folgende Abbildung zeigt das Workflow-Diagramm für das Projekt Make a call to API Gateway sample:



5. Wählen Sie Vorlage verwenden, um mit Ihrer Auswahl fortzufahren.
6. Führen Sie eine der folgenden Aktionen aus:

   - Wenn Sie „Darauf aufbauen“ ausgewählt haben, erstellt Step Functions den Workflow-Prototyp für das von Ihnen ausgewählte Beispielprojekt. Step Functions stellt die in der Workflow-Definition aufgeführten Ressourcen nicht bereit.

     Ziehen Sie in Workflow Studio Status per Drag-and-Drop aus demEntwurfsmodus, Bundesstaaten-Browser um mit der Erstellung Ihres Workflow-Prototyps fortzufahren. Oder wechseln Sie zu demCodemodus, der einen integrierten Code-Editor bietet, der VS Code ähnelt, um die Amazon States Language (ASL-) Definition Ihrer Zustandsmaschine in der Step Functions Functions-Konsole zu aktualisieren. Weitere Informationen zur Verwendung von Workflow Studio zum Erstellen Ihrer Zustandsmaschinen finden Sie unterVerwenden von Workflow Studio.

> **Important**
>
> Denken Sie daran, den Platzhalter Amazon Resource Name (ARN) für die im Beispielprojekt verwendeten Ressourcen zu aktualisieren, bevor Sie Ihren Workflow ausführen.

- Wenn Sie Eine Demo ausführen ausgewählt haben, erstellt Step Functions ein schreibgeschütztes Beispielprojekt, das eine AWS CloudFormation Vorlage verwendet, um die in dieser Vorlage aufgeführten AWS Ressourcen für Ihr Projekt bereitzustellen. AWS-Konto

  > **Tip**
  >
  > Um die State-Machine-Definition des Beispielprojekts anzuzeigen, wählen Sie Code.

  Wenn Sie bereit sind, wählen Sie Deploy and run aus, um das Beispielprojekt bereitzustellen und die Ressourcen zu erstellen.

  Es kann bis zu 10 Minuten dauern, bis diese Ressourcen und die zugehörigen IAM-Berechtigungen erstellt sind. Während der Bereitstellung Ihrer Ressourcen können Sie den CloudFormation Stack-ID-Link öffnen, um zu sehen, welche Ressourcen bereitgestellt werden.

  Nachdem alle Ressourcen im Beispielprojekt erstellt wurden, wird das neue Beispielprojekt auf der Seite State Machines aufgeführt.

  > **Important**
  >
  > Für jeden in der CloudFormation Vorlage verwendeten Dienst können Standardgebühren anfallen.

Schritt 2: Führen Sie die Zustandsmaschine aus

1. Wählen Sie auf der Seite State Machines Ihr Beispielprojekt aus.
2. Wählen Sie auf der Seite mit dem Beispielprojekt die Option Ausführung starten aus.
3. Gehen Sie im Dialogfeld Ausführung starten wie folgt vor:

1. (Optional) Um Ihre Ausführung zu identifizieren, können Sie im Feld Name einen Namen dafür angeben. Standardmäßig generiert Step Functions automatisch einen eindeutigen Ausführungsnamen.

   > **Note**
   >
   > Mit Step Functions können Sie Namen für Zustandsmaschinen, Ausführungen und Aktivitäten sowie Beschriftungen erstellen, die Nicht-ASCII-Zeichen enthalten. Diese Nicht-ASCII-Namen funktionieren nicht mit Amazon. CloudWatch Um sicherzustellen, dass Sie CloudWatch Messwerte verfolgen können, wählen Sie einen Namen, der nur ASCII-Zeichen verwendet.

2. (Optional) Geben Sie in das Eingabefeld Eingabewerte im JSON-Format ein, um Ihren Workflow auszuführen.

   Wenn Sie sich dafür entschieden haben, eine Demo auszuführen, müssen Sie keine Ausführungseingaben angeben.

   > **Note**
   >
   > Wenn das von Ihnen bereitgestellte Demo-Projekt vorab aufgefüllte Eingabedaten für die Ausführung enthält, verwenden Sie diese Eingabe, um die Zustandsmaschine auszuführen.

3. Wählen Sie Start execution (Ausführung starten) aus.
4. Die Step Functions Functions-Konsole leitet Sie zu einer Seite weiter, die mit Ihrer Ausführungs-ID betitelt ist. Diese Seite wird als Seite mit den Ausführungsdetails bezeichnet. Auf dieser Seite können Sie die Ausführungsergebnisse im Verlauf der Ausführung oder nach deren Abschluss überprüfen.

   Um die Ausführungsergebnisse zu überprüfen, wählen Sie in der Diagrammansicht einzelne Status aus und wählen Sie dann die einzelnen Registerkarten im Schrittdetails Bereich, um die Details der einzelnen Status, einschließlich Eingabe, Ausgabe und Definition, anzuzeigen. Einzelheiten zu den Ausführungsinformationen, die Sie auf der Seite mit den Ausführungsdetails einsehen können, finden Sie unterSeite mit Ausführungsdetails – Schnittstellenübersicht.

Code des Zustandsautomaten aus diesem Beispiel

Die Zustandsmaschine in diesem Beispielprojekt integriert sich in API Gateway, indem sie die API Gateway Gateway-REST-API aufruft und alle erforderlichen Parameter übergibt.

Sehen Sie sich diese Beispiel-Zustandsmaschine an, um zu sehen, wie Step Functions mit API Gateway interagiert.

Weitere Informationen darüber, wie AWS Step Functions Sie andere AWS Dienste steuern können, finden Sie unterVerwendung AWS Step Functions mit anderen Diensten.

```
{
  "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
    }
  }
}
```

Informationen zur Konfiguration von IAM bei der Verwendung von Step Functions mit anderen AWS Diensten finden Sie unterIAM-Richtlinien für integrierte Dienste.

IAM-Beispiel

Diese vom Beispielprojekt generierten Beispielrichtlinien AWS Identity and Access Management (IAM) beinhalten die geringsten Rechte, die für die Ausführung der Zustandsmaschine und der zugehörigen Ressourcen erforderlich sind. Wir empfehlen, dass Sie nur die Berechtigungen in Ihre IAM-Richtlinien aufnehmen, die erforderlich sind.

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

Informationen zur Konfiguration von IAM bei der Verwendung von Step Functions mit anderen AWS Diensten finden Sie unterIAM-Richtlinien für integrierte Dienste.

Rufen Sie mithilfe der API Gateway Gateway-Integration einen auf Fargate laufenden Microservice auf

Dieses Beispielprojekt zeigt, wie Step Functions verwendet werden kann, um API Gateway aufzurufen, um mit einem Service zu interagieren AWS Fargate, und um zu überprüfen, ob der Aufruf erfolgreich war.

Weitere Informationen zu den Dienstintegrationen von API Gateway und Step Functions finden Sie im Folgenden:

- Verwendung AWS Step Functions mit anderen Diensten
- API Gateway mit Step-Funktionen aufrufen

> **Note**
>
> Für dieses Beispielprojekt können Gebühren anfallen.
> Für neue AWS Benutzer ist ein kostenloses Nutzungskontingent verfügbar. Im Rahmen dieses Kontingents sind die Services bis zu einem bestimmten Nutzungsumfang kostenlos. Weitere Informationen zu den AWS Kosten und dem kostenlosen Kontingent finden Sie unter Preise.

Schritt 1: Erstellen Sie den Zustandsmaschine und stellen Sie Ressourcen bereit

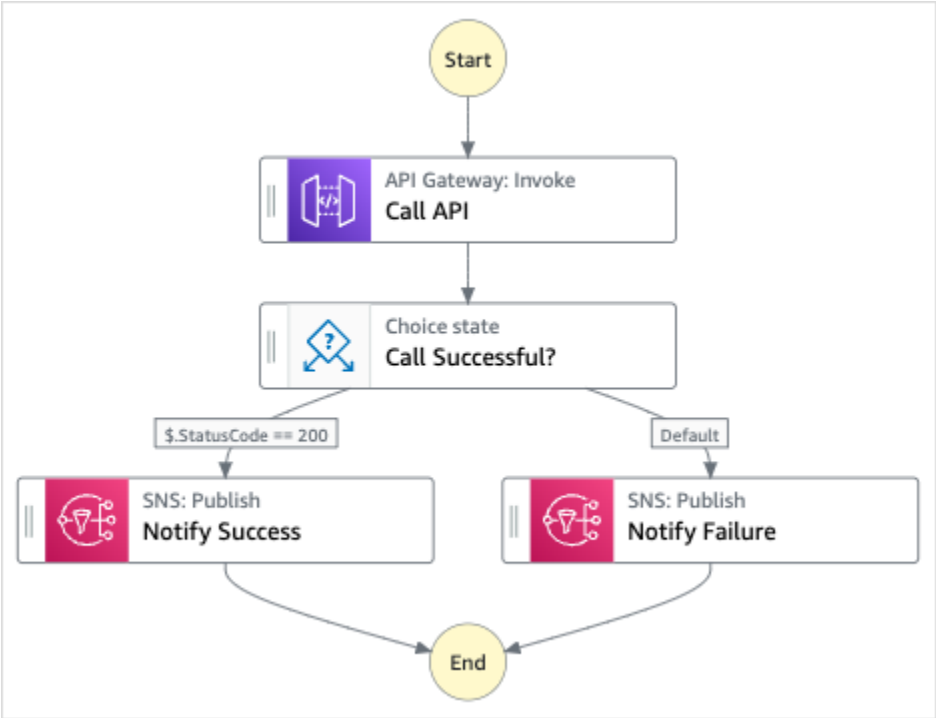
1. Öffnen Sie die Step Functions Functions-Konsole und wählen Sie Create State Machine.

2. Geben Sie **`Call a microservice with API Gateway`** etwas in das Suchfeld ein und wählen Sie dann API Gateway aus den zurückgegebenen Suchergebnissen die Option Einen Microservice aufrufen mit aus.
3. Wählen Sie Next (Weiter), um fortzufahren.
4. Step Functions listet die Funktionen auf, die in dem von Ihnen ausgewählten Beispielprojekt AWS-Services verwendet wurden. Außerdem wird ein Workflow-Diagramm für das Beispielprojekt angezeigt. Stellen Sie dieses Projekt für Sie bereit AWS-Konto oder verwenden Sie es als Ausgangspunkt für die Erstellung Ihrer eigenen Projekte. Je nachdem, wie Sie vorgehen möchten, wählen Sie „Demo ausführen“ oder „Darauf aufbauen“.

   Dieses Beispielprojekt stellt die folgenden Ressourcen bereit:

   - Eine Amazon API Gateway HTTP-API, die von der State Machine aufgerufen wird.
   - Ein Amazon API Gateway Amazon VPC Link.
   - Ein(e) Amazon Virtual Private Cloud.
   - Ein(e) Application Load Balancer.
   - Ein Fargate Cluster.
   - Ein Amazon SNS-Thema
   - Eine AWS Step Functions Zustandsmaschine
   - Verwandte AWS Identity and Access Management Rollen (IAM)
   - Verschiedene zusätzliche Dienste, die erforderlich sind, damit diese Ressourcen zusammenarbeiten können.

   Die folgende Abbildung zeigt das Workflow-Diagramm für das API Gateway Beispielprojekt Call a microservice with:



5. Wählen Sie Vorlage verwenden, um mit Ihrer Auswahl fortzufahren.
6. Führen Sie eine der folgenden Aktionen aus:

   - Wenn Sie „Darauf aufbauen“ ausgewählt haben, erstellt Step Functions den Workflow-Prototyp für das von Ihnen ausgewählte Beispielprojekt. Step Functions stellt die in der Workflow-Definition aufgeführten Ressourcen nicht bereit.

     Ziehen Sie in Workflow Studio Status per Drag-and-Drop aus demEntwurfsmodus, Bundesstaaten-Browser um mit der Erstellung Ihres Workflow-Prototyps fortzufahren. Oder wechseln Sie zu demCodemodus, der einen integrierten Code-Editor bietet, der VS Code ähnelt, um die Amazon States Language (ASL-) Definition Ihrer Zustandsmaschine in der Step Functions Functions-Konsole zu aktualisieren. Weitere Informationen zur Verwendung von Workflow Studio zum Erstellen Ihrer Zustandsmaschinen finden Sie unterVerwenden von Workflow Studio.

     > **Important**
     >
     > Denken Sie daran, den Platzhalter Amazon Resource Name (ARN) für die im Beispielprojekt verwendeten Ressourcen zu aktualisieren, bevor Sie Ihren Workflow ausführen.

- Wenn Sie Eine Demo ausführen ausgewählt haben, erstellt Step Functions ein schreibgeschütztes Beispielprojekt, das eine AWS CloudFormation Vorlage verwendet, um die in dieser Vorlage aufgeführten AWS Ressourcen für Ihr Projekt bereitzustellen. AWS-Konto

  > **Tip**
  >
  > Um die State-Machine-Definition des Beispielprojekts anzuzeigen, wählen Sie Code.

  Wenn Sie bereit sind, wählen Sie Deploy and run aus, um das Beispielprojekt bereitzustellen und die Ressourcen zu erstellen.

  Es kann bis zu 10 Minuten dauern, bis diese Ressourcen und die zugehörigen IAM-Berechtigungen erstellt sind. Während der Bereitstellung Ihrer Ressourcen können Sie den CloudFormation Stack-ID-Link öffnen, um zu sehen, welche Ressourcen bereitgestellt werden.

  Nachdem alle Ressourcen im Beispielprojekt erstellt wurden, wird das neue Beispielprojekt auf der Seite State Machines aufgeführt.

  > **Important**
  >
  > Für jeden in der CloudFormation Vorlage verwendeten Dienst können Standardgebühren anfallen.

Schritt 2: Führen Sie die Zustandsmaschine aus

1. Wählen Sie auf der Seite State Machines Ihr Beispielprojekt aus.
2. Wählen Sie auf der Seite mit dem Beispielprojekt die Option Ausführung starten aus.
3. Gehen Sie im Dialogfeld Ausführung starten wie folgt vor:

   1. (Optional) Um Ihre Ausführung zu identifizieren, können Sie im Feld Name einen Namen dafür angeben. Standardmäßig generiert Step Functions automatisch einen eindeutigen Ausführungsnamen.

> **Note**
>
> Mit Step Functions können Sie Namen für Zustandsmaschinen, Ausführungen und Aktivitäten sowie Labels erstellen, die Nicht-ASCII-Zeichen enthalten. Diese Nicht-ASCII-Namen funktionieren nicht mit Amazon. CloudWatch Um sicherzustellen, dass Sie CloudWatch Messwerte verfolgen können, wählen Sie einen Namen, der nur ASCII-Zeichen verwendet.

2. (Optional) Geben Sie in das Eingabefeld Eingabewerte im JSON-Format ein, um Ihren Workflow auszuführen.

   Wenn Sie sich dafür entschieden haben, eine Demo auszuführen, müssen Sie keine Ausführungseingaben angeben.

   > **Note**
   >
   > Wenn das von Ihnen bereitgestellte Demo-Projekt vorab aufgefüllte Eingabedaten für die Ausführung enthält, verwenden Sie diese Eingabe, um die Zustandsmaschine auszuführen.

3. Wählen Sie Start execution (Ausführung starten) aus.
4. Die Step Functions Functions-Konsole leitet Sie zu einer Seite weiter, die mit Ihrer Ausführungs-ID betitelt ist. Diese Seite wird als Seite mit den Ausführungsdetails bezeichnet. Auf dieser Seite können Sie die Ausführungsergebnisse im Verlauf oder nach Abschluss der Ausführung überprüfen.

   Um die Ausführungsergebnisse zu überprüfen, wählen Sie in der Diagrammansicht einzelne Status und dann die einzelnen Registerkarten im Schrittdetails Bereich aus, um die Details der einzelnen Status, einschließlich Eingabe, Ausgabe und Definition, anzuzeigen. Einzelheiten zu den Ausführungsinformationen, die Sie auf der Seite mit den Ausführungsdetails einsehen können, finden Sie unterSeite mit Ausführungsdetails – Schnittstellenübersicht.

Code des Zustandsautomaten aus diesem Beispiel

Die Zustandsmaschine in diesem Beispielprojekt integriert sich in API Gateway, indem sie eine API Gateway Gateway-HTTP-API aufruft, die mit einem Dienst auf Fargate verbunden ist. Diese wird

in einem privaten Subnetz gehostet und der Zugriff erfolgt über einen privaten Application Load Balancer.

Sehen Sie sich diese Beispiel-Zustandsmaschine an, um zu sehen, wie Step Functions mit API Gateway interagiert und Ergebnisse zurückgibt.

Weitere Informationen darüber, wie AWS Step Functions Sie andere AWS Dienste steuern können, finden Sie unterVerwendung AWS Step Functions mit anderen Diensten.

```
{
  "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
    }
  }
}
```

Informationen zur Konfiguration von IAM bei der Verwendung von Step Functions mit anderen AWS Diensten finden Sie unterIAM-Richtlinien für integrierte Dienste.

IAM-Beispiel

Diese vom Beispielprojekt generierten Beispielrichtlinien AWS Identity and Access Management (IAM) beinhalten die geringsten Rechte, die für die Ausführung der Zustandsmaschine und der zugehörigen Ressourcen erforderlich sind. Wir empfehlen, dass Sie nur die Berechtigungen in Ihre IAM-Richtlinien aufnehmen, die erforderlich sind.

```
{
    "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"
        }
    ]
```

```
}
```

Informationen zur Konfiguration von IAM bei der Verwendung von Step Functions mit anderen AWS Diensten finden Sie unterIAM-Richtlinien für integrierte Dienste.

Senden Sie ein benutzerdefiniertes Ereignis an EventBridge

Dieses Beispielprojekt zeigt, wie Step Functions verwendet werden kann, um ein benutzerdefiniertes Ereignis an einen Event-Bus zu senden, das einer Regel mit mehreren Zielen entspricht (Amazon EventBridge AWS Lambda, Amazon Simple Notification Service, Amazon Simple Queue Service).

Weitere Informationen zu Step Functions- und Step Functions Functions-Dienstintegrationen finden Sie im Folgenden:

- Verwendung AWS Step Functions mit anderen Diensten
- Rufen Sie EventBridge mit Step Functions auf

> **Note**
>
> Für dieses Beispielprojekt können Gebühren anfallen.
> Für neue AWS Benutzer ist ein kostenloses Nutzungskontingent verfügbar. Im Rahmen dieses Kontingents sind die Services bis zu einem bestimmten Nutzungsumfang kostenlos. Weitere Informationen zu den AWS Kosten und dem kostenlosen Kontingent finden Sie unter EventBridge Preise.

Schritt 1: Erstellen Sie den Zustandsmaschine und stellen Sie Ressourcen bereit

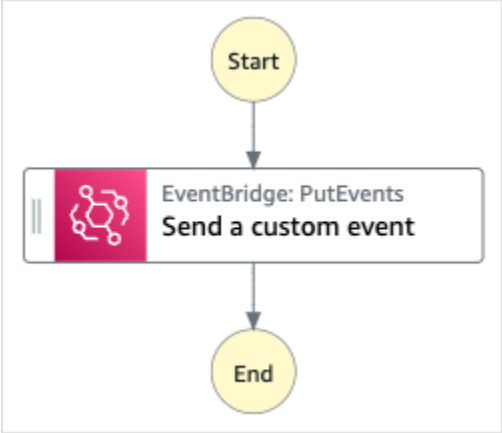
1. Öffnen Sie die Step Functions Functions-Konsole und wählen Sie Create State Machine.
2. Geben Sie **`Send a custom event to EventBridge`** etwas in das Suchfeld ein und wählen Sie dann EventBridge aus den zurückgegebenen Suchergebnissen die Option Benutzerdefiniertes Ereignis senden an aus.
3. Wählen Sie Next (Weiter), um fortzufahren.
4. Step Functions listet die Funktionen auf, die in dem von Ihnen ausgewählten Beispielprojekt AWS-Services verwendet wurden. Außerdem wird ein Workflow-Diagramm für das

Beispielprojekt angezeigt. Stellen Sie dieses Projekt für Sie bereit AWS-Konto oder verwenden Sie es als Ausgangspunkt für die Erstellung Ihrer eigenen Projekte. Je nachdem, wie Sie vorgehen möchten, wählen Sie „Demo ausführen“ oder „Darauf aufbauen“.

Dieses Beispielprojekt stellt die folgenden Ressourcen bereit:

- Ein Ereignis Amazon EventBridge
- Ein Amazon SNS-Thema
- Eine Amazon SQS-Warteschlange
- Eine Lambda-Funktion
- Eine AWS Step Functions Staatsmaschine
- Verwandte AWS Identity and Access Management Rollen (IAM)

Die folgende Abbildung zeigt das Workflow-Diagramm für das EventBridgeBeispielprojekt Ein benutzerdefiniertes Ereignis an:



5. Wählen Sie Vorlage verwenden, um mit Ihrer Auswahl fortzufahren.
6. Führen Sie eine der folgenden Aktionen aus:

   - Wenn Sie „Darauf aufbauen“ ausgewählt haben, erstellt Step Functions den Workflow-Prototyp für das von Ihnen ausgewählte Beispielprojekt. Step Functions stellt die in der Workflow-Definition aufgeführten Ressourcen nicht bereit.

     Ziehen Sie in Workflow Studio Status per Drag-and-Drop aus demEntwurfsmodus, Bundesstaaten-Browser um mit der Erstellung Ihres Workflow-Prototyps fortzufahren. Oder wechseln Sie zu demCodemodus, der einen integrierten Code-Editor bietet, der VS Code ähnelt, um die Amazon States Language (ASL-) Definition Ihrer Zustandsmaschine in der Step Functions Functions-Konsole zu aktualisieren. Weitere Informationen zur Verwendung

von Workflow Studio zum Erstellen Ihrer Zustandsmaschinen finden Sie unterVerwenden von Workflow Studio.

> **Important**
>
> Denken Sie daran, den Platzhalter Amazon Resource Name (ARN) für die im Beispielprojekt verwendeten Ressourcen zu aktualisieren, bevor Sie Ihren Workflow ausführen.

- Wenn Sie Eine Demo ausführen ausgewählt haben, erstellt Step Functions ein schreibgeschütztes Beispielprojekt, das eine AWS CloudFormation Vorlage verwendet, um die in dieser Vorlage aufgeführten AWS Ressourcen für Ihr Projekt bereitzustellen. AWS-Konto

> **Tip**
>
> Um die State-Machine-Definition des Beispielprojekts anzuzeigen, wählen Sie Code.

Wenn Sie bereit sind, wählen Sie Deploy and run aus, um das Beispielprojekt bereitzustellen und die Ressourcen zu erstellen.

Es kann bis zu 10 Minuten dauern, bis diese Ressourcen und die zugehörigen IAM-Berechtigungen erstellt sind. Während der Bereitstellung Ihrer Ressourcen können Sie den CloudFormation Stack-ID-Link öffnen, um zu sehen, welche Ressourcen bereitgestellt werden.

Nachdem alle Ressourcen im Beispielprojekt erstellt wurden, wird das neue Beispielprojekt auf der Seite State Machines aufgeführt.

> **Important**
>
> Für jeden in der CloudFormation Vorlage verwendeten Dienst können Standardgebühren anfallen.

Schritt 2: Führen Sie die Zustandsmaschine aus

1. Wählen Sie auf der Seite State Machines Ihr Beispielprojekt aus.
2. Wählen Sie auf der Seite mit dem Beispielprojekt die Option Ausführung starten aus.

3. Gehen Sie im Dialogfeld Ausführung starten wie folgt vor:

   1. (Optional) Um Ihre Ausführung zu identifizieren, können Sie im Feld Name einen Namen dafür angeben. Standardmäßig generiert Step Functions automatisch einen eindeutigen Ausführungsnamen.

      > **Note**
      >
      > Mit Step Functions können Sie Namen für Zustandsmaschinen, Ausführungen und Aktivitäten sowie Labels erstellen, die Nicht-ASCII-Zeichen enthalten. Diese Nicht-ASCII-Namen funktionieren nicht mit Amazon. CloudWatch Um sicherzustellen, dass Sie CloudWatch Messwerte verfolgen können, wählen Sie einen Namen, der nur ASCII-Zeichen verwendet.

   2. (Optional) Geben Sie in das Eingabefeld Eingabewerte im JSON-Format ein, um Ihren Workflow auszuführen.

      Wenn Sie sich dafür entschieden haben, eine Demo auszuführen, müssen Sie keine Ausführungseingaben angeben.

      > **Note**
      >
      > Wenn das von Ihnen bereitgestellte Demo-Projekt vorab aufgefüllte Eingabedaten für die Ausführung enthält, verwenden Sie diese Eingabe, um die Zustandsmaschine auszuführen.

   3. Wählen Sie Start execution (Ausführung starten) aus.
   4. Die Step Functions Functions-Konsole leitet Sie zu einer Seite weiter, die mit Ihrer Ausführungs-ID betitelt ist. Diese Seite wird als Seite mit den Ausführungsdetails bezeichnet. Auf dieser Seite können Sie die Ausführungsergebnisse im Verlauf oder nach Abschluss der Ausführung überprüfen.

      Um die Ausführungsergebnisse zu überprüfen, wählen Sie in der Diagrammansicht einzelne Status und dann die einzelnen Registerkarten im Schrittdetails Bereich aus, um die Details der einzelnen Status, einschließlich Eingabe, Ausgabe und Definition, anzuzeigen. Einzelheiten zu den Ausführungsinformationen, die Sie auf der Seite mit den Ausführungsdetails einsehen können, finden Sie unterSeite mit Ausführungsdetails – Schnittstellenübersicht.

Code des Zustandsautomaten aus diesem Beispiel

Die Zustandsmaschine in diesem Beispielprojekt integriert sich in, EventBridge indem sie ein benutzerdefiniertes Ereignis an einen EventBridge Event-Bus sendet. Das an den Event-Bus gesendete Ereignis entspricht einer EventBridge Regel, die eine Lambda-Funktion auslöst, die Nachrichten an ein Amazon SNS SNS-Thema und eine Amazon SQS SQS-Warteschlange sendet.

Sehen Sie sich diese Beispiel-Zustandsmaschine an, um zu sehen, wie Step Functions damit umgeht EventBridge.

Weitere Informationen darüber, wie AWS Step Functions Sie andere AWS Dienste steuern können, finden Sie unterVerwendung AWS Step Functions mit anderen Diensten.

```
{
  "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
    }
  }
}
```

Informationen zur Konfiguration von IAM bei der Verwendung von Step Functions mit anderen AWS Diensten finden Sie unterIAM-Richtlinien für integrierte Dienste.

IAM-Beispiel

Diese vom Beispielprojekt generierten Beispielrichtlinien AWS Identity and Access Management (IAM) beinhalten die geringsten Rechte, die für die Ausführung der Zustandsmaschine und der zugehörigen Ressourcen erforderlich sind. Wir empfehlen, dass Sie nur die Berechtigungen in Ihre IAM-Richtlinien aufnehmen, die erforderlich sind.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "events:PutEvents"
            ],
            "Resource": [
                "arn:aws:events:us-east-1:1234567890:event-bus/stepfunctions-
sampleproject-eventbus"
            ],
            "Effect": "Allow"
        }
    ]
}
```

Informationen zur Konfiguration von IAM bei der Verwendung von Step Functions mit anderen AWS Diensten finden Sie unterIAM-Richtlinien für integrierte Dienste.

Synchrone Express-Workflows aufrufen

Dieses Beispielprojekt zeigt, wie Synchronous Express-Workflows über Amazon API Gateway aufgerufen werden, um eine Mitarbeiterdatenbank zu verwalten.

In diesem Projekt verwendet Step Functions API Gateway Gateway-Endpunkte, um synchrone Express-Workflows von Step Functions zu starten. Diese verwenden dann DynamoDB, um Mitarbeiter in einer Mitarbeiterdatenbank zu suchen, hinzuzufügen und zu entfernen.

Weitere Informationen zu Step Functions Synchronous Express-Workflows finden Sie unterSynchrone und asynchrone Express-Workflows.

> **Note**
>
> Für dieses Beispielprojekt können Gebühren anfallen.

Für neue AWS Benutzer ist ein kostenloses Nutzungskontingent verfügbar. Im Rahmen dieses Kontingents sind die Services bis zu einem bestimmten Nutzungsumfang kostenlos. Weitere Informationen zu den AWS Kosten und dem kostenlosen Kontingent finden Sie unter Step Functions — Preise.

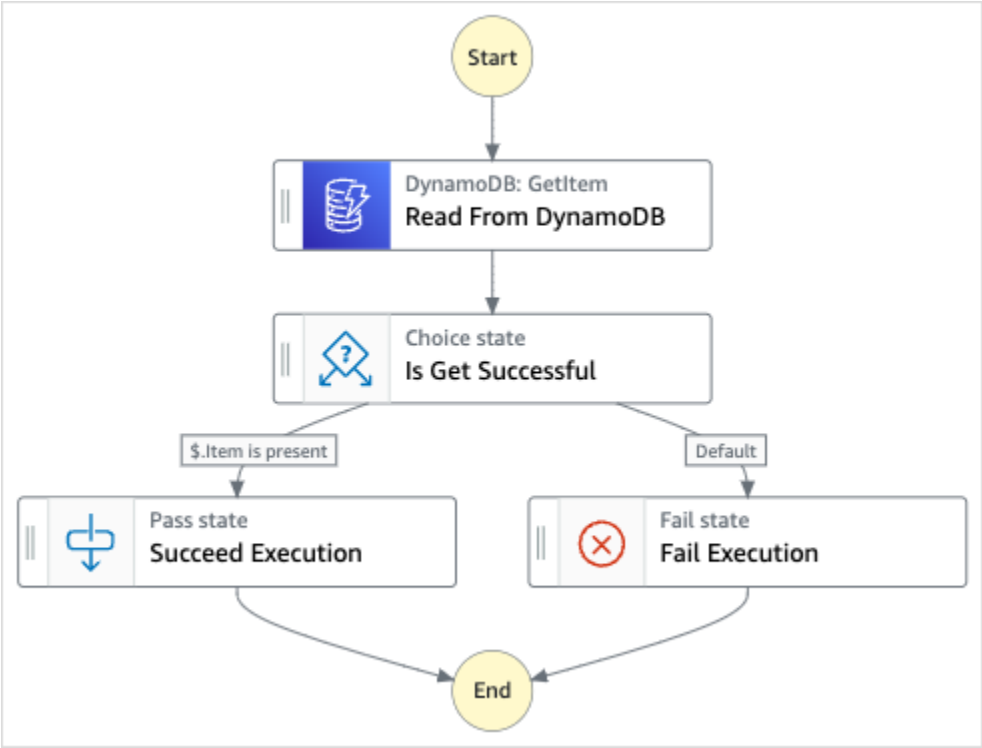
Schritt 1: Erstellen Sie die Zustandsmaschine und stellen Sie Ressourcen bereit

1. Öffnen Sie die Step Functions Functions-Konsole und wählen Sie Create State Machine.
2. Geben Sie **`Invoke Synchronous Express Workflows through API Gateway`** etwas in das Suchfeld ein, und wählen Sie dann aus den zurückgegebenen Suchergebnissen die Option Synchronous Express Workflows aufrufen API Gateway aus.
3. Wählen Sie Next (Weiter), um fortzufahren.
4. Step Functions listet die Funktionen auf, die in dem von Ihnen ausgewählten Beispielprojekt AWS-Services verwendet wurden. Außerdem wird ein Workflow-Diagramm für das Beispielprojekt angezeigt. Stellen Sie dieses Projekt für Sie bereit AWS-Konto oder verwenden Sie es als Ausgangspunkt für die Erstellung Ihrer eigenen Projekte. Je nachdem, wie Sie vorgehen möchten, wählen Sie „Demo ausführen“ oder „Darauf aufbauen“.

   In diesem Beispielprojekt werden die folgenden Ressourcen bereitgestellt:

   - Eine Amazon API Gateway HTTPS-API, die von einer Zustandsmaschine aufgerufen wird.
   - Eine Amazon DynamoDB-Tabelle.
   - Drei AWS Step Functions Zustandsmaschinen.
   - Verwandte Rollen AWS Identity and Access Management (IAM).

   Die folgende Abbildung zeigt das Workflow-Diagramm für das Beispielprojekt Invoke Synchronous Express Workflows: API Gateway



5. Wählen Sie Vorlage verwenden, um mit Ihrer Auswahl fortzufahren.
6. Führen Sie eine der folgenden Aktionen aus:

   - Wenn Sie „Darauf aufbauen“ ausgewählt haben, erstellt Step Functions den Workflow-Prototyp für das von Ihnen ausgewählte Beispielprojekt. Step Functions stellt die in der Workflow-Definition aufgeführten Ressourcen nicht bereit.

     Ziehen Sie in Workflow Studio Status per Drag-and-Drop aus demEntwurfsmodus, Bundesstaaten-Browser um mit der Erstellung Ihres Workflow-Prototyps fortzufahren. Oder wechseln Sie zu demCodemodus, der einen integrierten Code-Editor bietet, der VS Code ähnelt, um die Amazon States Language (ASL-) Definition Ihrer Zustandsmaschine in der Step Functions Functions-Konsole zu aktualisieren. Weitere Informationen zur Verwendung von Workflow Studio zum Erstellen Ihrer Zustandsmaschinen finden Sie unterVerwenden von Workflow Studio.

     > **Important**
     >
     > Denken Sie daran, den Platzhalter Amazon Resource Name (ARN) für die im Beispielprojekt verwendeten Ressourcen zu aktualisieren, bevor Sie Ihren Workflow ausführen.

- Wenn Sie Eine Demo ausführen ausgewählt haben, erstellt Step Functions ein schreibgeschütztes Beispielprojekt, das eine AWS CloudFormation Vorlage verwendet, um die in dieser Vorlage aufgeführten AWS Ressourcen für Ihr Projekt bereitzustellen. AWS-Konto

  > **Tip**
  >
  > Um die State-Machine-Definition des Beispielprojekts anzuzeigen, wählen Sie Code.

  Wenn Sie bereit sind, wählen Sie Deploy and run aus, um das Beispielprojekt bereitzustellen und die Ressourcen zu erstellen.

  Es kann bis zu 10 Minuten dauern, bis diese Ressourcen und die zugehörigen IAM-Berechtigungen erstellt sind. Während der Bereitstellung Ihrer Ressourcen können Sie den CloudFormation Stack-ID-Link öffnen, um zu sehen, welche Ressourcen bereitgestellt werden.

  Nachdem alle Ressourcen im Beispielprojekt erstellt wurden, wird das neue Beispielprojekt auf der Seite State Machines aufgeführt.

  > **Important**
  >
  > Für jeden in der CloudFormation Vorlage verwendeten Dienst können Standardgebühren anfallen.

Schritt 2: Führen Sie die Zustandsmaschine aus

1. Wählen Sie auf der Seite State Machines Ihr Beispielprojekt aus.
2. Wählen Sie auf der Seite mit dem Beispielprojekt die Option Ausführung starten aus.
3. Gehen Sie im Dialogfeld Ausführung starten wie folgt vor:

   1. (Optional) Um Ihre Ausführung zu identifizieren, können Sie im Feld Name einen Namen dafür angeben. Standardmäßig generiert Step Functions automatisch einen eindeutigen Ausführungsnamen.

> **Note**
>
> Mit Step Functions können Sie Namen für Zustandsmaschinen, Ausführungen und Aktivitäten sowie Beschriftungen erstellen, die Nicht-ASCII-Zeichen enthalten. Diese Nicht-ASCII-Namen funktionieren nicht mit Amazon. CloudWatch Um sicherzustellen, dass Sie CloudWatch Messwerte verfolgen können, wählen Sie einen Namen, der nur ASCII-Zeichen verwendet.

2. (Optional) Geben Sie in das Eingabefeld Eingabewerte im JSON-Format ein, um Ihren Workflow auszuführen.

   Wenn Sie sich dafür entschieden haben, eine Demo auszuführen, müssen Sie keine Ausführungseingaben angeben.

   > **Note**
   >
   > Wenn das von Ihnen bereitgestellte Demo-Projekt vorab aufgefüllte Eingabedaten für die Ausführung enthält, verwenden Sie diese Eingabe, um die Zustandsmaschine auszuführen.

3. Wählen Sie Start execution (Ausführung starten) aus.
4. Die Step Functions Functions-Konsole leitet Sie zu einer Seite weiter, die mit Ihrer Ausführungs-ID betitelt ist. Diese Seite wird als Seite mit den Ausführungsdetails bezeichnet. Auf dieser Seite können Sie die Ausführungsergebnisse im Verlauf der Ausführung oder nach deren Abschluss überprüfen.

   Um die Ausführungsergebnisse zu überprüfen, wählen Sie in der Diagrammansicht einzelne Status aus und wählen Sie dann die einzelnen Registerkarten im Schrittdetails Bereich, um die Details der einzelnen Status, einschließlich Eingabe, Ausgabe und Definition, anzuzeigen. Einzelheiten zu den Ausführungsinformationen, die Sie auf der Seite mit den Ausführungsdetails einsehen können, finden Sie unterSeite mit Ausführungsdetails – Schnittstellenübersicht.

Code des Zustandsautomaten aus diesem Beispiel

Die Zustandsmaschine in diesem Beispielprojekt lässt sich in API Gateway und DynamoDB integrieren, indem sie API Gateway verwendet, um einen synchronen Express-Workflow aufzurufen, der dann mithilfe von DynamoDB die Mitarbeiterdatenbank aktualisiert oder aus ihr liest.

Sehen Sie sich diese Beispiel-Zustandsmaschine an, um zu sehen, wie Step Functions Daten aus DynamoDB liest, um Mitarbeiterinformationen abzurufen.

Weitere Informationen zum Aufrufen von Step Functions mithilfe von API Gateway finden Sie im Folgenden.

- API Gateway mit Step-Funktionen aufrufen
- Informationen zum Aufrufen eines privaten Gateways finden Sie im API Gateway Developer Guide.

Weitere Informationen darüber, wie AWS Step Functions Sie andere AWS Dienste steuern können, finden Sie unterVerwendung AWS Step Functions mit anderen Diensten.

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

Informationen zur Konfiguration von IAM bei der Verwendung von Step Functions mit anderen AWS Diensten finden Sie unterIAM-Richtlinien für integrierte Dienste.

IAM-Beispiele

Diese vom Beispielprojekt generierten Beispielrichtlinien AWS Identity and Access Management (IAM) beinhalten die geringsten Rechte, die für die Ausführung der Zustandsmaschine und der zugehörigen Ressourcen erforderlich sind. Wir empfehlen, dass Sie nur die Berechtigungen in Ihre IAM-Richtlinien aufnehmen, die erforderlich sind.

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

Informationen zur Konfiguration von IAM bei der Verwendung von Step Functions mit anderen AWS Diensten finden Sie unterIAM-Richtlinien für integrierte Dienste.

Führen Sie ETL/ELT-Workflows mit Amazon Redshift aus (Lambda, Amazon Redshift Data API)

Dieses Beispielprojekt zeigt, wie Step Functions und die Amazon Redshift Data API verwendet werden, um einen ETL/ELT-Workflow auszuführen, der Daten in das Amazon Redshift Data Warehouse lädt.

In diesem Projekt verwendet Step Functions eine AWS Lambda Funktion und die Amazon Redshift Data API, um die erforderlichen Datenbankobjekte zu erstellen und eine Reihe von Beispieldaten zu generieren, und führt dann parallel zwei Jobs aus, die das Laden von Dimensionstabellen durchführen, gefolgt von einer Faktentabelle. Sobald beide Dimensions-Ladeaufträge erfolgreich beendet wurden, führt Step Functions den Ladejob für die Faktentabelle aus, führt den Validierungsjob aus und unterbricht dann den Amazon Redshift Redshift-Cluster.

> **Note**
>
> Sie können die ETL-Logik ändern, um Daten aus anderen Quellen wie Amazon S3 zu empfangen, das den COPY-Befehl verwenden kann, um Daten von Amazon S3 in eine Amazon Redshift Redshift-Tabelle zu kopieren.

Weitere Informationen zu den Serviceintegrationen von Amazon Redshift und Step Functions finden Sie im Folgenden:

- Verwendung AWS Step Functions mit anderen Diensten
- Verwenden der Amazon Redshift Data API
- Amazon Redshift Daten-API-Dienst
- Eine Step Functions Functions-Zustandsmaschine erstellen, die Lambda verwendet
- IAM-Richtlinien für:
  - IAM-Richtlinien für AWS Lambda
  - Autorisieren des Zugriffs auf die Amazon Redshift Data API

> **Note**
>
> Für dieses Beispielprojekt können Gebühren anfallen.

Für neue AWS Benutzer ist ein kostenloses Nutzungskontingent verfügbar. Im Rahmen dieses Kontingents sind die Services bis zu einem bestimmten Nutzungsumfang kostenlos. Weitere Informationen zu den AWS Kosten und dem kostenlosen Kontingent finden Sie unter AWS Step Functions Preise.

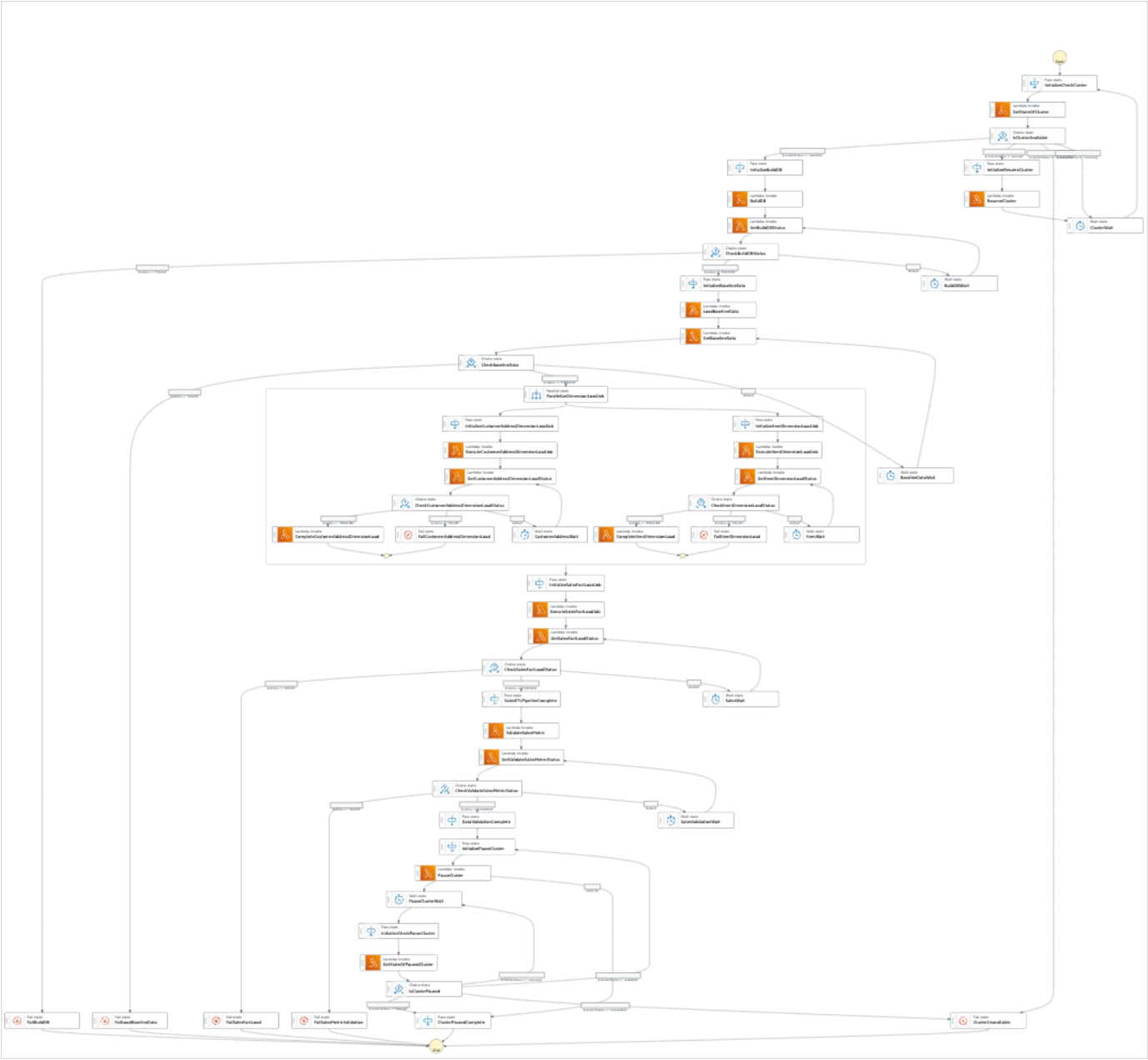
Schritt 1: Erstellen Sie die Zustandsmaschine und stellen Sie Ressourcen bereit

1. Öffnen Sie die Step Functions Functions-Konsole und wählen Sie Create State Machine.
2. Geben Sie **`ETL job in Amazon Redshift`** etwas in das Suchfeld ein und wählen Sie dann ETL Job in Amazon Redshift aus den zurückgegebenen Suchergebnissen aus.
3. Wählen Sie Next (Weiter), um fortzufahren.
4. Step Functions listet die Funktionen auf, die in dem von Ihnen ausgewählten Beispielprojekt AWS-Services verwendet wurden. Außerdem wird ein Workflow-Diagramm für das Beispielprojekt angezeigt. Stellen Sie dieses Projekt für Ihr Projekt bereit AWS-Konto oder verwenden Sie es als Ausgangspunkt für die Erstellung Ihrer eigenen Projekte. Je nachdem, wie Sie vorgehen möchten, wählen Sie „Demo ausführen“ oder „Darauf aufbauen“.

   In diesem Beispielprojekt werden die folgenden Ressourcen bereitgestellt:

   - Ein Cluster Amazon Redshift
   - Zwei Lambda-Funktionen
   - Ein Amazon Redshift Schema
   - Fünf Amazon Redshift Tabellen
   - Eine AWS Step Functions Staatsmaschine
   - Verwandte Rollen AWS Identity and Access Management (IAM).

   Die folgende Abbildung zeigt das Workflow-Diagramm für den ETL-Job im Amazon Redshift Beispielprojekt:



5. Wählen Sie Vorlage verwenden, um mit Ihrer Auswahl fortzufahren.
6. Führen Sie eine der folgenden Aktionen aus:

   - Wenn Sie „Darauf aufbauen“ ausgewählt haben, erstellt Step Functions den Workflow-Prototyp für das von Ihnen ausgewählte Beispielprojekt. Step Functions stellt die in der Workflow-Definition aufgeführten Ressourcen nicht bereit.

     Ziehen Sie in Workflow Studio Status per Drag-and-Drop aus demEntwurfsmodus, Bundesstaaten-Browser um mit der Erstellung Ihres Workflow-Prototyps fortzufahren. Oder wechseln Sie zu demCodemodus, der einen integrierten Code-Editor bietet, der VS Code

ähnelt, um die Amazon States Language (ASL-) Definition Ihrer Zustandsmaschine in der Step Functions Functions-Konsole zu aktualisieren. Weitere Informationen zur Verwendung von Workflow Studio zum Erstellen Ihrer Zustandsmaschinen finden Sie unterVerwenden von Workflow Studio.

> **Important**
>
> Denken Sie daran, den Platzhalter Amazon Resource Name (ARN) für die im Beispielprojekt verwendeten Ressourcen zu aktualisieren, bevor Sie Ihren Workflow ausführen.

- Wenn Sie Eine Demo ausführen ausgewählt haben, erstellt Step Functions ein schreibgeschütztes Beispielprojekt, das eine AWS CloudFormation Vorlage verwendet, um die in dieser Vorlage aufgeführten AWS Ressourcen für Ihr Projekt bereitzustellen. AWS-Konto

  > **Tip**
  >
  > Um die State-Machine-Definition des Beispielprojekts anzuzeigen, wählen Sie Code.

  Wenn Sie bereit sind, wählen Sie Deploy and run aus, um das Beispielprojekt bereitzustellen und die Ressourcen zu erstellen.

  Es kann bis zu 10 Minuten dauern, bis diese Ressourcen und die zugehörigen IAM-Berechtigungen erstellt sind. Während der Bereitstellung Ihrer Ressourcen können Sie den CloudFormation Stack-ID-Link öffnen, um zu sehen, welche Ressourcen bereitgestellt werden.

  Nachdem alle Ressourcen im Beispielprojekt erstellt wurden, wird das neue Beispielprojekt auf der Seite State Machines aufgeführt.

  > **Important**
  >
  > Für jeden in der CloudFormation Vorlage verwendeten Dienst können Standardgebühren anfallen.

Schritt 2: Führen Sie die Zustandsmaschine aus

1. Wählen Sie auf der Seite State Machines Ihr Beispielprojekt aus.
2. Wählen Sie auf der Seite mit dem Beispielprojekt die Option Ausführung starten aus.
3. Gehen Sie im Dialogfeld Ausführung starten wie folgt vor:

    1. (Optional) Um Ihre Ausführung zu identifizieren, können Sie im Feld Name einen Namen dafür angeben. Standardmäßig generiert Step Functions automatisch einen eindeutigen Ausführungsnamen.

        > **Note**
        >
        > Mit Step Functions können Sie Namen für Zustandsmaschinen, Ausführungen und Aktivitäten sowie Beschriftungen erstellen, die Nicht-ASCII-Zeichen enthalten. Diese Nicht-ASCII-Namen funktionieren nicht mit Amazon. CloudWatch Um sicherzustellen, dass Sie CloudWatch Messwerte verfolgen können, wählen Sie einen Namen, der nur ASCII-Zeichen verwendet.

    2. (Optional) Geben Sie in das Eingabefeld Eingabewerte im JSON-Format ein, um Ihren Workflow auszuführen.

        Wenn Sie sich dafür entschieden haben, eine Demo auszuführen, müssen Sie keine Ausführungseingaben angeben.

        > **Note**
        >
        > Wenn das von Ihnen bereitgestellte Demo-Projekt vorab aufgefüllte Eingabedaten für die Ausführung enthält, verwenden Sie diese Eingabe, um die Zustandsmaschine auszuführen.

    3. Wählen Sie Start execution (Ausführung starten) aus.
    4. Die Step Functions Functions-Konsole leitet Sie zu einer Seite weiter, die mit Ihrer Ausführungs-ID betitelt ist. Diese Seite wird als Seite mit den Ausführungsdetails bezeichnet. Auf dieser Seite können Sie die Ausführungsergebnisse im Verlauf der Ausführung oder nach deren Abschluss überprüfen.

        Um die Ausführungsergebnisse zu überprüfen, wählen Sie in der Diagrammansicht einzelne Status aus und wählen Sie dann die einzelnen Registerkarten im Schrittdetails Bereich,

um die Details der einzelnen Status, einschließlich Eingabe, Ausgabe und Definition, anzuzeigen. Einzelheiten zu den Ausführungsinformationen, die Sie auf der Seite mit den Ausführungsdetails einsehen können, finden Sie unterSeite mit Ausführungsdetails – Schnittstellenübersicht.

Code des Zustandsautomaten aus diesem Beispiel

Die Zustandsmaschine in diesem Beispielprojekt lässt sich integrieren, AWS Lambda indem sie die ETL-Logik InputPath direkt an diese Ressourcen weitergibt und asynchron mithilfe der Amazon Redshift Data API ausgeführt wird.

Sehen Sie sich diese Beispiel-Zustandsmaschine an, um zu sehen, wie Step Functions steuert AWS Lambda und welche Amazon Redshift Data API verwendet wird.

Weitere Informationen darüber, wie AWS Step Functions Sie andere AWS Dienste steuern können, finden Sie unterVerwendung AWS Step Functions mit anderen Diensten.

```
{
"Comment": "A simple ETL workflow for loading dimension and fact tables",
  "StartAt": "InitializeCheckCluster",
  "States": {
    "InitializeCheckCluster": {
      "Type": "Pass",
      "Next": "GetStateOfCluster",
      "Result": {
        "input": {
          "redshift_cluster_id": "cfn36-redshiftcluster-AKIAI44QH8DHBEXAMPLE",
          "operation": "status"
        }
      }
    },
    "GetStateOfCluster": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftOperations-AKIAIOSFODNN7EXAMPLE",
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
      "Next": "IsClusterAvailable",
      "InputPath": "$",
      "ResultPath": "$.clusterStatus"
    },
```

```
    "IsClusterAvailable": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.clusterStatus",
          "StringEquals": "available",
          "Next": "InitializeBuildDB"
        },
        {
          "Variable": "$.clusterStatus",
          "StringEquals": "paused",
          "Next": "InitializeResumeCluster"
        },
        {
          "Variable": "$.clusterStatus",
          "StringEquals": "unavailable",
          "Next": "ClusterUnavailable"
        },
        {
          "Variable": "$.clusterStatus",
          "StringEquals": "resuming",
          "Next": "ClusterWait"
        }
      ]
    },
    "ClusterWait": {
      "Type": "Wait",
      "Seconds": 720,
      "Next": "InitializeCheckCluster"
    },
    "InitializeResumeCluster": {
      "Type": "Pass",
      "Next": "ResumeCluster",
      "Result": {
        "input": {
          "redshift_cluster_id": "cfn36-redshiftcluster-AKIAI44QH8DHBEXAMPLE",
          "operation": "resume"
        }
      }
    },
    "ResumeCluster": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftOperations-AKIAIOSFODNN7EXAMPLE",
```

```
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
      "Next": "ClusterWait",
      "InputPath": "$",
      "ResultPath": "$"
    },
    "InitializeBuildDB": {
      "Type": "Pass",
      "Next": "BuildDB",
      "Result": {
        "input": {
          "redshift_cluster_id": "cfn36-redshiftcluster-AKIAI44QH8DHBEXAMPLE",
          "redshift_database": "dev",
          "redshift_user": "awsuser",
          "redshift_schema": "tpcds",
          "action": "build_database",
          "sql_statement": [
            "create schema if not exists {0} authorization {1};",
            "create table if not exists {0}.customer",
            "(c_customer_sk          int4     not null encode az64",
            ",c_customer_id          char(16) not null encode zstd",
            ",c_current_addr_sk      int4              encode az64",
            ",c_first_name           char(20)          encode zstd",
            ",c_last_name            char(30)          encode zstd",
            ",primary key (c_customer_sk)",
            ") distkey(c_customer_sk);",
            "--",
            "create table if not exists {0}.customer_address",
            "(ca_address_sk    int4     not null encode az64",
            ",ca_address_id    char(16) not null encode zstd",
            ",ca_state         char(2)           encode zstd",
            ",ca_zip           char(10)          encode zstd",
            ",ca_country       varchar(20)       encode zstd",
            ",primary key (ca_address_sk)",
            ") distkey(ca_address_sk);",
            "--",
            "create table if not exists {0}.date_dim",
            "(d_date_sk           integer  not null encode az64",
            ",d_date_id           char(16) not null encode zstd",
            ",d_date              date              encode az64",
            ",d_day_name          char(9)           encode zstd",
            ",primary key (d_date_sk)",
            ") diststyle all;",
            "--",
```

```
            "create table if not exists {0}.item",
            "(i_item_sk        int4     not null encode az64",
            ",i_item_id        char(16) not null encode zstd",
            ",i_rec_start_date date              encode az64",
            ",i_rec_end_date   date              encode az64",
            ",i_current_price  numeric(7,2)      encode az64",
            ",i_category       char(50)          encode zstd",
            ",i_product_name   char(50)          encode zstd",
            ",primary key (i_item_sk)",
            ") distkey(i_item_sk) sortkey(i_category);",
            "--",
            "create table if not exists {0}.store_sales",
            "(ss_sold_date_sk       int4",
            ",ss_item_sk            int4 not null encode az64",
            ",ss_customer_sk        int4          encode az64",
            ",ss_addr_sk            int4          encode az64",
            ",ss_store_sk           int4          encode az64",
            ",ss_ticket_number      int8 not null encode az64",
            ",ss_quantity           int4          encode az64",
            ",ss_net_paid           numeric(7,2)  encode az64",
            ",ss_net_profit         numeric(7,2)  encode az64",
            ",primary key (ss_item_sk, ss_ticket_number)",
            ") distkey(ss_item_sk) sortkey(ss_sold_date_sk);"
          ]
        }
      }
    },
    "BuildDB": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
      "Next": "GetBuildDBStatus",
      "InputPath": "$",
      "ResultPath": "$"
    },
    "GetBuildDBStatus": {
      "Type": "Task",
      "Next": "CheckBuildDBStatus",
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
```

```
      "InputPath": "$",
      "ResultPath": "$.status"
    },
    "CheckBuildDBStatus": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.status",
          "StringEquals": "FAILED",
          "Next": "FailBuildDB"
        },
        {
          "Variable": "$.status",
          "StringEquals": "FINISHED",
          "Next": "InitializeBaselineData"
        }
      ],
      "Default": "BuildDBWait"
    },
    "BuildDBWait": {
      "Type": "Wait",
      "Seconds": 15,
      "Next": "GetBuildDBStatus"
    },
    "FailBuildDB": {
      "Type": "Fail",
      "Cause": "Database Build Failed",
      "Error": "Error"
    },
    "InitializeBaselineData": {
      "Type": "Pass",
      "Next": "LoadBaselineData",
      "Result": {
        "input": {
          "redshift_cluster_id": "cfn36-redshiftcluster-AKIAI44QH8DHBEXAMPLE",
          "redshift_database": "dev",
          "redshift_user": "awsuser",
          "redshift_schema": "tpcds",
          "action": "load_baseline_data",
          "sql_statement": [
            "begin transaction;",
            "truncate table {0}.customer;",
            "insert into {0}.customer
  (c_customer_sk,c_customer_id,c_current_addr_sk,c_first_name,c_last_name)",
```

```
            "values",
            "(7550,'AAAAAAAAOHNBAAAA',9264662,'Michelle','Deaton'),",
            "(37079,'AAAAAAAAHNAJAAAA',13971208,'Michael','Simms'),",
            "(40626,'AAAAAAAACLOJAAAA',1959255,'Susan','Ryder'),",
            "(2142876,'AAAAAAAAMJCLACAA',7644556,'Justin','Brown');",
            "analyze {0}.customer;",
            "--",
            "truncate table {0}.customer_address;",
            "insert into {0}.customer_address
(ca_address_sk,ca_address_id,ca_state,ca_zip,ca_country)",
            "values",
            "(13971208,'AAAAAAAAIAPCFNAA','NE','63451','United States'),",
            "(7644556,'AAAAAAAAMIFKEHAA','SD','58883','United States'),",
            "(9264662,'AAAAAAAAGBOFNIAA','CA','99310','United States');",
            "analyze {0}.customer_address;",
            "--",
            "truncate table {0}.item;",
            "insert into {0}.item
(i_item_sk,i_item_id,i_rec_start_date,i_rec_end_date,i_current_price,i_category,i_product_name
            "values",

"(3417,'AAAAAAAAIFNAAAAA','1997-10-27',NULL,14.29,'Electronics','ationoughtesepri
'),",
            "(9615,'AAAAAAAAOIFCAAAA','1997-10-27',NULL,9.68,'Home','antioughtcallyn
st'),",
            "(3693,'AAAAAAAAMGOAAAAA','2001-03-12',NULL,2.10,'Men','prin
stcallypri'),",

"(3630,'AAAAAAAAMCOAAAAA','2001-10-27',NULL,2.95,'Electronics','barpricallypri'),",

"(16506,'AAAAAAAAIHAEAAAA','2001-10-27',NULL,3.85,'Home','callybaranticallyought'),",

"(7866,'AAAAAAAAILOBAAAA','2001-10-27',NULL,12.60,'Jewelry','callycallyeingation');",
            "--",
            "analyze {0}.item;",
            "truncate table {0}.date_dim;",
            "insert into {0}.date_dim (d_date_sk,d_date_id,d_date,d_day_name)",
            "values",
            "(2450521,'AAAAAAAAJFEGFCAA','1997-03-13','Thursday'),",
            "(2450749,'AAAAAAAANDFGFCAA','1997-10-27','Monday'),",
            "(2451251,'AAAAAAAADDHGFCAA','1999-03-13','Saturday'),",
            "(2451252,'AAAAAAAAEDHGFCAA','1999-03-14','Sunday'),",
            "(2451981,'AAAAAAAANAKGFCAA','2001-03-12','Monday'),",
            "(2451982,'AAAAAAAAOAKGFCAA','2001-03-13','Tuesday'),",
```

```
            "(2452210,'AAAAAAAACPKGFCAA','2001-10-27','Saturday'),",
            "(2452641,'AAAAAAAABKMGFCAA','2003-01-01','Wednesday'),",
            "(2452642,'AAAAAAAACKMGFCAA','2003-01-02','Thursday');",
            "--",
            "analyze {0}.date_dim;",
            "--  commit and End transaction",
            "commit;",
            "end transaction;"
          ]
        }
      }
    },
    "LoadBaselineData": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
      "Next": "GetBaselineData",
      "InputPath": "$",
      "ResultPath": "$"
    },
    "GetBaselineData": {
      "Type": "Task",
      "Next": "CheckBaselineData",
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
      "InputPath": "$",
      "ResultPath": "$.status"
    },
    "CheckBaselineData": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.status",
          "StringEquals": "FAILED",
          "Next": "FailLoadBaselineData"
        },
        {
          "Variable": "$.status",
          "StringEquals": "FINISHED",
          "Next": "ParallelizeDimensionLoadJob"
```

```
        }
      ],
      "Default": "BaselineDataWait"
    },
    "BaselineDataWait": {
      "Type": "Wait",
      "Seconds": 20,
      "Next": "GetBaselineData"
    },
    "FailLoadBaselineData": {
      "Type": "Fail",
      "Cause": "Load Baseline Data Failed",
      "Error": "Error"
    },
    "ParallelizeDimensionLoadJob": {
      "Type": "Parallel",
      "Next": "InitializeSalesFactLoadJob",
      "ResultPath": "$.status",
      "Branches": [
        {
          "StartAt": "InitializeCustomerAddressDimensionLoadJob",
          "States": {
            "InitializeCustomerAddressDimensionLoadJob": {
              "Type": "Pass",
              "Next": "ExecuteCustomerAddressDimensionLoadJob",
              "Result": {
                "input": {
                  "redshift_cluster_id": "cfn36-redshiftcluster-AKIAI44QH8DHBEXAMPLE",
                  "redshift_database": "dev",
                  "redshift_user": "awsuser",
                  "redshift_schema": "tpcds",
                  "action": "load_customer_address",
                  "sql_statement": [
                    "begin transaction;",
                    "/* Create a staging table to hold the input data. Staging table is
created with BACKUP NO option for faster inserts and also data temporary */",
                    "drop table if exists {0}.stg_customer_address;",
                    "create table if not exists {0}.stg_customer_address",
                    "(ca_address_id    varchar(16)  encode zstd",
                    ",ca_state         varchar(2)   encode zstd",
                    ",ca_zip           varchar(10)  encode zstd",
                    ",ca_country       varchar(20)  encode zstd",
                    ")",
                    "backup no",
```

```
                    "diststyle even;",
                    "/* Ingest data from source */",
                    "insert into {0}.stg_customer_address
(ca_address_id,ca_state,ca_zip,ca_country)",
                    "values",
                    "('AAAAAAAACFBBAAAA','NE','','United States'),",
                    "('AAAAAAAAGAEFAAAA','NE','61749','United States'),",
                    "('AAAAAAAAPJKKAAAA','OK','','United States'),",
                    "('AAAAAAAAMIHGAAAA','AL','','United States');",
                    "/* Perform UPDATE for existing data with refreshed attribute
values */",
                    "update {0}.customer_address",
                    "   set ca_state = stg_customer_address.ca_state,",
                    "       ca_zip = stg_customer_address.ca_zip,",
                    "       ca_country = stg_customer_address.ca_country",
                    "  from {0}.stg_customer_address",
                    " where customer_address.ca_address_id =
stg_customer_address.ca_address_id;",
                    "/* Perform insert for new rows  */",
                    "insert into {0}.customer_address",
                    "(ca_address_sk",
                    ",ca_address_id",
                    ",ca_state",
                    ",ca_zip",
                    ",ca_country",
                    ")",
                    "with max_customer_address_sk as",
                    "(select max(ca_address_sk) max_ca_address_sk",
                    "from {0}.customer_address)",
                    "select row_number() over (order by
stg_customer_address.ca_address_id) + max_customer_address_sk.max_ca_address_sk as
ca_address_sk",
                    ",stg_customer_address.ca_address_id",
                    ",stg_customer_address.ca_state",
                    ",stg_customer_address.ca_zip",
                    ",stg_customer_address.ca_country",
                    "from {0}.stg_customer_address,",
                    "max_customer_address_sk",
                    "where stg_customer_address.ca_address_id not in (select
customer_address.ca_address_id from {0}.customer_address);",
                    "/* Commit and End transaction */",
                    "commit;",
                    "end transaction;"
                ]
```

```
                }
              }
            },
            "ExecuteCustomerAddressDimensionLoadJob": {
              "Type": "Task",
              "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
              "TimeoutSeconds": 180,
              "HeartbeatSeconds": 60,
              "Next": "GetCustomerAddressDimensionLoadStatus",
              "InputPath": "$",
              "ResultPath": "$"
            },
            "GetCustomerAddressDimensionLoadStatus": {
              "Type": "Task",
              "Next": "CheckCustomerAddressDimensionLoadStatus",
              "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
              "TimeoutSeconds": 180,
              "HeartbeatSeconds": 60,
              "InputPath": "$",
              "ResultPath": "$.status"
            },
            "CheckCustomerAddressDimensionLoadStatus": {
              "Type": "Choice",
              "Choices": [
                {
                  "Variable": "$.status",
                  "StringEquals": "FAILED",
                  "Next": "FailCustomerAddressDimensionLoad"
                },
                {
                  "Variable": "$.status",
                  "StringEquals": "FINISHED",
                  "Next": "CompleteCustomerAddressDimensionLoad"
                }
              ],
              "Default": "CustomerAddressWait"
            },
            "CustomerAddressWait": {
              "Type": "Wait",
              "Seconds": 5,
              "Next": "GetCustomerAddressDimensionLoadStatus"
            },
```

```
            "CompleteCustomerAddressDimensionLoad": {
              "Type": "Task",
              "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
              "TimeoutSeconds": 180,
              "HeartbeatSeconds": 60,
              "End": true
            },
            "FailCustomerAddressDimensionLoad": {
              "Type": "Fail",
              "Cause": "ETL Workflow Failed",
              "Error": "Error"
            }
          }
        },
        {
          "StartAt": "InitializeItemDimensionLoadJob",
          "States": {
            "InitializeItemDimensionLoadJob": {
              "Type": "Pass",
              "Next": "ExecuteItemDimensionLoadJob",
              "Result": {
                "input": {
                  "redshift_cluster_id": "cfn36-redshiftcluster-AKIAI44QH8DHBEXAMPLE",
                  "redshift_database": "dev",
                  "redshift_user": "awsuser",
                  "redshift_schema": "tpcds",
                  "action": "load_item",
                  "sql_statement": [
                    "begin transaction;",
                    "/* Create a staging table to hold the input data. Staging table is
 created with BACKUP NO option for faster inserts and also data temporary */",
                    "drop table if exists {0}.stg_item;",
                    "create table if not exists {0}.stg_item",
                    "(i_item_id        varchar(16) encode zstd",
                    ",i_rec_start_date date encode zstd",
                    ",i_rec_end_date   date encode zstd",
                    ",i_current_price  numeric(7,2) encode zstd",
                    ",i_category       varchar(50) encode zstd",
                    ",i_product_name   varchar(50) encode zstd",
                    ")",
                    "backup no",
                    "diststyle even;",
                    "/* Ingest data from source */",
```

```
                    "insert into {0}.stg_item",

  "(i_item_id,i_rec_start_date,i_rec_end_date,i_current_price,i_category,i_product_name)",
                    "values",

  "('AAAAAAAAABJBAAAA','2000-10-27',NULL,4.10,'Books','ationoughtesecally'),",
                    "('AAAAAAAAOPKBAAAA','2001-10-27',NULL,4.22,'Books','ableoughtn
  stcally'),",
                    "('AAAAAAAAHGPAAAAA','1997-10-27',NULL,29.30,'Books','priesen
  stpri'),",

  "('AAAAAAAAICMAAAAA','2001-10-27',NULL,1.93,'Books','eseoughtoughtpri'),",

  "('AAAAAAAAGPGBAAAA','2001-10-27',NULL,9.96,'Books','bareingeinganti'),",
                    "('AAAAAAAANBEBAAAA','1997-10-27',NULL,2.25,'Music','n
  steseoughtanti'),",

  "('AAAAAAAACLAAAAAA','2001-10-27',NULL,1.71,'Home','bareingought'),",

  "('AAAAAAAAOBBDAAAA','2001-10-27',NULL,5.55,'Books','callyationantiableought');",
                    "/
**************************************************************************************
                    "** Type 2 is maintained for i_current_price column.",
                    "** Update all attributes for the item when the price is not
  changed",
                    "** Sunset existing active item record with current i_rec_end_date
  and insert a new record when the price does not match",

  "**************************************************************************************
                    "update {0}.item",
                    "   set i_category = stg_item.i_category,",
                    "       i_product_name = stg_item.i_product_name",
                    "  from {0}.stg_item",
                    " where item.i_item_id = stg_item.i_item_id",
                    "   and item.i_rec_end_date is null",
                    "   and item.i_current_price = stg_item.i_current_price;",
                    "insert into {0}.item",
                    "(i_item_sk",
                    ",i_item_id",
                    ",i_rec_start_date",
                    ",i_rec_end_date",
                    ",i_current_price",
                    ",i_category",
                    ",i_product_name",
```

```
                    ")",
                    "with max_item_sk as",
                    "(select max(i_item_sk) max_item_sk",
                    "   from {0}.item)",
                    "select row_number() over (order by stg_item.i_item_id) +
 max_item_sk as i_item_sk",
                    "      ,stg_item.i_item_id",
                    "      ,trunc(sysdate) as i_rec_start_date",
                    "      ,null as i_rec_end_date",
                    "      ,stg_item.i_current_price",
                    "      ,stg_item.i_category",
                    "      ,stg_item.i_product_name",
                    "  from {0}.stg_item, {0}.item, max_item_sk",
                    " where item.i_item_id = stg_item.i_item_id",
                    "   and item.i_rec_end_date is null",
                    "   and item.i_current_price <> stg_item.i_current_price;",
                    "/* Sunset penultimate records that were inserted as type 2 */",
                    "update {0}.item",
                    "   set i_rec_end_date = trunc(sysdate)",
                    "  from {0}.stg_item",
                    " where item.i_item_id = stg_item.i_item_id",
                    "   and item.i_rec_end_date is null",
                    "   and item.i_current_price <> stg_item.i_current_price;",
                    "/* Commit and End transaction */",
                    "commit;",
                    "end transaction;"
                  ]
                }
              }
            },
            "ExecuteItemDimensionLoadJob": {
              "Type": "Task",
              "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
              "TimeoutSeconds": 180,
              "HeartbeatSeconds": 60,
              "Next": "GetItemDimensionLoadStatus",
              "InputPath": "$",
              "ResultPath": "$"
            },
            "GetItemDimensionLoadStatus": {
              "Type": "Task",
              "Next": "CheckItemDimensionLoadStatus",
```

```
              "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
              "TimeoutSeconds": 180,
              "HeartbeatSeconds": 60,
              "InputPath": "$",
              "ResultPath": "$.status"
            },
            "CheckItemDimensionLoadStatus": {
              "Type": "Choice",
              "Choices": [
                {
                  "Variable": "$.status",
                  "StringEquals": "FAILED",
                  "Next": "FailItemDimensionLoad"
                },
                {
                  "Variable": "$.status",
                  "StringEquals": "FINISHED",
                  "Next": "CompleteItemDimensionLoad"
                }
              ],
              "Default": "ItemWait"
            },
            "ItemWait": {
              "Type": "Wait",
              "Seconds": 5,
              "Next": "GetItemDimensionLoadStatus"
            },
            "CompleteItemDimensionLoad": {
              "Type": "Task",
              "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
              "TimeoutSeconds": 180,
              "HeartbeatSeconds": 60,
              "End": true
            },
            "FailItemDimensionLoad": {
              "Type": "Fail",
              "Cause": "ETL Workflow Failed",
              "Error": "Error"
            }
          }
        }
      ]
```

```
    },
    "InitializeSalesFactLoadJob": {
      "Type": "Pass",
      "Next": "ExecuteSalesFactLoadJob",
      "Result": {
        "input": {
          "redshift_cluster_id": "cfn36-redshiftcluster-AKIAI44QH8DHBEXAMPLE",
          "redshift_database": "dev",
          "redshift_user": "awsuser",
          "redshift_schema": "tpcds",
          "snapshot_date": "2003-01-02",
          "action": "load_sales_fact",
          "sql_statement": [
            "begin transaction;",
            "/* Create a stg_store_sales staging table */",
            "drop table if exists {0}.stg_store_sales;",
            "create table {0}.stg_store_sales",
            "(sold_date             date encode zstd",
            ",i_item_id             varchar(16) encode zstd",
            ",c_customer_id         varchar(16) encode zstd",
            ",ca_address_id         varchar(16) encode zstd",
            ",ss_ticket_number      integer encode zstd",
            ",ss_quantity           integer encode zstd",
            ",ss_net_paid           numeric(7,2) encode zstd",
            ",ss_net_profit         numeric(7,2) encode zstd",
            ")",
            "backup no",
            "diststyle even;",
            "/* Ingest data from source */",
            "insert into {0}.stg_store_sales",

"(sold_date,i_item_id,c_customer_id,ca_address_id,ss_ticket_number,ss_quantity,ss_net_paid,ss_
            "values",

"('2003-01-02','AAAAAAAAIFNAAAAA','AAAAAAAAOHNBAAAA','AAAAAAAAGBOFNIAA',1403191,13,5046.37,150

"('2003-01-02','AAAAAAAAIFNAAAAA','AAAAAAAAOHNBAAAA','AAAAAAAAGBOFNIAA',1403191,13,2103.72,-12

"('2003-01-02','AAAAAAAAILOBAAAA','AAAAAAAAOHNBAAAA','AAAAAAAAGBOFNIAA',1403191,13,959.10,-130

"('2003-01-02','AAAAAAAAILOBAAAA','AAAAAAAAHNAJAAAA','AAAAAAAAIAPCFNAA',1403191,13,962.65,-475

"('2003-01-02','AAAAAAAAMCOAAAAA','AAAAAAAAHNAJAAAA','AAAAAAAAIAPCFNAA',1201746,17,111.60,-241
```

```
"('2003-01-02','AAAAAAAAMCOAAAAA','AAAAAAAAHNAJAAAA','AAAAAAAAIAPCFNAA',1201746,17,4013.02,-11

"('2003-01-02','AAAAAAAAMCOAAAAA','AAAAAAAAMJCLACAA','AAAAAAAAMIFKEHAA',1201746,17,2689.12,-55

"('2003-01-02','AAAAAAAAMGOAAAAA','AAAAAAAAMJCLACAA','AAAAAAAAMIFKEHAA',193971,18,1876.89,-556
          "/* Delete any rows from target store_sales for the input date for
idempotency */",
          "delete from {0}.store_sales where ss_sold_date_sk in (select d_date_sk
from {0}.date_dim where d_date='{1}');",
          "/* Insert data from staging table to the target table */",
          "insert into {0}.store_sales",
          "(ss_sold_date_sk",
          ",ss_item_sk",
          ",ss_customer_sk",
          ",ss_addr_sk",
          ",ss_ticket_number",
          ",ss_quantity",
          ",ss_net_paid",
          ",ss_net_profit",
          ")",
          "select date_dim.d_date_sk ss_sold_date_sk",
          "      ,item.i_item_sk ss_item_sk",
          "      ,customer.c_customer_sk ss_customer_sk",
          "      ,customer_address.ca_address_sk ss_addr_sk",
          "      ,ss_ticket_number",
          "      ,ss_quantity",
          "      ,ss_net_paid",
          "      ,ss_net_profit",
          "  from {0}.stg_store_sales as store_sales",
          "  inner join {0}.date_dim on store_sales.sold_date = date_dim.d_date",
          "  left join {0}.item on store_sales.i_item_id = item.i_item_id and
item.i_rec_end_date is null",
          "  left join {0}.customer on store_sales.c_customer_id =
customer.c_customer_id",
          "  left join {0}.customer_address on store_sales.ca_address_id =
customer_address.ca_address_id;",
          "/* Drop staging table */",
          "drop table if exists {0}.stg_store_sales;",
          "/* Commit and End transaction */",
          "commit;",
          "end transaction;"
        ]
      }
```

```
      }
    },
    "ExecuteSalesFactLoadJob": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
      "Next": "GetSalesFactLoadStatus",
      "InputPath": "$",
      "ResultPath": "$"
    },
    "GetSalesFactLoadStatus": {
      "Type": "Task",
      "Next": "CheckSalesFactLoadStatus",
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
      "InputPath": "$",
      "ResultPath": "$.status"
    },
    "CheckSalesFactLoadStatus": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.status",
          "StringEquals": "FAILED",
          "Next": "FailSalesFactLoad"
        },
        {
          "Variable": "$.status",
          "StringEquals": "FINISHED",
          "Next": "SalesETLPipelineComplete"
        }
      ],
      "Default": "SalesWait"
    },
    "SalesWait": {
      "Type": "Wait",
      "Seconds": 5,
      "Next": "GetSalesFactLoadStatus"
    },
    "FailSalesFactLoad": {
```

```
      "Type": "Fail",
      "Cause": "ETL Workflow Failed",
      "Error": "Error"
    },
    "ClusterUnavailable": {
      "Type": "Fail",
      "Cause": "Redshift cluster is not available",
      "Error": "Error"
    },
    "SalesETLPipelineComplete": {
      "Type": "Pass",
      "Next": "ValidateSalesMetric",
      "Result": {
        "input": {
          "redshift_cluster_id": "cfn36-redshiftcluster-AKIAI44QH8DHBEXAMPLE",
          "redshift_database": "dev",
          "redshift_user": "awsuser",
          "redshift_schema": "tpcds",
          "snapshot_date": "2003-01-02",
          "action": "validate_sales_metric",
          "sql_statement": [
            "select 1/count(1) from {0}.store_sales where ss_sold_date_sk in (select
 d_date_sk from {0}.date_dim where d_date='{1}')"
          ]
        }
      }
    },
    "ValidateSalesMetric": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
      "Next": "GetValidateSalesMetricStatus",
      "InputPath": "$",
      "ResultPath": "$"
    },
    "GetValidateSalesMetricStatus": {
      "Type": "Task",
      "Next": "CheckValidateSalesMetricStatus",
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
```

```
      "InputPath": "$",
      "ResultPath": "$.status"
    },
    "CheckValidateSalesMetricStatus": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.status",
          "StringEquals": "FAILED",
          "Next": "FailSalesMetricValidation"
        },
        {
          "Variable": "$.status",
          "StringEquals": "FINISHED",
          "Next": "DataValidationComplete"
        }
      ],
      "Default": "SalesValidationWait"
    },
    "SalesValidationWait": {
      "Type": "Wait",
      "Seconds": 5,
      "Next": "GetValidateSalesMetricStatus"
    },
    "FailSalesMetricValidation": {
      "Type": "Fail",
      "Cause": "Data Validation Failed",
      "Error": "Error"
    },
    "DataValidationComplete": {
      "Type": "Pass",
      "Next": "InitializePauseCluster"
    },
    "InitializePauseCluster": {
      "Type": "Pass",
      "Next": "PauseCluster",
      "Result": {
        "input": {
          "redshift_cluster_id": "cfn36-redshiftcluster-AKIAI44QH8DHBEXAMPLE",
          "operation": "pause"
        }
      }
    },
    "PauseCluster": {
```

```
      "Type": "Task",
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftOperations-AKIAIOSFODNN7EXAMPLE",
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
      "Next": "PauseClusterWait",
      "InputPath": "$",
      "ResultPath": "$.clusterStatus",
      "Catch": [
        {
          "ErrorEquals": [
            "States.ALL"
          ],
          "Next": "ClusterPausedComplete"
        }
      ]
    },
    "InitializeCheckPauseCluster": {
      "Type": "Pass",
      "Next": "GetStateOfPausedCluster",
      "Result": {
        "input": {
          "redshift_cluster_id": "cfn36-redshiftcluster-AKIAI44QH8DHBEXAMPLE",
          "operation": "status"
        }
      }
    },
    "GetStateOfPausedCluster": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftOperations-AKIAIOSFODNN7EXAMPLE",
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
      "Next": "IsClusterPaused",
      "InputPath": "$",
      "ResultPath": "$.clusterStatus"
    },
    "IsClusterPaused": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.clusterStatus",
          "StringEquals": "available",
          "Next": "InitializePauseCluster"
```

```
        },
        {
          "Variable": "$.clusterStatus",
          "StringEquals": "paused",
          "Next": "ClusterPausedComplete"
        },
        {
          "Variable": "$.clusterStatus",
          "StringEquals": "unavailable",
          "Next": "ClusterUnavailable"
        },
        {
          "Variable": "$.clusterStatus",
          "StringEquals": "resuming",
          "Next": "PauseClusterWait"
        }
      ]
    },
    "PauseClusterWait": {
      "Type": "Wait",
      "Seconds": 720,
      "Next": "InitializeCheckPauseCluster"
    },
    "ClusterPausedComplete": {
      "Type": "Pass",
      "End": true
    }
  }
}
```

Informationen zur Konfiguration von IAM bei der Verwendung von Step Functions mit anderen AWS Diensten finden Sie unterIAM-Richtlinien für integrierte Dienste.

IAM-Beispiel

Diese vom Beispielprojekt generierten Beispielrichtlinien AWS Identity and Access Management (IAM) beinhalten die geringsten Rechte, die für die Ausführung der Zustandsmaschine und der zugehörigen Ressourcen erforderlich sind. Wir empfehlen, dass Sie nur die Berechtigungen in Ihre IAM-Richtlinien aufnehmen, die erforderlich sind.

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

Informationen zur Konfiguration von IAM bei der Verwendung von Step Functions mit anderen AWS Diensten finden Sie unterIAM-Richtlinien für integrierte Dienste.

Verwenden Step Functions und AWS Batch mit Fehlerbehandlung

Dieses Beispielprojekt zeigt, wie ein AWS Batch Job mithilfe eines Zustandsautomaten mit Funktionen Step Functions zur Fehlerbehandlung ausgeführt wird.

In diesem Projekt verwendet Step Functions einen Zustandsautomaten, um den AWS Batch-Auftrag synchron aufzurufen. Es wartet dann, bis der Job erfolgreich ist oder nicht, versucht es erneut und fängt Fehler ab, wenn ein Job fehlschlägt, und sendet dann ein Amazon SNS Thema mit einer Meldung darüber, ob der Job erfolgreich war oder nicht.

Schritt 1: Erstellen Sie den Zustandsmaschine und stellen Sie Ressourcen bereit

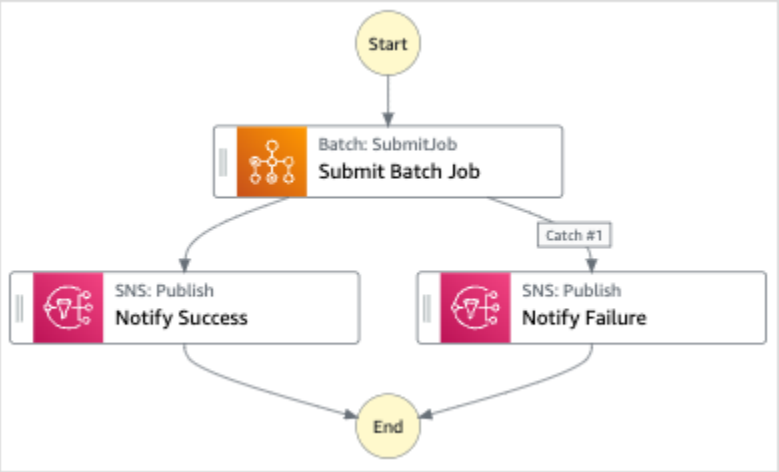
1. Öffnen Sie die Step Functions Functions-Konsole und wählen Sie Create State Machine.
2. Geben Sie **`Manage a batch job`** etwas in das Suchfeld ein und wählen Sie dann aus den zurückgegebenen Suchergebnissen die Option Batch-Job verwalten aus.
3. Wählen Sie Next (Weiter), um fortzufahren.
4. Step Functions listet die Funktionen auf, die in dem von Ihnen ausgewählten Beispielprojekt AWS-Services verwendet wurden. Außerdem wird ein Workflow-Diagramm für das Beispielprojekt angezeigt. Stellen Sie dieses Projekt für Ihr Projekt bereit AWS-Konto oder

verwenden Sie es als Ausgangspunkt für die Erstellung Ihrer eigenen Projekte. Je nachdem, wie Sie vorgehen möchten, wählen Sie „Demo ausführen“ oder „Darauf aufbauen“.

In diesem Beispielprojekt werden die folgenden Ressourcen bereitgestellt:

- Ein Job AWS Batch
- Amazon-SNS-Thema
- Eine AWS Step Functions Staatsmaschine
- Verwandte AWS Identity and Access Management Rollen (IAM)

Die folgende Abbildung zeigt das Workflow-Diagramm für das Beispielprojekt „Einen Batch-Job verwalten“:



5. Wählen Sie Vorlage verwenden, um mit Ihrer Auswahl fortzufahren.
6. Führen Sie eine der folgenden Aktionen aus:

   - Wenn Sie „Darauf aufbauen“ ausgewählt haben, erstellt Step Functions den Workflow-Prototyp für das von Ihnen ausgewählte Beispielprojekt. Step Functions stellt die in der Workflow-Definition aufgeführten Ressourcen nicht bereit.

     Ziehen Sie in Workflow Studio Status per Drag-and-Drop aus demEntwurfsmodus, Bundesstaaten-Browser um mit der Erstellung Ihres Workflow-Prototyps fortzufahren. Oder wechseln Sie zu demCodemodus, der einen integrierten Code-Editor bietet, der VS Code ähnelt, um die Amazon States Language (ASL-) Definition Ihrer Zustandsmaschine in der Step Functions Functions-Konsole zu aktualisieren. Weitere Informationen zur Verwendung von Workflow Studio zum Erstellen Ihrer Zustandsmaschinen finden Sie unterVerwenden von Workflow Studio.

> **Important**
>
> Denken Sie daran, den Platzhalter Amazon Resource Name (ARN) für die im Beispielprojekt verwendeten Ressourcen zu aktualisieren, bevor Sie Ihren Workflow ausführen.

- Wenn Sie Eine Demo ausführen ausgewählt haben, erstellt Step Functions ein schreibgeschütztes Beispielprojekt, das eine AWS CloudFormation Vorlage verwendet, um die in dieser Vorlage aufgeführten AWS Ressourcen für Ihr Projekt bereitzustellen. AWS-Konto

  > **Tip**
  >
  > Um die State-Machine-Definition des Beispielprojekts anzuzeigen, wählen Sie Code.

  Wenn Sie bereit sind, wählen Sie Deploy and run aus, um das Beispielprojekt bereitzustellen und die Ressourcen zu erstellen.

  Es kann bis zu 10 Minuten dauern, bis diese Ressourcen und die zugehörigen IAM-Berechtigungen erstellt sind. Während der Bereitstellung Ihrer Ressourcen können Sie den CloudFormation Stack-ID-Link öffnen, um zu sehen, welche Ressourcen bereitgestellt werden.

  Nachdem alle Ressourcen im Beispielprojekt erstellt wurden, wird das neue Beispielprojekt auf der Seite State Machines aufgeführt.

  > **Important**
  >
  > Für jeden in der CloudFormation Vorlage verwendeten Dienst können Standardgebühren anfallen.

Schritt 2: Führen Sie die Zustandsmaschine aus

1. Wählen Sie auf der Seite State Machines Ihr Beispielprojekt aus.
2. Wählen Sie auf der Seite mit dem Beispielprojekt die Option Ausführung starten aus.
3. Gehen Sie im Dialogfeld Ausführung starten wie folgt vor:

1. (Optional) Um Ihre Ausführung zu identifizieren, können Sie im Feld Name einen Namen dafür angeben. Standardmäßig generiert Step Functions automatisch einen eindeutigen Ausführungsnamen.

   > **Note**
   >
   > Mit Step Functions können Sie Namen für Zustandsmaschinen, Ausführungen und Aktivitäten sowie Beschriftungen erstellen, die Nicht-ASCII-Zeichen enthalten. Diese Nicht-ASCII-Namen funktionieren nicht mit Amazon. CloudWatch Um sicherzustellen, dass Sie CloudWatch Messwerte verfolgen können, wählen Sie einen Namen, der nur ASCII-Zeichen verwendet.

2. (Optional) Geben Sie in das Eingabefeld Eingabewerte im JSON-Format ein, um Ihren Workflow auszuführen.

   Wenn Sie sich dafür entschieden haben, eine Demo auszuführen, müssen Sie keine Ausführungseingaben angeben.

   > **Note**
   >
   > Wenn das von Ihnen bereitgestellte Demo-Projekt vorab aufgefüllte Eingabedaten für die Ausführung enthält, verwenden Sie diese Eingabe, um die Zustandsmaschine auszuführen.

3. Wählen Sie Start execution (Ausführung starten) aus.
4. Die Step Functions Functions-Konsole leitet Sie zu einer Seite weiter, die mit Ihrer Ausführungs-ID betitelt ist. Diese Seite wird als Seite mit den Ausführungsdetails bezeichnet. Auf dieser Seite können Sie die Ausführungsergebnisse im Verlauf oder nach Abschluss der Ausführung überprüfen.

   Um die Ausführungsergebnisse zu überprüfen, wählen Sie in der Diagrammansicht einzelne Status aus und wählen Sie dann die einzelnen Registerkarten im Schrittdetails Bereich, um die Details der einzelnen Status, einschließlich Eingabe, Ausgabe und Definition, anzuzeigen. Einzelheiten zu den Ausführungsinformationen, die Sie auf der Seite mit den Ausführungsdetails einsehen können, finden Sie unterSeite mit Ausführungsdetails – Schnittstellenübersicht.

Code des Zustandsautomaten aus diesem Beispiel

Die Zustandsmaschine in diesem Beispielprojekt lässt sich AWS Batch in Amazon SNS integrieren, indem Parameter direkt an diese Ressourcen übergeben werden.

Sehen Sie sich diese Beispiel-Zustandsmaschine an, um zu sehen, wie Step Functions AWS Batch und Amazon SNS steuern, indem Sie eine Verbindung zum Amazon-Ressourcennamen (ARN) im `Resource` Feld herstellen und `Parameters` an die Service-API übergeben.

Weitere Informationen darüber, wie AWS Step Functions Sie andere AWS Dienste steuern können, finden Sie unterVerwendung AWS Step Functions mit anderen Diensten.

```
{
  "Comment": "An example of the Amazon States Language for notification on an AWS Batch
 job completion",
  "StartAt": "Submit Batch Job",
  "TimeoutSeconds": 3600,
  "States": {
    "Submit Batch Job": {
      "Type": "Task",
      "Resource": "arn:aws:states:::batch:submitJob.sync",
      "Parameters": {
        "JobName": "BatchJobNotification",
        "JobQueue": "arn:aws:batch:us-west-2:123456789012:job-queue/
BatchJobQueue-123456789abcdef",
        "JobDefinition": "arn:aws:batch:us-west-2:123456789012:job-definition/
BatchJobDefinition-123456789abcdef:1"
      },
      "Next": "Notify Success",
      "Retry": [
        {
          "ErrorEquals": [
            "States.ALL"
          ],
          "IntervalSeconds": 30,
          "MaxAttempts": 2,
          "BackoffRate": 1.5
        }
      ],
      "Catch": [
        {
          "ErrorEquals": [ "States.ALL" ],
          "Next": "Notify Failure"
```

```
        }
      ]
    },
    "Notify Success": {
      "Type": "Task",
      "Resource": "arn:aws:states:::sns:publish",
      "Parameters": {
        "Message": "Batch job submitted through Step Functions succeeded",
        "TopicArn": "arn:aws:sns:us-west-2:123456789012:StepFunctionsSample-
BatchJobManagement12345678-9abc-def0-1234-567890abcdef-SNSTopic-A2B3C4D5E6F7G"
      },
      "End": true
    },
    "Notify Failure": {
      "Type": "Task",
      "Resource": "arn:aws:states:::sns:publish",
      "Parameters": {
        "Message": "Batch job submitted through Step Functions failed",
        "TopicArn": "arn:aws:sns:us-west-2:123456789012:StepFunctionsSample-
BatchJobManagement12345678-9abc-def0-1234-567890abcdef-SNSTopic-A2B3C4D5E6F7G"
      },
      "End": true
    }
  }
}
```

IAM-Beispiel

Diese vom Beispielprojekt generierte Beispielrichtlinie AWS Identity and Access Management (IAM) beinhaltet die geringsten Rechte, die für die Ausführung der Zustandsmaschine und der zugehörigen Ressourcen erforderlich sind. Wir empfehlen, dass Sie nur die Berechtigungen in Ihre IAM-Richtlinien aufnehmen, die erforderlich sind.

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"
        }
    ]
}
```

Informationen zur Konfiguration von IAM bei der Verwendung von Step Functions mit anderen AWS Diensten finden Sie unterIAM-Richtlinien für integrierte Dienste.

Einen AWS Batch Job ausfindig machen

Dieses Beispielprojekt zeigt, wie der Zuordnung Status von Step Functions verwendet werden kann, um AWS Batch Jobs aufzufächern.

In diesem Projekt verwendet Step Functions eine Zustandsmaschine, um eine Lambda-Funktion aufzurufen, um eine einfache Vorverarbeitung durchzuführen, und ruft dann mehrere AWS Batch Jobs parallel unter Verwendung des Status auf. Zuordnung

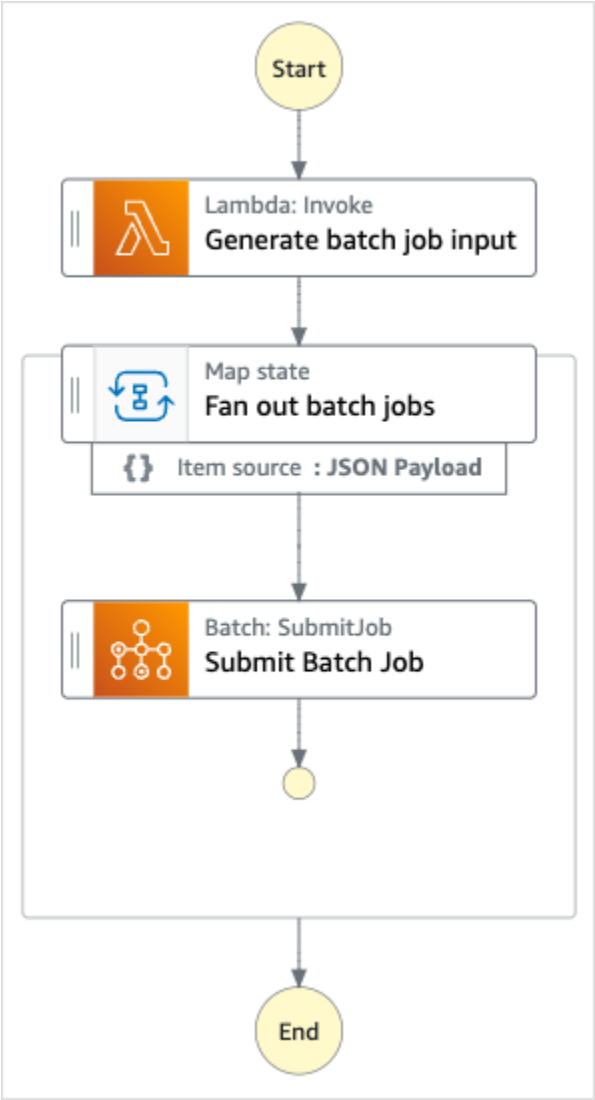
Schritt 1: Erstellen Sie die Zustandsmaschine und stellen Sie Ressourcen bereit

1. Öffnen Sie die Step Functions Functions-Konsole und wählen Sie Create State Machine.
2. Geben Sie **`Fan out a batch job`** etwas in das Suchfeld ein und wählen Sie dann aus den zurückgegebenen Suchergebnissen die Option Batch-Job auffächern aus.
3. Wählen Sie Next (Weiter), um fortzufahren.
4. Step Functions listet die Funktionen auf, die in dem von Ihnen ausgewählten Beispielprojekt AWS-Services verwendet wurden. Außerdem wird ein Workflow-Diagramm für das Beispielprojekt angezeigt. Stellen Sie dieses Projekt für Sie bereit AWS-Konto oder verwenden Sie es als Ausgangspunkt für die Erstellung Ihrer eigenen Projekte. Je nachdem, wie Sie vorgehen möchten, wählen Sie „Demo ausführen“ oder „Darauf aufbauen“.

   Dieses Beispielprojekt stellt die folgenden Ressourcen bereit:

   - Eine Lambda-Funktion
   - Eine AWS Batch Auftragswarteschlange
   - Eine AWS Step Functions Zustandsmaschine
   - Verwandte AWS Identity and Access Management Rollen (IAM)

   Die folgende Abbildung zeigt das Workflow-Diagramm für das Beispielprojekt Fan out a batch:



5. Wählen Sie Vorlage verwenden, um mit Ihrer Auswahl fortzufahren.
6. Führen Sie eine der folgenden Aktionen aus:

   - Wenn Sie „Darauf aufbauen“ ausgewählt haben, erstellt Step Functions den Workflow-Prototyp für das von Ihnen ausgewählte Beispielprojekt. Step Functions stellt die in der Workflow-Definition aufgeführten Ressourcen nicht bereit.

     Ziehen Sie in Workflow Studio Status per Drag-and-Drop aus demEntwurfsmodus, Bundesstaaten-Browser um mit der Erstellung Ihres Workflow-Prototyps fortzufahren. Oder wechseln Sie zu demCodemodus, der einen integrierten Code-Editor bietet, der VS Code ähnelt, um die Amazon States Language (ASL-) Definition Ihrer Zustandsmaschine in der Step Functions Functions-Konsole zu aktualisieren. Weitere Informationen zur Verwendung

von Workflow Studio zum Erstellen Ihrer Zustandsmaschinen finden Sie unterVerwenden von Workflow Studio.

> **Important**
>
> Denken Sie daran, den Platzhalter Amazon Resource Name (ARN) für die im Beispielprojekt verwendeten Ressourcen zu aktualisieren, bevor Sie Ihren Workflow ausführen.

- Wenn Sie Eine Demo ausführen ausgewählt haben, erstellt Step Functions ein schreibgeschütztes Beispielprojekt, das eine AWS CloudFormation Vorlage verwendet, um die in dieser Vorlage aufgeführten AWS Ressourcen für Ihr Projekt bereitzustellen. AWS-Konto

> **Tip**
>
> Um die State-Machine-Definition des Beispielprojekts anzuzeigen, wählen Sie Code.

Wenn Sie bereit sind, wählen Sie Deploy and run aus, um das Beispielprojekt bereitzustellen und die Ressourcen zu erstellen.

Es kann bis zu 10 Minuten dauern, bis diese Ressourcen und die zugehörigen IAM-Berechtigungen erstellt sind. Während der Bereitstellung Ihrer Ressourcen können Sie den CloudFormation Stack-ID-Link öffnen, um zu sehen, welche Ressourcen bereitgestellt werden.

Nachdem alle Ressourcen im Beispielprojekt erstellt wurden, wird das neue Beispielprojekt auf der Seite State Machines aufgeführt.

> **Important**
>
> Für jeden in der CloudFormation Vorlage verwendeten Dienst können Standardgebühren anfallen.

Schritt 2: Führen Sie die Zustandsmaschine aus

1. Wählen Sie auf der Seite State Machines Ihr Beispielprojekt aus.
2. Wählen Sie auf der Seite mit dem Beispielprojekt die Option Ausführung starten aus.

3. Gehen Sie im Dialogfeld Ausführung starten wie folgt vor:

   1. (Optional) Um Ihre Ausführung zu identifizieren, können Sie im Feld Name einen Namen dafür angeben. Standardmäßig generiert Step Functions automatisch einen eindeutigen Ausführungsnamen.

      > **Note**
      >
      > Mit Step Functions können Sie Namen für Zustandsmaschinen, Ausführungen und Aktivitäten sowie Beschriftungen erstellen, die Nicht-ASCII-Zeichen enthalten. Diese Nicht-ASCII-Namen funktionieren nicht mit Amazon. CloudWatch Um sicherzustellen, dass Sie CloudWatch Messwerte verfolgen können, wählen Sie einen Namen, der nur ASCII-Zeichen verwendet.

   2. (Optional) Geben Sie in das Eingabefeld Eingabewerte im JSON-Format ein, um Ihren Workflow auszuführen.

      Wenn Sie sich dafür entschieden haben, eine Demo auszuführen, müssen Sie keine Ausführungseingaben angeben.

      > **Note**
      >
      > Wenn das von Ihnen bereitgestellte Demo-Projekt vorab aufgefüllte Eingabedaten für die Ausführung enthält, verwenden Sie diese Eingabe, um die Zustandsmaschine auszuführen.

   3. Wählen Sie Start execution (Ausführung starten) aus.
   4. Die Step Functions Functions-Konsole leitet Sie zu einer Seite weiter, die mit Ihrer Ausführungs-ID betitelt ist. Diese Seite wird als Seite mit den Ausführungsdetails bezeichnet. Auf dieser Seite können Sie die Ausführungsergebnisse im Verlauf oder nach Abschluss der Ausführung überprüfen.

      Um die Ausführungsergebnisse zu überprüfen, wählen Sie in der Diagrammansicht einzelne Status aus und wählen Sie dann die einzelnen Registerkarten im Schrittdetails Bereich, um die Details der einzelnen Status, einschließlich Eingabe, Ausgabe und Definition, anzuzeigen. Einzelheiten zu den Ausführungsinformationen, die Sie auf der Seite mit den Ausführungsdetails einsehen können, finden Sie unterSeite mit Ausführungsdetails – Schnittstellenübersicht.

Code des Zustandsautomaten aus diesem Beispiel

Die Zustandsmaschine in diesem Beispielprojekt lässt sich AWS Batch in Amazon SNS integrieren, indem Parameter direkt an diese Ressourcen übergeben werden.

Sehen Sie sich diese Beispiel-Zustandsmaschine an, um zu sehen, wie Step Functions AWS Batch und Amazon SNS steuern, indem Sie eine Verbindung zum Amazon-Ressourcennamen (ARN) im `Resource` Feld herstellen und `Parameters` an die Service-API übergeben.

Weitere Informationen darüber, wie AWS Step Functions Sie andere AWS Dienste steuern können, finden Sie unterVerwendung AWS Step Functions mit anderen Diensten.

```
{
  "Comment": "An example of the Amazon States Language for fanning out AWS Batch job",
  "StartAt": "Generate batch job input",
  "TimeoutSeconds": 3600,
  "States": {
    "Generate batch job input": {
      "Type": "Task",
      "Resource": "arn:aws:states:::lambda:invoke",
      "OutputPath": "$.Payload",
      "Parameters": {
        "FunctionName": "<GENERATE_BATCH_JOB_INPUT_LAMBDA_FUNCTION_NAME>"
      },
      "Next": "Fan out batch jobs"
    },
    "Fan out batch jobs": {
      "Comment": "Start multiple executions of batch job depending on pre-processed
 data",
      "Type": "Map",
      "End": true,
      "ItemsPath": "$",
      "Parameters": {
        "BatchNumber.$": "$$.Map.Item.Value"
      },
      "Iterator": {
        "StartAt": "Submit Batch Job",
        "States": {
          "Submit Batch Job": {
            "Type": "Task",
            "Resource": "arn:aws:states:::batch:submitJob.sync",
            "Parameters": {
              "JobName": "BatchJobFanOut",
```

```
              "JobQueue": "<BATCH_QUEUE_ARN>",
              "JobDefinition": "<BATCH_JOB_DEFINITION_ARN>"
            },
            "End": true
          }
        }
      }
    }
  }
}
```

IAM-Beispiel

Diese vom Beispielprojekt generierten Beispielrichtlinien AWS Identity and Access Management (IAM) beinhalten die geringsten Rechte, die für die Ausführung der Zustandsmaschine und der zugehörigen Ressourcen erforderlich sind. Wir empfehlen, dass Sie nur die Berechtigungen in Ihre IAM-Richtlinien aufnehmen, die erforderlich sind.

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"
        }
    ]
}
```

Informationen zur Konfiguration von IAM bei der Verwendung von Step Functions mit anderen AWS Diensten finden Sie unterIAM-Richtlinien für integrierte Dienste.

AWS Batch mit Lambda

Dieses Beispielprojekt zeigt, wie Step Functions verwendet werden kann, um Daten mit AWS Lambda Funktionen vorzuverarbeiten und anschließend Jobs zu orchestrieren AWS Batch .

In diesem Projekt verwendet Step Functions eine Zustandsmaschine, um eine Lambda-Funktion aufzurufen, um eine einfache Vorverarbeitung durchzuführen, bevor ein AWS Batch Job eingereicht wird. Je nach Ergebnis oder Erfolg des vorherigen Jobs können mehrere Jobs aufgerufen werden.

Schritt 1: Erstellen Sie den Zustandsmaschine und stellen Sie Ressourcen bereit

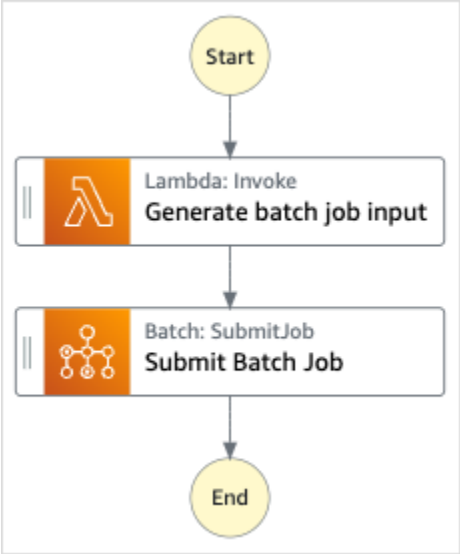
1. Öffnen Sie die Step Functions Functions-Konsole und wählen Sie Create State Machine.
2. Geben Sie **`Batch job with Lambda`** etwas in das Suchfeld ein, und wählen Sie dann Batch-Job mit Lambda aus den zurückgegebenen Suchergebnissen aus.
3. Wählen Sie Next (Weiter), um fortzufahren.

4. Step Functions listet die Funktionen auf, die in dem von Ihnen ausgewählten Beispielprojekt AWS-Services verwendet wurden. Außerdem wird ein Workflow-Diagramm für das Beispielprojekt angezeigt. Stellen Sie dieses Projekt für Ihr Projekt bereit AWS-Konto oder verwenden Sie es als Ausgangspunkt für die Erstellung Ihrer eigenen Projekte. Je nachdem, wie Sie vorgehen möchten, wählen Sie „Demo ausführen“ oder „Darauf aufbauen“.

   In diesem Beispielprojekt werden die folgenden Ressourcen bereitgestellt:

   - Eine Lambda-Funktion
   - Ein AWS Batch-Auftrag
   - Eine AWS Step Functions Zustandsmaschine
   - Verwandte AWS Identity and Access Management Rollen (IAM)

   Die folgende Abbildung zeigt das Workflow-Diagramm für den Batch-Job mit dem Lambda Beispielprojekt:



5. Wählen Sie Vorlage verwenden, um mit Ihrer Auswahl fortzufahren.
6. Führen Sie eine der folgenden Aktionen aus:

   - Wenn Sie „Darauf aufbauen“ ausgewählt haben, erstellt Step Functions den Workflow-Prototyp für das von Ihnen ausgewählte Beispielprojekt. Step Functions stellt die in der Workflow-Definition aufgeführten Ressourcen nicht bereit.

     Ziehen Sie in Workflow Studio Status per Drag-and-Drop aus demEntwurfsmodus, Bundesstaaten-Browser um mit der Erstellung Ihres Workflow-Prototyps fortzufahren. Oder

wechseln Sie zu demCodemodus, der einen integrierten Code-Editor bietet, der VS Code ähnelt, um die Amazon States Language (ASL-) Definition Ihrer Zustandsmaschine in der Step Functions Functions-Konsole zu aktualisieren. Weitere Informationen zur Verwendung von Workflow Studio zum Erstellen Ihrer Zustandsmaschinen finden Sie unterVerwenden von Workflow Studio.

> **Important**
>
> Denken Sie daran, den Platzhalter Amazon Resource Name (ARN) für die im Beispielprojekt verwendeten Ressourcen zu aktualisieren, bevor Sie Ihren Workflow ausführen.

- Wenn Sie Eine Demo ausführen ausgewählt haben, erstellt Step Functions ein schreibgeschütztes Beispielprojekt, das eine AWS CloudFormation Vorlage verwendet, um die in dieser Vorlage aufgeführten AWS Ressourcen für Ihr Projekt bereitzustellen. AWS-Konto

  > **Tip**
  >
  > Um die State-Machine-Definition des Beispielprojekts anzuzeigen, wählen Sie Code.

  Wenn Sie bereit sind, wählen Sie Deploy and run aus, um das Beispielprojekt bereitzustellen und die Ressourcen zu erstellen.

  Es kann bis zu 10 Minuten dauern, bis diese Ressourcen und die zugehörigen IAM-Berechtigungen erstellt sind. Während der Bereitstellung Ihrer Ressourcen können Sie den CloudFormation Stack-ID-Link öffnen, um zu sehen, welche Ressourcen bereitgestellt werden.

  Nachdem alle Ressourcen im Beispielprojekt erstellt wurden, wird das neue Beispielprojekt auf der Seite State Machines aufgeführt.

  > **Important**
  >
  > Für jeden in der CloudFormation Vorlage verwendeten Dienst können Standardgebühren anfallen.

Schritt 2: Führen Sie die Zustandsmaschine aus

1. Wählen Sie auf der Seite State Machines Ihr Beispielprojekt aus.
2. Wählen Sie auf der Seite mit dem Beispielprojekt die Option Ausführung starten aus.
3. Gehen Sie im Dialogfeld Ausführung starten wie folgt vor:

    1. (Optional) Um Ihre Ausführung zu identifizieren, können Sie im Feld Name einen Namen dafür angeben. Standardmäßig generiert Step Functions automatisch einen eindeutigen Ausführungsnamen.

        > **Note**
        >
        > Mit Step Functions können Sie Namen für Zustandsmaschinen, Ausführungen und Aktivitäten sowie Beschriftungen erstellen, die Nicht-ASCII-Zeichen enthalten. Diese Nicht-ASCII-Namen funktionieren nicht mit Amazon. CloudWatch Um sicherzustellen, dass Sie CloudWatch Messwerte verfolgen können, wählen Sie einen Namen, der nur ASCII-Zeichen verwendet.

    2. (Optional) Geben Sie in das Eingabefeld Eingabewerte im JSON-Format ein, um Ihren Workflow auszuführen.

        Wenn Sie sich dafür entschieden haben, eine Demo auszuführen, müssen Sie keine Ausführungseingaben angeben.

        > **Note**
        >
        > Wenn das von Ihnen bereitgestellte Demo-Projekt vorab aufgefüllte Eingabedaten für die Ausführung enthält, verwenden Sie diese Eingabe, um die Zustandsmaschine auszuführen.

    3. Wählen Sie Start execution (Ausführung starten) aus.
    4. Die Step Functions Functions-Konsole leitet Sie zu einer Seite weiter, die mit Ihrer Ausführungs-ID betitelt ist. Diese Seite wird als Seite mit den Ausführungsdetails bezeichnet. Auf dieser Seite können Sie die Ausführungsergebnisse im Verlauf der Ausführung oder nach deren Abschluss überprüfen.

        Um die Ausführungsergebnisse zu überprüfen, wählen Sie in der Diagrammansicht einzelne Status aus und wählen Sie dann die einzelnen Registerkarten im Schrittdetails Bereich,

um die Details der einzelnen Status, einschließlich Eingabe, Ausgabe und Definition, anzuzeigen. Einzelheiten zu den Ausführungsinformationen, die Sie auf der Seite mit den Ausführungsdetails einsehen können, finden Sie unterSeite mit Ausführungsdetails – Schnittstellenübersicht.

Code des Zustandsautomaten aus diesem Beispiel

Die Zustandsmaschine in diesem Beispielprojekt lässt sich AWS Batch in Amazon SNS integrieren, indem Parameter direkt an diese Ressourcen übergeben werden.

Sehen Sie sich diese Beispiel-Zustandsmaschine an, um zu sehen, wie Step Functions AWS Batch und Amazon SNS steuern, indem Sie eine Verbindung zum Amazon-Ressourcennamen (ARN) im `Resource` Feld herstellen und `Parameters` an die Service-API weiterleiten.

Weitere Informationen darüber, wie AWS Step Functions Sie andere AWS Dienste steuern können, finden Sie unterVerwendung AWS Step Functions mit anderen Diensten.

```
{
  "Comment": "An example of the Amazon States Language for using batch job with pre-
processing lambda",
  "StartAt": "Generate batch job input",
  "TimeoutSeconds": 3600,
  "States": {
    "Generate batch job input": {
      "Type": "Task",
      "Resource": "arn:aws:states:::lambda:invoke",
      "OutputPath": "$.batch_input",
      "Parameters": {
        "FunctionName": "<GENERATE_BATCH_JOB_INPUT_LAMBDA_FUNCTION_NAME>"
      },
      "Next": "Submit Batch Job"
    },
    "Submit Batch Job": {
      "Type": "Task",
      "Resource": "arn:aws:states:::batch:submitJob.sync",
      "Parameters": {
        "JobName": "BatchJobFanOut",
        "JobQueue": "<BATCH_QUEUE_ARN>",
        "JobDefinition": "<BATCH_JOB_DEFINITION_ARN>",
        "Parameters.$": "$.batch_input"
      },
```

```
      "End": true
    }
  }
}
```

IAM-Beispiel

Diese vom Beispielprojekt generierten Beispielrichtlinien AWS Identity and Access Management (IAM) beinhalten die geringsten Rechte, die für die Ausführung der Zustandsmaschine und der zugehörigen Ressourcen erforderlich sind. Wir empfehlen, dass Sie nur die Berechtigungen in Ihre IAM-Richtlinien aufnehmen, die erforderlich sind.

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"
        }
    ]
}
```

Informationen zur Konfiguration von IAM bei der Verwendung von Step Functions mit anderen AWS Diensten finden Sie unterIAM-Richtlinien für integrierte Dienste.

Führen Sie AI-Prompt-Chaining durch mit Amazon Bedrock

Dieses Beispielprojekt zeigt, wie Sie sich integrieren können, um Amazon Bedrock AI-Prompt-Chaining durchzuführen. Dieses Beispielprojekt zeigt, wie Sie mithilfe von qualitativ hochwertige Chatbots erstellen können. Amazon Bedrock Das Projekt verkettet einige Eingabeaufforderungen und löst sie in der Reihenfolge, in der sie bereitgestellt werden. Die Verkettung dieser Eingabeaufforderungen verbessert die Fähigkeit des verwendeten Sprachmodells, eine sorgfältig kuratierte Antwort zu liefern.

In diesem Beispielprojekt werden die Zustandsmaschine und die unterstützenden AWS Ressourcen erstellt und die zugehörigen IAM-Berechtigungen konfiguriert. Erkunden Sie dieses Beispielprojekt, um mehr über die Verwendung der Amazon Bedrock optimierten Serviceintegration mit Step Functions Zustandsmaschinen zu erfahren, oder verwenden Sie es als Ausgangspunkt für Ihre eigenen Projekte.

Themen

AWS CloudFormation-Vorlage und zusätzliche Ressourcen

Sie verwenden eine CloudFormation Vorlage, um dieses Beispielprojekt bereitzustellen. Diese Vorlage erstellt die folgenden Ressourcen in Ihrem AWS-Konto:

- Eine Step Functions Zustandsmaschine.
- Ausführungsrolle für die Zustandsmaschine. Diese Rolle gewährt die Berechtigungen, die Ihre Zustandsmaschine für den Zugriff auf andere AWS-Services Ressourcen wie die Amazon Bedrock InvokeModelAktion benötigt.

Voraussetzungen

In diesem Beispielprojekt wird das Large Language Model (LLM) von Cohere Command verwendet. Um dieses Beispielprojekt erfolgreich auszuführen, müssen Sie von der Konsole aus Zugriff auf dieses LLM hinzufügen. Amazon Bedrock Gehen Sie wie folgt vor, um den Modellzugriff hinzuzufügen:

1. Öffnen Sie die Amazon Bedrock-Konsole.
2. Wählen Sie im Navigationsbereich Model Access aus.
3. Wählen Sie Modellzugriff verwalten aus.
4. Aktivieren Sie das Kontrollkästchen neben Cohere.
5. Wählen Sie Zugriff anfordern aus. Der Zugriffsstatus für das Modell Cohere wird als Zugriff gewährt angezeigt.

Schritt 1: Erstellen Sie den Zustandsmaschine und stellen Sie Ressourcen bereit

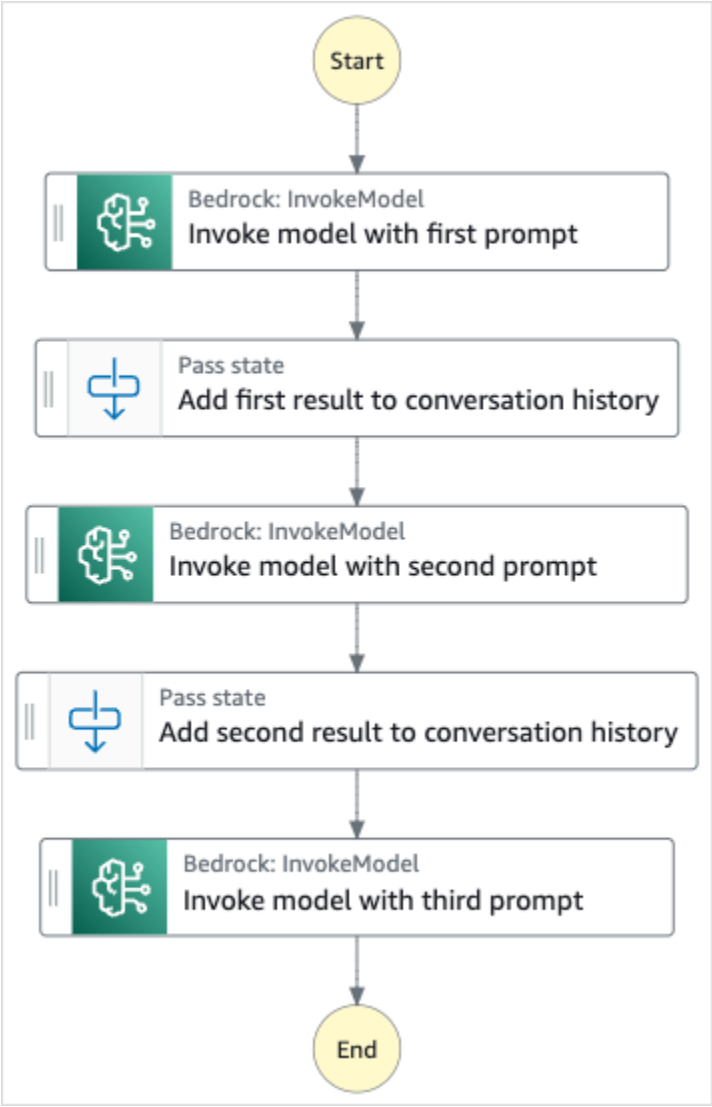
1. Öffnen Sie die Step Functions Functions-Konsole und wählen Sie Create State Machine.

2. Geben Sie **`bedrock`** etwas in das Suchfeld ein und wählen Sie dann aus den zurückgegebenen Suchergebnissen die Option KI-Prompt-Chaining durchführen Bedrock aus.
3. Wählen Sie Next (Weiter), um fortzufahren.
4. Step Functions listet die Funktionen auf, die in dem von Ihnen ausgewählten Beispielprojekt AWS-Services verwendet wurden. Außerdem wird ein Workflow-Diagramm für das Beispielprojekt angezeigt. Stellen Sie dieses Projekt für Ihr Projekt bereit AWS-Konto oder verwenden Sie es als Ausgangspunkt für die Erstellung Ihrer eigenen Projekte. Je nachdem, wie Sie vorgehen möchten, wählen Sie „Demo ausführen“ oder „Darauf aufbauen“.

   In diesem Beispielprojekt werden die folgenden Ressourcen bereitgestellt:

   - Eine AWS Step Functions Zustandsmaschine
   - Verwandte AWS Identity and Access Management Rollen (IAM)

   Die folgende Abbildung zeigt das Workflow-Diagramm für das Beispielprojekt Perform AI Prompt-Chaining with: Bedrock



5. Wählen Sie Vorlage verwenden, um mit Ihrer Auswahl fortzufahren.
6. Führen Sie eine der folgenden Aktionen aus:

   - Wenn Sie „Darauf aufbauen“ ausgewählt haben, erstellt Step Functions den Workflow-Prototyp für das von Ihnen ausgewählte Beispielprojekt. Step Functions stellt die in der Workflow-Definition aufgeführten Ressourcen nicht bereit.

     Ziehen Sie in Workflow Studio Status per Drag-and-Drop aus demEntwurfsmodus, Bundesstaaten-Browser um mit der Erstellung Ihres Workflow-Prototyps fortzufahren. Oder wechseln Sie zu demCodemodus, der einen integrierten Code-Editor bietet, der VS Code ähnelt, um die Amazon States Language (ASL-) Definition Ihrer Zustandsmaschine in der Step Functions Functions-Konsole zu aktualisieren. Weitere Informationen zur Verwendung

von Workflow Studio zum Erstellen Ihrer Zustandsmaschinen finden Sie unterVerwenden von Workflow Studio.

> **Important**
>
> Denken Sie daran, den Platzhalter Amazon Resource Name (ARN) für die im Beispielprojekt verwendeten Ressourcen zu aktualisieren, bevor Sie Ihren Workflow ausführen.

- Wenn Sie Eine Demo ausführen ausgewählt haben, erstellt Step Functions ein schreibgeschütztes Beispielprojekt, das eine AWS CloudFormation Vorlage verwendet, um die in dieser Vorlage aufgeführten AWS Ressourcen für Ihr Projekt bereitzustellen. AWS-Konto

  > **Tip**
  >
  > Um die State-Machine-Definition des Beispielprojekts anzuzeigen, wählen Sie Code.

  Wenn Sie bereit sind, wählen Sie Deploy and run aus, um das Beispielprojekt bereitzustellen und die Ressourcen zu erstellen.

  Es kann bis zu 10 Minuten dauern, bis diese Ressourcen und die zugehörigen IAM-Berechtigungen erstellt sind. Während der Bereitstellung Ihrer Ressourcen können Sie den CloudFormation Stack-ID-Link öffnen, um zu sehen, welche Ressourcen bereitgestellt werden.

  Nachdem alle Ressourcen im Beispielprojekt erstellt wurden, wird das neue Beispielprojekt auf der Seite State Machines aufgeführt.

  > **Important**
  >
  > Für jeden in der CloudFormation Vorlage verwendeten Dienst können Standardgebühren anfallen.

Schritt 2: Führen Sie die Zustandsmaschine aus

1. Wählen Sie auf der Seite State Machines Ihr Beispielprojekt aus.
2. Wählen Sie auf der Seite mit dem Beispielprojekt die Option Ausführung starten aus.

3. Gehen Sie im Dialogfeld Ausführung starten wie folgt vor:

   1. (Optional) Um Ihre Ausführung zu identifizieren, können Sie im Feld Name einen Namen dafür angeben. Standardmäßig generiert Step Functions automatisch einen eindeutigen Ausführungsnamen.

      > **Note**
      >
      > Mit Step Functions können Sie Namen für Zustandsmaschinen, Ausführungen, Aktivitäten und Labels erstellen, die Nicht-ASCII-Zeichen enthalten. Diese Nicht-ASCII-Namen funktionieren nicht mit Amazon. CloudWatch Um sicherzustellen, dass Sie CloudWatch Messwerte verfolgen können, wählen Sie einen Namen, der nur ASCII-Zeichen verwendet.

   2. (Optional) Geben Sie in das Eingabefeld Eingabewerte im JSON-Format ein, um Ihren Workflow auszuführen.

      Wenn Sie sich dafür entschieden haben, eine Demo auszuführen, müssen Sie keine Ausführungseingaben angeben.

   3. Wählen Sie Start execution (Ausführung starten) aus.
   4. Die Step Functions Functions-Konsole leitet Sie zu einer Seite weiter, die mit Ihrer Ausführungs-ID betitelt ist. Diese Seite wird als Seite mit den Ausführungsdetails bezeichnet. Auf dieser Seite können Sie die Ausführungsergebnisse im Verlauf der Ausführung oder nach deren Abschluss überprüfen.

      Um die Ausführungsergebnisse zu überprüfen, wählen Sie in der Diagrammansicht einzelne Status aus und wählen Sie dann die einzelnen Registerkarten im Schrittdetails Bereich, um die Details der einzelnen Status, einschließlich Eingabe, Ausgabe und Definition, anzuzeigen. Einzelheiten zu den Ausführungsinformationen, die Sie auf der Seite mit den Ausführungsdetails einsehen können, finden Sie unterSeite mit Ausführungsdetails – Schnittstellenübersicht.

Kontingente

AWS Step Functions legt Kontingente für die Größe bestimmter Zustandsmaschinen-Parameter fest, z. B. für die Anzahl der API-Aktionen in einem bestimmten Zeitraum oder für die Anzahl der Zustandsmaschinen, die Sie definieren können. Obwohl diese Kontingente entwickelt wurden, um zu verhindern, dass ein falsch konfigurierter Zustandsautomat alle Ressourcen des Systems belegt, sind viele davon keine harten Kontingente.

Um eine Erhöhung der Servicekontingenten zu beantragen, können Sie einen der folgenden Schritte ausführen:

- Verwenden Sie die Service Quotas Quotas-Konsole unter https://console.aws.amazon.com/servicequotas/home. Informationen zum Beantragen einer Kontingenterhöhung mit der Service Quotas Quota-Konsole finden Sie unter Eine Kontingenterhöhung beantragen im Service Quotas Quota-Benutzerhandbuch.
- Verwenden Sie die Seite Support Center in AWS Management Console , um eine Erhöhung des Kontingents für Ressourcen zu beantragen, die von AWS Step Functions pro Region bereitgestellt werden. Weitere Informationen finden Sie unter AWS Servicekontingente im Allgemeine AWS-Referenz.

> **Note**
>
> Wenn eine bestimmte Stufe der Ausführung von Zustandsautomaten oder Aktivitäten zu lange dauert, können Sie einen Zustandsautomaten-Timeout konfigurieren, um ein Timeout-Ereignis zu verursachen.

Themen

- Kontingente im Zusammenhang mit der Ausführung von Aufgaben
- Kontingente in Bezug auf Versionen und Aliase
- Einschränkungen im Zusammenhang mit dem Tagging

Allgemeine Kontingente

Kontingent	Beschreibung
Namen in Step Functions	Die Namen von Zustandsmaschinen, Ausführungen und Aktivitätsaufgaben dürfen nicht länger als 80 Zeichen sein. Diese Namen müssen für Ihr Konto und Ihre AWS Region eindeutig sein und dürfen keine der folgenden Angaben enthalten: • Leerraum • Platzhalterzeichen () ? * • Klammerzeichen () < > { } [ ] • Sonderzeichen (" # % \ ^ | ~ ` $ & , ; : /) • Steuerzeichen (\\u0000- \\u001f oder \ \u007f -\\u009f). Wenn Ihre Zustandsmaschine vom Typ Express ist, können Sie denselben Namen für mehrere Ausführungen der Zustandsmaschine angeben. Step Functions generiert für jede Ausführung von Express State Machine einen eindeutigen Ausführungs-ARN, auch wenn mehrere Ausführungen denselben Namen haben. Mit Step Functions können Sie Namen für Zustandsmaschinen, Ausführungen und Aktivitäten sowie Beschriftungen erstellen, die

Kontingent	Beschreibung
	Nicht-ASCII-Zeichen enthalten. Diese Nicht-ASCII-Namen funktionieren nicht mit Amazon. CloudWatch Um sicherzustellen, dass Sie CloudWatch Messwerte verfolgen können, wählen Sie einen Namen, der nur ASCII-Zei chen verwendet.

Kontingente im Zusammenhang mit Konten

Ressource	Standardkontingent	Kann auf erhöht werden
Maximale Anzahl registrierter Zustandsautomaten	10.000	25,000
Maximale Anzahl registrierter Aktivitäten	10.000	15 000
Maximale Anforderungsgröße	1 MB pro Anforderung. Dies ist die Gesamtdatengröße pro Step Functions Functions-API-Anfrage, einschließlich des Anforderungsheaders und aller anderen zugehörigen Anforderungsdaten.	Festes Kontingent
Maximale Anzahl offener Ausführungen pro Konto	AWS-Konto Jeweils 1.000.000 Hinrichtungen. AWS-Regio n Eine Überschreitung dieses Wertes führt zu einem `ExecutionLimitExce eded` -Fehler. Dies gilt nicht für Express Workflows.	Millionen
Maximale Anzahl geöffneter Kartenläufe	1000	Festes Kontingent

Ressource	Standardkontingent	Kann auf erhöht werden
Ein offener Kartenlauf ist ein Kartenlauf, der begonnen, aber noch nicht abgeschlossen wurde. Geplante Kartenläufe warten bei der MapRunStartedVeranstaltung, bis die Gesamtzahl der offenen Kartenläufe unter dem Standardkontingent von 1000 liegt.	Dieses Kontingent gilt für den Status Distributed Map.	
Maximaler redrivesWert eines Kartenlaufs.	1000 Dieses Kontingent gilt für den Status Distributed Map.	Festes Kontingent
Maximale Anzahl parallel Map Run-Unterausführungen	10.000	Festes Kontingent

Kontingente im Zusammenhang mit HTTP-Tasks

HTTP-Aufgaben werden mithilfe eines Token-Bucket-Schemas gedrosselt, um die Step Functions Dienstbandbreite aufrechtzuerhalten.

Ressource	Bucket-Größe	Nachfüllrate pro Sekunde
HTTP-Aufgabe	300	300

In der folgenden Tabelle ist das Kontingent für die Dauer einer HTTP-Task aufgeführt.

Ressource	Standardkontingent
Dauer einer HTTP-Aufgabe	60 Sekunden

Ressource	Standardkontingent
Eine HTTP-Task-Dauer bezieht sich auf die Zeit, die eine HTTP-Task benötigt, um eine HTTP-Anfrage zu senden und eine Antwort zu erhalten.	Dies ist ein festes Kontingent, das nicht geändert werden kann.

Kontingente im Zusammenhang mit staatlicher Drosselung

Statusübergänge von Step Functions werden mithilfe eines Token-Bucket-Schemas gedrosselt, um die Dienstbandbreite aufrechtzuerhalten. Bei Standard-Workflows und Express-Workflows gibt es unterschiedliche Drosselungen bei Zustandsübergängen. Bei den Quoten für Standard-Workflows handelt es sich um vorläufige Kontingente, die erhöht werden können.

> **Note**
>
> Die Drosselung der `StateTransition` Servicemetrik wird wie `ExecutionThrottled` bei Amazon gemeldet. CloudWatch Weitere Informationen finden Sie in der `ExecutionThrottled` CloudWatch Metrik.

	Standard		Express	
Servicemetrik	Bucket-Größe	Nachfüllrate pro Sekunde	Bucket-Größe	Nachfüllrate pro Sekunde
`StateTransition` — In den USA Ost (Nord-Virginia), USA West (Oregon) und Europa (Irland)	5,000	5,000	Unbegrenzt	Unbegrenzt
`StateTransition` —	800	800	Unbegrenzt	Unbegrenzt

	Standard		Express	
Servicemetrik	Bucket-Größe	Nachfüllrate pro Sekunde	Bucket-Größe	Nachfüllrate pro Sekunde
Alle anderen Regionen				

Kontingente im Zusammenhang mit der Drosselung von API-Aktionen

Einige API-Aktionen von Step Functions werden mithilfe eines Token-Bucket-Schemas gedrosselt, um die Dienstbandbreite aufrechtzuerhalten. Bei diesen Kontingenten handelt es sich um unverbindliche Kontingente, die erhöht werden können.

> **Note**
>
> Die Drosselungskontingente gelten pro Konto und Region. AWS
> AWS Step Functions kann sowohl die Eimergröße als auch die Nachfüllrate jederzeit erhöhen.

	Standard		Express	
API-Name	Bucket-Größe	Nachfüllrate pro Sekunde	Bucket-Größe	Nachfüllrate pro Sekunde
`StartExecution` — In den USA Ost (Nord-Virginia), den USA West (Oregon) und Europa (Irland)	1.300	300	6 000	6 000
`StartExecution` —	800	150	6 000	6 000

	Standard		Express	
API-Name	Bucket-Größe	Nachfüllrate pro Sekunde	Bucket-Größe	Nachfüllrate pro Sekunde
Alle anderen Regionen				

Kontingent im Zusammenhang mit der TestState API

API-Name	Kontingent	Kann auf erhöht werden
TestState	1 Transaktion pro Sekunde (TPS)	Festes Kontingent

Andere Kontingente

Bei diesen Quoten handelt es sich um weiche Kontingente, die erhöht werden können.

	In US East (N. Virginia), US West (Oregon), and Europe (Ireland)		All other regions	
API-Name	Bucket-Größe	Nachfüllrate pro Sekunde	Bucket-Größe	Nachfüllrate pro Sekunde
`CreateActivity`	100	1	100	1
`CreateStateMachine`	100	1	100	1
`DeleteActivity`	100	1	100	1
`DeleteStateMachine`	100	1	100	1

	In US East (N. Virginia), US West (Oregon), and Europe (Ireland)		All other regions	
API-Name	Bucket-Größe	Nachfüllrate pro Sekunde	Bucket-Größe	Nachfüllrate pro Sekunde
`DescribeActivity`	200	1	200	1
`DescribeExecution`	300	15	250	10
`DescribeStateMachine`	200	20	200	20
`DescribeStateMachineForExecution`	200	1	200	1
`GetActivityTask`	3,000	500	1.500	300
`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

	In US East (N. Virginia), US West (Oregon), and Europe (Ireland)		All other regions	
API-Name	Bucket-Größe	Nachfüllrate pro Sekunde	Bucket-Größe	Nachfüllrate pro Sekunde
`SendTaskFailure`	3,000	500	1.500	300
`SendTaskHeartbeat`	3,000	500	1.500	300
`SendTaskSuccess`	3,000	500	1.500	300
StartSyncExecution	Synchrone API-Aufrufe zur Express-Ausführung tragen nicht zu den bestehenden Kapazitätsgrenzen für Konten bei. Step Functions stellt Kapazität nach Bedarf bereit und skaliert automatisch bei anhaltender Arbeitslast. Ein Anstieg der Arbeitslast kann gedrosselt werden, bis Kapazität verfügbar ist. Wenn Sie eine Drosselung feststellen, versuchen Sie es nach einiger Zeit erneut. Informationen zu Synchronous Express-Workflows finden Sie unter. Synchrone und asynchrone Express-Workflows			
`StopExecution`	1.000	200	500	25
`TagResource`	200	1	200	1
`UntagResource`	200	1	200	1
`UpdateStateMachine`	100	1	100	1

Kontingente im Zusammenhang mit der Ausführung von Zustandsmaschinen

In der folgenden Tabelle werden die Kontingente für die Ausführung von Zustandsmaschinen beschrieben. Bei den Ausführungsquoten für Zustandsmaschinen handelt es sich um feste Kontingente, die nicht geändert werden können, mit Ausnahme des Zeitkontingents für die Aufbewahrung des Ausführungsverlaufs.

Kontingent	Standard	Express
Maximale Ausführungszeit	1 Jahr. Wenn eine Ausführung länger als das Maximum von einem Jahr läuft, schlägt sie mit einem `States.Timeout` Fehler fehl und gibt eine `ExecutionsTimedOut` CloudWatch Metrik aus.	5 Minuten. Wenn eine Ausführung länger als das Maximum von 5 Minuten dauert, schlägt sie mit einem `States.Timeout` Fehler fehl und gibt eine `ExecutionsTimedOut` CloudWatch Metrik aus.
Maximale Größe des Ausführungsverlaufs	25.000 Ereignisse in der Ausführungshistorie einer Maschine mit nur einem Status. Wenn der Ausführungsverlauf dieses Kontingent erreicht, schlägt die Ausführung fehl. Um dies zu vermeiden, beachten Sie Vermeiden Sie es, das historische Kontingent zu erreichen.	Unbegrenzt.
Maximale Ausführungs-Leerlaufzeit	1 Jahr (eingeschränkt durch die maximale Ausführungszeit).	5 Minuten (begrenzt durch die maximale Ausführungszeit).
Aufbewahrungszeit des Ausführungsverlaufs	90 Tage nach Abschluss einer Ausführung. Nach dieser Zeit können Sie den Ausführun	Um den Ausführungsverlauf zu sehen, muss die Amazon CloudWatch Logs-Prot

Kontingent	Standard	Express
	gsverlauf nicht mehr abrufen oder anzeigen. Es gibt kein weiteres Kontingent für die Anzahl der geschlossenen Ausführungen, die Step Functions beibehält. Um die Einhaltung gesetzlic her, organisatorischer oder behördlicher Anforderungen zu erfüllen, können Sie die Aufbewahrungsfrist für den Ausführungsverlauf auf 30 Tage reduzieren, indem Sie eine Kontingentanfrage senden. Verwenden Sie dazu den AWS Support Center Console und erstellen Sie einen neuen Fall. Die Änderung zur Verkürzun g der Aufbewahrungsfrist auf 30 Tage gilt für jedes Konto in einer Region.	okollierung konfiguriert sein. Weitere Informationen finden Sie unter Protokollierung mitCloudWatchProtokolle.

Kontingent	Standard	Express
redrivableAusführungszeitraum RedrivableDer Zeitraum bezieht sich auf den Zeitraum, in dem Sie redriveeinen bestimmten Standard-Workflow ausführen können. Dieser Zeitraum beginnt an dem Tag, an dem eine Zustandsmaschine ihre Ausführung abschließt.	14 Tage. Dieses feste Kontingent gilt für den Status Distributed Map.	Redrivewird derzeit nicht für Express-Workflows unterstützt.

Kontingente im Zusammenhang mit der Ausführung von Aufgaben

In der folgenden Tabelle werden Kontingente für Aufgabenausführungen beschrieben. Dies sind alles feste Kontingente, die nicht geändert werden können.

Kontingent	Standard	Express
Maximale Aufgabenausführungszeit	1 Jahr (begrenzt durch die maximale Ausführungszeit)	5 Minuten (begrenzt durch die maximale Ausführungszeit)
Maximale Zeit, für die Step Functions eine Aufgabe in der Warteschlange hält	1 Jahr (begrenzt durch die maximale Ausführungszeit)	5 Minuten (begrenzt durch die maximale Ausführungszeit)
Maximale Anzahl von Aktivitätsabfragen pro Amazon-Ressourcenname (ARN)	1.000 Poller, die `GetActivityTask` pro ARN aufrufen. Ein Überschreiten dieses Kontingents führt zu folgendem Fehler: „The maximum number of workers concurrently polling for activity tasks has been reached	Gilt nicht für Express-Workflows.

Kontingent	Standard	Express
	(Die maximale Anzahl der Worker, die gleichzeitig Aktivitätsaufgaben aufrufen, ist erreicht.)“	
Maximale Eingabe- oder Ausgabegröße für eine Aufgabe, einen Status oder eine Ausführung	256 KB Daten als UTF-8-kodierte Zeichenfolge. Dieses Kontingent wirkt sich auf Aufgaben (Aktivität, Lambda-Funktion oder integrierter Dienst), Status- oder Ausführungsausgabe und Eingabedaten aus, wenn eine Aufgabe geplant, in einen Status eingegeben oder eine Ausführung gestartet wird.	256 KB Daten als UTF-8-kodierte Zeichenfolge. Dieses Kontingent wirkt sich auf Aufgaben (Aktivität, Lambda-Funktion oder integrierter Dienst), Status- oder Ausführungsausgabe und Eingabedaten aus, wenn eine Aufgabe geplant, in einen Status eingegeben oder eine Ausführung gestartet wird.

Kontingente in Bezug auf Versionen und Aliase

Ressource	Standardkontingent
Maximale Anzahl veröffentlichter State-Machine-Versionen	1000 für jede Zustandsmaschine. Um eine Erhöhung dieses Soft-Limits zu beantragen, verwenden Sie die Support Center-Seite in der AWS Management Console.
Maximale Anzahl von Aliasnamen für Zustandsmaschinen	100 für jede Zustandsmaschine. Um eine Erhöhung dieses Soft-Limits zu beantragen, verwenden Sie die Support Center-Seite in der AWS Management Console.

Einschränkungen im Zusammenhang mit dem Tagging

Beachten Sie diese Einschränkungen, wenn Sie Step Functions Functions-Ressourcen taggen.

> **Note**
>
> Markierungseinschränkungen können anders als andere Kontingente nicht erhöht werden.

Einschränkung	Beschreibung
Maximale Anzahl von Tags pro Ressource	50
Maximale Schlüssellänge	128 Unicode-Zeichen in UTF-8
Maximale Länge des Wertes	256 Unicode-Zeichen in UTF-8
Präfixeinschränkung	Verwenden Sie das `aws:` Präfix nicht in Ihren Tagnamen oder -Werten, da es für die AWS Verwendung reserviert ist. Sie können keine Tag-Namen oder Werte mit diesem Präfix bearbeiten oder löschen. Tags mit diesem Präfix werden nicht auf Ihre Tags pro Ressourcenkontingent angerechnet.
Zeicheneinschränkungen	Tags dürfen nur Unicode-Buchstaben, Ziffern, Leerzeichen oder die folgenden Symbole enthalten: `_ . : / = + - @`.

Einloggen und Überwachen AWS Step Functions

Protokollierung und Überwachung sind wichtig, um die Zuverlässigkeit, Verfügbarkeit und Leistung von Step Functions und Ihren AWS Lösungen aufrechtzuerhalten. Für die Verwendung mit Step Functions stehen mehrere Tools zur Verfügung:

> **Tip**
>
> In Modul 12 — Observability of The Workshop erfahren Sie, wie Sie einen Beispiel-Workflow für Sie bereitstellen und erfahren, wie Sie Metriken, Logs und Traces der AWS Step Functions Workflow-Ausführung überwachen können. AWS-Konto

Themen

Step Functions überwachen mit CloudWatch

Die Überwachung ist ein wichtiger Bestandteil der Aufrechterhaltung der Zuverlässigkeit, Verfügbarkeit AWS Step Functions und Leistung Ihrer AWS Lösungen. Sie sollten so viele Überwachungsdaten von den von Ihnen verwendeten AWS Diensten sammeln, dass Sie Fehler an mehreren Punkten debuggen können. Bevor Sie mit der Überwachung von Step Functions beginnen, sollten Sie einen Überwachungsplan erstellen, der die folgenden Fragen beantwortet:

- Was sind Ihre Ziele bei der Überwachung?
- Welche Ressourcen werden überwacht?
- Wie oft werden diese Ressourcen überwacht?

- Welche Überwachungstools werden verwendet?
- Wer soll die Überwachungsaufgaben ausführen?
- Wer soll benachrichtigt werden, wenn Fehler auftreten?

Der nächste Schritt besteht darin, eine Baseline für normale -Performance in Ihrer Umgebung aufzustellen. Dafür sollten Sie die Performance zu verschiedenen Zeiten und unter verschiedenen Belastungsbedingungen messen. Denken Sie bei der Überwachung von Step Functions darüber nach, historische Überwachungsdaten zu speichern. Solche Daten können eine Basis für den Vergleich mit aktuellen Leistungsdaten bieten, um normale Leistungsmuster und Leistungsanomalien zu identifizieren sowie Verfahren für den Umgang mit Problemen zu entwickeln.

Mit Step Functions können Sie beispielsweise überwachen, wie viele Aktivitäten oder AWS Lambda Aufgaben aufgrund eines Heartbeat-Timeouts fehlschlagen. Wenn die Performance außerhalb Ihrer festgelegten Baseline fällt, müssen Sie eventuell Ihr Heartbeat-Intervall ändern.

Zur Festlegung einer Baseline sollten Sie mindestens die folgenden Metriken überwachen:

- `ActivitiesStarted`
- `ActivitiesTimedOut`
- `ExecutionsStarted`
- `ExecutionsTimedOut`
- `LambdaFunctionsStarted`
- `LambdaFunctionsTimedOut`

In den folgenden Abschnitten werden Metriken beschrieben, die Step Functions Amazon zur Verfügung stellt CloudWatch. Mit diesen Metriken können Sie Ihren Zustandsautomaten und Aktivitäten überwachen sowie Alarme für Schwellenwerte festlegen. Sie können Metriken mit dem anzeigen AWS Management Console.

Metriken, die ein Zeitintervall angeben

Einige der Step Functions CloudWatch Functions-Metriken sind Zeitintervalle, die immer in Millisekunden gemessen werden. Diese Metriken entsprechen im Allgemeinen Phasen Ihrer Ausführung, für die Sie Timeouts für Zustandsmaschinen, Aktivitäten und Lambda-Funktionen mit aussagekräftigen Namen festlegen können.

Die Metrik `ActivityRunTime` misst die Zeit für die Ausführung einer Aktivität vom Beginn bis zum Abschluss. Sie können einen Timeout-Wert für den gleichen Zeitraum festlegen.

In der CloudWatch Konsole können Sie die besten Ergebnisse erzielen, wenn Sie „Durchschnitt“ als Anzeigestatistik für Zeitintervallmetriken wählen.

Metriken, die eine Anzahl melden

Einige der Step Functions CloudWatch Functions-Metriken geben Ergebnisse als Zählung an. Beispielsweise erfasst `ExecutionsFailed` die Anzahl fehlgeschlagener Ausführungen von Zustandsautomaten.

Step Functions gibt zwei `ExecutionsStarted` Metriken für jede State-Machine-Ausführung aus. Dadurch zeigt die SampleCountStatistik für die `ExecutionsStarted` Metrik für jede Zustandsmaschinen-Ausführung den Wert 2 an. Die SampleCount Statistik zeigt an, `ExecutionStarted=0` wann `ExecutionStarted=1` die Ausführung abgeschlossen ist.

> **Tip**
>
> Wir empfehlen, Summe als Anzeigestatistik für Metriken auszuwählen, die eine Anzahl in der CloudWatch Konsole melden.

Ausführungsmetriken

Der `AWS/States` Namespace enthält die folgenden Metriken für alle Step Functions Functions-Ausführungen. Dies sind dimensionslose Metriken, die für Ihr gesamtes Konto in einer Region gelten.

Metrik	Beschreibung
`OpenExecutionCount`	Ungefähre Anzahl der derzeit offenen Ausführungen — Workflows, die derzeit in Ihrem Konto ausgeführt werden. Ziel ist es, Ihnen einen Einblick zu geben, wann sich Ihre Workflows dem maximalen Ausführungslimit nähern, um ExecutionLimitExceededFehler beim Aufrufen `StartExecution` oder `RedriveExecution` bei Standard-Workflows zu vermeiden.

Metrik	Beschreibung
	Die Metrik hängt von den Zustandsübergängen im aktiven Workflow ab. Bei niedrigen Werten stimmt die Schätzung daher möglicherweise nicht mit der beobachteten Anzahl laufender Workflows überein.
`OpenExecutionLimit`	Maximale Anzahl offener Ausführungen. Weitere Informationen finden Sie unter Kontingente im Zusammenhang mit Konten. Dieses Limit gilt nicht für Express Workflows.

Ausführungsmetriken für State Machine mit Version oder Alias

Wenn Sie eine State-Machine-Ausführung mit einer Version oder einem Alias ausführen, gibt Step Functions die folgenden Metriken aus. Die `ExecutionThrottled` Metrik wird nur bei gedrosselter Ausführung ausgegeben. Zu diesen Metriken gehört auch eine`StateMachineArn`, mit der ein bestimmter Zustandsmaschine identifiziert werden kann.

Metrik	Beschreibung
`ExecutionTime`	Intervall in Millisekunden zwischen dem Start der Ausführung und dem Zeitpunkt, zu dem sie beendet wird.
`ExecutionThrottled`	Anzahl der `StateEntered` Ereignisse und Wiederholungen, die gedrosselt wurden. Dies hängt mit der `StateTransition` - Ablehnung zusammen. Weitere Informationen finden Sie unter Kontingente im Zusammenhang mit staatlicher Drosselung.
`ExecutionsAborted`	Anzahl der abgebrochenen oder abgebrochenen Ausführungen.
`ExecutionsFailed`	Anzahl der fehlgeschlagenen Ausführungen.
`ExecutionsStarted`	Anzahl der gestarteten Ausführungen.
`ExecutionsSucceeded`	Anzahl der erfolgreich abgeschlossenen Ausführungen.
`ExecutionsTimedOut`	Anzahl der Ausführungen, bei denen aus irgendeinem Grund ein Timeout aufgetreten ist.

Ausführungsmetriken für Express Workflows

Der `AWS/States` Namespace enthält die folgenden Metriken für die Ausführung von Step Functions Express Workflows.

Metrik	Beschreibung
`ExpressExecutionMemory`	Der gesamte Arbeitsspeicher, der von einem Express-Workflow verbraucht wird.
`ExpressExecutionBilledDuration`	Die Dauer, für die ein Express-Workflow berechnet wird.
`ExpressExecutionBilledMemory`	Die Menge an verbrauchtem Speicher, für die ein Express-Workflow berechnet wird.

RedriveAusführungsmetriken für Standard-Workflows

Wenn Sie redriveeine State-Machine-Ausführung ausführen, gibt Step Functions die folgenden Metriken aus.

Für alle redriven Ausführungen wird die `Executions*` Metrik ausgegeben. Nehmen wir zum Beispiel an, eine redriven Ausführung wird abgebrochen. Diese Ausführung gibt Datenpunkte ungleich Null für sowohl als auch aus. `RedrivenExecutionsAborted` `ExecutionsAborted`

Metrik	Beschreibung
`ExecutionsRedriven`	Anzahl der Ausführungen. redriven
`RedrivenExecutionsAborted`	Anzahl der redriven Ausführungen, die abgebrochen oder beendet wurden.
`RedrivenExecutionsTimedOut`	Anzahl der redriven Ausführungen, bei denen aus irgendeinem Grund ein Timeout auftritt.
`RedrivenExecutionsSucceeded`	Anzahl der redriven Ausführungen, die erfolgreich abgeschlossen wurden.

Metrik	Beschreibung
`RedrivenExecutionsFailed`	Anzahl der redriven fehlgeschlagenen Ausführungen.

Dimension für die Ausführungsmetriken von Step Functions

Dimension	Beschreibung
`StateMachineArn`	Der Amazon-Ressourcenname (ARN) der Zustandsmaschine für die fragliche Ausführung.

Abmessungen für Ausführungen mit Version

Dimension	Beschreibung
`StateMachineArn`	Der Amazon-Ressourcenname (ARN) der Zustandsmaschine, deren Ausführung durch eine Version gestartet wurde.
`Version`	State-Machine-Version, mit der die Ausführung gestartet wurde.

Dimensionen für Ausführungen mit einem Alias

Dimension	Beschreibung
`StateMachineArn`	Der Amazon-Ressourcenname (ARN) der Zustandsmaschine, deren Ausführung durch einen Alias gestartet wurde.
`Alias`	Alias der Zustandsmaschine, der zum Starten der Ausführung verwendet wurde.

Metriken zur Anzahl der Ressourcen für Versionen und Aliase

Der `AWS/States` Namespace umfasst die folgenden Metriken für die Anzahl der Versionen und Aliase einer Zustandsmaschine.

Metrik	Beschreibung
AliasCount	Anzahl der Aliase, die für den Zustandsmaschine erstellt wurden. Sie können bis zu 100 Aliase für jede Zustandsmaschine erstellen.
VersionCount	Anzahl der für den State Machine veröffentlichten Versionen. Sie können bis zu 1000 Versionen einer State Machine veröffentlichen.

Dimension für die Anzahl der Ressourcen, Metriken für Versionen und Aliase

Dimension	Beschreibung
`ResourceArn`	Der Amazon-Ressourcenname (ARN) der Zustandsmaschine mit einer Version oder einem Alias.

Metriken für Aktivitäten

Der `AWS/States` Namespace enthält die folgenden Metriken für Step Functions Functions-Aktivitäten.

Metrik	Beschreibung
`ActivityRunTime`	Intervall in Millisekunden zwischen dem Start der Aktivität und dem Zeitpunkt, zu dem sie endet.
`ActivityScheduleTime`	Intervall in Millisekunden, für das die Aktivität im Zeitplanstatus verbleibt.

Metrik	Beschreibung
`ActivityTime`	Intervall in Millisekunden zwischen dem Zeitpunkt, zu dem die Aktivität geplant ist, und dem Zeitpunkt, zu dem sie endet.
`ActivitiesFailed`	Anzahl der fehlgeschlagenen Aktivitäten.
`ActivitiesHeartbeatTimedOut`	Anzahl der Aktivitäten, bei denen aufgrund eines Heartbeat-Timeouts ein Timeout auftritt.
`ActivitiesScheduled`	Anzahl der geplanten Aktivitäten.
`ActivitiesStarted`	Anzahl der gestarteten Aktivitäten.
`ActivitiesSucceeded`	Anzahl der erfolgreich abgeschlossenen Aktivitäten.
`ActivitiesTimedOut`	Anzahl der Aktivitäten, bei denen das Timeout beim Abschluss abgelaufen ist.

Dimension für Step Functions, Aktivitätsmetriken

Dimension	Beschreibung
`ActivityArn`	ARN der Aktivität.

Metriken für die Lambda-Funktion

Der `AWS/States` Namespace enthält die folgenden Metriken für Step Functions Lambda-Funktionen.

Metrik	Beschreibung
`LambdaFunctionRunTime`	Intervall in Millisekunden zwischen dem Start der Lambda-Funktion und dem Zeitpunkt, zu dem sie geschlossen wird.
`LambdaFunctionScheduleTime`	Intervall in Millisekunden, für das die Lambda-Funktion im Zeitplanstatus verbleibt.

Metrik	Beschreibung
`LambdaFunctionTime`	Intervall in Millisekunden zwischen dem Zeitpunkt, zu dem die Lambda-Funktion geplant ist, und dem Zeitpunkt, zu dem sie geschlossen wird.
`LambdaFunctionsFailed`	Anzahl der fehlgeschlagenen Lambda-Funktionen.
`LambdaFunctionsScheduled`	Anzahl der geplanten Lambda-Funktionen.
`LambdaFunctionsStarted`	Anzahl der gestarteten Lambda-Funktionen.
`LambdaFunctionsSucceeded`	Anzahl der erfolgreich abgeschlossenen Lambda-Funktionen.
`LambdaFunctionsTimedOut`	Anzahl der Lambda-Funktionen, bei denen beim Schließen ein Timeout auftritt.

Dimension für Step Functions Lambda-Funktionsmetriken

Dimension	Beschreibung
`LambdaFunctionArn`	ARN der Lambda-Funktion.

> **Note**
>
> Lambda-Funktionsmetriken werden für Task-Status ausgegeben, die den ARN der Lambda-Funktion im `Resource` Feld angeben. Aufgabenstatus, die stattdessen Service Integration-Metriken verwenden`"Resource": "arn:aws:states:::lambda:invoke"`, werden ausgegeben. Weitere Informationen finden Sie unter Lambda mit Step-Funktionen aufrufen.

Serviceintegrationsmetriken

Der `AWS/States` Namespace enthält die folgenden Metriken für Step Functions Functions-Dienstintegrationen. Weitere Informationen finden Sie unter Verwendung AWS Step Functions mit anderen Diensten.

Metrik	Beschreibung
`ServiceIntegrationRunTime`	Intervall in Millisekunden zwischen dem Start der Service-Task und dem Zeitpunkt, zu dem sie geschlossen wird.
`ServiceIntegrationScheduleTime`	Intervall in Millisekunden, für das der Service-Task im Zeitplanstatus verbleibt.
`ServiceIntegrationTime`	Intervall in Millisekunden zwischen dem Zeitpunkt, zu dem der Service Task geplant ist, und dem Zeitpunkt, zu dem er geschlossen wird.
`ServiceIntegrationsFailed`	Anzahl der fehlgeschlagenen Serviceaufgaben.
`ServiceIntegrationsScheduled`	Anzahl der geplanten Serviceaufgaben.
`ServiceIntegrationsStarted`	Anzahl der gestarteten Serviceaufgaben.
`ServiceIntegrationsSucceeded`	Anzahl der erfolgreich abgeschlossenen Serviceaufgaben.
`ServiceIntegrationsTimedOut`	Anzahl der Serviceaufgaben, bei denen beim Abschluss ein Timeout auftritt.

Dimension für Step Functions — Serviceintegrationsmetriken

Dimension	Beschreibung
`ServiceIntegration ResourceArn`	Der Ressourcen-ARN des integrierten Service.

Servicemetriken

Der `AWS/States` Namespace umfasst die folgenden Metriken für den Step Functions Functions-Dienst.

Metrik	Beschreibung
`ThrottledEvents`	Anzahl der Anfragen, die gedrosselt wurden.
`ProvisionedBucketS ize`	Anzahl der verfügbaren Anfragen pro Sekunde.
`ProvisionedRefillR ate`	Anzahl der Anfragen pro Sekunde, die in den Bucket aufgenommen werden dürfen.
`ConsumedCapacity`	Anzahl der Anfragen pro Sekunde.

Dimension für Step Functions Servicemetriken

Dimension	Beschreibung
`ServiceMetric`	Filtert Daten, um Zustandsübergang-Metriken anzuzeigen.

API-Metriken

Der `AWS/States` Namespace enthält die folgenden Metriken für die Step Functions Functions-API.

Metrik	Beschreibung
`ThrottledEvents`	Anzahl der Anfragen, die gedrosselt wurden.
`ProvisionedBucketSize`	Anzahl der verfügbaren Anfragen pro Sekunde.
`ProvisionedRefillRate`	Anzahl der Anfragen pro Sekunde, die in den Bucket aufgenommen werden dürfen.
`ConsumedCapacity`	Anzahl der Anfragen pro Sekunde.

Dimension für Step Functions API-Metriken

Dimension	Beschreibung
`APIName`	Filtert Daten nach der API mit dem angegebenen API-Namen

Bereitstellung von CloudWatch Metriken nach bestem Wissen

CloudWatch-Metriken werden auf einer Best-Effort-Basis bereitgestellt.

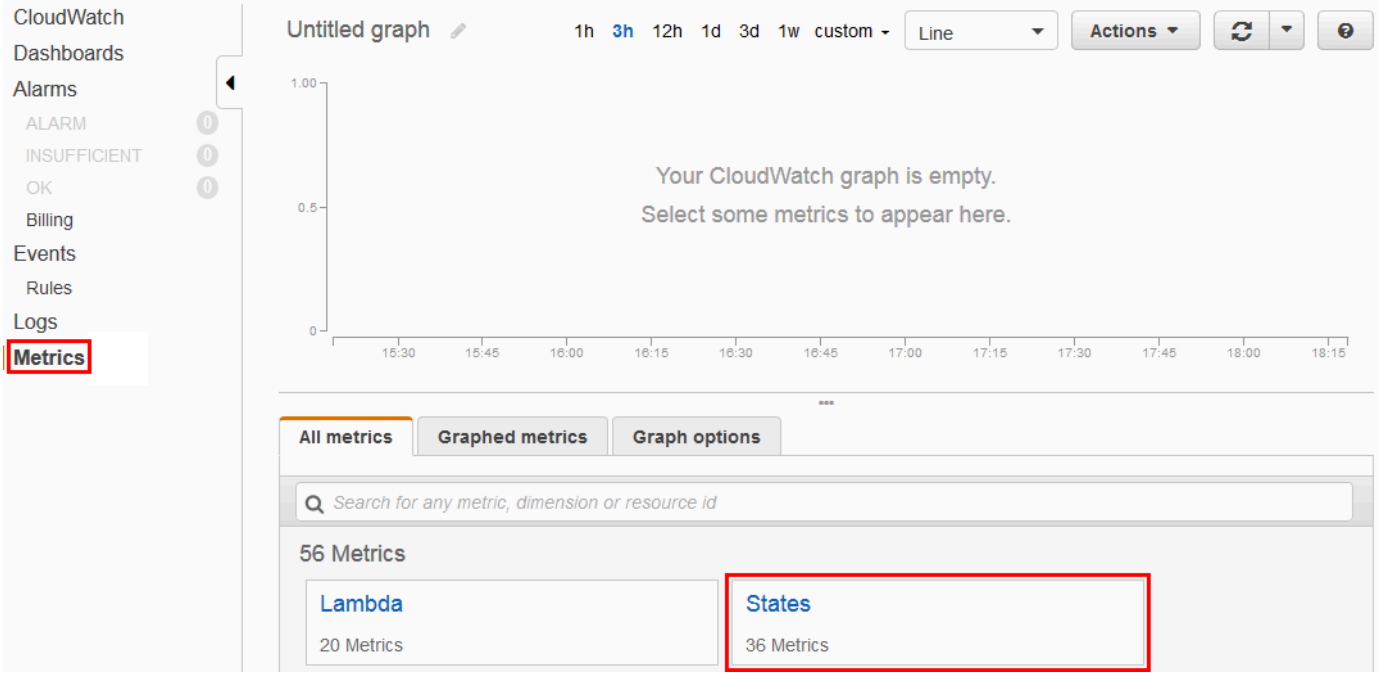
Die Vollständigkeit und Rechtzeitigkeit der Metriken ist nicht garantiert. Der Datenpunkt für eine bestimmte Anforderung wird möglicherweise mit einem Zeitstempel zurückgegeben, der nach der tatsächlichen Anforderungsverarbeitung liegt. Der Datenpunkt kann für eine Minute verzögert sein, bevor er verfügbar ist CloudWatch, oder er wird möglicherweise gar nicht geliefert. CloudWatchAnforderungsmetriken geben Ihnen nahezu in Echtzeit einen Überblick über die Ausführung der State-Machine-Vorgänge. Es handelt sich nicht um eine vollständige Erfassung aller Ausführungskennzahlen.

Aufgrund des Best-Effort-Charakters dieser Funktion können die im Billing & Cost Management-Dashboard verfügbaren Berichte eine oder mehrere Zugriffsanforderungen enthalten, die nicht in den Ausführungskennzahlen enthalten sind.

Metriken für Step Functions anzeigen

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CloudWatch Konsole.

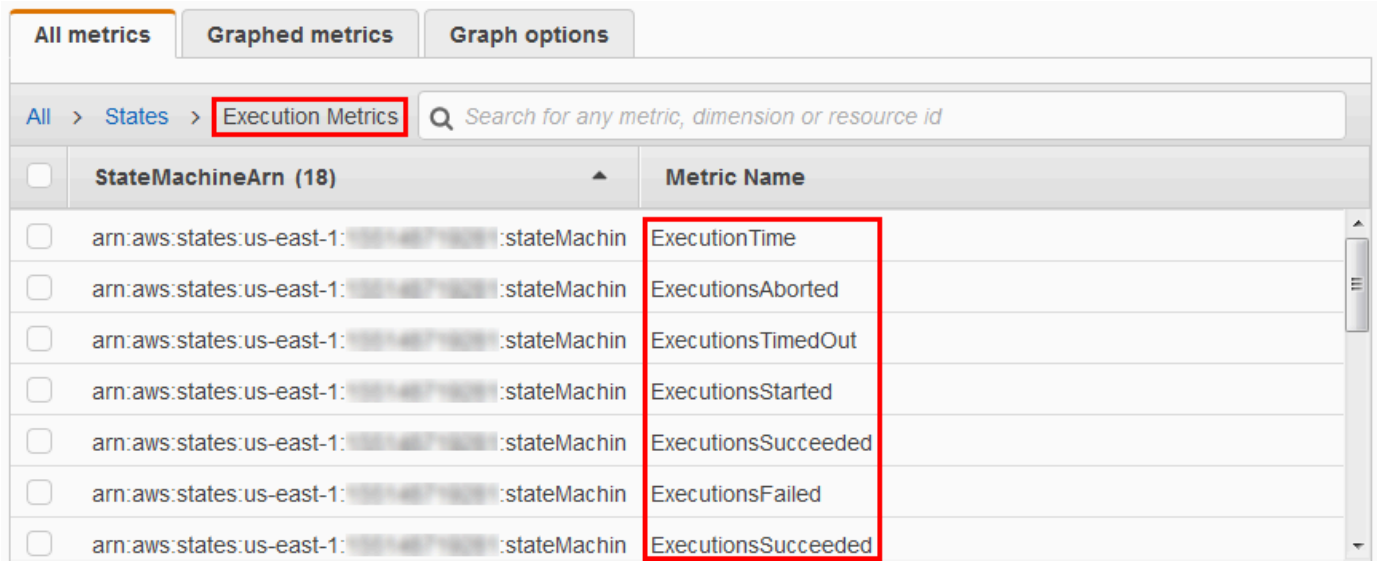
2. Wählen Sie Metrics (Metriken) und auf der Registerkarte All Metrics (Alle Metriken) States (Zustände) aus.



Wenn Sie kürzlich Ausführungen durchgeführt haben, werden Ihnen bis zu vier Arten von Metriken angezeigt:

- Execution Metrics (Ausführungsmetriken)
- Activity Function Metrics (Aktivitätsfunktionsmetriken)
- Lambda-Funktionsmetriken
- Kennzahlen zur Serviceintegration

3. Wählen Sie einen Metriktyp aus, um eine Liste der Metriken zu sehen.

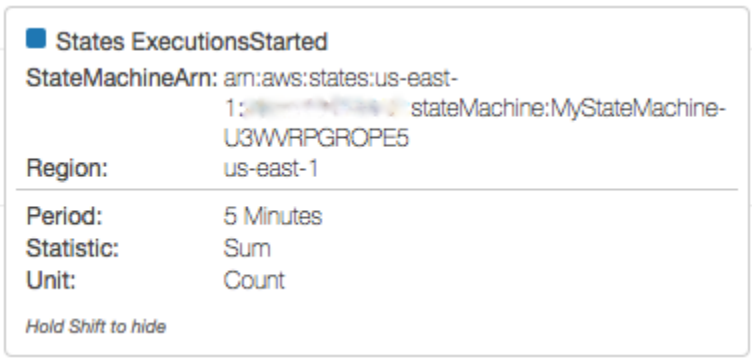


- Um Ihre Metriken nach Metrikname oder zu sortieren StateMachineArn, verwenden Sie die Spaltenüberschriften.
- Um Diagramme für eine Metrik anzuzeigen, aktivieren Sie das Kontrollkästchen neben der Metrik auf der Liste. Sie können die Graph-Parameter mithilfe der Zeitraumsteuerungen oberhalb der Diagrammansicht ändern.

  Sie können benutzerdefinierte Zeiträume mit relativen oder absoluten Werten auswählen (bestimmte Tage und Zeiten). Sie können auch die Drop-down-Liste verwenden, um Werte wie Zeilen, gestapelte Bereiche oder Zahlen (Werte) anzuzeigen.
- Um Details zu einem Diagramm anzuzeigen, bewegen Sie die Maus über den Farbcode der Metrik, der unter dem Diagramm angezeigt wird.

  Die Details der Metrik werden angezeigt.

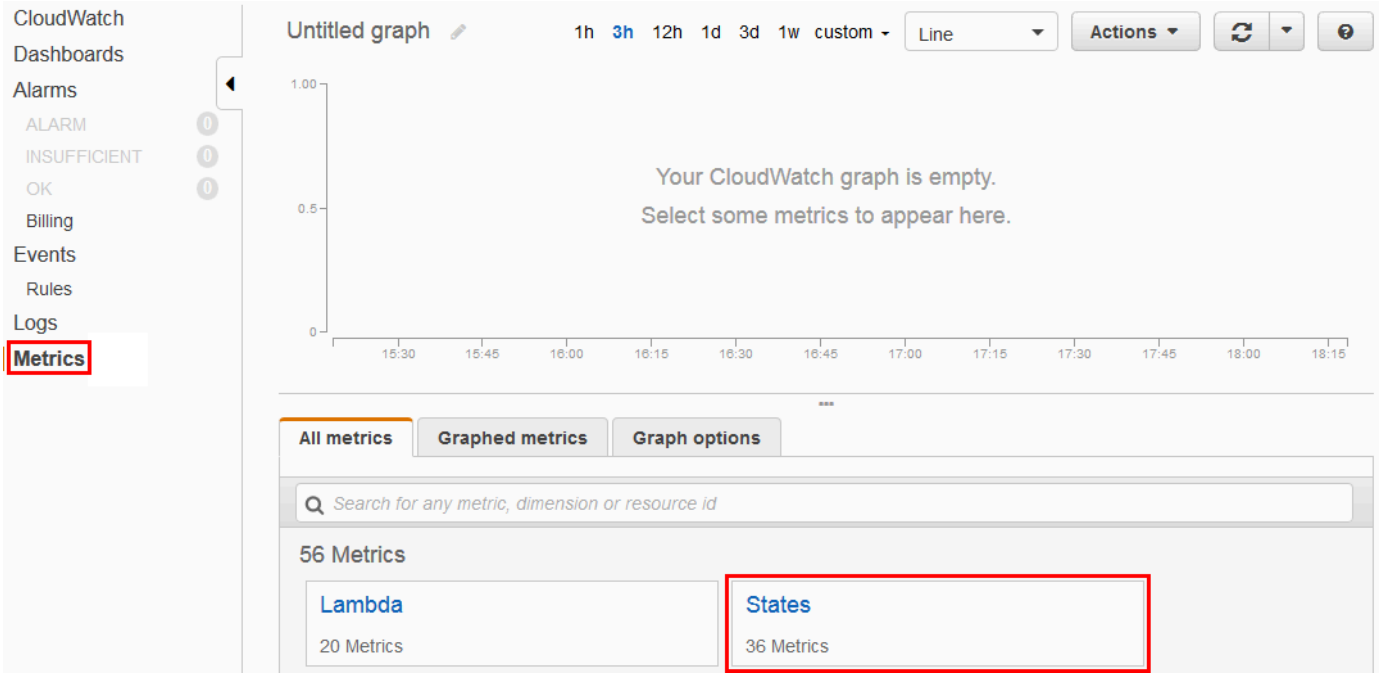
Weitere Informationen zur Arbeit mit CloudWatch Metriken finden Sie unter Using Amazon CloudWatch Metrics im CloudWatch Amazon-Benutzerhandbuch.

Alarme für Step Functions einstellen

Sie können CloudWatch Amazon-Alarme verwenden, um Aktionen auszuführen. Wenn Sie beispielsweise wissen möchten, wann ein Alarmschwellenwert erreicht ist, können Sie einen Alarm so einrichten, dass eine Benachrichtigung an ein Amazon SNS SNS-Thema gesendet wird oder dass eine E-Mail gesendet wird, wenn die `StateMachinesFailed` Metrik einen bestimmten Schwellenwert überschreitet.

Einrichten eines Alarms zu einer Metrik

1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CloudWatch Konsole.
2. Wählen Sie Metrics (Metriken) und auf der Registerkarte All Metrics (Alle Metriken) States (Zustände) aus.

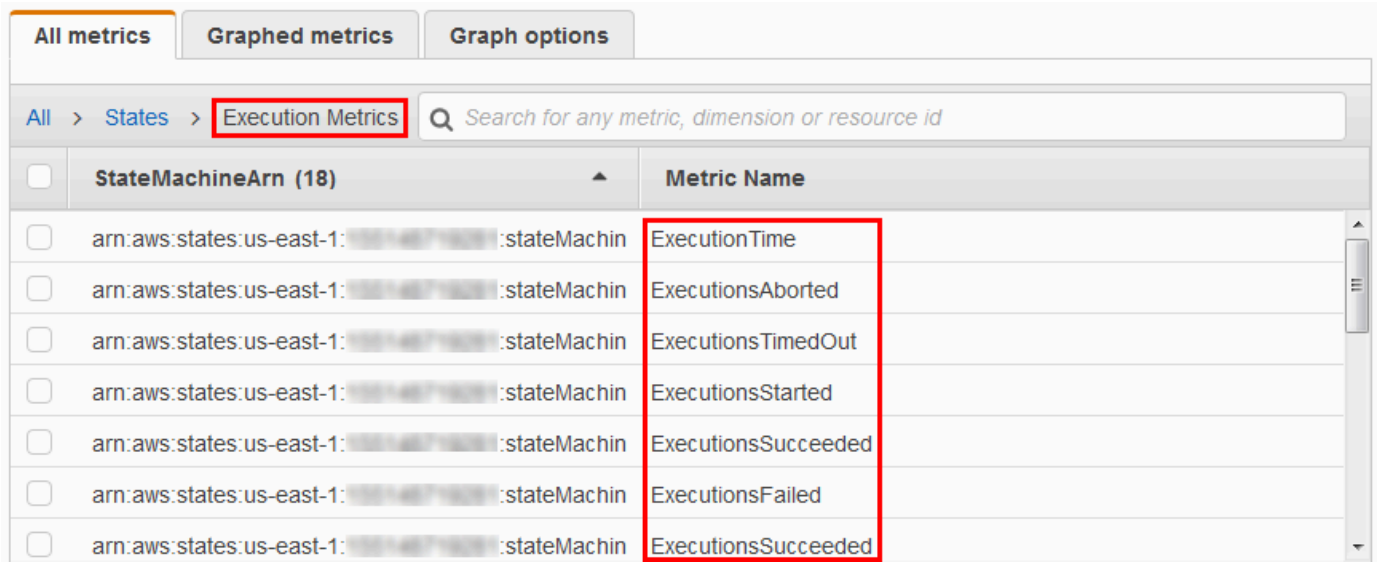


Wenn Sie kürzlich Ausführungen durchgeführt haben, werden Ihnen bis zu vier Arten von Metriken angezeigt:

- Execution Metrics (Ausführungsmetriken)
- Activity Function Metrics (Aktivitätsfunktionsmetriken)
- Lambda-Funktionsmetriken

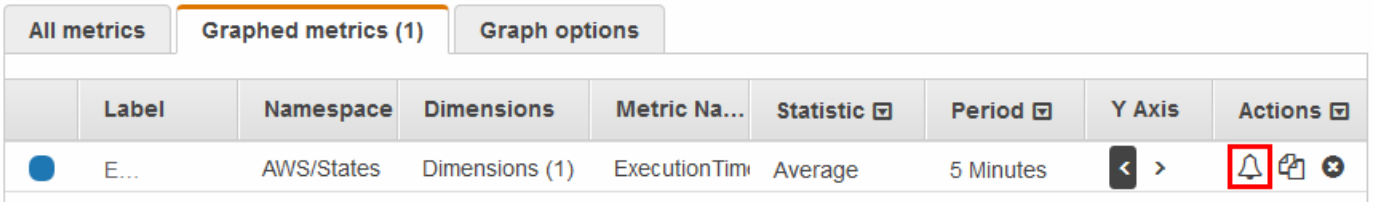
- Kennzahlen zur Serviceintegration

3. Wählen Sie einen Metriktyp aus, um eine Liste der Metriken zu sehen.

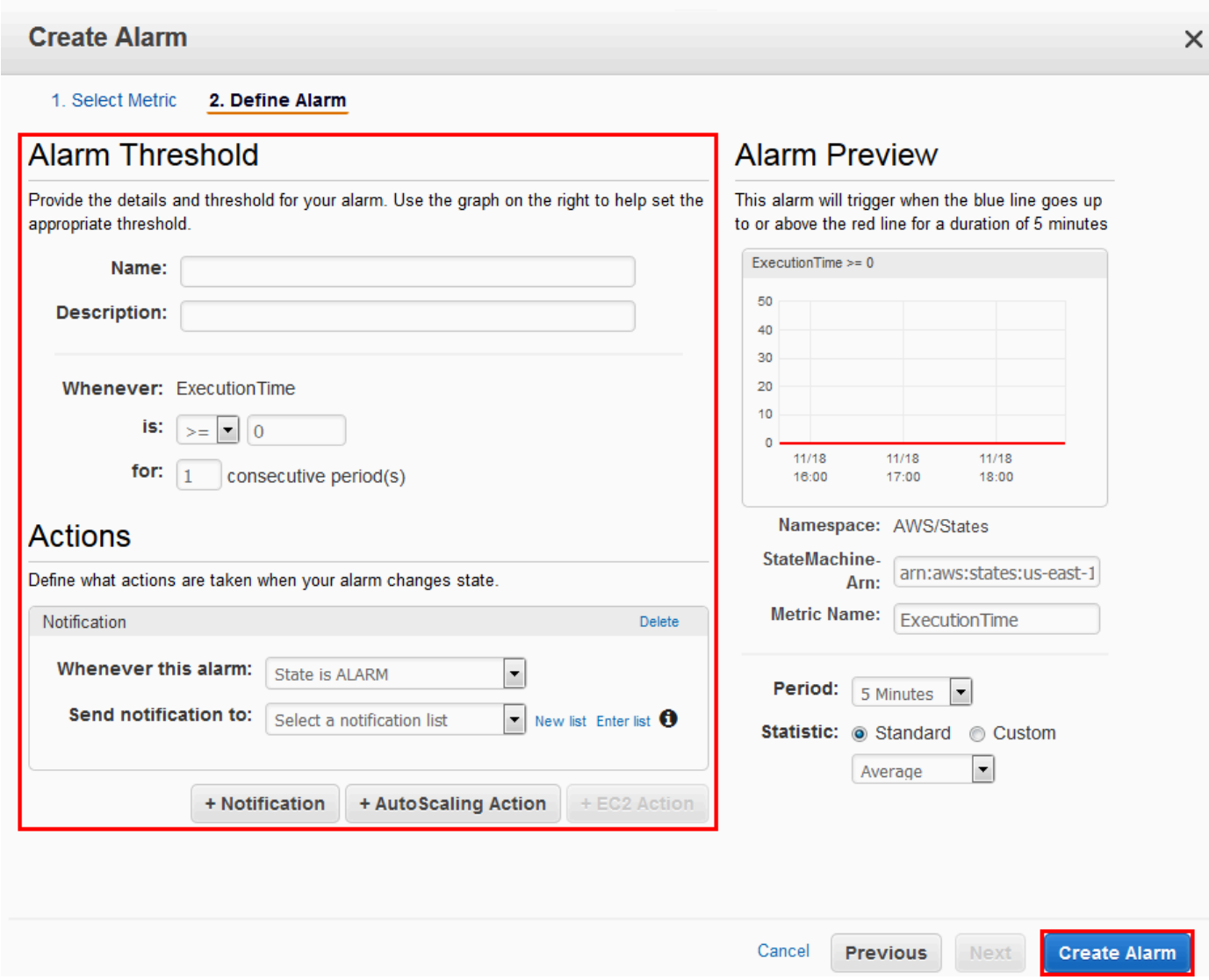


4. Wählen Sie eine Metrik und anschließend Graphed metrics (Grafisch dargestellte Metriken) aus.
5. Wählen Sie

neben einer Metrik auf der Liste aus.



Die Seite Create Alarm (Alarm erstellen) wird angezeigt.



6. Geben Sie die Werte für Alarm threshold (Alarmschwellenwert) und Actions (Aktionen) ein und wählen Sie dann Create Alarm (Alarm erstellen) aus.

Weitere Informationen zur Einstellung und Verwendung von CloudWatch Alarmen finden Sie unter CloudWatch Amazon-Alarme erstellen im CloudWatch Amazon-Benutzerhandbuch.

EventBridge (CloudWatch Ereignisse) für Statusänderungen bei der Ausführung von Step Functions

Amazon EventBridge ist ein - AWS Service, mit dem Sie auf Statusänderungen in einer - AWS Ressource reagieren können. Sie können beispielsweise auf die Änderungen des Ausführungsstatus eines Step Functions Standard Workflow mit EventBridge auf zwei Arten reagieren:

- Sie können EventBridge Regeln so konfigurieren, dass sie auf Ereignisse reagieren, die ausgegeben werden, wenn sich der Ausführungsstatus eines Step-Functions-Zustandsautomaten ändert. Auf diese Weise haben Sie die Möglichkeit, Ihre Workflows zu überwachen, ohne ständig abfragen zu müssen, indem Sie die DescribeExecution API verwenden. Basierend auf Änderungen bei der Ausführung von Zustandsautomaten können Sie ein - EventBridge Ziel verwenden, um neue Zustandsautomaten-Ausführungen zu starten, AWS Lambda Funktionen aufzurufen, Nachrichten in Amazon Simple Notification Service (Amazon SNS)-Themen zu veröffentlichen und vieles mehr.
- Sie können einen Step-Functions-Zustandsautomaten auch als Ziel in konfigurieren EventBridge. Auf diese Weise können Sie eine Ausführung eines Step Functions-Workflows als Reaktion auf ein Ereignis von einem anderen AWS Service auslösen.

  Weitere Informationen finden Sie im Amazon- EventBridge Benutzerhandbuch.

Express Workflows gibt jedoch keine Ereignisse an aus EventBridge. Um die Ausführung eines Express-Workflows zu überwachen, können Sie - CloudWatch Protokolle verwenden. Wählen Sie dazu auf der Seite Ausführungsdetails des Zustandsautomaten die Registerkarten Überwachung und Protokollierung aus. Auf der Registerkarte Überwachung können Sie die CloudWatch Metriken für Ereignisse anzeigen, z. B. Ausführungsdauer, Ausführungsfehler und fakturierter Speicher. Auf der Registerkarte Protokollierung können Sie die aktuellen Protokolle und die Protokollierungskonfiguration anzeigen.

> **Tip**
>
> Ein Beispiel für einen Express-Workflow in Ihrem bereitstellen AWS-Konto und erfahren Sie, wie Sie Express-Workflows überwachen können, im Modul Monitoring Express Workflows des - AWS Step Functions Workshops.

EventBridge Nutzlasten

Ein EventBridge Ereignis kann eine Eingabeeigenschaft in seiner Definition enthalten. Bei einigen Ereignissen kann ein EventBridge Ereignis auch eine Ausgabeeigenschaft in seiner Definition enthalten.

- Wenn die kombinierte Escape-Eingabe und die an gesendete Escape-Ausgabe 248KB EventBridge überschreiten, wird die Eingabe ausgeschlossen. Wenn die Escape-Ausgabe 248KB

überschreitet, wird die Ausgabe ebenfalls ausgeschlossen. Dies ist das Ergebnis der EventBridge Ereigniskontingente.

- Sie können mit den `outputDetails` Eigenschaften `inputDetails` und feststellen, ob eine Nutzlast gekürzt wurde. Weitere Informationen finden Sie unter `CloudWatchEventsExecutionDataDetails` Datentyp .
- Für Standard-Workflows können Sie die vollständige Eingabe und Ausgabe mithilfe von anzeigenDescribeExecution.
- `DescribeExecution` ist für Express Workflows nicht verfügbar. Wenn Sie die vollständige Eingabe/Ausgabe sehen möchten, können Sie Ihren Express-Workflow mit einem Standard-Workflow umschließen. Eine weitere Option ist die Verwendung von Amazon S3-ARNs . Informationen zur Verwendung von ARNs finden Sie unter the section called “Verwenden Sie Amazon S3 S3-ARNs, anstatt große Nutzlasten weiterzuleiten”.

Themen

Step-Functions-Ereignisbeispiele

Im Folgenden finden Sie Beispiele für das Senden von Step Functions-Ereignissen an EventBridge:

Themen

In jedem Fall bietet der Abschnitt `detail` in den Ereignisdaten die gleichen Informationen wie die `DescribeExecution` API. Das Feld `status` gibt den Status der Ausführung zu dem Zeitpunkt an, zu dem das Ereignis gesendet wurde, je nach ausgelöstem Ereignis werden folgende Status angezeigt: `RUNNING`, `SUCCEEDED`, `FAILED`, `TIMED_OUT` oder `ABORTED`.

Ausführung gestartet

```
{
    "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
    }
}
```

Ausführung erfolgreich

```
{
    "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
        }
    }
}
```

Ausführung fehlgeschlagen

```
{
    "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
    }
}
```

Zeitüberschreitung bei der Ausführung

```
{
    "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
    }
}
```

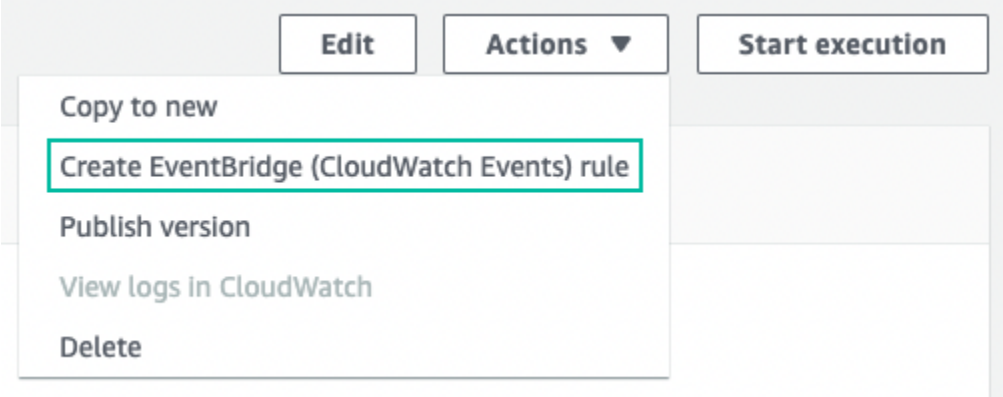
Ausführung abgebrochen

```
{
    "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
    }
}
```

Weiterleiten eines Step-Functions-Ereignisses an EventBridge in der EventBridge Konsole

Verwenden Sie die folgenden Anweisungen, um zu erfahren, wie Sie die Ausführung eines Step-Functions-Zustandsautomaten auslösen, wenn ein bestimmter Step-Functions-Zustandsautomat erfolgreich abgeschlossen wird. Sie verwenden die Amazon- EventBridge Konsole, um den Zustandsautomaten anzugeben, dessen Ausführung Sie auslösen möchten.

1. Wählen Sie auf der Seite Details eines Zustandsautomaten Aktionen und dann EventBridge (CloudWatch Ereignisse) Regel erstellen aus.



   Alternativ können Sie die - EventBridge Konsole unter https://console.aws.amazon.com/events/ öffnen. Wählen Sie im Navigationsbereich unter Buses die Option Regeln aus.
2. Wählen Sie Regel erstellen aus. Dadurch wird die Seite Regeldetail definieren geöffnet.
3. Geben Sie einen Name nfür Ihre Regel ein (z. B. `StepFunctionsEventRule`) und geben Sie optional eine Beschreibung für die Regel ein.
4. Behalten Sie für Event Bus und Regeltyp die Standardauswahl bei.
5. Wählen Sie Weiter aus. Dadurch wird die Seite Ereignismuster erstellen geöffnet.
6. Behalten Sie unter Ereignisquelle die Standardauswahl von AWS Ereignissen oder EventBridge Partnerereignissen bei.
7. Behalten Sie die Standardauswahl für die Abschnitte Beispielereignis und Erstellungsmethode bei.
8. Gehen Sie unter Ereignismuster wie folgt vor:
   a. Behalten Sie in der Dropdownliste Ereignisquelle die Standardauswahl der AWS Services bei.
   b. Wählen Sie in der Dropdownliste AWS Service die Option Step Functions aus.
   c. Wählen Sie in der Dropdown-Liste Ereignistyp die Option Änderung des Ausführungsstatus von Step Functions aus.
   d. (Optional) Konfigurieren Sie einen bestimmten Status, den Amazon-Ressourcennamen (ARN) des Zustandsautomaten oder den Ausführungs-ARN. Wählen Sie für dieses Verfahren Spezifischer Status(e) und dann SUCCEEDED aus der Dropdown-Liste aus.
9. Wählen Sie Weiter aus. Dadurch wird die Seite Ziel(e) auswählen geöffnet.
10. Behalten Sie unter Zieltypen die Standardauswahl des AWS Services bei.

11. Wählen Sie in der Dropdownliste Ziel auswählen einen - AWS Service aus. Sie können beispielsweise eine Lambda-Funktion starten oder einen Step-Functions-Zustandsautomaten ausführen. Wählen Sie für dieses Verfahren Step Functions State Machine aus.
12. Wählen Sie in der Dropdownliste Zustandsautomat einen Zustandsautomaten aus.
13. Behalten Sie unter Ausführungsrolle die Standardauswahl von Neue Rolle für diese spezifische Ressource erstellen bei.
14. Wählen Sie Weiter aus. Dadurch wird die Seite Tags konfigurieren geöffnet.
15. Wählen Sie erneut Weiter aus. Dadurch wird die Seite Überprüfen und erstellen geöffnet.
16. Überprüfen Sie die Details der Regel und wählen Sie dann Create rule (Regel erstellen) aus.

    Die Regel wird erstellt und die Seite Regeln wird angezeigt, auf der alle Ihre Amazon-EventBridge Regeln aufgeführt sind.

API-Aufrufe aufzeichnen mit AWS CloudTrail

AWS Step Functions ist in einen Dienst integriert AWS CloudTrail, der eine Aufzeichnung der von einem Benutzer, einer Rolle oder einem ausgeführten Aktionen bereitstellt AWS-Service. CloudTrail erfasst alle API-Aufrufe Step Functions als Ereignisse. Zu den erfassten Aufrufen gehören Aufrufe von der Step Functions Konsole und Codeaufrufen für die Step Functions API-Operationen. Anhand der von gesammelten Informationen können Sie die Anfrage CloudTrail, an die die Anfrage gestellt wurde Step Functions, die IP-Adresse, von der aus die Anfrage gestellt wurde, den Zeitpunkt der Anfrage und weitere Details ermitteln.

Jeder Ereignis- oder Protokolleintrag enthält Informationen zu dem Benutzer, der die Anforderung generiert hat. Die Identitätsinformationen unterstützen Sie bei der Ermittlung der folgenden Punkte:

- Ob die Anfrage mit Anmeldeinformationen des Root-Benutzers oder des Benutzers gestellt wurde.
- Ob die Anfrage im Namen eines IAM Identity Center-Benutzers gestellt wurde.
- Gibt an, ob die Anforderung mit temporären Sicherheitsanmeldeinformationen für eine Rolle oder einen Verbundbenutzer gesendet wurde.
- Ob die Anforderung aus einem anderen AWS-Service gesendet wurde.

CloudTrail ist in Ihrem aktiv AWS-Konto , wenn Sie das Konto erstellen, und Sie haben automatisch Zugriff auf den CloudTrail Eventverlauf. Der CloudTrail Ereignisverlauf bietet eine einsehbare, durchsuchbare, herunterladbare und unveränderliche Aufzeichnung der aufgezeichneten

Verwaltungsereignisse der letzten 90 Tage in einem. AWS-Region Weitere Informationen finden Sie im AWS CloudTrail Benutzerhandbuch unter Arbeiten mit dem CloudTrail Ereignisverlauf. Für die Anzeige des Eventverlaufs CloudTrail fallen keine Gebühren an.

Für eine fortlaufende Aufzeichnung der Ereignisse in AWS-Konto den letzten 90 Tagen erstellen Sie einen Trail- oder CloudTrailLake-Event-Datenspeicher.

CloudTrail Pfade

Ein Trail ermöglicht CloudTrail die Übermittlung von Protokolldateien an einen Amazon S3 S3-Bucket. Alle mit dem erstellten Pfade AWS Management Console sind regionsübergreifend. Sie können einen Pfad mit einer oder mehreren Regionen erstellen, indem Sie den verwenden. AWS CLI Es wird empfohlen, einen Trail mit mehreren Regionen zu erstellen, da Sie alle Aktivitäten in Ihrem Konto AWS-Regionen erfassen. Wenn du einen Trail mit nur einer Region erstellst, kannst du dir nur die Ereignisse ansehen, die in den Trails protokolliert wurden. AWS-Region Weitere Informationen zu Trails finden Sie unter Einen Trail für Sie erstellen AWS-Konto und Einen Trail für eine Organisation erstellen im AWS CloudTrail Benutzerhandbuch.

Sie können eine Kopie Ihrer laufenden Verwaltungsereignisse kostenlos an Ihren Amazon S3 S3-Bucket senden, CloudTrail indem Sie einen Trail erstellen. Es fallen jedoch Amazon S3 S3-Speichergebühren an. Weitere Informationen zur CloudTrail Preisgestaltung finden Sie unter AWS CloudTrail Preise. Informationen zu Amazon-S3-Preisen finden Sie unter Amazon S3-Preise.

CloudTrail Datenspeicher für Ereignisse in Lake

CloudTrail Mit Lake können Sie SQL-basierte Abfragen für Ihre Ereignisse ausführen. CloudTrail Lake konvertiert bestehende Ereignisse im zeilenbasierten JSON-Format in das Apache ORC-Format. ORC ist ein spaltenförmiges Speicherformat, das für den schnellen Abruf von Daten optimiert ist. Die Ereignisse werden in Ereignisdatenspeichern zusammengefasst, bei denen es sich um unveränderliche Sammlungen von Ereignissen handelt, die auf Kriterien basieren, die Sie mit Hilfe von erweiterten Ereignisselektoren auswählen. Die Selektoren, die Sie auf einen Ereignisdatenspeicher anwenden, steuern, welche Ereignisse bestehen bleiben und für Sie zur Abfrage verfügbar sind. Weitere Informationen zu CloudTrail Lake finden Sie unter Arbeiten mit AWS CloudTrail Lake im AWS CloudTrail Benutzerhandbuch.

CloudTrail Für das Speichern und Abfragen von Ereignisdaten in Lake fallen Kosten an. Beim Erstellen eines Ereignisdatenspeichers wählen Sie die Preisoption aus, die für den Ereignisdatenspeicher genutzt werden soll. Die Preisoption bestimmt die Kosten für die Erfassung und Speicherung von Ereignissen sowie die standardmäßige und maximale Aufbewahrungsdauer

für den Ereignisdatenspeicher. Weitere Informationen zur Preisgestaltung finden Sie unter CloudTrail AWS CloudTrail Preisgestaltung.

Datenereignisse in CloudTrail

Datenereignisse liefern Informationen über die Ressourcenoperationen, die auf oder in einer Ressource ausgeführt werden (z. B. Lesen oder Schreiben in ein Amazon-S3-Objekt). Sie werden auch als Vorgänge auf Datenebene bezeichnet. Datenereignisse sind oft Aktivitäten mit hohem Volume. Protokolliert standardmäßig CloudTrail keine Datenereignisse. Der CloudTrail Ereignisverlauf zeichnet keine Datenereignisse auf.

Für Datenereignisse werden zusätzliche Gebühren fällig. Weitere Informationen zur CloudTrail Preisgestaltung finden Sie unter AWS CloudTrail Preisgestaltung.

Sie können Datenereignisse für die Step Functions Ressourcentypen mithilfe der CloudTrail Konsole oder CloudTrail API-Operationen protokollieren. AWS CLI Weitere Informationen zum Protokollieren von Datenereignissen finden Sie unter Protokollieren von Datenereignissen mit der AWS Management Console und Protokollieren von Datenereignissen mit dem AWS Command Line Interface im AWS CloudTrail Benutzerhandbuch.

In der folgenden Tabelle sind die Step Functions Ressourcentypen aufgeführt, für die Sie Datenereignisse protokollieren können. In der Spalte Datenereignistyp wird der Wert angezeigt, den Sie in der Liste Datenereignistyp auf der CloudTrail Konsole auswählen können. In der Wertspalte resources.type wird der `resources.type` Wert angezeigt, den Sie angeben würden, wenn Sie erweiterte Event-Selektoren mithilfe der AWS CLI APIs oder konfigurieren würden. CloudTrail In der CloudTrail Spalte „Protokollierte Daten-APIs“ werden die API-Aufrufe angezeigt, die CloudTrail für den Ressourcentyp protokolliert wurden.

Sie können erweiterte Event-Selektoren so konfigurieren, dass sie nach den `resources.ARN` Feldern `eventNamereadOnly`, und filtern, sodass nur die Ereignisse protokolliert werden, die für Sie wichtig sind. Weitere Informationen zu diesen Feldern finden Sie AdvancedFieldSelectorin der AWS CloudTrail API-Referenz.

Typ des Datenereignisses	resources.type-Wert	Daten-APIs, bei denen Sie angemeldet sind CloudTrail
Step-Functions-Zustandsautomat	`AWS::StepFunctions::StateMachine`	• InvokeHTTPEndpoint

Verwaltungsereignisse in CloudTrail

Verwaltungsereignisse enthalten Informationen zu Verwaltungsvorgängen, die an Ressourcen in Ihrem ausgeführt werden AWS-Konto. Sie werden auch als Vorgänge auf Steuerebene bezeichnet. In der Standardeinstellung werden Verwaltungsereignisse CloudTrail protokolliert.

State Machine (Zustandsautomat)

- CreateStateMachine
- ListStateMachines
- DescribeStateMachine
- UpdateStateMachine
- DeleteStateMachine
- ValidateStateMachineDefinition
- TestState

Alias der Zustandsmaschine

- CreateStateMachineAlias
- ListStateMachineAliases
- DescribeStateMachineAlias
- UpdateStateMachineAlias
- DeleteStateMachineAlias

Zustandsmaschinen-Version

- ListStateMachineVersions
- PublishStateMachineVersion
- DeleteStateMachineVersion

Hinrichtungen

- StartExecution
- StartSyncExecution

- RedriveExecution
- ListExecutions
- DescribeExecution
- GetExecutionHistory
- DescribeStateMachineForExecution
- StopExecution

Aktivität

- CreateActivity
- ListActivities
- DescribeActivity
- DeleteActivity
- GetActivityTask

Aufgaben-Token

- SendTaskSuccess
- SendTaskHeartbeat
- SendTaskFailure

MapRun

- ListMapRuns
- DescribeMapRun
- UpdateMapRun

Tags

- ListTagsForResource
- TagResource
- UntagResource

Beispiele für Ereignisse

Ein Ereignis stellt eine einzelne Anfrage aus einer beliebigen Quelle dar und enthält Informationen über den angeforderten API-Vorgang, Datum und Uhrzeit des Vorgangs, Anforderungsparameter usw. CloudTrail Protokolldateien sind kein geordneter Stack-Trace der öffentlichen API-Aufrufe, sodass Ereignisse nicht in einer bestimmten Reihenfolge angezeigt werden.

Das folgende Beispiel zeigt ein CloudTrail Datenereignis, das demonstriert`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"
}
```

Das folgende Beispiel zeigt ein CloudTrail Verwaltungsereignis, das den `CreateStateMachine` Vorgang demonstriert.


```
{
    "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"
}
```

Informationen zu CloudTrail Datensatzinhalten finden Sie im AWS CloudTrail Benutzerhandbuch unter CloudTrailDatensatzinhalt.

Protokollierung mitCloudWatchProtokolle

Standard-Workflows zeichnen die Ausführungshistorie auf inAWS Step Functions, obwohl Sie optional die Protokollierung bei Amazon konfigurieren könnenCloudWatchProtokolle.

Anders als Standard-Workflows zeichnen Express-Workflows den Ausführungsverlauf nicht in AWS Step Functions auf. Um den Ausführungsverlauf und die Ergebnisse für einen Express-Workflow zu sehen, müssen Sie die Protokollierung bei Amazon konfigurierenCloudWatchProtokolle. Durch das Veröffentlichen von Protokollen werden die Ausführungen nicht blockiert oder verlangsamt.

> **Note**
>
> Wenn Sie die Protokollierung konfigurieren,CloudWatchLoggt Gebührenwird beantragt und Ihnen wird der Preis für verkaufte Logs in Rechnung gestellt. Weitere Informationen finden Sie unterVerkaufte Baumstämmeunter demProtokolleTab auf derCloudWatchSeite mit den Preisen.

Konfigurieren der -Protokollierung

Wenn Sie mit der Step Functions-Konsole einen Standard-Workflow erstellen, wird dieser nicht so konfiguriert, dass er die Protokollierung ermöglichtCloudWatchProtokolle. Ein mit der Step Functions-Konsole erstellter Express-Workflow wird standardmäßig so konfiguriert, dass er die Protokollierung ermöglichtCloudWatchProtokolle.

Für Express-Workflows kann Step Functions eine Rolle mit den erforderlichenAWS Identity and Access Management(IAM) -Richtlinie fürCloudWatchProtokolle. Wenn Sie einen Standard-Workflow oder einen Express-Workflow mithilfe der API, CLI oderAWS CloudFormation, Step Functions aktiviert die Protokollierung standardmäßig nicht, und Sie müssen sicherstellen, dass Ihre Rolle über die erforderlichen Berechtigungen verfügt.

Für jede von der Konsole aus gestartete Ausführung bietet Step Functions einen Link zuCloudWatchProtokolle, konfiguriert mit dem richtigen Filter, um Protokollereignisse abzurufen, die für diese Ausführung spezifisch sind.

Um die Protokollierung zu konfigurieren, können Sie dieLoggingConfigurationParameter bei VerwendungCreateStateMachineoderUpdateStateMachine. Sie können Ihre Daten weiter analysieren inCloudWatchLoggt mithilfe vonCloudWatchLoggt Einblicke. Weitere Informationen finden Sie unterAnalysieren von Protokolldaten mitCloudWatchLoggt Einblicke.

CloudWatchProtokolliert Payloads

Ereignisse in der Ausführungshistorie können in ihren Definitionen entweder Eingabe- oder Ausgabeeigenschaften enthalten. Wenn die Eingabe maskiert oder die Ausgabe maskiert wurde anCloudWatchLogs überschreiten 248 KB und werden aufgrund vonCloudWatchProtokolliert Kontingente.

- Sie können feststellen, ob eine Payload gekürzt wurde, indem Sie sich die`inputDetails`und`outputDetails`Eigenschaften. Weitere Informationen finden Sie auf der`HistoryEventExecutionDataDetails`Datentyp.
- Für Standard-Workflows können Sie den vollständigen Ausführungsverlauf einsehen, indem Sie`GetExecutionHistory`.
- `GetExecutionHistory`ist nicht für Express Workflows verfügbar. Wenn Sie die vollständige Eingabe und Ausgabe sehen möchten, können Sie Amazon S3-ARNs verwenden. Weitere Informationen finden Sie unter the section called “Verwenden Sie Amazon S3 S3-ARNs, anstatt große Nutzlasten weiterzuleiten”.

IAM-Richtlinien für die Anmeldung beiCloudWatchProtokolle

Sie müssen außerdem die Ausführungs-IAM-Rolle Ihres State-Computers konfigurieren, damit Sie über die erforderlichen Anmeldeberechtigungen verfügenCloudWatchLoggt, wie im folgenden Beispiel gezeigt.

Beispiel für eine IAM-Richtlinie

Im Folgenden finden Sie eine Beispielrichtlinie, mit der Sie Ihre Berechtigungen konfigurieren können. Wie im folgenden Beispiel gezeigt, müssen Sie Folgendes angeben*in der`Resource`Feld weilCloudWatchAPI-Aktionen wie CreateLogDelivery und DescribeLogGroups, unterstütze

nichtRessourcentypen definiert durchAmazon CloudWatch Logs. Weitere Informationen finden Sie unterAktionen definiert vonAmazon CloudWatch Logs.

- Für Informationen überCloudWatchRessourcen, siheCloudWatch LogsRessourcen und Abläufein derAmazonasCloudWatchBenutzerleitfaden.
- Für Informationen zu den Berechtigungen müssen Sie das Senden von Protokollen einrichten anCloudWatchProtokolle, sieheBenutzerberechtigungenim Abschnitt mit dem TitelProtokolle gesendet anCloudWatch 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": "*"
        }
    ]
}
```

Zugriff auf die nicht möglichCloudWatchProtokolle

Wenn Sie nicht auf die zugreifen könnenCloudWatchLogs, stellen Sie sicher, dass Sie Folgendes getan haben:

1. Die Execution-IAM-Rolle Ihres State Computers wurde so konfiguriert, dass Sie über die entsprechenden Anmeldeberechtigungen verfügenCloudWatchProtokolle.

Wenn Sie das verwendenCreateStateMachineoderUpdateStateMachineAnfragen, stellen Sie sicher, dass Sie die IAM-Rolle in der angegeben habenroleArnParameter, der die Berechtigungen enthält, wie in dervorheriges Beispiel.

2. Habe das überprüftCloudWatchDie Ressourcenrichtlinie für Protokolle überschreitet nicht das Limit von 5120 Zeichen fürCloudWatchProtokolliert Ressourcenrichtlinien.

   Wenn Sie das Zeichenlimit überschritten haben, entfernen Sie unnötige Berechtigungen aus demCloudWatchProtokolliert die Ressourcenrichtlinie oder stellen Sie dem Namen der Protokollgruppe das Präfix`/aws/vendedlogs`, wodurch der Protokollgruppe Berechtigungen erteilt werden, ohne dass weitere Zeichen an die Ressourcenrichtlinie angehängt werden. Wenn Sie in der Step Functions-Konsole eine Protokollgruppe erstellen, erhalten die Protokollgruppennamen das Präfix`/aws/vendedlogs/states`. Weitere Informationen finden Sie unter Größenbeschränkungen der Amazon CloudWatch Logs-Ressourcenrichtlinie.

Protokollstufen

Sie können zwischen `OFF`, `ALL`, `ERROR` oder `FATAL` wählen. Bei der Einstellung `OFF` werden keine Ereignistypen protokolliert, und bei der Einstellung `ALL` werden alle Ereignistypen protokolliert. Informationen zu `ERROR` und `FATAL` finden Sie in der folgenden Tabelle.

Weitere Informationen zu den Ausführungsdaten, die für darauf basierende Express Workflow-Ausführungen angezeigt werdenEbenen protokollieren, sieheStandard- und Express-Workflow-Ausführungen in der Konsole.

Ereignistyp	**ALL**	**ERROR**	**FATAL**	**OFF**
ChoiceStateEntered	✓			
ChoiceStateExited	✓			
ExecutionAborted	✓	✓	✓	
ExecutionFailed	✓	✓	✓	

Ereignistyp	ALL	ERROR	FATAL	OFF
ExecutionStarted	✓			
Execution Succeeded	✓			
Execution TimedOut	✓	✓	✓	
FailStateEntered	✓	✓		
LambdaFunctionFailed	✓	✓		
LambdaFunctionScheduled	✓			
LambdaFunctionScheduleFailed	✓	✓		
LambdaFunctionStarted	✓			
LambdaFunctionStartFailed	✓	✓		
LambdaFunctionSucceeded	✓			
LambdaFunctionTimedOut	✓	✓		
MapIterationAborted	✓	✓		
MapIterationFailed	✓	✓		

Ereignistyp	ALL	ERROR	FATAL	OFF
MapIterationStarted	✓			
MapIterationSucceeded	✓			
MapRunAborted	✓	✓		
MapRunFailed	✓	✓		
MapStateAborted	✓	✓		
MapStateEntered	✓			
MapStateExited	✓			
MapStateFailed	✓	✓		
MapStateStarted	✓			
MapStateSucceeded	✓			
ParallelStateAborted	✓	✓		
ParallelStateEntered	✓			
ParallelStateExited	✓			
ParallelStateFailed	✓	✓		

Ereignistyp	ALL	ERROR	FATAL	OFF
ParallelStateStarted	✓			
ParallelStateSucceeded	✓			
PassStateEntered	✓			
PassStateExited	✓			
SucceedStateEntered	✓			
SucceedStateExited	✓			
TaskFailed	✓	✓		
TaskScheduled	✓			
TaskStarted	✓			
TaskStartFailed	✓	✓		
TaskStateAborted	✓	✓		
TaskStateEntered	✓			
TaskStateExited	✓			
TaskSubmitFailed	✓	✓		
TaskSubmitted	✓			
TaskSucceeded	✓			

Ereignistyp	ALL	ERROR	FATAL	OFF
TaskTimedOut	✓	✓		
WaitState Aborted	✓	✓		
WaitState Entered	✓			
WaitStateExited	✓			

AWS X-Ray und Step Functions

Sie können verwenden, AWS X-Ray um die Komponenten Ihres Zustandsautomaten zu visualisieren, Leistungsengpässe zu identifizieren und Anforderungen zu beheben, die zu einem Fehler geführt haben. Ihr Zustandsautomat sendet Trace-Daten an X-Ray und X-Ray verarbeitet die Daten, um eine Service-Übersicht und durchsuchbare Trace-Zusammenfassungen zu generieren.

Wenn X-Ray für Ihren Zustandsautomaten aktiviert ist, können Sie Anforderungen verfolgen, während sie in Step Functions in allen AWS Regionen ausgeführt werden, in denen X-Ray verfügbar ist. Auf diese Weise erhalten Sie einen detaillierten Überblick über eine gesamte Step-Functions-Anforderung. Step Functions sendet Ablaufverfolgungen an X-Ray für Ausführungen von Zustandsautomaten, auch wenn eine Ablaufverfolgungs-ID nicht von einem Upstream-Service übergeben wird. Sie können eine X-Ray-Service-Übersicht verwenden, um die Latenz einer Anforderung anzuzeigen, einschließlich aller AWS Services, die in X-Ray integriert sind. Sie können auch Samplingregeln konfigurieren, um X-Ray mitzuteilen, welche Anforderungen gemäß den von Ihnen angegebenen Kriterien aufgezeichnet werden sollen und mit welchen Sampling-Raten.

Wenn X-Ray für Ihren Zustandsautomaten nicht aktiviert ist und ein Upstream-Service keine Ablaufverfolgungs-ID übergibt, sendet Step Functions keine Ablaufverfolgungen für die Ausführung von Zustandsautomaten an X-Ray. Wenn jedoch eine Ablaufverfolgungs-ID von einem Upstream-Service übergeben wird, sendet Step Functions Ablaufverfolgungen für Zustandsautomatenausführungen an X-Ray.

Sie können AWS X-Ray mit Step Functions in Regionen verwenden, in denen beide unterstützt werden. Informationen zur Unterstützung von Regionen für X-Ray und Step Functions finden Sie auf den Seiten Step Functions und X-Ray-Endpunkte und -Kontingente.

Kombinierte Kontingente für X-Ray und Step Functions

Sie können Daten bis zu sieben Tage lang zu einem Trace hinzufügen und Trace-Daten abfragen, die auf einunddreißig Tage zurückreichen. Dies ist die Zeitspanne, in der X-Ray Trace-Daten speichert. Ihre Ablaufverfolgungen unterliegen X-Ray-Kontingenten. Zusätzlich zu anderen Kontingenten bietet X-Ray eine garantierte Mindestverfolgungsgröße von 100KB für Step-Functions-Zustandsautomaten. Wenn X-Ray mehr als 100KB Trace-Daten zur Verfügung gestellt werden, kann dies zu einer eingefrorenen Trace führen. Weitere Informationen zu anderen Kontingenten für X-Ray finden Sie im Abschnitt Service Quotas auf der Seite X-Ray-Endpunkte und -Kontingente.

Important

Step Functions unterstützt keine X-Ray-Nachverfolgung für die untergeordneten Workflow-Ausführungen, die mit einem Distributed Map-Status gestartet wurden, da es einfach ist, die Größenbeschränkung für Trace-Dokumente für solche Ausführungen zu überschreiten.

Themen

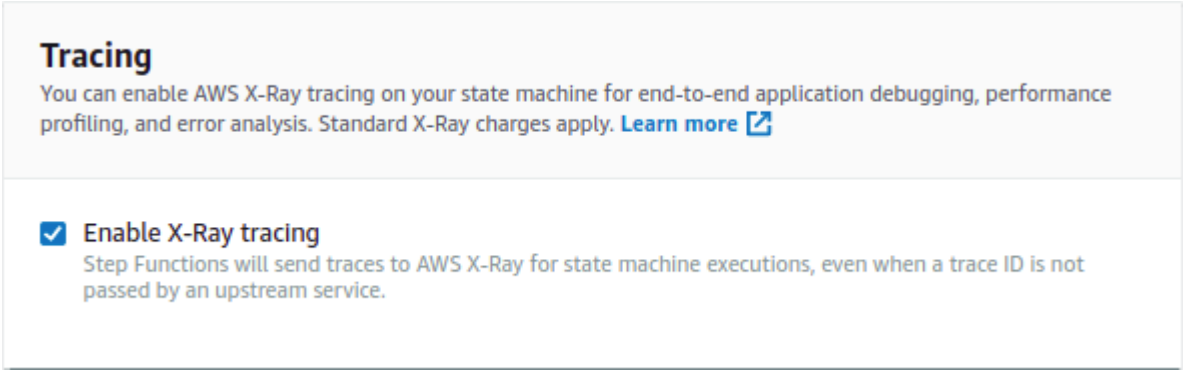
Einrichtung und Konfiguration

Aktivieren der X-Ray-Nachverfolgung beim Erstellen eines Zustandsautomaten

Sie können die X-Ray-Ablaufverfolgung aktivieren, wenn Sie einen neuen Zustandsautomaten erstellen, indem Sie auf der Seite Details angeben die Option X-Ray-Ablaufverfolgung aktivieren auswählen.

1. Öffnen Sie die Step-Functions-Konsole und wählen Sie Zustandsautomaten erstellen aus.
2. Wählen Sie auf der Seite Erstellungsmethode auswählen eine geeignete Option zum Erstellen Ihres Zustandsautomaten aus. Wenn Sie Beispielprojekt ausführen wählen, können Sie die X-Ray-Ablaufverfolgung während der Erstellung des Zustandsautomaten nicht aktivieren, und Sie müssen die X-Ray-Ablaufverfolgung aktivieren, nachdem Ihr Zustandsautomat erstellt wurde. Weitere Informationen zum Aktivieren von X-Ray in einem vorhandenen Zustandsautomaten finden Sie unter Aktivieren von X-Ray in einem vorhandenen Zustandsautomaten.

   Wählen Sie Weiter aus.
3. Konfigurieren Sie auf der Seite Details angeben Ihren Zustandsautomaten.
4. Wählen Sie X-Ray-Nachverfolgung aktivieren aus.



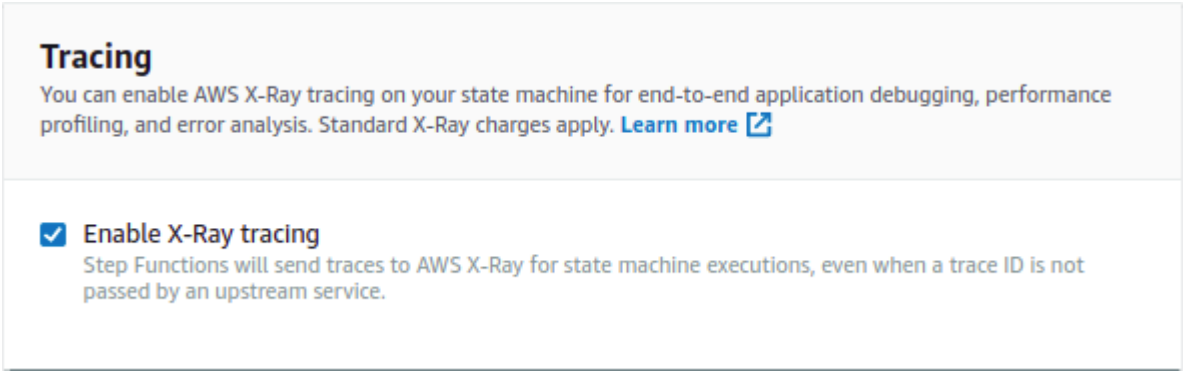
Ihr Step-Functions-Zustandsautomat sendet jetzt Ablaufverfolgungen für Zustandsautomatenausführungen an X-Ray.

> **Note**
>
> Wenn Sie eine vorhandene IAM-Rolle verwenden möchten, sollten Sie sicherstellen, dass X-Ray-Schreibvorgänge zulässig sind. Weitere Informationen zu den erforderlichen Berechtigungen finden Sie unter IAM-Richtlinien für X-Ray .

Aktivieren von X-Ray in einem vorhandenen Zustandsautomaten

So aktivieren Sie X-Ray in einem vorhandenen Zustandsautomaten:

1. Wählen Sie in der Step-Functions-Konsole den Zustandsautomaten aus, für den Sie die Ablaufverfolgung aktivieren möchten.
2. Wählen Sie Bearbeiten aus.
3. Wählen Sie X-Ray-Nachverfolgung aktivieren aus.



   Es wird eine Benachrichtigung angezeigt, in der Sie darüber informiert werden, dass Sie möglicherweise zusätzliche Änderungen vornehmen müssen.

> **Note**
>
> Wenn Sie X-Ray für einen vorhandenen Zustandsautomaten aktivieren, müssen Sie sicherstellen, dass Sie über eine IAM-Richtlinie verfügen, die X-Ray ausreichende Berechtigungen zum Ausführen von Ablaufverfolgungen gewährt. Sie können entweder manuell eine hinzufügen oder eine generieren. Weitere Informationen finden Sie im Abschnitt IAM-Richtlinie für IAM-Richtlinien für AWS X-Ray.

4. (Optional) Generieren Sie automatisch eine neue Rolle für Ihren Zustandsautomaten, um X-Ray-Berechtigungen einzuschließen.
5. Wählen Sie Speichern.

Konfigurieren der X-Ray-Ablaufverfolgung für Step Functions

Wenn Sie zum ersten Mal einen Zustandsautomaten mit aktivierter X-Ray-Nachverfolgung ausführen, werden die Standardkonfigurationswerte für X-Ray Tracing. AWS X-Ray does nicht für jede Anforderung erfasst, die an eine Anwendung gesendet wird. Stattdessen werden Daten für eine

statistisch signifikante Anzahl von Anfragen erfasst. Standardmäßig werden die erste Anfrage pro Sekunde und fünf Prozent aller zusätzlichen Anfragen aufgezeichnet. Eine Anfrage pro Sekunde ist das Reservoir. Dadurch wird sichergestellt, dass jede Sekunde mindestens eine Ablaufverfolgung aufgezeichnet wird, solange der Dienst Anfragen verarbeitet. Fünf Prozent ist die Rate, mit der die über die Reservoirgröße hinausgehenden Anforderungen geprüft werden.

Um zu vermeiden, dass bei den ersten Schritten Servicegebühren anfallen, ist die Standard-Samplingrate konservativ. Sie können X-Ray so konfigurieren, dass die Standard-Samplingregel geändert und zusätzliche Regeln konfiguriert werden, die Sampling basierend auf den Eigenschaften des Services oder der Anforderung anwenden.

Sie können beispielsweise das Sampling deaktivieren und alle Anfragen für Anrufe verfolgen, die den Status ändern oder AWS-Konten - oder -Transaktionen verarbeiten. Bei schreibgeschützten Aufrufen mit hohem Volumen wie Hintergrundabfragen, Zustandsprüfungen oder Verbindungswartungen können Sie eine Stichprobe mit einer niedrigen Rate durchführen und dennoch genügend Daten abrufen, um auftretende Probleme zu beobachten.

So konfigurieren Sie eine Samplingregel für Ihren Zustandsautomaten:

1. Gehen Sie zur X-Ray-Konsole .
2. Wählen Sie Sampling aus.
3. Um eine Regel zu erstellen, wählen Sie Create sampling rule (Samplingregel erstellen) aus.

   Um eine Regel zu bearbeiten, wählen Sie den Namen einer Regel aus.

   Um eine Regel zu löschen, wählen Sie eine Regel aus und verwenden das Menü Actions (Aktionen), um diese zu löschen.

Einige Teile vorhandener Samplingregeln, wie Name und Priorität, können nicht geändert werden. Fügen Sie stattdessen eine vorhandene Regel hinzu oder klonen Sie sie, nehmen Sie die gewünschten Änderungen vor und verwenden Sie dann die neue Regel.

Ausführliche Informationen zu X-Ray-Samplingregeln und zur Konfiguration der verschiedenen Parameter finden Sie unter Konfigurieren von Samplingregeln in der X-Ray-Konsole.

Integrieren von Upstream-Services

Um die Ausführung von Step-Functions-Workflows wie Express-, synchrone und Standard-Workflows in einen Upstream-Service zu integrieren, müssen Sie die festlegen`traceHeader`. Dies erfolgt

automatisch für Sie, wenn Sie eine HTTP-API in API Gateway verwenden. Wenn Sie jedoch eine Lambda-Funktion und/oder ein -SDK verwenden, müssen Sie die `traceHeader` für die -`StartExecution` oder `StartSyncExecution`-API-Aufrufe selbst festlegen.

Sie müssen das `traceHeader` Format als angeben`\p{ASCII}#`. Damit Step Functions dieselbe Ablaufverfolgungs-ID verwenden kann, müssen Sie das Format als angeben`Root={TRACE_ID};Sampled={1 or 0}`. Wenn Sie eine Lambda-Funktion verwenden, ersetzen Sie durch `TRACE_ID` die Ablaufverfolgungs-ID in Ihrem aktuellen Segment und legen Sie das Feld Stichprobe so fest, als `1` ob Ihr Sampling-Modus wahr ist und `0` wenn Ihr Sampling-Modus falsch ist. Die Angabe der Ablaufverfolgungs-ID in diesem Format stellt sicher, dass Sie eine vollständige Ablaufverfolgung erhalten.

Im Folgenden finden Sie ein Beispiel, das in Python geschrieben wurde, um zu demonstrieren, wie die angegeben wird`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)
```

Konzepte

Die X-Ray-Konsole

Mit der AWS X-Ray Konsole können Sie Service-Übersichten und Ablaufverfolgungen für Anforderungen anzeigen, die Ihre Anwendungen bedienen. Sie können auf die -Konsole zugreifen,

um detaillierte Informationen anzuzeigen, die von X-Ray gesammelt werden, wenn es für Ihren Zustandsautomaten aktiviert ist.

Anzeigen der X-Ray-Konsole Informationen zum Zugriff auf die X-Ray-Konsole für Ihre Zustandsautomatenausführungen finden Sie unter .

Ausführliche Informationen zur X-Ray-Konsole finden Sie in der Dokumentation zur X-Ray-Konsole.

Segmente, Untersegmente und Ablaufverfolgungen

Ein Segment zeichnet Informationen zu einer Anforderung an Ihren Zustandsautomaten auf. Es enthält Informationen wie die Arbeit, die Ihr Zustandsautomat ausführt, und kann auch Untersegmente mit Informationen über nachgelagerte Aufrufe enthalten.

Ein Trace erfasst alle Segmente, die von einer einzelnen Anforderung generiert werden.

Sampling

Um eine effiziente Nachverfolgung zu gewährleisten und eine repräsentative Stichprobe der Anforderungen bereitzustellen, die Ihre Anwendung bedient, wendet X-Ray einen Sampling-Algorithmus an, um zu bestimmen, welche Anforderungen verfolgt werden. Dies kann durch Bearbeiten der Samplingregeln geändert werden.

Metriken

Für Ihren Zustandsautomaten misst X-Ray die Aufrufzeit, die Statusübergangszeit, die Gesamtausführungszeit von Step Functions und Abweichungen in dieser Ausführungszeit. Auf diese Informationen kann über die X-Ray-Konsole zugegriffen werden.

Analysen

Die AWS X-Ray Analytics-Konsole ist ein interaktives Tool zur Interpretation von Ablaufverfolgungsdaten. Sie können die aktiven Datasets mit zunehmend granularen Filtern anpassen, indem Sie auf die Diagramme und die Bereiche der Metriken und Felder klicken, die der aktuellen Ablaufverfolgungsreihe zugeordnet sind. Auf diese Weise können Sie analysieren, wie Ihr Zustandsautomat funktioniert, und Leistungsprobleme schnell finden und identifizieren.

Ausführliche Informationen zu X-Ray-Analysen finden Sie unter Interaktion mit der AWS X-Ray Analysekonsole.

Step-Functions-Serviceintegrationen und X-Ray

Einige der AWS Services, die in Step Functions integriert sind, bieten die Integration mit , AWS X-Ray indem sie Anfragen einen Nachverfolgungs-Header hinzufügen, den X-Ray-Daemon ausführen oder Sampling-Entscheidungen treffen und Ablaufverfolgungsdaten in X-Ray hochladen. Andere müssen mit dem AWS X-Ray SDK instrumentiert werden. Einige unterstützen noch keine X-Ray-Integration. Die X-Ray-Integration ist erforderlich, um vollständige Ablaufverfolgungsdaten bereitzustellen, wenn eine Serviceintegration mit Step Functions verwendet wird

Native X-Ray-Unterstützung

Zu den Serviceintegrationen mit nativer X-Ray-Unterstützung gehören:

- Amazon Simple Notification Service
- Amazon Simple Queue Service
- AWS Lambda
- AWS Step Functions

Instrumentierung erforderlich

Serviceintegrationen, die eine X-Ray-Instrumentierung erfordern:

- Amazon Elastic Container Service
- AWS Batch
- AWS Fargate

Nur clientseitige Ablaufverfolgung

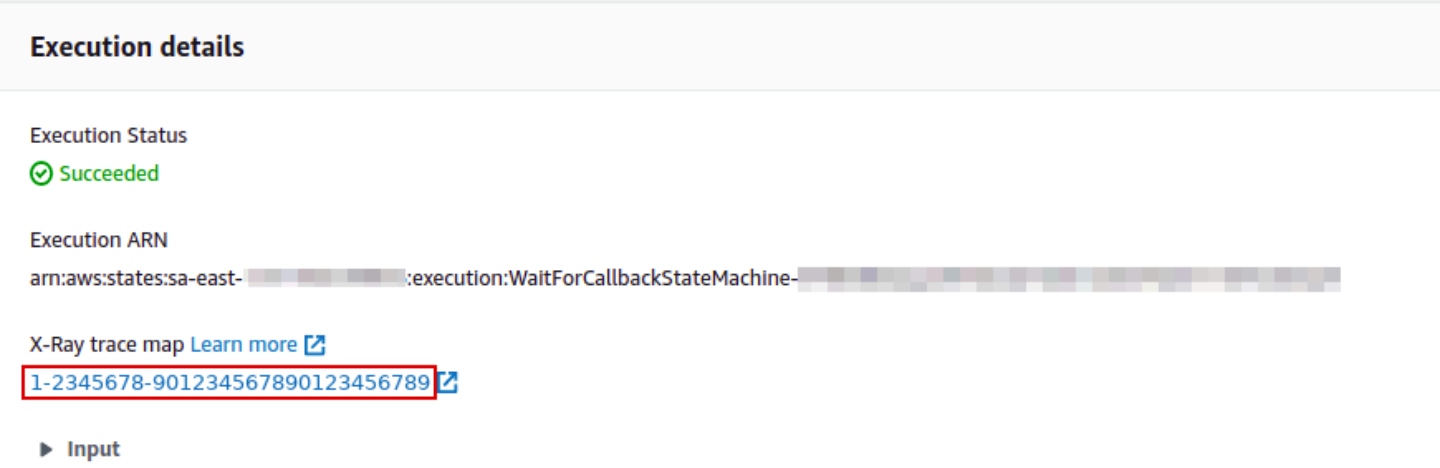
Andere Service-Integrationen unterstützen keine X-Ray-Ablaufverfolgungen. Clientseitige Ablaufverfolgungen können jedoch weiterhin erfasst werden:

- Amazon DynamoDB
- Amazon EMR
- Amazon SageMaker
- AWS CodeBuild
- AWS Glue

Anzeigen der X-Ray-Konsole

X-Ray empfängt Daten von -Services als Segmente. X-Ray gruppiert Segmente, die eine gemeinsame Anforderung haben, in Ablaufverfolgungen. X-Ray verarbeitet die Ablaufverfolgungen, um ein Servicediagramm zu generieren, das eine visuelle Darstellung Ihrer Anwendung bietet.

Nachdem Sie die Ausführung Ihres Zustandsautomaten gestartet haben, können Sie seine X-Ray-Ablaufverfolgungen anzeigen, indem Sie im Abschnitt Ausführungsdetails den Link X-Ray-Ablaufverfolgungszuordnung auswählen.



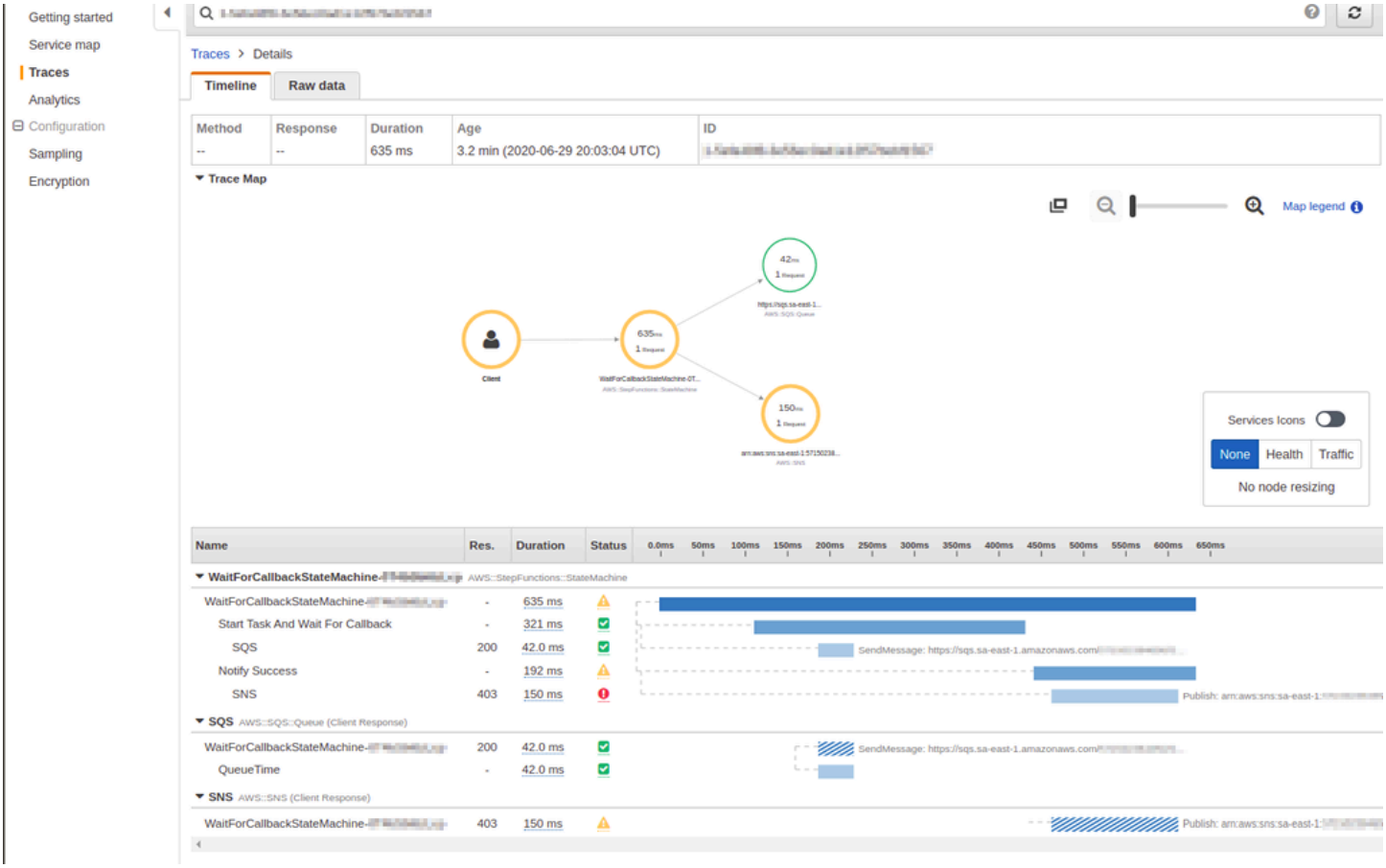
Nachdem Sie X-Ray für Ihren Zustandsautomaten aktiviert haben, können Sie Ablaufverfolgungsinformationen für seine Ausführungen in der X-Ray-Konsole anzeigen.

Anzeigen von X-Ray-Ablaufverfolgungsinformationen für Step Functions

Die folgenden Schritte veranschaulichen, welche Art von Informationen Sie in der Konsole sehen können, nachdem Sie X-Ray aktiviert und eine Ausführung ausgeführt haben. X-Ray-Ablaufverfolgungen für das Beispiel für ein Rückrufmuster (Amazon SQS, Amazon SNS, Lambda) Beispielprojekt werden angezeigt.

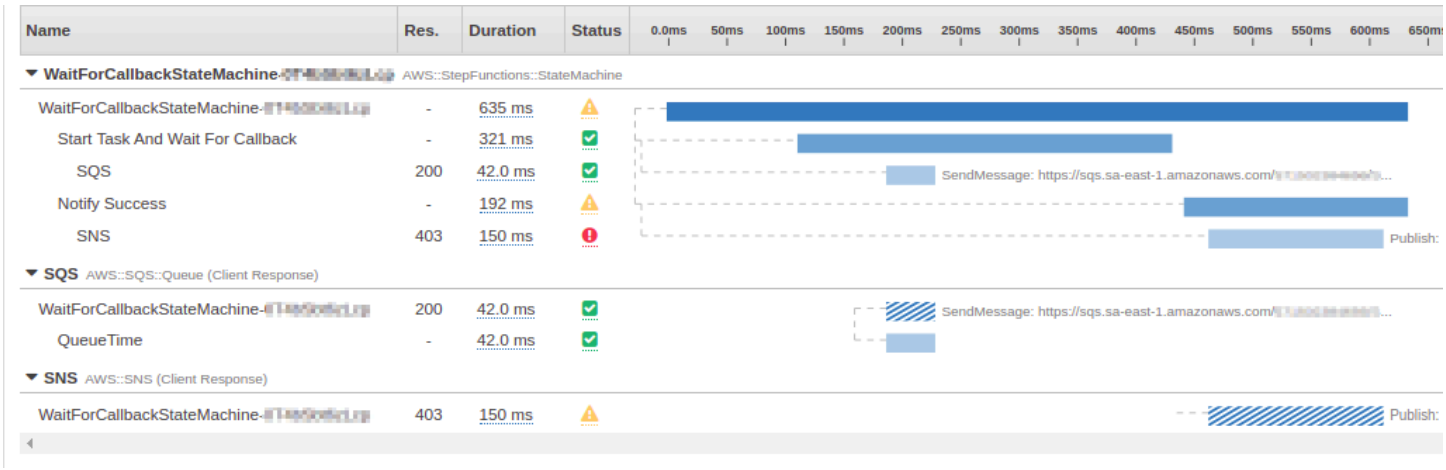
Ablaufverfolgungen

Nachdem die Ausführung abgeschlossen ist, können Sie zur X-Ray-Konsole navigieren, auf der Sie die Seite X-Ray Traces sehen. Dadurch wird eine Übersicht über die Service-Übersicht sowie Ablaufverfolgungs- und Segmentinformationen für Ihren Zustandsautomaten angezeigt.

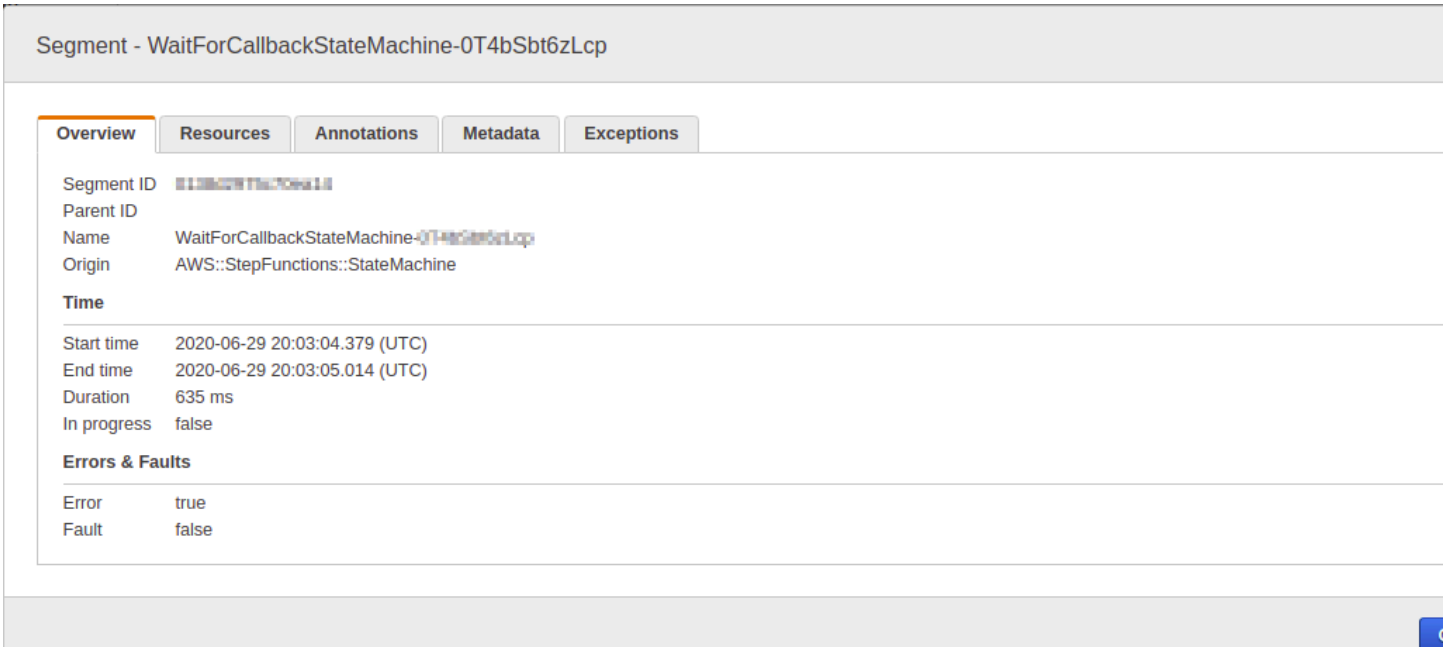


Service-Übersicht

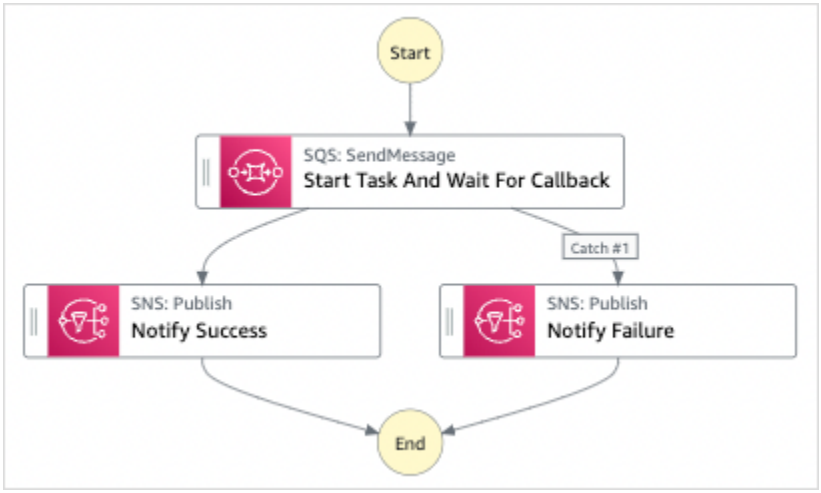
Die Service-Übersicht in der X-Ray-Konsole hilft Ihnen, Services zu identifizieren, bei denen Fehler auftreten, bei denen Verbindungen mit hoher Latenz bestehen oder Ablaufverfolgungen für Anfragen angezeigt werden, die nicht erfolgreich waren.



In der Trace-Übersicht können Sie einen Serviceknoten auswählen, um Anforderungen für diesen Knoten anzuzeigen, oder eine Kante zwischen zwei Knoten, um Anforderungen anzuzeigen, die diese Verbindung zurückgelegt haben. Hier wurde der `WaitForCallBack` Knoten ausgewählt, und Sie können zusätzliche Informationen über seinen Ausführungs- und Antwortstatus anzeigen.



Sie können sehen, wie die X-Ray-Servicekarte mit dem Zustandsautomaten korreliert. Es gibt einen Service-Map-Knoten für jede Service-Integration, die von Step Functions aufgerufen wird, sofern er X-Ray unterstützt.



Segmente und Untersegmente

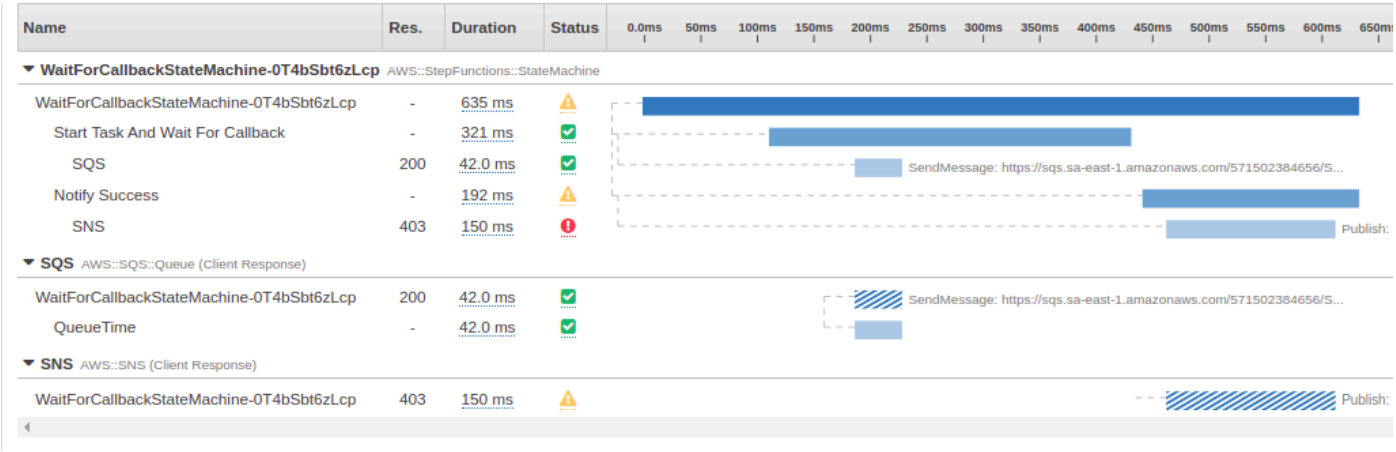
Ein Trace ist eine Sammlung von Segmenten, die von einer einzelnen Anforderung generiert werden. Jedes Segment stellt den Namen der Ressource, Details zur Anforderung und Details zu den

ausgeführten Aufgaben bereit. Auf der Seite Ablaufverfolgungen können Sie die Segmente und, falls erweitert, die entsprechenden Untersegmente sehen. Sie können ein Segment oder Untersegment auswählen, um detaillierte Informationen darüber anzuzeigen.

Wählen Sie jede Registerkarte aus, um zu sehen, wie Informationen für Segmente und Untersegmente angezeigt werden.

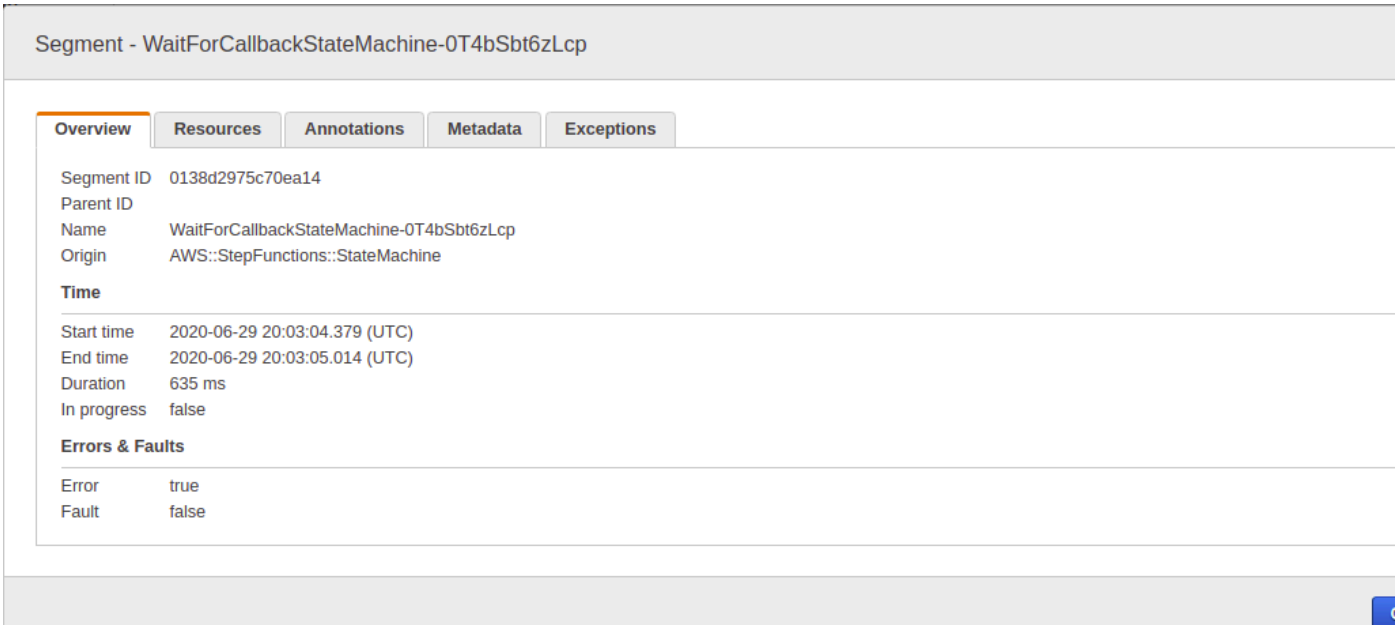
Overview of Segments

Eine Übersicht über Segmente und Untersegmente für diesen Zustandsautomaten. Für jeden Knoten auf der Service-Übersicht gibt es ein anderes Segment.



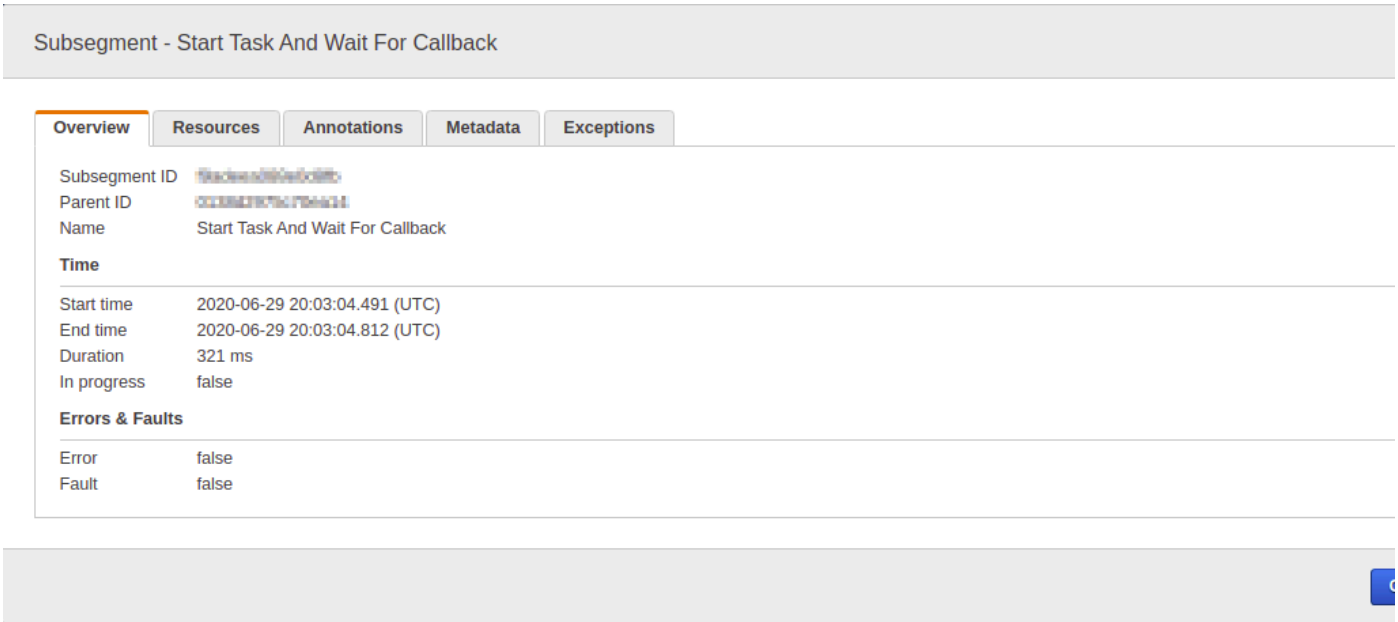
View segment detail

Wenn Sie ein Segment auswählen, werden der Name der Ressource, Details zur Anforderung und Details zu den ausgeführten Aufgaben bereitgestellt.



View subsegment detail

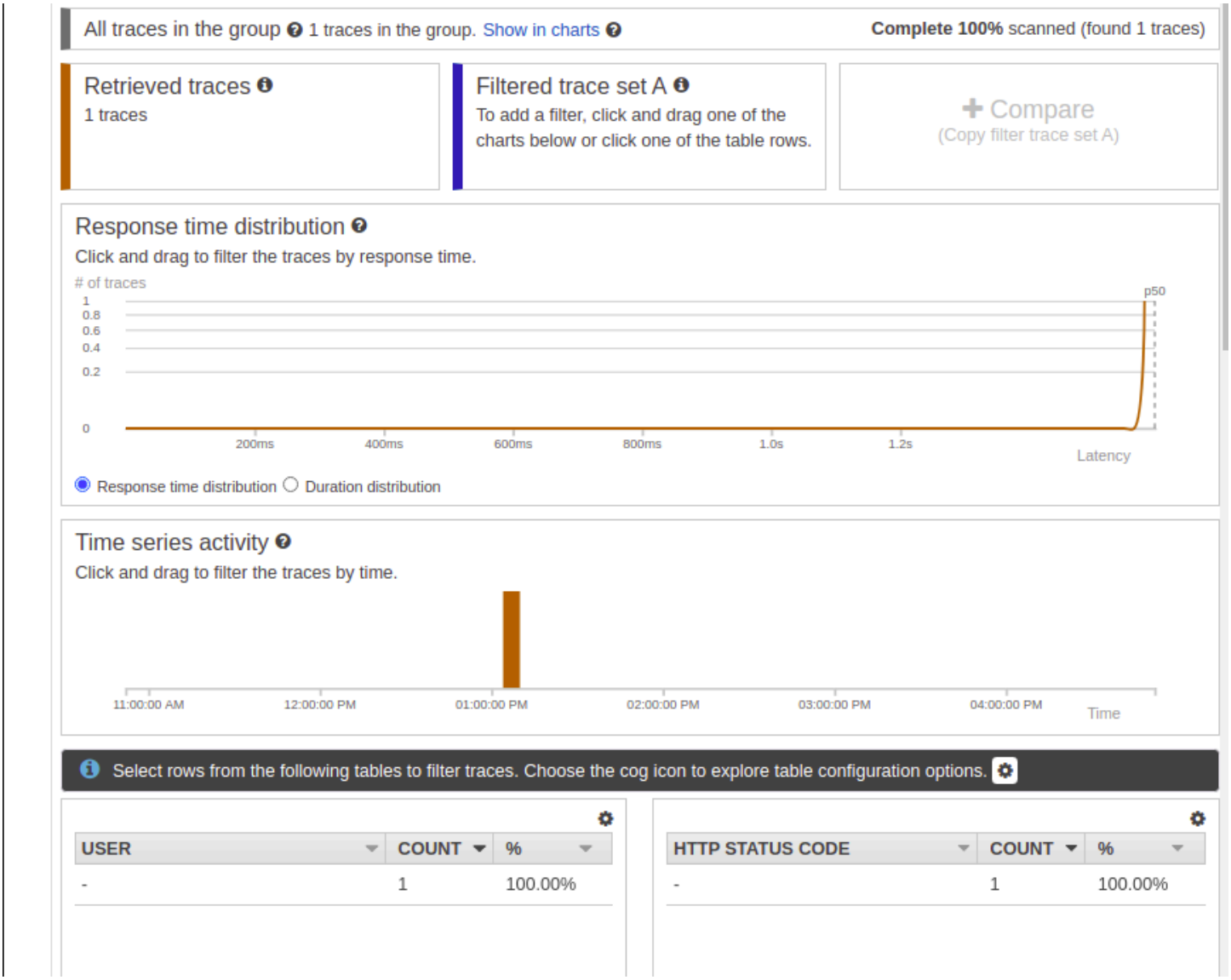
Ein Segment kann die Daten zu der geleisteten Arbeit in Untersegmente unterteilen. Durch die Auswahl eines Untersegments können Sie detailliertere Zeitinformationen und Details anzeigen. Ein Untersegment kann zusätzliche Details zu einem Aufruf an einen - AWS Service, eine externe HTTP-API oder eine SQL-Datenbank enthalten.



Analysen

Die AWS X-Ray Analytics-Konsole ist ein interaktives Tool zur Interpretation von Ablaufverfolgungsdaten. Sie können dies verwenden, um leichter zu verstehen, wie Ihr Zustandsautomat funktioniert. Die Konsole ermöglicht Ihnen das Erkunden, Analysieren und Visualisieren von Ablaufverfolgungen durch interaktive Reaktionszeit und Zeitreihen-Diagramme. Auf diese Weise können Sie Leistungs- und Latenzprobleme schnell finden.

Sie können die aktiven Datasets mit zunehmend granularen Filtern anpassen, indem Sie auf die Diagramme und die Bereiche der Metriken und Felder klicken, die der aktuellen Ablaufverfolgungsreihe zugeordnet sind.

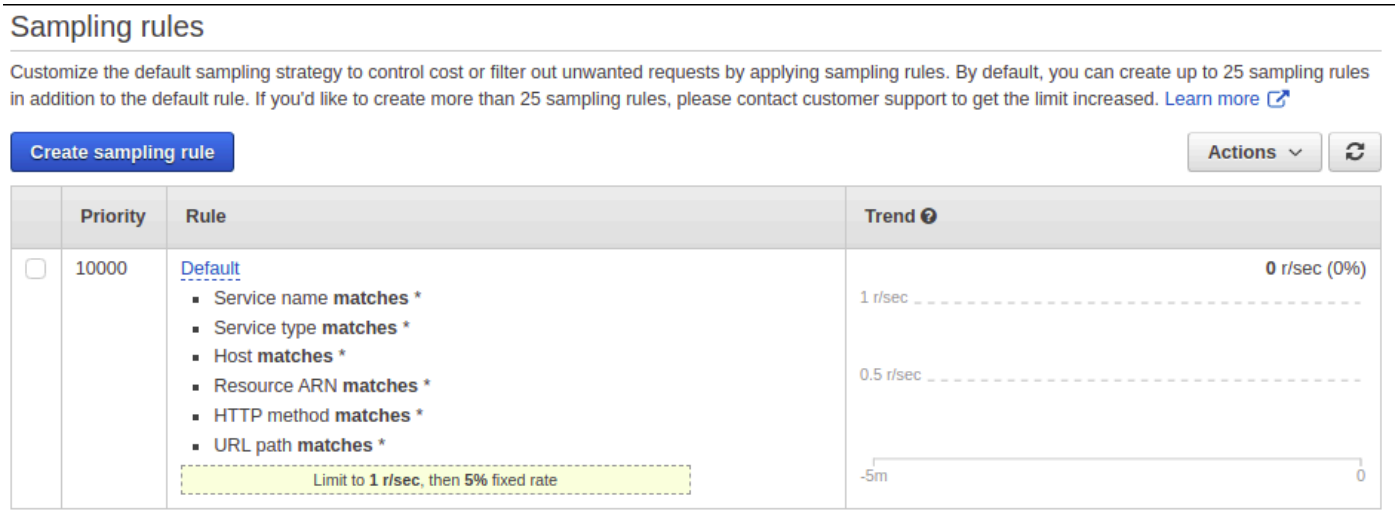


Konfiguration

Sie können Sampling- und Verschlüsselungsoptionen über die X-Ray-Konsole konfigurieren.

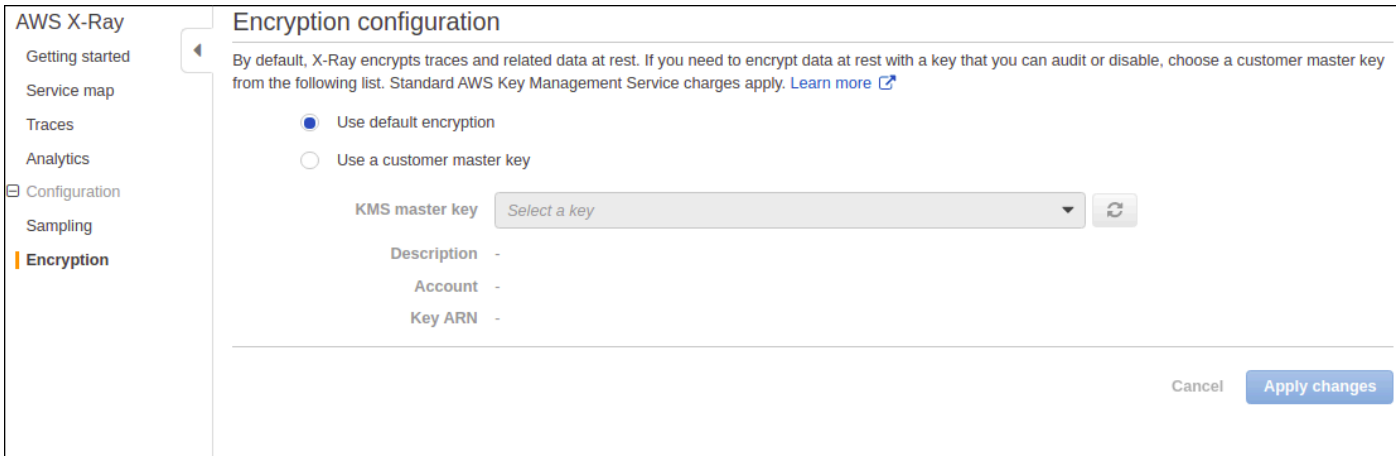
Sampling

Wählen Sie Sampling, um Details zur Abtastrate und Konfiguration anzuzeigen. Sie können die Samplingregeln ändern, um die Menge der von Ihnen aufgezeichneten Daten zu steuern, und das Sampling-Verhalten an Ihre spezifischen Anforderungen anpassen.



Encryption

Wählen Sie Verschlüsselung, um die Verschlüsselungseinstellungen zu ändern. Sie können die Standardeinstellung verwenden, bei der X-Ray Ablaufverfolgungen und das Datum im Ruhezustand verschlüsselt, oder bei Bedarf einen Kundenmasterschlüssel auswählen. Im letzteren Fall fallen AWS KMS Standardgebühren an.



Was ist, wenn die Ablaufverfolgungs- oder Service-Übersicht keine Daten enthält?

Wenn Sie X-Ray aktiviert haben, aber keine Daten in der X-Ray-Konsole sehen können, überprüfen Sie Folgendes:

- Ihre IAM-Rollen sind korrekt eingerichtet, um das Schreiben in X-Ray zu ermöglichen.
- Samplingregeln ermöglichen das Sampling von Daten.

- Da es zu einer kurzen Verzögerung kommen kann, bevor neu erstellte oder geänderte IAM-Rollen angewendet werden, überprüfen Sie die Ablaufverfolgung oder Service-Übersichten nach einigen Minuten erneut.
- Wenn im Bereich X-Ray Traces Daten nicht gefunden angezeigt werden, überprüfen Sie Ihre IAM-Kontoeinstellungen und stellen Sie sicher, dass für die gewünschte Region aktiviert AWS Security Token Service ist. Weitere Informationen finden Sie unter Aktivieren und Deaktivieren von AWS STS in einer AWS-Region im IAM-Benutzerhandbuch.

Verwenden von AWS-Benutzerbenachrichtigungen mit AWS Step Functions

Sie können AWS-BenutzerbenachrichtigungenLieferkanäle einrichten, um über AWS Step Functions Ereignisse informiert zu werden. Sie erhalten eine Benachrichtigung, wenn ein Ereignis einer von Ihnen angegebenen Regel entspricht. Sie können Benachrichtigungen für Ereignisse über mehrere Kanäle erhalten, einschließlich E-Mail-, AWS Chatbot-Chat- oder AWS Console Mobile Application-Push-Benachrichtigungen. Sie können Benachrichtigungen auch im Konsolen-Benachrichtigungscenter anzeigen. Benutzerbenachrichtigungen unterstützt die Aggregation, wodurch die Anzahl der Benachrichtigungen, die Sie bei bestimmten Ereignissen erhalten, verringert werden kann.

Sicherheit in AWS Step Functions

Dieser Abschnitt enthält Informationen zu AWS Step Functions Sicherheit und Authentifizierung.

Step Functions verwendet IAM, um den Zugriff auf andere AWS Dienste und Ressourcen zu steuern. Eine Übersicht der Funktionsweise von IAM finden Sie unter Übersicht über die Zugriffsverwaltung im IAM-Benutzerhandbuch. Eine Übersicht der Sicherheitsanmeldeinformationen finden Sie unter AWS - Sicherheitsanmeldeinformationen in der Allgemeine Amazon Web Services-Referenz.

Datenschutz in AWS Step Functions

Das Modell der AWS gemeinsamen Verantwortung und geteilter Verantwortung gilt für den Datenschutz in AWS Step Functions. Wie in diesem Modell beschrieben, AWS ist verantwortlich für den Schutz der globalen Infrastruktur, auf der alle Systeme laufen AWS Cloud. Sie sind dafür verantwortlich, die Kontrolle über Ihre in dieser Infrastruktur gehosteten Inhalte zu behalten. Sie sind auch für die Sicherheitskonfiguration und die Verwaltungsaufgaben für die von Ihnen verwendeten AWS-Services verantwortlich. Weitere Informationen zum Datenschutz finden Sie unter Häufig gestellte Fragen zum Datenschutz. Informationen zum Datenschutz in Europa finden Sie im Blog-Beitrag AWS -Modell der geteilten Verantwortung und in der DSGVO im AWS -Sicherheitsblog.

Aus Datenschutzgründen empfehlen wir, dass Sie AWS-Konto Anmeldeinformationen schützen und einzelne Benutzer mit AWS IAM Identity Center oder AWS Identity and Access Management (IAM) einrichten. So erhält jeder Benutzer nur die Berechtigungen, die zum Durchführen seiner Aufgaben erforderlich sind. Außerdem empfehlen wir, die Daten mit folgenden Methoden schützen:

- Verwenden Sie für jedes Konto die Multi-Faktor-Authentifizierung (MFA).
- Verwenden Sie SSL/TLS, um mit Ressourcen zu kommunizieren. AWS Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Richten Sie die API und die Protokollierung von Benutzeraktivitäten mit ein. AWS CloudTrail
- Verwenden Sie AWS Verschlüsselungslösungen zusammen mit allen darin enthaltenen Standardsicherheitskontrollen AWS-Services.
- Verwenden Sie erweiterte verwaltete Sicherheitsservices wie Amazon Macie, die dabei helfen, in Amazon S3 gespeicherte persönliche Daten zu erkennen und zu schützen.
- Wenn Sie für den Zugriff AWS über eine Befehlszeilenschnittstelle oder eine API FIPS 140-2-validierte kryptografische Module benötigen, verwenden Sie einen FIPS-Endpunkt. Weitere

Informationen über verfügbare FIPS-Endpunkte finden Sie unter Federal Information Processing Standard (FIPS) 140-2.

Wir empfehlen dringend, in Freitextfeldern, z. B. im Feld Name, keine vertraulichen oder sensiblen Informationen wie die E-Mail-Adressen Ihrer Kunden einzugeben. Dies gilt auch, wenn Sie mit Step Functions oder anderen Funktionen arbeiten und die Konsole, die API oder AWS SDKs AWS-Services verwenden. AWS CLI Alle Daten, die Sie in Tags oder Freitextfelder eingeben, die für Namen verwendet werden, können für Abrechnungs- oder Diagnoseprotokolle verwendet werden. Wenn Sie eine URL für einen externen Server bereitstellen, empfehlen wir dringend, keine Anmeldeinformationen zur Validierung Ihrer Anforderung an den betreffenden Server in die URL einzuschließen.

Verschlüsselung in AWS Step Functions

Verschlüsselung im Ruhezustand

Step Functions verschlüsselt Ihre Daten im Ruhezustand immer. Eingeschlossene Daten werden im AWS Step Functions Ruhezustand mit transparenter serverseitiger Verschlüsselung verschlüsselt. Dieser Service reduziert den Ausführungsaufwand und die Komplexität, die mit dem Schutz sensibler Daten verbunden sind. Dank der Verschlüsselung von Daten im Ruhezustand können Sie sicherheitsrelevante Anwendungen erstellen, die die Verschlüsselungs-Compliance und gesetzliche Bestimmungen einhalten.

Verschlüsselung während der Übertragung

Step Functions verschlüsselt Daten, die zwischen dem Dienst und anderen integrierten AWS Diensten übertragen werden (siehe Verwendung AWS Step Functions mit anderen Diensten). Alle Daten, die zwischen Step Functions und integrierten Diensten übertragen werden, werden mit Transport Layer Security (TLS) verschlüsselt.

Identity and Access Management in AWS Step Functions

Für den Zugriff auf AWS Step Functions sind Anmeldeinformationen erforderlich, mit denen Sie Ihre Anfragen authentifizieren AWS können. Diese Anmeldeinformationen müssen über Berechtigungen für den Zugriff auf AWS Ressourcen verfügen, z. B. für das Abrufen von Ereignisdaten aus anderen AWS Ressourcen. In den folgenden Abschnitten erfahren Sie, wie Sie AWS Identity and Access

Management (IAM) und Step Functions verwenden können, um Ihre Ressourcen zu schützen, indem Sie kontrollieren, wer darauf zugreifen kann.

AWS Identity and Access Management (IAM) hilft einem Administrator AWS-Service , den Zugriff auf Ressourcen sicher zu AWS kontrollieren. IAM-Administratoren kontrollieren, wer authentifiziert (angemeldet) und autorisiert werden kann (über Berechtigungen verfügt), um Ressourcen zu verwenden. Step Functions IAM ist ein Programm AWS-Service , das Sie ohne zusätzliche Kosten nutzen können.

Zielgruppe

Die Art und Weise, wie Sie AWS Identity and Access Management (IAM) verwenden, hängt von der Arbeit ab, in der Sie tätig sind. Step Functions

Dienstbenutzer — Wenn Sie den Step Functions Dienst für Ihre Arbeit verwenden, stellt Ihnen Ihr Administrator die erforderlichen Anmeldeinformationen und Berechtigungen zur Verfügung. Wenn Sie für Ihre Arbeit mehr Step Functions Funktionen verwenden, benötigen Sie möglicherweise zusätzliche Berechtigungen. Wenn Sie die Fuktionsweise der Zugriffskontrolle nachvollziehen, wissen Sie bereits, welche Berechtigungen Sie von Ihrem Administrator anzufordern müssen. Unter Problembehandlung bei AWS Step Functions Identität und Zugriff finden Sie nützliche Informationen für den Fall, dass Sie keinen Zugriff auf eine Feature in Step Functions haben.

Serviceadministrator — Wenn Sie in Ihrem Unternehmen für die Step Functions Ressourcen verantwortlich sind, haben Sie wahrscheinlich vollen Zugriff auf Step Functions. Es ist Ihre Aufgabe, zu bestimmen, auf welche Step Functions Funktionen und Ressourcen Ihre Servicebenutzer zugreifen sollen. Sie müssen dann Anträge an Ihren IAM-Administrator stellen, um die Berechtigungen Ihrer Servicenutzer zu ändern. Lesen Sie die Informationen auf dieser Seite, um die Grundkonzepte von IAM nachzuvollziehen. Weitere Informationen darüber, wie Ihr Unternehmen IAM nutzen kann Step Functions, finden Sie unterWie AWS Step Functions funktioniert mit IAM.

IAM-Administrator: Wenn Sie als IAM-Administrator fungieren, sollten Sie Einzelheiten dazu kennen, wie Sie Richtlinien zur Verwaltung des Zugriffs auf Step Functions verfassen können. Beispiele für Step Functions identitätsbasierte Richtlinien, die Sie in IAM verwenden können, finden Sie unter. Beispiele für identitätsbasierte Richtlinien für AWS Step Functions

Authentifizierung mit Identitäten

Authentifizierung ist die Art und Weise, wie Sie sich AWS mit Ihren Identitätsdaten anmelden. Sie müssen als IAM-Benutzer authentifiziert (angemeldet AWS) sein oder eine IAM-Rolle annehmen. Root-Benutzer des AWS-Kontos

Sie können sich AWS als föderierte Identität anmelden, indem Sie Anmeldeinformationen verwenden, die über eine Identitätsquelle bereitgestellt wurden. AWS IAM Identity Center (IAM Identity Center) -Benutzer, die Single Sign-On-Authentifizierung Ihres Unternehmens und Ihre Google- oder Facebook-Anmeldeinformationen sind Beispiele für föderierte Identitäten. Wenn Sie sich als Verbundidentität anmelden, hat der Administrator vorher mithilfe von IAM-Rollen einen Identitätsverbund eingerichtet. Wenn Sie über den Verbund darauf zugreifen AWS , übernehmen Sie indirekt eine Rolle.

Je nachdem, welcher Benutzertyp Sie sind, können Sie sich beim AWS Management Console oder beim AWS Zugangsportal anmelden. Weitere Informationen zur Anmeldung finden Sie AWS unter So melden Sie sich bei Ihrem an AWS-Konto im AWS-Anmeldung Benutzerhandbuch.

Wenn Sie AWS programmgesteuert darauf zugreifen, AWS stellt es ein Software Development Kit (SDK) und eine Befehlszeilenschnittstelle (CLI) bereit, um Ihre Anfragen mithilfe Ihrer Anmeldeinformationen kryptografisch zu signieren. Wenn Sie keine AWS Tools verwenden, müssen Sie Anfragen selbst signieren. Weitere Informationen zur Verwendung der empfohlenen Methode, um Anfragen selbst zu signieren, finden Sie im IAM-Benutzerhandbuch unter AWS API-Anfragen signieren.

Unabhängig von der verwendeten Authentifizierungsmethode müssen Sie möglicherweise zusätzliche Sicherheitsinformationen angeben. AWS Empfiehlt beispielsweise, die Multi-Faktor-Authentifizierung (MFA) zu verwenden, um die Sicherheit Ihres Kontos zu erhöhen. Weitere Informationen finden Sie unter Multi-Faktor-Authentifizierung im AWS IAM Identity Center - Benutzerhandbuch und Verwenden der Multi-Faktor-Authentifizierung (MFA) in AWS im IAM-Benutzerhandbuch.

AWS-Konto Root-Benutzer

Wenn Sie einen erstellen AWS-Konto, beginnen Sie mit einer Anmeldeidentität, die vollständigen Zugriff auf alle AWS-Services Ressourcen im Konto hat. Diese Identität wird als AWS-Konto Root-Benutzer bezeichnet. Sie können darauf zugreifen, indem Sie sich mit der E-Mail-Adresse und dem Passwort anmelden, mit denen Sie das Konto erstellt haben. Wir raten ausdrücklich davon ab, den Root-Benutzer für Alltagsaufgaben zu verwenden. Schützen Sie Ihre Root-Benutzer-Anmeldeinformationen und verwenden Sie diese, um die Aufgaben auszuführen, die nur der Root-Benutzer ausführen kann. Eine vollständige Liste der Aufgaben, für die Sie sich als Root-Benutzer anmelden müssen, finden Sie unter Aufgaben, die Root-Benutzer-Anmeldeinformationen erfordern im IAM-Benutzerhandbuch.

Verbundidentität

Als bewährte Methode sollten menschliche Benutzer, einschließlich Benutzer, die Administratorzugriff benötigen, für den Zugriff AWS-Services mithilfe temporärer Anmeldeinformationen den Verbund mit einem Identitätsanbieter verwenden.

Eine föderierte Identität ist ein Benutzer aus Ihrem Unternehmensbenutzerverzeichnis, einem Web-Identitätsanbieter AWS Directory Service, dem Identity Center-Verzeichnis oder einem beliebigen Benutzer, der mithilfe AWS-Services von Anmeldeinformationen zugreift, die über eine Identitätsquelle bereitgestellt wurden. Wenn föderierte Identitäten darauf zugreifen AWS-Konten, übernehmen sie Rollen, und die Rollen stellen temporäre Anmeldeinformationen bereit.

Für die zentrale Zugriffsverwaltung empfehlen wir Ihnen, AWS IAM Identity Center zu verwenden. Sie können Benutzer und Gruppen in IAM Identity Center erstellen, oder Sie können eine Verbindung zu einer Gruppe von Benutzern und Gruppen in Ihrer eigenen Identitätsquelle herstellen und diese synchronisieren, um sie in all Ihren AWS-Konten Anwendungen zu verwenden. Informationen zu IAM Identity Center finden Sie unter Was ist IAM Identity Center? im AWS IAM Identity Center - Benutzerhandbuch.

IAM-Benutzer und -Gruppen

Ein IAM-Benutzer ist eine Identität innerhalb Ihres Unternehmens AWS-Konto , die über spezifische Berechtigungen für eine einzelne Person oder Anwendung verfügt. Wenn möglich, empfehlen wir, temporäre Anmeldeinformationen zu verwenden, anstatt IAM-Benutzer zu erstellen, die langfristige Anmeldeinformationen wie Passwörter und Zugriffsschlüssel haben. Bei speziellen Anwendungsfällen, die langfristige Anmeldeinformationen mit IAM-Benutzern erfordern, empfehlen wir jedoch, die Zugriffsschlüssel zu rotieren. Weitere Informationen finden Sie unter Regelmäßiges Rotieren von Zugriffsschlüsseln für Anwendungsfälle, die langfristige Anmeldeinformationen erfordern im IAM-Benutzerhandbuch.

Eine IAM-Gruppe ist eine Identität, die eine Sammlung von IAM-Benutzern angibt. Sie können sich nicht als Gruppe anmelden. Mithilfe von Gruppen können Sie Berechtigungen für mehrere Benutzer gleichzeitig angeben. Gruppen vereinfachen die Verwaltung von Berechtigungen, wenn es zahlreiche Benutzer gibt. Sie könnten beispielsweise einer Gruppe mit dem Namen IAMAdmins Berechtigungen zum Verwalten von IAM-Ressourcen erteilen.

Benutzer unterscheiden sich von Rollen. Ein Benutzer ist einer einzigen Person oder Anwendung eindeutig zugeordnet. Eine Rolle kann von allen Personen angenommen werden, die sie benötigen. Benutzer besitzen dauerhafte Anmeldeinformationen. Rollen stellen temporäre Anmeldeinformationen

bereit. Weitere Informationen finden Sie unter Erstellen eines IAM-Benutzers (anstatt einer Rolle) im IAM-Benutzerhandbuch.

IAM-Rollen

Eine IAM-Rolle ist eine Identität innerhalb Ihres Unternehmens AWS-Konto , die über bestimmte Berechtigungen verfügt. Sie ist einem IAM-Benutzer vergleichbar, ist aber nicht mit einer bestimmten Person verknüpft. Sie können vorübergehend eine IAM-Rolle in der übernehmen, AWS Management Console indem Sie die Rollen wechseln. Sie können eine Rolle übernehmen, indem Sie eine AWS CLI oder AWS API-Operation aufrufen oder eine benutzerdefinierte URL verwenden. Weitere Informationen zu Methoden für die Verwendung von Rollen finden Sie unter Verwenden von IAM-Rollen im IAM-Benutzerhandbuch.

IAM-Rollen mit temporären Anmeldeinformationen sind in folgenden Situationen hilfreich:

- Verbundbenutzerzugriff – Um einer Verbundidentität Berechtigungen zuzuweisen, erstellen Sie eine Rolle und definieren Berechtigungen für die Rolle. Wird eine Verbundidentität authentifiziert, so wird die Identität der Rolle zugeordnet und erhält die von der Rolle definierten Berechtigungen. Informationen zu Rollen für den Verbund finden Sie unter Erstellen von Rollen für externe Identitätsanbieter im IAM-Benutzerhandbuch. Wenn Sie IAM Identity Center verwenden, konfigurieren Sie einen Berechtigungssatz. Wenn Sie steuern möchten, worauf Ihre Identitäten nach der Authentifizierung zugreifen können, korreliert IAM Identity Center den Berechtigungssatz mit einer Rolle in IAM. Informationen zu Berechtigungssätzen finden Sie unter Berechtigungssätze im AWS IAM Identity Center -Benutzerhandbuch.
- Temporäre IAM-Benutzerberechtigungen – Ein IAM-Benutzer oder eine -Rolle kann eine IAM-Rolle übernehmen, um vorübergehend andere Berechtigungen für eine bestimmte Aufgabe zu erhalten.
- Kontoübergreifender Zugriff – Sie können eine IAM-Rolle verwenden, um einem vertrauenswürdigen Prinzipal in einem anderen Konto den Zugriff auf Ressourcen in Ihrem Konto zu ermöglichen. Rollen stellen die primäre Möglichkeit dar, um kontoübergreifendem Zugriff zu gewähren. Bei einigen können Sie AWS-Services jedoch eine Richtlinie direkt an eine Ressource anhängen (anstatt eine Rolle als Proxy zu verwenden). Informationen zum Unterschied zwischen Rollen und ressourcenbasierten Richtlinien für den kontenübergreifenden Zugriff finden Sie unter Kontenübergreifender Ressourcenzugriff in IAM im IAM-Benutzerhandbuch.
- Serviceübergreifender Zugriff — Einige verwenden Funktionen in anderen. AWS-Services AWS-Services Wenn Sie beispielsweise einen Aufruf in einem Service tätigen, führt dieser Service häufig Anwendungen in Amazon-EC2 aus oder speichert Objekte in Amazon-S3. Ein Dienst kann

dies mit den Berechtigungen des aufrufenden Prinzipals mit einer Servicerolle oder mit einer serviceverknüpften Rolle tun.

- Forward Access Sessions (FAS) — Wenn Sie einen IAM-Benutzer oder eine IAM-Rolle verwenden, um Aktionen auszuführen AWS, gelten Sie als Principal. Bei einigen Services könnte es Aktionen geben, die dann eine andere Aktion in einem anderen Service initiieren. FAS verwendet die Berechtigungen des Prinzipals, der einen aufruft AWS-Service, in Kombination mit der Anfrage, Anfragen an AWS-Service nachgelagerte Dienste zu stellen. FAS-Anfragen werden nur gestellt, wenn ein Dienst eine Anfrage erhält, für deren Abschluss Interaktionen mit anderen AWS-Services oder Ressourcen erforderlich sind. In diesem Fall müssen Sie über Berechtigungen zum Ausführen beider Aktionen verfügen. Einzelheiten zu den Richtlinien für FAS-Anfragen finden Sie unter Zugriffssitzungen weiterleiten.
- Servicerolle – Eine Servicerolle ist eine IAM-Rolle, die ein Service übernimmt, um Aktionen in Ihrem Namen auszuführen. Ein IAM-Administrator kann eine Servicerolle innerhalb von IAM erstellen, ändern und löschen. Weitere Informationen finden Sie unter Erstellen einer Rolle zum Delegieren von Berechtigungen an einen AWS-Service im IAM-Benutzerhandbuch.
- Dienstbezogene Rolle — Eine dienstbezogene Rolle ist eine Art von Servicerolle, die mit einer verknüpft ist. AWS-Service Der Service kann die Rolle übernehmen, um eine Aktion in Ihrem Namen auszuführen. Servicebezogene Rollen erscheinen in Ihrem Dienst AWS-Konto und gehören dem Dienst. Ein IAM-Administrator kann die Berechtigungen für Service-verknüpfte Rollen anzeigen, aber nicht bearbeiten.

- Auf Amazon EC2 ausgeführte Anwendungen — Sie können eine IAM-Rolle verwenden, um temporäre Anmeldeinformationen für Anwendungen zu verwalten, die auf einer EC2-Instance ausgeführt werden und API-Anfragen stellen AWS CLI . AWS Das ist eher zu empfehlen, als Zugriffsschlüssel innerhalb der EC2-Instance zu speichern. Um einer EC2-Instance eine AWS Rolle zuzuweisen und sie allen ihren Anwendungen zur Verfügung zu stellen, erstellen Sie ein Instance-Profil, das an die Instance angehängt ist. Ein Instance-Profil enthält die Rolle und ermöglicht, dass Programme, die in der EC2-Instance ausgeführt werden, temporäre Anmeldeinformationen erhalten. Weitere Informationen finden Sie unter Verwenden einer IAM-Rolle zum Erteilen von Berechtigungen für Anwendungen, die auf Amazon-EC2-Instances ausgeführt werden im IAM-Benutzerhandbuch.

Informationen dazu, wann Sie IAM-Rollen oder IAM-Benutzer verwenden sollten, finden Sie unter Erstellen einer IAM-Rolle (anstatt eines Benutzers) im IAM-Benutzerhandbuch.

Verwalten des Zugriffs mit Richtlinien

Sie kontrollieren den Zugriff, AWS indem Sie Richtlinien erstellen und diese an AWS Identitäten oder Ressourcen anhängen. Eine Richtlinie ist ein Objekt, AWS das, wenn es einer Identität oder Ressource zugeordnet ist, deren Berechtigungen definiert. AWS wertet diese Richtlinien aus, wenn ein Prinzipal (Benutzer, Root-Benutzer oder Rollensitzung) eine Anfrage stellt. Berechtigungen in den Richtlinien bestimmen, ob die Anforderung zugelassen oder abgelehnt wird. Die meisten Richtlinien werden AWS als JSON-Dokumente gespeichert. Weitere Informationen zu Struktur und Inhalten von JSON-Richtliniendokumenten finden Sie unter Übersicht über JSON-Richtlinien im IAM-Benutzerhandbuch.

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer auf was Zugriff hat. Das bedeutet, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

Standardmäßig haben Benutzer, Gruppen und Rollen keine Berechtigungen. Ein IAM-Administrator muss IAM-Richtlinien erstellen, die Benutzern die Berechtigung erteilen, Aktionen für die Ressourcen auszuführen, die sie benötigen. Der Administrator kann dann die IAM-Richtlinien zu Rollen hinzufügen, und Benutzer können die Rollen annehmen.

IAM-Richtlinien definieren Berechtigungen für eine Aktion unabhängig von der Methode, die Sie zur Ausführung der Aktion verwenden. Angenommen, es gibt eine Richtlinie, die Berechtigungen für die `iam:GetRole`-Aktion erteilt. Ein Benutzer mit dieser Richtlinie kann Rolleninformationen von der AWS Management Console AWS CLI, der oder der AWS API abrufen.

Identitätsbasierte Richtlinien

Identitätsbasierte Richtlinien sind JSON-Berechtigungsrichtliniendokumente, die Sie einer Identität anfügen können, wie z. B. IAM-Benutzern, -Benutzergruppen oder -Rollen. Diese Richtlinien steuern, welche Aktionen die Benutzer und Rollen für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen identitätsbasierter Richtlinien finden Sie unter Erstellen von IAM-Richtlinien im IAM-Benutzerhandbuch.

Identitätsbasierte Richtlinien können weiter als Inline-Richtlinien oder verwaltete Richtlinien kategorisiert werden. Inline-Richtlinien sind direkt in einen einzelnen Benutzer, eine einzelne Gruppe oder eine einzelne Rolle eingebettet. Verwaltete Richtlinien sind eigenständige Richtlinien, die Sie mehreren Benutzern, Gruppen und Rollen in Ihrem System zuordnen können AWS-Konto. Zu den verwalteten Richtlinien gehören AWS verwaltete Richtlinien und vom Kunden verwaltete Richtlinien. Informationen dazu, wie Sie zwischen einer verwalteten Richtlinie und einer eingebundenen Richtlinie

wählen, finden Sie unter Auswahl zwischen verwalteten und eingebundenen Richtlinien im IAM-Benutzerhandbuch.

Ressourcenbasierte Richtlinien

Ressourcenbasierte Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource anfügen. Beispiele für ressourcenbasierte Richtlinien sind IAM-Rollen-Vertrauensrichtlinien und Amazon-S3-Bucket-Richtlinien. In Services, die ressourcenbasierte Richtlinien unterstützen, können Service-Administratoren sie verwenden, um den Zugriff auf eine bestimmte Ressource zu steuern. Für die Ressource, an welche die Richtlinie angehängt ist, legt die Richtlinie fest, welche Aktionen ein bestimmter Prinzipal unter welchen Bedingungen für diese Ressource ausführen kann. Sie müssen in einer ressourcenbasierten Richtlinie einen Prinzipal angeben. Zu den Prinzipalen können Konten, Benutzer, Rollen, Verbundbenutzer oder gehören. AWS-Services

Ressourcenbasierte Richtlinien sind Richtlinien innerhalb dieses Diensts. Sie können AWS verwaltete Richtlinien von IAM nicht in einer ressourcenbasierten Richtlinie verwenden.

Zugriffssteuerungslisten (ACLs)

Zugriffssteuerungslisten (ACLs) steuern, welche Prinzipale (Kontomitglieder, Benutzer oder Rollen) auf eine Ressource zugreifen können. ACLs sind ähnlich wie ressourcenbasierte Richtlinien, verwenden jedoch nicht das JSON-Richtliniendokumentformat.

Amazon S3 und Amazon VPC sind Beispiele für Services, die ACLs unterstützen. AWS WAF Weitere Informationen“ zu ACLs finden Sie unter Zugriffskontrollliste (ACL) – Übersicht (Access Control List) im Amazon-Simple-Storage-Service-Entwicklerhandbuch.

Weitere Richtlinientypen

AWS unterstützt zusätzliche, weniger verbreitete Richtlinientypen. Diese Richtlinientypen können die maximalen Berechtigungen festlegen, die Ihnen von den häufiger verwendeten Richtlinientypen erteilt werden können.

- Berechtigungsgrenzen – Eine Berechtigungsgrenze ist ein erweitertes Feature, mit der Sie die maximalen Berechtigungen festlegen können, die eine identitätsbasierte Richtlinie einer IAM-Entität (IAM-Benutzer oder -Rolle) erteilen kann. Sie können eine Berechtigungsgrenze für eine Entität festlegen. Die daraus resultierenden Berechtigungen sind der Schnittpunkt der identitätsbasierten Richtlinien einer Entität und ihrer Berechtigungsgrenzen. Ressourcenbasierte Richtlinien, die den Benutzer oder die Rolle im Feld `Principal` angeben, werden nicht durch

Berechtigungsgrenzen eingeschränkt. Eine explizite Zugriffsverweigerung in einer dieser Richtlinien setzt eine Zugriffserlaubnis außer Kraft. Weitere Informationen über Berechtigungsgrenzen finden Sie unter Berechtigungsgrenzen für IAM-Entitäten im IAM-Benutzerhandbuch.

- Service Control Policies (SCPs) — SCPs sind JSON-Richtlinien, die die maximalen Berechtigungen für eine Organisation oder Organisationseinheit (OU) in festlegen. AWS Organizations AWS Organizations ist ein Dienst zur Gruppierung und zentralen Verwaltung mehrerer Objekte AWS-Konten , die Ihrem Unternehmen gehören. Wenn Sie innerhalb einer Organisation alle Features aktivieren, können Sie Service-Kontrollrichtlinien (SCPs) auf alle oder einzelne Ihrer Konten anwenden. Das SCP schränkt die Berechtigungen für Entitäten in Mitgliedskonten ein, einschließlich der einzelnen Entitäten. Root-Benutzer des AWS-Kontos Weitere Informationen zu Organizations und SCPs finden Sie unter Funktionsweise von SCPs im AWS Organizations -Benutzerhandbuch.
- Sitzungsrichtlinien – Sitzungsrichtlinien sind erweiterte Richtlinien, die Sie als Parameter übergeben, wenn Sie eine temporäre Sitzung für eine Rolle oder einen verbundenen Benutzer programmgesteuert erstellen. Die resultierenden Sitzungsberechtigungen sind eine Schnittmenge der auf der Identität des Benutzers oder der Rolle basierenden Richtlinien und der Sitzungsrichtlinien. Berechtigungen können auch aus einer ressourcenbasierten Richtlinie stammen. Eine explizite Zugriffsverweigerung in einer dieser Richtlinien setzt eine Zugriffserlaubnis außer Kraft. Weitere Informationen finden Sie unter Sitzungsrichtlinien im IAM-Benutzerhandbuch.

Mehrere Richtlinientypen

Wenn mehrere auf eine Anforderung mehrere Richtlinientypen angewendet werden können, sind die entsprechenden Berechtigungen komplizierter. Informationen darüber, wie AWS bestimmt wird, ob eine Anfrage zulässig ist, wenn mehrere Richtlinientypen betroffen sind, finden Sie im IAM-Benutzerhandbuch unter Bewertungslogik für Richtlinien.

Zugriffskontrolle

Sie können über gültige Anmeldeinformationen verfügen, um Ihre Anfragen zu authentifizieren, aber ohne die entsprechenden Berechtigungen können Sie keine Step Functions Functions-Ressourcen erstellen oder darauf zugreifen. Sie benötigen beispielsweise Berechtigungen zum Aufrufen AWS Lambda von Zielen für Amazon Simple Notification Service (Amazon SNS) und Amazon Simple Queue Service (Amazon SQS), die mit Ihren Step Functions Functions-Regeln verknüpft sind.

In den folgenden Abschnitten wird beschrieben, wie Sie Berechtigungen für Step Functions verwalten.

- Erstellen Sie eine IAM-Rolle für Ihren State Machine
- Granulare IAM-Berechtigungen für Benutzer ohne Administratorrechte erstellen
- Amazon VPC-Endpunkte für Step Functions
- IAM-Richtlinien für integrierte Dienste
- IAM-Richtlinien für die Verwendung des Distributed Map-Status

Richtlinienaktionen für Step Functions

Unterstützt Richtlinienaktionen	Ja

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer Zugriff auf was hat. Das heißt, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

Das Element `Action` einer JSON-Richtlinie beschreibt die Aktionen, mit denen Sie den Zugriff in einer Richtlinie zulassen oder verweigern können. Richtlinienaktionen haben normalerweise denselben Namen wie der zugehörige AWS API-Vorgang. Es gibt einige Ausnahmen, z. B. Aktionen, die nur mit Genehmigung durchgeführt werden können und für die es keinen passenden API-Vorgang gibt. Es gibt auch einige Operationen, die mehrere Aktionen in einer Richtlinie erfordern. Diese zusätzlichen Aktionen werden als abhängige Aktionen bezeichnet.

Schließen Sie Aktionen in eine Richtlinie ein, um Berechtigungen zur Durchführung der zugeordneten Operation zu erteilen.

Eine Liste der Step Functions Aktionen finden Sie unter Resources Defined by AWS Step Functions in der Service Authorization Reference.

Bei Richtlinienaktionen wird vor der Aktion das folgende Präfix Step Functions verwendet:

```
states
```

Um mehrere Aktionen in einer einzigen Anweisung anzugeben, trennen Sie sie mit Kommata:

```
"Action": [
      "states:action1",
      "states:action2"
```

```
        ]
```

Beispiele für Step Functions identitätsbasierte Richtlinien finden Sie unter. Beispiele für identitätsbasierte Richtlinien für AWS Step Functions

Politische Ressourcen für Step Functions

Unterstützt Richtlinienressourcen	Ja

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer auf was Zugriff hat. Das bedeutet die Festlegung, welcher Prinzipal Aktionen für welche Ressourcen unter welchen Bedingungen ausführen kann.

Das JSON-Richtlinienelement `Resource` gibt die Objekte an, auf welche die Aktion angewendet wird. Anweisungen müssen entweder ein – `Resource`oder ein `NotResource`-Element enthalten. Als bewährte Methode geben Sie eine Ressource mit dem zugehörigen Amazon-Ressourcennamen (ARN) an. Sie können dies für Aktionen tun, die einen bestimmten Ressourcentyp unterstützen, der als Berechtigungen auf Ressourcenebene bezeichnet wird.

Verwenden Sie für Aktionen, die keine Berechtigungen auf Ressourcenebene unterstützen, z. B. Auflistungsoperationen, einen Platzhalter (*), um anzugeben, dass die Anweisung für alle Ressourcen gilt.

```
"Resource": "*"
```

Eine Liste der Step Functions Ressourcentypen und ihrer ARNs finden Sie unter Definierte Aktionen von AWS Step Functions in der Serviceautorisierungsreferenz. Informationen darüber, mit welchen Aktionen Sie den ARN jeder Ressource angeben können, finden Sie unter Ressourcen definiert von AWS Step Functions.

Beispiele für Step Functions identitätsbasierte Richtlinien finden Sie unter. Beispiele für identitätsbasierte Richtlinien für AWS Step Functions

Bedingungsschlüssel für Richtlinien für Step Functions

Unterstützt servicespezifische Richtlini enbedingungsschlüssel	Ja

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer auf was Zugriff hat. Das heißt, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

Das Element `Condition` (oder `Condition` block) ermöglicht Ihnen die Angabe der Bedingungen, unter denen eine Anweisung wirksam ist. Das Element `Condition` ist optional. Sie können bedingte Ausdrücke erstellen, die Bedingungsoperatoren verwenden, z. B. ist gleich oder kleiner als, damit die Bedingung in der Richtlinie mit Werten in der Anforderung übereinstimmt.

Wenn Sie mehrere `Condition`-Elemente in einer Anweisung oder mehrere Schlüssel in einem einzelnen `Condition`-Element angeben, wertet AWS diese mittels einer logischen `AND`-Operation aus. Wenn Sie mehrere Werte für einen einzelnen Bedingungsschlüssel angeben, AWS wertet die Bedingung mithilfe einer logischen `OR` Operation aus. Alle Bedingungen müssen erfüllt werden, bevor die Berechtigungen der Anweisung gewährt werden.

Sie können auch Platzhaltervariablen verwenden, wenn Sie Bedingungen angeben. Beispielsweise können Sie einem IAM-Benutzer die Berechtigung für den Zugriff auf eine Ressource nur dann gewähren, wenn sie mit dessen IAM-Benutzernamen gekennzeichnet ist. Weitere Informationen finden Sie unter IAM-Richtlinienelemente: Variablen und Tags im IAM-Benutzerhandbuch.

AWS unterstützt globale Bedingungsschlüssel und dienstspezifische Bedingungsschlüssel. Eine Übersicht aller AWS globalen Bedingungsschlüssel finden Sie unter Kontextschlüssel für AWS globale Bedingungen im IAM-Benutzerhandbuch.

Eine Liste der Step Functions Bedingungsschlüssel finden Sie unter Bedingungsschlüssel für AWS Step Functions in der Service Authorization Reference. Informationen zu den Aktionen und Ressourcen, mit denen Sie einen Bedingungsschlüssel verwenden können, finden Sie unter Ressourcen, die definiert von sind AWS Step Functions.

Beispiele für Step Functions identitätsbasierte Richtlinien finden Sie unter. Beispiele für identitätsbasierte Richtlinien für AWS Step Functions

ACLs in Step Functions

Unterstützt ACLs	Nein

Zugriffssteuerungslisten (ACLs) steuern, welche Prinzipale (Kontomitglieder, Benutzer oder Rollen) auf eine Ressource zugreifen können. ACLs sind ähnlich wie ressourcenbasierte Richtlinien, verwenden jedoch nicht das JSON-Richtliniendokumentformat.

ABAC mit Step Functions

Unterstützt ABAC (Tags in Richtlinien)	Teilweise

Die attributbasierte Zugriffskontrolle (ABAC) ist eine Autorisierungsstrategie, bei der Berechtigungen basierend auf Attributen definiert werden. In AWS werden diese Attribute als Tags bezeichnet. Sie können Tags an IAM-Entitäten (Benutzer oder Rollen) und an viele AWS Ressourcen anhängen. Das Markieren von Entitäten und Ressourcen ist der erste Schritt von ABAC. Anschließend entwerfen Sie ABAC-Richtlinien, um Operationen zuzulassen, wenn das Tag des Prinzipals mit dem Tag der Ressource übereinstimmt, auf die sie zugreifen möchten.

ABAC ist in Umgebungen hilfreich, die schnell wachsen, und unterstützt Sie in Situationen, in denen die Richtlinienverwaltung mühsam wird.

Um den Zugriff auf der Grundlage von Tags zu steuern, geben Sie im Bedingungselement einer Richtlinie Tag-Informationen an, indem Sie die Schlüssel `aws:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`*, oder Bedingung `aws:TagKeys` verwenden.

Wenn ein Service alle drei Bedingungsschlüssel für jeden Ressourcentyp unterstützt, lautet der Wert für den Service Ja. Wenn ein Service alle drei Bedingungsschlüssel für nur einige Ressourcentypen unterstützt, lautet der Wert Teilweise.

Weitere Informationen zu ABAC finden Sie unter Was ist ABAC? im IAM-Benutzerhandbuch. Um ein Tutorial mit Schritten zur Einstellung von ABAC anzuzeigen, siehe Attributbasierte Zugriffskontrolle (ABAC) verwenden im IAM-Benutzerhandbuch.

Verwenden temporärer Anmeldeinformationen mit Step Functions

Unterstützt temporäre Anmeldeinformationen	Ja

Einige funktionieren AWS-Services nicht, wenn Sie sich mit temporären Anmeldeinformationen anmelden. Weitere Informationen, einschließlich Informationen, die mit temporären Anmeldeinformationen AWS-Services funktionieren AWS-Services , finden Sie im IAM-Benutzerhandbuch unter Diese Option funktioniert mit IAM.

Sie verwenden temporäre Anmeldeinformationen, wenn Sie sich mit einer anderen AWS Management Console Methode als einem Benutzernamen und einem Passwort anmelden. Wenn

Sie beispielsweise AWS über den Single Sign-On-Link (SSO) Ihres Unternehmens darauf zugreifen, werden bei diesem Vorgang automatisch temporäre Anmeldeinformationen erstellt. Sie erstellen auch automatisch temporäre Anmeldeinformationen, wenn Sie sich als Benutzer bei der Konsole anmelden und dann die Rollen wechseln. Weitere Informationen zum Wechseln von Rollen finden Sie unter Wechseln zu einer Rolle (Konsole) im IAM-Benutzerhandbuch.

Mithilfe der AWS API AWS CLI oder können Sie temporäre Anmeldeinformationen manuell erstellen. Sie können diese temporären Anmeldeinformationen dann für den Zugriff verwenden AWS. AWS empfiehlt, temporäre Anmeldeinformationen dynamisch zu generieren, anstatt langfristige Zugriffsschlüssel zu verwenden. Weitere Informationen finden Sie unter Temporäre Sicherheitsanmeldeinformationen in IAM.

Serviceübergreifende Prinzipalberechtigungen für Step Functions

Unterstützt Forward Access Sessions (FAS)	Ja

Wenn Sie einen IAM-Benutzer oder eine IAM-Rolle verwenden, um Aktionen auszuführen AWS, gelten Sie als Principal. Bei einigen Services könnte es Aktionen geben, die dann eine andere Aktion in einem anderen Service initiieren. FAS verwendet die Berechtigungen des Prinzipals, der einen aufruft AWS-Service, kombiniert mit der Anforderung, Anfragen an nachgelagerte Dienste AWS-Service zu stellen. FAS-Anfragen werden nur gestellt, wenn ein Dienst eine Anfrage erhält, für deren Abschluss Interaktionen mit anderen AWS-Services oder Ressourcen erforderlich sind. In diesem Fall müssen Sie über Berechtigungen zum Ausführen beider Aktionen verfügen. Einzelheiten zu den Richtlinien für FAS-Anfragen finden Sie unter Zugriffssitzungen weiterleiten.

Servicerollen für Step Functions

Unterstützt Servicerollen	Ja

Eine Servicerolle ist eine IAM-Rolle, die ein Service annimmt, um Aktionen in Ihrem Namen auszuführen. Ein IAM-Administrator kann eine Servicerolle innerhalb von IAM erstellen, ändern und löschen. Weitere Informationen finden Sie unter Erstellen einer Rolle zum Delegieren von Berechtigungen an einen AWS-Service im IAM-Benutzerhandbuch.

> **Warning**
>
> Durch das Ändern der Berechtigungen für eine Servicerolle kann die Step Functions Funktionalität beeinträchtigt werden. Bearbeiten Sie Servicerollen nur, Step Functions wenn Sie dazu eine Anleitung erhalten.

Dienstbezogene Rollen für Step Functions

Unterstützt serviceverknüpfte Rollen	Nein

Eine dienstbezogene Rolle ist eine Art von Servicerolle, die mit einer verknüpft ist. AWS-Service Der Service kann die Rolle übernehmen, um eine Aktion in Ihrem Namen auszuführen. Dienstbezogene Rollen werden in Ihrem Dienst angezeigt AWS-Konto und gehören dem Dienst. Ein IAM-Administrator kann die Berechtigungen für Service-verknüpfte Rollen anzeigen, aber nicht bearbeiten.

Details zum Erstellen oder Verwalten von serviceverknüpften Rollen finden Sie unter AWS -Services, die mit IAM funktionieren. Suchen Sie in der Tabelle nach einem Service mit einem `Yes` in der Spalte Service-linked role (Serviceverknüpfte Rolle). Wählen Sie den Link Yes (Ja) aus, um die Dokumentation für die serviceverknüpfte Rolle für diesen Service anzuzeigen.

Wie AWS Step Functions funktioniert mit IAM

Bevor Sie IAM zur Verwaltung des Zugriffs auf verwenden, sollten Sie sich darüber informieren Step Functions, mit welchen IAM-Funktionen Sie arbeiten können. Step Functions

IAM-Funktionen, die Sie mit verwenden können AWS Step Functions

IAM-Feature	Step Functions Unterstützung
Identitätsbasierte Richtlinien	Ja
Ressourcenbasierte Richtlinien	Nein
Richtlinienaktionen	Ja
Richtlinienressourcen	Ja

IAM-Feature	Step Functions Unterstützung
Richtlinienbedingungsschlüssel (services pezifisch)	Ja
ACLs	Nein
ABAC (Tags in Richtlinien)	Teilweise
Temporäre Anmeldeinformationen	Ja
Hauptberechtigungen	Ja
Servicerollen	Ja
Service-verknüpfte Rollen	Nein

Einen allgemeinen Überblick darüber, wie Step Functions und andere AWS Dienste mit den meisten IAM-Funktionen funktionieren, finden Sie im IAM-Benutzerhandbuch unter AWS Dienste, die mit IAM funktionieren.

Beispiele für identitätsbasierte Richtlinien für AWS Step Functions

Benutzer und Rollen haben standardmäßig nicht die Berechtigung, Step Functions -Ressourcen zu erstellen oder zu ändern. Sie können auch keine Aufgaben mithilfe der AWS Management Console, AWS Command Line Interface (AWS CLI) oder AWS API ausführen. Ein IAM-Administrator muss IAM-Richtlinien erstellen, die Benutzern die Berechtigung erteilen, Aktionen für die Ressourcen auszuführen, die sie benötigen. Der Administrator kann dann die IAM-Richtlinien zu Rollen hinzufügen, und Benutzer können die Rollen annehmen.

Informationen dazu, wie Sie unter Verwendung dieser beispielhaften JSON-Richtliniendokumente eine identitätsbasierte IAM-Richtlinie erstellen, finden Sie unter Erstellen von IAM-Richtlinien im IAM-Benutzerhandbuch.

Einzelheiten zu Aktionen und Ressourcentypen, die von definiert wurden Step Functions, einschließlich des Formats der ARNs für jeden der Ressourcentypen, finden Sie unter Aktionen, Ressourcen und Bedingungsschlüssel für AWS Step Functions in der Service Authorization Reference.

Themen

Bewährte Methoden für Richtlinien

Identitätsbasierte Richtlinien legen fest, ob jemand Step Functions Ressourcen in Ihrem Konto erstellen, darauf zugreifen oder sie löschen kann. Dies kann zusätzliche Kosten für Ihr verursachen AWS-Konto. Befolgen Sie beim Erstellen oder Bearbeiten identitätsbasierter Richtlinien die folgenden Anleitungen und Empfehlungen:

- Beginnen Sie mit AWS verwalteten Richtlinien und wechseln Sie zu Berechtigungen mit den geringsten Rechten — Verwenden Sie die AWS verwalteten Richtlinien, die Berechtigungen für viele gängige Anwendungsfälle gewähren, um Ihren Benutzern und Workloads zunächst Berechtigungen zu gewähren. Sie sind in Ihrem verfügbar. AWS-Konto Wir empfehlen Ihnen, die Berechtigungen weiter zu reduzieren, indem Sie vom AWS Kunden verwaltete Richtlinien definieren, die speziell auf Ihre Anwendungsfälle zugeschnitten sind. Weitere Informationen finden Sie unter AWS -verwaltete Richtlinien oder AWS -verwaltete Richtlinien für Auftrags-Funktionen im IAM-Benutzerhandbuch.
- Anwendung von Berechtigungen mit den geringsten Rechten – Wenn Sie mit IAM-Richtlinien Berechtigungen festlegen, gewähren Sie nur die Berechtigungen, die für die Durchführung einer Aufgabe erforderlich sind. Sie tun dies, indem Sie die Aktionen definieren, die für bestimmte Ressourcen unter bestimmten Bedingungen durchgeführt werden können, auch bekannt als die geringsten Berechtigungen. Weitere Informationen zur Verwendung von IAM zum Anwenden von Berechtigungen finden Sie unter Richtlinien und Berechtigungen in IAM im IAM-Benutzerhandbuch.
- Verwenden von Bedingungen in IAM-Richtlinien zur weiteren Einschränkung des Zugriffs – Sie können Ihren Richtlinien eine Bedingung hinzufügen, um den Zugriff auf Aktionen und Ressourcen zu beschränken. Sie können beispielsweise eine Richtlinienbedingung schreiben, um festzulegen, dass alle Anforderungen mithilfe von SSL gesendet werden müssen. Sie können auch Bedingungen verwenden, um Zugriff auf Serviceaktionen zu gewähren, wenn diese für einen bestimmten Zweck verwendet werden AWS-Service, z. AWS CloudFormation B. Weitere Informationen finden Sie unter IAM-JSON-Richtlinienelemente: Bedingung im IAM-Benutzerhandbuch.

- Verwenden von IAM Access Analyzer zur Validierung Ihrer IAM-Richtlinien, um sichere und funktionale Berechtigungen zu gewährleisten – IAM Access Analyzer validiert neue und vorhandene Richtlinien, damit die Richtlinien der IAM-Richtliniensprache (JSON) und den bewährten IAM-Methoden entsprechen. IAM Access Analyzer stellt mehr als 100 Richtlinienprüfungen und umsetzbare Empfehlungen zur Verfügung, damit Sie sichere und funktionale Richtlinien erstellen können. Weitere Informationen finden Sie unter Richtlinienvalidierung zum IAM Access Analyzer im IAM-Benutzerhandbuch.
- Multi-Faktor-Authentifizierung (MFA) erforderlich — Wenn Sie ein Szenario haben, das IAM-Benutzer oder einen Root-Benutzer in Ihrem System erfordert AWS-Konto, aktivieren Sie MFA für zusätzliche Sicherheit. Um MFA beim Aufrufen von API-Vorgängen anzufordern, fügen Sie Ihren Richtlinien MFA-Bedingungen hinzu. Weitere Informationen finden Sie unter Konfigurieren eines MFA-geschützten API-Zugriffs im IAM-Benutzerhandbuch.

Weitere Informationen zu bewährten Methoden in IAM finden Sie unter Bewährte Methoden für die Sicherheit in IAM im IAM-Benutzerhandbuch.

Verwenden der Step Functions -Konsole

Um auf die AWS Step Functions Konsole zugreifen zu können, benötigen Sie ein Mindestmaß an Berechtigungen. Diese Berechtigungen müssen es Ihnen ermöglichen, Details zu den Step Functions Ressourcen in Ihrem aufzulisten und anzuzeigen AWS-Konto. Wenn Sie eine identitätsbasierte Richtlinie erstellen, die strenger ist als die mindestens erforderlichen Berechtigungen, funktioniert die Konsole nicht wie vorgesehen für Entitäten (Benutzer oder Rollen) mit dieser Richtlinie.

Sie müssen Benutzern, die nur die API AWS CLI oder die AWS API aufrufen, keine Mindestberechtigungen für die Konsole gewähren. Stattdessen sollten Sie nur Zugriff auf die Aktionen zulassen, die der API-Operation entsprechen, die die Benutzer ausführen möchten.

Um sicherzustellen, dass Benutzer und Rollen die Step Functions Konsole weiterhin verwenden können, fügen Sie den Entitäten auch die Step Functions `ConsoleAccess` oder die `ReadOnly` AWS verwaltete Richtlinie hinzu. Weitere Informationen finden Sie unter Hinzufügen von Berechtigungen zu einem Benutzer im IAM-Benutzerhandbuch.

Gewähren der Berechtigung zur Anzeige der eigenen Berechtigungen für Benutzer

In diesem Beispiel wird gezeigt, wie Sie eine Richtlinie erstellen, die IAM-Benutzern die Berechtigung zum Anzeigen der eingebundenen Richtlinien und verwalteten Richtlinien gewährt, die ihrer

Benutzeridentität angefügt sind. Diese Richtlinie umfasst Berechtigungen zum Ausführen dieser Aktion auf der Konsole oder programmgesteuert mithilfe der AWS CLI AWS OR-API.

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

Identitätsbasierte Richtlinien für Step Functions

Unterstützt Richtlinien auf Identitätsbasis.	Ja

Identitätsbasierte Richtlinien sind JSON-Berechtigungsrichtliniendokumente, die Sie einer Identität anfügen können, wie z. B. IAM-Benutzern, -Benutzergruppen oder -Rollen. Diese Richtlinien steuern, welche Aktionen die Benutzer und Rollen für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen identitätsbasierter Richtlinien finden Sie unter Erstellen von IAM-Richtlinien im IAM-Benutzerhandbuch.

Mit identitätsbasierten IAM-Richtlinien können Sie angeben, welche Aktionen und Ressourcen zugelassen oder abgelehnt werden. Darüber hinaus können Sie die Bedingungen festlegen, unter denen Aktionen zugelassen oder abgelehnt werden. Sie können den Prinzipal nicht in einer identitätsbasierten Richtlinie angeben, da er für den Benutzer oder die Rolle gilt, dem er zugeordnet ist. Informationen zu sämtlichen Elementen, die Sie in einer JSON-Richtlinie verwenden, finden Sie in der IAM-Referenz für JSON-Richtlinienelemente im IAM-Benutzerhandbuch.

Beispiele für identitätsbasierte Richtlinien für Step Functions

Beispiele für Step Functions identitätsbasierte Richtlinien finden Sie unter. Beispiele für identitätsbasierte Richtlinien für AWS Step Functions

Ressourcenbasierte Richtlinien finden Sie in Step Functions

Unterstützt ressourcenbasierte Richtlinien	Nein

Ressourcenbasierte Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource anfügen. Beispiele für ressourcenbasierte Richtlinien sind IAM-Rollen-Vertrauensrichtlinien und Amazon-S3-Bucket-Richtlinien. In Services, die ressourcenbasierte Richtlinien unterstützen, können Service-Administratoren sie verwenden, um den Zugriff auf eine bestimmte Ressource zu steuern. Für die Ressource, an welche die Richtlinie angehängt ist, legt die Richtlinie fest, welche Aktionen ein bestimmter Prinzipal unter welchen Bedingungen für diese Ressource ausführen kann. Sie müssen in einer ressourcenbasierten Richtlinie einen Prinzipal angeben. Zu den Prinzipalen können Konten, Benutzer, Rollen, Verbundbenutzer oder gehören. AWS-Services

Um kontoübergreifenden Zugriff zu ermöglichen, können Sie ein gesamtes Konto oder IAM-Entitäten in einem anderen Konto als Prinzipal in einer ressourcenbasierten Richtlinie angeben. Durch das Hinzufügen eines kontoübergreifenden Auftraggebers zu einer ressourcenbasierten Richtlinie ist nur die halbe Vertrauensbeziehung eingerichtet. Wenn sich der Prinzipal und die Ressource unterscheiden AWS-Konten, muss ein IAM-Administrator des vertrauenswürdigen Kontos auch der Prinzipalentität (Benutzer oder Rolle) die Berechtigung zum Zugriff auf die Ressource erteilen. Sie

erteilen Berechtigungen, indem Sie der juristischen Stelle eine identitätsbasierte Richtlinie anfügen. Wenn jedoch eine ressourcenbasierte Richtlinie Zugriff auf einen Prinzipal in demselben Konto gewährt, ist keine zusätzliche identitätsbasierte Richtlinie erforderlich. Weitere Informationen finden Sie unter Kontenübergreifender Ressourcenzugriff in IAM im IAM-Benutzerhandbuch.

AWS verwaltete Richtlinien für AWS Step Functions

Eine AWS verwaltete Richtlinie ist eine eigenständige Richtlinie, die von erstellt und verwaltet wird AWS. AWS Verwaltete Richtlinien sind so konzipiert, dass sie Berechtigungen für viele gängige Anwendungsfälle bereitstellen, sodass Sie damit beginnen können, Benutzern, Gruppen und Rollen Berechtigungen zuzuweisen.

Beachten Sie, dass AWS verwaltete Richtlinien für Ihre speziellen Anwendungsfälle möglicherweise keine Berechtigungen mit den geringsten Rechten gewähren, da sie allen AWS Kunden zur Verfügung stehen. Wir empfehlen Ihnen, die Berechtigungen weiter zu reduzieren, indem Sie kundenverwaltete Richtlinien definieren, die speziell auf Ihre Anwendungsfälle zugeschnitten sind.

Sie können die in AWS verwalteten Richtlinien definierten Berechtigungen nicht ändern. Wenn die in einer AWS verwalteten Richtlinie definierten Berechtigungen AWS aktualisiert werden, wirkt sich das Update auf alle Prinzidentitäten (Benutzer, Gruppen und Rollen) aus, denen die Richtlinie zugeordnet ist. AWS aktualisiert eine AWS verwaltete Richtlinie höchstwahrscheinlich, wenn eine neue Richtlinie eingeführt AWS-Service wird oder neue API-Operationen für bestehende Dienste verfügbar werden.

Weitere Informationen finden Sie unter Von AWS verwaltete Richtlinien im IAM-Benutzerhandbuch.

AWS verwaltete Richtlinie: AWSStepFunctionsConsoleFullAccess

Sie können die `AWSStepFunctionsConsoleFullAccess`-Richtlinie an Ihre IAM-Identitäten anfügen.

Diese Richtlinie gewährt *`Administratorberechtigungen`*, die einem Benutzer den Zugriff auf die Step Functions-Konsole ermöglichen. Für ein vollständiges Konsolenerlebnis benötigt ein Benutzer möglicherweise auch iam: PassRole -Berechtigungen für andere IAM-Rollen, die vom Dienst übernommen werden können.

AWS verwaltete Richtlinie: AWSStepFunctionsReadOnlyAccess

Sie können die `AWSStepFunctionsReadOnlyAccess`-Richtlinie an Ihre IAM-Identitäten anfügen.

Diese Richtlinie gewährt *nur Leseberechtigungen*, die es einem Benutzer oder einer Rolle ermöglichen, Zustandsmaschinen, Aktivitäten, Ausführungen, Aktivitäten, Tags sowie Alias und Versionen von Zustandsmaschinen aufzulisten und zu beschreiben. MapRuns Diese Richtlinie gewährt auch die Berechtigung, die Syntax der von Ihnen bereitgestellten Zustandsmaschinendefinitionen zu überprüfen.

AWS verwaltete Richtlinie: AWSStepFunctionsFullAccess

Sie können die `AWSStepFunctionsFullAccess`-Richtlinie an Ihre IAM-Identitäten anfügen.

Diese Richtlinie gewährt einem Benutzer oder einer Rolle *volle* Berechtigungen zur Verwendung der Step Functions Functions-API. Für den Vollzugriff muss ein Benutzer über die *iam: PassRole* -Berechtigung für mindestens eine IAM-Rolle verfügen, die vom Dienst übernommen werden kann.

Step Functions Aktualisierungen der verwalteten AWS Richtlinien

Hier finden Sie Informationen zu Aktualisierungen AWS verwalteter Richtlinien, die Step Functions seit Beginn der Nachverfolgung dieser Änderungen durch diesen Dienst vorgenommen wurden. Um automatische Warnungen über Änderungen an dieser Seite zu erhalten, abonnieren Sie den RSS-Feed auf der Step Functions Dokumentverlauf-Seite.

Änderung	Beschreibung	Datum
AWSStepFunctionsReadOnlyAccess – Aktualisierung auf eine bestehende Richtlinie	Step Functions Es wurden neue Berechtigungen hinzugefügt, mit denen `states:ValidateStateMachineDefinition` API-Aktionen aufgerufen werden können, um die Syntax der von Ihnen bereitgestellten State-Machine-Definitionen zu überprüfen.	25. April 2024

Änderung	Beschreibung	Datum
AWSStepFunctionsReadOnlyAccess – Aktualisierung auf eine bestehende Richtlinie	Step Functions Es wurden neue Berechtigungen hinzugefügt, um das Auflisten und Lesen von Daten zu folgenden Themen zu ermöglichen: Tags (ListTagsForResource), Distributed ListMap Map (Runs, DescribeMapRun), Versionen und Aliase (DescribeStateMachineAlias, ListStateMachineAliases, ListStateMachineVersions).	02. April 2024
Step Functions hat begonnen, Änderungen zu verfolgen	Step Functions hat begonnen, Änderungen für die AWS verwalteten Richtlinien zu verfolgen.	02. April 2024

Erstellen Sie eine IAM-Rolle für Ihren State Machine

AWS Step Functions kann Code ausführen und auf AWS Ressourcen zugreifen (z. B. eine AWS Lambda Funktion aufrufen). Zur Gewährleistung der Sicherheit müssen Sie Step Functions mithilfe einer IAM-Rolle Zugriff auf diese Ressourcen gewähren.

Die Tutorials für Step Functions in diesem Handbuch enthaltenen Funktionen ermöglichen es Ihnen, automatisch generierte IAM-Rollen zu nutzen, die für die AWS Region gültig sind, in der Sie den State Machine erstellen. Sie können jedoch Ihre eigene IAM-Rolle für eine Zustandsmaschine erstellen.

Wenn Sie eine IAM-Richtlinie für Ihre Zustandsmaschinen erstellen, sollte die Richtlinie die Berechtigungen enthalten, die die Zustandsmaschinen annehmen sollen. Sie können eine bestehende AWS verwaltete Richtlinie als Beispiel verwenden oder Sie können eine benutzerdefinierte Richtlinie von Grund auf neu erstellen, die Ihren spezifischen Anforderungen

entspricht. Weitere Informationen finden Sie im IAM-Benutzerhandbuch unter Erstellen von IAM-Richtlinien

Gehen Sie wie in diesem Abschnitt beschrieben vor, um Ihre eigene IAM-Rolle für eine Zustandsmaschine zu erstellen.

In diesem Beispiel erstellen Sie eine IAM-Rolle mit der Berechtigung, eine Lambda-Funktion aufzurufen.

Eine Rolle für Step Functions erstellen

1. Melden Sie sich bei der IAM-Konsole an und wählen Sie dann Rollen, Rolle erstellen aus.
2. Wählen Sie auf der Seite Vertrauenswürdige Entität auswählen unter AWS Service die Option Step Functions aus der Liste aus, und klicken Sie dann auf Weiter: Berechtigungen.
3. Wählen Sie auf der Seite Attached permissions policy Next: Review aus.
4. Geben Sie auf der Seite Review (Prüfen) `StepFunctionsLambdaRole` für Role Name (Rollenname) ein und wählen Sie dann Create role (Rolle erstellen).

   Die IAM-Rolle wird in der Rollenliste angezeigt.

Weitere Informationen zu IAM-Berechtigungen und -Richtlinien finden Sie unter Access Management im IAM-Benutzerhandbuch.

Vermeiden Sie dienstübergreifende Probleme mit verwirrten Stellvertretern

Das Problem des verwirrten Stellvertreters ist ein Sicherheitsproblem, bei dem eine Entität, die keine Berechtigung zur Durchführung einer Aktion hat, eine privilegiertere Entität zur Durchführung der Aktion zwingen kann. In: AWS Dienststellenübergreifender Identitätswechsel kann zu einem Problem mit dem verwirrten Stellvertreter führen. Ein serviceübergreifender Identitätswechsel kann auftreten, wenn ein Service (der Anruf-Service) einen anderen Service anruft (den aufgerufenen Service). Diese Art des Identitätswechsels kann konto- und dienstübergreifend erfolgen. Der Anruf-Dienst kann so manipuliert werden, dass er seine Berechtigungen verwendet, um auf die Ressourcen eines anderen Kunden zu reagieren, auf die er sonst nicht zugreifen dürfte.

Um zu verhindern, dass Abgeordnete verwirrt werden, AWS bietet dieses Tool Tools, mit denen Sie Ihre Daten für alle Dienste mit Dienstprinzipalen schützen können, denen Zugriff auf Ressourcen in Ihrem Konto gewährt wurde. In diesem Abschnitt geht es speziell um die Vermeidung von Problemen mit verwirrten AWS Step Functions Stellvertretern zwischen den einzelnen Diensten. Weitere

Informationen zu diesem Thema finden Sie jedoch im IAM-Benutzerhandbuch im Abschnitt „Probleme mit verwirrten Stellvertretern“.

Wir empfehlen die Verwendung der Kontextschlüssel `aws:SourceArn`und der `aws:SourceAccount`globalen Bedingungsschlüssel in Ressourcenrichtlinien, um die Berechtigungen einzuschränken, Step Functions die einem anderen Dienst den Zugriff auf Ihre Ressourcen gewähren. Verwenden Sie `aws:SourceArn`, wenn Sie nur eine Ressource mit dem betriebsübergreifenden Zugriff verknüpfen möchten. Verwenden Sie `aws:SourceAccount`, wenn Sie zulassen möchten, dass Ressourcen in diesem Konto mit der betriebsübergreifenden Verwendung verknüpft werden.

Der effektivste Weg, um sich vor dem Confused-Deputy-Problem zu schützen, ist die Verwendung des globalen Bedingungskontext-Schlüssels `aws:SourceArn` mit dem vollständigen ARN der Ressource. Wenn Sie den vollständigen ARN der Ressource nicht kennen oder wenn Sie mehrere Ressourcen angeben, verwenden Sie den `aws:SourceArn` globalen Kontextbedingungsschlüssel mit Platzhalterzeichen (*) für die unbekannten Teile des ARN. z. B. `arn:aws:states:*:111122223333:*`.

Hier ist ein Beispiel für eine vertrauenswürdige Richtlinie, das zeigt, wie Sie Step Functions verwenden `aws:SourceArn` können, um das Problem des verwirrten Stellvertreters zu verhindern. `aws:SourceAccount`

```
{
  "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"
           }
        }
```

```
      }
   ]
}
```

Anfügen einer Inline-Richtlinie

Step Functions kann andere Dienste direkt in einem `Task` Bundesstaat steuern. Fügen Sie Inline-Richtlinien hinzu, damit Step Functions auf die API-Aktionen der Dienste zugreifen kann, die Sie kontrollieren müssen.

1. Öffnen Sie die IAM-Konsole, wählen Sie Rollen, suchen Sie nach Ihrer Step Functions Functions-Rolle und wählen Sie diese Rolle aus.
2. Wählen Sie Add inline Policy (Inline-Richtlinie auswählen) aus.
3. Verwenden Sie den Visual Editor (visuellen Editor) oder die Registerkarte JSON zum Erstellen von Richtlinien für Ihre Rolle.

Weitere Informationen darüber, wie AWS Step Functions Sie andere AWS Dienste steuern können, finden Sie unterVerwendung AWS Step Functions mit anderen Diensten.

> **Note**
>
> Beispiele für IAM-Richtlinien, die von der Step Functions Functions-Konsole erstellt wurden, finden Sie unterIAM-Richtlinien für integrierte Dienste.

Granulare IAM-Berechtigungen für Benutzer ohne Administratorrechte erstellen

Die standardmäßigen verwalteten Richtlinien in IAM, z. B.`ReadOnly`, decken nicht alle Arten von Berechtigungen vollständig ab. AWS Step Functions In diesem Abschnitt werden diese verschiedenen Arten von Berechtigungen beschrieben und Beispielkonfigurationen bereitgestellt.

Step Functions hat vier Kategorien von Berechtigungen. Je nach der Art des Zugriffs, den Sie einem Benutzer bereitstellen möchten, können Sie den Zugriff anhand der Berechtigungen in diesen Kategorien steuern.

Service Level-Berechtigungen

Gilt für Komponenten der API, die nicht auf eine bestimmte Ressource einwirken.

Berechtigungen auf Stufe des Zustandsautomaten

Wenden Sie sie auf alle API-Komponenten an, die an einem spezifischen Zustandsautomaten agieren.

Berechtigungen auf Ausführungsebene

Wenden Sie sie auf alle API-Komponenten an, die an einer spezifischen Ausführung agieren.

Berechtigungen auf Aktivitätsebene

Wenden Sie sie auf alle API-Komponenten an, die an einer spezifischen Aktivität oder auf einer bestimmten Instance einer Aktivität agieren.

Service Level-Berechtigungen

Diese Berechtigungsstufe gilt für alle API-Aktionen, die sich nicht auf eine bestimmte Ressource auswirken. Dazu gehören`CreateStateMachine`, `CreateActivity`,`ListStateMachines`,`ListActivities`, und`ValidationStateMachineDefinition`.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:ListStateMachines",
        "states:ListActivities",
        "states:CreateStateMachine",
        "states:CreateActivity",
        "states:ValidationStateMachineDefinition",
      ],
      "Resource": [
        "arn:aws:states:*:*:*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": [
```

```
        "arn:aws:iam:::role/my-execution-role"
      ]
    }
  ]
}
```

Berechtigungen auf Stufe des Zustandsautomaten

Diese Berechtigungsstufe gilt für alle API-Aktionen, die an einem spezifischen Zustandsautomaten agieren. Für diese API-Operationen ist der Amazon-Ressourcenname (ARN) der Zustandsmaschine als Teil der Anfrage erforderlich, z. B. DeleteStateMachineDescribeStateMachine,StartExecution, undListExecutions.

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

Berechtigungen auf Ausführungsebene

Diese Berechtigungsstufe gilt für alle API-Aktionen, die an einer spezifischen Ausführung agieren. Diese API-Vorgänge erfordern die ARN der Ausführung als Teil der Anforderung, wie etwaDescribeExecution, GetExecutionHistory und StopExecution.

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:DescribeExecution",
        "states:DescribeStateMachineForExecution",
        "states:GetExecutionHistory",
        "states:StopExecution"
      ],
      "Resource": [
        "arn:aws:states:*:*:execution:*:ExecutionPrefix*"
      ]
    }
  ]
}
```

Berechtigungen auf Aktivitätsebene

Diese Berechtigungsstufe gilt für sämtliche API-Aktionen, die an einer spezifischen Aktivität oder auf einer bestimmten Instance davon agieren. Für diese API-Operationen ist der ARN der Aktivität oder das Token der Instanz als Teil der Anfrage erforderlich, z. B. DeleteActivityDescribeActivity,GetActivityTask, undSendTaskHeartbeat.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:DescribeActivity",
        "states:DeleteActivity",
        "states:GetActivityTask",
        "states:SendTaskHeartbeat"
      ],
      "Resource": [
        "arn:aws:states:*:*:activity:ActivityPrefix*"
      ]
    }
  ]
}
```

Zugreifen auf Ressourcen AWS-Konten in anderen Workflows

Step Functions bietet kontoübergreifenden Zugriff auf Ressourcen, die AWS-Konten in Ihren Workflows unterschiedlich konfiguriert sind. Mithilfe der Step Functions Functions-Dienstintegrationen können Sie jede kontoübergreifende AWS Ressource aufrufen, auch wenn diese AWS-Service keine ressourcenbasierten Richtlinien oder kontenübergreifenden Aufrufe unterstützt.

Nehmen wir zum Beispiel an, Sie besitzen zwei Geräte AWS-Konten, die als Entwicklung und Testen bezeichnet werden. AWS-Region Mithilfe des kontoübergreifenden Zugriffs kann Ihr Workflow im Entwicklungskonto auf Ressourcen wie Amazon S3 S3-Buckets, Amazon DynamoDB-Tabellen und Lambda-Funktionen zugreifen, die im Testkonto verfügbar sind.

> **Important**
>
> IAM-Rollen und ressourcenbasierte Richtlinien delegieren den Zugriff auf Konten nur innerhalb einer einzelnen Partition. Nehmen wir zum Beispiel an, Sie haben ein Konto in US West (Nordkalifornien) in der Standardpartition `aws`. Sie haben auch ein Konto in China (Peking) in der `aws-cn`-Partition. Sie können keine ressourcenbasierte Amazon S3-Richtlinie in Ihrem Konto in China (Peking) verwenden, um Benutzern in Ihrem Standard-`aws`-Konto den Zugriff zu ermöglichen.

Weitere Informationen zum kontoübergreifenden Zugriff finden Sie unter Logik zur kontenübergreifenden Bewertung von Richtlinien im IAM-Benutzerhandbuch.

Obwohl jeder die vollständige Kontrolle über seine eigenen Ressourcen AWS-Konto behält, können Sie mit Step Functions Schritte in Ihren Workflows neu organisieren, austauschen, hinzufügen oder entfernen, ohne Code anpassen zu müssen. Sie können dies auch dann tun, wenn sich die Prozesse ändern oder Anwendungen weiterentwickeln.

Sie können auch Ausführungen von verschachtelten Zustandsmaschinen aufrufen, sodass sie für verschiedene Konten verfügbar sind. Auf diese Weise werden Ihre Arbeitsabläufe effizient getrennt und isoliert. Wenn Sie das `.sync`Serviceintegrationsmuster in Ihren Workflows verwenden, die auf einen anderen Step Functions-Workflow in einem anderen Konto zugreifen, verwendet Step Functions Polling, das Ihr zugewiesenes Kontingent verbraucht. Weitere Informationen finden Sie unter Ausführen einer Aufgabe (.sync).

> **Note**
>
> Derzeit sind die regionsübergreifende AWS SDK-Integration und der regionsübergreifende AWS Ressourcenzugriff in Step Functions nicht verfügbar.

Inhalt

Die wichtigsten Konzepte in diesem Thema

Rolle bei der Ausführung

Eine IAM-Rolle, die Step Functions verwendet, um Code auszuführen und auf AWS Ressourcen zuzugreifen, z. B. die Invoke-Aktion der AWS Lambda Funktion.

Serviceintegration

Die API-Aktionen für die AWS SDK-Integration, die von einem `Task` Status in Ihren Workflows aus aufgerufen werden können.

Quellkonto

Ein AWS-Konto , dem die Zustandsmaschine gehört und deren Ausführung gestartet hat.

Zielkonto

Ein AWS-Konto , zu dem Sie kontoübergreifende Anrufe tätigen.

Zielrolle

Eine IAM-Rolle im Zielkonto, die die Zustandsmaschine für Aufrufe von Ressourcen übernimmt, die dem Zielkonto gehören.

Einen Job ausführen (.sync)

Ein Dienstintegrationsmuster, das zum Aufrufen von Diensten verwendet wird, wie AWS Batch z. Außerdem wartet eine Step Functions Functions-Zustandsmaschine darauf, dass ein Job

abgeschlossen ist, bevor sie zum nächsten Status übergeht. Um anzugeben, dass Step Functions warten soll, fügen Sie das `.sync` Suffix in das `Resource` Feld in Ihrer `Task` Statusdefinition an.

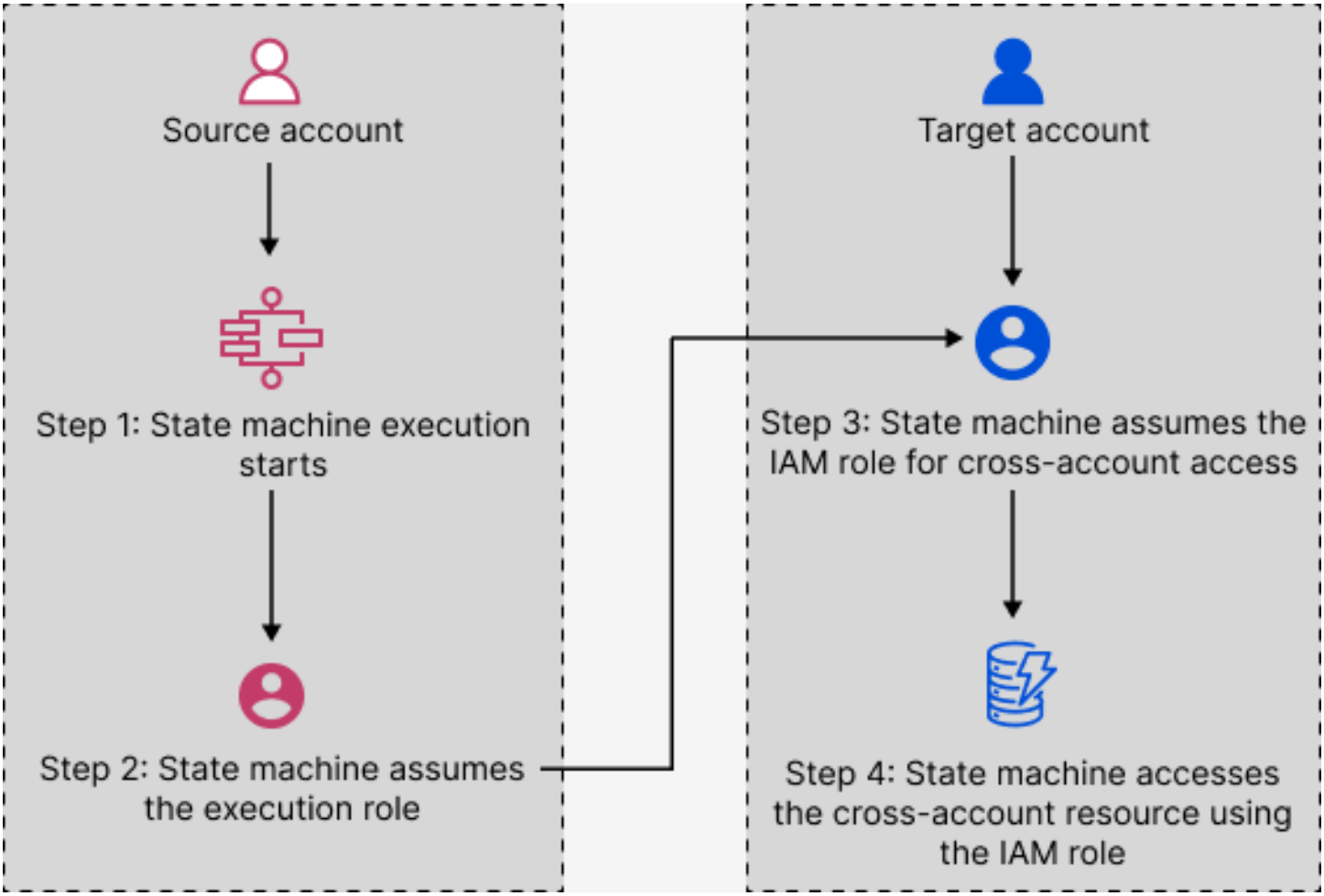
Kontenübergreifende Ressourcen aufrufen

Gehen Sie wie folgt vor, um eine kontoübergreifende Ressource in Ihren Workflows aufzurufen:

1. Erstellen Sie eine IAM-Rolle in dem Zielkonto, das die Ressource enthält. Diese Rolle gewährt dem Quellkonto, das die Zustandsmaschine enthält, Berechtigungen für den Zugriff auf die Ressourcen des Zielkontos.
2. Geben Sie in der Definition des `Task` Status die IAM-Zielrolle an, die von der Zustandsmaschine übernommen werden soll, bevor die kontoübergreifende Ressource aufgerufen wird.
3. Ändern Sie die Vertrauensrichtlinie in der Ziel-IAM-Rolle, sodass das Quellkonto diese Rolle vorübergehend übernehmen kann. Die Vertrauensrichtlinie muss den Amazon-Ressourcennamen (ARN) der Zustandsmaschine enthalten, die im Quellkonto definiert ist. Definieren Sie außerdem die entsprechenden Berechtigungen in der IAM-Zielrolle, um die AWS Ressource aufzurufen.
4. Aktualisieren Sie die Ausführungsrolle des Quellkontos so, dass sie die erforderliche Berechtigung für die Übernahme der Ziel-IAM-Rolle enthält.

Ein Beispiel finden Sie unter Tutorial: Zugriff auf kontoübergreifende Ressourcen AWS.

> **Note**
>
> Sie können Ihren Zustandsmaschine so konfigurieren, dass er eine IAM-Rolle für den Zugriff auf Ressourcen von mehreren Personen annimmt. AWS-Konten Eine Zustandsmaschine kann jedoch jeweils nur eine IAM-Rolle übernehmen.



Tutorial: Zugriff auf kontoübergreifende Ressourcen AWS

Mit der Unterstützung für kontoübergreifenden Zugriff in Step Functions können Sie Ressourcen gemeinsam nutzen, die unterschiedlich AWS-Konten konfiguriert sind. In diesem Tutorial führen wir Sie durch den Prozess des Zugriffs auf eine kontoübergreifende Lambda-Funktion, die in einem Konto namens Production definiert ist. Diese Funktion wird von einer Zustandsmaschine in einem Konto namens Development aus aufgerufen. In diesem Tutorial wird das Entwicklungskonto als Quellkonto bezeichnet und das Produktionskonto ist das Zielkonto, das die Ziel-IAM-Rolle enthält.

Zunächst geben Sie in der Definition Ihres `Task` Bundesstaates die IAM-Zielrolle an, die die Zustandsmaschine annehmen muss, bevor sie die kontoübergreifende Lambda-Funktion aufruft. Ändern Sie anschließend die Vertrauensrichtlinie in der Ziel-IAM-Rolle, sodass das Quellkonto vorübergehend die Zielrolle übernehmen kann. Um die AWS Ressource aufzurufen, definieren Sie außerdem die entsprechenden Berechtigungen in der Ziel-IAM-Rolle. Aktualisieren Sie abschließend die Ausführungsrolle des Quellkontos, um die erforderliche Berechtigung für die Übernahme der Zielrolle anzugeben.

Sie können Ihren Zustandsmaschine so konfigurieren, dass er eine IAM-Rolle für den Zugriff auf Ressourcen aus mehreren AWS-Konten Quellen annimmt. Eine Zustandsmaschine kann jedoch je nach Definition des `Task` Status jeweils nur eine IAM-Rolle übernehmen.

> **Note**
>
> Derzeit sind die regionsübergreifende AWS SDK-Integration und der regionsübergreifende AWS Ressourcenzugriff in Step Functions nicht verfügbar.

Inhalt

Voraussetzungen

- In diesem Tutorial wird am Beispiel einer Lambda-Funktion demonstriert, wie der kontenübergreifende Zugriff eingerichtet wird. Sie können jede andere AWS Ressource verwenden, stellen Sie jedoch sicher, dass Sie die Ressource in einem anderen Konto konfiguriert haben.

  > **Important**
  >
  > IAM-Rollen und ressourcenbasierte Richtlinien delegieren den Zugriff auf Konten nur innerhalb einer einzelnen Partition. Nehmen wir zum Beispiel an, Sie haben ein Konto in US West (Nordkalifornien) in der Standardpartition `aws`. Sie haben auch ein Konto in China (Peking) in der `aws-cn`-Partition. Sie können keine ressourcenbasierte Amazon S3-Richtlinie in Ihrem Konto in China (Peking) verwenden, um Benutzern in Ihrem Standard-`aws`-Konto den Zugriff zu ermöglichen.

- Notieren Sie sich den Amazon-Ressourcennamen (ARN) der kontoübergreifenden Ressource in einer Textdatei. Später in diesem Tutorial werden Sie diesen ARN in der Zustandsdefinition Ihrer Zustandsmaschine `Task` angeben. Das Folgende ist ein Beispiel für eine Lambda-Funktion ARN:

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

- Stellen Sie sicher, dass Sie die Ziel-IAM-Rolle erstellt haben, die die Zustandsmaschine übernehmen muss.

Schritt 1: Aktualisieren Sie die Aufgabenstatusdefinition, um die Zielrolle anzugeben

Fügen Sie im `Task` Status Ihres Workflows ein `Credentials` Feld hinzu, das die Identität enthält, die die Zustandsmaschine annehmen muss, bevor sie die kontoübergreifende Lambda-Funktion aufruft.

Das folgende Verfahren zeigt, wie Sie auf eine kontoübergreifende Lambda-Funktion zugreifen, die aufgerufen wird. `Echo` Sie können jede AWS Ressource aufrufen, indem Sie die folgenden Schritte ausführen.

1. Öffnen Sie die Step Functions Functions-Konsole und wählen Sie Create State Machine.
2. Wählen Sie auf der Seite „Erstellungsmethode auswählen“ die Option Workflow visuell gestalten aus und behalten Sie alle Standardauswahlen bei.
3. Um Workflow Studio zu öffnen, wählen Sie Weiter.
4. Ziehen Sie auf der Registerkarte Aktionen einen `Task` Status per Drag-and-Drop auf die Arbeitsfläche. Dadurch wird die kontoübergreifende Lambda-Funktion aufgerufen, die diesen Status verwendet. `Task`
5. Gehen Sie auf der Registerkarte Konfiguration wie folgt vor:

   a. Benennen Sie den Status um in**`Cross-account call`**.

   b. Wählen Sie für Funktionsname die Option Funktionsname eingeben aus, und geben Sie dann den ARN der Lambda-Funktion in das Feld ein. z. B. `arn:aws:lambda:us-east-2:111122223333:function:Echo`.

   c. Geben Sie für Provide IAM role ARN den ARN für die IAM-Zielrolle an. z. B. `arn:aws:iam::111122223333:role/LambdaRole`.

   > **Tip**
   >
   > Alternativ können Sie auch einen Referenzpfad zu einem vorhandenen Schlüssel-Wert-Paar in der JSON-Eingabe des Bundesstaates angeben, die den ARN der IAM-Rolle enthält. Wählen Sie dazu Get IAM role ARN at runtime from state input

aus. Ein Beispiel für die Angabe eines Werts mithilfe eines Referenzpfads finden Sie unterJsonPath als IAM-Rollen-ARN angeben.

6. Wählen Sie Weiter aus.
7. Wählen Sie auf der Seite „Generierten Code überprüfen“ die Option Weiter aus.
8. Geben Sie auf der Seite „Einstellungen für Zustandsmaschine angeben“ Details für den neuen Zustandsmaschine an, z. B. einen Namen, Berechtigungen und eine Protokollierungsebene.
9. Wählen Sie Create State Machine (Zustandsautomaten erstellen).
10. Notieren Sie sich den IAM-Rollen-ARN der Zustandsmaschine und den ARN der Zustandsmaschine in einer Textdatei. Sie müssen diese ARNs in der Vertrauensrichtlinie des Zielkontos angeben.

Ihre `Task` Bundesstaatendefinition sollte jetzt der folgenden Definition ähneln.

```
{
  "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
    }
  }
}
```

Schritt 2: Aktualisieren Sie die Vertrauensrichtlinie der Zielrolle

Die IAM-Rolle muss im Zielkonto vorhanden sein und Sie müssen dessen Vertrauensrichtlinie ändern, damit das Quellkonto diese Rolle vorübergehend übernehmen kann. Darüber hinaus können Sie steuern, wer die Ziel-IAM-Rolle übernehmen kann.

Nachdem Sie die Vertrauensstellung erstellt haben, kann ein Benutzer aus dem Quellkonto den AssumeRoleAPI-Vorgang AWS Security Token Service (AWS STS) verwenden. Dieser Vorgang stellt

temporäre Sicherheitsanmeldedaten bereit, die den Zugriff auf AWS Ressourcen in einem Zielkonto ermöglichen.

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie im Navigationsbereich der Konsole Rollen aus und verwenden Sie dann das Suchfeld, um nach der IAM-Zielrolle zu suchen. z. B. *LambdaRole*.
3. Wählen Sie die Registerkarte Trust relationships (Vertrauensstellungen).
4. Wählen Sie Vertrauensrichtlinie bearbeiten und fügen Sie die folgende Vertrauensrichtlinie ein. Achten Sie darauf, die AWS-Konto Nummer und den ARN der IAM-Rolle zu ersetzen. Das `sts:ExternalId` Feld steuert außerdem, wer die Rolle übernehmen kann. Der Name der Zustandsmaschine darf nur Zeichen enthalten, die die AWS Security Token Service `AssumeRole` API unterstützt. Weitere Informationen finden Sie AssumeRolein der AWS Security Token Service API-Referenz.

```
{
  "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. Lassen Sie dieses Fenster geöffnet und fahren Sie mit dem nächsten Schritt fort, um weitere Aktionen durchzuführen.

Schritt 3: Fügen Sie die erforderliche Berechtigung in der Zielrolle hinzu

Die Berechtigungen in den IAM-Richtlinien bestimmen, ob eine bestimmte Anfrage zulässig oder verweigert wird. Die Ziel-IAM-Rolle muss über die richtige Berechtigung zum Aufrufen der Lambda-Funktion verfügen.

1. Wählen Sie die Registerkarte Berechtigungen.
2. Wählen Sie Add permissions (Berechtigungen hinzufügen) und dann Create inline policy (Inline-Richtlinie erstellen) aus.
3. Wählen Sie die Registerkarte JSON und ersetzen Sie den vorhandenen Inhalt durch die folgende Berechtigung. Stellen Sie sicher, dass Sie den ARN Ihrer Lambda-Funktion ersetzen.

```
{
  "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. Wählen Sie Richtlinie prüfen.
5. Geben Sie auf der Seite „Richtlinie überprüfen“ einen Namen für die Berechtigung ein und wählen Sie dann Richtlinie erstellen aus.

Schritt 4: Fügen Sie der Ausführungsrolle die Berechtigung hinzu, um die Zielrolle anzunehmen

Step Functions generiert die AssumeRoleRichtlinie nicht automatisch für alle kontoübergreifenden Serviceintegrationen. Sie müssen der Ausführungsrolle der Zustandsmaschine die erforderliche Berechtigung hinzufügen, damit sie eine IAM-Zielrolle in einer oder mehreren Rollen übernehmen kann. AWS-Konten

1. Öffnen Sie die Ausführungsrolle Ihrer Zustandsmaschine in der IAM-Konsole unter https://console.aws.amazon.com/iam/. So gehen Sie vor:

   a. Öffnen Sie die Zustandsmaschine, die Sie in Schritt 1 erstellt haben, im Quellkonto.

b. Wählen Sie auf der Detailseite der Statusmaschine die IAM-Rolle ARN aus.

2. Wählen Sie auf der Registerkarte Berechtigungen die Option Berechtigungen hinzufügen und dann Inline-Richtlinie erstellen aus.
3. Wählen Sie die Registerkarte JSON und ersetzen Sie den vorhandenen Inhalt durch die folgende Berechtigung. Stellen Sie sicher, dass Sie den ARN Ihrer Lambda-Funktion ersetzen.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Resource": "arn:aws:iam::111122223333:role/LambdaRole"  // The target role
 to be assumed
    }
  ]
}
```

4. Wählen Sie Richtlinie prüfen.
5. Geben Sie auf der Seite „Richtlinie überprüfen“ einen Namen für die Berechtigung ein und wählen Sie dann Richtlinie erstellen aus.

Kontoübergreifender Zugriff für das Integrationsmuster „.sync“

Wenn Sie die `.sync` Serviceintegrationsmuster in Ihren Workflows verwenden, fragt Step Functions die aufgerufene kontenübergreifende Ressource ab, um zu bestätigen, dass die Aufgabe abgeschlossen ist. Dies führt zu einer leichten Verzögerung zwischen der tatsächlichen Abschlusszeit der Aufgabe und dem Zeitpunkt, zu dem Step Functions die Aufgabe als abgeschlossen erkennt. Die Ziel-IAM-Rolle benötigt die erforderlichen Berechtigungen für einen `.sync` Aufruf, um diese Abfrageschleife abzuschließen. Zu diesem Zweck muss die Ziel-IAM-Rolle über eine Vertrauensrichtlinie verfügen, die es dem Quellkonto ermöglicht, diese zu übernehmen. Darüber hinaus benötigt die Ziel-IAM-Rolle die erforderlichen Berechtigungen, um die Abfrageschleife abzuschließen.

> **Note**
>
> Für verschachtelte Express-Workflows wird dies derzeit `arn:aws:states:::states:startExecution.sync` nicht unterstützt. Verwenden Sie stattdessen `arn:aws:states:::aws-sdk:sfn:startSyncExecution`.

Aktualisierung der Vertrauensrichtlinie für .sync-Aufrufe

Aktualisieren Sie die Vertrauensrichtlinie Ihrer IAM-Zielrolle, wie im folgenden Beispiel gezeigt. Das `sts:ExternalId` Feld steuert außerdem, wer die Rolle übernehmen kann. Der Name der Zustandsmaschine darf nur Zeichen enthalten, die die AWS Security Token Service `AssumeRole` API unterstützt. Weitere Informationen finden Sie AssumeRolein der AWS Security Token Service API-Referenz.

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

Für .sync-Aufrufe sind Berechtigungen erforderlich

Um die für Ihre Zustandsmaschine erforderlichen Berechtigungen zu gewähren, aktualisieren Sie die erforderlichen Berechtigungen für die IAM-Zielrolle. Weitere Informationen finden Sie unter the section called “IAM-Richtlinien für integrierte Dienste”. Die EventBridge Amazon-Berechtigungen aus den Beispielrichtlinien sind nicht erforderlich. Um beispielsweise eine Zustandsmaschine zu starten, fügen Sie die folgenden Berechtigungen hinzu.

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

Amazon VPC-Endpunkte für Step Functions

Wenn Sie Amazon Virtual Private Cloud (Amazon VPC) zum Hosten Ihrer AWS Ressourcen verwenden, können Sie eine Verbindung zwischen Ihrer Amazon VPC und AWS Step Functions Workflows herstellen. Sie können diese Verbindung mit Ihren Step Functions Functions-Workflows verwenden, ohne das öffentliche Internet zu überqueren. Amazon VPC-Endpunkte werden von Standard-Workflows, Express-Workflows und synchronen Express-Workflows unterstützt.

Mit Amazon VPC können Sie AWS Ressourcen in einem benutzerdefinierten virtuellen Netzwerk starten. Mit einer VPC können Sie Netzwerkeinstellungen, wie IP-Adressbereich, Subnetze, Routing-Tabellen und Netzwerk-Gateways, steuern. Weitere Informationen zu VPCs finden Sie im Amazon-VPC-Benutzerhandbuch.

Um Ihre Amazon VPC mit Step Functions zu verbinden, müssen Sie zunächst einen VPC-Schnittstellen-Endpunkt definieren, über den Sie Ihre VPC mit anderen Services verbinden können. AWS Der Endpunkt bietet eine zuverlässige, skalierbare Konnektivität, ohne dass ein Internet-

Gateway, eine NAT-Instance (Network Address Translation) oder eine VPN-Verbindung erforderlich ist. Weitere Informationen finden Sie unter Schnittstellen-VPC-Endpunkte (AWS PrivateLink) im Amazon-VPC-Benutzerhandbuch.

Erstellen des Endpunkts

Sie können einen AWS Step Functions Endpunkt in Ihrer VPC mit dem AWS Management Console, dem AWS Command Line Interface (AWS CLI), einem AWS SDK, der AWS Step Functions API oder AWS CloudFormation erstellen.

Informationen zum Erstellen und Konfigurieren eines Endpunkts über die Amazon-VPC-Konsole oder die AWS CLI finden Sie unter Creating an Interface Endpoint (Erstellen eines Schnittstellenendpunkts) im Amazon-VPC-Benutzerhandbuch.

> **Note**
>
> Wenn Sie einen Endpunkt erstellen, geben Sie Step Functions als den Service an, zu dem Ihre VPC eine Verbindung herstellen soll. In der Amazon VPC-Konsole variieren die Servicenamen je nach AWS Region. Wenn Sie beispielsweise USA Ost (Nord-Virginia) wählen, lautet der Service-Name für Standard-Workflows und Express-Workflows com.amazonaws.us-east-1.states und der Servicename für Synchronous Express Workflows ist com.amazonaws.us-east-1.sync-states.

> **Note**
>
> Es ist möglich, VPC-Endpunkte über Private DNS zu verwenden, ohne den Endpunkt im SDK zu überschreiben. Wenn Sie den Endpunkt im SDK für synchrone Express-Workflows jedoch überschreiben möchten, müssen Sie die Konfiguration auf setzen. `DisableHostPrefixInjection` `true` Beispiel (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();
```

Informationen zum Erstellen und Konfigurieren eines Endpunkts mithilfe AWS CloudFormation von finden Sie in der Ressource AWS: :EC2: :VpcEndpoint im Benutzerhandbuch.AWS CloudFormation

Amazon VPC-Endpunktrichtlinien

Um den Konnektivitätszugriff auf Step Functions zu kontrollieren, können Sie beim Erstellen eines Amazon VPC-Endpunkts eine AWS Identity and Access Management (IAM-) Endpunktrichtlinie anhängen. Sie können komplexe IAM-Regeln erstellen, indem Sie mehrere Endpunktrichtlinien anhängen. Weitere Informationen finden Sie hier:

- Amazon Virtual Private Cloud-Endpunktrichtlinien für Step Functions
- Granulare IAM-Berechtigungen für Benutzer ohne Administratorrechte erstellen
- Kontrollieren des Zugriffs auf Services mit VPC-Endpunkten

Amazon Virtual Private Cloud-Endpunktrichtlinien für Step Functions

Sie können eine Amazon VPC-Endpunktrichtlinie für Step Functions erstellen, in der Sie Folgendes angeben:

- Prinzipal, der die Aktionen ausführen kann.
- Aktionen, die ausgeführt werden können
- Die Ressourcen, auf denen die Aktionen ausgeführt werden können.

Das folgende Beispiel zeigt eine Amazon VPC-Endpunktrichtlinie, die es einem Benutzer ermöglicht, Zustandsmaschinen zu erstellen, und allen anderen Benutzern die Erlaubnis verweigert, Zustandsmaschinen zu löschen. Die Beispielrichtlinie gewährt auch allen -Benutzern die Ausführungsberechtigung.

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

Weitere Informationen zum Erstellen von Endpunktrichtlinien finden Sie unter:

- Granulare IAM-Berechtigungen für Benutzer ohne Administratorrechte erstellen
- Kontrollieren des Zugriffs auf Services mit VPC-Endpunkten

IAM-Richtlinien für integrierte Dienste

Wenn Sie eine Zustandsmaschine in der AWS Step Functions Konsole erstellen, erstellt Step Functions eine AWS Identity and Access Management (IAM-) Richtlinie, die auf den in Ihrer Zustandsmaschinen-Definition verwendeten Ressourcen wie folgt basiert:

- Wenn Ihre Zustandsmaschine eine der optimierten Integrationen verwendet, erstellt Step Functions eine Richtlinie mit den erforderlichen Berechtigungen und Rollen für Ihre Zustandsmaschine. (Ausnahme: Für die MediaConvert Integration müssen Sie Berechtigungen manuell einrichten — siehe IAM-Richtlinien für AWS Elemental MediaConvert.)
- Wenn Ihre Zustandsmaschine eine der AWS SDK-Integrationen verwendet, wird eine IAM-Rolle mit Teilberechtigungen erstellt. Anschließend können Sie die IAM-Konsole verwenden, um alle fehlenden Rollenrichtlinien hinzuzufügen.

Die folgenden Beispiele zeigen, wie Step Functions eine IAM-Richtlinie auf der Grundlage Ihrer State-Machine-Definition generiert. Elemente in der Beispiel-Code, wie z. B. *[[resourceName]]*, werden durch die statischen Ressourcen ersetzt, die in Ihrer Definition des Zustandsautomaten angegeben sind. Wenn Sie über mehrere statische Ressourcen verfügen, gibt es für jede Ressource einen Eintrag in der IAM-Rolle.

Dynamische und statische Ressourcen

Statische Ressourcen werden direkt im Aufgabenstatus Ihres Zustandsautomaten definiert. Wenn Sie die Informationen zu den Ressourcen, die Sie aufrufen möchten, direkt in Ihren Aufgabenstatus aufnehmen, erstellt Step Functions eine IAM-Rolle nur für diese Ressourcen.

Dynamische Ressourcen sind die Ressourcen, die an Ihre Zustandseingabe übergeben werden und auf die über einen Pfad zugegriffen wird (siehe Pfade). Wenn Sie dynamische Ressourcen an Ihre Aufgabe übergeben, erstellt Step Functions eine Richtlinie mit mehr Rechten, die Folgendes spezifiziert:`"Resource": "*"`.

Zusätzliche Berechtigungen für Aufgaben, die das Run a Job-Muster verwenden

Für Aufgaben, die das Run a Job-Muster verwenden (solche, die auf enden`.sync`), sind zusätzliche Berechtigungen erforderlich, um die API-Aktionen verbundener Dienste zu überwachen und eine Antwort von ihnen zu erhalten. Die entsprechenden Richtlinien beinhalten mehr Berechtigungen als für Aufgaben, die die Muster „Antwort anfordern“ oder „Auf Rückruf warten“ verwenden. Informationen Muster der Serviceintegration zu synchronen Aufgaben finden Sie unter.

> **Note**
>
> Sie müssen zusätzliche Berechtigungen für Dienstintegrationen bereitstellen, die das Run a Job (.sync) -Muster unterstützen.

Step Functions verwendet zwei Methoden, um den Status eines Jobs zu überwachen, wenn ein Job auf einem verbundenen Dienst ausgeführt wird: Polling und Ereignisse.

Für Abfragen sind Genehmigungen `Describe` oder `Get` API-Aktionen wie `ecs:DescribeTasks` oder erforderlich. `glue:GetJobRun` Wenn diese Berechtigungen in Ihrer Rolle fehlen, kann Step Functions den Status Ihres Jobs möglicherweise nicht ermitteln. Dies liegt daran, dass einige Run a Job (.sync) -Dienstintegrationen keine EventBridge Ereignisse unterstützen und einige Dienste Ereignisse nur nach bestem Wissen senden.

Ereignisse, die von AWS Services an Amazon gesendet EventBridge werden, werden mithilfe einer verwalteten Regel an Step Functions weitergeleitet und erfordern Berechtigungen für `events:PutTargetsevents:PutRule`, und`events:DescribeRule`. Wenn diese Berechtigungen in Ihrer Rolle fehlen, kann es zu einer Verzögerung kommen, bis Step Functions von der Fertigstellung Ihres Jobs erfährt. Weitere Informationen zu EventBridge Ereignissen finden Sie unter Ereignisse von AWS Diensten.

> **Note**
>
> Bei Aufgaben vom Typ Run a Job (.sync), die sowohl Abfragen als auch Ereignisse unterstützen, kann es sein, dass Ihre Aufgabe mithilfe von Ereignissen trotzdem ordnungsgemäß abgeschlossen wird. Dies kann auch dann der Fall sein, wenn Ihre Rolle nicht über die erforderlichen Berechtigungen für Abfragen verfügt. In diesem Fall stellen Sie möglicherweise nicht sofort fest, dass die Abfrageberechtigungen falsch sind oder fehlen. In dem seltenen Fall, dass das Ereignis nicht an Step Functions übermittelt oder von Step Functions verarbeitet werden kann, kann Ihre Ausführung hängen bleiben. Um zu überprüfen, ob Ihre Abfrageberechtigungen korrekt konfiguriert sind, können Sie eine Ausführung in einer Umgebung ohne EventBridge Ereignisse auf folgende Weise ausführen:
>
> - Löschen Sie die verwaltete Regel von EventBridge, die für die Weiterleitung von Ereignissen an Step Functions zuständig ist. Diese verwaltete Regel wird von allen Zustandsmaschinen in Ihrem Konto gemeinsam genutzt. Sie sollten diese Aktion daher nur in einem Test- oder Entwicklungskonto ausführen, um unbeabsichtigte Auswirkungen auf andere Zustandsmaschinen zu vermeiden. Sie können die spezifische verwaltete Regel, die gelöscht werden soll, anhand des `Resource` Felds ermitteln, das `events:PutRule` in der Richtlinienvorlage für den Zieldienst verwendet wird. Die verwaltete Regel wird neu erstellt, wenn Sie das nächste Mal eine Zustandsmaschine erstellen oder aktualisieren, die diese Dienstintegration verwendet. Weitere Informationen zum Löschen von EventBridge Regeln finden Sie unter Regel deaktivieren oder löschen.
> - Testen Sie mit Step Functions Local, das die Verwendung von Ereignissen zum Ausführen von Aufgaben vom Typ Job (.sync) nicht unterstützt. Um Step Functions Local zu verwenden, nehmen Sie die IAM-Rolle an, die von Ihrem State Machine verwendet wird. Möglicherweise müssen Sie die Vertrauensstellung bearbeiten. Setzen Sie die `AWS_SESSION_TOKEN` Umgebungsvariablen `AWS_ACCESS_KEY_IDAWS_SECRET_ACCESS_KEY`, und auf die Werte der angenommenen Rolle und starten Sie dann Step Functions Local mitjava -jar StepFunctionsLocal.jar. Verwenden Sie zuletzt den --endpoint-url Parameter AWS CLI with the, um eine

Zustandsmaschine zu erstellen, eine Ausführung zu starten und den Ausführungsverlauf abzurufen. Weitere Informationen finden Sie unter Zustandsmaschinen lokal testen.

Wenn eine Aufgabe, die das Run a Job (.sync) -Muster verwendet, beendet wird, versucht Step Functions nach besten Kräften, die Aufgabe abzubrechen. Dies erfordert die Erlaubnis für `CancelStop`,`Terminate`, oder `Delete` API-Aktionen wie `batch:TerminateJob` oder. `eks:DeleteCluster` Wenn diese Berechtigungen in Ihrer Rolle fehlen, kann Step Functions Ihre Aufgabe nicht stornieren, und es können zusätzliche Kosten anfallen, während sie weiterhin ausgeführt wird. Weitere Informationen zum Stoppen von Aufgaben finden Sie unter Job ausführen.

Richtlinienvorlagen, die zum Erstellen von IAM-Rollen verwendet werden

Die folgenden Themen enthalten die Richtlinienvorlagen, die verwendet werden, wenn Sie festlegen, dass Step Functions eine neue Rolle für Sie erstellt.

> **Note**
>
> Sehen Sie sich diese Vorlagen an, um zu verstehen, wie Step Functions Ihre IAM-Richtlinien erstellt, und als Beispiel dafür, wie Sie IAM-Richtlinien für Step Functions manuell erstellen können, wenn Sie mit anderen AWS Services arbeiten. Weitere Informationen zu Step Functions Functions-Dienstintegrationen finden Sie unterVerwendung AWS Step Functions mit anderen Diensten.

Themen

- IAM-Richtlinien für Amazon API Gateway
- IAM-Richtlinien für Amazon Athena
- IAM-Richtlinien für AWS Batch
- IAM policies for Amazon Bedrock
- IAM-Richtlinien für AWS CodeBuild
- IAM-Richtlinien für Amazon DynamoDB
- IAM-Richtlinien für Amazon ECS/AWS Fargate
- IAM-Richtlinien für Amazon EKS
- IAM-Richtlinien für Amazon EMR
- IAM-Richtlinien für Amazon EMR auf EKS

- IAM-Richtlinien für Amazon EMR Serverless
- IAM-Richtlinien für Amazon EventBridge
- IAM-Richtlinien für AWS Lambda
- IAM-Richtlinien für AWS Elemental MediaConvert
- IAM-Richtlinien für AWS Glue
- IAM-Richtlinien für AWS Glue DataBrew
- IAM-Richtlinien für Amazon SageMaker
- IAM-Richtlinien für Amazon SNS
- IAM-Richtlinien für Amazon SQS
- IAM-Richtlinien für AWS Step Functions
- IAM-Richtlinien für AWS X-Ray
- Aktivitäten oder keine Aufgaben

IAM-Richtlinien für Amazon API Gateway

Die folgenden Beispielvorlagen zeigen, wie IAM-Richtlinien auf der Grundlage der Ressourcen in Ihrer State-Machine-Definition AWS Step Functions generiert werden. Weitere Informationen finden Sie unter IAM-Richtlinien für integrierte Dienste und Muster der Serviceintegration.

Ressourcen:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "execute-api:Invoke"
            ],
            "Resource": [
                "arn:[[region]]:[[accountId]]:*"
            ]
        }
    ]
}
```

Das folgende Codebeispiel zeigt eine Ressourcenrichtlinie für den Aufruf von 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>"
                    ]
                }
            }
        }
    ]
}
```

IAM-Richtlinien für Amazon Athena

Die folgenden Beispielvorlagen zeigen, wie IAM-Richtlinien auf der Grundlage der Ressourcen in Ihrer State-Machine-Definition AWS Step Functions generiert werden. Weitere Informationen finden Sie unter IAM-Richtlinien für integrierte Dienste und Muster der Serviceintegration.

`StartQueryExecution`

Statische Ressourcen

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": [
            "*"
        ]
    }
]
}
```

Dynamische Ressourcen

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

Ressourcen

```
{
    "Version": "2012-10-17",
    "Statement":[
    {
        "Effect": "Allow",
        "Action": [
            "athena:stopQueryExecution"
        ],
        "Resource": [
```

```
            "arn:aws:athena:{{region}}:{{accountId}}:workgroup/*"
        ]
    }
    ]
}
```

GetQueryExecution

Ressourcen

```
{
    "Version": "2012-10-17",
    "Statement":[
    {
        "Effect": "Allow",
        "Action": [
            "athena:getQueryExecution"
        ],
        "Resource": [
            "arn:aws:athena:{{region}}:{{accountId}}:workgroup/*"
        ]
    }
    ]
}
```

GetQueryResults

Ressourcen

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

IAM-Richtlinien für AWS Batch

Die folgenden Beispielvorlagen zeigen, wie IAM-Richtlinien auf der Grundlage der Ressourcen in Ihrer State-Machine-Definition AWS Step Functions generiert werden. Weitere Informationen finden Sie unter IAM-Richtlinien für integrierte Dienste und Muster der Serviceintegration.

Da die Zugriffskontrolle auf Ressourcenebene teilweise AWS Batch unterstützt wird, müssen Sie Folgendes verwenden. `"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

Wenn Sie eine Zustandsmaschine mithilfe der Konsole erstellen, Step Functions wird automatisch eine Ausführungsrolle für Ihre Zustandsmaschine mit den geringsten erforderlichen Rechten erstellt. Diese automatisch generierten IAM Rollen gelten für den, AWS-Region in dem Sie den Zustandsmaschine erstellen.

Die folgenden Beispielvorlagen zeigen, wie IAM-Richtlinien auf der Grundlage der Ressourcen in Ihrer State-Machine-Definition AWS Step Functions generiert werden. Weitere Informationen finden Sie unter IAM-Richtlinien für integrierte Dienste und Muster der Serviceintegration.

Wir empfehlen, bei der Erstellung von IAM Richtlinien keine Platzhalter in die Richtlinien aufzunehmen. Aus Sicherheitsgründen sollten Sie Ihre Richtlinien so weit wie möglich einschränken. Sie sollten dynamische Richtlinien nur verwenden, wenn bestimmte Eingabeparameter während der Laufzeit nicht bekannt sind.

In diesem Thema

- IAMRichtlinienbeispiele für die Amazon Bedrock Integration mit Step Functions

IAMRichtlinienbeispiele für die Amazon Bedrock Integration mit Step Functions

Im folgenden Abschnitt werden die IAM Berechtigungen beschrieben, die Sie auf der Grundlage der Amazon Bedrock API benötigen, die Sie für ein bestimmtes Foundation- oder Bereitstellungsmodell verwenden. Dieser Abschnitt enthält auch Beispiele für Richtlinien, die vollen Zugriff gewähren.

Denken Sie daran, den *kursiv geschriebenen* Text durch Ihre ressourcenspezifischen Informationen zu ersetzen.

- IAMRichtlinienbeispiel für den Zugriff auf ein bestimmtes Foundation-Modell mit InvokeModel
- IAMRichtlinienbeispiel für den Zugriff auf ein bestimmtes bereitgestelltes Modell mit InvokeModel
- Zu verwendendes Beispiel IAM für eine Richtlinie mit vollständigem Zugriff InvokeModel
- IAMRichtlinienbeispiel für den Zugriff auf ein bestimmtes Foundation-Modell als Basismodell
- IAMRichtlinienbeispiel für den Zugriff auf ein bestimmtes benutzerdefiniertes Modell als Basismodell
- Beispiel IAM für eine Richtlinie mit vollständigem Zugriff zur Verwendung von CreateModelCustomizationJob .sync
- IAMRichtlinienbeispiel für den Zugriff auf ein bestimmtes Foundation-Modell mithilfe von CreateModelCustomizationJob .sync
- IAMRichtlinienbeispiel für den Zugriff auf ein benutzerdefiniertes Modell mithilfe von .sync CreateModelCustomizationJob
- Beispiel IAM für eine vollständige Zugriffsrichtlinie zur Verwendung von .sync CreateModelCustomizationJob

IAMRichtlinienbeispiel für den Zugriff auf ein bestimmtes Foundation-Modell mit InvokeModel

Im Folgenden finden Sie ein IAM Richtlinienbeispiel für eine Zustandsmaschine, die auf ein bestimmtes Foundation-Modell zugreift, das `amazon.titan-text-express-v1` mithilfe der InvokeModelAPI-Aktion benannt wurde.

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

IAMRichtlinienbeispiel für den Zugriff auf ein bestimmtes bereitgestelltes Modell mit InvokeModel

Im Folgenden finden Sie ein IAM Richtlinienbeispiel für eine Zustandsmaschine, die auf ein bestimmtes bereitgestelltes Modell zugreift, das `c2oi931ulksx` mithilfe der InvokeModelAPI-Aktion benannt wurde.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Sid": "InvokeModel1",
      "Action": [
        "bedrock:InvokeModel"
      ],
      "Resource": [
        "arn:aws:bedrock:us-east-2:123456789012:provisioned-model/c2oi931ulksx"
      ]
    }
  ]
}
```

Zu verwendendes Beispiel IAM für eine Richtlinie mit vollständigem Zugriff InvokeModel

Im Folgenden finden Sie ein IAM Richtlinienbeispiel für eine Zustandsmaschine, die vollen Zugriff gewährt, wenn Sie die InvokeModelAPI-Aktion verwenden.

```
{
    "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/*"
            ]
        }
    ]
}
```

IAMRichtlinienbeispiel für den Zugriff auf ein bestimmtes Foundation-Modell als Basismodell

Im Folgenden finden Sie ein IAM Richtlinienbeispiel für eine Zustandsmaschine, um mithilfe der CreateModelCustomizationJobAPI-Aktion auf ein bestimmtes Foundation-Modell zuzugreifen, das `amazon.titan-text-express-v1` als Basismodell bezeichnet wird.

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

```
}
```

IAMRichtlinienbeispiel für den Zugriff auf ein bestimmtes benutzerdefiniertes Modell als Basismodell

Im Folgenden finden Sie ein IAM Richtlinienbeispiel für eine Zustandsmaschine, um mithilfe der CreateModelCustomizationJobAPI-Aktion auf ein bestimmtes benutzerdefiniertes Modell als Basismodell zuzugreifen.

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

Beispiel IAM für eine Richtlinie mit vollständigem Zugriff zur Verwendung von CreateModelCustomizationJob .sync

Im Folgenden finden Sie ein IAM Richtlinienbeispiel für eine Zustandsmaschine, die vollen Zugriff gewährt, wenn Sie die CreateModelCustomizationJobAPI-Aktion verwenden.

```
{
```

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

IAMRichtlinienbeispiel für den Zugriff auf ein bestimmtes Foundation-Modell mithilfe von CreateModelCustomizationJob .sync

Im Folgenden finden Sie ein IAM Richtlinienbeispiel für einen Zustandsmaschine für den Zugriff auf ein bestimmtes Foundation-Modell, das `amazon.titan-text-express-v1` mithilfe der API-Aktion CreateModelCustomizationJob.sync benannt wurde.

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

IAMRichtlinienbeispiel für den Zugriff auf ein benutzerdefiniertes Modell mithilfe von .sync CreateModelCustomizationJob

Im Folgenden finden Sie ein IAM Richtlinienbeispiel für eine Zustandsmaschine für den Zugriff auf ein benutzerdefiniertes Modell mithilfe der CreateModelCustomizationJobAPI-Aktion .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"
            ]
        }
    ]
}
```

Beispiel IAM für eine vollständige Zugriffsrichtlinie zur Verwendung von .sync CreateModelCustomizationJob

Im Folgenden finden Sie ein IAM Richtlinienbeispiel für eine Zustandsmaschine, die vollen Zugriff gewährt, wenn Sie die API-Aktion CreateModelCustomizationJob.sync verwenden.

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

IAM-Richtlinien für AWS CodeBuild

Die folgenden Beispielvorlagen zeigen, wie IAM-Richtlinien auf der Grundlage der Ressourcen in Ihrer State-Machine-Definition AWS Step Functions generiert werden. Weitere Informationen finden Sie unter IAM-Richtlinien für integrierte Dienste und Muster der Serviceintegration.

Ressourcen:

```
{
```

```
    "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`

Statische Ressourcen

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]]"
      ]
    }
```

```
  ]
}
```

Dynamische Ressourcen

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`

Statische Ressourcen

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codebuild:StopBuild"
      ],
      "Resource": [
        "arn:aws:codebuild:[[region]]:[[accountId]]:project/[[projectName]]"
      ]
    }
  ]
}
```

Dynamische Ressourcen

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codebuild:StopBuild"
      ],
```

```
      "Resource": [
        "arn:aws:codebuild:[[region]]:*:project/*"
      ]
    }
  ]
}
```

`BatchDeleteBuilds`

Statische Ressourcen

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codebuild:BatchDeleteBuilds"
      ],
      "Resource": [
        "arn:aws:codebuild:[[region]]:[[accountId]]:project/[[projectName]]"
      ]
    }
  ]
}
```

Dynamische Ressourcen

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codebuild:BatchDeleteBuilds"
      ],
      "Resource": [
        "arn:aws:codebuild:[[region]]:*:project/*"
      ]
    }
  ]
}
```

`BatchGetReports`

Statische Ressourcen

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codebuild:BatchGetReports"
      ],
      "Resource": [
        "arn:aws:codebuild:[[region]]:[[accountId]]:report-group/[[reportName]]"
      ]
    }
  ]
}
```

Dynamische Ressourcen

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codebuild:BatchGetReports"
      ],
      "Resource": [
        "arn:aws:codebuild:[[region]]:*:report-group/*"
      ]
    }
  ]
}
```

`StartBuildBatch`

Statische Ressourcen

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]]"
            ]
        }
```

```
    ]
}
```

Dynamische Ressourcen

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`

Statische Ressourcen

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codebuild:StopBuildBatch"
            ],
            "Resource": [
                "arn:aws:codebuild:[[region]]:[[accountId]]:project/[[projectName]]"
            ]
        }
    ]
}
```

Dynamische Ressourcen

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codebuild:StopBuildBatch"
            ],
```

```
            "Resource": [
                "arn:aws:codebuild:[[region]]:[[accountId]]:project/*"
            ]
        }
    ]
}
```

`RetryBuildBatch`

Statische Ressourcen

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]]"
```

```
            ]
        }
    ]
}
```

Dynamische Ressourcen

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`

Statische Ressourcen

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codebuild:DeleteBuildBatch"
            ],
            "Resource": [
                "arn:aws:codebuild:[[region]]:[[accountId]]:project/[[projectName]]"
            ]
        }
    ]
}
```

Dynamische Ressourcen

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codebuild:DeleteBuildBatch"
            ],
            "Resource": [
                "arn:aws:codebuild:[[region]]:[[accountId]]:project/*"
            ]
        }
    ]
}
```

IAM-Richtlinien für Amazon DynamoDB

Die folgenden Beispielvorlagen zeigen, wie IAM-Richtlinien auf der Grundlage der Ressourcen in Ihrer State-Machine-Definition AWS Step Functions generiert werden. Weitere Informationen finden Sie unter IAM-Richtlinien für integrierte Dienste und Muster der Serviceintegration.

Statische Ressourcen

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "dynamodb:GetItem",
                "dynamodb:PutItem",
                "dynamodb:UpdateItem",
                "dynamodb:DeleteItem"
            ],
            "Resource": [
                "arn:aws:dynamodb:[[region]]:[[accountId]]:table/[[tableName]]"
            ]
        }
    ]
}
```

Dynamische Ressourcen

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "dynamodb:GetItem",
                "dynamodb:PutItem",
                "dynamodb:UpdateItem",
                "dynamodb:DeleteItem"
            ],
            "Resource": "*"
        }
    ]
```

```
}
```

Weitere Informationen zu den IAM-Richtlinien für alle DynamoDB-API-Aktionen finden Sie unter IAM-Richtlinien mit DynamoDB im Amazon DynamoDB DynamoDB-Entwicklerhandbuch. Weitere Informationen zu den IAM-Richtlinien für PartiQL for DynamoDB finden Sie unter IAM-Richtlinien mit PartiQL for DynamoDB im Amazon DynamoDB Developer Guide.

IAM-Richtlinien für Amazon ECS/AWS Fargate

Die folgenden Beispielvorlagen zeigen, wie IAM-Richtlinien auf der Grundlage der Ressourcen in Ihrer State-Machine-Definition AWS Step Functions generiert werden. Weitere Informationen finden Sie unter IAM-Richtlinien für integrierte Dienste und Muster der Serviceintegration.

Da der Wert für erst bekannt `TaskId` ist, wenn die Aufgabe gesendet wird, erstellt Step Functions eine `"Resource": "*"` Richtlinie mit mehr Rechten.

> **Note**
>
> Sie können trotz der `"*"` IAM-Richtlinie nur Amazon Elastic Container Service (Amazon ECS) -Aufgaben beenden, die von Step Functions gestartet wurden.

Run a Job (.sync)

Statische Ressourcen

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

Dynamische Ressourcen

```
{
    "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)

Statische Ressourcen

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:RunTask"
            ],
            "Resource": [
                "arn:aws:ecs:[[region]]:
[[accountId]]:task-definition/[[taskDefinition]]:[[revisionNumber]]"
            ]
        }
    ]
}
```

Dynamische Ressourcen

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:RunTask"
            ],
            "Resource": "*"
        }
    ]
```

```
}
```

Wenn Ihre geplanten Amazon ECS-Aufgaben die Verwendung einer Aufgabenausführungsrolle, einer Aufgabenrolle oder einer Aufgabenrollenüberschreibung erfordern, müssen Sie der CloudWatch Events-IAM-Rolle der aufrufenden Entität, in diesem Fall Step Functions, `iam:PassRole` Berechtigungen für jede Aufgabenausführungsrolle, Aufgabenrolle oder Aufgabenrollenüberschreibung hinzufügen.

IAM-Richtlinien für Amazon EKS

Die folgenden Beispielvorlagen zeigen, wie IAM-Richtlinien auf der Grundlage der Ressourcen in Ihrer State-Machine-Definition AWS Step Functions generiert werden. Weitere Informationen finden Sie unter IAM-Richtlinien für integrierte Dienste und Muster der Serviceintegration.

`CreateCluster`

Ressourcen

```
{
    "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`

Ressourcen

```
{
    "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`

Ressourcen

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks:DeleteCluster",
                "eks:DescribeCluster"
            ],
            "Resource": [
                "arn:aws:eks:sa-east-1:444455556666:cluster/ExampleCluster"
            ]
        }
    ]
}
```

`DeleteNodegroup`

Ressourcen

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

```
        {
            "Effect": "Allow",
            "Action": [
                "eks:DeleteNodegroup",
                "eks:DescribeNodegroup"
            ],
            "Resource": [
                "arn:aws:eks:sa-east-1:444455556666:nodegroup/ExampleCluster/
ExampleNodegroup/*"
            ]
        }
    ]
}
```

Weitere Informationen zur Verwendung von Amazon EKS mit Step Functions finden Sie unterRufen Sie Amazon EKS mit Step Functions auf.

IAM-Richtlinien für Amazon EMR

Die folgenden Beispielvorlagen zeigen, wie IAM-Richtlinien auf der Grundlage der Ressourcen in Ihrer State-Machine-Definition AWS Step Functions generiert werden. Weitere Informationen finden Sie unter IAM-Richtlinien für integrierte Dienste und Muster der Serviceintegration.

`addStep`

Statische Ressourcen

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "elasticmapreduce:AddJobFlowSteps",
        "elasticmapreduce:DescribeStep",
        "elasticmapreduce:CancelSteps"
      ],
      "Resource": [
        "arn:aws:elasticmapreduce:[[region]]:[[accountId]]:cluster/[[clusterId]]"
      ]
    }
  ]
}
```

Dynamische Ressourcen

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "elasticmapreduce:AddJobFlowSteps",
        "elasticmapreduce:DescribeStep",
        "elasticmapreduce:CancelSteps"
      ],
      "Resource": "arn:aws:elasticmapreduce:*:*:cluster/*"
    }
  ]
}
```

`cancelStep`

Statische Ressourcen

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "elasticmapreduce:CancelSteps",
            "Resource": [

 "arn:aws:elasticmapreduce:[[region]]:[[accountId]]:cluster/[[clusterId]]"
            ]
        }
    ]
}
```

Dynamische Ressourcen

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": "elasticmapreduce:CancelSteps",
            "Resource": "arn:aws:elasticmapreduce:*:*:cluster/*"
        }
    ]
}
```

`createCluster`

Statische Ressourcen

```
{
  "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`

Statische Ressourcen

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "elasticmapreduce:SetTerminationProtection",
            "Resource": [
```

```
 "arn:aws:elasticmapreduce:[[region]]:[[accountId]]:cluster/[[clusterId]]"
            ]
        }
    ]
}
```

Dynamische Ressourcen

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "elasticmapreduce:SetTerminationProtection",
            "Resource": "arn:aws:elasticmapreduce:*:*:cluster/*"
        }
    ]
}
```

`modifyInstanceFleetByName`

Statische Ressourcen

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elasticmapreduce:ModifyInstanceFleet",
                "elasticmapreduce:ListInstanceFleets"
            ],
            "Resource": [

 "arn:aws:elasticmapreduce:[[region]]:[[accountId]]:cluster/[[clusterId]]"
            ]
        }
    ]
}
```

Dynamische Ressourcen

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elasticmapreduce:ModifyInstanceFleet",
                "elasticmapreduce:ListInstanceFleets"
            ],
            "Resource": "arn:aws:elasticmapreduce:*:*:cluster/*"
        }
    ]
}
```

`modifyInstanceGroupByName`

Statische Ressourcen

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elasticmapreduce:ModifyInstanceGroups",
                "elasticmapreduce:ListInstanceGroups"
            ],
            "Resource": [

 "arn:aws:elasticmapreduce:[[region]]:[[accountId]]:cluster/[[clusterId]]"
            ]
        }
    ]
}
```

Dynamische Ressourcen

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
```

```
                "elasticmapreduce:ModifyInstanceGroups",
                "elasticmapreduce:ListInstanceGroups"
            ],
            "Resource": "*"
        }
    ]
}
```

`terminateCluster`

Statische Ressourcen

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "elasticmapreduce:TerminateJobFlows",
        "elasticmapreduce:DescribeCluster"
      ],
      "Resource": [
        "arn:aws:elasticmapreduce:[[region]]:[[accountId]]:cluster/[[clusterId]]"
      ]
    }
  ]
}
```

Dynamische Ressourcen

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "elasticmapreduce:TerminateJobFlows",
        "elasticmapreduce:DescribeCluster"
      ],
      "Resource": "arn:aws:elasticmapreduce:*:*:cluster/*"
    }
  ]
}
```

IAM-Richtlinien für Amazon EMR auf EKS

Die folgenden Beispielvorlagen zeigen, wie IAM-Richtlinien auf der Grundlage der Ressourcen in Ihrer State-Machine-Definition AWS Step Functions generiert werden. Weitere Informationen finden Sie unter IAM-Richtlinien für integrierte Dienste und Muster der Serviceintegration.

`CreateVirtualCluster`

Ressourcen

```
{
    "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`

Statische Ressourcen

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]]"
      ]
    }
  ]
}
```

Dynamische Ressourcen

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`

Statische Ressourcen

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]]"
          ]
        }
      }
    }
```

```
  ]
}
```

Dynamische Ressourcen

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

IAM-Richtlinien für Amazon EMR Serverless

Wenn Sie eine Zustandsmaschine mithilfe der Konsole erstellen, Step Functions wird automatisch eine Ausführungsrolle für Ihre Zustandsmaschine mit den geringsten erforderlichen Rechten erstellt. Diese automatisch generierten IAM Rollen gelten für den, AWS-Region in dem Sie den Zustandsmaschine erstellen.

Die folgenden Beispielvorlagen zeigen, wie IAM-Richtlinien auf der Grundlage der Ressourcen in Ihrer State-Machine-Definition AWS Step Functions generiert werden. Weitere Informationen finden Sie unter IAM-Richtlinien für integrierte Dienste und Muster der Serviceintegration.

Wir empfehlen, bei der Erstellung von IAM Richtlinien keine Platzhalter in die Richtlinien aufzunehmen. Aus Sicherheitsgründen sollten Sie Ihre Richtlinien so weit wie möglich einschränken. Sie sollten dynamische Richtlinien nur verwenden, wenn bestimmte Eingabeparameter während der Laufzeit nicht bekannt sind.

Darüber hinaus sollten Administratorbenutzer vorsichtig sein, wenn sie Benutzern, die keine Administratoren sind, Ausführungsrollen für die Ausführung der Zustandsmaschinen zuweisen. Wir empfehlen, dass Sie PassRole-Richtlinien in die Ausführungsrollen aufnehmen, wenn Sie Richtlinien selbst erstellen. Wir empfehlen außerdem, die Kontextschlüssel `aws:SourceARN` und die `aws:SourceAccount` Kontextschlüssel zu den Ausführungsrollen hinzuzufügen.

In diesem Thema

- Beispiele für IAM-Richtlinien für die serverlose EMR-Integration mit Step Functions

Beispiele für IAM-Richtlinien für die serverlose EMR-Integration mit Step Functions

- Beispiel für eine IAM-Richtlinie für CreateApplication
- Beispiel für eine IAM-Richtlinie für StartApplication
- Beispiel für eine IAM-Richtlinie für StopApplication
- Beispiel für eine IAM-Richtlinie für DeleteApplication
- Beispiel für eine IAM-Richtlinie für StartJobRun
- Beispiel für eine IAM-Richtlinie für CancelJobRun

Beispiel für eine IAM-Richtlinie für CreateApplication

Im Folgenden finden Sie ein Beispiel für eine IAM-Richtlinie für eine Zustandsmaschine mit einem CreateApplication Status der Aufgabe Status.

> **Note**
>
> Sie müssen die CreateServiceLinkedRole Berechtigungen in Ihren IAM-Richtlinien bei der Erstellung der allerersten Anwendung in Ihrem Konto angeben. Danach müssen Sie diese Berechtigung nicht mehr hinzufügen. Weitere Informationen CreateServiceLinkedRole dazu finden Sie CreateServiceLinkedRoleunter https://docs.aws.amazon.com/IAM/latest/APIReference/.

Statische und dynamische Ressourcen für die folgenden Richtlinien sind identisch.

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"
                }
            }
        }
    ]
}
```

Beispiel für eine IAM-Richtlinie für StartApplication

Statische Ressourcen

Im Folgenden finden Sie Beispiele für IAM-Richtlinien für statische Ressourcen, wenn Sie eine Zustandsmaschine mit einem StartApplication Status der Aufgabe verwenden.

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]]"
            ]
        }
    ]
}
```

Dynamische Ressourcen

Im Folgenden finden Sie Beispiele für IAM-Richtlinien für dynamische Ressourcen, wenn Sie eine Zustandsmaschine mit einem StartApplication Status der Aufgabe verwenden.

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/*"
            ]
        }
    ]
}
```

Beispiel für eine IAM-Richtlinie für StopApplication

Statische Ressourcen

Im Folgenden finden Sie Beispiele für IAM-Richtlinien für statische Ressourcen, wenn Sie eine Zustandsmaschine mit einem StopApplication Status der Aufgabe verwenden.

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]]"
            ]
        }
    ]
}
```

Dynamische Ressourcen

Im Folgenden finden Sie Beispiele für IAM-Richtlinien für dynamische Ressourcen, wenn Sie eine Zustandsmaschine mit einem StopApplication Status der Aufgabe verwenden.

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/*"
            ]
        }
    ]
}
```

Beispiel für eine IAM-Richtlinie für DeleteApplication

Statische Ressourcen

Im Folgenden finden Sie Beispiele für IAM-Richtlinien für statische Ressourcen, wenn Sie eine Zustandsmaschine mit einem DeleteApplication Status der Aufgabe verwenden.

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]]"
            ]
       }
    ]
```

```
}
```

Dynamische Ressourcen

Im Folgenden finden Sie Beispiele für IAM-Richtlinien für dynamische Ressourcen, wenn Sie eine Zustandsmaschine mit einem DeleteApplication Status der Aufgabe verwenden.

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/*"
            ]
        }
    ]
}
```

Beispiel für eine IAM-Richtlinie für StartJobRun

Statische Ressourcen

Im Folgenden finden Sie Beispiele für IAM-Richtlinien für statische Ressourcen, wenn Sie eine Zustandsmaschine mit einem StartJobRun Status der Aufgabe verwenden.

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"
                }
            }
        }
    ]
}
```

Dynamische Ressourcen

Im Folgenden finden Sie Beispiele für IAM-Richtlinien für dynamische Ressourcen, wenn Sie eine Zustandsmaschine mit einem StartJobRun Status der Aufgabe Status verwenden.

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"
                }
            }
        }
    ]
}
```

Beispiel für eine IAM-Richtlinie für CancelJobRun

Statische Ressourcen

Im Folgenden finden Sie Beispiele für IAM-Richtlinien für statische Ressourcen, wenn Sie eine Zustandsmaschine mit einem CancelJobRun Status der Aufgabe verwenden.

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]]"
            ]
        }
    ]
}
```

Dynamische Ressourcen

Im Folgenden finden Sie Beispiele für IAM-Richtlinien für dynamische Ressourcen, wenn Sie eine Zustandsmaschine mit einem CancelJobRun [Status der Aufgabe] Status verwenden.

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/*"
            ]
        }
    ]
}
```

IAM-Richtlinien für Amazon EventBridge

Die folgenden Beispielvorlagen zeigen, wie IAM-Richtlinien auf der Grundlage der Ressourcen in Ihrer State Machine-Definition AWS Step Functions generiert werden. Weitere Informationen finden Sie unter IAM-Richtlinien für integrierte Dienste und Muster der Serviceintegration.

`PutEvents`

Statische Ressourcen

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "events:PutEvents"
            ],
            "Resource": [
                "arn:aws:events:us-east-1:123456789012:event-bus/stepfunctions-
sampleproject-eventbus"
            ],
            "Effect": "Allow"
        }
    ]
}
```

Dynamische Ressourcen

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

Weitere Hinweise zur Verwendung EventBridge mit Step Functions finden Sie unterRufen Sie EventBridge mit Step Functions auf.

IAM-Richtlinien für AWS Lambda

Die folgenden Beispielvorlagen zeigen, wie IAM-Richtlinien auf der Grundlage der Ressourcen in Ihrer State-Machine-Definition AWS Step Functions generiert werden. Weitere Informationen finden Sie unter IAM-Richtlinien für integrierte Dienste und Muster der Serviceintegration.

AWS Step Functions generiert eine IAM-Richtlinie auf der Grundlage Ihrer State-Machine-Definition. Für eine Zustandsmaschine mit zwei AWS Lambda Aufgabenstatus, die `function1`

und aufrufen`function2`, muss eine Richtlinie mit `lambda:Invoke` Berechtigungen für die beiden Funktionen verwendet werden.

Dies wird im folgenden Beispiel veranschaulicht.

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

IAM-Richtlinien für AWS Elemental MediaConvert

Die folgenden Beispielvorlagen zeigen, wie Sie Ihre IAM-Richtlinien auf der Grundlage der Ressourcen in Ihrer State-Machine-Definition einrichten AWS Step Functions müssen. Sie können die IAM-Konsole verwenden, um fehlende Rollenrichtlinien hinzuzufügen. Weitere Informationen finden Sie unter IAM-Richtlinien für integrierte Dienste und Muster der Serviceintegration.

Da die Zugriffskontrolle auf Ressourcenebene teilweise MediaConvert unterstützt wird, müssen Sie `"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": "*"
        }
```

```
    ]
}
```

IAM-Richtlinien für AWS Glue

Die folgenden Beispielvorlagen zeigen, wie IAM-Richtlinien auf der Grundlage der Ressourcen in Ihrer State-Machine-Definition AWS Step Functions generiert werden. Weitere Informationen finden Sie unter IAM-Richtlinien für integrierte Dienste und Muster der Serviceintegration.

AWS Glue verfügt nicht über eine ressourcenbasierte Steuerung.

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": "*"
        }
    ]
```

```
}
```

IAM-Richtlinien für AWS Glue DataBrew

Die folgenden Beispielvorlagen zeigen, wie IAM-Richtlinien auf der Grundlage der Ressourcen in Ihrer State-Machine-Definition AWS Step Functions generiert werden. Weitere Informationen finden Sie unter IAM-Richtlinien für integrierte Dienste und Muster der Serviceintegration.

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/*"
            ]
        }
    ]
```

```
}
```

IAM-Richtlinien für Amazon SageMaker

Die folgenden Beispielvorlagen zeigen, wie IAM-Richtlinien auf der Grundlage der Ressourcen in Ihrer State Machine-Definition AWS Step Functions generiert werden. Weitere Informationen finden Sie unter IAM-Richtlinien für integrierte Dienste und Muster der Serviceintegration.

> **Note**
>
> *[[roleArn]]*Bezieht sich für diese Beispiele auf den Amazon-Ressourcennamen (ARN) der IAM-Rolle, die für den Zugriff auf Modellartefakte und Docker-Images für die Bereitstellung auf ML-Compute-Instances oder für Batch-Transformationsjobs SageMaker verwendet wird. Weitere Informationen finden Sie unter Amazon SageMaker Roles.

`CreateTrainingJob`

Statische Ressourcen

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"
        }
      }
    }
  ]
}
```

Dynamische Ressourcen

.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 erstellt nicht automatisch eine Richtlinie für den Fall`CreateTransformJob`, dass Sie eine Zustandsmaschine erstellen, die sich in integriert SageMaker. Sie müssen der erstellten Rolle eine Inline-Richtlinie hinzufügen, die auf einem der folgenden IAM-Beispiele basiert.

Statische Ressourcen

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"
        }
      }
    }
  ]
}
```

Dynamische Ressourcen

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"
        }
      }
    }
  ]
}
```

IAM-Richtlinien für Amazon SNS

Die folgenden Beispielvorlagen zeigen, wie IAM-Richtlinien auf der Grundlage der Ressourcen in Ihrer State-Machine-Definition AWS Step Functions generiert werden. Weitere Informationen finden Sie unter IAM-Richtlinien für integrierte Dienste und Muster der Serviceintegration.

Statische Ressourcen

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sns:Publish"
            ],
            "Resource": [
                "arn:aws:sns:[[region]]:[[accountId]]:[[topicName]]"
            ]
        }
    ]
}
```

Ressourcen, die auf einem Pfad basieren oder in *TargetArn* oder *PhoneNumber* veröffentlichen

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sns:Publish"
            ],
            "Resource": "*"
        }
    ]
}
```

IAM-Richtlinien für Amazon SQS

Die folgenden Beispielvorlagen zeigen, wie IAM-Richtlinien auf der Grundlage der Ressourcen in Ihrer State-Machine-Definition AWS Step Functions generiert werden. Weitere Informationen finden Sie unter IAM-Richtlinien für integrierte Dienste und Muster der Serviceintegration.

Statische Ressourcen

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sqs:SendMessage"
            ],
            "Resource": [
                "arn:aws:sqs:[[region]]:[[accountId]]:[[queueName]]"
            ]
        }
    ]
}
```

Dynamische Ressourcen

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sqs:SendMessage"
            ],
            "Resource": "*"
        }
    ]
}
```

IAM-Richtlinien für AWS Step Functions

Verwenden Sie für eine Zustandsmaschine, die die Ausführung eines einzelnen verschachtelten Workflows erfordert`StartExecution`, eine IAM-Richtlinie, die die Berechtigungen auf diese Zustandsmaschine beschränkt.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "states:StartExecution"
            ],
            "Resource": [

 "arn:aws:states:[[region]]:[[accountId]]:stateMachine:[[stateMachineName]]"
            ]
        }
    ]
}
```

Weitere Informationen finden Sie hier:

- Verwendung AWS Step Functions mit anderen Diensten
- Parameter an eine Service-API übergeben
- Managen Sie AWS Step Functions Ausführungen als integrierten Service

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]]"
```

```
            ]
        }
    ]
}
```

Weitere Informationen zu verschachtelten Workflow-Ausführungen finden Sie unter Starten von Workflow-Ausführungen über einen Aufgabenstatus.

IAM-Richtlinien für AWS X-Ray

Die folgenden Beispielvorlagen zeigen, wie IAM-Richtlinien auf der Grundlage der Ressourcen in Ihrer State-Machine-Definition AWS Step Functions generiert werden. Weitere Informationen finden Sie unter IAM-Richtlinien für integrierte Dienste und Muster der Serviceintegration.

Um X-Ray Tracing zu aktivieren, benötigen Sie eine IAM-Richtlinie mit entsprechenden Berechtigungen, um die Ablaufverfolgung zu ermöglichen. Wenn Ihre Zustandsmaschine andere integrierte Dienste verwendet, benötigen Sie möglicherweise zusätzliche IAM-Richtlinien. Sehen Sie sich die IAM-Richtlinien für Ihre spezifischen Serviceintegrationen an.

Wenn Sie eine Zustandsmaschine mit aktiviertem X-Ray-Tracing erstellen, wird automatisch eine IAM-Richtlinie erstellt.

> **Note**
>
> Wenn Sie X-Ray Tracing für einen vorhandenen State Machine aktivieren, müssen Sie sicherstellen, dass Sie eine Richtlinie mit ausreichenden Berechtigungen hinzufügen, um X-Ray Traces zu aktivieren.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "xray:PutTraceSegments",
                "xray:PutTelemetryRecords",
                "xray:GetSamplingRules",
                "xray:GetSamplingTargets"
            ],
```

```
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Weitere Informationen zur Verwendung von X-Ray mit Step Functions finden Sie unterAWS X-Ray und Step Functions.

Aktivitäten oder keine Aufgaben

Verwenden Sie für eine Zustandsmaschine, die nur `Activity` Aufgaben oder gar keine Aufgaben hat, eine IAM-Richtlinie, die den Zugriff auf alle Aktionen und Ressourcen verweigert.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "*",
            "Resource": "*"
        }
    ]
}
```

Weitere Hinweise zur Verwendung von `Activity` -Aufgaben finden Sie unter Aktivitäten.

IAM-Richtlinien für die Verwendung des Distributed Map-Status

Wenn Sie Workflows mit der Step Functions-Konsole erstellen, kann Step Functions automatisch IAM-Richtlinien auf der Grundlage der Ressourcen in Ihrer Workflow-Definition generieren. Diese Richtlinien beinhalten die geringsten Rechte, die erforderlich sind, damit die Zustandsmaschinen-Rolle die `StartExecution` API-Aktion für den Status Distributed Map aufrufen kann. Diese Richtlinien beinhalten auch die Step Functions mit den geringsten Rechten, die für den Zugriff auf AWS Ressourcen wie Amazon S3 S3-Buckets und -Objekte sowie Lambda-Funktionen erforderlich sind. Wir empfehlen dringend, dass Sie nur die Berechtigungen in Ihre IAM-Richtlinien aufnehmen, die erforderlich sind. Wenn Ihr Workflow beispielsweise einen `Map` Status im Modus „Verteilt“ umfasst, beschränken Sie Ihre Richtlinien auf den spezifischen Amazon S3 S3-Bucket und -Ordner, der Ihren Datensatz enthält.

> **Important**
>
> Wenn Sie einen Amazon S3 S3-Bucket und ein Objekt oder ein Präfix mit einem Referenzpfad zu einem vorhandenen Schlüssel-Wert-Paar in Ihrer Distributed Map-Statuseingabe angeben, stellen Sie sicher, dass Sie die IAM-Richtlinien für Ihren Workflow aktualisieren. Beschränken Sie die Richtlinien auf die Bucket- und Objektnamen, zu denen der Pfad zur Laufzeit aufgelöst wird.

In diesem Thema:

Beispiel für eine IAM-Richtlinie für die Ausführung eines Distributed-Map-Status

Wenn Sie einen Distributed Map-Status in Ihre Workflows aufnehmen, benötigt Step Functions die entsprechenden Berechtigungen, damit die Zustandsmaschinenrolle die `StartExecution` API-Aktion für den Distributed-Map-Status aufrufen kann.

Das folgende Beispiel für eine IAM-Richtlinie gewährt Ihrer State-Machine-Rolle die geringsten Rechte, die für die Ausführung des Status Distributed Map erforderlich sind.

> **Note**
>
> Stellen Sie sicher, dass Sie den Status *`stateMachineName`* durch den Namen des Zustandsmaschinen ersetzen, in dem Sie den Status Distributed Map verwenden. z. B. `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:*"
    }
  ]
}
```

Beispiel für eine IAM-Richtlinie für redriving eine Distributed Map

Sie können erfolglose untergeordnete Workflow-Ausführungen in einem Map Run von redrivingIhrem übergeordneten Workflow neu starten. Ein redriven übergeordneter Workflow mit redrives allen erfolglosen Status, einschließlich Distributed Map. Stellen Sie sicher, dass Ihre Ausführungsrolle über die geringsten Rechte verfügt, die erforderlich sind, um die `RedriveExecution` API-Aktion für den übergeordneten Workflow aufrufen zu können.

Das folgende Beispiel für eine IAM-Richtlinie gewährt Ihrer State-Machine-Rolle die geringsten Rechte, die für redriving einen Distributed-Map-Status erforderlich sind.

> **Note**
>
> Stellen Sie sicher, dass Sie den Status *`stateMachineName`* durch den Namen des Zustandsmaschinen ersetzen, in dem Sie den Status Distributed Map verwenden. z. B. `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:*"
    }
  ]
}
```

Beispiele für IAM-Richtlinien zum Lesen von Daten aus Amazon S3 S3-Datensätzen

Die folgenden Beispiele für IAM-Richtlinien gewähren die geringsten Rechte, die für den Zugriff auf Ihre Amazon S3 S3-Datensätze mithilfe der ListObjectsV2 - und GetObjectAPI-Aktionen erforderlich sind.

Example IAM-Richtlinie für Amazon S3 S3-Objekt als Datensatz

Das folgende Beispiel zeigt eine IAM-Richtlinie, die die geringsten Rechte für den Zugriff auf die Objekte gewährt, die *processImages* in einem Amazon S3 S3-Bucket mit dem Namen *myBucket* organisiert sind.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::myBucket"
            ],
            "Condition": {
                "StringLike": {
                    "s3:prefix": [
                        "processImages"
                    ]
                }
            }
        }
    ]
}
```

Example IAM-Richtlinie für eine CSV-Datei als Datensatz

Das folgende Beispiel zeigt eine IAM-Richtlinie, die die geringsten Zugriffsrechte auf eine CSV-Datei mit dem Namen gewährt. *ratings.csv*

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::myBucket/csvDataset/ratings.csv"
            ]
        }
    ]
}
```

Example IAM-Richtlinie für ein Amazon S3 S3-Inventar als Datensatz

Das folgende Beispiel zeigt eine IAM-Richtlinie, die die geringsten Rechte für den Zugriff auf einen Amazon S3 S3-Inventarbericht gewährt.

```
{
    "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/*"
            ]
        }
    ]
}
```

Beispiel für eine IAM-Richtlinie zum Schreiben von Daten in einen Amazon S3 S3-Bucket

Das folgende Beispiel für eine IAM-Richtlinie gewährt die geringsten Rechte, die erforderlich sind, um die Ergebnisse der Workflow-Ausführung Ihres untergeordneten Workflows mithilfe der API-Aktion in einen Ordner mit dem Namen *csvJobs* in einem Amazon S3 S3-Bucket zu schreiben. PutObject

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListMultipartUploadParts",
                "s3:AbortMultipartUpload"
            ],
            "Resource": [
                "arn:aws:s3:::resultBucket/csvJobs/*"
            ]
        }
    ]
}
```

IAM-Berechtigungen für AWS KMS key verschlüsselten Amazon S3 S3-Bucket

Distributed Map State verwendet mehrteilige Uploads, um die Ergebnisse der untergeordneten Workflow-Ausführung in einen Amazon S3 S3-Bucket zu schreiben. Wenn der Bucket mit einem AWS Key Management Service (AWS KMS) Schlüssel verschlüsselt ist, müssen Sie auch Berechtigungen in Ihre IAM Richtlinie aufnehmen, um die `kms:GenerateDataKey` Aktionen `kms:Decrypt``kms:Encrypt`, und für den Schlüssel auszuführen. Diese Berechtigungen sind erforderlich, da Amazon S3 Daten aus den verschlüsselten Teilen der Datei entschlüsseln und lesen muss, bevor es den Multipart-Upload vornehmen kann.

Das folgende Beispiel für eine IAM-Richtlinie erteilt Berechtigungen für die `kms:GenerateDataKey` Aktionen `kms:Decrypt``kms:Encrypt`, und für den Schlüssel, der zur Verschlüsselung Ihres Amazon S3 S3-Buckets verwendet wurde.

```
{
```

```
  "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"
    ]
  }
}
```

Weitere Informationen finden Sie unter Uploading a large file to Amazon S3 with encryption using an AWS KMS key CMK (Hochladen einer großen Datei zu Amazon S3 mit Verschlüsselung über einen KMS-CMK) im AWS -Wissenscenter.

Wenn Ihr IAM-Benutzer oder Ihre IAM-Rolle der gleiche ist AWS-Konto wie derKMS key, dann müssen Sie über diese Berechtigungen für die Schlüsselrichtlinie verfügen. Wenn Ihr IAM-Benutzer oder Ihre IAM-Rolle zu einem anderen Konto als dem gehörtKMS key, benötigen Sie die entsprechenden Berechtigungen sowohl für die Schlüsselrichtlinie als auch für Ihren IAM-Benutzer oder Ihre IAM-Rolle.

Tagbasierte Richtlinien

Step Functions unterstützt Richtlinien, die auf Tags basieren. Sie könnten beispielsweise den Zugriff auf alle Step Functions-Ressourcen einschränken, die ein Tag mit dem Schlüssel `environment` und dem Wert enthaltenproduction.

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

Diese Richtlinie setzt die Möglichkeit zum Löschen von Zustandsautomaten oder Aktivitäten auf `Deny` (Verweigern), stoppt Ausführungen und fügt allen Ressourcen, die als `environment/production` markiert wurden, neue Tags hinzu bzw. löscht diese.

Bei der Tag-basierten Autorisierung erben die Ressourcen zur Ausführung von Zustandsmaschinen, wie im folgenden Beispiel gezeigt, die einem Zustandsmaschine zugewiesenen Tags.

```
arn:<partition>:states:<Region>:<account-id>:execution:<StateMachineName>:<ExecutionId>
```

Wenn Sie andere APIs aufrufen DescribeExecution, in denen Sie die Ausführungsressource ARN angeben, verwendet Step Functions Tags, die der Zustandsmaschine zugeordnet sind, um die Anfrage anzunehmen oder abzulehnen, während die Tag-basierte Autorisierung durchgeführt wird. Auf diese Weise können Sie den Zugriff auf Zustandsmaschinenausführungen auf Zustandsmaschinenebene zulassen oder verweigern.

Weitere Informationen zum Tagging finden Sie hier:

- Funktionen zum Taggen in Step
- Zugriffssteuerung mit IAM-Tags

Problembehandlung bei AWS Step Functions Identität und Zugriff

Verwenden Sie die folgenden Informationen, um häufig auftretende Probleme zu diagnostizieren und zu beheben, die bei der Arbeit mit Step Functions und IAM auftreten können.

Themen

- Ich bin nicht berechtigt, eine Aktion in Step Functions durchzuführen
- Ich bin nicht berechtigt, iam auszuführen: PassRole
- Ich möchte Personen außerhalb von mir den Zugriff auf meine AWS-Konto Step Functions Functions-Ressourcen ermöglichen

Ich bin nicht berechtigt, eine Aktion in Step Functions durchzuführen

Wenn Sie die Fehlermeldung erhalten, dass Sie nicht zum Durchführen einer Aktion autorisiert sind, müssen Ihre Richtlinien aktualisiert werden, um die Aktion durchführen zu können.

Der folgende Beispielfehler tritt auf, wenn der `mateojackson`-Benutzer versucht, die Konsole zum Anzeigen von Details zu einer fiktiven *`my-example-widget`*-Ressource zu verwenden, jedoch nicht über `states:`*`GetWidget`*-Berechtigungen verfügt.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 states:GetWidget on resource: my-example-widget
```

In diesem Fall muss die Mateo-Richtlinie aktualisiert werden, damit er mit der `states:`*`GetWidget`*-Aktion auf die *`my-example-widget`*-Ressource zugreifen kann.

Wenn Sie Hilfe benötigen, wenden Sie sich an Ihren AWS Administrator. Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

Ich bin nicht berechtigt, iam auszuführen: PassRole

Wenn Sie eine Fehlermeldung erhalten, dass Sie nicht berechtigt sind, die `iam:PassRole` Aktion auszuführen, müssen Ihre Richtlinien aktualisiert werden, damit Sie eine Rolle an Step Functions übergeben können.

Einige AWS-Services ermöglichen es Ihnen, eine bestehende Rolle an diesen Dienst zu übergeben, anstatt eine neue Servicerolle oder eine dienstverknüpfte Rolle zu erstellen. Hierzu benötigen Sie Berechtigungen für die Übergabe der Rolle an den Dienst.

Der folgende Beispielfehler tritt auf, wenn ein IAM-Benutzer mit dem Namen `marymajor` versucht, die Konsole zu verwenden, um eine Aktion in Step Functions auszuführen. Die Aktion erfordert jedoch, dass der Service über Berechtigungen verfügt, die durch eine Servicerolle gewährt werden. Mary besitzt keine Berechtigungen für die Übergabe der Rolle an den Dienst.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In diesem Fall müssen die Richtlinien von Mary aktualisiert werden, um die Aktion `iam:PassRole` ausführen zu können.

Wenn Sie Hilfe benötigen, wenden Sie sich an Ihren AWS Administrator. Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

Ich möchte Personen außerhalb von mir den Zugriff auf meine AWS-Konto Step Functions Functions-Ressourcen ermöglichen

Sie können eine Rolle erstellen, die Benutzer in anderen Konten oder Personen außerhalb Ihrer Organisation für den Zugriff auf Ihre Ressourcen verwenden können. Sie können festlegen, wem die Übernahme der Rolle anvertraut wird. Im Fall von Diensten, die ressourcenbasierte Richtlinien oder Zugriffskontrolllisten (Access Control Lists, ACLs) verwenden, können Sie diese Richtlinien verwenden, um Personen Zugriff auf Ihre Ressourcen zu gewähren.

Weitere Informationen dazu finden Sie hier:

- Informationen darüber, ob Step Functions diese Funktionen unterstützt, finden Sie unterWie AWS Step Functions funktioniert mit IAM.
- Informationen dazu, wie Sie Zugriff auf Ihre Ressourcen gewähren können, AWS-Konten die Ihnen gehören, finden Sie im IAM-Benutzerhandbuch unter Gewähren von Zugriff für einen IAM-Benutzer in einem anderen AWS-Konto , den Sie besitzen.
- Informationen dazu, wie Sie Dritten Zugriff auf Ihre Ressourcen gewähren können AWS-Konten, finden Sie AWS-Konten im IAM-Benutzerhandbuch unter Gewähren des Zugriffs für Dritte.
- Informationen dazu, wie Sie über einen Identitätsverbund Zugriff gewähren, finden Sie unter Gewähren von Zugriff für extern authentifizierte Benutzer (Identitätsverbund) im IAM-Benutzerhandbuch.
- Informationen zum Unterschied zwischen der Verwendung von Rollen und ressourcenbasierten Richtlinien für den kontoübergreifenden Zugriff finden Sie im IAM-Benutzerhandbuch unter Kontenübergreifender Ressourcenzugriff in IAM.

Protokollieren und Überwachen

Informationen zur Anmeldung und Überwachung finden Sie im AWS Step Functions Abschnitt. Protokollierung und Überwachung

Konformitätsprüfung für AWS Step Functions

Externe Prüfer bewerten die Sicherheit und Einhaltung von Vorschriften im AWS Step Functions Rahmen mehrerer AWS Compliance-Programme. Hierzu zählen unter anderem SOC, PCI, FedRAMP und HIPAA.

Eine Liste der AWS Services im Rahmen bestimmter Compliance-Programme finden Sie unter AWS Services im Umfang nach Compliance-Programmen AWS . Allgemeine Informationen finden Sie unter AWS Compliance-Programme AWS .

Sie können Prüfberichte von Drittanbietern unter herunterladen AWS Artifact. Weitere Informationen finden Sie unter Berichte herunterladen unter .

Ihre Compliance-Verantwortung bei der Nutzung von Step Functions hängt von der Sensibilität Ihrer Daten, den Compliance-Zielen Ihres Unternehmens und den geltenden Gesetzen und Vorschriften ab. AWS stellt die folgenden Ressourcen zur Verfügung, die Sie bei der Einhaltung von Vorschriften unterstützen:

- Schnellstartanleitungen zu Sicherheit und Compliance Schnellstartanleitungen zu — In diesen Bereitstellungshandbüchern werden architektonische Überlegungen erörtert und Schritte für die Implementierung von sicherheits- und Compliance-orientierten Basisumgebungen beschrieben. AWS
- Architecting for HIPAA Security and Compliance on Amazon Web Services — In diesem Whitepaper wird beschrieben, wie Unternehmen HIPAA-konforme Anwendungen erstellen AWS können.
- AWS Ressourcen zur Einhaltung von — Diese Sammlung von Arbeitsmappen und Leitfäden kann auf Ihre Branche und Ihren Standort zutreffen.
- Bewertung von Ressourcen anhand von Regeln im AWS Config Entwicklerhandbuch — Der AWS Config Service bewertet, wie gut Ihre Ressourcenkonfigurationen den internen Praktiken, Branchenrichtlinien und Vorschriften entsprechen.
- AWS Security Hub— Dieser AWS Service bietet einen umfassenden Überblick über Ihren Sicherheitsstatus, sodass Sie überprüfen können AWS , ob Sie die Sicherheitsstandards und Best Practices der Branche einhalten.

Resilienz in AWS Step Functions

Die AWS globale Infrastruktur basiert auf AWS Regionen und Availability Zones. AWS Regionen bieten mehrere physisch getrennte und isolierte Availability Zones, die über Netzwerke mit niedriger Latenz, hohem Durchsatz und hoher Redundanz miteinander verbunden sind. Mithilfe von Availability Zones können Sie Anwendungen und Datenbanken erstellen und ausführen, die automatisch Failover zwischen Zonen ausführen, ohne dass es zu Unterbrechungen kommt. Availability Zones sind besser verfügbar, fehlertoleranter und skalierbarer als herkömmliche Infrastrukturen mit einem oder mehreren Rechenzentren.

Weitere Informationen zu AWS Regionen und Availability Zones finden Sie unter AWS Globale Infrastruktur.

Neben der AWS globalen Infrastruktur bietet Step Functions mehrere Funktionen, um Ihre Anforderungen an Datenstabilität und Backup zu erfüllen.

Sicherheit der Infrastruktur in AWS Step Functions

Als verwalteter Dienst AWS Step Functions wird es durch AWS globale Netzwerksicherheit geschützt. Informationen zu AWS Sicherheitsdiensten und zum AWS Schutz der Infrastruktur finden Sie unter AWS Cloud-Sicherheit. Informationen zum Entwerfen Ihrer AWS Umgebung unter Verwendung der bewährten Methoden für die Infrastruktursicherheit finden Sie unter Infrastructure Protection in Security Pillar AWS Well-Architected Framework.

Sie verwenden AWS veröffentlichte API-Aufrufe für den Zugriff Step Functions über das Netzwerk. Kunden müssen Folgendes unterstützen:

- Transport Layer Security (TLS). Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Verschlüsselungs-Suiten mit Perfect Forward Secrecy (PFS) wie DHE (Ephemeral Diffie-Hellman) oder ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Die meisten modernen Systeme wie Java 7 und höher unterstützen diese Modi.

Außerdem müssen Anforderungen mit einer Zugriffsschlüssel-ID und einem geheimen Zugriffsschlüssel signiert sein, der einem IAM-Prinzipal zugeordnet ist. Alternativ können Sie mit AWS Security Token Service (AWS STS) temporäre Sicherheitsanmeldeinformationen erstellen, um die Anforderungen zu signieren.

Sie können die AWS API-Operationen von jedem Netzwerkstandort aus aufrufen, unterstützt jedoch Step Functions keine ressourcenbasierten Zugriffsrichtlinien, die Einschränkungen auf der Quell-IP-Adresse beinhalten können. Sie können auch Step Functions-Richtlinien verwenden, um den Zugriff von bestimmten Amazon Virtual Private Cloud (Amazon VPC)-Endpunkten oder bestimmten VPCs aus zu steuern. Dadurch wird der Netzwerkzugriff auf eine bestimmte Step Functions Ressource effektiv nur von der spezifischen VPC innerhalb des AWS Netzwerks isoliert.

Konfiguration und Schwachstellenanalyse in AWS Step Functions

Konfiguration und IT-Kontrollen liegen in der gemeinsamen Verantwortung AWS von Ihnen, unserem Kunden. Weitere Informationen finden Sie im Modell der AWS gemeinsamen Verantwortung.

Migrieren von Workloads von AWS Data Pipeline zu Step Functions

AWS hat den AWS Data Pipeline Service 2012 eingeführt. Zu diesem Zeitpunkt wollten Kunden einen Service, der es ihnen ermöglicht, eine Vielzahl von Rechenoptionen zu verwenden, um Daten zwischen verschiedenen Datenquellen zu verschieben. Da sich die Datenübertragung im Laufe der Zeit ändern muss, sollten Sie auch die Lösungen für diese Anforderungen haben. Sie haben jetzt die Möglichkeit, die Lösung auszuwählen, die Ihren Geschäftsanforderungen am ehesten entspricht. Sie können z. B. Folgendes machen:

- Verwenden Sie Step Functions, um Workflows zwischen mehreren zu orchestrieren AWS-Services.
- Verwenden Sie Amazon Managed Workflows for Apache Airflow (Amazon MWAA), um die Workflow-Orchestrierung für Apache Airflow zu verwalten.
- Verwenden Sie AWS Glue , um Apache-Spark-Anwendungen auszuführen und zu orchestrieren.

Sie können typische Anwendungsfälle von entweder AWS Data Pipeline zu AWS Glue, Step Functions oder Amazon MWAA migrieren. Die von Ihnen gewählte Option hängt von Ihrem aktuellen Workload auf ab AWS Data Pipeline. In diesem Thema wird erläutert, wie Sie von AWS Data Pipeline zu Step Functions migrieren.

Themen

Migrieren von Workloads von AWS Data Pipeline

Step Functions ist ein Serverless-Orchestrierungsservice, bei dem Sie Workflows für geschäftskritische Anwendungen erstellen. Mit Step Functions Workflow Studio können Sie Workflows erstellen und sie in mehr als 11 000 API-Aktionen aus über 250 integrieren AWS-Services. Dazu gehören AWS-Services wie AWS Lambda, Amazon EMR und Amazon DynamoDB . Sie können Step Functions auch verwenden, um Datenverarbeitungspipelines zu orchestrieren, Fehler

zu behandeln und mit Drosselungslimits für die zugrunde liegenden zu arbeiten AWS-Services. Sie können Workflows erstellen, die Machine-Learning-Modelle verarbeiten und veröffentlichen, Microservices orchestrieren und ETL-Workflows (Extract, Transform, Load) mit behandeln AWS Glue. Sie können auch lang laufende, automatisierte Workflows für Anwendungen erstellen, die menschliche Interaktion erfordern.

Step Functions ist ein vollständig verwalteter Service, der von bereitgestellt wird AWS. Das bedeutet, dass AWS Aufgaben wie die Wartung der Infrastruktur, das Patchen von Workern und die Verwaltung von Betriebssystemversionsaktualisierungen für Sie verwaltet.

Wenn Ihr Anwendungsfall den folgenden Bedingungen entspricht, empfehlen wir Ihnen, von zu Step Functions AWS Data Pipeline zu migrieren:

- Sie bevorzugen einen serverlosen, hochverfügbaren Workflow-Orchestrierungsservice.
- Sie benötigen eine Lösung, die mit der Granularität einer einzelnen Aufgabenausführung berechnet.
- Ihre Workloads beinhalten die Orchestrierung von Aufgaben für mehrere andere AWS-Services, wie Amazon EMR AWS Glue, Lambda oder DynamoDB .
- Sie benötigen eine Low-Code-Lösung mit einem drag-and-drop visuellen Designer für die Workflow-Erstellung. Diese Lösung sollte nicht erfordern, unbekannte, komplexe Programmierkonzepte zu erlernen.
- Sie benötigen einen Service, der in über 250 integriert werden kann AWS-Services , die mehr als 11.000 API-Aktionen abdecken. Dieser Service muss auch in benutzerdefinierte Services und Aktivitäten außerhalb von integriert werden AWS.

Konzeptzuordnung zwischen Step Functions und AWS Data Pipeline

AWS Data Pipeline und Step Functions teilen sich einige gängige Konzepte. Um beispielsweise Ihre Workflows zu definieren, verwenden Sie das JSON-Format sowohl in AWS Data Pipeline als auch in Step Functions. In Step Functions verwenden Sie Amazon States Language, eine JSON-basierte, strukturierte Sprache. Sie verwenden Amazon States Language (ASL), um Ihre Workflows zu definieren und zwischen der Text- und der visuellen Darstellung Ihres Workflows zu wechseln. Dieses JSON-basierte Format vereinfacht das Speichern Ihrer Workflows in einem Quellcodeverwaltungs-Tool. Es hilft Ihnen auch, mehrere Versionen Ihrer Workflows zu verwalten, ihren Zugriff zu steuern oder ihre Orchestrierung mit CI/CD-Methoden zu automatisieren.

In der folgenden Tabelle wird die Zuordnung zwischen den wichtigsten Konzepten beschrieben, die in beiden Services verwendet werden. In der Spalte Konzepte für Datenpipelines auf der linken Seite werden die Konzepte in aufgeführt AWS Data Pipeline, während in der Spalte Konzepte für Step Functions auf der rechten Seite die entsprechenden Konzepte in Step Functions aufgeführt sind.

Konzepte für Datenpipelines	Konzepte für Step Functions
Pipelines	Workflows
Pipeline-Definition	Amazon States Language (ASL)
Aktivitäten	Zustände und Status der Aufgabe
Instances	Ausführungen
Attempts	Catcher und Wiederholungen
Pipeline-Zeitplan	• Ausführungen mit Amazon EventBridge Scheduler • Über EventBridge Pipes ausgelöste Ereignisse
Pipeline-Ausdrücke und -Funktionen	• Intrinsische Funktionen • Lambda-Funktionen mit Serviceintegration

Step-Functions-Beispielprojekte

Eine Einführung in Step Functions finden Sie im folgenden Video:

Erste Schritte mit AWS Step Functions für die Service-Orchestrierung

In der folgenden Liste werden einige Beispielprojekte beschrieben, die die häufigsten AWS Data Pipeline Anwendungsfälle mit Step Functions implementieren. Sie können diese Beispielprojekte als Referenz für die Migration von AWS Data Pipeline zu Step Functions verwenden. Sie können sie auch als Boilerplate verwenden, um Ihre eigenen Workflows zu erstellen und je nach Anwendungsfall in das unterstützte AWS-Services zu integrieren.

- Einen Amazon EMR-Job verwalten

- Ausführen eines Datenverarbeitungsauftrags auf Amazon EMR Serverless
- Ausführen von Hive/Pig/Hadoop-Aufträgen
- Große Datensätze abfragen (Amazon Athena, Amazon S3 AWS Glue, Amazon SNS)
- Führen Sie ETL/ELT-Workflows mit Amazon Redshift aus
- Orchestrieren von AWS Glue Crawlern
- Ausführen eines Shell-Skripts mit Step Functions

Weitere Informationen zu Step Functions finden Sie in den folgenden Themen und Ressourcen:

- Tutorials für Step Functions
- Beispielprojekte für Step Functions
- Der AWS Step Functions -Workshop

Preisvergleich

AWS Data Pipeline wird nach Anzahl der Pipelines und deren Nutzungsgrad berechnet. Aktivitäten, die mehr als einmal pro Tag (Hochfrequenz) ausgeführt werden, werden zu einem Preis von 1 USD pro Monat und Aktivität berechnet. Aktivitäten, die einmal pro Tag oder weniger (niedrige Häufigkeit) ausgeführt werden, werden zu einem Preis von 0,60 USD pro Monat und Aktivität angeboten. Inaktive Pipelines werden zu einem Preis von 1 USD pro Pipeline berechnet. Weitere Informationen zu Preisen finden Sie auf der Seite -AWS Data Pipeline Preise.

Step Functions hat zwei Arten von Workflows: Standard und Express. Jeder Workflow-Typ hat ein anderes Preismodell. Dieser Vergleich basiert auf dem Standard-Workflow, da er am besten mit gängigen Anwendungsfällen von übereinstimmt AWS Data Pipeline. Standard-Workflows werden für 1000 Statusübergänge zu einem Preis von 0,025 USD angeboten. Für inaktive Zustandsautomaten fallen keine Kosten an. Sie zahlen nur für das, was Sie tatsächlich nutzen. Weitere Informationen zu Preisen finden Sie auf der Seite -AWS Step Functions Preise.

Fehlerbehebung

Wenn Sie bei der Arbeit mit Step Functions auf Schwierigkeiten stoßen, nutzen Sie die folgenden Ressourcen zur Fehlerbehebung.

Themen

Allgemeine Problembehebung

Ich kann keine State Machine erstellen.

Die der State-Maschine zugeordnete IAM-Rolle verfügt möglicherweise nicht über ausreichende Berechtigungen. Prüfen Sie die Berechtigungen der IAM-Rolle, unter anderem für AWS Service-Integrationsaufgaben, X-Ray und CloudWatch Protokollierung. Für `.sync` Aufgabenstatus sind zusätzliche Berechtigungen erforderlich.

Ich kann a nicht verwenden, JsonPath um auf die Ausgabe der vorherigen Aufgabe zu verweisen.

Für a JsonPath muss ein JSON-Schlüssel mit enden `.$`. Das bedeutet, dass a JsonPath nur in einem Schlüssel-Wert-Paar verwendet werden kann. Wenn Sie JsonPath andere Stellen verwenden möchten, z. B. ein Array, können Sie intrinsische Funktionen verwenden. Sie könnten beispielsweise etwas Ähnliches wie das Folgende verwenden:

Aufgabe A: Ausgang:

```
{
    "sample": "test"
}
```

Aufgabe B:

```
{
    "JsonPathSample.$": "$.sample"
}
```

> **Tip**
>
> Verwenden Sie den Datenflusssimulator in der Step Functions-Konsole, um die JSON-Pfadsyntax zu testen, besser zu verstehen, wie Daten innerhalb eines Zustands manipuliert werden, und um zu sehen, wie Daten zwischen Zuständen weitergegeben werden.

Die Zustandsübergänge verzögerten sich.

Bei Standard-Workflows ist die Anzahl der Zustandsübergänge begrenzt. Wenn Sie das Limit für den Statusübergang überschreiten, verzögert Step Functions die Zustandsübergänge, bis der Bucket für das Kontingent gefüllt ist. Die Drosselung der Übergangsgrenzen in den Bundesstaaten kann überwacht werden, indem Sie die `ExecutionThrottled` Metrik im Ausführungsmetriken Abschnitt der CloudWatch Metrikenseite überprüfen.

Wenn ich neue Standard-Workflow-Ausführungen starte, schlagen sie mit dem **`ExecutionLimitExceeded`** Fehler fehl.

Step Functions hat ein Limit von jeweils AWS-Konto 1.000.000 offenen Ausführungen. AWS-Region Wenn Sie dieses Limit überschreiten, gibt Step Functions einen `ExecutionLimitExceeded` Fehler aus. Dieses Limit gilt nicht für Express Workflows. Sie können die folgende CloudWatchMetrik-Mathematik im CloudWatchAmazon-Benutzerhandbuch verwenden, um die ungefähre Anzahl der offenen Ausführungen zu ermitteln:`ExecutionsStarted - (ExecutionsSucceeded + ExecutionsTimedOut + ExecutionsFailed + ExecutionsAborted)`.

Ein Ausfall in einem Zweig in einem parallelen Zustand führt dazu, dass die gesamte Ausführung fehlschlägt.

Dies ist ein erwartetes Verhalten. Um zu vermeiden, dass bei Verwendung eines parallelen Zustands Fehler auftreten, konfigurieren Sie Step Functions so, dass sie die von den einzelnen Zweigen ausgelösten Fehler abfangen.

Problembehandlung bei Serviceintegrationen

Mein Job im Downstream-Dienst ist abgeschlossen, aber in Step Functions bleibt der Aufgabenstatus „In Bearbeitung“ oder die Fertigstellung ist verzögert.

Für `.sync` Dienstintegrationsmuster verwendet Step Functions EventBridge Regeln, Downstream-APIs oder eine Kombination aus beiden, um den Downstream-Auftragsstatus zu ermitteln. Für einige Dienste erstellt Step Functions keine EventBridge Regeln zur Überwachung. Beispielsweise führt Step Functions für die AWS Glue Serviceintegration einen `glue:GetJobRun` Aufruf durch, anstatt EventBridge Regeln zu verwenden. Aufgrund der Häufigkeit von API-Aufrufen besteht ein Unterschied zwischen der Erledigung der Downstream-Aufgabe und der Erledigungszeit der Step Functions-Aufgabe. Step Functions benötigt IAM-Berechtigungen, um die EventBridge Regeln zu verwalten und Aufrufe an den Downstream-Service zu tätigen. Weitere Informationen darüber, wie sich unzureichende Berechtigungen für Ihre Ausführungsrolle auf die Erledigung von Aufgaben auswirken können, finden Sie unterZusätzliche Berechtigungen für Aufgaben, die das Run a Job-Muster verwenden.

Ich möchte eine JSON-Ausgabe aus einer Nested State Machine-Ausführung zurückgeben.

Es gibt zwei synchrone Dienstintegrationen von Step Functions für Step Functions: `startExecution.sync` und. `startExecution.sync:2` Beide warten, bis die Nested-State-Maschine fertig ist, geben aber unterschiedliche `Output` Formate zurück. Sie können verwenden`startExecution.sync:2`, um eine JSON-Ausgabe unter zurückzugeben`Output`.

Ich kann eine Lambda-Funktion nicht von einem anderen Konto aus aufrufen.

Zugriff auf die Lambda-Funktion mit kontoübergreifender Unterstützung

Wenn in Ihrer Region kontoübergreifender Zugriff auf AWS Ressourcen verfügbar ist, verwenden Sie die folgende Methode, um eine Lambda-Funktion von einem anderen Konto aus aufzurufen.

Gehen Sie wie folgt vor, um eine kontenübergreifende Ressource in Ihren Workflows aufzurufen:

1. Erstellen Sie eine IAM-Rolle in dem Zielkonto, das die Ressource enthält. Diese Rolle gewährt dem Quellkonto, das die State-Maschine enthält, Berechtigungen für den Zugriff auf die Ressourcen des Zielkontos.
2. Geben Sie in der `Task` Statusdefinition die Ziel-IAM-Rolle an, die von der State-Maschine übernommen werden soll, bevor die kontoübergreifende Ressource aufgerufen wird.
3. Ändern Sie die Vertrauensrichtlinie in der Ziel-IAM-Rolle, damit das Quellkonto diese Rolle vorübergehend übernehmen kann. Die Vertrauensrichtlinie muss den Amazon-Ressourcennamen (ARN) der im Quellkonto definierten State-Maschine enthalten. Definieren Sie außerdem die entsprechenden Berechtigungen in der Ziel-IAM-Rolle, um die AWS Ressource aufzurufen.
4. Aktualisieren Sie die Ausführungsrolle des Quellkontos, sodass sie die erforderliche Berechtigung enthält, um die Ziel-IAM-Rolle zu übernehmen.

Ein Beispiel finden Sie unter Tutorial: Zugriff auf kontoübergreifende Ressourcen AWS.

> **Note**
>
> Sie können Ihren State Machine so konfigurieren, dass er eine IAM-Rolle für den Zugriff auf Ressourcen von mehreren AWS-Konten übernimmt. Eine State Machine kann jedoch zu einem bestimmten Zeitpunkt nur eine IAM-Rolle übernehmen.

Ein Beispiel für eine `Task` Statusdefinition, die eine kontenübergreifende Ressource spezifiziert, finden Sie unterBeispiele für das Feld Anmeldeinformationen für den Aufgabenstatus.

Zugriff auf die Lambda-Funktion ohne kontoübergreifende Unterstützung

Wenn in Ihrer Region kein kontoübergreifender Zugriff auf AWS Ressourcen verfügbar ist, verwenden Sie die folgende Methode, um eine Lambda-Funktion von einem anderen Konto aus aufzurufen.

Verwenden Sie im `Resource` Feld des `Task` Bundesstaates die `FunctionArn` In-Parameter `arn:aws:states:::lambda:invoke` und übergeben Sie sie. Die IAM-Rolle, die der State-Maschine zugeordnet ist, muss über die richtigen Berechtigungen verfügen, um kontoübergreifende Lambda-Funktionen aufzurufen:. `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
      }
   }
}
```

Ich kann keine von **`.waitForTaskToken`** Staaten übergebenen Aufgabentokens sehen.

Im `Parameters` Feld des `Task` Bundesstaates müssen Sie ein Task-Token übergeben. Sie könnten beispielsweise etwas Ähnliches wie den folgenden Code verwenden.

```
{
   "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**
>
> Sie können versuchen, es `.waitForTaskToken` mit jeder API-Aktion zu verwenden. Einige APIs haben jedoch keine geeigneten Parameter.

Aktivitäten zur Fehlerbehebung

Die Ausführung meiner State Machine steckt in einem Aktivitätsstatus fest.

Der Status einer Aktivitätsaufgabe beginnt erst, wenn Sie mithilfe der GetActivityTaskAPI-Aktion ein Task-Token abfragen. Es hat sich bewährt, ein Timeout auf Taskebene hinzuzufügen, um zu vermeiden, dass die Ausführung hängen bleibt. Weitere Informationen finden Sie unter Verwenden Sie Timeouts, um festgefahrene Ausführungen zu vermeiden.

Wenn Ihr State Machine bei dem ActivityScheduledEvent nicht weiterkommt, deutet dies darauf hin, dass Ihre Activity Worker-Flotte Probleme hat oder dass sie unterdimensioniert ist. Sie sollten die `ActivityScheduleTime`CloudWatchMetrik überwachen und einen Alarm auslösen, wenn diese Zeit zunimmt. Um jedoch festgefahrene Zustandsmaschinen-Ausführungen, bei denen der `Activity` Status nicht in den Status übergeht, zu beenden, definieren Sie einen Timeout auf `ActivityStarted` Zustandsmaschinenebene. Geben Sie dazu ein `TimeoutSeconds` Feld am Anfang der State-Machine-Definition außerhalb des `States` Felds an.

Mein Activity Worker hat ein Timeout, während er auf ein Task-Token wartet.

Worker verwenden die GetActivityTaskAPI-Aktion, um eine Aufgabe mit dem angegebenen Aktivitäts-ARN abzurufen, die für die Ausführung durch eine laufende State-Maschine geplant ist. `GetActivityTask`startet eine lange Umfrage, sodass der Dienst die HTTP-Verbindung offen hält und reagiert, sobald eine Aufgabe verfügbar wird. Die maximale Zeit, in der der Dienst die Anfrage zurückhält, bevor er antwortet, beträgt 60 Sekunden. Wenn innerhalb von 60 Sekunden keine Aufgabe verfügbar ist, gibt die Umfrage a `taskToken` mit einer Nullzeichenfolge zurück. Um dieses Timeout zu vermeiden, konfigurieren Sie einen clientseitigen Socket mit einem Timeout von mindestens 65 Sekunden im AWS SDK oder in dem Client, den Sie für den API-Aufruf verwenden.

Problembehandlung bei Express-Workflows

Bei meiner Anwendung tritt ein Timeout auf, bevor ich eine Antwort von einem **`StartSyncExecution`** API-Aufruf erhalte.

Konfigurieren Sie ein clientseitiges Socket-Timeout in dem AWS SDK oder Client, den Sie für den API-Aufruf verwenden. Um eine Antwort zu erhalten, muss das Timeout einen Wert haben, der höher ist als die Dauer der Express Workflow-Ausführungen.

Ich kann den Ausführungsverlauf nicht einsehen, um Express Workflow-Fehler zu beheben.

Express Workflows zeichnen keine Ausführungshistorie aufAWS Step Functions. Stattdessen müssen Sie die CloudWatch Protokollierung aktivieren. Sobald die Protokollierung aktiviert ist, können Sie CloudWatch Logs Insights-Abfragen verwenden, um Ihre Express Workflow-Ausführungen zu überprüfen. Sie können den Ausführungsverlauf für Express Workflow-Ausführungen auch auf der Step Functions-Konsole einsehen, wenn Sie auf der Registerkarte Ausführungen auf die Schaltfläche Aktivieren klicken. Weitere Informationen finden Sie unter Anzeigen und Debuggen von Ausführungen in der Step Functions-Konsole.

Um Hinrichtungen anhand der Dauer aufzulisten:

```
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
```

Um fehlgeschlagene und abgesagte Hinrichtungen aufzulisten:

```
fields ispresent(execution_arn) as isRes | filter type in ["ExecutionFailed",
 "ExecutionAborted", "ExecutionTimedOut"]
```

Ähnliche Informationen

Die folgende Tabelle enthält verwandte Ressourcen, die für die Arbeit mit diesem Service nützlich sind.

Ressource	Beschreibung
AWS Step Functions API Referenz	Beschreibungen der API-Aktionen, -Parameter und -Datentypen sowie eine Liste von Fehlern, die der Service zurückgibt.
AWS Step FunctionsBefehlszeilenreferenz	Beschreibungen der AWS CLI-Befehle, die Sie für die Arbeit mit AWS Step Functions verwenden können.
Produktinformationen für Step Functions	Die Hauptwebseite für Informationen zu Step Functions.
Diskussionsforen	Ein Community-Forum, in dem Entwickler technische Fragen zu Step Functions und anderen AWS Diensten erörtern können.
AWS SupportInformationen	Die wichtigste Webseite für Informationen über AWS Support einen schnell reagieren den Supportkanalne-on-one, der Sie bei der Entwicklung und Ausführung von Anwendung en auf AWS Infrastrukturdiensten unterstützt.

Aktuelle Feature-Starts

In der folgenden Tabelle sind die Regionen aufgeführt, in denen neue Step Functions-Funktionen verfügbar sind.

Startdatum	Feature name	Verfügbare Regionen
26. November 2023	Rufen Sie öffentliche HTTPS-Endpunkte auf und testen Sie einzelne Bundesstaaten	• USA Ost (Nord-Virginia) – us-east-1 • USA West (Oregon) – us-west-2 • USA Ost (Ohio) – us-east-2 • Europa (Irland) – eu-west-1 • Europa (Frankfurt) – eu-central-1 • Europa (Stockholm) – eu-north-1 • Asien-Pazifik (Sydney) – ap-southeast-2 • Asien-Pazifik (Tokio) – ap-northeast-1 • Asien-Pazifik (Singapur) – ap-southeast-1
15. November 2023	RedriveHinrichtungen	Eine vollständige Liste der Optionen, AWS-Regionen in denen diese Funktion verfügbar ist, finden Sie in den Optionen in der Dropdownliste Region auf der Seite mit AWS-Servicesdem Titel Region.
12. Oktober 2023	Optimierte Integration für Amazon EMR Serverless	Eine vollständige Liste der Optionen, AWS-Regionen in denen diese Funktion

Startdatum	Feature name	Verfügbare Regionen
		verfügbar ist, finden Sie in den Optionen in der Dropdownliste Region auf der Seite mit AWS-Servicesdem Titel Region.
07. September 2023	Verbesserte Fehlerbehandlung	Eine vollständige Liste der Optionen, AWS-Regionen in denen diese Funktion verfügbar ist, finden Sie in den Optionen in der Dropdownliste Region auf der Seite mit AWS-Servicesdem Titel „Region“.
31. August 2023	Verbesserungen von Workflow Studio für ein optimiertes Authoring-Erlebnis	Eine vollständige Liste der Länder, AWS-Regionen in denen diese Funktion verfügbar ist, finden Sie in den Optionen in der Dropdownliste „Region“ auf der Seite mit dem Titel „AWS-ServicesRegion“.
22. Juni 2023	Versionen und Aliase	Eine vollständige Liste der Optionen, AWS-Regionen in denen diese Funktion verfügbar ist, finden Sie in den Optionen in der Dropdownliste Region auf der Seite mit AWS-Servicesdem Titel Region.
16. Juni 2023	Neue AWS SDK-Integrationen	Eine vollständige Liste der Optionen, AWS-Regionen in denen diese Funktion verfügbar ist, finden Sie in den Optionen in der Dropdownliste Region auf der Seite mit AWS-Servicesdem Titel „Region“.

Startdatum	Feature name	Verfügbare Regionen
01. Dezember 2022	Orchestrieren Sie umfangreiche parallel Workflows für die Datenverarbeitung mit Distributed Map State	Eine vollständige Liste der Länder, AWS-Regionen in denen diese Funktion verfügbar ist, finden Sie in den Optionen in der Dropdownliste „Region“ auf der Seite mit AWS-Servicesdem Titel „Region“.

Dokumentverlauf

In diesem Abschnitt werden die wichtigsten Änderungen am AWS Step Functions Entwicklerhandbuch aufgeführt.

Änderung	Beschreibung	Datum geändert
Aktualisierungen	AWS verwaltete Richtlinienaktualisierungen — neue Berechtigung: `states:ValidateStateMachineDefinition` Es wurden Informationen über die neue Berechtigung zur Überprüfung der Syntax einer von Ihnen bereitgestellten Zustandsmaschine hinzugefügt. Weitere Informationen hierzu finden Sie unter AWS verwaltete Richtlinien für AWS Step Functions.	29. April 2024
Neues Feature	Step Functions bietet eine optimierte Integration für AWS Elemental MediaConvert AWS Elemental MediaConvert bietet Transcodierung von Video- und Audiodateien in Sendequalität, die Kunden mit Code automatisieren können, der ihren Medienworkflows entspricht. Dank der optimierten Integration für AWS Step Functions in ist es jetzt möglich MediaConvert, die Orchestrierung mithilfe des visuellen Low-Code-Tools Workflow Studio durchzuführen. Weitere Informationen finden Sie in der Dokumentation Manage AWS Elemental MediaConvert with Step Functions.	12. April 2024
Aktualisierungen	AWS verwaltete Richtlinienaktualisierungen — Aktualisierung einer bestehenden Richtlinie: `AWSStepFunctionsReadOnlyAccess` Es wurden Informationen über neue Nur-Lese-Berechtigungen für Tags, verteilte Maps sowie Versionen und Aliase	02. April 2024

Änderung	Beschreibung	Datum geändert
	hinzugefügt. Weitere Informationen hierzu finden Sie unter AWS verwaltete Richtlinien für AWS Step Functions.	
Aktualisierungen	Step Functions bietet Unterstützung für Open Workflow-Metriken Mit Metriken für offene Workflows haben Sie jetzt auf Kontoebene einen Überblick über die Anzahl der laufenden Standard-Workflows sowie über Ihr Limit für offene Workflows. Sie können Workloads für alle Workflows verwalten, unabhängig davon, wie sie gestartet wurden, um einen reibungslosen Workflow-Betrieb zu gewährlei sten. Sie können CloudWatch Alarme einrichten, um Ihre Workflows zu überwachen und proaktiv Benachric htigungen zu erhalten, wenn Sie sich Ihren Grenzwerten nähern. Sobald Sie eine Benachrichtigung erhalten, können Sie Ihre Workflows effektiv verwalten, indem Sie beispiels weise bestimmte Workflows stoppen oder eine Erhöhung des Limits beantragen. Offene Workflow-Metriken können CloudWatch für Standard-Workflows verwendet werden, ohne dass eine zusätzliche Konfiguration erforderlich ist. Weitere Informati onen hierzu finden Sie unter Ausführungsmetriken.	29. Februar 2024
Aktualisierungen	Ergänzungen und Aktualisierungen der Serviceintegration . Eine Liste der neuen und aktualisierten AWS SDK-Integ rationen finden Sie unterÄnderungsprotokoll für unterstützte AWS SDK-Integrationen. Die vollständige Liste der Dienste finden Sie unterUnterstützte AWS SDK-Dienstintegrat ionen.	18. Januar 2024

Änderung	Beschreibung	Datum geändert
Neues Feature	Verwenden Sie Workflow StudioApplication Composer, um serverlose Workflows mithilfe von AWS CloudForm ation Vorlagen zu erstellen. Weitere Informationen finden Sie unter Verwenden von Workflow Studio in Application Composer.	8. November 2023
Neues Feature	Step Functionsermöglicht es Ihnen jetzt, öffentliche HTTPS-Endpunkte direkt aufzurufen und einzelne Status mithilfe einer neuen Teststatus-API zu testen. Weitere Informationen finden Sie hier: • Rufen Sie APIs von Drittanbietern auf • Verwenden einer TestState API zum Testen eines Zustands	26. November 202
Neues Feature	Step Functions ist jetzt in Amazon Bedrock integriert. Weitere Informationen finden Sie unter den folgenden Themen: • Aufrufen von Amazon Bedrock mit Step Functions • IAM-Berechtigungen für Amazon Bedrock • Führen Sie AI-Prompt-Chaining durch mit Amazon Bedrock • Verwendung AWS Step Functions mit anderen Diensten	26. November 202
Neues Feature	Mit Step Functions können Sie jetzt redrive Workflow-Ausführungen vom Typ Standard ab dem Fehlerpun kt ausführen. Weitere Informationen finden Sie unter Redriving -Ausführungen und RedrivingKarte läuft.	15. November 202

Änderung	Beschreibung	Datum geändert
Aktualisierung nur für die Dokumentation	Es wurde ein neues Thema veröffentlicht, in dem erklärt wird, wie Zustandsmaschinen nach einem Zeitplan ausgeführt werden. Verwenden Sie dazu. Amazon EventBridge Scheduler Weitere Informationen finden Sie unter Verwenden von Amazon EventBridge Scheduler mit AWS Step Functions.	16. Oktober 2023
Neues Feature	Step Functions ist jetzt in Amazon EMR Serverless integriert. Weitere Informationen finden Sie unter den folgenden Themen: • Aufrufen von Amazon EMR Serverless mit Step Functions • Einen EMR Serverless Job ausführen • Optimierte Integrationen für Step Functions • Verwendung AWS Step Functions mit anderen Diensten	12. Oktober 2023
Update nur für die Dokumentation	Es wurden Informationen zum planmäßigen Ausführen von Zustandsmaschinen hinzugefügt. Amazon EventBridge Scheduler Weitere Informationen finden Sie unter Scheduler verwenden EventBridge .	5. Oktober 2023
Aktualisierung	Die Statusthemen von Distributed Map wurden neu organisiert und aktualisiert, um sie übersichtlicher und übersichtlicher zu gestalten und eine klare Übersicht für neue Benutzer zu schaffen. Weitere Informationen finden Sie unter Verwenden von Map State im verteilten Modus zur Orchestrierung umfangreicher parallel Workloads.	06. Oktober 2023
Behobene Probleme	Codebeispiele in einem Tutorial zur Verwendung AWS CDK von Version 2 wurden korrigiert. Weitere Informationen finden Sie unter Erstellen einer Lambda Zustandsmaschine für die Step Functions Verwendung AWS CDK.	19. September 202

Änderung	Beschreibung	Datum geändert
Aktualisierung	Es wurden Informationen zu den erweiterten Funktionen zur Fehlerbehandlung hinzugefügt, die Step Functions eingeführt hat, um Fehler eindeutig zu identifizieren und Wiederholungsversuche mit größerer Kontrolle zu implementieren. Weitere Informationen finden Sie unter Fehler und Wiederholter Versuch nach einem Fehler.	07. September 2023
Aktualisierung	Step Functions hat Workflow Studio um Verbesserungen erweitert, um die Workflow-Erstellung zu optimieren. Weitere Informationen finden Sie unter AWS Step Functions Workflow-Studio.	31. August 2023
Update nur für die Dokumentation	Es wurden Informationen hinzugefügt, die etwa doppelt so hoch sind wie die tatsächliche Metrikzahl, die für die `ExecutionsStarted` Metrik gemeldet wurde. Weitere Informationen finden Sie unter Metriken, die eine Anzahl melden.	25. Juli 2023
Aktualisierung nur für die Dokumentation	Step Functions hat zwei neue Beispielprojekte hinzugefügt, die die folgenden häufigen Anwendungsfälle für den Status Distributed Map demonstrieren: • Eine CSV-Datei wird verarbeitet • Verarbeitung von Daten in einem Amazon S3 S3-Bucket	17. Juli 2023
Aktualisierung nur für die Dokumentation	Hat ein neues Thema zur Bereitstellung von Zustandsmaschinen mithilfe von Terraform veröffentlicht. Weitere Informationen finden Sie unter Bereitstellen von Zustandsmaschinen mit Terraform.	5. Juli 2023

Änderung	Beschreibung	Datum geändert
Update nur für die Dokumentation	Die folgenden Verfahren wurden aktualisiert, um den Änderungen an der EventBridge Amazon-Benutzerobe rfläche Rechnung zu tragen. • Weiterleiten eines Step-Functions-Ereignisses an EventBridge • Starten einer State Machine-Ausführung als Reaktion auf Amazon S3 S3-Ereignisse	26. Juni 2023
Neues Feature	Step Functions bietet jetzt die Möglichkeit, mehrere State-Machine-Versionen und Aliase zu erstellen, um die Ausfallsicherheit bei der Bereitstellung serverloser Workflows zu verbessern. Weitere Informationen finden Sie unter Verwaltung kontinuierlicher Bereitstellungen mit Versionen und Aliasnamen.	22. Juni 2023
Update nur für die Dokumentation	Die Beschreibung von `TimeoutSeconds` und die `HeartbeatSeconds` Felder wurden verbessert, um zu beschreiben, wie sie sich voneinander unterscheiden. Weitere Informationen finden Sie unter Felder für den Aufgabenstatus.	22. Juni 2023
Update nur für die Dokumentation	Es wurde ein neuer Abschnitt veröffentlicht, in dem beschrieben wird, wie ein Array von Arrays reduziert wird, die normalerweise als Ergebnis für den Status Parallel und Map zurückgegeben werden. Weitere Informationen finden Sie unter Ein Array von Arrays reduzieren.	20. Juni 2023
Aktualisierung	Step Functions hat die Unterstützung für AWS SDK-Integ rationen um sieben AWS-Services und 468 neue API-Aktionen erweitert. Weitere Informationen finden Sie unter Unterstützte AWS SDK-Dienstintegrationen und Änderungs protokoll für unterstützte AWS SDK-Integrationen.	16. Juni 2023

Änderung	Beschreibung	Datum geändert
Update nur für die Dokumentation	Es wurde ein neues Thema veröffentlicht, AWS-Regionen in dem die kürzlich eingeführten Step Functions-Funktion en aufgeführt sind. Weitere Informationen finden Sie unter Aktuelle Feature-Starts.	16. Juni 2023
Update nur für die Dokumentation	Step Functions enthält jetzt einen Abschnitt über AWS-Benutzerbenachrichtigungen, der als zentraler Ort für Ihre AWS Benachrichtigungen in der dient AWS Managemen t Console. AWS-Service Weitere Informationen finden Sie unter AWS-BenutzerbenachrichtigungenMit Step-Funk tionen verwenden.	4. Mai 2023
Update nur für die Dokumentation	Es wurde ein neuer Abschnitt hinzugefügt, in dem die Berechtigungen erläutert werden, die erforderlich sind, um untergeordnete Workflow-Ausführungsergebnisse in einen Amazon S3 S3-Bucket zu schreiben, der mit einem AWS Key Management Service (AWS KMS) Schlüssel verschlüs selt ist. Weitere Informationen finden Sie unter IAM-Berec htigungen für AWS KMS key verschlüsselten Amazon S3 S3-Bucket.	29. April 2023
Update nur für die Dokumentation	Es wurde ein neues Thema hinzugefügt, das die Funktion des Datenflusssimulators erklärt. Weitere Informationen finden Sie unter Datenflusssimulator.	14. April 2023
Aktualisierung des Kontingents	Es wurden Informationen zum Standardkontingent von 1000 für offene Map Runs in jedem Konto hinzugefügt. Weitere Informationen finden Sie unter Kontingente im Zusammenhang mit Konten.	05. April 2023

Änderung	Beschreibung	Datum geändert
Update nur für die Dokumentation	Es wurde ein Thema hinzugefügt, das beschreibt, wann AWS Data Pipeline Workloads zu Step Functions migriert werden sollten. Dieses Thema enthält auch eine Liste mit Beispielen, in denen erklärt wird, wie die Migration durchgeführt wird. Weitere Informationen finden Sie unter Migrieren von Workloads von AWS Data Pipeline zu Step Functions.	30. März 2023
Update nur für die Dokumentation	Es wurde ein Hinweis zur Nichtverfügbarkeit von X-Ray Tracing für den Status Distributed Map hinzugefügt. Weitere Informationen finden Sie unter AWS X-Ray und Step Functions.	21. März 2023
Update nur für die Dokumentation	Es wurden Informationen darüber hinzugefügt, wie Step Functions die Tag-basierte Autorisierung handhabt. Weitere Informationen finden Sie unter Funktionen zum Taggen in Step und Tagbasierte Richtlinien.	15. März 2023
Update nur für die Dokumentation	Es wurden Informationen darüber hinzugefügt, wie Step Functions CSV-Dateien analysiert, die als Eingabe im Status Distributed Map verwendet werden. Weitere Informationen finden Sie unter CSV-Datei in einem Amazon S3 S3-Bucket.	14. März 2023
Update nur für die Dokumentation	Es wurden Informationen darüber hinzugefügt, wie Step Functions kontenübergreifende Aufrufe für das Run a Job (.sync) -Muster verarbeitet. Weitere Informationen finden Sie unter Job ausführen (.sync).	01. März 2023
Update nur für die Dokumentation	Reduzieren Sie den Aufbewahrungszeitraum für den Verlauf Ihrer abgeschlossenen Workflow-Ausführungen von 90 Tagen auf 30 Tage. Weitere Informationen zur Anpassung der Aufbewahrungsfrist finden Sie unter Die Ausführung garantiert undKontingente im Zusammenhang mit der Ausführung von Zustandsmaschinen.	21. Februar 2023

Änderung	Beschreibung	Datum geändert
Aktualisierung	Step Functions hat die Unterstützung für AWS SDK-Integrationen um 35 AWS Dienste und 1100 neue API-Aktionen erweitert. Weitere Informationen finden Sie unter Unterstützte AWS SDK-Dienstintegrationen und Änderungsprotokoll für unterstützte AWS SDK-Integrationen.	17. Februar 2023
Update nur für die Dokumentation	Es wurde eine Tutorialserie „Erste Schritte“ veröffentlicht, die Sie durch den Prozess der Erstellung eines Workflows für Kreditkartenanträge mithilfe von Step Functions führt. Weitere Informationen finden Sie unter Erste Schritte mit AWS Step Functions.	30. Dezember 2022
Neues Feature	Step Functions bietet Unterstützung für die Orchestrierung umfangreicher parallel Workflows für die Datenverarbeitung mithilfe eines neuen verteilten `Map` Zustandsmodus. Weitere Informationen finden Sie unter Verwenden von Map State im verteilten Modus zur Orchestrierung umfangreicher parallel Workloads.	01. Dezember 2022
Neues Feature	Step Functions unterstützt jetzt den Zugriff auf kontoübergreifende AWS Ressourcen, die in anderen Konten konfiguriert sind. Weitere Informationen finden Sie unter • Zugreifen auf Ressourcen AWS-Konten in anderen Workflows • Tutorial: Zugriff auf kontoübergreifende Ressourcen AWS • Task state	18. November 202

Änderung	Beschreibung	Datum geändert
Aktualisierung	Step Functions bietet jetzt eine neue Konsolenoberfläche zum Anzeigen und Debuggen von Express-Workflow-Ausführungen. Weitere Informationen finden Sie unter: • Standard- und Express-Workflow-Ausführungen in der Konsole • Anzeigen und Debuggen von Ausführungen in der Step Functions-Konsole	18. Oktober 2022
Aktualisierung	Unterstützung für die optionale Angabe des `ExecutionRoleArn` Parameters bei der Verwendung der `addStep.sync` APIs `addStep` und für die optimierte Amazon EMR-Serviceintegration hinzugefügt. Weitere Informationen finden Sie unter Rufen Sie Amazon EMR mit Step Functions auf.	20. September 2022
Aktualisierung nur für die Dokumentation	Es wurde ein neues Thema hinzugefügt, das Empfehlungen zur Kostenoptimierung bei der Erstellung serverloser Workflows mithilfe von Step Functions enthält. Weitere Informationen finden Sie unter Kostenoptimierung mit Express Workflows.	15. September 2022

Änderung	Beschreibung	Datum geändert
Aktualisierung	Step Functions bietet Unterstützung für 14 neue interne Funktionen zur Ausführung von Datenverarbeitungsaufgaben, wie Array-Manipulationen, Datenkodierung und -dekodierung, Hash-Berechnungen, JSON-Datenmanipulation, mathematische Funktionsoperationen und Generierung eindeutiger Identifikatoren. Update nur für die Dokumentation: Alle vorhandenen und neu eingeführten systeminternen Funktionen wurden je nach Art der Datenverarbeitungsaufgabe, bei deren Ausführung sie Ihnen helfen, in die folgenden Kategorien eingeteilt: • Intrinsische Eigenschaften für Arrays • Intrinsische Funktionen für die Datenkodierung und -dekodierung • Intrinsisch für die Hash-Berechnung • Intrinsik für die Manipulation von JSON-Daten • Intrinsik für mathematische Operationen • Systemimmanent für die String-Operation • Intrinsisch für die Generierung eindeutiger Identifikatoren • Intrinsisch für den generischen Betrieb Weitere Informationen finden Sie unter Intrinsische Funktionen.	31. August 2022
Aktualisierung	Step Functions hat die Unterstützung für AWS SDK-Integrationen um drei weitere AWS Dienste erweitert — AWS Billing ConductorAmazon GameSparks, undAmazon Pinpoint SMS and Voice V2. Weitere Informationen finden Sie unter Änderungsprotokoll für unterstützte AWS SDK-Integrationen.	26. Juli 2022

Änderung	Beschreibung	Datum geändert
Update nur für die Dokumentation	Es wurde ein neues Thema hinzugefügt, das eine Zusammenfassung aller Aktualisierungen der von Step Functions unterstützten AWS SDK-Integrationen enthält. Weitere Informationen finden Sie unter Änderungsprotokoll für unterstützte AWS SDK-Integrationen.	26. Juli 2022
Update nur für die Dokumentation	AWS Step Functions Das Entwicklerhandbuch enthält jetzt Einzelheiten zu den Ausführungsmetriken, die speziell für Express Workflows ausgegeben werden. Weitere Informationen finden Sie unter Ausführungsmetriken für Express Workflows.	09. Juni 2022

Änderung	Beschreibung	Datum geändert
Aktualisierung	Verbesserungen der Step Functions Functions-Konsole Die Konsole verfügt jetzt über eine neu gestaltete Seite mit den Ausführungsdetails, die die folgenden Verbesserungen enthält: • Möglichkeit, den Grund für eine fehlgeschlagene Ausführung auf einen Blick zu erkennen. • Zwei neue Visualisierungsmodi für Ihre Zustandsm aschine — Tabellenansicht und Ereignisansicht. Diese Ansichten bieten Ihnen auch die Möglichkeit, Filter anzuwenden, um nur die Informationen anzuzeigen, die für Sie von Interesse sind. Darüber hinaus können Sie den Inhalt der Ereignisansicht anhand der Zeitstempel des Ereignisses sortieren. • Wechseln Sie im Modus der Diagrammansicht mithilfe einer Dropdownliste oder in der Strukturansicht für **`Map`** Status im Tabellenansichtsmodus zwischen den verschiedenen Statusiterationen. `Map` • Sehen Sie sich detaillierte Informationen zu jedem Status im Workflow an, einschließlich des vollständ igen Datenübertragungspfads für `Task` Eingabe und Ausgabe, und versuchen Sie es erneut mit oder. `Parallel` • Verschiedene Verbesserungen, darunter die Option, den Amazon-Ressourcennamen der Zustandsmaschine zu kopieren, die Gesamtzahl der Zustandsmaschinen-Übergänge anzuzeigen und die Ausführungsdetails im JSON-Format zu exportieren. Aktualisierungen nur für die Dokumentation	09. Mai 2022

Änderung	Beschreibung	Datum geändert
	Es wurde ein neues Thema hinzugefügt, um die verschied enen Arten von Informationen zu erläutern, die auf der Seite mit den Ausführungsdetails angezeigt werden. Außerdem wurde ein Tutorial hinzugefügt, das zeigt, wie diese Informationen untersucht werden können. Weitere Informationen finden Sie hier: • Anzeigen und Debuggen von Ausführungen in der Step Functions-Konsole • Tutorial: Untersuchen von Ausführungen von Zustandsa utomaten mithilfe der Step-Functions-Konsole	
Aktualisierung	Step Functions bietet jetzt eine Problemumgehung, um das Sicherheitsproblem „Confused Deputy“ zu verhindern, das entsteht, wenn eine Entität (ein Dienst oder ein Konto) von einer anderen Entität gezwungen wird, eine Aktion auszuführen. Weitere Informationen finden Sie hier: • Vermeiden Sie dienstübergreifende Probleme mit verwirrten Stellvertretern	02. Mai 2022

Änderung	Beschreibung	Datum geändert

Aktualisierung	• Step Functions hat die Unterstützung für AWS SDK-Integrationen um 21 weitere AWS Dienste erweitert. Weitere Informationen finden Sie unter: Unterstützte AWS SDK-Dienstintegrationen.
	• Updates nur für die Dokumentation:
	  • Es wurde eine Liste aller Ausnahmepräfixe hinzugefügt, die in den Ausnahmen enthalten sind, die generiert werden, wenn Sie fälschlicherweise eine AWS SDK-Dienstintegration mit Step Functions durchführen. Weitere Informationen finden Sie unter: Unterstützte AWS SDK-Dienstintegrationen.
	  • Es wurde eine Liste aller nicht unterstützten API-Aktionen für unterstützte SDK-Integrationen hinzugefügt. AWS Weitere Informationen finden Sie unter: Nicht unterstützte API-Aktionen für unterstützte Dienste.
	  • Es wurde eine Liste aller unterstützten AWS SDK-Integrationen hinzugefügt, die jetzt veraltet sind. Weitere Informationen finden Sie unter: Veraltete AWS SDK-Serviceintegrationen.
		19. April 2022

Neues Feature	Step Functions Local unterstützt jetzt die AWS SDK-Integration und das Mocking von Serviceintegrationen. Weitere Informationen finden Sie hier: • Verwenden von Mocked-Service-Integrationen	28. Januar 2022

Änderung	Beschreibung	Datum geändert
Neues Feature	AWS Step Functions unterstützt jetzt die Erstellung einer Amazon API Gateway Gateway-REST-API mit synchrone r Express-Zustandsmaschine als Backend-Integration mithilfe von. AWS Cloud Development Kit (AWS CDK) Weitere Informationen finden Sie hier: • Erstellen einer API Gateway-REST-API mit einem synchronen Express-Zustandsautomaten mithilfe der AWS CDK	10. Dezember 202
Aktualisierung	Step Functions hat drei neue Beispielprojekte hinzugefü gt, die die Integration von Step Functions und der aktualisi erten Konsole von Amazon Athena demonstrieren. Weitere Informationen finden Sie hier: • Führen Sie mehrere Abfragen aus (Amazon Athena, Amazon SNS) • Große Datensätze abfragen (Amazon Athena, Amazon S3 AWS Glue, Amazon SNS) • Daten auf dem neuesten Stand halten (Amazon Athena, Amazon S3, AWS Glue)	22. November 2021
Neues Feature	Step Functions hat Amazon VPC-Endpunkt-Unterstützung für Synchronous Express-Workflows hinzugefügt. Weitere Informationen finden Sie hier: • Amazon VPC-Endpunkte für Step Functions	15. November 202

Änderung	Beschreibung	Datum geändert
Aktualisierung	AWS Step Functions hat drei neue Beispielprojekte hinzugefügt, die die Verwendung der Step AWS Batch Functions-Integration demonstrieren. Weitere Informationen finden Sie hier: • Einen AWS Batch Job ausfindig machen • AWS Batch mit Lambda • Verwenden Step Functions und AWS Batch mit Fehlerbehandlung	14. Oktober 2021
Neues Feature	AWS Step Functions hat AWS SDK-Integrationen hinzugefügt, sodass Sie die API-Aktionen für alle der mehr als zweihundert AWS Dienste verwenden können. Weitere Informationen finden Sie hier: • AWS SDK-Serviceintegrationen • Sammeln Sie Amazon S3 S3-Bucket-Informationen mithilfe von AWS SDK-Serviceintegrationen	30. September 2021
Neues Feature	AWS Step Functions hat einen visuellen Workflow-Designer hinzugefügt, das AWS Step Functions Workflow Studio. Weitere Informationen finden Sie hier: • AWS Step Functions Workflow-Studio • Lernen Sie, das AWS Step Functions Workflow Studio zu verwenden	17. Juni 2021
Aktualisierung	AWS Step Functions hat der CodeBuild Integration vier neue APIs `StartBuildBatch` `StopBuildBatch` ,`DeleteBuildBatch` , `RetryBuildBatch` und, hinzugefügt. Weitere Informationen finden Sie hier: • Rufen Sie AWS CodeBuild mit Step Functions auf	4. Juni 2021

Änderung	Beschreibung	Datum geändert
Neues Feature	AWS Step Functions lässt sich jetzt in Amazon integrieren EventBridge. Weitere Informationen finden Sie hier: • Rufen Sie EventBridge mit Step Functions auf • IAM-Richtlinien für Step Functions und IAM-Richtlinien für Amazon EventBridge • Ein Beispielprojekt, das zeigt, wie Senden Sie ein benutzerdefiniertes Ereignis an EventBridge	14. Mai 2021
Aktualisierung	AWS Step Functions hat ein neues Beispielprojekt hinzugefügt, das zeigt, wie Step Functions und die Amazon Redshift Data API verwendet werden, um einen ETL/ELT-Workflow auszuführen. Weitere Informationen finden Sie hier: • Führen Sie ETL/ELT-Workflows mit Amazon Redshift aus (Lambda, Amazon Redshift Data API)	16. April 2021
Neues Feature	AWS Step Functions hat einen neuen Datenflusssimulator in der Konsole. Weitere Informationen finden Sie hier: • Step Functions Functions-Konsole	8. April 2021
Neues Feature	AWS Step Functions lässt sich jetzt mit Amazon EMR auf EKS integrieren. Weitere Informationen finden Sie hier: • Rufen Sie Amazon EMR auf EKS an mit AWS Step Functions	29. März 2021

Änderung	Beschreibung	Datum geändert
Aktualisierung	YAML-Unterstützung für State-Machine-Definitionen wurde zu AWS Toolkit for Visual Studio Code und hinzugefügt. AWS CloudFormation Weitere Informationen finden Sie hier: • Unterstützung für Definitionsformate • AWS Toolkit for Visual Studio Code	4. März 2021
Neues Feature	AWS Step Functions integriert sich jetzt in. AWS Glue DataBrew Weitere Informationen finden Sie hier: • AWS Glue DataBrew Jobs mit Step-Funktionen verwalten • Was ist AWS Glue DataBrew? im DataBrew Entwickle rhandbuch.	06. Januar 2021
Neues Feature	AWS Step Functions Synchronous Express Workflows sind jetzt verfügbar und bieten Ihnen eine einfache Möglichke it, Microservices zu orchestrieren. Weitere Informationen finden Sie hier: • Synchrone und asynchrone Express-Workflows • Ein Beispielprojekt, das zeigt, wie Synchrone Express-W orkflows aufrufen • Die StartSyncExecutionAPI-Dokumentation.	24. November 202
Neues Feature	AWS Step Functions lässt sich jetzt in Amazon API Gateway integrieren. Weitere Informationen finden Sie hier: • API Gateway mit Step-Funktionen aufrufen • IAM-Richtlinien für Step Functions und IAM-Richtlinien für Amazon API Gateway • Ein Beispielprojekt, das zeigt, wie Rufen Sie API Gateway auf	17. November 2020

Änderung	Beschreibung	Datum geändert
Neues Feature	AWS Step Functions lässt sich jetzt in Amazon Elastic Kubernetes Service integrieren. Weitere Informationen finden Sie hier: • Rufen Sie Amazon EKS mit Step Functions auf • IAM-Richtlinien für Step Functions und IAM-Richtlinien für Amazon EKS • Ein Beispielprojekt, das zeigt, wie Einen Amazon EKS-Cluster verwalten	16. November 2020
Neues Feature	AWS Step Functions lässt sich jetzt in Amazon Athena integrieren. Weitere Informationen finden Sie hier: • Rufen Sie Athena mit Step Functions auf • IAM-Richtlinien für Step Functions und IAM-Richtlinien für Amazon Athena • Ein Beispielprojekt, das zeigt, wie Eine Athena-Abfrage starten	22. Oktober 2020
Neues Feature	AWS Step Functions unterstützt jetzt end-to-end Tracing-Workflows mit AWS X-Ray, bietet Ihnen einen vollständigen Überblick über die Ausführung von Zustandsmaschinen und erleichtert die Analyse und das Debuggen Ihrer verteilten Anwendungen. Weitere Informationen finden Sie hier: • AWS X-Ray und Step Functions • IAM-Richtlinien für Step Functions und IAM-Richtlinien für AWS X-Ray • AWS Step Functions API Reference • TracingConfiguration	14. September 202

Änderung	Beschreibung	Datum geändert

Aktualisierung	AWS Step Functions unterstützt jetzt Nutzdatengrößen von bis zu 256 KB als UTF-8-kodierte Zeichenfolge. Auf diese Weise können Sie größere Payloads sowohl in Standard- als auch in Express-Workflows verarbeiten.
	
	Ihre vorhandenen Zustandsmaschinen müssen nicht geändert werden, um die größeren Payloads nutzen zu können. Sie müssen jedoch auf die neuesten Versionen des Step Functions SDK und Local Runner aktualisieren, um die aktualisierten APIs verwenden zu können. Weitere Informationen finden Sie hier:
	
	• Kontingente
	• the section called “Verwenden Sie Amazon S3 S3-ARNs, anstatt große Nutzlasten weiterzuleiten”
	• `States.DataLimitExceeded`
	• the section called “CloudWatchProtokolliert Payloads”
	• the section called “EventBridge Nutzlasten”
	• AWS Step Functions API Reference
	
	  • CloudWatchEventsExecutionDataDetails
	  • HistoryEventExecutionDataDetails
	  • GetExecutionHistory
	  • ActivityScheduledEventDetails
	  • ActivitySucceededEventDetails
	  • CloudWatchEventsExecutionDataDetails
	  • ExecutionSucceededEventDetails
	  • LambdaFunctionScheduledEventDetails
	  • ExecutionSucceededEventDetails
	  • StateEnteredEventDetails
	  • StateExitedEventDetails
		3. September 2020

Änderung	Beschreibung	Datum geändert
	• TaskSubmittedEventDetails • TaskSucceededEventDetails	

Änderung	Beschreibung	Datum geändert

Aktualisierung	Die Sprache der Amazon-Staaten wurde wie folgt aktualisiert:
	• Auswahlregelnwurde hinzugefügt
	  • Ein Nullvergleichsoperator,`IsNull`. `IsNull`testet anhand des JSON-Nullwerts und kann verwendet werden, um festzustellen, ob die Ausgabe eines vorherigen Status Null ist oder nicht.
	  • Vier weitere neue Operatoren wurden hinzugefügt IsBoolean, IsNumeric, IsString und IsTimestamp.
	  • Ein Test auf Existenz oder Nichtexistenz eines Feldes unter Verwendung des `IsPresent` Operators. `IsPresent` kann verwendet werden, um `States.Runtime` Fehler zu verhindern, wenn versucht wird, auf einen nicht existierenden Schlüssel zuzugreifen.
	  • Musterabgleich mit Platzhaltern zur Unterstützung des Vergleichs von Zeichenketten mit Mustern mit einem oder mehreren Platzhaltern.
	  • Vergleich zwischen zwei Variablen für unterstützte Vergleichsoperatoren.
	• Timeout- und Heartbeat-Werte in einem `Task` Status können nun dynamisch aus der Statuseingabe statt eines festen Werts mithilfe der Felder `TimeoutSecondsPath` und `HeartbeatSecondsPath` bereitgestellt werden. Weitere Informationen finden Sie im Status der Aufgabe Bundesland.
	• Das neue ResultSelector Feld bietet eine Möglichkeit, das Ergebnis eines Zustands zu bearbeiten, bevor er angewendet `ResultPath` wird. Das `ResultSelector` Feld ist ein optionales Feld in den Status der Aufgabe Bundesstaaten ZuordnungParallel, und.
		13. August 2020

Änderung	Beschreibung	Datum geändert
	• Intrinsische Funktionenwurden hinzugefügt, um grundlegende Operationen ohne `Task` Status zu ermöglichen. Intrinsische Funktionen können innerhalb der Felder `Parameters` und `ResultSelector` verwendet werden.	
Aktualisierung	AWS Step Functions unterstützt jetzt den SageMaker `CreateProcessingJob` Amazon-API-Aufruf. Weitere Informationen finden Sie hier: • SageMaker Mit Step Functions verwalten • Daten vorverarbeiten und ein Modell für maschinelles Lernen trainieren, ein Beispielprojekt, das demonstri ert`CreateProcessingJob` .	4. August 2020
Neues Feature	AWS Step Functions wird jetzt von unterstützt AWS Serverless Application Model und erleichtert so die Integrati on der Workflow-Orchestrierung in Ihre serverlosen Anwendungen. Weitere Informationen finden Sie hier: • AWS Step Functions und AWS SAM • AWS::Serverless::StateMachine • AWS SAM Vorlagen für Richtlinien	27. Mai 2020

Änderung	Beschreibung	Datum geändert
Neues Feature	AWS Step Functions hat einen neuen synchronen Aufruf für die Verschachtelung von Step Functions Functions-Ausführungen eingeführt. Der neue Aufruf `arn:aws:states:::states:startExecution.sync:2` gibt ein JSON-Objekt zurück. Der ursprüngliche Aufruf `arn:aws:states:::states:startExecution.sync` wird weiterhin unterstützt und gibt eine in JSON maskierte Zeichenfolge zurück. Weitere Informationen finden Sie hier: • Managen Sie AWS Step Functions Ausführungen als integrierten Service	19. Mai 2020
Neues Feature	AWS Step Functions integriert sich jetzt mit. AWS CodeBuild Weitere Informationen finden Sie hier: • Verwendung AWS Step Functions mit anderen Diensten • Rufen Sie AWS CodeBuild mit Step Functions auf • Optimierte Integrationen für Step Functions	5. Mai 2020
Neues Feature	Step Functions wird jetzt in unterstützt AWS Toolkit for Visual Studio Code, sodass es einfacher ist, State-Machine-basierte Workflows zu erstellen und zu visualisieren, ohne den Code-Editor verlassen zu müssen.	31. März 2020
Aktualisierung	Sie können jetzt die Protokollierung in Amazon CloudWatch Logs für Standard-Workflows konfigurieren. Weitere Informationen finden Sie hier: • Protokollierung mitCloudWatchProtokolle	25. Februar 2020

Änderung	Beschreibung	Datum geändert
Neues Feature	AWS Step Functions kann jetzt direkt von Amazon Virtual Private Cloud (VPC) aus aufgerufen werden, ohne dass eine öffentliche IP-Adresse erforderlich ist. Weitere Informationen finden Sie hier: • Amazon VPC-Endpunkte für Step Functions	23. Dezember 2019

Änderung	Beschreibung	Datum geändert

Neues Feature	Express-Workflows sind ein neuer Workflow-Typ, der sich für hochvolumige Ereignisverarbeitungs-Workloads wie IoT-Datenaufnahme, Streaming-Datenverarbeitung und -transformation sowie mobile Anwendungs-Backends eignet.
	
	Weitere Informationen finden Sie in den folgenden neuen und aktualisierten Themen.
	
	• Standard- und Express-Workflows
	  • Die Ausführung garantiert
	• Verwendung AWS Step Functions mit anderen Diensten
	  • Optimierte Integrationen für Step Functions
	• Verarbeiten von Nachrichten mit hohem Volumen aus Amazon SQS (Express Workflows)
	• Beispiel für selektives Checkpointing (Express-Workflows)
	• Kontingente
	  • Kontingente
	• Protokollierung mitCloudWatchProtokolle
	• AWS Step Functions API Reference
	  • CreateStateMachine
	  • UpdateStateMachine
	  • DescribeStateMachine
	  • DescribeStateMachineForExecution
	  • StopExecution
	  • DescribeExecution
	  • GetExecutionHistory
	  • ListExecutions
	  • ListStateMachines
		3. Dezember 2019

Änderung	Beschreibung	Datum geändert
	• StartExecution • CloudWatchLogsLogGroup • LogDestination • LoggingConfiguration	
Neues Feature	AWS Step Functions lässt sich jetzt in Amazon EMR integrieren. Weitere Informationen finden Sie hier: • Verwendung AWS Step Functions mit anderen Diensten • Rufen Sie Amazon EMR mit Step Functions auf • Optimierte Integrationen für Step Functions	19. November 201

Aktualisierung	AWS Step Functions hat das AWS Step Functions Data Science SDK veröffentlicht. Weitere Informationen finden Sie unter den folgenden Topics.
	• Projekt auf Github
	• SDK-Dokumentation
	• Die folgenden Beispiel-Notebooks, die in der SageMaker Konsole und dem zugehörigen GitHub Projekt verfügbar sind.
	  • `hello_world_workflow.ipynb`
	  • `machine_learning_workflow_abalone.ipynb`
	  • `training_pipeline_pytorch_mnist.ipynb`
		7. November 2019

Änderung	Beschreibung	Datum geändert
Aktualisierung	Step Functions unterstützt jetzt mehr API-Aktionen für Amazon SageMaker und beinhaltet zwei neue Beispielp rojekte zur Demonstration der Funktionalität. Weitere Informationen finden Sie unter den folgenden Topics. • SageMaker Mit Step Functions verwalten • Verwendung AWS Step Functions mit anderen Diensten • Schulen eines Machine Learning-Modells • Optimieren eines Machine Learning-Modells	3. Oktober 2019
Neues Feature	Step Functions unterstützt das Starten neuer Workflow-Ausführungen durch Aufrufen `StartExecution` als integrierte Service-API. Siehe: • Starten von Workflow-Ausführungen über einen Aufgabenstatus • Managen Sie AWS Step Functions Ausführungen als integrierten Service • Verwendung AWS Step Functions mit anderen Diensten • IAM-Richtlinien für das Starten von Step Functions Functions-Workflow-Ausführungen	12. August 2019

Änderung	Beschreibung	Datum geändert
Neues Feature	Step Functions beinhaltet die Möglichkeit, ein Task-Token an integrierte Services zu übergeben und die Ausführung anzuhalten, bis das Task-Token mit `SendTaskSuccess` oder zurückgegeben wird`SendTaskFailure` . Siehe: • Muster der Serviceintegration • Warten auf einen Callback mit dem Aufgabentoken • Beispiel für ein Rückrufmuster (Amazon SQS, Amazon SNS, Lambda) • Optimierte Integrationen für Step Functions • Bereitstellen eines Beispielprojekts für Genehmigungen durch Menschen • Serviceintegrationsmetriken Step Functions bietet jetzt eine Möglichkeit, direkt im `"Parameters"` Feld einer Statusdefinition auf dynamische Informationen über Ihre aktuelle Ausführung zuzugreifen. Siehe: • Context-Objekt • Übergeben von Kontext-Knoten als Parameter	23. Mai 2019
Neues Feature	Step Functions unterstützt CloudWatch Ereignisse für Änderungen des Ausführungsstatus, siehe: • EventBridge (CloudWatch Ereignisse) für Statusänderungen bei der Ausführung von Step Functions • Amazon CloudWatch Events-Benutzerhandbuch	8. Mai 2019

Änderung	Beschreibung	Datum geändert
Neues Feature	Step Functions unterstützt IAM-Berechtigungen mithilfe von Tags. Weitere Informationen finden Sie hier: • Funktionen zum Taggen in Step • Tagbasierte Richtlinien	5. März 2019
Neues Feature	Step Functions Local ist jetzt verfügbar. Sie können Step Functions zu Test- und Entwicklungszwecken auf Ihrem lokalen Computer ausführen. Step Functions Local kann entweder als Java-Anwendung oder als Docker-Image heruntergeladen werden. Siehe Zustandsmaschinen lokal testen.	4. Februar 2019
Neues Feature	AWS Step Functions ist jetzt in den Regionen Peking und Ningxia verfügbar. Siehe Unterstützte -Regionen.	15. Januar 2018
Neues Feature	Step Functions unterstützt die Kennzeichnung von Ressourcen, damit Sie Ihre Kostenzuweisung verfolgen können. Sie können Zustandsautomaten auf der Seite Details oder mithilfe von API-Aktionen markieren. Siehe Funktionen zum Taggen in Step.	7. Januar 2019
Neues Feature	AWS Step Functions ist jetzt in den Regionen Europa (Paris) und Südamerika (São Paulo) verfügbar. Siehe Unterstützte -Regionen.	13. Dezember 2018
Neues Feature	AWS Step Functions ist jetzt in der Region Europa (Stockholm) verfügbar. Eine Liste der unterstützten Regionen finden Sie unter Unterstützte -Regionen.	12. Dezember 201

Änderung	Beschreibung	Datum geändert
Neues Feature	Step Functions ist jetzt in einige AWS Dienste integriert. Sie können jetzt Parameter direkt von einem Aufgabenstatus in der Amazon States-Sprache aus aufrufen und an die API dieser integrierten Dienste übergeben. Weitere Informationen finden Sie hier: • Verwendung AWS Step Functions mit anderen Diensten • Parameter an eine Service-API übergeben • Optimierte Integrationen für Step Functions	29. November 2018
Aktualisierung	Die Beschreibungen von `TimeoutSeconds` und `HeartbeatSeconds` in der Dokumentation für Aufgabenstatus wurden verbessert. Siehe Status der Aufgabe.	24. Oktober 2018
Aktualisierung	Beschreibung für das Limit Maximum execution history size (Maximale Größe des Ausführungsverlaufs) verbessert und Link zum Thema über zugehörige bewährte Methoden angegeben. • Kontingente im Zusammenhang mit der Ausführung von Zustandsmaschinen • Vermeiden Sie es, das historische Kontingent zu erreichen	17. Oktober 2018
Aktualisierung	Der AWS Step Functions Dokumentation wurde ein neues Tutorial hinzugefügt: SieheStarten einer State Machine-Ausführung als Reaktion auf Amazon S3 S3-Ereignisse.	25. September 2018
Aktualisierung	Eintrag Maximum in Step Functions-Konsole angezeigter Ausführungen aus der Dokumentation zu Limits entfernt. Siehe Kontingente.	13. September 2018

Änderung	Beschreibung	Datum geändert
Aktualisierung	Der AWS Step Functions Dokumentation wurde ein Thema mit bewährten Methoden zur Verbesserung der Latenz bei der Abfrage von Aktivitätsaufgaben hinzugefügt. Siehe Vermeiden Sie Latenz bei der Abfrage von Aktivitätsaufgaben.	30. August 2018
Aktualisierung	Das AWS Step Functions Thema „Aktivitäten“ und „Aktivitätshelfer“ wurde verbessert. Siehe Aktivitäten.	29. August 2018
Aktualisierung	Das AWS Step Functions Thema CloudTrail Integration wurde verbessert. Siehe API-Aufrufe aufzeichnen mit AWS CloudTrail.	7. August 2018
Aktualisierung	JSON-Beispiele wurden dem AWS CloudFormation Tutorial hinzugefügt. Siehe Erstellen einer Lambda-Zustandsmaschine für Step Functions mit AWS CloudFormation.	23. Juni 2018
Aktualisierung	Es wurde ein neues Thema zur Behandlung von Lambda-Servicefehlern hinzugefügt. Siehe Lambda-Serviceausnahmen behandeln.	20. Juni 2018
Neues Feature	AWS Step Functions ist jetzt in der Region Asien-Pazifik (Mumbai) verfügbar. Eine Liste der unterstützten Regionen finden Sie unter Unterstützte -Regionen.	28. Juni 2018
Neues Feature	AWS Step Functions ist jetzt in der Region AWS GovCloud (US-West) verfügbar. Eine Liste der unterstützten Regionen finden Sie unter Unterstützte -Regionen. Informationen zur Verwendung von Step Functions in der Region AWS GovCloud (US-West) finden Sie unter AWS GovCloud (US).	28. Juni 2018
Aktualisierung	Verbesserte Dokumentation zur Fehlerbehebung für `Parallel`-Status. Siehe Fehlerbehandlung.	20. Juni 2018

Änderung	Beschreibung	Datum geändert
Aktualisierung	Verbesserte Dokumentation zur Eingabe- und Ausgabeverarbeitung in Step Functions. Erfahren Sie, wie Sie `InputPath` , `ResultPath` und `OutputPath` verwenden, um den JSON-Fluss durch Ihre Workflows, Status und Aufgaben zu steuern. Siehe: • Eingabe- und Ausgabeverarbeitung in Step Functions • ResultPath	7. Juni 2018
Aktualisierung	Verbesserte Codebeispiele für parallele Status. Siehe Parallel.	4. Juni 2018
Neues Feature	Sie können jetzt API- und Servicemetriken in überwachen CloudWatch. Siehe Step Functions überwachen mit CloudWatch.	25. Mai 2018
Aktualisierung	`StartExecution` , `StopExecution` und `StateTransition` verfügen jetzt über erhöhte Ablehnungslimits in den folgenden Regionen: • USA Ost (Nord-Virginia) • USA West (Oregon) • Europa (Irland) Weitere Informationen finden Sie unter Kontingente.	16. Mai 2018
Neues Feature	AWS Step Functions ist jetzt in den Regionen USA West (Nordkalifornien) und Asien-Pazifik (Seoul) verfügbar. Eine Liste der unterstützten Regionen finden Sie unter Unterstützte -Regionen.	5. Mai 2018
Aktualisierung	Aktualisierte Prozeduren und Abbilder zum Anpassen an Änderungen an der Schnittstelle.	25. April 2018

Änderung	Beschreibung	Datum geändert
Aktualisierung	Ein neues Tutorial wurde hinzugefügt, das zeigt, wie eine neue Ausführung gestartet wird, um die Arbeit fortzusetzen. Siehe Fortsetzung lang andauernder Workflow-Ausführungen als neue Ausführung. In diesem Tutorial wird ein Entwurfsmuster beschrieben, das dabei helfen kann, einige Serviceeinschränkungen zu vermeiden. Siehe Vermeiden Sie es, das historische Kontingent zu erreichen.	19. April 2018
Aktualisierung	Verbesserte Einführung in Zustandsdokumentation, indem konzeptionelle Informationen über Zustandscomputer hinzugefügt werden. Siehe Zustände.	9. März 2018
Aktualisierung	Neben HTML, PDF und Kindle ist das AWS Step Functions Entwicklerhandbuch auch auf verfügbar GitHub. Um Feedback zu hinterlassen, wählen Sie das GitHub Symbol in der oberen rechten Ecke. 	2. März 2018
Aktualisierung	Es wurde ein Thema hinzugefügt, das andere Ressourcen im Zusammenhang mit Step Functions beschreibt. Siehe Ähnliche Informationen.	20. Februar 2018

Änderung	Beschreibung	Datum geändert

Neues Feature	• Wenn Sie eine neue Zustandsmaschine erstellen, müssen Sie bestätigen, dass dadurch eine IAM-Rolle erstellt AWS Step Functions wird, die den Zugriff auf Ihre Lambda-Funktionen ermöglicht.
	• Die folgenden Tutorials wurden aktualisiert, um die geringfügigen Änderungen am Erstellungsworkflow des Zustandsautomaten hervorzuheben:
	  • Eine Step Functions Functions-Zustandsmaschine erstellen, die Lambda verwendet
	  • Einen Activity State Machine mithilfe von Step Functions erstellen
	  • Behandlung von Fehlerbedingungen mithilfe einer Step Functions Functions-Zustandsmaschine
	  • Iteriere eine Schleife mit Lambda
		19. Februar 2018

Aktualisierung	Thema hinzugefügt, das ein in Ruby geschriebenes Beispiel für einen Aktivitäts-Worker beschreibt. Diese Implementierung kann verwendet werden, um direkt einen Aktivitäts-Workers in Ruby zu erstellen, oder ein Entwurfsmuster, um einen Aktivitäts-Workers in einer anderen Sprache zu erstellen. Siehe Beispiel für einen Aktivitäts-Worker in Ruby.	6. Februar 2018
Aktualisierung	Es wurde ein neues Tutorial hinzugefügt, das ein Entwurfsmuster beschreibt, das eine Lambda-Funktion verwendet, um eine Zählung zu iterieren. Siehe Eine Step Functions Functions-Zustandsmaschine erstellen, die Lambda verwendet.	31. Januar 2018

Änderung	Beschreibung	Datum geändert
Aktualisierung	Der Inhalt zu IAM-Einschlussberechtigungen und APIs wurde aktualisiert. `DescribeStateMachineForExecution` `UpdateStateMachine` Siehe Granulare IAM-Berechtigungen für Benutzer ohne Administratorrechte erstellen.	26. Januar 2018
Aktualisierung	Neu verfügbare Regionen hinzugefügt: Kanada (Zentral), Asien-Pazifik (Singapur). Siehe Unterstützte -Regionen.	25. Januar 2018
Aktualisierung	Die Tutorials und Verfahren wurden aktualisiert, um zu verdeutlichen, dass Sie mit IAM Step Functions als Rolle auswählen können.	24. Januar 2018
Aktualisierung	Neues Thema Bewährte Methoden hinzugefügt, in dem empfohlen wird, keine großen Nutzlasten zwischen den Zuständen zu übergeben. Siehe Verwenden Sie Amazon S3 S3-ARNs, anstatt große Nutzlasten weiterzuleiten.	23. Januar 2018
Aktualisierung	Verfahren korrigiert, um die aktualisierte Schnittstelle zum Erstellen eines Zustandsautomaten zu berücksichtigen: • Eine Step Functions Functions-Zustandsmaschine erstellen, die Lambda verwendet • Einen Activity State Machine mithilfe von Step Functions erstellen • Behandlung von Fehlerbedingungen mithilfe einer Step Functions Functions-Zustandsmaschine	17. Januar 2018

Änderung	Beschreibung	Datum geändert
Neues Feature	Sie können Beispielprojekte zur schnellen Bereitstellung von Zustandsautomaten und aller dazu gehörenden AWS -Ressourcen verwenden. Weitere Informationen finden Sie unter Beispielprojekte für Step Functions, Verfügbare Beispielprojekte sind unter anderem: • Umfrage zum Jobstatus (Lambda, AWS Batch) • Aufgabentimer (Lambda, Amazon SNS) **Note** Diese Beispielprojekte und die dazu gehörende Dokumentation ersetzen Tutorials, in denen die Implementierung der gleichen Funktionalität beschrieben wurde.	11. Januar 2018
Aktualisierung	Ein Abschnitt Bewährte Methoden wurde hinzugefügt, der Informationen enthält, um hängen gebliebene Ausführungen zu verhindern. Siehe Bewährte Methoden für Step Functions.	5. Januar 2018
Aktualisierung	Es wurde ein Hinweis hinzugefügt, wie sich Wiederholungsversuche auf die Preisgestaltung auswirken können: **Note** Wiederholungen werden als Statusübergänge behandelt. Informationen darüber, wie sich Statusübergänge auf die Abrechnung auswirken, finden Sie unter Step Functions — Preisgestaltung.	8. Dezember 2017

Änderung	Beschreibung	Datum geändert
Aktualisierung	Es wurden Informationen im Zusammenhang mit Ressourcennamen hinzugefügt: **Note** Mit Step Functions können Sie Namen für Zustandsmaschinen, Ausführungen und Aktivitäten sowie Labels erstellen, die Nicht-ASCII-Zeichen enthalten. Diese Nicht-ASCII-Namen funktionieren nicht mit Amazon. CloudWatch Um sicherzustellen, dass Sie CloudWatch Messwerte verfolgen können, wählen Sie einen Namen, der nur ASCII-Zeichen verwendet.	6. Dezember 2017
Aktualisierung	Die Informationen zur Sicherheitsübersicht wurden verbessert und ein Thema zu detaillierten IAM-Berechtigungen hinzugefügt. Siehe Sicherheit in AWS Step Functions und Granulare IAM-Berechtigungen für Benutzer ohne Administratorrechte erstellen.	27. November 2017
Neues Feature	Sie können einen vorhandenen Zustandsautomaten aktualisieren. Weitere Informationen finden Sie unter Aktualisieren Ihres Zustandsmaschinen.	15. November 2017

Änderung	Beschreibung	Datum geändert
Aktualisierung	Es wurde eine Notiz hinzugefügt, um `Lambda.Unknown` -Fehlern zu klären, und mit der Lambda-Dokumentation in den folgenden Abschnitten verknüpft: • Namen der Fehler • Schritt 3: Erstellen Sie eine Zustandsmaschine mit einem Catch-Feld **Note** Unbehandelte Fehler in Lambda werden wie `Lambda.Unknown` in der Fehlerausgabe gemeldet. Dazu gehören out-of-memory Fehler und Funktions-Timeouts. Sie können nach, oder abgleichen `Lambda.Unknown` `States.ALL` , `States.TaskFailed` um diese Fehler zu behandeln. Wenn Lambda die maximale Anzahl von Aufrufen erreicht, lautet der Fehler. `Lambda.TooManyRequestsException` Weitere Informationen zu Lambda-Funktionsfehlern finden Sie unter Fehlerbehandlung und automatische Wiederholungen im AWS Lambda Entwicklerhandbuch.	17. Oktober 2017
Aktualisierung	Die IAM-Anweisungen wurden korrigiert und klarer formuliert und die Screenshots in allen Tutorials aktualisiert.	11. Oktober 2017

Änderung	Beschreibung	Datum geändert

Aktualisierung	• Es wurden neue Screenshots für die Ergebnisse der State-Machine-Ausführung hinzugefügt, um Änderungen in der Step Functions Functions-Konsole widerzuspiegeln. Die Lambda-Anweisungen in den folgenden Tutorials wurden neu geschrieben, um die Änderungen in der Lambda-Konsole widerzuspiegeln:
	  • Eine Step Functions Functions-Zustandsmaschine erstellen, die Lambda verwendet
	  • Erstellen eines Auftragsstatus-Pollers
	  • Erstellen eines Zustands-Timers
	  • Behandlung von Fehlerbedingungen mithilfe einer Step Functions Functions-Zustandsmaschine
	• Informationen zum Erstellen von Zustandsautomaten wurden in den folgenden Abschnitten korrigiert und geklärt:
	  • Einen Activity State Machine mithilfe von Step Functions erstellen
		6. Oktober 2017

Aktualisierung	Die IAM-Anweisungen in den folgenden Abschnitten wurden neu geschrieben, um den Änderungen in der IAM-Konsole Rechnung zu tragen: • Erstellen Sie eine IAM-Rolle für Ihren State Machine • Eine Step Functions Functions-Zustandsmaschine erstellen, die Lambda verwendet • Erstellen eines Auftragsstatus-Pollers • Erstellen eines Zustands-Timers • Behandlung von Fehlerbedingungen mithilfe einer Step Functions Functions-Zustandsmaschine • Erstellen einer Step-Functions-API mit API Gateway	5. Oktober 2017

Änderung	Beschreibung	Datum geändert
Aktualisierung	Der Abschnitt Zustandsautomatendaten wurde neu geschrieben.	28. September 2017
Neues Feature	Die Grenzwerte für die Drosselung von API-Aktionen wurden für alle Regionen erhöht, in denen Step Functions verfügbar ist.	18. September 2017
Aktualisierung	• Informationen zum Starten neuer Ausführungen wurden in allen Tutorials korrigiert und präzisiert. • Informationen im Abschnitt Kontingente im Zusammenhang mit Konten wurden korrigiert und präzisiert.	14. September 2017
Aktualisierung	Die folgenden Tutorials wurden neu geschrieben, um die Änderungen in der Lambda-Konsole widerzuspiegeln: • Eine Step Functions Functions-Zustandsmaschine erstellen, die Lambda verwendet • Behandlung von Fehlerbedingungen mithilfe einer Step Functions Functions-Zustandsmaschine • Erstellen eines Auftragsstatus-Pollers	28. August 2017
Neues Feature	Step Functions ist in Europa (London) verfügbar.	23. August 2017
Neues Feature	Die visuellen Workflows von Zustandsautomaten erlauben Ihnen, Diagramme zu vergrößern, zu verkleinern und zu zentrieren.	21. August 2017

Änderung	Beschreibung	Datum geändert
Neues Feature	**Important** Eine Ausführung darf 90 Tage lang keinen Namen einer Ausführung verwenden. Wenn Sie mehrere `StartExecution` Aufrufe mit demselben Namen tätigen, wird die neue Ausführung nicht ausgeführt. Weitere Informationen finden Sie unter dem `name`Anforderungsparameter der `StartExecution` API-Aktion in der AWS Step Functions API-Referenz.	18. August 2017
Aktualisierung	Es wurden Informationen über eine alternative Möglichkeit der Übergabe des Zustandsautomaten-ARN zum Tutorial Erstellen einer Step-Functions-API mit API Gateway hinzugefügt.	17. August 2017
Aktualisierung	Das neue Tutorial Erstellen eines Auftragsstatus-Pollers wurde hinzugefügt.	10. August 2017
Neues Feature	• Step Functions gibt die `ExecutionThrottled` CloudWatch Metrik aus. Weitere Informationen finden Sie unter Step Functions überwachen mit CloudWatch. • Der Abschnitt Kontingente im Zusammenhang mit staatlicher Drosselung wurde hinzugefügt.	3. August 2017
Aktualisierung	Die Anweisungen im Abschnitt Schritt 1: Erstellen einer IAM-Rolle für API Gateway wurden aktualisiert.	18. Juli 2017
Aktualisierung	Informationen im Abschnitt Choice wurden korrigiert und präzisiert.	23. Juni 2017

Änderung	Beschreibung	Datum geändert
Aktualisierung	Den folgenden Tutorials wurden Informationen zur Nutzung von Ressourcen unter anderen AWS Konten hinzugefügt: • Eine Step Functions Functions-Zustandsmaschine erstellen, die Lambda verwendet • Erstellen einer Lambda-Zustandsmaschine für Step Functions mit AWS CloudFormation • Einen Activity State Machine mithilfe von Step Functions erstellen • Behandlung von Fehlerbedingungen mithilfe einer Step Functions Functions-Zustandsmaschine	22. Juni 2017
Aktualisierung	Informationen wurden in den folgenden Abschnitten korrigiert und präzisiert: • Behandlung von Fehlerbedingungen mithilfe einer Step Functions Functions-Zustandsmaschine • Zustände • Fehlerbehandlung in Step Functions	21. Juni 2017
Aktualisierung	Alle Tutorials wurden neu geschrieben, damit sie der Aktualisierung der Step Functions Functions-Konsole entsprechen.	12. Juni 2017
Neues Feature	Step Functions ist im asiatisch-pazifischen Raum (Sydney) verfügbar.	8. Juni 2017
Aktualisierung	Der Abschnitt Amazon States Language wurde umstruktu riert.	7. Juni 2017
Aktualisierung	Informationen im Abschnitt Einen Activity State Machine mithilfe von Step Functions erstellen wurden korrigiert und präzisiert.	6. Juni 2017

Änderung	Beschreibung	Datum geändert
Aktualisierung	Die Code-Beispiele im Abschnitt Beispiele für Zustandsmaschinen mit Retry und Catch wurden korrigiert.	5. Juni 2017
Aktualisierung	Dieses Handbuch wurde anhand von AWS Dokumentationsstandards neu strukturiert.	31. Mai 2017
Aktualisierung	Informationen im Abschnitt Parallel wurden korrigiert und präzisiert.	25. Mai 2017
Aktualisierung	Abschnitte zu Pfaden und Filtern wurden in den Abschnitt Eingabe- und Ausgabeverarbeitung in Step Functions zusammengeführt.	24. Mai 2017
Aktualisierung	Informationen im Abschnitt Step Functions überwachen mit CloudWatch wurden korrigiert und präzisiert.	15. Mai 2017
Aktualisierung	Der `GreeterActivities.java` -Worker-Code im Tutorial Einen Activity State Machine mithilfe von Step Functions erstellen wurde aktualisiert.	9. Mai 2017
Aktualisierung	Ein einführendes Video zum Abschnitt Was ist AWS Step Functions? wurde hinzugefügt.	19. April 2017
Aktualisierung	Informationen in den folgenden Abschnitten wurden korrigiert und präzisiert: • Eine Step Functions Functions-Zustandsmaschine erstellen, die Lambda verwendet • Einen Activity State Machine mithilfe von Step Functions erstellen • Behandlung von Fehlerbedingungen mithilfe einer Step Functions Functions-Zustandsmaschine	19. April 2017

Änderung	Beschreibung	Datum geändert
Aktualisierung	Informationen zu Lambda-Vorlagen wurden zu den Eine Step Functions Functions-Zustandsmaschine erstellen, die Lambda verwendet Behandlung von Fehlerbedingungen mithilfe einer Step Functions Functions-Zustandsmaschine Tutorials hinzugefügt.	6. April 2017
Aktualisierung	Die Beschränkung „Maximale Größe von Eingabe- oder Ergebnisdaten“ wurde zu „Maximale Größe von Eingabe- oder Ergebnisdaten für eine Aufgabe, einen Zustand oder eine Ausführung“ (32.768 Zeichen) geändert. Weitere Informationen finden Sie unter Kontingente im Zusammenhang mit der Ausführung von Aufgaben.	31. März 2017
Neues Feature	• Step Functions unterstützt die Ausführung von Zustandsmaschinen, indem Step Functions als Amazon CloudWatch Events-Ziele festgelegt werden.	21. März 2017
Neues Feature	• Step Functions ermöglicht die Fehlerbehandlung von Lambda-Funktionen als bevorzugte Methode zur Fehlerbehandlung. • Das Tutorial Behandlung von Fehlerbedingungen mithilfe einer Step Functions Functions-Zustandsmaschine und der Abschnitt Fehlerbehandlung in Step Functions wurden aktualisiert.	16. März 2017
Neues Feature	Step Functions ist in Europa (Frankfurt) verfügbar.	7. März 2017

Änderung	Beschreibung	Datum geändert
Aktualisierung	Die Themen im Inhaltsverzeichnis wurden umstrukturiert und die folgenden Tutorials wurden aktualisiert: • Eine Step Functions Functions-Zustandsmaschine erstellen, die Lambda verwendet • Einen Activity State Machine mithilfe von Step Functions erstellen • Behandlung von Fehlerbedingungen mithilfe einer Step Functions Functions-Zustandsmaschine	23. Februar 2017
Neues Feature	• Die Seite „State Machines“ der Step Functions Functions -Konsole enthält die Schaltflächen „In neue Version kopieren“ und „Löschen“. • Die Screenshots wurden aktualisiert, um die Änderungen an der Konsole widerzuspiegeln.	23. Februar 2017
Neues Feature	• Step Functions unterstützt die Erstellung von APIs mithilfe von API Gateway. • Das Tutorial Erstellen einer Step-Functions-API mit API Gateway wurde hinzugefügt.	14. Februar 2017
Neues Feature	• Step Functions unterstützt die Integration mit AWS CloudFormation. • Das Tutorial Erstellen einer Lambda-Zustandsmaschine für Step Functions mit AWS CloudFormation wurde hinzugefügt.	10. Februar 2017
Aktualisierung	Das derzeitige Verhalten der Felder `ResultPath` und `OutputPath` in Bezug auf `Parallel` Zustände wurde präzisiert.	6. Februar 2017
Aktualisierung	• Einschränkungen bei der Benennung von Zustandsa utomaten in Tutorials wurden präzisiert. • Einige Code-Beispiele wurden korrigiert.	5. Januar 2017

Änderung	Beschreibung	Datum geändert
Aktualisierung	Die Beispiele für Lambda-Funktionen wurden aktualisiert, um das neueste Programmiermodell zu verwenden.	9. Dezember 2016
Erstversion	Erste Version von. AWS Step Functions	1. Dezember 2016

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.