aws

Guide du développeur

AWS Step Functions

AWS Step Functions: Guide du développeur

Table of Contents

Qu'est-ce que c'est AWS Step Functions ?

AWS Step Functions est un service de flux de travail visuel qui vous aide à créer des applications distribuées, à automatiser des processus, à orchestrer des microservices et à créer des pipelines de données et d'apprentissage automatique (ML).

Dans la console graphique de Step Functions, vous pouvez voir le flux de travail de votre application sous la forme d'une série d'étapes pilotées par des événements.

Step Functions est basé sur des machines à états et des tâches. Dans Step Functions, les machines à états sont appelées flux de travail, qui sont une série d'étapes pilotées par des événements. Chaque étape d'un flux de travail est appelée état. Par exemple, un état de tâche représente une unité de travail exécutée par un autre AWS service, comme l'appel d'un autre service Service AWS ou d'une API.

Grâce aux commandes intégrées de Step Functions, vous pouvez examiner l'état de chaque étape de votre flux de travail pour vous assurer que votre application s'exécute dans l'ordre et comme prévu. Selon votre cas d'utilisation, vous pouvez demander à Step Functions d'appeler AWS des services, tels que Lambda, pour effectuer des tâches. Vous pouvez créer des flux de travail qui traitent et publient des modèles d'apprentissage automatique. Vous pouvez utiliser les AWS services de contrôle Step Functions, par exemple pour créer des flux de travail d'extraction, de transformation et de chargement (ETL). AWS Glue Vous pouvez également créer des flux de travail automatisés à long terme pour les applications qui nécessitent une interaction humaine.

> **Tip**
>
> Pour apprendre à utiliser Step Functions, suivez les modules interactifs de l'AWS Step Functions atelier ou lisez la section Getting Started de ce guide pour créer un flux de travail pour les demandes de carte de crédit.

Rubriques

- Régions prises en charge
- Est-ce la première fois que vous utilisez Step Functions ?

AWS SDK et intégrations optimisées

Pour appeler d'autres AWS services, vous pouvez utiliser les intégrations du AWS SDK de Step Functions, ou vous pouvez utiliser l'une des intégrations optimisées de Step Functions.

- Les intégrations du AWS SDK vous permettent d'appeler n'importe lequel des plus de deux cents AWS services directement depuis votre machine d'état, ce qui vous donne accès à plus de neuf mille actions d'API.
- Les intégrations optimisées de Step Functions ont été personnalisées pour simplifier l'utilisation dans vos machines d'état.

Flux de travail standard et express

Step Functions propose deux types de flux de travail. Les flux de travail standard ne sont exécutés qu'une seule fois et peuvent durer jusqu'à un an. Cela signifie que chaque étape d'un flux de travail standard ne sera exécutée qu'une seule fois. Les flux de travail Express, quant at-least-once à eux, peuvent être exécutés pendant cinq minutes au maximum. Cela signifie qu'une ou plusieurs étapes d'un flux de travail express peuvent potentiellement être exécutées plusieurs fois, chaque étape du flux de travail étant exécutée au moins une fois.

Les exécutions sont des instances dans lesquelles vous exécutez votre flux de travail pour effectuer des tâches. Les flux de travail standard sont idéaux pour les flux de travail auditables de longue durée, car ils affichent l'historique d'exécution et le débogage visuel. Les flux de travail Express sont idéaux pour les high-event-rate charges de travail, telles que le traitement des données en streaming et l'ingestion de données IoT.

Spécifications des flux de travail standard

- Taux d'exécution de 2 000 par seconde
- Taux de transition entre États de 4 000 par seconde
- Tarification en fonction de la transition entre États
- Afficher l'historique d'exécution et le débogage visuel
- Support de tous les modèles et intégrations de services

Spécifications des flux de travail Express

- Taux d'exécution de 100 000 par seconde
- Taux de transition entre États presque illimité
- Tarification en fonction du nombre et de la durée des exécutions
- Envoyer l'historique des exécutions à Amazon CloudWatch
- Afficher l'historique d'exécution et le débogage visuel en fonction du niveau de journalisation activé
- Support de toutes les intégrations de services et de la plupart des modèles

Pour plus d'informations sur les flux de travail Standard et Express, y compris la tarification de Step Functions, consultez les pages suivantes :

- Flux de travail standard ou express
- Tarification AWS Step Functions

Cas d’utilisation

Step Functions gère les composants et la logique de votre application, afin que vous puissiez écrire moins de code et vous concentrer sur la création et la mise à jour rapides de votre application. Cette section décrit les cas d'utilisation typiques de l'utilisation de Step Functions.

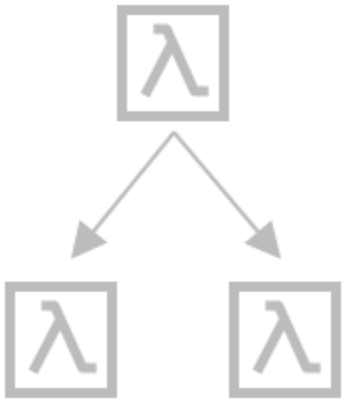
Cas d'utilisation #1 : Orchestration de fonctions

Vous créez un flux de travail qui exécute un groupe de fonctions Lambda (étapes) dans un ordre spécifique. La sortie d'une fonction Lambda passe à l'entrée de la fonction Lambda suivante. La dernière étape de votre flux de travail donne un résultat. Avec Step Functions, vous pouvez voir comment chaque étape de votre flux de travail interagit les unes avec les autres, afin de vous assurer que chaque étape exécute la fonction prévue.

Pour un didacticiel qui explique comment créer une machine à états avec un groupe de fonctions, consultez ce qui suit :

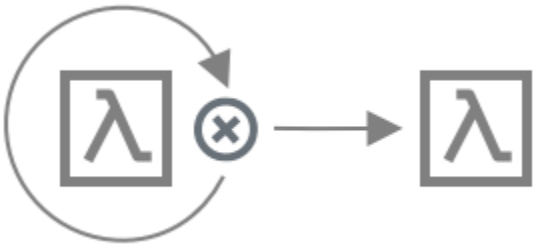
- Démarrer avec AWS Step Functions

Cas d'utilisation #2 : Branching



Un client demande une augmentation de sa limite de crédit. À l'aide d'un `Choice` état, vous pouvez demander à Step Functions de prendre des décisions en fonction des données saisies par `Choice` l'état. Si la demande dépasse la limite de crédit préapprouvée de votre client, vous pouvez demander à Step Functions d'envoyer la demande de votre client à un responsable pour approbation. Si la demande est inférieure à la limite de crédit préapprouvée par votre client, vous pouvez demander à Step Functions d'approuver automatiquement la demande.

Cas d'utilisation #3 : gestion des erreurs



`Retry`

Dans ce cas d'utilisation, un client demande un nom d'utilisateur. La première fois, la demande de votre client échoue. À l'aide d'un `Retry` relevé, vous pouvez demander à Step Functions de réessayer la demande de votre client. La deuxième fois, la demande de votre client est acceptée.

`Catch`

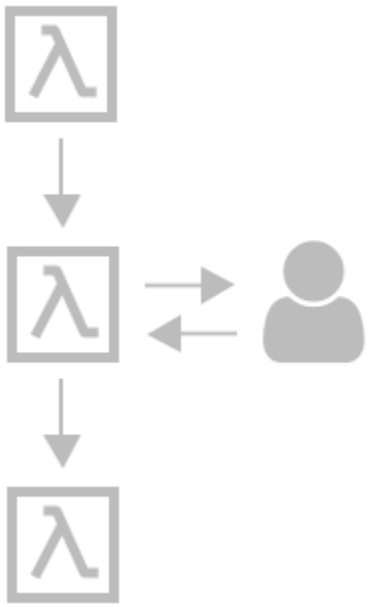
Dans un cas d'utilisation similaire, un client demande un nom d'utilisateur non disponible. À l'aide d'une `Catch` instruction, Step Functions vous suggère un nom d'utilisateur disponible. Si votre client

utilise le nom d'utilisateur disponible, vous pouvez demander à Step Functions de passer à l'étape suivante de votre flux de travail, qui consiste à envoyer un e-mail de confirmation. Si votre client n'utilise pas le nom d'utilisateur disponible, Step Functions passe à une autre étape de votre flux de travail, qui consiste à recommencer le processus d'inscription.

Pour des exemples `Retry` et des `Catch` déclarations plus détaillés, consultez les rubriques suivantes :

- Gestion des erreurs dans Step Functions

Cas d'utilisation #4 : Un humain dans la boucle

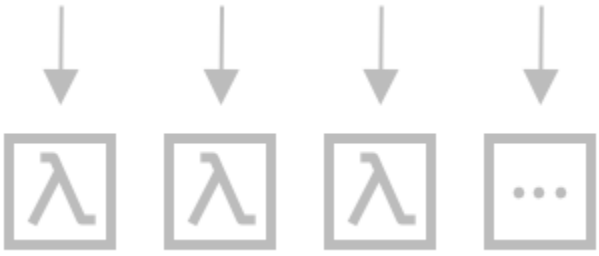


À l'aide d'une application bancaire, l'un de vos clients envoie de l'argent à un ami. Votre client attend un e-mail de confirmation. À l'aide d'un rappel et d'un jeton de tâche, Step Functions demande à Lambda d'envoyer l'argent à votre client et de faire rapport lorsque l'ami de votre client l'aura reçu. Une fois que Lambda aura indiqué que l'ami de votre client a reçu l'argent, vous pouvez demander à Step Functions de passer à l'étape suivante de votre flux de travail, qui consiste à envoyer un e-mail de confirmation à votre client.

Pour voir un exemple de projet qui montre un rappel avec un jeton de tâche, consultez ce qui suit :

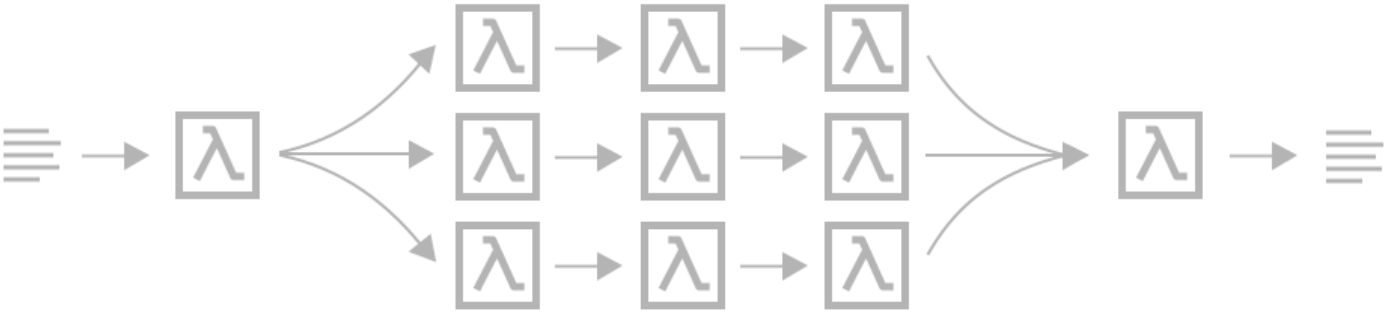
- Exemple de modèle de rappel (Amazon SQS, Amazon SNS, Lambda)

Cas d'utilisation #5 : traitement parallèle



Un client convertit un fichier vidéo en cinq résolutions d'affichage différentes, afin que les spectateurs puissent regarder la vidéo sur plusieurs appareils. À l'aide d'un `Parallel` état, Step Functions saisit le fichier vidéo, afin que Lambda puisse le traiter dans les cinq résolutions d'affichage en même temps.

Cas d'utilisation #6 : parallélisme dynamique



Un client commande trois articles, et vous devez préparer chaque article pour la livraison. Vous vérifiez la disponibilité de chaque article, vous rassemblez chaque article, puis vous emballez chaque article pour livraison. À l'aide d'un `Map` état, Step Functions permet à Lambda de traiter chacun des articles de votre client en parallèle. Une fois que tous les articles de votre client sont emballés pour la livraison, Step Functions passe à l'étape suivante de votre flux de travail, qui consiste à envoyer à votre client un e-mail de confirmation contenant des informations de suivi.

Pour voir un exemple de projet illustrant le parallélisme dynamique à l'aide d'un `Map` état, consultez ce qui suit :

- Traitement dynamique des données avec un état cartographique

Intégrations de service

Step Functions s'intègre à de nombreux AWS services. Pour associer Step Functions à ces services, utilisez les modèles d'intégration de services suivants :

Demander une réponse (par défaut)

- Appelez un service et laissez Step Functions passer à l'état suivant après avoir reçu une réponse HTTP.

Exécuter une tâche (.sync)

- Appelez un service et demandez à Step Functions d'attendre qu'une tâche soit terminée.

Attendez un rappel avec un jeton de tâche (. waitForTaskJeton)

- Appelez un service avec un jeton de tâche et demandez à Step Functions d'attendre que le jeton de tâche revienne avec un rappel.

Le tableau ci-dessous présente les intégrations de services disponibles et les modèles d'intégration de services pour Step Functions.

Les flux de travail standard et les flux de travail express prennent en charge les mêmes intégrations, mais pas les mêmes modèles d'intégration.

- La prise en charge des modèles d'intégration optimisés est différente pour chaque intégration.
- Express Workflows ne prend pas en charge Run a Job (.sync) ou Wait for Callback (. waitForTaskJeton).
- Pour plus d’informations, consultez Flux de travail standard ou express.

Standard Workflows

Intégrations de services prises en charge

	Service	Réponse à la requête	Exécuter une tâche (`.sync`)	Attendre le rappel (`.waitForTaskToken`)
Intégrations optimisées	Amazon API Gateway	✓		✓
	Amazon Athena	✓	✓	
	AWS Batch	✓	✓	
	Amazon Bedrock	✓	✓	✓
	AWS CodeBuild	✓	✓	
	Amazon DynamoDB	✓		
	Amazon ECS/Fargate	✓	✓	✓
	Amazon EKS	✓	✓	✓
	Amazon EMR	✓	✓	
	Amazon EMR on EKS	✓	✓	
	Amazon EMR Serverless	✓	✓	
	Amazon EventBridge	✓		✓
	AWS Glue	✓	✓	
	AWS Glue DataBrew	✓	✓	
	AWS Lambda	✓		✓
	AWS Elemental MediaConvert	✓	✓	
	Amazon SageMaker	✓	✓	

	Service	Réponse à la requête	Exécuter une tâche (`.sync`)	Attendre le rappel (`.waitForTaskToken`)
	Amazon SNS	✓		✓
	Amazon SQS	✓		✓
	AWS Step Functions	✓	✓	✓
AWS Intégrations du SDK	Plus de deux cents	✓		✓

Express Workflows

Intégrations de services prises en charge

	Service	Réponse à la requête	Exécuter une tâche (`.sync`)	Attendre le rappel (`.waitForTaskToken`)
Intégrations optimisées	Amazon API Gateway	✓		
	Amazon Athena	✓		
	AWS Batch	✓		
	Amazon Bedrock	✓		
	AWS CodeBuild	✓		
	Amazon DynamoDB	✓		
	Amazon ECS/Fargate	✓		
	Amazon EKS	✓		

	Service	Réponse à la requête	Exécuter une tâche (`.sync`)	Attendre le rappel (`.waitForTaskToken`)
	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 Intégrations du SDK	Plus de deux cents	✓		

Régions prises en charge

La plupart des AWS régions prennent en charge Step Functions. Pour une liste complète des AWS régions dans lesquelles Step Functions est disponible, consultez le tableau des AWS régions.

Est-ce la première fois que vous utilisez Step Functions ?

Si c'est la première fois que vous utilisez Step Functions, les rubriques suivantes vous aideront à comprendre les différents aspects de l'utilisation de Step Functions, notamment la façon dont Step Functions se combine à d'autres AWS services :

- Tutoriels pour Step Functions
- Exemples de projets pour Step Functions
- AWS Step Functions SDK de science des données pour Python

Conditions préalables pour démarrer avec AWS Step Functions

Avant de commencer AWS Step Functions pour la première fois, remplissez les conditions requises répertoriées sur cette page.

Rubriques

Étape 1 : Inscrivez-vous pour un utilisateur Compte AWS et un utilisateur IAM

Pour accéder à AWS un service, vous devez d'abord créer un Compte AWS. Vous pouvez utiliser votre Compte AWS pour consulter vos rapports d'activité et d'utilisation et pour gérer l'authentification et l'accès. Vous n'êtes facturé que pour les produits et services que vous utilisez, et vous pouvez commencer à les utiliser AWS gratuitement. Pour plus d’informations, consultez la page sur l’offre gratuite AWS.

Pour éviter d'utiliser vos actions Utilisateur racine d'un compte AWS pour Step Functions, il est recommandé de créer un utilisateur IAM pour chaque personne ayant besoin d'un accès administratif à Step Functions.

Si vous avez déjà un AWS compte, passez au prérequis suivant.

Inscrivez-vous pour un Compte AWS

Si vous n'en avez pas Compte AWS, procédez comme suit pour en créer un.

Pour vous inscrire à un Compte AWS

1. Ouvrez https://portal.aws.amazon.com/billing/signup.
2. Suivez les instructions en ligne.

   Dans le cadre de la procédure d‘inscription, vous recevrez un appel téléphonique et vous saisirez un code de vérification en utilisant le clavier numérique du téléphone.

Lorsque vous vous inscrivez à un Compte AWS, un Utilisateur racine d'un compte AWSest créé. Par défaut, seul l'utilisateur racine a accès à l'ensemble des Services AWS et des ressources de ce compte. La meilleure pratique en matière de sécurité consiste à attribuer un accès administratif à un utilisateur et à n'utiliser que l'utilisateur root pour effectuer les tâches nécessitant un accès utilisateur root.

AWS vous envoie un e-mail de confirmation une fois le processus d'inscription terminé. Vous pouvez afficher l'activité en cours de votre compte et gérer votre compte à tout moment en accédant à https://aws.amazon.com/ et en choisissant Mon compte.

Création d'un utilisateur doté d'un accès administratif

Une fois que vous vous êtes inscrit à un utilisateur administratif Compte AWS, que vous Utilisateur racine d'un compte AWS l'avez sécurisé AWS IAM Identity Center, que vous l'avez activé et que vous en avez créé un, afin de ne pas utiliser l'utilisateur root pour les tâches quotidiennes.

Sécurisez votre Utilisateur racine d'un compte AWS

1. Connectez-vous en AWS Management Consoletant que propriétaire du compte en choisissant Utilisateur root et en saisissant votre adresse Compte AWS e-mail. Sur la page suivante, saisissez votre mot de passe.

   Pour obtenir de l'aide pour vous connecter en utilisant l'utilisateur racine, consultez Connexion en tant qu'utilisateur racine dans le Guide de l'utilisateur Connexion à AWS .
2. Activez l'authentification multifactorielle (MFA) pour votre utilisateur racine.

   Pour obtenir des instructions, consultez la section Activer un périphérique MFA virtuel pour votre utilisateur Compte AWS root (console) dans le guide de l'utilisateur IAM.

Création d'un utilisateur doté d'un accès administratif

1. Activez IAM Identity Center.

   Pour obtenir des instructions, consultez Activation d' AWS IAM Identity Center dans le Guide de l'utilisateur AWS IAM Identity Center .
2. Dans IAM Identity Center, accordez un accès administratif à un utilisateur.

Pour un didacticiel sur l'utilisation du Répertoire IAM Identity Center comme source d'identité, voir Configurer l'accès utilisateur par défaut Répertoire IAM Identity Center dans le Guide de AWS IAM Identity Center l'utilisateur.

Connectez-vous en tant qu'utilisateur disposant d'un accès administratif

- Pour vous connecter avec votre utilisateur IAM Identity Center, utilisez l‘URL de connexion qui a été envoyée à votre adresse e-mail lorsque vous avez créé l‘utilisateur IAM Identity Center.

  Pour obtenir de l'aide pour vous connecter en utilisant un utilisateur d'IAM Identity Center, consultez la section Connexion au portail AWS d'accès dans le guide de l'Connexion à AWS utilisateur.

Attribuer l'accès à des utilisateurs supplémentaires

1. Dans IAM Identity Center, créez un ensemble d'autorisations conforme aux meilleures pratiques en matière d'application des autorisations du moindre privilège.

   Pour obtenir des instructions, voir Création d'un ensemble d'autorisations dans le guide de AWS IAM Identity Center l'utilisateur.
2. Affectez des utilisateurs à un groupe, puis attribuez un accès d'authentification unique au groupe.

   Pour obtenir des instructions, consultez la section Ajouter des groupes dans le guide de AWS IAM Identity Center l'utilisateur.

Étape 2 : Accorder un accès par programmation

Les utilisateurs ont besoin d'un accès programmatique s'ils souhaitent interagir avec AWS l'extérieur du AWS Management Console. La manière d'accorder un accès programmatique dépend du type d'utilisateur qui y accède AWS.

Pour accorder aux utilisateurs un accès programmatique, choisissez l’une des options suivantes.

Quel utilisateur a besoin d'un accès programmatique ?	Pour	Par
Identité de la main-d'œuvre (Utilisateurs gérés dans IAM Identity Center)	Utilisez des informations d'identification temporaires pour signer les demandes programmatiques adressées aux AWS CLI AWS SDK ou AWS aux API.	Suivez les instructions de l'interface que vous souhaitez utiliser. • Pour le AWS CLI, voir Configuration du AWS CLI à utiliser AWS IAM Identity Center dans le guide de AWS Command Line Interface l'utilisateur. • Pour les AWS SDK, les outils et les AWS API, consultez la section Authentification IAM Identity Center dans le Guide de référence AWS des SDK et des outils.
IAM	Utilisez des informations d'identification temporaires pour signer les demandes programmatiques adressées aux AWS CLI AWS SDK ou AWS aux API.	Suivez les instructions de la section Utilisation d'informations d'identification temporaires avec AWS les ressources du Guide de l'utilisateur IAM.
IAM	(Non recommandé) Utilisez des informations d'identification à long terme pour signer les AWS CLI demandes programmatiques adressées aux AWS SDK ou AWS aux API.	Suivez les instructions de l'interface que vous souhaitez utiliser. • Pour le AWS CLI, voir Authentification à l'aide des informations d'identification utilisateur IAM dans le

Quel utilisateur a besoin d'un accès programmatique ?	Pour	Par
		Guide de l'AWS Command Line Interface utilisateur. • Pour les AWS SDK et les outils, voir Authentifier à l'aide d'informations d'identification à long terme dans le Guide de AWS référence des SDK et des outils. • Pour les AWS API, consultez la section Gestion des clés d'accès pour les utilisateurs IAM dans le guide de l'utilisateur IAM.

Démarrer avec AWS Step Functions

Bienvenue sur Step FunctionsCommencersérie de tutoriels.

Step Functions est un service d'orchestration sans serveur qui vous permet de définir un flux de travail d'application comme une série d'étapes métier essentielles. Chaque étape du flux de travail est appeléeétat. Vous utilisez le plus souvent des états, tels queÉtat de la tâche,Choice,Parallèle, etMap, pour définir vos flux de travail. À l'intérieurTaskÉtats, vous pouvez utiliserAWSIntégrations de SDK prises en charge par Step Functions et orchestrant plusieursServices AWSdans vos flux de travail.

Rubriques

- Concepts clés
- Tutoriels de cette série
- Tutoriel 1 : Création du prototype de votre machine à états
- Tutoriel 2 : Définir la première intégration de service à l'aide d'une fonction Lambda
- Tutoriel 3 : Implémenter une condition if-else dans votre flux de travail
- Tutoriel 4 : Définir plusieurs tâches à effectuer en parallèle
- Tutoriel 5 : itérer simultanément sur une collection d'éléments
- Tutoriel 6 : Enregistrer le flux de travail et exécuter la machine à états
- Tutoriel 7 : Configuration des entrées et des sorties
- Tutoriel 8 : Déboguer les erreurs dans la console

Concepts clés

Cette section vous présente les concepts importants de Step Functions. Avant de commencer, passez en revue les concepts clés suivants.

Terme	Description
Flux de travail	Décrit une séquence d'étapes et correspond souvent à un processus métier.

Terme	Description
Studio de flux de travail	Un concepteur visuel de flux de travail qui vous aide à prototyper et à créer des flux de travail plus rapidement. Pour plus d'informations, veuillez consulter AWS Step Functions Studio de flux de travail.
States	Étapes individuelles de votre machine à états, qui exécutent diverses fonctions dans la machine à états. Pour plus d'informations, veuillez consulter States.
Machines d'état	Un flux de travail défini à l'aide d'un texte JSON représentant les différents états ou étapes du flux de travail ainsi que des champs, tels que`StartAt`,`TimeoutSeconds` , et`Version`. Pour plus d'informations, veuillez consulter Structure de la machine d'État.
Amazon States Language	Langage structuré basé sur JSON qui vous permet de définir votre machine d'état. Il s'agit d'une collection deétatsqui peut fonctionner (`Task`état), déterminez les états vers lesquels passer ensuite (`Choice`état), et arrête une exécution avec une erreur (`Fail`état). Pour plus d'informations, veuillez consulter Amazon States Language.
Configuration d'entrée et de sortie	Les états individuels d'un flux de travail reçoivent des données JSON en entrée et transmettent généralement les données JSON en sortie à l'état suivant. Step Functions fournit plusieurs filtres pour contrôler le flux de données d'entrée et de sortie entre les états. Pour plus d'informations, veuillez consulter Traitement des entrées et des sorties dans Step Functions.
Intégration de service	Step Functions s'intègre directement àServices AWS, vous permettant d'appeler les actions d'API de chaque service depuis votre flux de travail. Pour plus d'informations, veuillez consulter Utilisation AWS Step Functions avec d'autres services.

Terme	Description
Type d'intégration de services	Step Functions propose les types d'intégration de services suivants : • Intégrations optimisées— Personnalisé par Step Functions pour fournir des fonctionnalités spéciales pour un flux de travail. Par exempleLambda Invokeconvertira sa sortie d'API d'une chaîne JSON échappée en un objet JSON. • AWSIntégrations du SDK— Se comporte exactement comme un appel d'API standard à l'aide duAWSSDK. Vous pouvez appeler n'importe lequel des plus de deux centsServices AWSdirectement depuis votre machine d'état et accédez à plus de neuf mille actions d'API. Pour plus d'informations, veuillez consulter Utilisation AWS Step Functions avec d'autres services.
Schéma d'intégration des services	Pour appeler un service intégréService AWSdans votre flux de travail, vous utilisez l'un des modèles d'intégration de services suivants fournis par Step Functions : • Demander une réponse (par défaut)— Appelez un service et laissez Step Functions passer à l'état suivant après avoir reçu une réponse HTTP. • Exécuter une tâche (.sync)— Appelez un service et demandez à Step Functions d'attendre qu'une tâche soit terminée. • Attendez un rappel avec un jeton de tâche (.waitForTaskJeton)— Appelez un service avec un jeton de tâche et demandez à Step Functions d'attendre que le jeton de tâche revienne avec un rappel.
Exécution	Les exécutions par State Machine sont des instances dans lesquelles vous exécutez votre flux de travail pour effectuer des tâches. Pour plus d'informations, veuillez consulter Exécutions dans Step Functions.

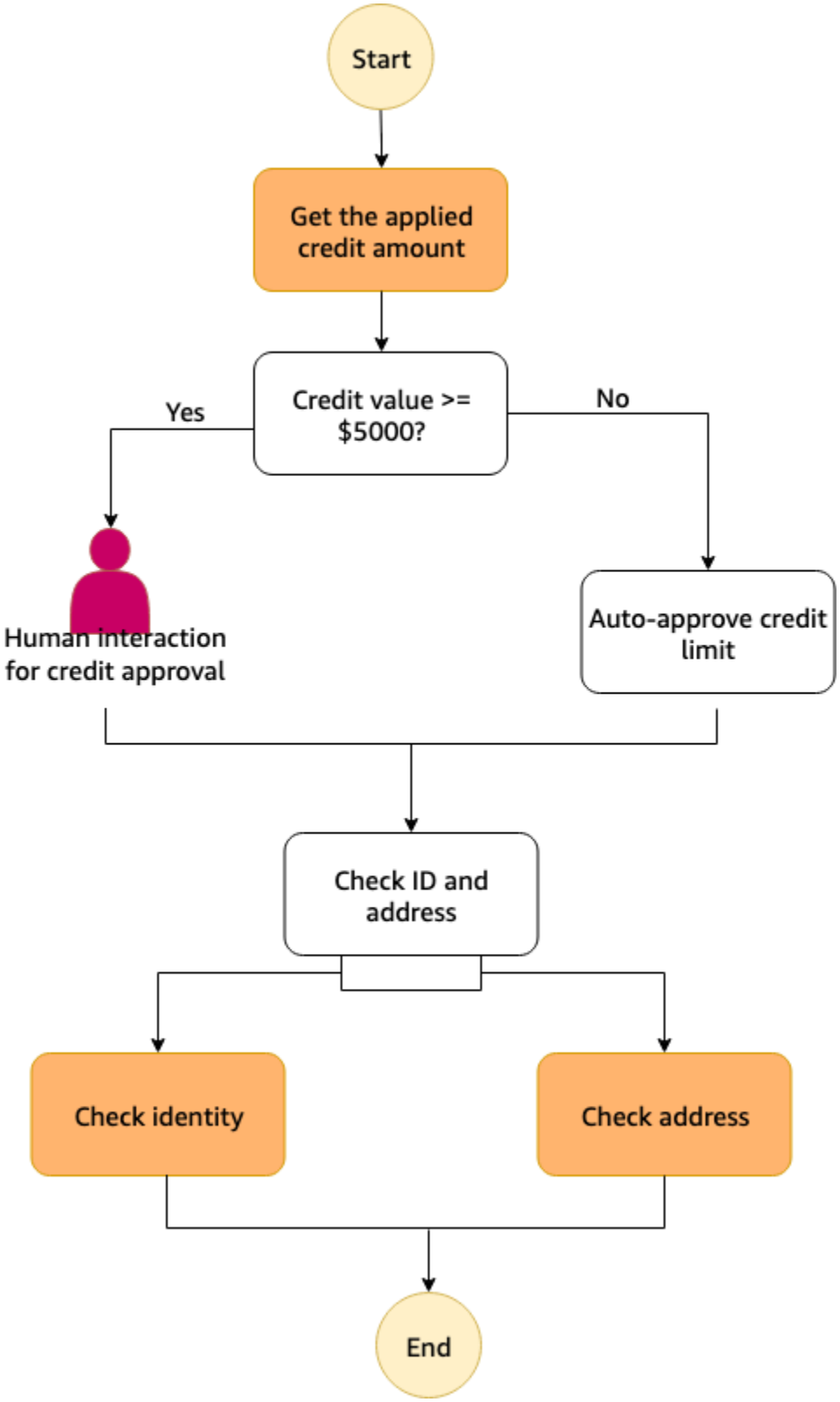
Tutoriels de cette série

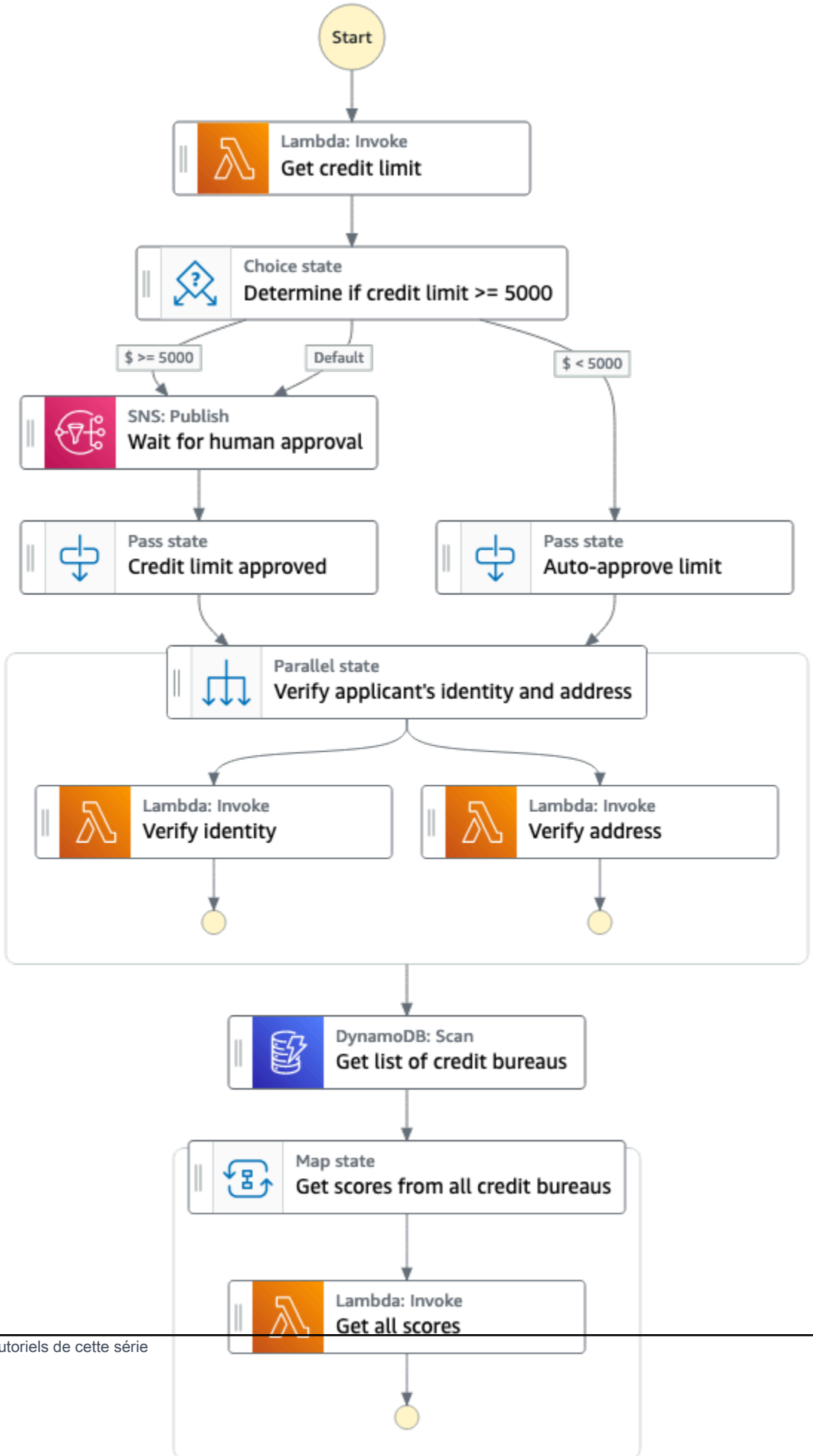
LeCommencerles didacticiels de ce chapitre vous expliquent comment créer un flux de travail de base pour le traitement des demandes de carte de crédit. Dans ces didacticiels, vous allez apprendre

à utiliser des états couramment utilisés dans Step Functions. Vous allez intégrer votre flux de travail à d'autresServices AWS, tels queAWS Lambdaet Amazon Simple Notification Service. Après avoir terminé ces didacticiels, vous disposerez d'un flux de travail simple qui simule le traitement d'une demande de carte de crédit.

> **Note**
>
> Alors que ceux-ciCommencerles didacticiels décrivent un flux de travail de demande de carte de crédit. Vous pouvez utiliser Step Functions pour créer plusieurs types de flux de travail. Par exemple, vous pouvez créer des flux de travail pour le traitement des données, l'automatisation informatique, l'apprentissage automatique, le traitement multimédia ou le traitement des commandes.

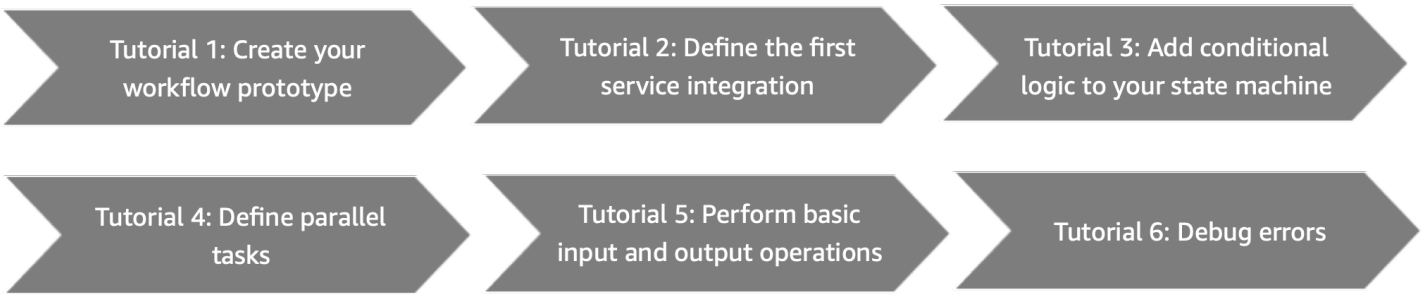
Les images suivantes représentent un flux de travail de demande de carte de crédit et son apparence lorsqu'il est orchestré à l'aide de Step Functions. Chaque étape de l'organigramme est représentée par un état dans le flux de travail Step Functions.


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


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

> **Note**
>
> Nous vous recommandons de suivre tous ces didacticiels dans l'ordre. En suivant les didacticiels complets, vous apprendrez à utiliser les concepts et les fonctionnalités essentiels à l'utilisation de Step Functions dans les flux de production.

La feuille de route suivante indique les étapes que vous allez suivre pour créer le flux de travail de traitement des cartes de crédit à l'aide du Workflow Studio de Step Functions. Ces étapes sont présentées sous la forme d'une série de didacticiels contenant des instructions sur la façon de réaliser cette étape.



Avant de commencer, assurez-vous de remplirprérequis.

Tutoriel 1 : Création du prototype de votre machine à états

Dans ce didacticiel, vous allez créer le prototype de votre flux de traitement des cartes de crédit en utilisantStudio de flux de travail de Step Functions. Vous choisirez les actions et les états d'API requis dans leActionsetFluxonglets respectivement, et utilisez la fonction glisser-déposer de Workflow Studio pour créer le prototype de flux de travail. Dans les didacticiels suivants, vous apprendrez à configurer leServices AWSet les états des fonctions d'étape que vous utiliserez dans ce flux de travail.

Pour créer le prototype de machine à états

1. Ouvrez leConsole Step Functionset choisissezCréer une machine à états.
2. Dans leChoisissez un modèleboîte de dialogue, sélectionnezVide.
3. Choisissez Select (Sélectionner). Cela ouvre Workflow Studio dansMode de conception.

4. Dans Workflow Studio, depuis leActionsonglet, faites glisser unAWS LambdaInvoquerAction de l'API et déposez-la à l'état vide étiquetéFaites glisser le premier état ici. Vous pouvez le configurer comme suit :

   - Dans leConfigurationonglet, pourNom de l'État, entrez**`Get credit limit`**.

5. À partir duFluxtablez, glissez et déposez unChoixétat en dessous duObtenez une limite de créditétat. Renommez leChoixÉtat à**`Credit applied >= 5000?`**.

6. Faites glisser et déposez les états suivants en tant que branches duCrédit appliqué **>=** 5000 ?état.

   a. Amazon SNS Publish— À partir duActionstablez, faites glisser et déposez leAmazon SNS PublishAction de l'API. Renommez cet état**`Wait for human approval`**.

   b. PasseÉtat — À partir duFluxtablez, faites glisser et déposez lePasseétat. Renommez cette branche**`Auto-approve limit`**.

7. Faites glisser et déposez unPasseétat en dessous duAttendre l'approbation humaineétat. Renommez ceciPasseÉtat à**`Credit limit approved`**.

8. Faites glisser et déposez unParallèleÉtat après leChoixénoncez comme suit :

   a. Déposez leParallèleÉtat après leLimite de crédit approuvéeétat.

   b. Renommez leParallèleÉtat à**`Verify applicant's identity and address`**.

   c. Dans le cadre des deux branches duParallèleétat, glisser-déposer deuxAWS LambdaInvoquerActions de l'API.

   d. Renommez ces états en**`Verify identity`**et**`Verify address`**respectivement.

   e. Choisissez leLimite d'approbation automatiqueÉtat et pourÉtat suivant, sélectionnezVérifier l'identité et l'adresse du demandeur.

9. Faites glisser unScan DynamoDBindiquez-le et déposez-le en dessous duVérifier l'identité et l'adresse du demandeurétat. Renommez leScan DynamoDBÉtat à**`Get list of credit bureaus`**.

10. Faites glisser et déposez unCarteÉtat après leObtenez la liste des bureaux de créditétat. Configurez leCarteénoncez comme suit :

    a. Renommez-le en**`Get scores from all credit bureaus`**.

    b. PourMode de traitement, conservez la sélection par défautEn ligne.

    c. Glissez et déposez unAWS LambdaInvoquerAction de l'API vers l'état vide étiquetéDéposez l'état ici.

d. Renommez leAWS LambdaInvoquerÉtat à**`Get all scores`**.

11. Gardez cette fenêtre ouverte et passez au didacticiel suivant pour d'autres actions.

Étapes suivantes

Dans le didacticiel suivant, vous apprendrez comment intégrer la fonction Lambda utilisée parObtenez une limite de créditétat.

Tutoriel 2 : Définir la première intégration de service à l'aide d'une fonction Lambda

Dans ce didacticiel, vous apprendrez à définir la première intégration de services pour votre flux de travail. Vous utilisez l'`Task`état nommé Get credit limit pour appeler une fonction Lambda. Au sein `Task` des états, vous pouvez utiliser les intégrations du AWS SDK prises en charge par Step Functions.

Pour définir la première intégration de services pour votre flux de travail, créez d'abord une fonction Lambda. Mettez ensuite à jour votre flux de travail pour spécifier l'intégration du service avec la fonction Lambda. La fonction Lambda utilisée dans ce didacticiel renvoie un entier généré de manière aléatoire représentant la limite de crédit demandée par un candidat.

Rubriques

Étape 1 : Création et test de la fonction Lambda

Vous pouvez écrire le code de la fonction dans l'éditeur AWS Management Console ou dans votre éditeur préféré. Dans les étapes suivantes, vous allez créer une fonction Lambda Node.js intitulée. `RandomNumberforCredit`

> **Important**
>
> Assurez-vous que le prototype de flux de travail que vous avez créé dans le didacticiel 1 est Région AWS identique à la fonction Lambda que vous allez créer dans ce didacticiel.

1. Dans un nouvel onglet ou une nouvelle fenêtre, ouvrez la console Lambda et créez une fonction Lambda Node.js 16.x intitulée. **`RandomNumberforCredit`** Pour plus d'informations sur la création d'une fonction Lambda à l'aide de la console, voir Création d'une fonction Lambda dans la console dans le Guide du développeur.AWS Lambda
2. Sur la RandomNumberforCreditpage, choisissez index.mjs et remplacez le code existant dans la zone Source du code par le code suivant.

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

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

};
```

3. Dans la section Vue d'ensemble des fonctions, copiez le nom de ressource Amazon de la fonction Lambda et enregistrez-le dans un fichier texte. Vous aurez besoin de la fonction ARN pour spécifier l'intégration du service pour l'état Get credit limit. Voici un exemple d'ARN :

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

4. Choisissez Deploy, puis sélectionnez Test pour déployer les modifications et voir le résultat de la fonction Lambda.

Étape 2 : Mettre à jour le flux de travail — configurer l'état Obtenir une limite de crédit

Dans la console Step Functions, vous allez mettre à jour votre flux de travail pour spécifier l'intégration du service avec la fonction `RandomNumberforCredit` Lambda que vous avez créée à l'étape 1.

1. Ouvrez la fenêtre de console Step Functions contenant le prototype de flux de travail que vous avez créé dans le didacticiel 1.

2. Choisissez l'état Obtenir la limite de crédit, puis dans l'onglet Configuration, procédez comme suit :

   a. Pour le type d'intégration, conservez la sélection par défaut Optimized.

      À l'aide de Step Functions, vous pouvez les intégrer à d'autres Services AWS et les orchestrer dans vos flux de travail. Pour plus d'informations sur les intégrations de services et leurs types, consultezUtilisation AWS Step Functions avec d'autres services.

   b. Pour Nom de la fonction, choisissez la fonction RandomNumberforCreditLambda dans la liste déroulante.

   c. Conservez les sélections par défaut pour le reste des éléments.

3. Gardez cette fenêtre ouverte et passez au didacticiel suivant pour d'autres actions.

> **Note**
>
> Dans ce didacticiel, vous avez appris à intégrer une fonction Lambda dans un `Task` état de vos flux de travail. Vous pouvez également utiliser d'autres intégrations de AWS SDK prises en charge dans l'`Task`état en spécifiant le nom du service et l'appel d'API, comme indiqué dans la syntaxe suivante :
>
> ```
> arn:aws:states:::aws-sdk:serviceName:apiAction
> ```
>
> Pour plus d'informations, consultez Utilisation AWS Step Functions avec d'autres services.

Étapes suivantes

Dans le prochain didacticiel, vous allez implémenter la logique conditionnelle dans votre flux de travail. La logique conditionnelle des machines à états Step Functions se comporte de la même manière qu'une instruction if-else dans la plupart des langages de programmation courants. Vous allez utiliser la logique conditionnelle dans votre flux de travail pour déterminer le chemin d'exécution en fonction des informations conditionnelles.

Tutoriel 3 : Implémenter une condition if-else dans votre flux de travail

Vous pouvez implémenter des conditions if-else dans vos flux de travail à l'aide de l'`Choice`état. Il détermine le chemin d'exécution du flux de travail en fonction du fait qu'une condition spécifiée soit vraie ou fausse.

Dans ce didacticiel, vous allez ajouter une logique conditionnelle pour déterminer si le montant du crédit appliqué renvoyé par la fonction `RandomNumberforCredit` Lambda utilisée dans le didacticiel 2 dépasse un seuil spécifique. Si le montant dépasse le seuil, l'approbation de la demande nécessite une intervention humaine. Dans le cas contraire, la demande est approuvée automatiquement et passe à l'étape suivante.

Vous allez imiter l'étape de l'interaction humaine en interrompant l'exécution du flux de travail jusqu'à ce qu'un jeton de tâche soit renvoyé. Pour ce faire, vous allez transmettre un jeton de tâche à l'intégration du AWS SDK que vous utiliserez dans ce didacticiel, à savoir Amazon Simple Notification Service. L'exécution du flux de travail sera suspendue jusqu'à ce qu'il reçoive le jeton de tâche en retour avec un appel d'`SendTaskSuccess`API. Pour plus d'informations sur l'utilisation des jetons de tâche, consultezAttendre un rappel avec le jeton de tâche.

Comme vous avez déjà défini les étapes de l'approbation humaine et de l'approbation automatique dans votre prototype de flux de travail, dans ce didacticiel, vous allez d'abord créer une rubrique Amazon SNS qui reçoit le jeton de rappel. Vous créez ensuite une fonction Lambda pour implémenter la fonctionnalité de rappel. Enfin, vous mettez à jour votre prototype de flux de travail en ajoutant les détails de ces Service AWS intégrations.

Rubriques

Étape 1 : créer une rubrique Amazon SNS qui reçoit le jeton de rappel

Pour implémenter l'étape d'interaction humaine, vous allez publier dans une rubrique Amazon Simple Notification Service et transmettre le jeton de tâche de rappel à cette rubrique. La tâche de rappel

interrompt l'exécution du flux de travail jusqu'à ce que le jeton de tâche soit renvoyé avec une charge utile.

1. Ouvrez la console Amazon SNS et créez un type de rubrique standard. Pour plus d'informations sur la création d'une rubrique, consultez la section Créer une rubrique Amazon SNS dans le Guide du développeur Amazon Simple Notification Service.
2. Spécifiez le nom du sujet sous la forme**`TaskTokenTopic`**.
3. Veillez à copier l'ARN de la rubrique et à l'enregistrer dans un fichier texte. Vous aurez besoin de l'ARN du sujet lorsque vous spécifiez l'intégration du service pour l'état En attente d'approbation humaine. Voici un exemple de rubrique ARN :

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

4. Créez un abonnement par e-mail pour le sujet, puis confirmez votre abonnement. Pour plus d'informations sur l'abonnement à une rubrique, consultez la section Créer un abonnement à la rubrique dans le Guide du développeur Amazon Simple Notification Service.

Étape 2 : Création d'une fonction Lambda pour gérer le rappel

Pour gérer la fonctionnalité de rappel, vous allez définir une fonction Lambda et ajouter la rubrique Amazon SNS que vous avez créée à l'étape 1 en tant que déclencheur de cette fonction. Lorsque vous publiez sur la rubrique Amazon SNS à l'aide d'un jeton de tâche, la fonction Lambda est appelée avec la charge utile du message publié.

- Étape 2.1 : Création de la fonction Lambda pour gérer le rappel
- Étape 2.2 : Ajouter la rubrique Amazon SNS en tant que déclencheur pour la fonction Lambda
- Étape 2.3 : Fournir les autorisations nécessaires au rôle IAM de la fonction Lambda

Étape 2.1 : Création de la fonction Lambda pour gérer le rappel

Dans cette fonction, vous allez traiter la demande d'approbation de la limite de crédit et renvoyer le résultat de la demande comme étant un succès lors de l'appel `SendTaskSuccess`d'API. Cette fonction Lambda renverra également le jeton de tâche qu'elle a reçu dans le cadre de la rubrique Amazon SNS.

Pour des raisons de simplicité, la fonction Lambda utilisée pour l'étape d'interaction humaine approuve automatiquement n'importe quelle tâche et renvoie le jeton de tâche avec un appel

d'SendTaskSuccessAPI. Vous pouvez nommer la fonction Lambda comme suit. **`callback-human-approval`**

1. Dans un nouvel onglet ou une nouvelle fenêtre, ouvrez la console Lambda et créez une fonction Lambda Node.js 16.x intitulée. **`callback-human-approval`** Pour plus d'informations sur la création d'une fonction Lambda à l'aide de la console, voir Créer une fonction Lambda dans la console du Guide du développeur. AWS Lambda
2. Sur la callback-human-approvalpage, remplacez le code existant dans la zone Code source par le code suivant.

```
// 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. Laissez cette fenêtre ouverte et suivez les étapes décrites dans la section suivante pour d'autres actions.

Étape 2.2 : Ajouter la rubrique Amazon SNS en tant que déclencheur pour la fonction Lambda

Lorsque vous ajoutez la rubrique Amazon SNS que vous avez créée à l'étape 1 de ce didacticiel en tant que déclencheur pour la fonction Lambda que vous avez créée à l'étape 2.1 de ce didacticiel, la fonction Lambda est déclenchée chaque fois que vous publiez sur la rubrique Amazon SNS. Lorsque vous publiez sur la rubrique Amazon SNS à l'aide d'un jeton de tâche, la fonction Lambda est appelée avec la charge utile du message publié. Pour plus d'informations sur la configuration des déclencheurs pour les fonctions Lambda, consultez la section Configuration des déclencheurs dans le Guide du AWS Lambdadéveloppeur.

1. Dans la section Vue d'ensemble des fonctions de la fonction `callback-human-approval` Lambda, choisissez Ajouter un déclencheur.
2. Dans la liste déroulante des déclencheurs, choisissez SNS comme déclencheur.
3. Pour la rubrique SNS, commencez à saisir le nom de la rubrique Amazon SNS que vous avez créée à l'étape 1 de ce didacticiel, puis choisissez-la dans la liste déroulante qui s'affiche.
4. Choisissez Add (Ajouter).
5. Laissez cette fenêtre ouverte et suivez les étapes décrites dans la section suivante pour d'autres actions.

Étape 2.3 : Fournir les autorisations nécessaires au rôle IAM de la fonction Lambda

Vous devez fournir à la fonction `callback-human-approval` Lambda les autorisations nécessaires pour accéder à Step Functions afin de renvoyer le jeton de tâche avec l'appel d'`SendTaskSucess`API.

1. Sur la callback-human-approvalpage, choisissez l'onglet Configuration, puis choisissez Autorisations.
2. Sous Rôle d'exécution, choisissez le nom du rôle pour accéder à la page Rôles de la AWS Identity and Access Management console.
3. Pour ajouter l'autorisation requise, choisissez Ajouter des autorisations, puis choisissez Attacher des politiques.

4. Dans la zone de recherche, tapez **`AWSStepFunctions`** puis appuyez sur Entrée.
5. Choisissez, AWSStepFunctionsFullAccesspuis faites défiler la page vers le bas pour sélectionner Attacher des politiques. Cela ajoute la politique contenant l'autorisation nécessaire pour le rôle de fonction `callback-human-approval` Lambda.

Étape 3 : Mettre à jour le flux de travail : ajouter une logique de condition if-else dans l'état Choice

Dans la console Step Functions, définissez la logique conditionnelle pour votre flux de travail à l'aide de l'`Choice`état. Si la sortie renvoyée par la fonction `RandomNumberforCredit` Lambda est inférieure à 5 000, le crédit demandé est automatiquement approuvé. Si le résultat renvoyé est supérieur ou égal à 5 000, l'exécution du flux de travail passe à l'étape d'interaction humaine pour l'approbation de la limite de crédit.

Dans l'`Choice`état, vous utilisez un opérateur de comparaison pour comparer une variable d'entrée à une valeur spécifique. Vous pouvez spécifier la variable d'entrée comme entrée d'exécution lors du démarrage d'une exécution par machine d'état ou utiliser la sortie d'une étape précédente comme entrée pour l'étape en cours. Par défaut, la sortie d'une étape est stockée dans une variable appelée`Payload`. Pour utiliser la valeur de la `Payload` variable à des fins de comparaison dans l'`Choice`état, utilisez la $ syntaxe indiquée dans la procédure suivante.

Pour plus d'informations sur la façon dont les informations circulent d'un état à un autre et sur la spécification des entrées et des sorties dans vos flux de travail, voir Tutoriel 7 : Configuration des entrées et des sorties etTraitement des entrées et des sorties dans Step Functions.

> **Note**
>
> Si l'`Choice`état utilise une variable d'entrée spécifiée dans l'entrée d'exécution de la machine d'état à des fins de comparaison, utilisez la `$.variable_name` syntaxe pour effectuer la comparaison. Par exemple, pour comparer une variable`myAge`, utilisez la syntaxe`$.myAge`.

Étant donné qu'à cette étape, l'`Choice`état recevra des informations provenant de l'état Obtenir la limite de crédit, vous allez utiliser la $ syntaxe de configuration de l'`Choice`état. Pour découvrir en quoi le résultat de l'exécution de la machine d'état diffère lorsque vous utilisez la `$.variable_name` syntaxe de la configuration d'`Choice`état pour faire référence au résultat d'une étape précédente, consultez la Débogage de l'erreur d'état de choix de chemin non valide section du didacticiel 8.

Pour ajouter une logique de condition if-else à l'aide de l'état **`Choice`**

1. Ouvrez la fenêtre de console Step Functions contenant le prototype de flux de travail que vous avez créé dansTutoriel 1 : Création du prototype de votre machine à états.
2. Choisissez le crédit appliqué >= 5000 ? et dans l'onglet Configuration, spécifiez la logique conditionnelle comme suit :

   a. Sous Règles de choix, cliquez sur l'icône Modifier dans la vignette Règle #1 pour définir la règle de premier choix.

   b. Choisissez Ajouter des conditions.

   c. Dans la boîte de dialogue Conditions pour la règle #1, dans le champ Variable, entrez**$**.

   d. Pour Opérateur, choisissez une valeur inférieure à.

   e. Pour Valeur, choisissez Constante numérique, puis saisissez-la **`5000`** dans le champ à côté de la liste déroulante Valeur.

   f. Choisissez Enregistrer les conditions.

   g. Dans la liste déroulante Alors l'état suivant est :, choisissez Limite d'approbation automatique.

   h. Choisissez Ajouter une nouvelle règle de choix, puis définissez la deuxième règle de choix lorsque le montant du crédit est supérieur ou égal à 5 000 en répétant les sous-étapes 2.b à 2.f. Pour Opérateur, choisissez une valeur supérieure ou égale à.

   i. Dans la liste déroulante Alors l'état suivant est :, choisissez Attendre l'approbation humaine.

   j. Dans la zone Règle par défaut, cliquez sur l'icône Modifier pour définir la règle de choix par défaut, puis choisissez Attendre l'approbation humaine dans la liste déroulante État par défaut. Vous définissez la règle par défaut pour spécifier l'état suivant vers lequel passer si aucune des conditions de l'état Choice n'est vraie ou fausse.

3. Configurez l'état Attendre l'approbation humaine comme suit :

   a. Dans l'onglet Configuration, pour Rubrique, commencez à saisir le nom de la rubrique Amazon SNS TaskTokenTopic, puis choisissez le nom tel qu'il apparaît dans la liste déroulante.

   b. Pour Message, choisissez Entrer un message dans la liste déroulante. Dans le champ Message, vous spécifiez le message que vous souhaitez publier dans la rubrique Amazon SNS. Pour ce didacticiel, vous publiez un jeton de tâche sous forme de message.

Un jeton de tâche vous permet de suspendre un flux de travail Step Functions de type standard jusqu'à ce qu'un processus externe soit terminé et que le jeton de tâche soit renvoyé. Lorsque vous spécifiez un état de tâche en tant que tâche de rappel en spécifiant le modèle d'intégration du `.waitForTaskToken` service, un jeton de tâche est généré et placé dans l'objet contextuel lorsque la tâche est lancée. L'objet de contexte est une structure JSON interne qui est disponible lors d'une exécution et qui contient des informations sur votre machine d'état et son exécution. Pour plus d'informations sur les objets contextuels, reportez-vous à la sectionObjet Contexte.

c. Dans la zone qui s'affiche, saisissez le message suivant :

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

d. Cochez la case Attendre le rappel.

e. Choisissez OK dans la boîte de dialogue qui s'affiche.

4. Laissez cette fenêtre ouverte et passez au didacticiel suivant pour d'autres actions.

Étapes suivantes

Dans le didacticiel suivant, vous allez apprendre à effectuer plusieurs tâches en parallèle.

Tutoriel 4 : Définir plusieurs tâches à effectuer en parallèle

Jusqu'à présent, vous avez appris à exécuter des flux de travail de manière séquentielle. Toutefois, vous pouvez exécuter deux étapes ou plus en parallèle à l'aide de l'`Parallel`état. Un `Parallel` état amène l'interpréteur à exécuter chaque branche simultanément.

Les deux branches d'un `Parallel` état reçoivent la même entrée, mais chaque branche traite les parties d'entrée qui lui sont spécifiques. Step Functions attend la fin de l'exécution de chaque branche avant de passer à l'étape suivante.

Dans ce didacticiel, vous allez utiliser l'état parallèle pour vérifier simultanément l'identité et l'adresse du demandeur.

Rubriques

- Étape 1 : Création des fonctions Lambda pour effectuer les vérifications requises

- Étape 2 : Mettre à jour le flux de travail — Ajouter des tâches parallèles à effectuer

Étape 1 : Création des fonctions Lambda pour effectuer les vérifications requises

Ce flux de travail de demande de carte de crédit fait appel à deux fonctions Lambda dans l'état parallèle pour vérifier l'identité et l'adresse du demandeur. Ces vérifications sont effectuées simultanément à l'aide de l'état parallèle. La machine d'état ne termine l'exécution qu'une fois que les deux branches parallèles ont terminé leur exécution.

Pour créer les fonctions Lambda Checkidentity et Checkaddress

1. Dans un nouvel onglet ou une nouvelle fenêtre, ouvrez la console Lambda et créez deux fonctions Lambda Node.js 16.x intitulées et. `check-identity` `check-address` Pour plus d'informations sur la création d'une fonction Lambda à l'aide de la console, voir Créer une fonction Lambda dans la console du Guide du développeur. AWS Lambda
2. Ouvrez la page de la fonction check-identity et remplacez le code existant dans la zone Code source par le code suivant :

```
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. Ouvrez la page de fonction de vérification de l'adresse et remplacez le code existant dans la zone Code source par le code suivant :

```
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. Pour les deux fonctions Lambda, dans la section Présentation des fonctions, copiez leurs noms de ressources Amazon (ARN) respectifs et enregistrez-les dans un fichier texte. Vous aurez besoin des ARN de la fonction pour spécifier l'intégration du service pour vérifier l'identité et l'état de l'adresse du demandeur. Voici un exemple d'ARN :

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

Étape 2 : Mettre à jour le flux de travail — Ajouter des tâches parallèles à effectuer

Dans la console Step Functions, vous allez mettre à jour votre flux de travail pour spécifier l'intégration du service avec les fonctions Lambda d'identité de contrôle et d'adresse de contrôle que vous avez créées à l'étape 1.

Pour ajouter des tâches parallèles dans le flux de travail

1. Ouvrez la fenêtre de console Step Functions contenant le prototype de flux de travail que vous avez créé dansTutoriel 1 : Création du prototype de votre machine à états.
2. Choisissez l'état Vérifier l'identité et, dans l'onglet Configuration, procédez comme suit :

   a. Pour le type d'intégration, conservez la sélection par défaut sur Optimisé.

   > **Note**
   >
   > À l'aide de Step Functions, vous pouvez les intégrer à d'autres Services AWS et les orchestrer dans vos flux de travail. Pour plus d'informations sur les intégrations de services et leurs types, voir Utilisation AWS Step Functions avec d'autres services

   b. Pour le nom de la fonction, choisissez la fonction Lambda check-identity dans la liste déroulante.

   c. Pour la charge utile, choisissez Enter payload, puis remplacez l'exemple de charge utile par le suivant en tant que charge utile :

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

```
}
```

3. Choisissez l'état Vérifier l'adresse et, dans l'onglet Configuration, procédez comme suit :

   a. Pour le type d'intégration, conservez la sélection par défaut sur Optimisé.

   b. Pour le nom de la fonction, choisissez l'adresse de contrôle de la fonction Lambda dans la liste déroulante.

   c. Pour la charge utile, choisissez Enter payload, puis remplacez l'exemple de charge utile par le suivant en tant que charge utile :

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

4. Choisissez Suivant.

Tutoriel 5 : itérer simultanément sur une collection d'éléments

Dans le didacticiel précédent, vous avez appris à exécuter des branches distinctes d'étapes en parallèle à l'aide de l'`Parallel`état. À l'aide de `Map` l'état, vous pouvez exécuter un ensemble d'étapes de flux de travail pour chaque élément d'un ensemble de données. Les itérations de `Map` l'état s'exécutent en parallèle, ce qui permet de traiter rapidement un ensemble de données.

En incluant l'`Map`état dans vos flux de travail, vous pouvez effectuer des tâches, telles que le traitement des données, en utilisant l'un des deux modes Modes de traitement de l'état des cartes : en ligne et en mode distribué. Pour configurer un `Map` état, vous définissez un`ItemProcessor`, qui contient des objets JSON qui spécifient le mode de traitement de `Map` l'état et sa définition. Dans ce didacticiel, vous allez exécuter l'`Map`état dans le mode Inline par défaut, qui prend en charge jusqu'à 40 itérations simultanées. Lorsque vous exécutez l'`Map`état en mode distribué, il prend en charge jusqu'à 10 000 exécutions parallèles de flux de travail enfants.

Lorsque l'exécution de votre flux de travail entre dans `Map` cet état, elle itère sur un tableau JSON spécifié dans l'entrée d'état. Pour chaque élément du tableau, l'itération correspondante s'exécute dans le contexte du flux de travail contenant l'`Map`état. Lorsque toutes les itérations sont terminées, l'`Map`état renvoie un tableau contenant la sortie pour chaque élément traité par le`ItemProcessor`.

Dans ce didacticiel, vous apprendrez à utiliser l'`Map`état en mode Inline pour récupérer le pointage de crédit d'un candidat en effectuant une itération sur un ensemble de bureaux de crédit. Pour ce faire, vous devez d'abord récupérer les noms de toutes les agences de crédit stockées dans une table Amazon DynamoDB, puis vous utilisez l'état pour parcourir `Map` la liste des agences de crédit afin de récupérer la cote de solvabilité du candidat déclarée par chacune de ces agences.

Rubriques

Étape 1 : créer une table DynamoDB pour stocker le nom de toutes les agences de crédit

Au cours de cette étape, vous allez créer une table nommée à **`GetCreditBureau`** l'aide de la console DynamoDB. La table utilise l'attribut de chaîne Name comme clé de partition. Dans ce tableau, vous enregistrez le nom de toutes les agences de crédit auprès desquelles vous souhaitez obtenir le pointage de crédit du demandeur.

1. Connectez-vous à la console DynamoDB AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/dynamodb/.
2. Dans le volet de navigation de la console, choisissez Tables, puis Create table.
3. Saisissez les informations de la table comme suit :
    a. Comme Nom de la table, entrez **`GetCreditBureau`**.
    b. Pour la clé de partition, saisissez **`Name`**.
    c. Conservez les sélections par défaut, puis choisissez Create table.
4. Une fois votre table créée, dans la liste Tables, GetCreditBureauchoisissez-la.
5. Choisissez Actions, puis sélectionnez Créer un élément.
6. Dans Valeur, entrez le nom d'une agence de crédit. Par exemple, **`CredTrack`**.
7. Choisissez Créer un élément.

8. Répétez ce processus et créez des éléments pour les noms d'autres bureaux de crédit. Par exemple : **`KapFinn`** et **`CapTrust`**.

Étape 2 : Mettre à jour la machine à états — Extraire les résultats depuis la table DynamoDB

Dans la console Step Functions, vous allez ajouter un `Task` état et utiliser l'intégration du AWS SDK pour récupérer les noms des agences de crédit dans la table DynamoDB que vous avez créée à l'étape 1. Vous utiliserez le résultat de cette étape comme entrée pour l'`Map`état que vous ajouterez ultérieurement dans votre flux de travail dans ce didacticiel.

1. Ouvrez la machine CreditCardWorkflowd'état pour la mettre à jour.
2. Choisissez l'état Obtenir la liste des agences de crédit.
3. Pour les paramètres d'API, spécifiez la valeur du nom de la table sous la forme**`GetCreditBureau`**.

Étape 3 : Création d'une fonction Lambda qui renvoie les cotes de crédit de tous les bureaux de crédit

Au cours de cette étape, vous créez une fonction Lambda qui reçoit les noms de toutes les agences de crédit en entrée et renvoie la cote de solvabilité du demandeur pour chacune de ces agences de crédit. Cette fonction Lambda sera invoquée à partir de l'`Map`état que vous ajouterez à votre flux de travail à l'étape 4 de ce didacticiel.

1. Créez une fonction Lambda 16.x de Node.js et nommez-la. **`get-credit-score`**
2. Sur la page intitulée get-credit-score, collez le code suivant dans la zone Code source.

```
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. Déployez la fonction Lambda.

Étape 4 : Mettre à jour la machine à états — ajouter un état de carte pour récupérer de manière itérative les cotes de crédit

Dans la console Step Functions, vous ajoutez un `Map` état qui invoque la fonction get-credit-scoreLambda pour vérifier la cote de solvabilité du candidat pour toutes les agences de crédit renvoyées par l'état Get list of Credit Bureaus.

1. Ouvrez la machine CreditCardWorkflowd'état pour la mettre à jour.
2. Choisissez l'option Obtenir les scores de tous les bureaux de crédit de l'État.
3. Dans l'onglet Configuration, choisissez Fournir un chemin vers le tableau d'éléments, puis entrez**`$.Items`**.
4. Choisissez Get all scores step inside the `Map` state.
5. Dans l'onglet Configuration, assurez-vous que Type d'intégration, Optimisé est sélectionné.
6. Dans Nom de la fonction, commencez à saisir le nom de la fonction get-credit-scoreLambda et choisissez-le dans la liste déroulante qui apparaît.
7. Pour Charge utile, sélectionnez Aucune charge utile.

Tutoriel 6 : Enregistrer le flux de travail et exécuter la machine à états

Maintenant que vous avez configuré les ressources de toutes les ressources que Services AWS vous utilisez dans le prototype de flux de travail, vous pouvez l'enregistrer en tant que machine d'état Step Functions et commencer à l'exécuter.

Rubriques

- Étape 1 : Passez en revue la définition de la machine à états générée automatiquement et enregistrez la machine à états

- Étape 2 : ajouter les politiques IAM restantes
- Étape 3 : Exécutez la machine d'état

Étape 1 : Passez en revue la définition de la machine à états générée automatiquement et enregistrez la machine à états

Lorsque vous glissez et déposez les états de l'onglet Flow sur le canevas de Workflow Studio pour créer le prototype de flux de travail, Step Functions compose automatiquement la définition Amazon States Language (ASL) de votre flux de travail en temps réel. Vous pouvez modifier cette définition selon les besoins dans leÉditeur de code.

Pour revoir la définition ASL et enregistrer la machine à états

1. (Facultatif) Choisissez Definition sur le Inspector pour afficher la définition de la machine à états Amazon States Language (ASL), qui est automatiquement générée en fonction de vos sélections dans les onglets Actions et Flow et dans le panneau Inspector.

   > **Tip**
   >
   > Pour modifier la définition, vous pouvez ouvrir l'éditeur de code en choisissant Code en haut de la page. Pour ce didacticiel, poursuivez avec la définition générée automatiquement.

2. Spécifiez un nom pour votre machine à états. Pour ce faire, cliquez sur l'icône d'édition à côté du nom de la machine à états par défaut de MyStateMachine. Ensuite, dans Configuration de la machine d'état, spécifiez un nom dans le champ Nom de la machine d'état.

   Pour ce didacticiel, saisissez le nom **`CreditCardWorkflow`**.

3. (Facultatif) Dans Configuration de la machine à états, spécifiez d'autres paramètres de flux de travail, tels que le type de machine à états et son rôle d'exécution.

   Pour ce didacticiel, conservez toutes les sélections par défaut dans les paramètres State Machine.

> **Note**
>
> (Facultatif) Step Functions crée automatiquement un rôle d'exécution pour la machine à états disposant du moins de privilèges requis pour appeler la fonction `RandomNumberforCredit` Lambda et publier sur la rubrique Amazon SNS.

Si vous avez déjà créé un rôle IAM avec les autorisations appropriées pour votre machine d'état et que vous souhaitez l'utiliser, dans Autorisations, sélectionnez Choisir un rôle existant, puis sélectionnez un rôle dans la liste. Vous pouvez également sélectionner Entrer un ARN de rôle, puis fournir un ARN pour ce rôle IAM.

4. Dans la boîte de dialogue Confirmer la création du rôle, choisissez Confirmer pour continuer.

   Vous pouvez également choisir Afficher les paramètres des rôles pour revenir à la configuration de la machine State.

> **Note**
>
> Si vous supprimez le rôle IAM créé par Step Functions, Step Functions ne pourra pas le recréer ultérieurement. De même, si vous modifiez le rôle (par exemple, en supprimant Step Functions des principes de la politique IAM), Step Functions ne pourra pas restaurer ses paramètres d'origine ultérieurement.

Étape 2 : ajouter les politiques IAM restantes

Step Functions ne générant pas automatiquement les autorisations nécessaires pour appeler les fonctions Lambda utilisées dans `Parallel` l'état, vous devez ajouter la politique nécessaire.

Pour ajouter la politique restante

1. Sur la CreditCardWorkflowpage, choisissez le rôle IAM pour votre machine d'état pour accéder à la console IAM. Vous allez ajouter les autorisations nécessaires pour les autres fonctions Lambda sur cette page.
2. Choisissez Ajouter des autorisations, puis Attacher des politiques.
3. Dans le champ de recherche, tapez **`AWSLambdaRole`** puis appuyez sur Entrée.

4. Choisissez AWSLambdaRolepuis choisissez Joindre des politiques. Cette politique est désormais ajoutée au rôle d'exécution de votre machine d'état. Cette politique vous permet d'invoquer n'importe quelle fonction Lambda dans votre machine à états.

Étape 3 : Exécutez la machine d'état

Les exécutions par State Machine sont des instances dans lesquelles vous exécutez votre flux de travail pour effectuer des tâches.

Pour exécuter la machine à états

1. Sur la CreditCardWorkflowpage, choisissez Démarrer l'exécution.

   La boîte de dialogue Démarrer l'exécution s'affiche.

2. Dans la boîte de dialogue Démarrer l'exécution, procédez comme suit :

   a. (Facultatif) Pour identifier votre exécution, vous pouvez lui donner un nom dans le champ Nom. Par défaut, Step Functions génère automatiquement un nom d'exécution unique.

   > **Note**
   >
   > Step Functions vous permet de créer des noms pour les machines d'état, les exécutions, les activités et les étiquettes contenant des caractères non ASCII. Ces noms non ASCII ne fonctionnent pas avec Amazon. CloudWatch Pour être sûr de pouvoir suivre CloudWatch les métriques, choisissez un nom qui utilise uniquement des caractères ASCII.

   > **Note**
   >
   > Vous n'avez pas besoin de fournir d'entrée pour exécuter cette machine à états. Mais vous pouvez spécifier une entrée d'exécution, si nécessaire, dans la zone Entrée de la boîte de dialogue Démarrer l'exécution pour les autres machines à états. Pour un exemple de la manière de fournir une entrée d'exécution à une machine à états, voir Étape 4 : démarrer une nouvelle exécution du didacticiel Apprenez à utiliser le AWS Step Functions Workflow Studio.

   b. Choisissez Start execution (Démarrer l'exécution).

3. La console Step Functions vous dirige vers une page intitulée avec votre ID d'exécution. Cette page est connue sous le nom de page Détails de l'exécution. Sur cette page, vous pouvez consulter les résultats de l'exécution au fur et à mesure que l'exécution progresse ou une fois celle-ci terminée.

   Pour consulter les résultats de l'exécution, choisissez des états individuels dans la vue graphique, puis choisissez les onglets individuels du Détails de l'étape volet pour afficher les détails de chaque état, y compris les entrées, les sorties et la définition respectivement. Pour plus de détails sur les informations d'exécution que vous pouvez consulter sur la page Détails de l'exécution, voirPage de détails d'exécution — Vue d'ensemble de l'interface.

Tutoriel 7 : Configuration des entrées et des sorties

Une exécution de Step Functions reçoit un texte JSON en tant qu'entrée et transmet cette entrée au premier état du flux de travail. Les états individuels d'un flux de travail reçoivent des données JSON en entrée et transmettent généralement les données JSON en sortie à l'état suivant. Par défaut, les données passent d'un état à l'autre du flux de travail, sauf si vous avez configuré l'entrée et/ou la sortie pour un ou plusieurs états du flux de travail. Il est essentiel de comprendre comment les informations circulent d'un État à un autre et d'apprendre à filtrer et à manipuler ces données pour concevoir et implémenter efficacement des flux de travail dans Step Functions.

Step Functions fournit plusieurs filtres pour contrôler le flux de données d'entrée et de sortie entre les états. Les filtres suivants peuvent être utilisés dans vos flux de travail :

> **Note**
>
> Selon votre cas d'utilisation, il se peut que vous n'ayez pas besoin d'appliquer tous ces filtres à vos flux de travail.

`InputPath`

Sélectionne la partie de l'ensemble de la charge utile d'entrée à utiliser comme entrée d'une tâche. Si vous spécifiez ce champ, Step Functions applique d'abord ce champ.

`Paramètres`

Spécifie à quoi doit ressembler l'entrée avant d'appeler la tâche. Ce `Parameters` champ vous permet de créer une collection de paires clé-valeur qui sont transmises en entrée à une Service

AWSintégration, telle qu'une AWS Lambda fonction. Ces valeurs peuvent être statiques ou sélectionnées dynamiquement à partir de l'entrée d'état ou de l'objet de contexte du flux de travail.

`ResultSelector`

Détermine ce qu'il faut choisir parmi les résultats d'une tâche. Ce `ResultSelector` champ vous permet de créer une collection de paires clé-valeur qui remplacent le résultat d'un état et de transmettre cette collection à. `ResultPath`

`ResultPath`

Détermine où placer la sortie d'une tâche. Utilisez le `ResultPath` pour déterminer si la sortie d'un état est une copie de son entrée, du résultat qu'il produit ou une combinaison des deux.

`OutputPath`

Détermine CE qu'il faut envoyer à l'état suivant. Avec`OutputPath`, vous pouvez filtrer les informations indésirables et ne transmettre que la partie des données JSON qui vous intéresse.

> **Tip**
>
> Les `ResultSelector` filtres `Parameters` et fonctionnent en construisant du JSON, tandis que les `OutputPath` filtres `InputPath` et fonctionnent en filtrant des nœuds spécifiques au sein d'un objet de données JSON, et le `ResultPath` filtre fonctionne en créant un champ sous lequel la sortie peut être ajoutée.

Dans ce didacticiel, vous allez apprendre à effectuer les tâches suivantes :

- Sélectionnez des parties spécifiques de l'entrée brute à l'aide du InputPath filtre
- Manipuler l'entrée sélectionnée à l'aide du filtre Paramètres
- Configurez la sortie à l'aide des OutputPath filtres ResultSelectorResultPath, et

Pour plus d'informations sur la configuration des entrées et des sorties dans vos flux de travail, consultezTraitement des entrées et des sorties dans Step Functions.

Sélectionnez des parties spécifiques de l'entrée brute à l'aide du InputPath filtre

Utilisez le `InputPath` filtre pour sélectionner une partie spécifique de la charge utile d'entrée.

Si vous ne le spécifiez pas`InputPath`, sa valeur par défaut est`$`, ce qui fait que la tâche de l'état fait référence à la totalité de l'entrée brute plutôt qu'à une partie spécifique.

Pour savoir comment utiliser le InputPath filtre, effectuez les opérations suivantes :

- Étape 1 : créer une machine à états
- Étape 2 : Exécuter la machine d'état
- Étape 3 : utiliser le InputPath filtre pour sélectionner des parties spécifiques d'une entrée d'exécution

Étape 1 : créer une machine à états

> **Important**
>
> Assurez-vous que votre machine d'état se trouve sous le même AWS compte et dans la même région que la fonction Lambda que vous avez créée précédemment.

1. Utilisez l'exemple `Parallel` d'état que vous avez découvert dans le didacticiel 4 pour créer une nouvelle machine à états. Assurez-vous que votre prototype de flux de travail ressemble au prototype suivant.
2. Configurez les intégrations pour les fonctions `check-identity` et `check-address` Lambda. Pour plus d'informations sur la création des fonctions Lambda et leur utilisation dans votre machine à états, reportez-vous Étape 1 : Création des fonctions Lambda pour effectuer les vérifications requises aux sections et. Étape 2 : Mettre à jour le flux de travail — Ajouter des tâches parallèles à effectuer
3. Pour Payload, assurez-vous de conserver la sélection par défaut Use state input as payload.
4. Choisissez Next, puis suivez les étapes 1 à 3 Étape 1 : enregistrer la machine à états du didacticiel 5 pour créer une nouvelle machine à états. Pour ce didacticiel, nommez votre machine à états**`WorkflowInputOutput`**.

Étape 2 : Exécuter la machine d'état

1. Sur la WorkflowInputOutputpage, choisissez Démarrer l'exécution.
2. (Facultatif) Pour identifier votre exécution, vous pouvez lui donner un nom dans le champ Nom. Par défaut, Step Functions génère automatiquement un nom d'exécution unique.

> **Note**
>
> Step Functions vous permet de créer des noms pour les machines d'état, les exécutions, les activités et les étiquettes contenant des caractères non ASCII. Ces noms non ASCII ne fonctionnent pas avec Amazon. CloudWatch Pour être sûr de pouvoir suivre CloudWatch les métriques, choisissez un nom qui utilise uniquement des caractères ASCII.

3. Dans la zone Entrée, ajoutez les données JSON suivantes comme entrée d'exécution.

```
{
  "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. Choisissez Start execution (Démarrer l'exécution).
5. L'exécution de la machine à états entraîne une erreur car vous n'avez pas spécifié les parties de l'entrée d'exécution que les fonctions `check-identity` et `check-address` Lambda doivent utiliser pour effectuer la vérification d'identité et d'adresse requise.
6. Passez à l'étape 3 de ce didacticiel pour corriger l'erreur.

Étape 3 : utiliser le **`InputPath`** filtre pour sélectionner des parties spécifiques d'une entrée d'exécution

1. Sur la page Détails de l'exécution, choisissez Modifier l'état de la machine.

2. Pour vérifier l'identité du demandeur telle que mentionnée dans l'entrée d'exécution fournie dans[Étape 2 : Exécuter la machine d'état](), modifiez la définition de la tâche de vérification de l'identité comme suit :

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

Par conséquent, les données JSON suivantes deviennent disponibles en entrée pour la `check-identity` fonction.

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

3. Pour vérifier l'adresse du candidat telle qu'elle est mentionnée dans l'entrée d'exécution, modifiez la définition de la `Verify address` tâche comme suit :

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

Par conséquent, les données JSON suivantes deviennent disponibles en entrée pour la `check-address` fonction.

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

4. Choisissez Start execution (Démarrer l'exécution). L'exécution de la machine à états se termine désormais correctement.

Manipuler l'entrée sélectionnée à l'aide du filtre Paramètres

Bien que le `InputPath` filtre vous aide à limiter l'entrée JSON brute que vous fournissez, à l'aide du `Parameters` filtre, vous pouvez transmettre une collection de paires clé-valeur en entrée. Ces paires clé-valeur peuvent être soit des valeurs statiques que vous définissez dans la définition de votre machine d'état, soit des valeurs sélectionnées à partir de l'entrée brute à l'aide de. `InputPath`

Dans vos flux de travail, `Parameters` sont appliqués par la suite`InputPath`. `Parameters`vous aider à spécifier la manière dont la tâche sous-jacente accepte sa charge utile d'entrée. Par exemple, si la fonction `check-address` Lambda accepte un paramètre de chaîne comme entrée au lieu des données JSON, vous pouvez utiliser le `Parameters` filtre pour transformer l'entrée.

Dans l'exemple suivant, le `Parameters` filtre reçoit l'entrée que vous avez sélectionnée `InputPath` à l'aide de in Étape 3 : utiliser le `InputPath` filtre pour sélectionner des parties spécifiques d'une entrée d'exécution et applique la `States.Format` fonction intrinsèque aux éléments d'entrée pour créer une chaîne appelée`addressString`. Les fonctions intrinsèques vous aident à effectuer des

opérations de traitement de données de base sur une entrée donnée. Pour plus d'informations, veuillez consulter Fonctions intrinsèques.

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

Par conséquent, la chaîne suivante est créée et est fournie à la fonction `check-address` Lambda en tant qu'entrée.

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

Configurez la sortie à l'aide des OutputPath filtres ResultSelectorResultPath, et

Lorsque la fonction `check-address` Lambda est invoquée dans la machine d'WorkflowInputOutputétat, elle renvoie une charge utile en sortie après avoir effectué la vérification de l'adresse. Sur la page Détails de l'exécution, choisissez l'étape Vérifier l'adresse et affichez la charge utile en sortie dans le Détails de l'étape volet Résultat de la tâche.

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

En utilisant ResultSelector

Désormais, si vous devez fournir le résultat des vérifications d'identité et d'adresse aux états suivants de votre flux de travail, vous pouvez sélectionner le nœud Payload.body dans le JSON de sortie et utiliser la fonction `StringToJson` intrinsèque du `ResultSelector` filtre pour formater les données selon vos besoins.

`ResultSelector`sélectionne ce qui est nécessaire à partir de la sortie de la tâche. Dans l'exemple suivant, **`ResultSelector`** prend la chaîne dans $.payload.body et applique la fonction **`States.StringToJson`** intrinsèque pour convertir la chaîne en JSON et place le JSON obtenu dans le nœud d'identité.

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

Par conséquent, les données JSON suivantes sont créées.

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

Lorsque vous utilisez ces filtres d'entrée et de sortie, vous pouvez également rencontrer des erreurs d'exécution en raison de la spécification d'expressions de chemin JSON non valides. Pour plus d'informations, consultez .

En utilisant ResultPath

Vous pouvez spécifier un emplacement dans la charge utile d'entrée initiale pour enregistrer le résultat du traitement des tâches d'un état à l'aide du `ResultPath` champ. Si vous ne le spécifiez pas`ResultPath`, sa valeur par défaut est$, ce qui entraîne le remplacement de la charge utile d'entrée initiale par le résultat brut de la tâche. Si vous spécifiez `ResultPath` commenull, le résultat brut est supprimé et la charge utile d'entrée initiale devient la sortie effective.

Si vous appliquez le `ResultPath` champ aux données JSON créées à l'aide du `ResultSelector` champ, le résultat de la tâche est ajouté dans le nœud de résultats de la charge utile en entrée, comme illustré dans l'exemple suivant :

```
{
  "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
    }
  }
}
```

En utilisant OutputPath

Vous pouvez sélectionner une partie de la sortie d'état après l'application de `ResultPath` pour passer à l'état suivant. Ceci vous permet de filtrer les informations indésirables et de passer uniquement la portion de JSON qui vous intéresse.

Dans l'exemple suivant, le `OutputPath` champ enregistre la sortie d'état dans le nœud de résultats :`"OutputPath": "$.results"`. Par conséquent, la sortie finale de l'état, que vous pouvez passer à l'état suivant, est la suivante :

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

Utilisation des fonctionnalités de la console pour visualiser les flux de données d'entrée et de sortie

Vous pouvez visualiser le flux de données d'entrée et de sortie entre les états de vos flux de travail à l'aide du simulateur de flux de données de la console Step Functions ou de l'option d'affichage avancé de la page Détails d'exécution.

Tutoriel 8 : Déboguer les erreurs dans la console

Lorsque vous utilisez Step Functions, vous pouvez rencontrer des erreurs d'exécution pour des raisons telles que :

- Un chemin JSON non valide pour le champ Variable dans l'`Choice`état.
- Problème de définition de la machine à états, tel qu'aucune règle correspondante définie pour un `Choice` état.
- Expressions de chemin JSON non valides lors de l'application de filtres pour manipuler les entrées et les sorties.
- Échec de la tâche en raison d'une exception de fonction Lambda.
- Erreurs d'autorisation IAM.

Dans ce didacticiel, vous allez découvrir comment corriger certaines de ces erreurs à l'aide de la console Step Functions. Pour plus d'informations, veuillez consulter Gestion des erreurs dans Step Functions.

Rubriques

- Débogage de l'erreur d'état de choix de chemin non valide
- Déboguer les erreurs d'expression de chemin JSON lors de l'application de filtres d'entrée et de sortie

Débogage de l'erreur d'état de choix de chemin non valide

Lorsque vous spécifiez un chemin JSON incorrect ou impossible à résoudre dans le champ Variable de l'`Choice`état ou que vous ne définissez pas de règle correspondante dans l'`Choice`état, vous recevez un message d'erreur lors de l'exécution de votre flux de travail.

Pour illustrer l'erreur de chemin non valide, ce didacticiel présente une erreur d'`Choice`état dans votre flux de travail. Vous allez utiliser la machine CreditCardWorkflowd'état et modifier sa définition pour introduire l'erreur.

1. Ouvrez la console Step Functions, puis choisissez la machine à CreditCardWorkflowétats.
2. Choisissez Modifier pour modifier la définition de la machine d'état. Apportez la modification mise en évidence dans le code suivant à la définition de votre machine d'État.

```
{
  "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. Choisissez Enregistrer, puis cliquez sur Enregistrer quand même.
4. Lancez la machine d'État.
5. Sur la page Détails d'exécution de l'exécution de votre machine d'état, effectuez l'une des opérations suivantes :

a. Choisissez Cause dans le message d'erreur pour voir la raison de l'échec de l'exécution.

b. Choisissez Afficher les détails de l'étape dans le message d'erreur pour afficher l'étape à l'origine de l'erreur.

6. Dans l'onglet Entrée et sortie de la section Détails des étapes, cliquez sur le bouton Affichage avancé pour voir le chemin de transfert des données d'entrée et de sortie pour un état sélectionné.

7. Dans la vue graphique, assurez-vous que le crédit appliqué est supérieur ou égal à 5 000 ? est sélectionné et procédez comme suit :

a. Affichez la valeur d'entrée de l'état dans la zone de saisie.

b. Choisissez l'onglet Définition et notez le chemin JSON spécifié pour le champ Variable.

La valeur d'entrée pour le crédit appliqué est supérieure ou égale à 5 000 ? state est une valeur numérique, alors que vous avez spécifié le chemin JSON pour la valeur d'entrée sous `$.Payload` la forme. Pendant l'exécution de la machine d'état, l'`Choice`état ne peut pas résoudre ce chemin JSON car il n'existe pas.

8. Modifiez la machine d'état pour spécifier la valeur du champ variable comme**$**.

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

Déboguer les erreurs d'expression de chemin JSON lors de l'application de filtres d'entrée et de sortie

Lorsque vous utilisez les filtres d'entrée et de sortie, vous pouvez rencontrer des erreurs d'exécution en raison de la spécification d'expressions de chemin JSON non valides.

L'exemple suivant utilise la machine WorkflowInputOutputd'état que vous avez créée dans le didacticiel 5 et montre un scénario dans lequel vous utilisez le `ResultSelector` filtre pour sélectionner des parties du résultat de la tâche.

1. Appliquez le `ResultSelector` filtre pour sélectionner une partie du résultat de la tâche pour l'étape Vérifier l'identité. Pour ce faire, modifiez la définition de votre machine d'état comme suit :

```
{
  "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. Lancez la machine d'État.
3. Sur la page Détails d'exécution de l'exécution de votre machine d'état, procédez comme suit :
   a. Choisissez Cause dans le message d'erreur pour voir la raison de l'échec de l'exécution.
   b. Choisissez Afficher les détails de l'étape dans le message d'erreur pour afficher l'étape à l'origine de l'erreur.
4. Dans le message d'erreur, notez que le contenu du nœud $.payload.body est une chaîne JSON échappée. L'erreur s'est produite car vous ne pouvez pas faire référence à une chaîne à l'aide de la notation de chemin JSON.
5. Pour faire référence au nœud $.payload.body.Message, procédez comme suit :
   a. Utilisez la fonction `States.StringToJSON` intrinsèque pour convertir d'abord la chaîne au format JSON.
   b. Spécifiez le chemin JSON pour le nœud $.payload.body.message dans la fonction intrinsèque.

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

6. Exécutez à nouveau la machine d'état.

Cas d'utilisation

AWS Step Functionsvous permet de créer des flux de travail visuels qui permettent de traduire rapidement les exigences de l'entreprise en applications. Step Functions gère l'état, les points de contrôle et les redémarrages à votre place, et fournit des fonctionnalités intégrées pour traiter automatiquement les erreurs et les exceptions. Pour mieux comprendre les fonctionnalités que Step Functions peut vous fournir, consultez les cas d'utilisation suivants :

Rubriques

Traitement des données

À mesure que le volume de données augmente, provenant de sources de plus en plus diverses, les entreprises constatent qu'elles doivent agir rapidement pour traiter ces données afin de prendre des décisions commerciales plus rapides et éclairées. Pour traiter les données à grande échelle, les entreprises doivent provisionner de manière élastique les ressources nécessaires pour gérer les informations qu'elles reçoivent des appareils mobiles, des applications, des satellites, du marketing et des ventes, des magasins de données opérationnelles, de l'infrastructure, etc.

Step Functions fournit l'évolutivité, la fiabilité et la disponibilité nécessaires pour gérer avec succès vos flux de travail de traitement des données. Vous pouvez gérer des millions d'exécutions simultanées avec Step Functions, qui évolue horizontalement et fournit des flux de travail tolérants aux pannes. Traitez les données plus rapidement à l'aide d'exécutions parallèles telles que le type d'`Parallèle`état de Step Functions ou d'un parallélisme dynamique utilisant son `Map` type d'état. Dans le cadre de votre flux de travail, vous pouvez utiliser l'`Map`état pour parcourir les objets d'un magasin de données statique tel qu'un compartiment Amazon S3. Step Functions vous permet également de réessayer facilement les exécutions ayant échoué ou de choisir une méthode spécifique pour gérer les erreurs sans avoir à gérer un processus complexe.

En fonction de vos besoins en matière de traitement des données, Step Functions s'intègre directement à d'autres services de traitement de données fournis par AWS exemple AWS Batchpour

le traitement par lots, Amazon EMR pour le traitement des mégadonnées, AWS Gluepour la préparation des données, Athena pour l'analyse des données et AWS Lambdapour le calcul.

Voici quelques exemples des types de flux de travail de traitement des données que les clients utilisent Step Functions pour réaliser :

Traitement de fichiers, de vidéos et d'images

- Prenez une collection de fichiers vidéo et convertissez-les en d'autres tailles ou résolutions adaptées à l'appareil sur lequel ils seront affichés, tel qu'un téléphone portable, un ordinateur portable ou un téléviseur.
- Prenez une grande collection de photos téléchargées par les utilisateurs et convertissez-les en vignettes ou en images de différentes résolutions qui peuvent ensuite être affichées sur les sites Web des utilisateurs.
- Prenez des données semi-structurées, telles qu'un fichier CSV, et associez-les à des données non structurées, telles qu'une facture, pour produire un rapport commercial qui est envoyé aux parties prenantes de l'entreprise tous les mois.
- Prenez les données d'observation de la Terre collectées par les satellites, convertissez-les en formats qui s'alignent les uns avec les autres, puis ajoutez d'autres sources de données collectées sur Terre pour obtenir des informations supplémentaires.
- Consultez les journaux de transport des différents modes de transport pour les produits et recherchez des optimisations à l'aide des simulations Monte Carlo, puis envoyez des rapports aux organisations et aux personnes qui comptent sur vous pour expédier leurs marchandises.

Coordonner les tâches d'extraction, de transformation et de chargement (ETL) :

- Combinez les enregistrements d'opportunités de vente avec des ensembles de données d'indicateurs marketing grâce à une série d'étapes de préparation des données à l'aide AWS Glue de rapports d'informatique décisionnelle utilisables dans l'ensemble de l'organisation.
- Créez, démarrez et résiliez un cluster Amazon EMR pour le traitement du Big Data.

Charges de travail liées au traitement par lots et au calcul haute performance (HPC) :

- Créez un pipeline d'analyse génomique secondaire qui traite les séquences brutes du génome entier en appels de variants. Alignez les fichiers bruts sur une séquence de référence et appelez des variants sur une liste spécifiée de chromosomes à l'aide du parallélisme dynamique.

- Améliorez l'efficacité de la production de votre prochain appareil mobile ou d'autres appareils électroniques en simulant différentes configurations à l'aide de différents composés électriques et chimiques. Exécutez un traitement par lots volumineux de vos charges de travail grâce à diverses simulations afin d'obtenir une conception optimale.

Machine learning

L'apprentissage automatique permet aux entreprises d'analyser rapidement les données collectées afin d'identifier des modèles, puis de prendre des décisions avec une intervention humaine minimale. L'apprentissage automatique commence par un ensemble initial de données, appelées données d'entraînement. Ces données d'entraînement contribuent à améliorer la précision des prévisions d'un modèle d'apprentissage automatique et constituent la base de l'apprentissage de ce modèle. Une fois que le modèle est considéré comme suffisamment précis pour répondre aux besoins de l'entreprise, il est déployé en production. Le kit de développement logiciel (SDK) pour la science des AWS Step Functions données est une bibliothèque open source qui vous permet de créer facilement des flux de travail qui prétraitent les données, entraînent puis publient vos modèles à l'aide d'Amazon SageMaker et de Step Functions.

Le prétraitement des ensembles de données existants est la façon dont une organisation crée souvent des données de formation. Cette méthode ajoute des informations, par exemple en étiquetant des objets dans une image, en annotant du texte ou en traitant du son. Pour prétraiter les donnéesAWS Glue, vous pouvez utiliser ou créer une instance de SageMaker bloc-notes qui exécute l'application Jupyter Notebook. Une fois que vos données sont prêtes, vous pouvez les charger sur Amazon S3 pour y accéder facilement. Au fur et à mesure que les modèles d'apprentissage automatique sont entraînés, vous pouvez ajuster les paramètres de chaque modèle afin d'améliorer la précision jusqu'à ce qu'il soit prêt à être déployé.

Step Functions vous permet d'orchestrer des flux de travail de machine learning de bout en bout sur. SageMaker Ces flux de travail peuvent inclure le prétraitement des données, le post-traitement, l'ingénierie des fonctionnalités, la validation des données et l'évaluation des modèles. Une fois le modèle déployé en production, vous pouvez affiner et tester de nouvelles approches afin d'améliorer continuellement les résultats commerciaux. Vous pouvez créer des flux de travail prêts pour la production directement dans Python, ou vous pouvez utiliser le SDK Step Functions Data Science pour copier ce flux de travail, expérimenter de nouvelles options et placer le flux de travail affiné en production.

Certains types de flux de travail d'apprentissage automatique pour lesquels les clients utilisent Step Functions incluent :

Détection des fraudes

- Identifiez et empêchez les transactions frauduleuses, telles que la fraude au crédit, de se produire.
- Détectez et empêchez les piratages de comptes à l'aide de modèles d'apprentissage automatique entraînés.
- Identifiez les abus promotionnels, y compris la création de faux comptes, afin de pouvoir agir rapidement.

Personnalisation et recommandations

- Recommandez des produits à des clients cibles en fonction de ce qui devrait susciter leur intérêt.
- Prévoyez si un client fera passer son compte d'un niveau gratuit à un abonnement payant.

Enrichissement des données

- Utilisez l'enrichissement des données dans le cadre du prétraitement afin de fournir de meilleures données d'entraînement pour des modèles d'apprentissage automatique plus précis.
- Annotez du texte et des extraits audio pour ajouter des informations syntaxiques, telles que du sarcasme et de l'argot.
- Étiquetez des objets supplémentaires dans les images pour fournir des informations essentielles dont le modèle pourra tirer des leçons, par exemple s'il s'agit d'une pomme, d'un ballon de basket, d'un rocher ou d'un animal.

Orchestration de microservices

L'architecture de microservices divise les applications en services faiblement couplés. Les avantages incluent une évolutivité améliorée, une résilience accrue et une mise sur le marché plus rapide. Chaque microservice est indépendant, ce qui facilite la mise à l'échelle d'un seul service ou d'une seule fonction sans avoir à faire évoluer l'ensemble de l'application. Les services individuels sont faiblement couplés, ce qui permet aux équipes indépendantes de se concentrer sur un seul processus métier, sans avoir à comprendre l'ensemble de l'application. Les microservices vous permettent également de choisir les composants individuels qui répondent aux besoins de votre entreprise, ce qui vous donne la possibilité de modifier votre sélection sans réécrire l'intégralité de votre flux de travail. Différentes équipes peuvent utiliser les langages de programmation et les frameworks de leur choix pour travailler avec leur microservice, et ce microservice peut toujours

communiquer avec n'importe quel autre élément de l'application via des interfaces de programmation d'applications (API).

Step Functions vous propose plusieurs méthodes pour gérer vos flux de travail liés aux microservices. Pour les flux de travail de longue durée, vous pouvez utiliser les flux de travail standard avec l'AWS Fargateintégration pour orchestrer les applications exécutées dans des conteneurs. Pour les flux de travail de courte durée et à volume élevé qui nécessitent une réponse immédiate, les flux de travail Synchronous Express sont idéaux. Ils peuvent être utilisés pour les applications Web ou mobiles, qui ont souvent des flux de travail de courte durée et nécessitent l'exécution d'une série d'étapes avant de renvoyer une réponse. Vous pouvez déclencher directement un flux de travail Express synchrone à partir d'Amazon API Gateway, et la connexion reste ouverte jusqu'à la fin du flux de travail ou jusqu'à l'expiration des délais. Pour les flux de travail de courte durée qui ne nécessitent pas de réponse immédiate, Step Functions propose des flux de travail express asynchrones.

Voici quelques exemples d'orchestrations d'API qui utilisent Step Functions :

Flux de travail synchrones ou en temps réel

- Modifiez une valeur dans un enregistrement, par exemple en mettant à jour le nom de famille d'un employé, pour que la modification soit immédiatement visible à l'écran.
- Mettez à jour une commande lors du paiement, par exemple en ajoutant, en supprimant ou en modifiant la quantité d'un article, puis informez immédiatement le client de la mise à jour.
- Exécutez une tâche de traitement rapide et renvoyez immédiatement le résultat au demandeur.

Orchestration de conteneurs

- Exécutez des tâches sur Kubernetes avec Amazon Elastic Kubernetes Service ou sur Amazon Elastic Container Service (ECS) avec Fargate et intégrez d'autres AWS services, tels que l'envoi de notifications avec Amazon SNS, dans le cadre du même flux de travail.

Automatisation de l'informatique et de la sécurité

L'automatisation informatique peut aider à gérer des opérations de plus en plus complexes et chronophages, telles que la mise à niveau et l'application de correctifs logiciels, le déploiement de mises à jour de sécurité pour corriger les vulnérabilités, la sélection de l'infrastructure, la synchronisation des données, l'acheminement des tickets d'assistance, etc. L'automatisation des

tâches répétitives et chronophages peut permettre à votre organisation d'effectuer des opérations de routine rapidement et de manière cohérente à grande échelle. Cela vous permet de vous concentrer sur des tâches stratégiques telles que le développement de fonctionnalités, les demandes d'assistance complexes et l'innovation tout en répondant à ces demandes croissantes.

Step Functions vous permet de créer des flux de travail qui s'adaptent automatiquement aux besoins de votre entreprise sans nécessiter d'intervention manuelle. Dans les cas où une erreur se produit dans votre flux de travail, elle ne nécessite souvent aucune intervention manuelle. Step Functions vous permet de réessayer automatiquement les tâches qui ont échoué et d'effectuer une pause exponentielle afin de gérer les erreurs dans votre flux de travail.

Dans certaines situations, une intervention humaine est requise avant que le flux de travail puisse progresser. Par exemple, l'approbation d'une augmentation de crédit substantielle peut nécessiter une approbation humaine. Pour gérer cela, vous pouvez définir une logique de branchement dans Step Functions, de sorte que seules les demandes dépassant un montant défini nécessitent une approbation humaine, tandis que toutes les autres demandes sont automatiquement traitées. Dans les cas où une approbation humaine est requise, Step Functions vous permet de suspendre le flux de travail à une étape spécifique, d'attendre une réponse, puis de poursuivre le flux de travail une fois la réponse reçue.

Voici quelques exemples des types de flux de travail d'automatisation pour lesquels les clients utilisent Step Functions :

Automatisation informatique

- Résolvez automatiquement les incidents tels que l'ouverture d'un port SSH, le manque d'espace disque ou l'octroi d'un accès public à un compartiment Amazon S3.
- Automatisez le déploiement de AWS CloudFormation StackSets

Automatisation de la sécurité

- Automatisez la réponse à un scénario dans lequel un utilisateur et une clé d'accès utilisateur ont été exposés.
- Résolvez automatiquement les réponses aux incidents de sécurité en fonction des actions de politique définies, telles que la restriction des actions à des ARN spécifiques ou l'application d'autres actions.
- Avertissez les employés des e-mails de phishing quelques secondes après leur réception.

Approbation humaine

- Automatisez l'apprentissage du modèle d'apprentissage automatique, puis exigez l'approbation manuelle du modèle par un data scientist avant de déployer ou de rejeter automatiquement le modèle en fonction de la réponse reçue.
- Automatisez l'acheminement des commentaires des clients reçus en fonction de l'analyse des sentiments afin que ceux qui ont un sentiment négatif soient immédiatement redirigés vers un niveau supérieur pour examen manuel.

Comment fonctionne Step Functions

Cette section décrit les concepts importants pour vous aider à maîtriser AWS Step Functions et comprendre comment cela fonctionne.

Rubriques

Flux de travail standard ou express

Lorsque vous créez une machine à états, vous sélectionnez un type Standard ou Express. Le type par défaut pour les machines à états est Standard. Une machine à états dont le type est Standard est appelée flux de travail standard et une machine à états dont le type est Express est appelée flux de travail express.

Pour les flux de travail Standard et Express, vous définissez votre machine à états à l'aide duAmazon States Language. Les exécutions de vos machines à états se comporteront différemment selon le type que vous sélectionnez.

> **Important**
>
> Le type que vous choisissez ne peut pas être modifié une fois que vous avez créé la machine à états.

> **Note**
>
> Si vous définissez vos machines à états en dehors de la console Step Functions, par exemple dans un éditeur de votre choix, vous devez enregistrer les définitions de vos machines à états avec l'extension .asl.json.

Les flux de travail standard sont idéaux pour les flux de travail de longue durée (jusqu'à un an), durables et contrôlables. Vous pouvez récupérer l'historique complet des exécutions à l'aide de l'API Step Functions jusqu'à 90 jours après la fin de votre exécution. Les flux de travail standard suivent un modèle « une seule fois », dans lequel vos tâches et vos états ne sont jamais exécutés plus d'une fois, sauf si vous avez défini un `Retry` comportement en ASL. Cela rend les flux de travail standard adaptés à l'orchestration d'actions non idempotentes, telles que le démarrage d'un cluster Amazon EMR ou le traitement des paiements. Les exécutions de flux de travail standard sont facturées en fonction du nombre de transitions d'état traitées.

Les workflows express conviennent parfaitement aux charges de travail de traitement d'événements à volume élevé, telles que l'ingestion de données IoT, le traitement et la transformation des données en continu et les backends d'applications mobiles. Ils peuvent durer jusqu'à cinq minutes. Express Workflows utilise un at-least-oncemodèle dans lequel une exécution peut être exécutée plusieurs fois. Express Workflows est donc idéal pour orchestrer des actions idempotentes telles que la transformation des données d'entrée et le stockage au moyen d'une action PUT dans Amazon DynamoDB. Les exécutions d'Express Workflow sont facturées en fonction du nombre d'exécutions, de la durée d'exécution et de la mémoire consommée pendant l'exécution.

Les flux de travail standard et express peuvent démarrer automatiquement en réponse à des événements tels que les requêtes HTTP d'Amazon API Gateway (API entièrement gérées à grande échelle), les règles IoT et plus de 140 autres sources d'événements sur Amazon EventBridge.

> **Tip**
>
> Pour déployer un exemple de flux de travail Express sur votreCompte AWS, consultez le module 7 - API Gateway, Parallel State, flux de travail Express de The AWS Step Functions Workshop.

Pour plus d'informations sur l'expérience de console pour les exécutions Standard et Express Workflow, consultezExécutions de flux de travail standard et express dans la console.

Workflows standard et express

	Workflows standard	Flux de travail express : synchrones et asynchrones
Durée maximum	Un an	Cinq minutes
Taux de début d'exécution pris en charge	Pour plus d'informations sur les quotas liés au taux de démarrage d'exécution pris en charge, consultezQuotas liés à la limitation des actions des API.	Pour plus d'informations sur les quotas liés au taux de démarrage d'exécution pris en charge, consultezQuotas liés à la limitation des actions des API.
Taux de transition d'état pris en charge	Pour plus d'informations sur les quotas liés au taux de transition entre États pris en charge, voirQuotas liés à l'étranglement de l'État.	Aucune limite
Tarification	Tarification en fonction du nombre de transitions entre États. Une transition d'état est comptabilisée chaque fois qu'une étape de votre exécution est terminée.	Prix calculé en fonction du nombre d'exécutions, de leur durée et de leur consommat ion de mémoire.
Historique d'exécution	Les exécutions peuvent être répertoriées et décrites à	Historique d'exécution illimité, c'est-à-dire autant d'entrées

	Workflows standard	Flux de travail express : synchrones et asynchrones
	l'aide des API Step Functions . Les exécutions peuvent être déboguées visuellement via la console. Ils peuvent également être inspectés dans les CloudWatch journaux en activant la journalisation sur votre machine d'état. Pour plus d'informations sur le débogage des exécutions de flux de travail standard dans la console, consultez Exécutions de flux de travail standard et express dans la console etAffichage et débogage des exécutions.	d'historique d'exécution conservées que vous pouvez en générer dans un délai de 5 minutes. Les exécutions peuvent être inspectées dans CloudWatch Logs ou dans la console Step Functions en activant la journalisation sur votre machine d'état. Pour plus d'informations sur le débogage des exécutions d'Express Workflow dans la console, consultez Exécutions de flux de travail standard et express dans la console etAffichage et débogage des exécutions.
Sémantique d'exécution	Exécution du flux de travail en une seule fois.	Workflows express asynchrones : exécution d'un t-least-once flux de travail. Workflows Express synchrones : exécution d'un t-most-once flux de travail.

	Workflows standard	Flux de travail express : synchrones et asynchrones
Intégrations de service	Prend en charge toutes les intégrations de services et les modèles.	Prend en charge toutes les intégrations de services. **Note** Express Workflows ne prend pas en charge les modèles d'intégration des services Job-run (`.sync`) ou Callback (`.waitForTaskToken`).
Activités de Step Functions	Soutient les activités de Step Functions.	Ne prend pas en charge les activités Step Functions.

Flux de travail express synchrones et asynchrones

Vous pouvez choisir deux types de flux de travail express : les flux de travail express asynchrones et les flux de travail express synchrones.

- Les flux de travail express asynchrones renvoient une confirmation indiquant que le flux de travail a été démarré, mais n'attendez pas qu'il soit terminé. Pour obtenir le résultat, vous devez interroger les CloudWatch journaux du service. Vous pouvez utiliser des flux de travail express asynchrones lorsque vous n'avez pas besoin de réponse immédiate, tels que des services de messagerie ou des traitements de données dont les autres services ne dépendent pas. Vous pouvez démarrer des flux de travail express asynchrones en réponse à un événement, par un flux de travail imbriqué dans Step Functions ou en utilisant l'`StartExecution`appel d'API.
- Les flux de travail express synchrones démarrent un flux de travail, attendent qu'il soit terminé, puis renvoient le résultat. Les flux de travail express synchrones peuvent être utilisés pour orchestrer des microservices. Avec Synchronous Express Workflows, vous pouvez développer des applications sans avoir à développer de code supplémentaire pour gérer les erreurs, réessayer ou exécuter des tâches parallèles. Vous pouvez exécuter des flux de travail synchrones Express

invoqués depuis Amazon API Gateway ou en utilisant l'appel d'`StartSyncExecution`API. AWS Lambda

> **Note**
>
> Si vous exécutez Step Functions Express Workflows de manière synchrone depuis la console, la `StartSyncExecution` demande expire au bout de 60 secondes. Pour exécuter les Express Workflows de manière synchrone pendant une durée maximale de cinq minutes, effectuez la `StartSyncExecution` demande à l'aide du AWS SDK ou AWS Command Line Interface (AWS CLI) au lieu de la console Step Functions.

Les appels d'API d'exécution synchrone d'Express ne contribuent pas aux limites de capacité existantes du compte. Step Functions fournit des capacités à la demande et s'adapte automatiquement à une charge de travail soutenue. Les pics de charge de travail peuvent être limités jusqu'à ce que la capacité soit disponible.

Garanties d'exécution

Workflows standard	Flux de travail express asynchrones	Flux de travail express synchrones	
Exécution du flux de travail en une seule fois	Exécution d'un t-least-once flux de travail	Exécution d'un t-most-once flux de travail	
L'état d'exécution persiste en interne entre les transitions d'état.	L'état d'exécution ne persiste pas entre les transitions d'état.	L'état d'exécution ne persiste pas entre les transitions d'état.	
Renvoie automatiq uement une réponse idempotente au démarrage d'une exécution portant le même nom qu'un	L'impuissance n'est pas gérée automatiquement. Le démarrage de plusieurs flux de travail portant le	L'impuissance n'est pas gérée automatiq uement. Step Functions attend le début d'une exécution et renvoie le résultat	

Workflows standard	Flux de travail express asynchrones	Flux de travail express synchrones	
flux de travail en cours d'exécution. Le nouveau flux de travail ne démarre pas et une exception est déclenchée une fois que le flux de travail en cours est terminé.	même nom entraîne des exécutions simultanées. Peut entraîner une perte de l'état du flux de travail interne si la logique de la machine à états n'est pas idempoten te.	de la machine à états une fois l'exécution terminée. Les flux de travail ne redémarrent pas en cas d'excepti on.	

Workflows standard	Flux de travail express asynchrones	Flux de travail express synchrones	
Les données de l'historique d'exécution ont été supprimées après 90 jours. Les noms des flux de travail peuvent être réutilisés après la suppression des données out-of-date d'exécution. Pour répondre aux exigences de conformité, organisationnelles ou réglementaires, vous pouvez réduire la période de conservation de l'historique d'exécution à 30 jours en envoyant une demande de quota. Pour ce faire, utilisez le AWS Support Center Console et créez un nouveau boîtier.	L'historique d'exécution n'est pas enregistré par Step Functions. La journalisation doit être activée via Amazon CloudWatch Logs.	L'historique d'exécution n'est pas enregistré par Step Functions. La journalisation doit être activée via Amazon CloudWatch Logs.	

Optimisation des coûts à l'aide d'Express Workflows

Step Functions détermine le prix des flux de travail Standard et Express en fonction du type de flux de travail que vous utilisez pour créer vos machines d'état. Pour optimiser le coût de vos flux de travail sans serveur, vous pouvez suivre l'une des recommandations suivantes ou les deux :

Rubriques

Pour plus d'informations sur l'impact du choix d'un type de flux de travail Standard ou Express sur la facturation, consultez la section AWS Step FunctionsTarification.

Conseil #1 : Imbrication de flux de travail Express dans des flux de travail standard

Step Functions exécute des flux de travail dont la durée et le nombre d'étapes sont limités. Certains flux de travail peuvent être exécutés dans un court laps de temps. D'autres peuvent nécessiter une combinaison de longue durée et de high-event-rate flux de travail. Avec Step Functions, vous pouvez créer des flux de travail volumineux et complexes à partir de plusieurs flux de travail plus petits et plus simples.

Par exemple, pour créer un flux de travail de traitement des commandes, vous pouvez inclure toutes les actions non idempotentes dans un flux de travail standard. Cela pourrait inclure des actions telles que l'approbation de la commande par le biais d'une interaction humaine et le traitement des paiements. Vous pouvez ensuite combiner une série d'actions idempotentes, telles que l'envoi de notifications de paiement et la mise à jour de l'inventaire des produits, dans un flux de travail Express. Vous pouvez intégrer ce flux de travail Express au flux de travail standard. Dans cet exemple, le flux de travail standard est connu sous le nom de machine à états parent. Le flux de travail Express imbriqué est connu sous le nom de machine à états fils.

Conseil #2 : Convertissez les flux de travail standard en flux de travail Express

Vous pouvez convertir vos flux de travail standard existants en flux de travail Express s'ils répondent aux exigences suivantes :

- Le flux de travail doit être terminé dans les cinq minutes.
- Le flux de travail est conforme à un modèle at-least-onced'exécution. Cela signifie que chaque étape du flux de travail peut être exécutée plusieurs fois exactement.
- Le flux de travail n'utilise pas les `.waitForTaskToken` modèles d'intégration des `.sync` services.

> **Important**
>
> Les flux de travail Express utilisent Amazon CloudWatch Logs pour enregistrer les historiques d'exécution. L'utilisation de CloudWatch Logs entraîne des coûts supplémentaires.

Pour convertir un flux de travail standard en flux de travail express à l'aide de la console

1. Ouvrez la console Step Functions.
2. Sur la page Machines d'état, choisissez une machine d'état de type standard pour l'ouvrir.

   > **Tip**
   >
   > Dans la liste déroulante Tous les types, choisissez Standard pour filtrer la liste des machines à états et afficher uniquement les flux de travail standard.

3. Choisissez Copier vers un nouveau.

   Workflow Studio s'ouvre pour Mode de conception afficher le flux de travail de la machine à états que vous avez sélectionnée.
4. (Facultatif) Mettez à jour la conception du flux de travail.
5. Spécifiez un nom pour votre machine à états. Pour ce faire, cliquez sur l'icône d'édition à côté du nom de la machine à états par défaut de MyStateMachine. Ensuite, dans Configuration de la machine d'état, spécifiez un nom dans le champ Nom de la machine d'état.
6. (Facultatif) Dans Configuration de la machine à états, spécifiez d'autres paramètres de flux de travail, tels que le type de machine à états et son rôle d'exécution.

   Assurez-vous que pour Type, vous choisissez Express. Conservez toutes les autres sélections par défaut dans les paramètres State Machine.

   > **Note**
   >
   > Si vous convertissez un flux de travail standard défini précédemment dans AWS CDKouAWS SAM, vous devez modifier la valeur `Type` et le `Resource` nom de.
7. Dans la boîte de dialogue Confirmer la création du rôle, choisissez Confirmer pour continuer.

Vous pouvez également choisir Afficher les paramètres des rôles pour revenir à la configuration de la machine State.

> **Note**
>
> Si vous supprimez le rôle IAM créé par Step Functions, Step Functions ne pourra pas le recréer ultérieurement. De même, si vous modifiez le rôle (par exemple, en supprimant Step Functions des principes de la politique IAM), Step Functions ne pourra pas restaurer ses paramètres d'origine ultérieurement.

Pour plus d'informations sur les meilleures pratiques et les directives relatives à la gestion de l'optimisation des coûts de vos flux de travail, voir Création de AWS Step Functions flux de travail rentables.

States

Les États individuels peuvent prendre des décisions en fonction de leurs entrées, effectuer des actions à partir de ces entrées et transmettre les résultats à d'autres états. DansAWS Step Functions, vous définissez vos flux de travail dans la langue Amazon States (ASL). La console Step Functions fournit une représentation graphique de votre machine d'état pour vous aider à visualiser la logique de votre application.

> **Note**
>
> Si vous définissez vos machines d'état en dehors de la console Step Functions, par exemple dans l'éditeur de votre choix, vous devez enregistrer les définitions de vos machines d'état avec l'extension .asl.json.

Les états sont des éléments de votre machine d'État. Un état est désigné par son nom, qui peut être n'importe quelle chaîne, mais qui doit être unique au sein de toute la machine d'état.

Les états peuvent effectuer diverses fonctions dans votre machine d'état :

- Faire quelques tâches dans votre machine d'état (un état Task (Tâche)).
- Faire un choix entre les branches d'exécution (un état Choice)

- Arrêter une exécution avec un échec ou une réussite (un état Fail ou Succeed)
- Passez son entrée à sa sortie ou injectez des données fixes dans le flux de travail (état Pass)
- Prévoyez un délai pendant un certain temps, ou jusqu'à une date et une heure spécifiées (état d'attente)
- Commencer des branches parallèles d'exécution (un état Parallel)
- Itérer de manière dynamique des étapes à l'aide d'un état Map

Voici un exemple d'état nommé `HelloWorld` qui exécute une fonction AWS Lambda.

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

Les états partagent de nombreuses fonctionnalités :

- Un `Type` champ indiquant de quel type d'état il s'agit.
- `Comment`Champ facultatif contenant un commentaire lisible par l'utilisateur ou une description de l'état.
- Chaque état (à l'exception d'un état `Succeed` ou `Fail`) requiert un champ `Next` ou peut devenir un état terminal en spécifiant un champ `End`.

> **Note**
>
> Un état `Choice` peut avoir plusieurs `Next`, mais un seul au sein de chaque règle Choice. Un `Choice` État ne peut pas utiliser`End`.

Certains types d'état nécessitent des champs supplémentaires ou peuvent redéfinir l'utilisation courante du champ.

Après avoir créé et exécuté des flux de travail standard, vous pouvez accéder aux informations relatives à chaque état, à ses entrées et à ses sorties, à quel moment il était actif et pendant combien de temps, en consultant la page Détails d'exécution de la console Step Functions. Pour

plus d'informations, veuillez consulter Affichage et débogage des exécutions sur la console Step Functions.

Après avoir créé et exécuté des exécutions d'Express Workflow et si la journalisation est activée pour votre Express Workflow, vous pouvez accéder aux informations relatives à l'exécution dans Amazon CloudWatch Logs ou dans la console Step Functions. Pour plus d'informations, veuillez consulter Affichage et débogage des exécutions sur la console Step Functions.

Rubriques

Amazon States Language

Le langage Amazon States est un langage structuré basé sur JSON utilisé pour définir votre machine à états, un ensemble d'états qui peuvent fonctionner (`Task`états), déterminer les états vers lesquels passer au suivant (`Choice`états), arrêter une exécution en cas d'erreur (`Fail`états), etc.

Pour plus d'informations, consultez la spécification du Langage des états d'Amazon et Statelint, un outil qui valide le code du Langage des états d'Amazon.

Pour créer une machine d'état sur la console Step Functions à l'aide d'Amazon States Language, consultez la section Mise en route.

> **Note**
>
> Si vous définissez vos machines d'état en dehors de la console Step Functions, par exemple dans l'éditeur de votre choix, vous devez enregistrer les définitions de vos machines d'état avec l'extension .asl.json.

Exemple de spécification de langue 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
    }
  }
}
```

Rubriques

- Structure de la machine d'État
- Fonctions intrinsèques
- Champs d'état courants

Structure de la machine d'État

Les machines d'état sont définies à l'aide d'un texte JSON qui représente une structure contenant les champs suivants.

`Comment` (facultatif)

Une description lisible de la machine d'état.

`StartAt` (Obligatoire)

Une chaîne qui doit correspondre exactement (sensible à la casse) au nom de l'un des objets d'état.

`TimeoutSeconds`(Facultatif)

Le nombre maximal de secondes durant lequel une machine d'état peut être exécutée. Si elle dure plus longtemps que la durée spécifiée, l'exécution échoue avec un nom `States.Timeout` d'erreur.

`Version` (facultatif)

Version de l'Amazon States Language utilisée dans la machine à états (la valeur par défaut est « 1.0 »).

`States` (Obligatoire)

Objet contenant un ensemble d'états séparés par des virgules.

Le champ `States` contient States.

```
{
    "State1" : {
    },

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

Une machine d'état est définie par les états qu'elle contient et les relations entre ceux-ci.

Voici un exemple.

```
{
  "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
    }
  }
}
```

Lorsqu'une exécution de cette machine d'état est lancée, le système commence par l'état référencé dans le champ `StartAt` (`"HelloWorld"`). Si cet état a un champ `"End": true`, l'exécution s'arrête et renvoie un résultat. Dans le cas contraire, le système recherche un champ `"Next":` et continue avec cet état. Ce processus se répète jusqu'à ce que le système atteigne un état terminal (un état avec `"Type": "Succeed"`, `"Type": "Fail"` ou `"End": true`) ou jusqu'à ce qu'une erreur d'exécution se produise.

Les règles suivantes s'appliquent aux états au sein d'une machine d'état :

- Les états peuvent se produire dans n'importe quel ordre dans le bloc de délimitation, mais l'ordre dans lequel ils sont répertoriés n'affecte pas l'ordre dans lequel ils sont exécutés, qui est déterminé par le contenu des états eux-mêmes. Cet ordre est déterminé par le contenu des états.
- Dans une machine d'état, il ne peut y avoir qu'un seul état désigné comme l'état `start`, désigné par la valeur du champ `StartAt` de la structure de niveau supérieur. Cet état est celui qui est exécuté en premier lorsque l'exécution commence.
- Tout état dont le champ `End` est défini sur `true` est considéré comme un état `end` (ou `terminal`). Selon la logique de votre machine à états (par exemple, si votre machine à états possède plusieurs branches d'exécution), il se peut que vous ayez plusieurs états. `end`
- Si votre machine d'état est composée d'un seul état, il peut s'agir de `start` et de `end`.

Fonctions intrinsèques

L'Amazon States Language fournit plusieurs fonctions intrinsèques, également appelées intrinsèques, qui vous aident à effectuer des opérations de traitement de données de base sans utiliser d'`Task`état. Les intrinsèques sont des constructions qui ressemblent aux fonctions des langages de programmation. Ils peuvent être utilisés pour aider les constructeurs de charges utiles à traiter les données à destination et en provenance du `Resource` champ d'un `Task` État.

Dans Amazon States Language, les fonctions intrinsèques sont regroupées dans les catégories suivantes, en fonction du type de tâche de traitement des données que vous souhaitez effectuer :

- Intrinsèques pour les réseaux
- Intrinsèques pour le codage et le décodage des données
- Intrinsèque pour le calcul du hachage
- Intrinsèques pour la manipulation des données JSON
- Intrinsèques pour les opérations mathématiques
- Intrinsèque pour le fonctionnement des chaînes
- Intrinsèque pour la génération d'identifiants uniques
- Intrinsèque pour un fonctionnement générique

Note

- Pour utiliser des fonctions intrinsèques, vous devez spécifier `.$` la valeur clé dans les définitions de votre machine à états, comme indiqué dans l'exemple suivant :

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

- Vous pouvez intégrer jusqu'à 10 fonctions intrinsèques dans un champ de vos flux de travail. L'exemple suivant montre un champ nommé *myArn* qui inclut neuf fonctions intrinsèques imbriquées :

```
"myArn.$": "States.Format('{}.{}.{}',
 States.ArrayGetItem(States.StringSplit(States.ArrayGetItem(States.StringSplit($.ImageRe
 '/'), 2), '.'), 0),
 States.ArrayGetItem(States.StringSplit(States.ArrayGetItem(States.StringSplit($.ImageRe
 '/'), 2), '.'), 1))"
```

Tip

Si vous utilisez Step Functions dans un environnement de développement local, assurez-vous d'utiliser la version 1.12.0 ou supérieure pour pouvoir inclure toutes les fonctions intrinsèques dans vos flux de travail.

Champs prenant en charge les fonctions intrinsèques

Le tableau suivant indique quels champs prennent en charge les fonctions intrinsèques pour chaque état.

Champs prenant en charge les fonctions intrinsèques

State	Passe	Tâche	Choix	Attente	Réussir	Échouer	Parallèle	Map
InputPath								

State	Passe	Tâche	Choix	Attente	Réussir	Échouer	Parallèle	Map
Paramètres	✓	✓					✓	✓
ResultSelector		✓					✓	✓
ResultPath								
OutputPath								
Variable								
<ComparisonOperator>								
Chemin								
TimeoutSecondsPath								
HeartbeatSecondsPath								
Informations d'identification		✓						

Intrinsèques pour les réseaux

Utilisez les éléments intrinsèques suivants pour effectuer des manipulations de tableaux.

`States.Array`

La fonction `States.Array` intrinsèque ne prend aucun argument ou plus. L'interpréteur renvoie un tableau JSON contenant les valeurs des arguments dans l'ordre indiqué. Par exemple, avec les données d'entrée suivantes :

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

Vous pourriez utiliser

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

Ce qui renverrait le résultat suivant :

```
“BuildId”: [123456]
```

`States.ArrayPartition`

Utilisez la fonction `States.ArrayPartition` intrinsèque pour partitionner un grand tableau. Vous pouvez également utiliser cet élément intrinsèque pour découper les données, puis envoyer la charge utile en petits morceaux.

Cette fonction intrinsèque prend deux arguments. Le premier argument est un tableau, tandis que le second définit la taille du bloc. L'interpréteur découpe le tableau d'entrée en plusieurs tableaux de la taille spécifiée par la taille du morceau. La longueur du dernier segment de tableau peut être inférieure à la longueur des fragments de tableau précédents si le nombre d'éléments restants dans le tableau est inférieur à la taille du morceau.

Validation des entrées

- Vous devez spécifier un tableau comme valeur d'entrée pour le premier argument de la fonction.
- Vous devez spécifier un entier positif différent de zéro pour le deuxième argument représentant la valeur de la taille du segment.

  Si vous spécifiez une valeur non entière pour le deuxième argument, Step Functions l'arrondira à l'entier le plus proche.

- Le tableau d'entrée ne peut pas dépasser la limite de charge utile de 256 Ko fixée par Step Functions.

Par exemple, étant donné le tableau d'entrée suivant :

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

Vous pouvez utiliser cette `States.ArrayPartition` fonction pour diviser le tableau en morceaux de quatre valeurs :

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

Ce qui renverrait les fragments de tableau suivants :

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

Dans l'exemple précédent, la `States.ArrayPartition` fonction produit trois tableaux. Les deux premiers tableaux contiennent chacun quatre valeurs, telles que définies par la taille des morceaux. Un troisième tableau contient la valeur restante et est inférieur à la taille de bloc définie.

`States.ArrayContains`

Utilisez la fonction `States.ArrayContains` intrinsèque pour déterminer si une valeur spécifique est présente dans un tableau. Par exemple, vous pouvez utiliser cette fonction pour détecter s'il y a eu une erreur lors d'une itération d'`Map`état.

Cette fonction intrinsèque prend deux arguments. Le premier argument est un tableau, tandis que le second est la valeur à rechercher dans le tableau.

Validation des entrées

- Vous devez spécifier un tableau comme valeur d'entrée pour le premier argument de la fonction.
- Vous devez spécifier un objet JSON valide comme deuxième argument.
- Le tableau d'entrée ne peut pas dépasser la limite de charge utile de 256 Ko fixée par Step Functions.

Par exemple, étant donné le tableau d'entrée suivant :

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

Vous pouvez utiliser la `States.ArrayContains` fonction pour trouver la `lookingFor` valeur dans `inputArray` :

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

Comme la valeur stockée dans `lookingFor` est incluse dans le`inputArray`, `States.ArrayContains` renvoie le résultat suivant :

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

`States.ArrayRange`

Utilisez la fonction `States.ArrayRange` intrinsèque pour créer un nouveau tableau contenant une plage spécifique d'éléments. Le nouveau tableau peut contenir jusqu'à 1 000 éléments.

Cette fonction prend trois arguments. Le premier argument est le premier élément du nouveau tableau, le deuxième est le dernier élément du nouveau tableau et le troisième argument est la valeur d'incrément entre les éléments du nouveau tableau.

Validation des entrées

- Vous devez spécifier des valeurs entières pour tous les arguments.

  Si vous spécifiez une valeur non entière pour l'un des arguments, Step Functions l'arrondira à l'entier le plus proche.
- Vous devez spécifier une valeur différente de zéro pour le troisième argument.
- Le tableau nouvellement généré ne peut pas contenir plus de 1 000 éléments.

Par exemple, l'utilisation suivante de la `States.ArrayRange` fonction créera un tableau avec une première valeur de 1, une valeur finale de 9, et les valeurs comprises entre les première et dernière valeurs augmenteront de deux pour chaque élément :

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

Ce qui renverrait le tableau suivant :

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

States.ArrayGetItem

Cette fonction intrinsèque renvoie la valeur d'un indice spécifié. Cette fonction prend deux arguments. Le premier argument est un tableau de valeurs et le second est l'index du tableau de la valeur à renvoyer.

Par exemple, utilisez les `index` valeurs `inputArray` et suivantes :

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

À partir de ces valeurs, vous pouvez utiliser la `States.ArrayGetItem` fonction pour renvoyer la valeur à la `index` position 5 dans le tableau :

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

Dans cet exemple, `States.ArrayGetItem` renverrait le résultat suivant :

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

States.ArrayLength

La fonction `States.ArrayLength` intrinsèque renvoie la longueur d'un tableau. Il a un argument, le tableau dont la longueur doit être renvoyée.

Par exemple, étant donné le tableau d'entrée suivant :

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

Vous pouvez l'utiliser `States.ArrayLength` pour renvoyer la longueur de `inputArray` :

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

Dans cet exemple, `States.ArrayLength` renverrait l'objet JSON suivant qui représente la longueur du tableau :

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

`States.ArrayUnique`

La fonction `States.ArrayUnique` intrinsèque supprime les valeurs dupliquées d'un tableau et renvoie un tableau contenant uniquement des éléments uniques. Cette fonction prend un tableau, qui peut être détrié, comme seul argument.

Par exemple, le texte suivant `inputArray` contient une série de valeurs dupliquées :

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

Vous pouvez utiliser la `States.ArrayUnique` fonction as et spécifier le tableau dont vous souhaitez supprimer les doublons :

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

La `States.ArrayUnique` fonction renverrait le tableau suivant contenant uniquement des éléments uniques, en supprimant toutes les valeurs dupliquées :

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

Intrinsèques pour le codage et le décodage des données

Utilisez les fonctions intrinsèques suivantes pour coder ou décoder des données selon le schéma de codage Base64.

`States.Base64Encode`

Utilisez la fonction `States.Base64Encode` intrinsèque pour coder les données selon le schéma de codage MIME Base64. Vous pouvez utiliser cette fonction pour transmettre des données à d'autres AWS services sans utiliser de AWS Lambda fonction.

Cette fonction prend comme seul argument une chaîne de données de 10 000 caractères maximum à coder.

Par exemple, considérez la `input` chaîne suivante :

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

Vous pouvez utiliser la `States.Base64Encode` fonction pour encoder la `input` chaîne en tant que chaîne MIME Base64 :

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

La `States.Base64Encode` fonction renvoie les données codées suivantes en réponse :

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

`States.Base64Decode`

Utilisez la fonction `States.Base64Decode` intrinsèque pour décoder les données selon le schéma de décodage MIME Base64. Vous pouvez utiliser cette fonction pour transmettre des données à d'autres AWS services sans utiliser de fonction Lambda.

Cette fonction prend comme seul argument une chaîne de données codée en Base64 de 10 000 caractères maximum à décoder.

Par exemple, avec les données d'entrée suivantes :

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

Vous pouvez utiliser la `States.Base64Decode` fonction pour décoder la chaîne base64 en une chaîne lisible par l'homme :

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

Ils `States.Base64Decode function` renverraient les données décodées suivantes en réponse :

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

Intrinsèque pour le calcul du hachage

`States.Hash`

Utilisez la fonction `States.Hash` intrinsèque pour calculer la valeur de hachage d'une entrée donnée. Vous pouvez utiliser cette fonction pour transmettre des données à d'autres AWS services sans utiliser de fonction Lambda.

Cette fonction prend deux arguments. Le premier argument concerne les données dont vous souhaitez calculer la valeur de hachage. Le deuxième argument est l'algorithme de hachage à utiliser pour effectuer le calcul du hachage. Les données que vous fournissez doivent être une chaîne d'objets contenant 10 000 caractères ou moins.

L'algorithme de hachage que vous spécifiez peut être l'un des algorithmes suivants :

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

Par exemple, vous pouvez utiliser cette fonction pour calculer la valeur de hachage de la `Data` chaîne à l'aide de la valeur spécifiée `Algorithm` :

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

Vous pouvez utiliser la `States.Hash` fonction pour calculer la valeur de hachage :

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

La `States.Hash` fonction renvoie la valeur de hachage suivante en réponse :

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

Intrinsèques pour la manipulation des données JSON

Utilisez ces fonctions pour effectuer des opérations de traitement de données de base sur des objets JSON.

`States.JsonMerge`

Utilisez la fonction `States.JsonMerge` intrinsèque pour fusionner deux objets JSON en un seul objet. Cette fonction prend trois arguments. Les deux premiers arguments sont les objets JSON que vous souhaitez fusionner. Le troisième argument est une valeur booléenne de. `false` Cette valeur booléenne détermine si le mode de fusion profonde est activé.

Step Functions ne prend actuellement en charge que le mode de fusion superficielle ; vous devez donc spécifier la valeur booléenne sous la forme. `false` En mode superficiel, si la même clé existe dans les deux objets JSON, la clé du dernier objet remplace la même clé dans le premier objet. De plus, les objets imbriqués dans un objet JSON ne sont pas fusionnés lorsque vous utilisez une fusion superficielle.

Par exemple, vous pouvez utiliser la `States.JsonMerge` fonction pour fusionner les objets JSON suivants qui partagent la cléa.

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

Vous pouvez spécifier les objets json1 et json2 comme entrées dans la `States.JsonMerge` fonction pour les fusionner :

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

`States.JsonMerge`Renvoie l'objet JSON fusionné suivant en conséquence. Dans l'objet JSON fusionné`output`, la clé de l'`json2`objet a remplace la clé de l'`json1`objeta. De plus, l'objet imbriqué dans la clé de `json1` l'objet a est supprimé car le mode superficiel ne permet pas de fusionner des objets imbriqués.

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

```
      "c": 3
   }
}
```

States.StringToJson

La `States.StringToJson` fonction prend comme seul argument un chemin de référence vers une chaîne JSON échappée.

L'interpréteur applique un analyseur JSON et renvoie le formulaire JSON analysé de l'entrée. Par exemple, vous pouvez utiliser cette fonction pour échapper à la chaîne de saisie suivante :

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

Utilisez la `States.StringToJson` fonction et spécifiez le `escapedJsonString` comme argument d'entrée :

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

La `States.StringToJson` fonction renvoie le résultat suivant :

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

States.JsonToString

La `States.JsonToString` fonction ne prend qu'un seul argument, à savoir le chemin qui contient les données JSON à renvoyer sous forme de chaîne non échappée. L'interpréteur renvoie une chaîne contenant du texte JSON représentant les données spécifiées par le chemin. Par exemple, vous pouvez fournir le chemin JSON suivant contenant une valeur échappée :

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

Fournissez à la `States.JsonToString` fonction les données contenues dans `unescapedJson` :

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

La `States.JsonToString` fonction renvoie la réponse suivante :

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

Intrinsèques pour les opérations mathématiques

Utilisez ces fonctions pour effectuer des opérations mathématiques.

`States.MathRandom`

Utilisez la fonction `States.MathRandom` intrinsèque pour renvoyer un nombre aléatoire compris entre le numéro de début (inclus) et le numéro de fin (exclusif) spécifiés.

Vous pouvez utiliser cette fonction pour répartir une tâche spécifique entre deux ressources ou plus.

Cette fonction prend trois arguments. Le premier argument est le numéro de début, le deuxième est le numéro de fin et le dernier argument contrôle la valeur initiale. L'argument valeur initiale est facultatif. Si vous utilisez cette fonction avec la même valeur initiale, elle renvoie un nombre identique.

> **Important**
>
> Comme la `States.MathRandom` fonction ne renvoie pas de nombres aléatoires sécurisés par cryptographie, nous vous recommandons de ne pas l'utiliser pour les applications sensibles en termes de sécurité.

Validation des entrées

- Vous devez spécifier des valeurs entières pour les arguments du numéro de début et du numéro de fin.

  Si vous spécifiez une valeur non entière pour l'argument du numéro de début ou de fin, Step Functions l'arrondira à l'entier le plus proche.

Par exemple, pour générer un nombre aléatoire compris entre un et 999, vous pouvez utiliser les valeurs d'entrée suivantes :

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

Pour générer le nombre aléatoire, fournissez les end valeurs `start` et à la `States.MathRandom` fonction :

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

La `States.MathRandom` fonction renvoie le nombre aléatoire suivant en réponse :

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

States.MathAdd

Utilisez la fonction `States.MathAdd` intrinsèque pour renvoyer la somme de deux nombres. Par exemple, vous pouvez utiliser cette fonction pour incrémenter des valeurs dans une boucle sans appeler de fonction Lambda.

Validation des entrées

- Vous devez spécifier des valeurs entières pour tous les arguments.

  Si vous spécifiez une valeur non entière pour l'un des arguments ou pour les deux, Step Functions l'arrondira à l'entier le plus proche.
- Vous devez spécifier des valeurs entières comprises entre -2147483648 et 2147483647.

Par exemple, vous pouvez utiliser les valeurs suivantes pour soustraire un de 111 :

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

Utilisez ensuite la `States.MathAdd` fonction définissant `value1` comme valeur de départ et `step` comme valeur à incrémenter `value1` de :

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

La `States.MathAdd` fonction renverrait le nombre suivant en réponse :

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

Intrinsèque pour le fonctionnement des chaînes

`States.StringSplit`

Utilisez la fonction `States.StringSplit` intrinsèque pour diviser une chaîne en un tableau de valeurs. Cette fonction prend deux arguments. Le premier argument est une chaîne et le second est le caractère de délimitation que la fonction utilisera pour diviser la chaîne.

Example - Diviser une chaîne d'entrée en utilisant un seul caractère de délimitation

Pour cet exemple, utilisez `States.StringSplit` pour diviser ce qui suit`inputString`, qui contient une série de valeurs séparées par des virgules :

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

Utilisez la `States.StringSplit` fonction et définissez `inputString` comme premier argument, et le caractère de délimitation `splitter` comme second argument :

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

La `States.StringSplit` fonction renvoie le tableau de chaînes suivant comme résultat :

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

Example - Diviser une chaîne d'entrée à l'aide de plusieurs caractères de délimitation

Pour cet exemple, utilisez `States.StringSplit` pour diviser ce qui suit`inputString`, qui contient plusieurs caractères de délimitation :

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

Utilisez la `States.StringSplit` fonction comme suit :

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

La `States.StringSplit` fonction renvoie le tableau de chaînes suivant comme résultat :

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

Intrinsèque pour la génération d'identifiants uniques

`States.UUID`

Utilisez la fonction `States.UUID` intrinsèque pour renvoyer un identifiant unique universel (UUID v4) de version 4 généré à l'aide de nombres aléatoires. Par exemple, vous pouvez utiliser cette fonction pour appeler d'autres AWS services ou ressources nécessitant un paramètre UUID ou pour insérer des éléments dans une table DynamoDB.

La `States.UUID` fonction est appelée sans qu'aucun argument ne soit spécifié :

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

La fonction renvoie un UUID généré de manière aléatoire, comme dans l'exemple suivant :

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

Intrinsèque pour un fonctionnement générique

`States.Format`

Utilisez la fonction `States.Format` intrinsèque pour construire une chaîne à partir de valeurs littérales et interpolées. Cette fonction prend un ou plusieurs arguments. La valeur du premier argument doit être une chaîne et peut inclure zéro ou plusieurs instances de la séquence de caractères`{}`. Il doit rester autant d'arguments dans l'invocation de l'intrinsèque qu'il y a d'occurrences de`{}`. L'interpréteur renvoie la chaîne définie dans le premier argument, chacune étant `{}` remplacée par la valeur de l'argument correspondant à la position dans l'invocation intrinsèque.

Par exemple, vous pouvez utiliser les entrées suivantes parmi celles d'un individu`name`, ainsi qu'une `template` phrase dans laquelle insérer son nom :

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

Utilisez la `States.Format` fonction et spécifiez la `template` chaîne et la chaîne à insérer à la place des `{}` caractères :

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

or

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

Avec l'une des entrées précédentes, la `States.Format` fonction renvoie la chaîne complète en réponse :

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

Caractères réservés dans les fonctions intrinsèques

Les caractères suivants sont réservés aux fonctions intrinsèques et doivent être évités par une barre oblique inverse (« \ ») si vous souhaitez qu'ils apparaissent dans la valeur : ' { } et. \

Si le caractère \ doit apparaître dans la valeur sans servir de caractère d'échappement, vous devez y échapper par une barre oblique inverse. Les séquences de caractères échappées suivantes sont utilisées avec des fonctions intrinsèques :

- La chaîne littérale `\'` représente`'`.
- La chaîne littérale `\{` représente`{`.
- La chaîne littérale `\}` représente`}`.
- La chaîne littérale `\\` représente`\`.

En JSON, les barres obliques inverses contenues dans une valeur littérale de chaîne doivent être évitées par une autre barre oblique inverse. La liste équivalente pour JSON est la suivante :

- La chaîne échappée `\\\'` représente`\'`.
- La chaîne échappée `\\\{` représente`\{`.
- La chaîne échappée `\\\}` représente`\}`.
- La chaîne échappée `\\\\` représente`\\`.

> **Note**
>
> Si une barre oblique inverse ouverte \ est détectée dans la chaîne d'invocation intrinsèque, l'interpréteur renvoie une erreur d'exécution.

Champs d'état courants

`Type` (Obligatoire)

Type de l'état.

`Next`

Nom de l'état suivant qui sera exécuté lorsque l'état actuel sera terminé. Certains types d'états, par exemple `Choice`, permettent plusieurs états de transition.

Si l'état actuel est le dernier état de votre flux de travail ou un état de terminal, tel que Succeed ouFail, vous n'avez pas besoin de spécifier le `Next` champ.

`End`

Désigne cet état comme un état terminal (il termine l'exécution) s'il est défini sur `true`. Il peut y avoir n'importe quel nombre d'états terminaux par machine d'état. Un seul parmi `Next` ou `End` peut être utilisé dans un état. Certains types d'états, tels que`Choice`, ou les états de terminal, tels que Succeed etFail, ne prennent pas en charge ou n'utilisent pas `End` ce champ.

`Comment` (facultatif)

Contient une description lisible de l'état.

`InputPath` (facultatif)

Un chemin sélectionne une partie de l'entrée de l'état à transmettre à la tâche de l'état pour traitement. S'il est omis, il a la valeur $ qui désigne l'entrée complète. Pour plus d'informations, consultez Traitement des entrées et des sorties.

`OutputPath` (facultatif)

Un chemin qui sélectionne une partie de la sortie de l'état à transmettre à l'état suivant. S'il est omis, il possède la valeur $ qui désigne l'ensemble de la sortie. Pour plus d'informations, consultez Traitement des entrées et des sorties.

Pass

Un état `Pass` (`"Type": "Pass"`) transmet simplement son entrée à sa sortie, sans qu'aucun travail ne soit effectué. Les états `Pass` sont utiles lors de la construction et du débogage des machines d'état.

Vous pouvez également utiliser un `Pass` état pour transformer la saisie d'état JSON à l'aide de filtres, puis passer les données transformées à l'état suivant de vos flux de travail. Pour plus d'informations sur la transformation des entrées, reportez-vous à la sectionInputPath, Paramètres et ResultSelector.

Outre les champs d'état courants, les états `Pass` autorisent les champs suivants.

`Result` (facultatif)

Fait référence à la sortie d'une tâche virtuelle qui est transmise à l'état suivant. Si vous incluez le `ResultPath` champ dans la définition de votre machine d'état, `Result` il est placé comme spécifié par `ResultPath` et transmis à l'état suivant.

`ResultPath` (facultatif)

Spécifie où placer la sortie (par rapport à l'entrée) de la tâche virtuelle spécifiée dans`Result`. Les entrées sont ensuite filtrées telles que spécifiées par le champ `OutputPath` (s'il est présent) avant d'être utilisées comme sortie de l'état. Pour plus d'informations, consultez Traitement des entrées et des sorties.

`Parameters` (facultatif)

Crée une collection de paires clé-valeur qui seront transmises en entrée. Vous pouvez la spécifier `Parameters` sous forme de valeur statique ou la sélectionner dans l'entrée à l'aide d'un chemin. Pour plus d'informations, veuillez consulter InputPath, Paramètres et ResultSelector.

Exemple d'état Pass

Voici un exemple d'un état `Pass` qui injecte certaines données fixes dans la machine d'état, probablement à des fins de test.

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

Supposons que l'entrée de cet état soit le suivant.

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

La sortie serait celle-ci.

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

```
  }
}
```

État de la tâche

Un état `Task` (`"Type": "Task"`) représente une seule unité de travail effectuée par une machine d'état. Une tâche exécute un travail en utilisant une activité ou une AWS Lambda fonction, en s'intégrant à d'autres applications prises en charge Services AWS ou en invoquant une API tierce, telle que Stripe.

L'Amazon States Language représente les tâches en définissant le type d'état `Task` et en fournissant à la tâche le nom de ressource Amazon (ARN) de l'activité, la fonction Lambda ou le point de terminaison d'API tiers. La définition d'état de tâche suivante invoque une fonction Lambda nommée. *`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
}
```

Dans cette rubrique

- Types de tâches
- Champs d'état des tâches
- Exemples de définition de l'état des tâches
- Activités

Types de tâches

Step Functions prend en charge les types de tâches suivants que vous pouvez spécifier dans une définition d'état de tâche :

- Activité

- Fonctions Lambda
- A pris en charge Service AWS
- Une tâche HTTP

Vous spécifiez un type de tâche en fournissant son ARN dans le `Resource` champ de définition de l'état d'une tâche. L'exemple suivant montre la syntaxe du `Resource` champ. Tous les types de tâches, à l'exception de celui qui appelle une API tierce, utilisent la syntaxe suivante. Pour plus d'informations sur la syntaxe de la tâche HTTP, consultezAppelez des API tierces.

Dans la définition de l'état de votre tâche, remplacez le texte en italique dans la syntaxe suivante par les informations spécifiques à la AWS ressource.

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

La liste suivante décrit les différents composants de cette syntaxe :

- `partition`est la AWS Step Functions partition à utiliser le plus souvent`aws`.
- `service`indique la valeur Service AWS utilisée pour exécuter la tâche et peut prendre l'une des valeurs suivantes :
  - `states` pour une activité.
  - `lambda`pour une fonction Lambda. Si vous effectuez une intégration avec d'autres Services AWS, par exemple Amazon SNS ou Amazon DynamoDB, utilisez ou. `sns` `dynamodb`
- `region`est le code de AWS région dans lequel l'activité Step Functions ou le type de machine à états, la fonction Lambda ou toute autre AWS ressource ont été créés.
- `account`est l' Compte AWS ID sous lequel vous avez défini la ressource.
- `task_type` est le type de tâche à exécuter. Il peut avoir l'une des valeurs suivantes :
  - `activity`— Une activité.
  - `function`— Une fonction Lambda.
  - *`servicename`*— Le nom d'un service connecté pris en charge (voirIntégrations optimisées pour Step Functions).
- `name`est le nom de la ressource enregistrée (nom de l'activité, nom de la fonction Lambda ou action de l'API de service).

> **Note**
>
> Step Functions ne prend pas en charge le référencement des ARN entre les partitions ou les régions. Par exemple, `aws-cn` impossible d'invoquer des tâches dans la `aws` partition, et inversement.

Les sections suivantes fournissent plus de détails sur chaque type.

Activité

Les activités représentent les applications de travail (procédés ou threads), implémentées et hébergées par vous, qui effectuent une tâche spécifique. Ils sont pris en charge uniquement par les workflows standard, mais pas par les workflows express.

Les ARN `Resource` de l'activité utilisent la syntaxe suivante.

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

> **Note**
>
> Vous devez créer des activités avec Step Functions (à l'aide d'une CreateActivityaction d'API ou de la console Step Functions) avant leur première utilisation.

Pour plus d'informations sur la création d'une activité et la mise en œuvre de programmes exécutants, consultez la section Activités.

Fonctions Lambda

Les tâches Lambda exécutent une fonction en utilisant. AWS Lambda Pour spécifier une fonction Lambda, utilisez l'ARN de la fonction Lambda dans le champ. `Resource`

Selon le type d'intégration (intégration optimisée ou intégration AWS SDK) que vous utilisez pour spécifier une fonction Lambda, la syntaxe du champ de votre fonction Lambda varie. `Resource`

La syntaxe de `Resource` champ suivante est un exemple d'intégration optimisée avec une fonction Lambda.

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

La syntaxe de `Resource` champ suivante est un exemple d'intégration d'un AWS SDK avec une fonction Lambda.

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

La définition `Task` d'état suivante montre un exemple d'intégration optimisée avec une fonction Lambda nommée. *`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"
}
```

Une fois que la fonction Lambda spécifiée dans le `Resource` champ est terminée, sa sortie est envoyée à l'état identifié dans le `Next` champ (» NextState «).

A pris en charge Service AWS

Lorsque vous référencez une ressource connectée, Step Functions appelle directement les actions d'API d'un service pris en charge. Spécifiez le service et l'action dans le champ `Resource`.

Les ARN `Resource` de service connectés utilisent la syntaxe suivante.

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

> **Note**
>
> Pour créer une connexion synchrone connectée à une ressource, ajoutez `.sync` à l'entrée *`APIname`* dans l'ARN. Pour plus d'informations, consultez Utilisation avec d'autres services.

Par exemple :

```
{
```

```
 "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
    }
  }
}
```

Champs d'état des tâches

Outre les champs d'état courants, les états `Task` ont les champs suivants.

`Resource` (Obligatoire)

URI, en particulier un ARN qui identifie de manière unique la tâche spécifique à exécuter.

`Parameters` (facultatif)

Utilisé pour transmettre des informations aux actions d'API de ressources connectées. Les paramètres peuvent utiliser une combinaison de JSON statique et JsonPath. Pour plus d'informations, consultez Transmettre des paramètres à une API de service.

`Credentials` (facultatif)

Spécifie un rôle cible que le rôle d'exécution de la machine à états doit assumer avant d'invoquer le rôle spécifié`Resource`. Vous pouvez également spécifier une valeur JSONPath ou une fonction intrinsèque qui se résout en un ARN de rôle IAM au moment de l'exécution en fonction de l'entrée d'exécution. Si vous spécifiez une valeur JSONPath, vous devez la préfixer avec la notation. **`$.`**

Pour des exemples d'utilisation de ce champ dans l'`Task`État, consultezExemples de champs d'informations d'identification de l'état de la tâche. Pour un exemple d'utilisation de ce champ pour

accéder à une AWS ressource entre comptes depuis votre machine d'état, consultezTutoriel : Accès aux ressources multicomptes AWS.

> **Note**
>
> Ce champ est pris en charge par les fonctions Lambda Types de tâches qui utilisent des fonctions Lambda et un service pris en charge AWS.

`ResultPath` (facultatif)

Indique où (dans l'entrée) placer les résultats de l'exécution de la tâche spécifiée dans `Resource`. Les entrées sont ensuite filtrées telles que spécifiées par le champ `OutputPath` (s'il est présent) avant d'être utilisées comme sortie de l'état. Pour plus d'informations, consultez Traitement des entrées et des sorties.

`ResultSelector` (facultatif)

Transmettez une collection de paires clé-valeur, où les valeurs sont statiques ou sélectionnées à partir du résultat. Pour plus d'informations, consultez ResultSelector.

`Retry` (facultatif)

Tableau d'objets, nommés Réessayeurs, qui définissent une stratégie de nouvelle tentative si l'état rencontre des erreurs d'exécution. Pour plus d'informations, consultez Exemples de machines à états utilisant Retry et Catch.

`Catch` (facultatif)

Tableau d'objets, nommés Receveurs, qui définissent un état de secours. Cet état est exécuté lorsque l'état rencontre des erreurs d'exécution et que sa stratégie de nouvelle tentative est épuisée ou n'est pas définie. Pour plus d'informations, consultez États de secours.

`TimeoutSeconds` (facultatif)

Spécifie la durée maximale pendant laquelle une activité ou une tâche peut s'exécuter avant qu'elle n'expire en raison de l'States.Timeouterreur et qu'elle échoue. La valeur du délai d'attente doit être un entier positif différent de zéro. La valeur par défaut est 99999999.

Le délai d'expiration commence après le démarrage d'une tâche, par exemple lorsque `ActivityStarted` des `LambdaFunctionStarted` événements sont enregistrés dans l'historique des événements d'exécution. Pour les activités, le décompte commence à la

`GetActivityTask` réception d'un jeton et `ActivityStarted` est enregistré dans l'historique des événements d'exécution.

Lorsqu'une tâche démarre, Step Functions attend une réponse de réussite ou d'échec de la part du responsable de la tâche ou de l'activité dans le délai spécifié`TimeoutSeconds`. Si le responsable de la tâche ou de l'activité ne répond pas dans ce délai, Step Functions marque l'exécution du flux de travail comme ayant échoué.

`TimeoutSecondsPath` (facultatif)

Si vous souhaitez fournir une valeur de délai d'attente de manière dynamique à partir de l'entrée d'état à l'aide d'un chemin de référence, utilisez`TimeoutSecondsPath`. Une fois résolu, le chemin de référence doit sélectionner les champs dont les valeurs sont des entiers positifs.

> **Note**
>
> Un `Task` état ne peut pas inclure à la fois `TimeoutSeconds` et`TimeoutSecondsPath`.

`HeartbeatSeconds` (facultatif)

Détermine la fréquence des signaux de battement de cœur envoyés par un intervenant pendant l'exécution d'une tâche. Les battements de cœur indiquent qu'une tâche est toujours en cours d'exécution et qu'elle a besoin de plus de temps pour être terminée. Les battements cardiaques empêchent l'exécution d'une activité ou d'une tâche `TimeoutSeconds` pendant la durée prévue.

`HeartbeatSeconds`doit être un entier positif, différent de zéro, inférieur à la valeur du `TimeoutSeconds` champ. La valeur par défaut est `99999999`. Si plus de temps que les secondes spécifiées s'écoulent entre les pulsations de la tâche, l'état de la tâche échoue avec une erreur. States.Timeout

Pour les activités, le décompte commence à la `GetActivityTask` réception d'un jeton et `ActivityStarted` est enregistré dans l'historique des événements d'exécution.

`HeartbeatSecondsPath` (facultatif)

Si vous souhaitez fournir une valeur de battement cardiaque de manière dynamique à partir de l'entrée d'état à l'aide d'un chemin de référence, utilisez`HeartbeatSecondsPath`. Une fois résolu, le chemin de référence doit sélectionner les champs dont les valeurs sont des entiers positifs.

> **Note**
>
> Un `Task` état ne peut pas inclure à la fois `HeartbeatSeconds` et`HeartbeatSecondsPath`.

Un état `Task` doit définir le champ `End` sur `true` si l'état termine l'exécution ou doit fournir un état dans le champ `Next` qui sera exécuté lors de la fin de l'état `Task`.

Exemples de définition de l'état des tâches

Les exemples suivants montrent comment vous pouvez spécifier la définition de l'état de la tâche en fonction de vos besoins.

- Spécification de l'état des tâches, des délais d'expiration et des intervalles de pulsation
  - Exemple de délai d'expiration statique et de notification du rythme cardiaque
  - Exemple de délai d'expiration d'une tâche dynamique et de notification du rythme cardiaque
- Utilisation du champ Informations d'identification
  - Spécification de l'ARN du rôle IAM codé en dur
  - Spécifier JSONPath comme ARN du rôle IAM
  - Spécification d'une fonction intrinsèque en tant qu'ARN du rôle IAM

État des tâches, délais d'expiration et intervalles entre les pulsations

Il est recommandé de définir un délai d'expiration et un intervalle de pulsation pour les activités de longue durée. Cela peut être fait en spécifiant le délai d'expiration et les valeurs du rythme cardiaque, ou en les réglant dynamiquement.

Exemple de délai d'expiration statique et de notification du rythme cardiaque

Une fois `HelloWorld` terminé, l'état suivant (appelé ici `NextState`) est exécuté.

Si cette tâche échoue dans un délai de 300 secondes ou n'envoie pas de notifications relatives aux pulsations par intervalles de 60 secondes, la tâche est marquée comme `failed`.

```
"ActivityState": {
  "Type": "Task",
  "Resource": "arn:aws:states:us-east-1:123456789012:activity:HelloWorld",
```

```
  "TimeoutSeconds": 300,
  "HeartbeatSeconds": 60,
  "Next": "NextState"
}
```

Exemple de délai d'expiration d'une tâche dynamique et de notification du rythme cardiaque

Dans cet exemple, lorsque le AWS Glue travail est terminé, l'état suivant sera exécuté.

Si cette tâche ne s'exécute pas dans l'intervalle défini dynamiquement par la AWS Glue tâche, elle est marquée comme`failed`.

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

  "Next": "NextState"
}
```

Exemples de champs d'informations d'identification de l'état de la tâche

Spécification de l'ARN du rôle IAM codé en dur

L'exemple suivant spécifie un rôle IAM cible que le rôle d'exécution d'une machine à états doit assumer pour accéder à une fonction Lambda entre comptes nommée. `Echo` Dans cet exemple, l'ARN du rôle cible est spécifié sous forme de valeur codée en dur.

```
{
  "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
    }
  }
}
```

Spécifier JSONPath comme ARN du rôle IAM

L'exemple suivant spécifie une valeur JSONPath, qui sera convertie en un ARN de rôle IAM lors de l'exécution.

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

Spécification d'une fonction intrinsèque en tant qu'ARN du rôle IAM

L'exemple suivant utilise la fonction `States.Format`intrinsèque, qui se résout en un ARN de rôle IAM au moment de l'exécution.

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

```
}
```

Activités

Les activités sont une AWS Step Functions fonctionnalité qui vous permet de disposer d'une tâche sur votre machine d'État dans laquelle le travail est effectué par un travailleur et qui peut être hébergée sur Amazon Elastic Compute Cloud (Amazon EC2), Amazon Elastic Container Service (Amazon ECS), sur des appareils mobiles, pratiquement n'importe où.

Présentation

Dans AWS Step Functions, les activités sont un moyen d'associer du code s'exécutant à un endroit (appelé travail d'activité) à une tâche spécifique dans une machine d'état. Vous pouvez créer une activité à l'aide de la console Step Functions ou en appelant`CreateActivity`. Cela fournit un Amazon Resource Name (ARN) pour l'état de votre tâche. Utilisez cet ARN pour interroger l'état de la tâche à la recherche d'un travail de votre travail d'activité.

> **Note**
>
> Les versions des activités ne sont pas gérées et les activités doivent être rétrocompatibles. Si vous devez apporter une modification incompatible avec les versions antérieures à une activité, créez-en une nouvelle dans Step Functions en utilisant un nom unique.

Un worker d'activité peut être une application exécutée sur une instance Amazon EC2, une AWS Lambda fonction, un appareil mobile : toute application capable d'établir une connexion HTTP, hébergée n'importe où. Lorsque Step Functions atteint l'état d'une tâche d'activité, le flux de travail attend qu'un agent d'activité interroge pour une tâche. Un travailleur d'activité interroge Step Functions en utilisant `GetActivityTask` et en envoyant l'ARN pour l'activité associée. `GetActivityTask`renvoie une réponse comprenant `input` (une chaîne d'entrée JSON pour la tâche) et un `taskToken`(un identifiant unique pour la tâche). Une fois le travail d'activité terminé, il peut fournir un rapport de réussite ou d'échec en utilisant `SendTaskSuccess` ou `SendTaskFailure`. Ces deux appels utilisent le `taskToken` fourni par `GetActivityTask` pour associer le résultat à cette tâche.

API liées aux tâches d'activité

Step Functions fournit des API permettant de créer et de répertorier des activités, de demander une tâche et de gérer le flux de votre machine d'état en fonction des résultats de votre travailleur.

Les API Step Functions associées aux activités sont les suivantes :

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

> **Note**
>
> La recherche de tâches d'activité avec `GetActivityTask` peut provoquer une certaine latence dans les implémentations. Consultez Évitez la latence lorsque vous interrogez pour des tâches d'activité.

En attente de l'achèvement d'une tâche d'activité

Configurez la durée d'attente d'un état en définissant `TimeoutSeconds` dans la définition de tâche. Pour garder la tâche active et en attente, envoyez régulièrement une pulsation depuis votre travail d'activité en utilisant `SendTaskHeartbeat` dans le délai configuré dans `TimeoutSeconds`. En configurant un délai d'attente long et en envoyant activement un battement de cœur, une activité dans Step Functions peut attendre jusqu'à un an avant la fin de son exécution.

Par exemple, si vous avez besoin d'un flux de travail en attente du résultat d'un long processus, procédez comme suit.

1. Créez une activité dans la console ou en utilisant `CreateActivity`. Notez l'ARN d'activité.
2. Faites référence à l'ARN se trouvant en état de tâche d'activité dans la définition de votre machine d'état et définissez `TimeoutSeconds`.
3. Implémentez un travail d'activité qui recherche un travail à l'aide de `GetActivityTask`, faisant référence à cet ARN d'activité.
4. Utilisez `SendTaskHeartbeat` régulièrement au cours de la durée définie dans `HeartbeatSeconds` dans la définition de la tâche de la machine d'état pour éviter l'expiration de la tâche.
5. Démarrer l'exécution de la machine d'état.

6. Démarrez le processus de votre travail d'activité.

L'exécution s'interrompt au niveau de l'état de tâche d'activité et attend que le travail d'activité recherche une tâche. Dès qu'un `taskToken` est fourni à votre travail d'activité, votre flux de travail attend que `SendTaskSuccess` ou `SendTaskFailure` indique un statut. Si l'exécution ne reçoit pas l'un ou l'autre de ces éléments ou un appel `SendTaskHeartbeat` avant l'heure configurée dans `TimeoutSeconds`, l'exécution échoue et l'historique des exécutions comportera un événement `ExecutionTimedOut`.

Étapes suivantes

Pour un aperçu plus détaillé de la création de machines d'état qui utilisent des programmes de travail d'activité, voir :

- Création d'une machine à états d'activité à l'aide de Step Functions
- Exemple d'activité de travail en Ruby

Exemple d'activité de travail en Ruby

Ce qui suit est un exemple de travail d'activité utilisant le AWS SDK for Ruby pour montrer comment utiliser les bonnes pratiques et implémenter votre propre travail d'activité.

Le code implémente un modèle consommateur-producteur avec un nombre configurable de threads pour les observateurs et les travaux d'activité. Les threads de l'observateur interrogent constamment et longuement la tâche d'activité. Une fois qu'une tâche d'activité est extraite, elle est transmise via une file d'attente à blocage délimité pour que le thread de l'activité la sélectionne.

- Pour plus d'informations à ce sujetAWS SDK for Ruby, consultez la référence de AWS SDK for Ruby l'API.
- Pour télécharger ce code et les ressources associées, consultez le référentiel step-functions-ruby-activity-worker surGitHub.

Le code Ruby suivant est le point d'entrée principal pour cet exemple de travail d'activité.

```
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
```

Le code inclut les valeurs par défaut que vous pouvez modifier pour faire référence à votre activité et l'adapter à votre implémentation spécifique. Le code prend comme entrée la logique réelle de l'implémentation, vous permet de faire référence à votre activité spécifique et à vos informations d'identification, et vous permet de configurer le nombre de threads et le délai de pulsation. Pour plus d'informations et pour télécharger le code, consultez Step Functions Ruby Activity Worker.

Élément	Description
`require_relative`	Chemin d'accès relatif à l'exemple du code du travail d'activité suivant.
`region`	AWS Région de votre activité.
`workers_count`	Nombre de threads de votre travail d'activité. Pour la plupart des implémentations, entre 10 et 20 threads doivent suffire. Plus le temps de traitement de l'activité est long, plus le nombre de threads nécessaires est élevé. À titre d'estimation, multipliez le nombre d'activités de traitement par seconde par la latence du traitement d'activité du 99e percentile, en secondes.

Élément	Description
`pollers_count`	Nombre de threads de vos observateurs. Entre 10 et 20 threads doivent suffire pour la plupart des implémentations.
`heartbeat_delay`	Délai en secondes entre les pulsations.
`input`	Logique d'implémentation de votre activité.

Voici le travail d'activité Ruby, auquel il est fait référence dans votre code avec `../lib/step_functions/activity`.

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

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

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

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

  def self.with_retries(options = {}, &block)
    retries = 0
```

```
    base_delay_seconds = options[:base_delay_seconds] || 2
    max_retries = options[:max_retries] || 3
    begin
      block.call
    rescue => e
      puts e
      if retries < max_retries
        retries += 1
        sleep base_delay_seconds**retries
        retry
      end
      raise RetryError, 'All retries of operation had failed'
    end
  end

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

    def start(&block)
      @sink = SizedQueue.new(@queue_max)
      @activities = Set.new
      start_heartbeat_worker(@activities)
      start_workers(@activities, block, @sink)
      start_pollers(@activities, @sink)
      wait
    end

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

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

    private

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

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

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

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

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

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

    def shutdown_worker(worker)
      worker.kill
    end

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

    def wait_activities_drained
      wait_condition { @sink.empty? }
    end

    def wait_activities_completed
      wait_condition { @activities.empty? }
    end

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

```
    end
  end

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

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

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

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

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

    def stop
      @running = false
    end

    def kill
      return if @thread.nil?
```

```
      @thread.kill
      @thread = nil
    end

    def wait
      @thread.join
    end
  end

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

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

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

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

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

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

    class HeartbeatWorker < Worker
      def initialize(options = {})
        @states = options[:states]
        @activities = options[:activities]
        @heartbeat_delay = options[:heartbeat_delay]
```

```
        @max_retry = options[:max_retry]
        @logger = Logger.new(STDOUT)
      end

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

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

Choice

Un `Choice` state (`"Type": "Choice"`) ajoute une logique conditionnelle à une machine à états.

Outre la plupart des champs d'état courants, les `Choice` états contiennent les champs supplémentaires suivants.

`Choices` (Obligatoire)

Tableau des règles Choice qui détermine quel est l'état suivant de la machine d'état. Vous utilisez un opérateur de comparaison dans une règle de choix pour comparer une variable d'entrée à

une valeur spécifique. Par exemple, à l'aide des règles de choix, vous pouvez comparer si une variable d'entrée est supérieure ou inférieure à 100.

Lorsqu'un `Choice` état est exécuté, il évalue chaque règle de choix comme étant vraie ou fausse. Sur la base du résultat de cette évaluation, Step Functions passe à l'état suivant du flux de travail.

Vous devez définir au moins une règle dans l'`Choice`État.

`Default` (Facultatif, Recommandé)

Nom de l'état auquel passer si aucune des transitions de `Choices` n'a été prise.

> **Important**
>
> Les états `Choice` ne prennent pas en charge le champ `End`. De plus, ils utilisent `Next` uniquement dans le champ `Choices`.

> **Tip**
>
> Pour déployer un exemple de flux de travail qui utilise un `Choice` état pour vousCompte AWS, consultez le Module 5, intitulé Choice State and Map State of The AWS Step Functions Workshop.

Règles Choice

Un `Choice` état doit comporter un `Choices` champ dont la valeur est un tableau non vide. Chaque élément de ce tableau est un objet appelé Choice Rule, qui contient les éléments suivants :

- Comparaison : deux champs qui indiquent la variable d'entrée à comparer, le type de comparaison et la valeur à laquelle comparer la variable. Les règles de choix permettent de comparer deux variables. Dans une règle de choix, la valeur d'une variable peut être comparée à une autre valeur provenant de l'entrée d'état en l'ajoutant `Path` au nom des opérateurs de comparaison pris en charge. Les valeurs des champs `Variable` et Path d'une comparaison doivent être des chemins de référence valides.
- Un **`Next`**champ : la valeur de ce champ doit correspondre à un nom d'état dans la machine à états.

L'exemple suivant vérifie si la valeur numérique est égale à 1.

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

L'exemple suivant vérifie si la chaîne est égale à `MyString`.

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

L'exemple suivant vérifie si la chaîne est supérieure à `MyStringABC`.

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

L'exemple suivant vérifie si la chaîne est nulle.

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

L'exemple suivant montre comment la StringEquals règle n'est évaluée que lorsqu'elle `$.keyThatMightNotExist` existe en raison de la règle de `IsPresent` choix précédente.

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

```
]
```

L'exemple suivant vérifie si un modèle comportant un caractère générique correspond.

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

L'exemple suivant vérifie si l'horodatage est égal à `2001-01-01T12:00:00Z`.

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

L'exemple suivant compare une variable à une autre valeur de l'entrée d'état.

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

Step Functions examine chacune des règles de choix dans l'ordre indiqué dans le `Choices` champ. Ensuite, il passe à l'état spécifié dans le champ `Next` de la première règle Choice dans lequel la variable correspond à la valeur en fonction de l'opérateur de comparaison.

Les opérateurs de comparaison suivants sont pris en charge :

- `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`

Pour chacun de ces opérateurs, la valeur correspondante doit être du type approprié : chaîne, nombre, booléen ou horodatage. Step Functions ne tente pas de faire correspondre un champ numérique à une valeur de chaîne. Cependant, étant donné que les champs d'horodatage sont logiquement des chaînes, il est possible qu'un champ considéré comme un horodatage puisse être rapproché par un comparateur `StringEquals`.

> **Note**
>
> Pour l'interopérabilité, ne supposez pas que les comparaisons numériques fonctionnent avec des valeurs en dehors de l'ampleur ou de la précision représentée par le type de données IEEE 754-2008 `binary64`. Notez plus particulièrement que les entiers en dehors de la plage $[-2^{53}+1, 2^{53}-1]$ ne pourront pas être comparés comme prévu.
> Les horodatages (par exemple, `2016-08-18T17:33:00Z`) doivent respecter RFC3339 profil ISO 8601, avec davantage de restrictions :

- Un T majuscule doit séparer les parties date et heure.
- Un Z majuscule doit indiquer qu'un décalage de fuseau horaire numérique n'est pas présent.

Pour comprendre le comportement des comparaisons de chaînes, consultez la documentation JavacompareTo.
Les valeurs des opérateurs And et Or doivent être des tableaux non vides de règles Choice qui ne doivent pas elles-mêmes contenir des champs Next. De la même manière, la valeur d'un opérateur Not doit être une seule règle Choice qui ne doit pas contenir de champs Next.
Vous pouvez créer des règles Choice imbriquées complexes à l'aide de And, Not et Or. Toutefois, le champ Next peut s'afficher uniquement dans une règle Choice de niveau supérieur.
La comparaison de chaînes avec des modèles comportant un ou plusieurs caractères génériques (« * ») peut être effectuée à l'aide de l'opérateur de StringMatches comparaison. Le caractère générique est échappé à l'aide du caractère standard\\ (Ex: “*”). Aucun caractère autre que « * » n'a de signification particulière lors de la mise en correspondance.

Exemple d'état Choice

Voici un exemple d'état Choice et des autres états auquel il est possible de passer.

> **Note**
>
> Vous devez spécifier le champ $.type. Si l'entrée d'état ne contient pas le champ $.type, l'exécution échoue et une erreur s'affiche dans l'historique d'exécution. Vous pouvez uniquement spécifier une chaîne dans le StringEquals champ qui correspond à une valeur littérale. Par exemple, "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!"
}
```

Dans cet exemple, la machine d'état commence par la valeur d'entrée suivante :

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

Step Functions passe à l'`ValueInTwenties`état en fonction du `value` champ.

S'il n'y a aucune correspondance pour l'état `Choice` et ses `Choices`, l'état fourni dans le champ `Default` est exécuté à la place. Si l'état `Default` n'est pas spécifié, l'exécution échoue avec une erreur.

Attente

Un état `Wait` (`"Type": "Wait"`) empêche la machine d'état de continuer à fonctionner pendant une durée spécifiée. Vous pouvez choisir une durée relative, spécifiée en secondes à partir du moment où l'état commence, ou une heure de fin absolue, spécifiée comme un horodatage.

Outre les champs d'état courants, les états `Wait` ont l'un des champs suivants.

`Seconds`

Une durée d'attente (en secondes) avant le début de l'état spécifié dans le champ `Next`. Vous devez spécifier l'heure sous forme de valeur entière positive comprise entre 0 et 99999999.

`Timestamp`

Une durée d'attente absolue avant le début de l'état spécifié dans le champ `Next`.

Les horodatages doivent respecter le profil RFC3339 de la norme ISO 8601, avec les restrictions supplémentaires selon lesquelles un `T` majuscule doit séparer la date et l'heure, et un `Z` majuscule doit indiquer qu'un décalage de fuseau horaire numérique n'est pas présent, par exemple `2024-08-18T17:33:00Z`.

> **Note**
>
> Actuellement, si vous spécifiez le temps d'attente sous forme d'horodatage, Step Functions prend en compte la valeur temporelle maximale en secondes et tronque les millisecondes.

`SecondsPath`

Une durée d'attente (en secondes) avant le début de l'état spécifié dans le champ `Next`, spécifiée à l'aide d'un chemin à partir des données d'entrée de l'état.

Vous devez spécifier une valeur entière pour ce champ.

`TimestampPath`

Une durée d'attente absolue avant le début de l'état spécifié dans le champ `Next`, spécifiée à l'aide d'un chemin à partir des données d'entrée de l'état.

> **Note**
>
> Vous devez spécifier exactement `Seconds`, `Timestamp`, `SecondsPath` ou `TimestampPath`. En outre, le temps d'attente maximal que vous pouvez spécifier pour les flux de travail standard et les flux de travail express est d'un an et cinq minutes respectivement.

Exemples d'état Wait

L'état `Wait` suivant introduit un retard de dix secondes dans une machine d'état.

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

Dans l'exemple suivant, l'`Wait`État attend une heure absolue : le 14 mars 2024, à 1 h 59 UTC.

```
"wait_until" : {
```

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

La durée d'attente n'a pas besoin d'être codée en dur. Par exemple, avec les données d'entrée suivantes :

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

Vous pouvez sélectionner la valeur « expirydate » de l'entrée à l'aide d'un chemin de référence pour la sélectionner dans les données d'entrée.

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

Succeed

Un état `Succeed` (`"Type": "Succeed"`) arrête une exécution avec succès. L'état `Succeed` est une cible utile pour les branches d'état `Choice` qui sont chargées d'arrêter l'exécution.

Comme les états `Succeed` sont des états terminaux, ils n'ont pas de champ `Next` et n'ont pas besoin d'un champ `End`, comme illustré dans l'exemple suivant.

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

Fail

UN`Fail`État (`"Type": "Fail"`) arrête l'exécution de la machine à états et la marque comme une défaillance, sauf si elle est interceptée par un`Catch`bloquer.

L'état `Fail` autorise uniquement l'utilisation de champs `Type` et `Comment` à partir de l'ensemble de champs d'état courants. En outre, l'état `Fail` autorise les champs suivants :

`Cause` (facultatif)

Chaîne personnalisée qui décrit la cause de l'erreur. Vous pouvez spécifier ce champ à des fins opérationnelles ou diagnostiques.

`CausePath` (facultatif)

Si vous souhaitez fournir une description détaillée de la cause de l'erreur de manière dynamique à partir de l'entrée d'état à l'aide d'unchemin de référence, utiliser`CausePath`. Une fois résolu, le chemin de référence doit sélectionner un champ contenant une valeur de chaîne.

Vous pouvez également spécifier`CausePath`à l'aide d'unfonction intrinsèquequi renvoie une chaîne. Ces éléments intrinsèques sont les suivants :États. Format,States.JsonToString,States.ArrayGetItem,States.Base64Encode,States.Base64Decode,States.Ha etStates.UUID.

> **Important**
>
> - Vous pouvez utiliser les`Cause`ou`CausePath`, mais pas les deux dans votre définition de l'état d'échec.
> - Comme bonne pratique en matière de sécurité, nous vous recommandons de supprimer les informations sensibles ou les informations relatives aux systèmes internes dans la description de la cause.

`Error` (facultatif)

Nom d'erreur que vous pouvez fournir pour effectuer la gestion des erreurs à l'aide deRéessayerouAttraperchamps. Vous pouvez également fournir un nom d'erreur à des fins opérationnelles ou de diagnostic.

`ErrorPath` (facultatif)

Si vous souhaitez attribuer un nom à l'erreur de manière dynamique à partir de l'entrée d'état à l'aide d'unchemin de référence, utiliser`ErrorPath`. Une fois résolu, le chemin de référence doit sélectionner un champ contenant une valeur de chaîne.

Vous pouvez également spécifier`ErrorPath`à l'aide d'unfonction intrinsèquequi renvoie une chaîne. Ces éléments intrinsèques sont les suivants :États. Format,States.JsonToString,States.ArrayGetItem,States.Base64Encode,States.Base64Decode,States.Ha etStates.UUID.

> **Important**
>
> - Vous pouvez utiliser les`Error`ou`ErrorPath`, mais pas les deux dans votre définition de l'état d'échec.
> - Comme bonne pratique en matière de sécurité, nous vous recommandons de supprimer les informations sensibles ou les informations relatives aux systèmes internes du nom de l'erreur.

Comme les états `Fail` quittent toujours la machine d'état, ils n'ont pas de champ `Next` ni ne nécessitent un champ `End`.

Exemples de définition de l'état d'échec

L'exemple de définition de l'état d'échec suivant spécifie une valeur statique`Error`et`Cause`valeurs relatives aux champs.

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

L'exemple de définition de l'état d'échec suivant utilise des chemins de référence de manière dynamique pour résoudre le`Error`et`Cause`valeurs relatives aux champs.

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

L'exemple de définition de l'état d'échec suivant utiliseÉtats. Formatfonction intrinsèque pour spécifier le`Error`et`Cause`valeurs relatives aux champs.

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

```
}
```

Parallèle

Le `Parallel` state (`"Type": "Parallel"`) peut être utilisé pour ajouter des branches d'exécution distinctes dans votre machine à états.

Outre les champs d'état courants, les états `Parallel` incluent les champs suivants :

`Branches` (Obligatoire)

Tableau d'objets qui spécifient les machines d'état à exécuter en parallèle. Chaque objet de ce type de la machine d'état doit avoir des champs nommés `States` et `StartAt` dont les significations sont exactement comme celles du niveau supérieur d'une machine d'état.

`ResultPath` (facultatif)

Indique où (dans l'entrée) placer la sortie des branches. Les entrées sont ensuite filtrées telles que spécifiées par le champ `OutputPath` (s'il est présent) avant d'être utilisées comme sortie de l'état. Pour plus d'informations, consultez Traitement des entrées et des sorties.

`ResultSelector` (facultatif)

Transmettez une collection de paires clé-valeur, où les valeurs sont statiques ou sélectionnées à partir du résultat. Pour de plus amples informations, veuillez consulter ResultSelector.

`Retry` (facultatif)

Tableau d'objets, nommés Réessayeurs, qui définissent une stratégie de nouvelle tentative si l'état rencontre des erreurs d'exécution. Pour de plus amples informations, veuillez consulter Exemples de machines à états utilisant Retry et Catch.

`Catch` (facultatif)

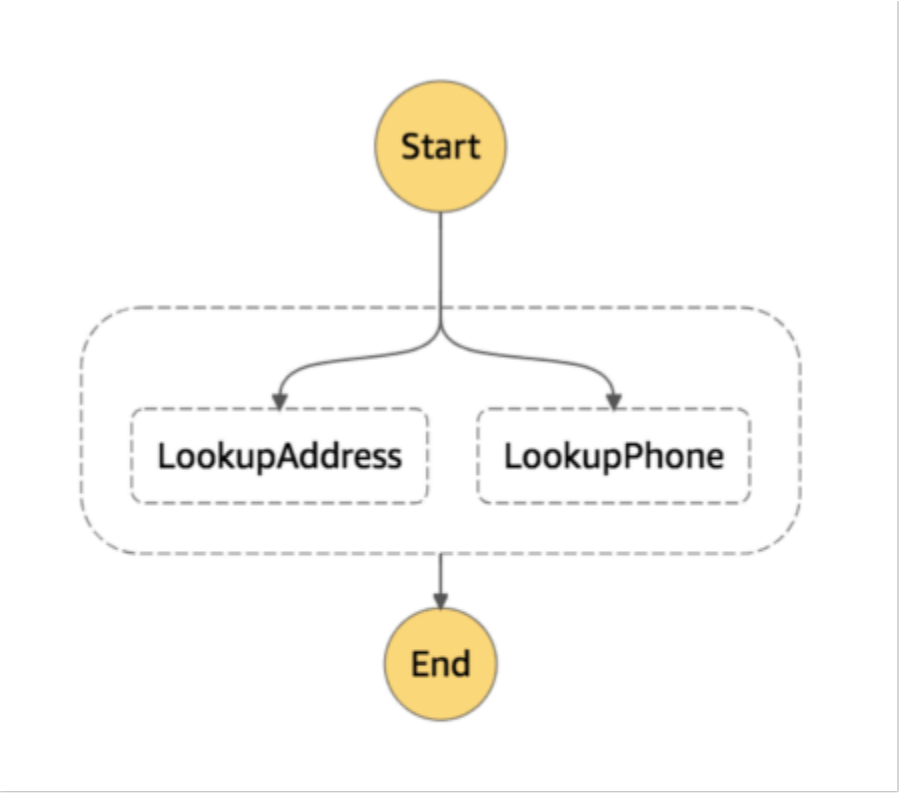
Tableau d'objets, nommés Receveurs, qui définissent un état de secours exécuté lorsque l'état rencontre des erreurs d'exécution et que sa stratégie de nouvelle tentative a été épuisée ou n'est pas définie. Pour plus d'informations, consultez États de secours.

Un `Parallel` état AWS Step Functions entraîne l'exécution de chaque branche, en commençant par l'état nommé dans le `StartAt` champ de cette branche, le plus simultanément possible, et l'attente que toutes les branches se terminent (atteignent un état terminal) avant de traiter le `Next` champ de `Parallel` l'état.

Exemple d'état Parallel

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

Dans cet exemple, les branches `LookupAddress` et `LookupPhone` sont exécutées en parallèle. Voici à quoi ressemble le flux de travail visuel dans la console Step Functions.



Chaque branche doit être autonome. Un état dans une branche d'un état `Parallel` ne doit pas avoir de champ `Next` qui cible un champ à l'extérieur de cette branche, ni aucun autre état en dehors de la transition de branche vers cette branche.

Traitement des entrées et des sorties d’état Parallel

Un état `Parallel` fournit à chaque branche une copie de ses propres données d'entrée (soumises à modification par le champ `InputPath`). Il génère une sortie qui est un tableau avec un élément pour chaque branche contenant la sortie à partir de cette branche. Tous les éléments ne doivent pas nécessairement être du même type. Le tableau de sortie peut être inséré dans les données d'entrée (et l'ensemble envoyé en tant que sortie de l'état `Parallel`) en utilisant un champ `ResultPath` de façon classique (voir Traitement des entrées et des sorties).

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

Si l'état `FunWithMath` a reçu le tableau `[3, 2]` en tant qu'entrée, les deux états `Add` et `Subtract` reçoivent ce même tableau en entrée. Le résultat des `Subtract` tâches `Add` et serait la somme et la différence entre les éléments 3 et 2 du tableau, c'est-à-dire `5` et`1`, tandis que le résultat de l'`Parallel`état serait un tableau.

```
[ 5, 1 ]
```

> **Tip**
>
> Si l'état Parallel ou Map que vous utilisez dans vos machines d'état renvoie un tableau de tableaux, vous pouvez les transformer en tableau plat avec le ResultSelector champ. Pour de plus amples informations, veuillez consulter Aplatir un tableau de tableaux.

Gestion des erreurs

Si une branche échoue, en raison d'une erreur non gérée ou en passant à l'état `Fail`, tout l'état `Parallel` est considéré comme ayant échoué et toutes ses branches sont arrêtées. Si l'erreur n'est pas gérée par l'`Parallel`état lui-même, Step Functions arrête l'exécution avec une erreur.

> **Note**
>
> Lorsqu'un état parallèle échoue, les fonctions Lambda invoquées continuent de s'exécuter et les opérateurs d'activité qui traitent un jeton de tâche ne sont pas arrêtés.
>
> - Pour arrêter les activités de longue durée, utilisez les battements de cœur pour détecter si la succursale a été arrêtée par Step Functions et arrêtez les travailleurs en train de traiter des tâches. Le fait d'appeler `SendTaskHeartbeat`, `SendTaskSuccess` ou `SendTaskFailure` génère une erreur si l'état a échoué. Voir Erreurs de pulsation.
> - L'exécution des fonctions Lambda ne peut pas être arrêtée. Si vous avez implémenté une solution de secours, utilisez un `Wait` état pour que le nettoyage soit effectué une fois la fonction Lambda terminée.

Map

Utilisez l'`Map`état pour exécuter un ensemble d'étapes de flux de travail pour chaque élément d'un ensemble de données. Les itérations de `Map` l'état s'exécutent en parallèle, ce qui permet de traiter rapidement un ensemble de données. `Map`les états peuvent utiliser différents types d'entrées, notamment un tableau JSON, une liste d'objets Amazon S3 ou un fichier CSV.

Step Functions propose deux types de modes de traitement pour utiliser l'`Map`état dans vos flux de travail : le mode en ligne et le mode distribué.

Pour plus d'informations sur ces modes et sur la façon d'utiliser l'`Map`état dans l'un ou l'autre mode, consultez les rubriques suivantes :

> **Tip**
>
> Pour déployer un exemple de flux de travail qui utilise un `Map` état pour votreCompte AWS, consultez le module 5 - Choix de l'état et état de la carte de l'AWS Step Functionsatelier.

Modes de traitement de l'état des cartes

Step Functions propose les modes de traitement suivants pour l'`Map`état en fonction de la manière dont vous souhaitez traiter les éléments d'un ensemble de données.

- Inline — Mode de simultanéité limité. Dans ce mode, chaque itération de l'`Map`état s'exécute dans le contexte du flux de travail qui contient l'`Map`état. Step Functions ajoute l'historique d'exécution de ces itérations à l'historique d'exécution du flux de travail parent. Par défaut, `Map` les états s'exécutent en mode Inline.

  Dans ce mode, l'`Map`État accepte uniquement un tableau JSON en entrée. De plus, ce mode prend en charge jusqu'à 40 itérations simultanées.

  Pour plus d'informations, veuillez consulter Utilisation de l'état de la carte en mode Inline.
- Distribué — Mode de simultanéité élevé. Dans ce mode, l'`Map`état exécute chaque itération en tant qu'exécution d'un flux de travail enfant, ce qui permet une simultanéité élevée de jusqu'à 10 000 exécutions parallèles de flux de travail enfant. Chaque exécution de flux de travail enfant possède son propre historique d'exécution distinct de celui du flux de travail parent.

  Dans ce mode, l'`Map`État peut accepter un tableau JSON ou une source de données Amazon S3, telle qu'un fichier CSV, comme entrée.

  Pour plus d'informations, veuillez consulter Utilisation de l'état de la carte en mode distribué.

Le mode à utiliser dépend de la manière dont vous souhaitez traiter les éléments d'un ensemble de données. Utilisez l'`Map`état en mode Inline si l'historique d'exécution de votre flux de travail ne doit pas dépasser 25 000 entrées ou si vous n'avez pas besoin de plus de 40 itérations simultanées.

Utilisez l'`Map`état en mode distribué lorsque vous devez orchestrer des charges de travail parallèles à grande échelle répondant à une combinaison des conditions suivantes :

- La taille de votre jeu de données dépasse 256 Ko.
- L'historique des événements d'exécution du flux de travail dépasse 25 000 entrées.
- Vous avez besoin d'une simultanéité de plus de 40 itérations parallèles.

Rubriques

Différences entre le mode en ligne et le mode distribué

Le tableau suivant met en évidence les différences entre les modes en ligne et distribué.

Mode en ligne	Mode distribué
Supported data sources	
Accepte un tableau JSON transmis depuis une étape précédente du flux de travail en entrée.	Accepte les sources de données suivantes en entrée : • Tableau JSON transmis depuis une étape précédente du flux de travail • Fichier JSON dans un compartiment Amazon S3 contenant un tableau • Fichier CSV dans un compartiment Amazon S3 • Liste d'objets Amazon S3 • Inventaire Amazon S3
Map iterations	

Mode en ligne	Mode distribué
Dans ce mode, chaque itération de l'`Map`état s'exécute dans le contexte du flux de travail qui contient l'`Map`état. Step Functions ajoute l'histori que d'exécution de ces itérations à l'historique d'exécution du flux de travail parent.	Dans ce mode, l'`Map`état exécute chaque itération en tant qu'exécution d'un flux de travail enfant, ce qui permet une simultanéité élevée de jusqu'à 10 000 exécutions parallèles de flux de travail enfant. Chaque exécution de flux de travail enfant possède son propre historique d'exécution distinct de celui du flux de travail parent.
Maximum concurrency for parallel iterations	
Permet d'exécuter jusqu'à 40 itérations aussi simultanément que possible.	Vous permet d'exécuter jusqu'à 10 000 exécutions parallèles de flux de travail enfant pour traiter des millions d'éléments de données à la fois.
Input payload and event history sizes	
Impose une limite de 256 Ko à la taille de la charge utile d'entrée et à 25 000 entrées dans l'historique des événements d'exécution.	Permet de contourner la limite de taille de la charge utile, car l'`Map`État peut lire les entrées directement depuis les sources de données Amazon S3. Dans ce mode, vous pouvez également contourner les limites de l'historique d'exécuti on, car les exécutions du flux de travail enfant lancées par l'`Map`État conservent leur propre historique d'exécution distinct de l'historique d'exécution du flux de travail parent.
Monitoring and observability	

Mode en ligne	Mode distribué
Vous pouvez consulter l'historique d'exécution du flux de travail depuis la console ou en appelant l'action d'[GetExecutionHistory](#) API. Vous pouvez également consulter l'historique des exécutions via CloudWatch X-Ray.	Lorsque vous exécutez un `Map` état en mode distribué, Step Functions crée une ressource Map Run. Une exécution de carte fait référence à un ensemble d'exécutions de flux de travail secondaires lancées par un état de carte distribuée. Vous pouvez consulter un Map Run dans la console Step Functions. Vous pouvez également invoquer l'action d'[DescribeMapRun](#) API. Un Map Run envoie également des métriques à. CloudWatch Pour plus d'informations, veuillez consulter [Examen de l'exécution cartographique d'une exécution de l'état d'une carte distribuée](#).

Utilisation de l'état de la carte en mode Inline

Par défaut, `Map` les états s'exécutent en mode Inline. En mode Inline, l'état de la carte accepte uniquement un tableau JSON en entrée. Il reçoit ce tableau lors d'une étape précédente du flux de travail. Dans ce mode, chaque itération de l'`Map`état s'exécute dans le contexte du flux de travail qui contient l'`Map`état. Step Functions ajoute l'historique d'exécution de ces itérations à l'historique d'exécution du flux de travail parent.

Dans ce mode, l'`Map`état prend en charge jusqu'à 40 itérations simultanées.

Un `Map` état défini sur Inline est appelé état Inline Map. Utilisez l'`Map`état en mode Inline si l'historique d'exécution de votre flux de travail ne doit pas dépasser 25 000 entrées ou si vous n'avez pas besoin de plus de 40 itérations simultanées.

Pour une introduction à l'utilisation de l'état Inline Map, consultez le didacticielRépéter une action à l'aide de l'état de la carte intégrée.

Table des matières

- Champs déconseillés
- Exemple d'état d'une carte intégrée
- Exemple d'état de la carte intégrée avec ItemSelector
- Traitement d'entrée et de sortie d'Mapétat en ligne

Concepts clés de cette rubrique

Mode en ligne

Mode de simultanéité limité de l'État. `Map` Dans ce mode, chaque itération de l'`Map`état s'exécute dans le contexte du flux de travail qui contient l'`Map`état. Step Functions ajoute l'historique d'exécution de ces itérations à l'historique d'exécution du flux de travail parent. `Map`les états s'exécutent en mode Inline par défaut.

Ce mode n'accepte qu'un tableau JSON en entrée et prend en charge jusqu'à 40 itérations simultanées.

État de la carte intégrée

`Map`État défini sur le mode Inline.

Flux de travail cartographique

Ensemble d'étapes que l'`Map`état exécute pour chaque itération.

Itération de l'état de la carte

Une répétition du flux de travail défini dans l'`Map`état.

Champs d'état de la carte intégrée

Pour utiliser l'état Inline Map dans vos flux de travail, spécifiez un ou plusieurs de ces champs. Vous spécifiez ces champs en plus des champs d'état courants.

`Type` (Obligatoire)

Définit le type d'état, tel que`Map`.

`ItemProcessor` (Obligatoire)

Contient les objets JSON suivants qui spécifient le mode et la définition de traitement de `Map` l'état.

La définition contient l'ensemble des étapes à répéter pour traiter chaque élément du tableau.

- `ProcessorConfig`— Objet JSON facultatif qui spécifie le mode de traitement de l'`Map`état. Cet objet contient le `Mode` sous-champ. La valeur par défaut de ce champ est`INLINE`, qui utilise l'`Map`état en mode Inline.

  Dans ce mode, l'échec d'une itération entraîne l'échec de l'`Map`état. Toutes les itérations s'arrêtent lorsque l'`Map`état échoue.
- `StartAt`— Spécifie une chaîne qui indique le premier état d'un flux de travail. Cette chaîne distingue les majuscules et minuscules et doit correspondre au nom de l'un des objets d'état. Cet état s'exécute d'abord pour chaque élément de l'ensemble de données. Toute entrée d'exécution que vous fournissez à l'`Map`état passe d'abord à l'`StartAt`état.
- `States`— Objet JSON contenant un ensemble d'états séparés par des virgules. Dans cet objet, vous définissez leMap workflow.

> **Note**
>
> - Les États du `ItemProcessor` domaine ne peuvent que passer les uns aux autres. Aucun état extérieur au `ItemProcessor` champ ne peut passer à un état intérieur.
> - Le `ItemProcessor` champ remplace le champ désormais obsolète. `Iterator` Bien que vous puissiez continuer à inclure `Map` les états qui utilisent le `Iterator` champ, nous vous recommandons vivement de le remplacer par`ItemProcessor`.
>
>   Step Functions Local ne prend actuellement pas en charge `ItemProcessor` ce domaine. Nous vous recommandons d'utiliser le `Iterator` champ avec Step Functions Local.

`ItemsPath` (facultatif)

Spécifie un chemin de référence à l'aide de la JsonPathsyntaxe. Ce chemin sélectionne le nœud JSON qui contient le tableau d'éléments à l'intérieur de l'entrée d'état. Pour plus d'informations, veuillez consulter ItemsPath.

`ItemSelector` (facultatif)

Remplace les valeurs des éléments du tableau d'entrée avant qu'elles ne soient transmises à chaque itération `Map` d'état.

Dans ce champ, vous spécifiez un JSON valide contenant une collection de paires clé-valeur. Ces paires peuvent contenir l'un des éléments suivants :

- Valeurs statiques que vous définissez dans la définition de votre machine à états.
- Valeurs sélectionnées à partir de l'entrée d'état à l'aide d'un chemin.
- Valeurs accessibles depuis l'objet de contexte.

Pour plus d'informations, veuillez consulter ItemSelector.

Le `ItemSelector` champ remplace le champ désormais obsolète. `Parameters` Bien que vous puissiez continuer à inclure `Map` les états qui utilisent le `Parameters` champ, nous vous recommandons vivement de le remplacer par`ItemSelector`.

`MaxConcurrency` (facultatif)

Spécifie une valeur entière qui fournit la limite supérieure du nombre d'itérations d'`Map`états pouvant être exécutées en parallèle. Par exemple, une `MaxConcurrency` valeur de 10 limite l'`Map`état à 10 itérations simultanées exécutées simultanément.

> **Note**
>
> Les itérations simultanées peuvent être limitées. Dans ce cas, certaines itérations ne commenceront pas tant que les itérations précédentes ne seront pas terminées. La probabilité que cela se produise augmente lorsque votre tableau d'entrée comporte plus de 40 éléments.
> Pour obtenir une plus grande simultanéité, considérezUtilisation de l'état de la carte en mode distribué.

La valeur par défaut est`0`, ce qui n'impose aucune limite à la simultanéité. Step Functions invoque des itérations aussi simultanément que possible.

Une `MaxConcurrency` valeur de `1` invoque `ItemProcessor` une fois pour chaque élément du tableau. Les éléments du tableau sont traités dans l'ordre de leur apparition dans l'entrée. Step Functions ne démarre pas une nouvelle itération tant qu'elle n'a pas terminé l'itération précédente.

`MaxConcurrencyPath` (facultatif)

Si vous souhaitez fournir une valeur de simultanéité maximale de manière dynamique à partir de l'entrée d'état à l'aide d'un chemin de référence, utilisez`MaxConcurrencyPath`. Une fois résolu, le chemin de référence doit sélectionner un champ dont la valeur est un entier non négatif.

> **Note**
>
> Un `Map` état ne peut pas inclure à la fois `MaxConcurrency` et`MaxConcurrencyPath`.

`ResultPath` (facultatif)

Spécifie l'endroit de l'entrée où stocker la sortie des itérations de l'`Map`état. L'état de la carte filtre ensuite l'entrée comme spécifié par le `OutputPath`champ, le cas échéant. Ensuite, il utilise l'entrée filtrée comme sortie de l'état. Pour plus d'informations, consultez Traitement des entrées et des sorties.

`ResultSelector` (facultatif)

Transmettez une collection de paires clé-valeur, dont les valeurs sont soit statiques, soit sélectionnées à partir du résultat. Pour plus d'informations, veuillez consulter ResultSelector.

> **Tip**
>
> Si l'état Parallel ou Map que vous utilisez dans vos machines d'état renvoie un tableau de tableaux, vous pouvez les transformer en tableau plat avec le ResultSelector champ. Pour plus d'informations, veuillez consulter Aplatir un tableau de tableaux.

`Retry` (facultatif)

Tableau d'objets, appelés Retriers, qui définit une politique de nouvelle tentative. Les États utilisent une politique de nouvelle tentative lorsqu'ils rencontrent des erreurs d'exécution. Pour plus d'informations, veuillez consulter Exemples de machines à états utilisant Retry et Catch.

> **Note**
>
> Si vous définissez des récupérateurs pour l'état de la carte intégrée, la politique de nouvelles tentatives s'applique à toutes les itérations d'`Map`état, et non aux seules itérations ayant échoué. Par exemple, votre `Map` état contient deux itérations réussies et une itération échouée. Si vous avez défini le `Retry` champ correspondant à l'`Map`état, la politique de nouvelle tentative s'applique aux trois itérations d'`Map`état et non uniquement à l'itération ayant échoué.

`Catch` (facultatif)

Tableau d'objets, nommés Receveurs, qui définissent un état de secours. Les États exécutent un catcher s'ils rencontrent des erreurs d'exécution et s'ils n'ont pas de politique de nouvelle tentative ou si leur politique de relance est épuisée. Pour plus d'informations, consultez États de secours.

Champs déconseillés

> **Note**
>
> Bien que vous puissiez continuer à inclure `Map` les états qui utilisent les champs suivants, nous vous recommandons vivement de les `Iterator` remplacer par `ItemProcessor` et `Parameters` par`ItemSelector`.

`Iterator`

Spécifie un objet JSON qui définit un ensemble d'étapes qui traitent chaque élément du tableau.

`Parameters`

Spécifie une collection de paires clé-valeur, dont les valeurs peuvent contenir l'un des éléments suivants :

- Valeurs statiques que vous définissez dans la définition de votre machine à états.
- Valeurs sélectionnées à partir de l'entrée à l'aide d'un chemin.

Exemple d'état d'une carte intégrée

Tenez compte des données d'entrée suivantes pour un `Map` état exécuté en mode Inline.

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

```
      { "prod": "R40", "dest-code": 9860, "quantity": 887 },
      { "prod": "R40", "dest-code": 9511, "quantity": 1220 }
    ]
  }
}
```

Compte tenu de l'entrée précédente, l'`Map`état de l'exemple suivant invoque une AWS Lambda fonction nommée `ship-val` une fois pour chaque élément du tableau dans le `shipped` champ.

```
"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"
}
```

Chaque itération de l'`Map`état envoie un élément du tableau, sélectionné avec le `ItemsPath`champ, en entrée de la fonction `ship-val` Lambda. Les valeurs suivantes sont un exemple d'entrée que l'`Map`état envoie à un appel de la fonction Lambda :

```
{
  "prod": "R31",
  "dest-code": 9511,
```

```
  "quantity": 1344
}
```

Une fois terminé, la sortie de l'état `Map` est un tableau JSON, où chaque élément est la sortie d'une itération. Dans ce cas, ce tableau contient la sortie de la fonction `ship-val` Lambda.

Exemple d'état de la carte intégrée avec **`ItemSelector`**

Supposons que la fonction `ship-val` Lambda de l'exemple précédent ait également besoin d'informations sur le transporteur de l'expédition. Ces informations s'ajoutent aux éléments du tableau pour chaque itération. Vous pouvez inclure des informations provenant de l'entrée, ainsi que des informations spécifiques à l'itération actuelle de l'`Map`état. Notez le `ItemSelector` champ dans l'exemple suivant :

```
"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
}
```

Le `ItemSelector` bloc remplace l'entrée des itérations par un nœud JSON. Ce nœud contient à la fois les données de l'article actuel provenant de l'objet contextuel et les informations de messagerie provenant du `delivery-partner` champ de saisie de `Map` l'état. Voici un exemple de saisie pour une seule itération. L'`Map`état transmet cette entrée à un appel de la fonction `ship-val` Lambda.

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

Dans l'exemple d'état Inline Map précédent, le `ResultPath` champ produit une sortie au même format que l'entrée. Cependant, il remplace le `detail.shipped` champ par un tableau dans lequel chaque élément est le résultat de l'appel Lambda `ship-val` de chaque itération.

Pour plus d'informations sur l'utilisation de l'état de la carte intégrée et de ses champs, consultez ce qui suit.

- Répéter une action à l'aide de l'état de la carte intégrée
- Traitement des entrées et des sorties dans Step Functions
- ItemsPath
- Données d'objet de contexte pour les états Map

Traitement d'entrée et de sortie d'**Map**état en ligne

Pour un `Map` état donné, `InputPath`sélectionne un sous-ensemble de l'entrée de l'état.

L'entrée d'un `Map` état doit inclure un tableau JSON. L'`Map`état exécute la `ItemProcessor` section une fois pour chaque élément du tableau. Si vous spécifiez le `ItemsPath`champ, l'`Map`état sélectionne l'endroit de l'entrée où se trouve le tableau à itérer. Si elle n'est pas spécifiée, la valeur de `ItemsPath` est `$`, et la section `ItemProcessor` s'attend à ce que le tableau soit la seule entrée. Si vous spécifiez le `ItemsPath` champ, sa valeur doit être un chemin de référence. L'`Map`État applique ce chemin à l'entrée effective après avoir appliqué le`InputPath`. Le `ItemsPath` doit identifier un champ dont la valeur est un tableau JSON.

L'entrée de chaque itération, par défaut, est un élément unique du champ du tableau identifié par la `ItemsPath` valeur. Vous pouvez remplacer cette valeur par le `ItemSelector` champ.

Une fois terminé, la sortie de l'état `Map` est un tableau JSON, où chaque élément est la sortie d'une itération.

Pour plus d'informations sur les entrées et sorties d'état d'Inline Map, consultez les rubriques suivantes :

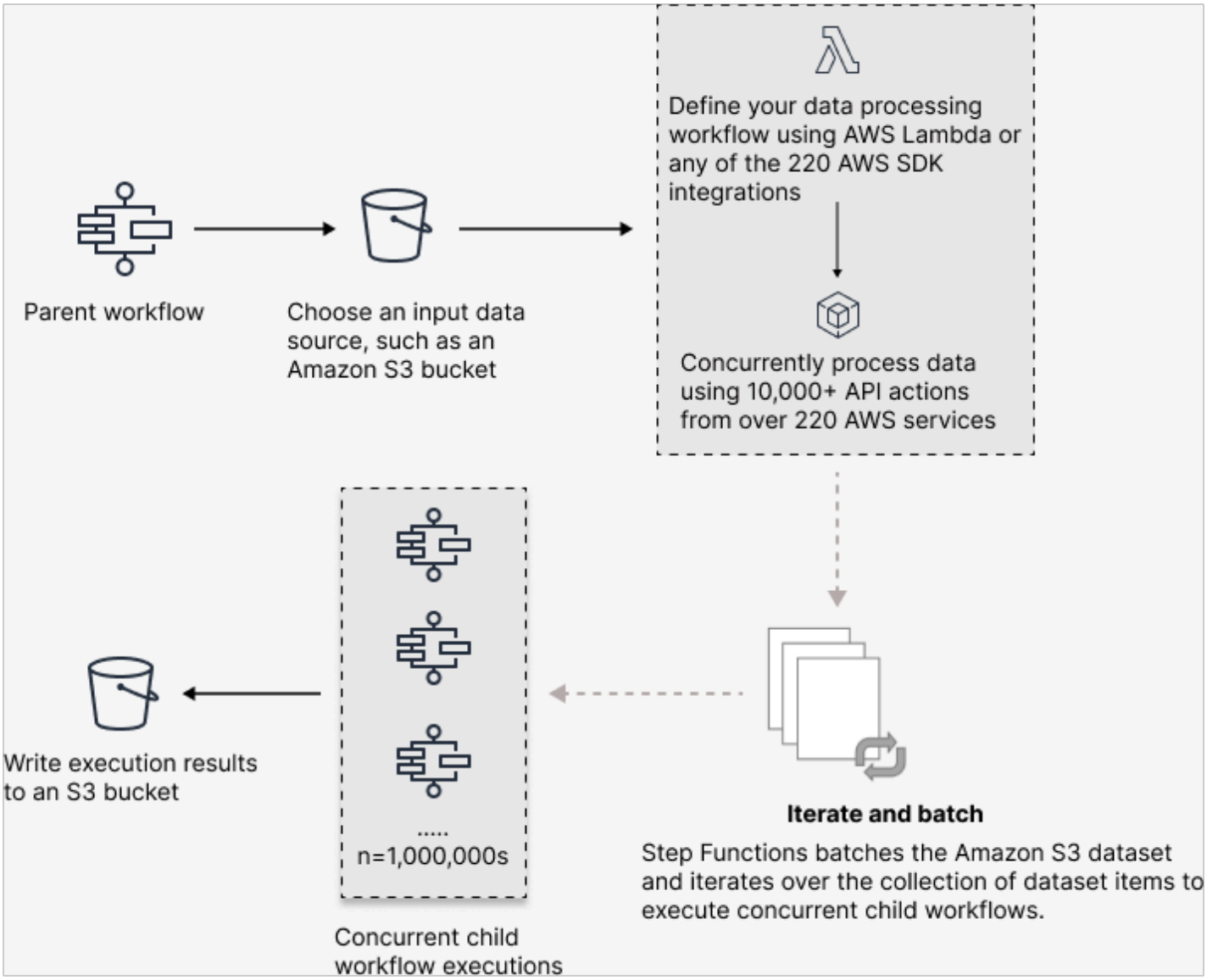
- Répéter une action à l'aide de l'état de la carte intégrée
- Exemple d'état de la carte intégrée avec `ItemSelector`
- Traitement des entrées et des sorties dans Step Functions
- Données d'objet de contexte pour les états Map
- Traitement dynamique des données avec un état cartographique

Utilisation de l'état de la carte en mode distribué pour orchestrer des charges de travail parallèles à grande échelle

Avec Step Functions, vous pouvez orchestrer des charges de travail parallèles à grande échelle pour effectuer des tâches telles que le traitement à la demande de données semi-structurées. Ces charges de travail parallèles vous permettent de traiter simultanément des sources de données à grande échelle stockées dans Amazon S3. Par exemple, vous pouvez traiter un seul fichier JSON ou CSV contenant de grandes quantités de données. Vous pouvez également traiter un grand nombre d'objets Amazon S3.

Pour configurer une charge de travail parallèle à grande échelle dans vos flux de travail, incluez un `Map` état en mode distribué. L'état de la carte traite les éléments d'un jeu de données simultanément. Un `Map` état défini sur Distribué est appelé état de carte distribuée. En mode distribué, l'`Map`état autorise un traitement simultané élevé. En mode distribué, l'`Map`État traite les éléments de l'ensemble de données par itérations appelées exécutions de flux de travail secondaires. Vous pouvez spécifier le nombre d'exécutions de flux de travail enfants qui peuvent être exécutées en parallèle. Chaque exécution de flux de travail enfant possède son propre historique d'exécution distinct de celui du flux de travail parent. Si vous ne le spécifiez pas, Step Functions exécute 10 000 workflows enfants parallèles en parallèle.

L'illustration suivante explique comment configurer des charges de travail parallèles à grande échelle dans vos flux de travail.



Dans cette rubrique

Termes clés

Mode distribué

Mode de traitement de l'état de la carte. Dans ce mode, chaque itération de l'`Map`état s'exécute comme une exécution de flux de travail secondaire qui permet une simultanéité élevée. Chaque exécution de flux de travail enfant possède son propre historique d'exécution, distinct de l'historique d'exécution du flux de travail parent. Ce mode prend en charge la lecture des entrées provenant de sources de données Amazon S3 à grande échelle.

État de la carte distribuée

État de la carte défini sur Mode de traitement distribué.

Flux de travail cartographique

Ensemble d'étapes exécutées par un `Map` État.

Flux de travail parent

Un flux de travail qui contient un ou plusieurs états de cartes distribuées.

Exécution d'un flux de travail

Une itération de l'état de la carte distribuée. L'exécution d'un flux de travail enfant possède son propre historique d'exécution, distinct de l'historique d'exécution du flux de travail parent.

Map Run

Lorsque vous exécutez un `Map` état en mode distribué, Step Functions crée une ressource Map Run. Une exécution de carte fait référence à un ensemble d'exécutions de flux de travail enfants lancées par un état de carte distribuée, ainsi qu'aux paramètres d'exécution qui contrôlent ces exécutions. Step Functions attribue un Amazon Resource Name (ARN) à votre Map Run. Vous pouvez examiner un Map Run dans la console Step Functions. Vous pouvez également invoquer l'action d'`DescribeMapRun`API. Un Map Run envoie également des métriques à. CloudWatch

Pour plus d'informations, consultez Examen de Map Run.

Exemple de définition de l'état d'une carte distribuée

Utilisez l'`Map`état en mode distribué lorsque vous devez orchestrer des charges de travail parallèles à grande échelle répondant à une combinaison des conditions suivantes :

- La taille de votre jeu de données dépasse 256 Ko.
- L'historique des événements d'exécution du flux de travail dépasse 25 000 entrées.
- Vous avez besoin d'une simultanéité de plus de 40 itérations parallèles.

L'exemple de définition d'état de carte distribuée suivant spécifie l'ensemble de données sous la forme d'un fichier CSV stocké dans un compartiment Amazon S3. Elle spécifie également une fonction Lambda qui traite les données de chaque ligne du fichier CSV. Comme cet exemple utilise un fichier CSV, il indique également l'emplacement des en-têtes de colonne CSV. Pour voir la définition complète de la machine à états de cet exemple, consultez le didacticiel Copier des données CSV à grande échelle à l'aide d'une carte distribuée.

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

Autorisations pour exécuter une carte distribuée

Lorsque vous incluez un état de carte distribuée dans vos flux de travail, Step Functions a besoin des autorisations appropriées pour permettre au rôle de machine à états d'invoquer l'action d'[StartExecution](#)API pour l'état de carte distribuée.

L'exemple de politique IAM suivant accorde le minimum de privilèges requis à votre rôle de machine d'état pour exécuter l'état de carte distribuée.

> **Note**
>
> Assurez-vous de *stateMachineName* remplacer par le nom de la machine à états dans laquelle vous utilisez l'état Distributed Map. Par exemple, `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:*"
    }
  ]
}
```

En outre, vous devez vous assurer que vous disposez du minimum de privilèges nécessaires pour accéder aux AWS ressources utilisées dans l'état de la carte distribuée, telles que les buckets Amazon S3. Pour plus d'informations, consultez Politiques IAM pour l'utilisation de l'état de la carte distribuée.

Champs d'état de la carte distribuée

Pour utiliser l'état de la carte distribuée dans vos flux de travail, spécifiez un ou plusieurs de ces champs. Vous spécifiez ces champs en plus des champs d'état courants.

`Type` (Obligatoire)

Définit le type d'état, tel que`Map`.

`ItemProcessor` (Obligatoire)

Contient les objets JSON suivants qui spécifient le mode et la définition de traitement de `Map` l'état.

- `ProcessorConfig`— Objet JSON qui spécifie la configuration de l'`Map`état. Cet objet contient les sous-champs suivants :
  - `Mode`— Paramétré **`DISTRIBUTED`** pour utiliser l'`Map`état en mode distribué.

    > **Note**
    >
    > Actuellement, si vous utilisez l'`Map`état dans les flux de travail Express, vous ne pouvez pas le `Mode` définir sur`DISTRIBUTED`. Toutefois, si vous utilisez l'`Map`état dans les flux de travail standard, vous pouvez `Mode` définir le sur`DISTRIBUTED`.

- `ExecutionType`— Spécifie le type d'exécution du flux de travail cartographique : STANDARD ou EXPRESS. Vous devez fournir ce champ si vous l'avez spécifié `DISTRIBUTED` pour le `Mode` sous-champ. Pour plus d'informations sur les types de flux de travail, consultezFlux de travail standard ou express.

- `StartAt`— Spécifie une chaîne qui indique le premier état d'un flux de travail. Cette chaîne distingue les majuscules et minuscules et doit correspondre au nom de l'un des objets d'état. Cet état s'exécute d'abord pour chaque élément de l'ensemble de données. Toute entrée d'exécution que vous fournissez à l'`Map`état passe d'abord à l'`StartAt`état.

- `States`— Objet JSON contenant un ensemble d'états séparés par des virgules. Dans cet objet, vous définissez leMap workflow.

`ItemReader`

Spécifie un ensemble de données et son emplacement. L'`Map`État reçoit ses données d'entrée de l'ensemble de données spécifié.

En mode distribué, vous pouvez utiliser soit une charge utile JSON transmise depuis un état précédent, soit une source de données Amazon S3 à grande échelle comme ensemble de données. Pour plus d'informations, consultez ItemReader.

`ItemsPath` (facultatif)

Spécifie un chemin de référence en utilisant la JsonPathsyntaxe pour sélectionner le nœud JSON qui contient un tableau d'éléments dans l'entrée d'état.

En mode distribué, vous ne spécifiez ce champ que lorsque vous utilisez un tableau JSON d'une étape précédente comme entrée d'état. Pour plus d'informations, consultez ItemsPath.

`ItemSelector` (facultatif)

Remplace les valeurs des éléments individuels de l'ensemble de données avant qu'elles ne soient transmises à chaque itération `Map` d'état.

Dans ce champ, vous spécifiez une entrée JSON valide contenant une collection de paires clé-valeur. Ces paires peuvent être soit des valeurs statiques que vous définissez dans la définition de votre machine à états, soit des valeurs sélectionnées à partir de l'entrée d'état à l'aide d'un chemin, soit des valeurs accessibles depuis l'objet de contexte. Pour plus d'informations, consultez ItemSelector.

`ItemBatcher` (facultatif)

Spécifie de traiter les éléments de l'ensemble de données par lots. Chaque exécution du flux de travail enfant reçoit ensuite un lot de ces éléments en entrée. Pour plus d'informations, consultez ItemBatcher.

`MaxConcurrency` (facultatif)

Spécifie le nombre d'exécutions de flux de travail enfants qui peuvent être exécutées en parallèle. L'interpréteur n'autorise que le nombre spécifié d'exécutions parallèles de flux de travail enfants. Si vous ne spécifiez pas de valeur de simultanéité ou si vous la définissez sur zéro, Step Functions ne limite pas la simultanéité et exécute 10 000 exécutions parallèles de flux de travail enfants.

> **Note**
>
> Bien que vous puissiez spécifier une limite de simultanéité plus élevée pour les exécutions de flux de travail secondaires parallèles, nous vous recommandons de ne pas dépasser la capacité d'un AWS service en aval, tel que AWS Lambda.

`MaxConcurrencyPath` (facultatif)

Si vous souhaitez fournir une valeur de simultanéité maximale de manière dynamique à partir de l'entrée d'état à l'aide d'un chemin de référence, utilisez`MaxConcurrencyPath`. Une fois résolu, le chemin de référence doit sélectionner un champ dont la valeur est un entier non négatif.

> **Note**
>
> Un `Map` état ne peut pas inclure à la fois `MaxConcurrency` et`MaxConcurrencyPath`.

`ToleratedFailurePercentage` (facultatif)

Définit le pourcentage d'objets ayant échoué à tolérer lors d'une exécution cartographique. Le Map Run échoue automatiquement s'il dépasse ce pourcentage. Step Functions calcule le pourcentage d'éléments ayant échoué en divisant le nombre total d'éléments défaillants ou ayant dépassé le délai imparti par le nombre total d'éléments. Vous devez spécifier une valeur comprise entre zéro et 100. Pour plus d'informations, consultez Seuil de défaillance toléré pour l'état de la carte distribuée.

`ToleratedFailurePercentagePath` (facultatif)

Si vous souhaitez fournir une valeur de pourcentage de défaillance tolérée de manière dynamique à partir de l'entrée d'état en utilisant un chemin de référence, utilisez`ToleratedFailurePercentagePath`. Une fois résolu, le chemin de référence doit sélectionner un champ dont la valeur est comprise entre zéro et 100.

`ToleratedFailureCount` (facultatif)

Définit le nombre d'objets ayant échoué à tolérer lors d'une exécution de carte. Le Map Run échoue automatiquement s'il dépasse ce nombre. Pour plus d'informations, consultez Seuil de défaillance toléré pour l'état de la carte distribuée.

`ToleratedFailureCountPath` (facultatif)

Si vous souhaitez fournir une valeur de nombre de défaillances tolérées de manière dynamique à partir de l'entrée d'état en utilisant un chemin de référence, utilisez`ToleratedFailureCountPath`. Une fois résolu, le chemin de référence doit sélectionner un champ dont la valeur est un entier non négatif.

`Label` (facultatif)

Chaîne qui identifie un `Map` état de manière unique. Pour chaque Map Run, Step Functions ajoute l'étiquette à l'ARN Map Run. Voici un exemple d'ARN Map Run avec une étiquette personnalisée nommée `demoLabel` :

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

Si vous ne spécifiez aucune étiquette, Step Functions génère automatiquement une étiquette unique.

> **Note**
>
> Les étiquettes ne peuvent pas dépasser 40 caractères, doivent être uniques au sein d'une définition de machine à états et ne peuvent contenir aucun des caractères suivants :
>
> - Personnages Whitespace
> - Caractères génériques `() ? *`
> - Caractères entre crochets `(< > { } [ ])`
> - Caractères spéciaux `(: ; , \ | ^ ~ $ # % & ` ")`

- Caractères de contrôle (\\u0000- \\u001f ou \\u007f -\\u009f).

Step Functions vous permet de créer des noms pour les machines d'état, les exécutions, les activités et les étiquettes contenant des caractères non ASCII. Ces noms non ASCII ne fonctionnent pas avec Amazon. CloudWatch Pour être sûr de pouvoir suivre CloudWatch les métriques, choisissez un nom qui utilise uniquement des caractères ASCII.

`ResultWriter` (facultatif)

Spécifie l'emplacement Amazon S3 où Step Functions écrit tous les résultats d'exécution du flux de travail enfant.

Step Functions consolide toutes les données d'exécution du flux de travail enfant, telles que les entrées et sorties d'exécution, l'ARN et le statut d'exécution. Il exporte ensuite les exécutions avec le même statut vers leurs fichiers respectifs à l'emplacement Amazon S3 spécifié. Pour plus d'informations, consultez ResultWriter.

Si vous n'exportez pas les résultats de `Map` l'état, il renvoie un tableau de tous les résultats d'exécution du flux de travail enfant. Par exemple :

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

`ResultPath` (facultatif)

Spécifie l'endroit de l'entrée où placer la sortie des itérations. L'entrée est ensuite filtrée comme spécifié par le `OutputPath`champ s'il est présent, avant d'être transmise comme sortie de l'état. Pour plus d'informations, consultez Traitement des entrées et des sorties.

`ResultSelector` (facultatif)

Transmettez une collection de paires clé-valeur, dont les valeurs sont statiques ou sélectionnées à partir du résultat. Pour plus d'informations, consultez ResultSelector.

> **Tip**
>
> Si l'état Parallel ou Map que vous utilisez dans vos machines d'état renvoie un tableau de tableaux, vous pouvez les transformer en tableau plat avec le ResultSelector champ. Pour plus d'informations, consultez Aplatir un tableau de tableaux.

`Retry` (facultatif)

Tableau d'objets, appelés Retriers, qui définit une politique de nouvelle tentative. Une exécution utilise la politique de nouvelle tentative si l'état rencontre des erreurs d'exécution. Pour plus d'informations, consultez Exemples de machines à états utilisant Retry et Catch.

> **Note**
>
> Si vous définissez des récupérateurs pour l'état de la carte distribuée, la politique de nouvelles tentatives s'applique à toutes les exécutions de flux de travail enfants lancées par l'`Map`état. Par exemple, imaginez que votre `Map` État a lancé trois exécutions de flux de travail secondaires, dont une échoue. Lorsque l'échec se produit, l'exécution utilise le `Retry` champ, s'il est défini, pour l'`Map`état. La politique de nouvelle tentative s'applique à toutes les exécutions de flux de travail secondaires et pas seulement à celles qui ont échoué. Si une ou plusieurs exécutions de flux de travail enfants échouent, le Map Run échoue.
>
> Lorsque vous réessayez un `Map` état, il crée un nouveau Map Run.

`Catch` (facultatif)

Tableau d'objets, nommés Receveurs, qui définissent un état de secours. Step Functions utilise les Catchers définis dans `Catch` si l'état rencontre des erreurs d'exécution. Lorsqu'une erreur se produit, l'exécution utilise d'abord les récupérateurs définis dans`Retry`. Si la politique de nouvelle tentative n'est pas définie ou est épuisée, l'exécution utilise ses Catchers, s'ils sont définis. Pour plus d'informations, consultez États de secours.

Étapes suivantes

Pour en savoir plus sur l'état des cartes distribuées, consultez les ressources suivantes :

- Traitement des entrées et des sorties

  Pour configurer l'entrée qu'un état de carte distribuée reçoit et la sortie qu'il génère, Step Functions fournit les champs suivants :

  - ItemReader
  - ItemsPath
  - ItemSelector

- ItemBatcher
- ResultWriter
- Analyse d'un fichier CSV d'entrée

  Outre ces champs, Step Functions vous permet également de définir un seuil d'échec toléré pour Distributed Map. Cette valeur vous permet de spécifier le nombre maximum ou le pourcentage d'éléments ayant échoué comme seuil d'échec pour une exécution cartographique. Pour plus d'informations sur la configuration du seuil de défaillance toléré, consultezSeuil de défaillance toléré pour l'état de la carte distribuée.

- Utilisation de l'état de la carte distribuée

  Reportez-vous aux didacticiels et exemples de projets suivants pour commencer à utiliser l'état des cartes distribuées.

  - Commencer à utiliser Distributed Map State
  - Traitement d'un lot complet de données avec une fonction Lambda
  - Traitement d'éléments de données individuels à l'aide d'une fonction Lambda
  - Exemple de projet : traitement d'un fichier CSV avec une carte distribuée
  - Exemple de projet : traiter des données dans un compartiment Amazon S3 avec Distributed Map

- Examiner l'exécution de l'état des cartes distribuées

  La console Step Functions fournit une page Map Run Details, qui affiche toutes les informations relatives à l'exécution d'un état de carte distribuée. Pour plus d'informations sur la façon d'examiner les informations affichées sur cette page, consultezExamen de Map Run.

Seuil de défaillance toléré pour l'état de la carte distribuée

Lorsque vous orchestrez des charges de travail parallèles à grande échelle, vous pouvez également définir un seuil de défaillance toléré. Cette valeur vous permet de spécifier le nombre maximum ou le pourcentage d'éléments ayant échoué comme seuil d'échec pour une exécution cartographique. En fonction de la valeur que vous spécifiez, votre Map Run échoue automatiquement si elle dépasse le seuil. Si vous spécifiez les deux valeurs, le flux de travail échoue lorsqu'il dépasse l'une ou l'autre de ces valeurs.

La spécification d'un seuil vous permet d'échouer à un certain nombre d'éléments avant que la totalité de la Map Run échoue. Step Functions renvoie une

`States.ExceedToleratedFailureThreshold` erreur lorsque le Map Run échoue parce que le seuil spécifié est dépassé.

> **Note**
>
> Step Functions peut continuer à exécuter des flux de travail secondaires dans un Map Run même après le dépassement du seuil d'échec toléré, mais avant que le Map Run échoue.

Pour spécifier la valeur du seuil dans Workflow Studio, sélectionnez Définir un seuil d'échec toléré dans Configuration supplémentaire dans le champ Paramètres d'exécution.

Pourcentage de défaillances tolérées

Définit le pourcentage d'éléments défaillants à tolérer. Votre Map Run échoue si cette valeur est dépassée. Step Functions calcule le pourcentage d'éléments ayant échoué en divisant le nombre total d'éléments défaillants ou ayant dépassé le délai imparti par le nombre total d'éléments. Vous devez spécifier une valeur comprise entre zéro et 100. La valeur en pourcentage par défaut est zéro, ce qui signifie que le flux de travail échoue si l'une de ses exécutions de flux de travail enfant échoue ou expire. Si vous spécifiez le pourcentage comme 100, le flux de travail n'échouera pas même si toutes les exécutions de flux de travail enfants échouent.

Vous pouvez également spécifier le pourcentage comme chemin de référence vers une paire clé-valeur existante dans l'entrée d'état de votre carte distribuée. Ce chemin doit être résolu en un entier positif compris entre 0 et 100 au moment de l'exécution. Vous spécifiez le chemin de référence dans le `ToleratedFailurePercentagePath` sous-champ.

Par exemple, avec les données d'entrée suivantes :

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

Vous pouvez spécifier le pourcentage en utilisant un chemin de référence vers cette entrée comme suit :

```
{
  ...
  "Map": {
```

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

> **Important**
>
> Vous pouvez spécifier l'un `ToleratedFailurePercentage` ou l'autre`ToleratedFailurePercentagePath`, mais pas les deux dans la définition de l'état de votre carte distribuée.

Nombre de défaillances tolérées

Définit le nombre d'éléments défaillants à tolérer. Votre Map Run échoue si cette valeur est dépassée.

Vous pouvez également spécifier le nombre comme chemin de référence vers une paire clé-valeur existante dans l'entrée d'état de votre carte distribuée. Ce chemin doit être résolu en un entier positif au moment de l'exécution. Vous spécifiez le chemin de référence dans le `ToleratedFailureCountPath` sous-champ.

Par exemple, avec les données d'entrée suivantes :

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

Vous pouvez spécifier le numéro en utilisant un chemin de référence vers cette entrée comme suit :

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

```
  }
}
```

> **Important**
>
> Vous pouvez spécifier l'un `ToleratedFailureCount` ou l'autre`ToleratedFailureCountPath`, mais pas les deux dans la définition de l'état de votre carte distribuée.

Transitions

Lorsque vous lancez une nouvelle exécution de votre machine à états, le système commence par l'état référencé dans le `StartAt` champ de niveau supérieur. Ce champ, fourni sous forme de chaîne, doit correspondre exactement, majuscules et minuscules, au nom d'un état dans le flux de travail.

Après l'exécution d'un état, AWS Step Functions utilise la valeur du `Next` champ pour déterminer l'état suivant vers lequel passer.

`Next`les champs spécifient également les noms d'état sous forme de chaînes. Cette chaîne distingue les majuscules et minuscules et doit correspondre exactement au nom d'un état spécifié dans la description de la machine à états

Par exemple, l'état suivant comprend une transition vers `NextState`.

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

La plupart des États n'autorisent qu'une seule règle de transition avec `Next` ce champ. Cependant, certains états de contrôle de flux, tels qu'un `Choice` état, vous permettent de définir plusieurs règles de transition, chacune avec son propre champ. `Next` Le Langage des états d'Amazon fournit des détails sur chaque type d'état que vous pouvez spécifier, notamment les informations relatives à la spécification des transitions.

Les états peuvent avoir plusieurs transitions entrantes issues d'autres états.

Le processus se répète jusqu'à ce qu'il atteigne un état terminal (un état avec `"Type": Succeed"Type": Fail`, ou`"End": true`) ou qu'une erreur d'exécution se produise.

Lorsqu'il s'agit d'redriveune exécution, elle est considérée comme une transition d'état. En outre, tous les états réexécutés dans a redrive sont également considérés comme des transitions d'états.

Les règles suivantes s'appliquent aux états au sein d'une machine d'état :

- Les états peuvent apparaître dans n'importe quel ordre au sein du bloc englobant. Toutefois, l'ordre dans lequel ils sont listés n'a aucune incidence sur l'ordre dans lequel ils sont exécutés. Cet ordre est déterminé par le contenu des états.
- Dans une machine à états, il ne peut y avoir qu'un seul état désigné comme `start` état. L'`start`état est défini par la valeur du `StartAt` champ dans la structure de niveau supérieur.
- Selon la logique de votre machine à états (par exemple, si votre machine à états possède plusieurs branches logiques), vous pouvez avoir plusieurs `end` états.
- Si votre machine à états ne comprend qu'un seul état, il peut s'agir à la fois de l'état de début et de fin.

Transitions dans l'état de la carte distribuée

Lorsque vous utilisez l'`Map`état en mode distribué, une transition d'état vous est facturée pour chaque exécution de flux de travail enfant lancée par l'état de carte distribuée. Lorsque vous utilisez l'`Map`état en mode Inline, aucune transition d'état ne vous est facturée pour chaque itération de l'état Inline Map.

Vous pouvez optimiser les coûts en utilisant l'`Map`état en mode distribué et en incluant un flux de travail imbriqué dans la définition de l'`Map`état. L'état de la carte distribuée ajoute également de la valeur lorsque vous lancez des exécutions de flux de travail enfants de type Express. Step Functions enregistre la réponse et le statut des exécutions du flux de travail Express Child, ce qui réduit le besoin de stocker les données d'exécution dans CloudWatch les journaux. Vous pouvez également accéder aux contrôles de flux disponibles avec un état de carte distribuée, tels que la définition de seuils d'erreur ou le traitement par lots d'un groupe d'éléments. Pour plus d'informations sur la tarification de Step Functions, consultez la section AWS Step Functionstarification.

Données de la machine d'état

Les données de la machine d'état peuvent prendre les formes suivantes :

- L'entrée initiale dans une machine d'état
- Les données transmises entre les états
- La sortie d'une machine d'état

Cette section décrit la façon dont les données de la machine d'état sont mises en forme et utilisées dans AWS Step Functions.

Rubriques

Format de données

Les données de la machine d'état sont représentées par du texte JSON. Vous pouvez fournir des valeurs à une machine d'état à l'aide de n'importe quel type de données pris en charge par JSON.

> **Note**
>
> - Les nombres au format texte JSON sont conformes à la JavaScript sémantique. Ces numéros correspondent généralement à des valeurs IEEE 854 à double précision.
> - Le texte JSON suivant est valide :
>   - Chaînes autonomes, délimitées par des guillemets
>   - Objets
>   - Arrays (tableaux)
>   - Nombres
>   - Valeurs booléennes
>   - `null`
> - La sortie d'un état devient l'entrée de l'état suivant. Toutefois, vous pouvez restreindre les états pour qu'ils fonctionnent sur un sous-ensemble des données d'entrée en utilisant le traitement des entrées et des sorties.

Entrées/Sorties de la machine d'état

Vous pouvez transmettre vos données d'entrée initiales à une AWS Step Functions machine à états de deux manières. Vous pouvez transmettre les données à une `StartExecution` action lorsque vous démarrez une exécution. Vous pouvez également transmettre les données à la machine d'état depuis la console Step Functions. Les données initiales sont transmises à l'état `StartAt` de la machine d'état. Si aucune entrée n'est fournie, la valeur par défaut est un objet vide (`{}`).

La sortie de l'exécution est renvoyée par le dernier état (`terminal`). Cette sortie s'affiche comme texte JSON dans le résultat de l'exécution.

Pour les flux de travail standard, vous pouvez récupérer les résultats d'exécution à partir de l'historique des exécutions à l'aide d'appelants externes, tels que l'`DescribeExecution`action. Vous pouvez consulter les résultats de l'exécution sur la console Step Functions.

Pour Express Workflows, si vous avez activé la journalisation, vous pouvez récupérer les résultats à partir CloudWatch des journaux ou afficher et déboguer les exécutions dans la console Step Functions. Pour plus d'informations, consultez Journalisation à l'aideCloudWatchJournaux et Affichage et débogage des exécutions sur la console Step Functions.

Vous devez également prendre en compte les quotas liés à votre machine d'État. Pour de plus amples informations, consultez Quotas.

Entrées/Sorties d'état

Chaque entrée d'état est composée d'un texte JSON issu de l'état précédent ou, pour l'état `StartAt`, de l'entrée dans l'exécution. Certains états de contrôle de flux ont une entrée et une sortie identiques.

Dans l'exemple suivant, la machine d'état ajoute deux nombres l'un à l'autre.

1. Définissez la fonction AWS Lambda.

```
function Add(input) {
  var numbers = JSON.parse(input).numbers;
  var total = numbers.reduce(
    function(previousValue, currentValue, index, array) {
      return previousValue + currentValue; });
  return JSON.stringify({ result: total });
}
```

2. Définissez la machine d'état .

```
{
  "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. Commencez une exécution avec le texte JSON suivant.

```
{ "numbers": [3, 4] }
```

L'Addétat reçoit le texte JSON et le transmet à la fonction Lambda.

La fonction Lambda renvoie le résultat du calcul à l'état.

L'état renvoie la valeur suivante dans sa sortie.

```
{ "result": 7 }
```

Dans la mesure où Add est également l'état final dans la machine d'état, cette valeur est retournée comme sortie de la machine d'état.

Si l'état final ne retourne aucune sortie, cela signifie que la machine d'état renvoie un objet vide ({}).

Pour plus d'informations, veuillez consulter Traitement des entrées et des sorties dans Step Functions.

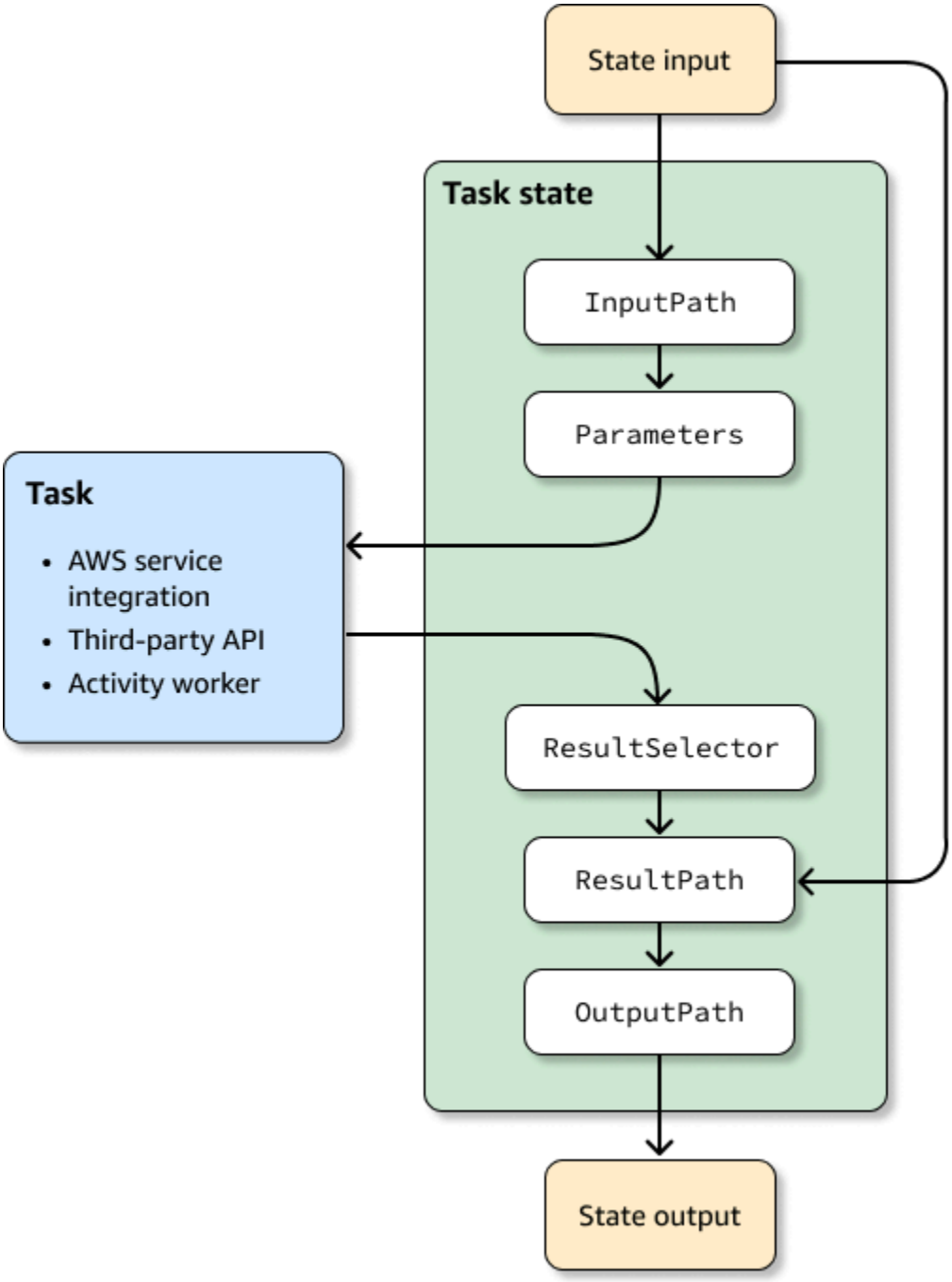
Traitement des entrées et des sorties dans Step Functions

Une exécution de Step Functions reçoit un texte JSON en entrée et transmet cette entrée au premier état du flux de travail. Chaque état reçoit un fichier JSON comme entrée, qu'il transmet généralement en tant que sortie à l'état suivant. Pour concevoir et implémenter efficacement des flux de travail dans AWS Step Functions, il est essentiel de bien comprendre comment ces informations passent d'un état à l'autre et d'apprendre à filtrer et à manipuler ces données.

Dans l'Amazon States Language, ces champs filtrent et contrôlent le flux de JSON d'un État à l'autre :

- `InputPath`
- `Parameters`
- `ResultSelector`
- `ResultPath`
- `OutputPath`

Le schéma suivant montre comment les informations JSON passent par l'état d'une tâche. `InputPath`sélectionne les parties de l'entrée JSON à transmettre à la tâche de l'`Task`état (par exemple, une AWS Lambda fonction). `ResultPath`sélectionne ensuite la combinaison de l'entrée d'état et du résultat de la tâche à transmettre à la sortie. `OutputPath`peut filtrer la sortie JSON pour limiter davantage les informations transmises à la sortie.



`InputPath`,, `Parameters` `ResultSelector``ResultPath`, et `OutputPath` chacun manipule le JSON au fur et à mesure qu'il passe par chaque état de votre flux de travail.

Chacun peut utiliser des chemins pour sélectionner des parties du JSON de l'entrée ou du résultat. Un chemin est une chaîne commençant par$, qui identifie les nœuds dans le texte JSON. Les chemins Step Functions utilisent JsonPathla syntaxe.

> **Tip**
>
> Utilisez le simulateur de flux de données de la console Step Functions pour tester la syntaxe des chemins JSON, pour mieux comprendre comment les données sont manipulées au sein d'un état et pour voir comment les données sont transmises entre les états.

> **Tip**
>
> Pour déployer un exemple de flux de travail incluant le traitement des entrées et des sorties sur votre ordinateur Compte AWS, consultez le Module 6 - Traitement des entrées et des sorties de The AWS Step Functions Workshop.

Rubriques

Chemins

Dans l'Amazon States Language, un chemin est une chaîne commençant par $ laquelle vous pouvez identifier les composants dans le texte JSON. Les chemins suivent JsonPathla syntaxe. Vous pouvez spécifier un chemin pour accéder à des sous-ensembles de l'entrée lors de la spécification des valeurs pour `InputPath`, `ResultPath` et `OutputPath`. Pour plus d'informations, consultez Traitement des entrées et des sorties dans Step Functions.

> **Note**
>
> Vous pouvez également spécifier un nœud JSON de l'entrée ou de l'objet de contexte à l'aide de chemins d'accès dans le champ `Parameters` d'une définition d'état. veuillez consulter Transmettre des paramètres à une API de service.

Vous devez utiliser la notation entre crochets si le nom de votre champ contient un caractère qui n'est pas inclus dans la `member-name-shorthand` définition de la règle JsonPath ABNF. Par conséquent, pour coder des caractères spéciaux, tels que les signes de ponctuation (sauf_), vous devez utiliser la notation entre crochets. Par exemple, `$.abc.['def ghi']`.

Chemins de référence

Un chemin de référence est un chemin dont la syntaxe est limitée de telle sorte qu'elle ne peut identifier qu'un seul nœud dans une structure JSON :

- Vous pouvez accéder aux champs des objets uniquement en utilisant la notation point (`.`) et crochet (`[ ]`).
- Les fonctions telles que `length()` ne sont pas prises en charge.
- Les opérateurs lexicaux, qui ne sont pas symboliques, par exemple, `subsetof` ne sont pas pris en charge.
- Le filtrage par expression régulière ou par référence à une autre valeur dans la structure JSON n'est pas pris en charge.
- Les opérateurs`@`, `,`, `:`, et ne `?` sont pas pris en charge

Par exemple, les données d'entrée d'état contiennent les valeurs suivantes :

```
{
  "foo": 123,
  "bar": ["a", "b", "c"],
  "car": {
      "cdr": true
  }
}
```

Dans ce cas, les chemins de référence suivants renverraient :

```
$.foo => 123
$.bar => ["a", "b", "c"]
$.car.cdr => true
```

Certains états utilisent des chemins d'accès et des chemins de référence pour contrôler le flux d'une machine d'état ou configurer les paramètres ou les options d'un état. Pour plus d'informations, consultez Modélisation du traitement des chemins d'entrée et de sortie du flux de travail avec un simulateur de flux de données et Utilisation efficace de JSONPath dans. AWS Step Functions

Aplatir un tableau de tableaux

Si l'Mapétat Parallèle ou de vos machines d'état renvoie un tableau de tableaux, vous pouvez les transformer en tableau plat avec le ResultSelector champ. Vous pouvez inclure ce champ dans la définition de l'état parallèle ou de l'état cartographique pour manipuler le résultat de ces états.

Pour aplatir les tableaux, utilisez la syntaxe JMESpath `[*]` dans le `ResultSelector` champ, comme indiqué dans l'exemple suivant.

```
"ResultSelector": {
    "flattenArray.$": "$[*][*]"
  }
```

Pour des exemples illustrant comment aplatir un tableau, reportez-vous à l'étape 3 des didacticiels suivants :

- Traitement d'un lot complet de données avec une fonction Lambda
- Traitement d'éléments de données individuels à l'aide d'une fonction Lambda

InputPath, Paramètres et ResultSelector

Les `ResultSelector` champs`InputPath`, `Parameters` et permettent de manipuler le JSON au fur et à mesure qu'il se déplace dans votre flux de travail. `InputPath`peut limiter l'entrée transmise en filtrant la notation JSON à l'aide d'un chemin (voirChemins). Le champ `Parameters` vous permet de transmettre un ensemble de paires clé-valeur, où les valeurs sont des valeurs statiques que vous définissez dans la définition de votre machine d'état, ou qui sont sélectionnées à partir de l'entrée à l'aide d'un chemin. Le `ResultSelector` champ permet de manipuler le résultat de l'état avant `ResultPath` son application.

AWS Step Functions applique d'abord `InputPath` le champ, puis le `Parameters` champ. Vous pouvez d'abord filtrer vos données d'entrée brutes que vous souhaitez à l'aide de `InputPath`, puis appliquer `Parameters` pour manipuler davantage cette entrée, ou pour ajouter de nouvelles valeurs. Vous pouvez ensuite utiliser le `ResultSelector` champ pour manipuler la sortie de l'état avant `ResultPath` son application.

> **Tip**
>
> Utilisez le simulateur de flux de données de la console Step Functions pour tester la syntaxe des chemins JSON, pour mieux comprendre comment les données sont manipulées au sein d'un état et pour voir comment les données sont transmises entre les états.

InputPath

Utilisez `InputPath` pour sélectionner une partie de l'entrée de l'état.

Par exemple, si l'entrée de votre état comprend les éléments suivants :

```
{
  "comment": "Example for InputPath.",
  "dataset1": {
    "val1": 1,
    "val2": 2,
    "val3": 3
  },
  "dataset2": {
    "val1": "a",
    "val2": "b",
    "val3": "c"
  }
}
```

Vous pouvez appliquer le `InputPath`.

```
"InputPath": "$.dataset2",
```

Avec le précédent `InputPath`, ce qui suit est le JSON transmis en tant qu'entrée.

```
{
```

```
  "val1": "a",
  "val2": "b",
  "val3": "c"
}
```

> **Note**
>
> Un chemin peut produire une sélection de valeurs. Prenez l'exemple de code suivant.
>
> ```
> { "a": [1, 2, 3, 4] }
> ```
>
> Si vous appliquez le chemin `$.a[0:2]`, le résultat est le suivant :
>
> ```
> [ 1, 2 ]
> ```

Paramètres

Cette section décrit les différentes manières d'utiliser le champ Paramètres.

Paires clé-valeur

Utilisez le `Parameters` champ pour créer une collection de paires clé-valeur qui sont transmises en entrée. Les valeurs de chacun peuvent être des valeurs statiques que vous intégrez dans la définition de votre machine d'état ou que vous sélectionnez à partir de l'entrée avec un chemin. Pour les paires clé-valeur dans lesquelles la valeur est sélectionnée à l'aide d'un chemin, le nom de clé doit se terminer par `.$`.

Par exemple, supposons que vous fournissiez les entrées suivantes.

```
{
  "comment": "Example for Parameters.",
  "product": {
    "details": {
       "color": "blue",
       "size": "small",
       "material": "cotton"
    },
    "availability": "in stock",
    "sku": "2317",
```

```
    "cost": "$23"
  }
}
```

Pour sélectionner certaines informations, vous pouvez spécifier ces paramètres dans la définition de votre machine d'état.

```
"Parameters": {
        "comment": "Selecting what I care about.",
        "MyDetails": {
          "size.$": "$.product.details.size",
          "exists.$": "$.product.availability",
          "StaticValue": "foo"
        }
      },
```

Soit les entrées précédentes et le champ `Parameters`, il s'agit du JSON qui est transmis.

```
{
  "comment": "Selecting what I care about.",
  "MyDetails": {
      "size": "small",
      "exists": "in stock",
      "StaticValue": "foo"
  }
},
```

En plus de l'entrée, vous pouvez accéder à un objet JSON spécial, connu sous le nom d'objet de contexte. L'objet de contexte comprend des informations sur l'exécution de votre machine d'état. veuillez consulter Objet Contexte.

Ressources connectées

Le champ `Parameters` peut également transmettre des informations aux ressources connectées. Par exemple, si l'état de votre tâche orchestre une AWS Batch tâche, vous pouvez transmettre les paramètres d'API pertinents directement aux actions d'API de ce service. Pour plus d’informations, consultez :

- Transmettre des paramètres à une API de service
- Utilisation avec d'autres services

Amazon S3

Si les données de la fonction Lambda que vous transmettez entre les états peuvent atteindre plus de 262 144 octets, nous vous recommandons d'utiliser Amazon S3 pour stocker les données et d'implémenter l'une des méthodes suivantes :

- Utilisez l'état de la carte distribuée dans votre flux de travail afin que l'`Map`état puisse lire les entrées directement depuis les sources de données Amazon S3. Pour plus d'informations, consultez Utilisation de l'état de la carte en mode distribué.
- Analysez le nom Amazon Resource (ARN) du bucket dans le `Payload` paramètre pour obtenir le nom du bucket et la valeur clé. Pour plus d'informations, consultez Utilisez les ARN d'Amazon S3 au lieu de transmettre des charges utiles importantes.

Vous pouvez également ajuster votre implémentation pour transmettre des charges utiles plus faibles lors de vos exécutions.

ResultSelector

Utilisez le `ResultSelector` champ pour manipuler le résultat d'un état avant `ResultPath` son application. Le `ResultSelector` champ vous permet de créer une collection de paires clé-valeur, dont les valeurs sont statiques ou sélectionnées à partir du résultat de l'état. À l'aide du `ResultSelector` champ, vous pouvez choisir les parties du résultat d'un état que vous souhaitez transmettre au `ResultPath` champ.

> **Note**
>
> Avec le `ResultPath` champ, vous pouvez ajouter la sortie du `ResultSelector` champ à l'entrée d'origine.

`ResultSelector`est un champ facultatif dans les états suivants :

- Map
- État de la tâche
- Parallèle

Par exemple, les intégrations du service Step Functions renvoient des métadonnées en plus de la charge utile contenue dans le résultat. `ResultSelector`peut sélectionner des parties du résultat et

les fusionner avec l'entrée d'état avec`ResultPath`. Dans cet exemple, nous voulons sélectionner uniquement le `resourceType` et`ClusterId`, puis le fusionner avec l'entrée d'état d'un Amazon EMR CreateCluster.sync. Compte tenu de ce qui suit :

```
{
  "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"
  }
}
```

Vous pouvez ensuite sélectionner `resourceType` et `ClusterId` en utilisant `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"
}
```

Avec l'entrée donnée, l'utilisation `ResultSelector` produit :

```
{
  "OtherDataFromInput": {},
  "EMROutput": {
    "ResourceType": "elasticmapreduce",
    "ClusterId": "AKIAIOSFODNN7EXAMPLE"
  }
}
```

Aplatir un tableau de tableaux

Si l'Mapétat Parallèle ou de vos machines d'état renvoie un tableau de tableaux, vous pouvez les transformer en tableau plat avec le ResultSelector champ. Vous pouvez inclure ce champ dans la définition de l'état parallèle ou de l'état cartographique pour manipuler le résultat de ces états.

Pour aplatir les tableaux, utilisez la syntaxe JMESpath `[*]` dans le `ResultSelector` champ, comme indiqué dans l'exemple suivant.

```
"ResultSelector": {
    "flattenArray.$": "$[*][*]"
  }
```

Pour des exemples illustrant comment aplatir un tableau, reportez-vous à l'étape 3 des didacticiels suivants :

- Traitement d'un lot complet de données avec une fonction Lambda
- Traitement d'éléments de données individuels à l'aide d'une fonction Lambda

ResultPath

La sortie d'un état peut être une copie de son entrée, le résultat qu'il génère (par exemple, la sortie d'une fonction Lambda pour l'état d'une `Task`) ou une combinaison de l'entrée et du résultat. Utilisez `ResultPath` pour contrôler les combinaisons de ces stratégies transmises à la sortie de l'état.

Les types d'état suivants peuvent générer un résultat et peuvent inclure `ResultPath:`

- Pass
- État de la tâche
- Parallèle

- Map

Utilisez `ResultPath` pour combiner un résultat de tâche avec une entrée de tâche ou pour sélectionner l'une de ces règles. Le chemin que vous fournissez à `ResultPath` détermine quelles informations sont transmises à la sortie.

> **Note**
>
> Le `ResultPath` est limité à l'utilisation des chemins de référence, laquelle limite la portée afin d'identifier un seul nœud dans JSON. Voir Chemins de référence dans le document Langage des états d'Amazon.

Ces exemples sont basés sur la machine à états et la fonction Lambda décrites dans le Création d'une machine d'état Step Functions utilisant Lambda didacticiel. Passez en revue ce didacticiel et testez les différentes sorties en essayant d'indiquer différents chemins dans un champ `ResultPath`.

Utilisez ResultPath pour :

- ResultPath À utiliser pour remplacer l'entrée par le résultat
- Supprimer le résultat et conserver l'entrée d'origine
- ResultPath À utiliser pour inclure le résultat dans l'entrée
- ResultPath À utiliser pour mettre à jour un nœud dans l'entrée avec le résultat
- ResultPath À utiliser pour inclure à la fois une erreur et une entrée dans un Catch

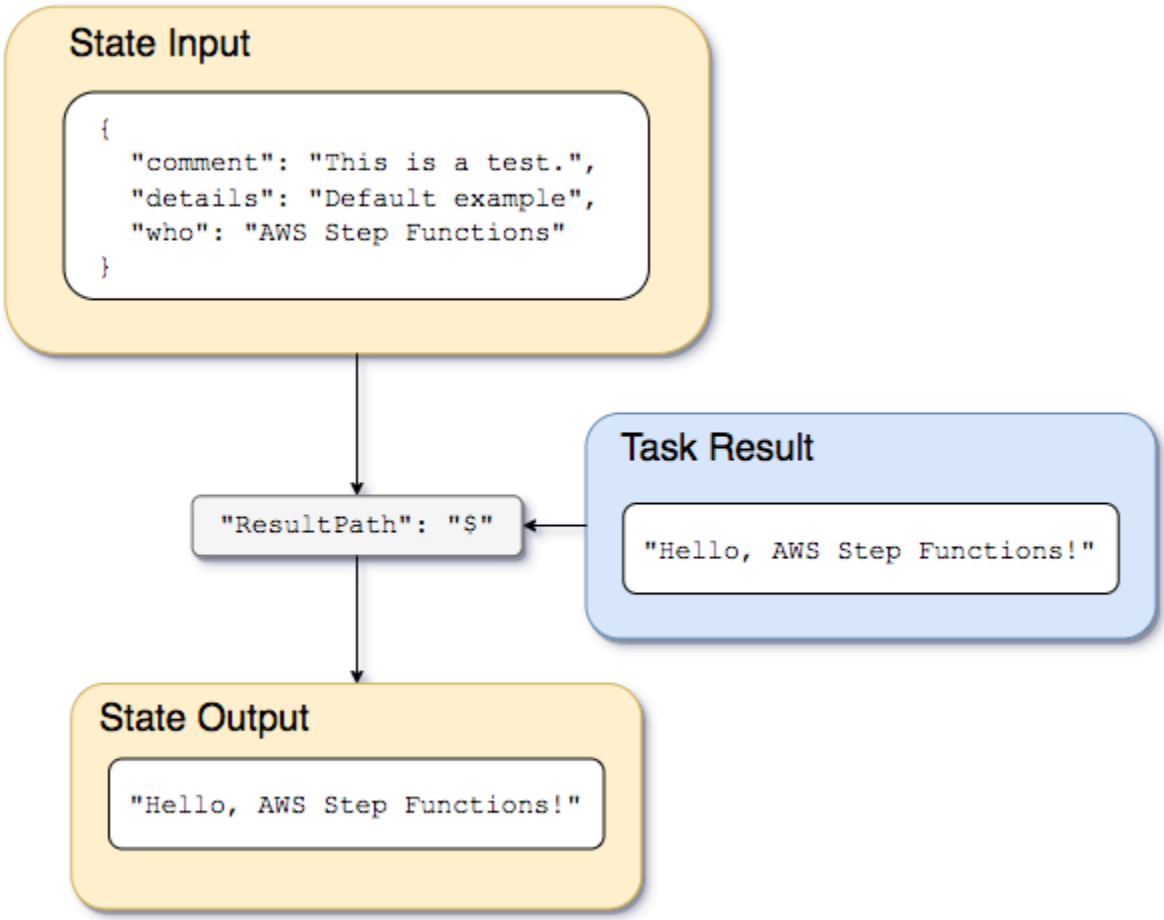
> **Tip**
>
> Utilisez le simulateur de flux de données de la console Step Functions pour tester la syntaxe des chemins JSON, pour mieux comprendre comment les données sont manipulées au sein d'un état et pour voir comment les données sont transmises entre les états.

ResultPath À utiliser pour remplacer l'entrée par le résultat

Si vous n'indiquez pas de `ResultPath`, le comportement par défaut est tel que si vous aviez indiqué `"ResultPath": "$"`. Étant donné que cette option indique à l'état de remplacer la totalité de

l'entrée par le résultat, l'entrée de l'état est complètement remplacée par le résultat issu du résultat de la tâche.

Le schéma suivant montre comment `ResultPath` peut totalement remplacer l'entrée par le résultat de la tâche.



Utilisez la machine à états et la fonction Lambda décrites dansCréation d'une machine d'état Step Functions utilisant Lambda, et remplacez le type d'intégration de service par intégration AWS SDK pour la fonction Lambda. Pour ce faire, spécifiez la fonction Lambda Amazon Resource Name (ARN) dans le `Resource` champ de l'`Task`état, comme indiqué dans l'exemple suivant. L'intégration du AWS SDK garantit que le résultat de l'`Task`état ne contient que la sortie de la fonction Lambda, sans aucune métadonnée.

```
{
   "StartAt":"CallFunction",
   "States":{
```

```
      "CallFunction": {
         "Type":"Task",
         "Resource":"arn:aws:lambda:us-east-2:123456789012:function:HelloFunction",
         "End": true
      }
   }
}
```

Transmettez ensuite l'entrée suivante :

```
{
  "comment": "This is a test of the input and output of a Task state.",
  "details": "Default example",
  "who": "AWS Step Functions"
}
```

La fonction Lambda fournit le résultat suivant.

```
"Hello, AWS Step Functions!"
```

> **Tip**
>
> Vous pouvez consulter ce résultat sur la Step Functionsconsole. Pour ce faire, sur la page Détails de l'exécution de la console, choisissez la Lambda fonction dans la vue graphique. Choisissez ensuite l'onglet Sortie dans le Détails de l'étape volet pour voir ce résultat.

S'il `ResultPath` n'est pas spécifié dans l'état, ou s'il `"ResultPath": "$"` est défini, l'entrée de l'état est remplacée par le résultat de la fonction Lambda, et la sortie de l'état est la suivante.

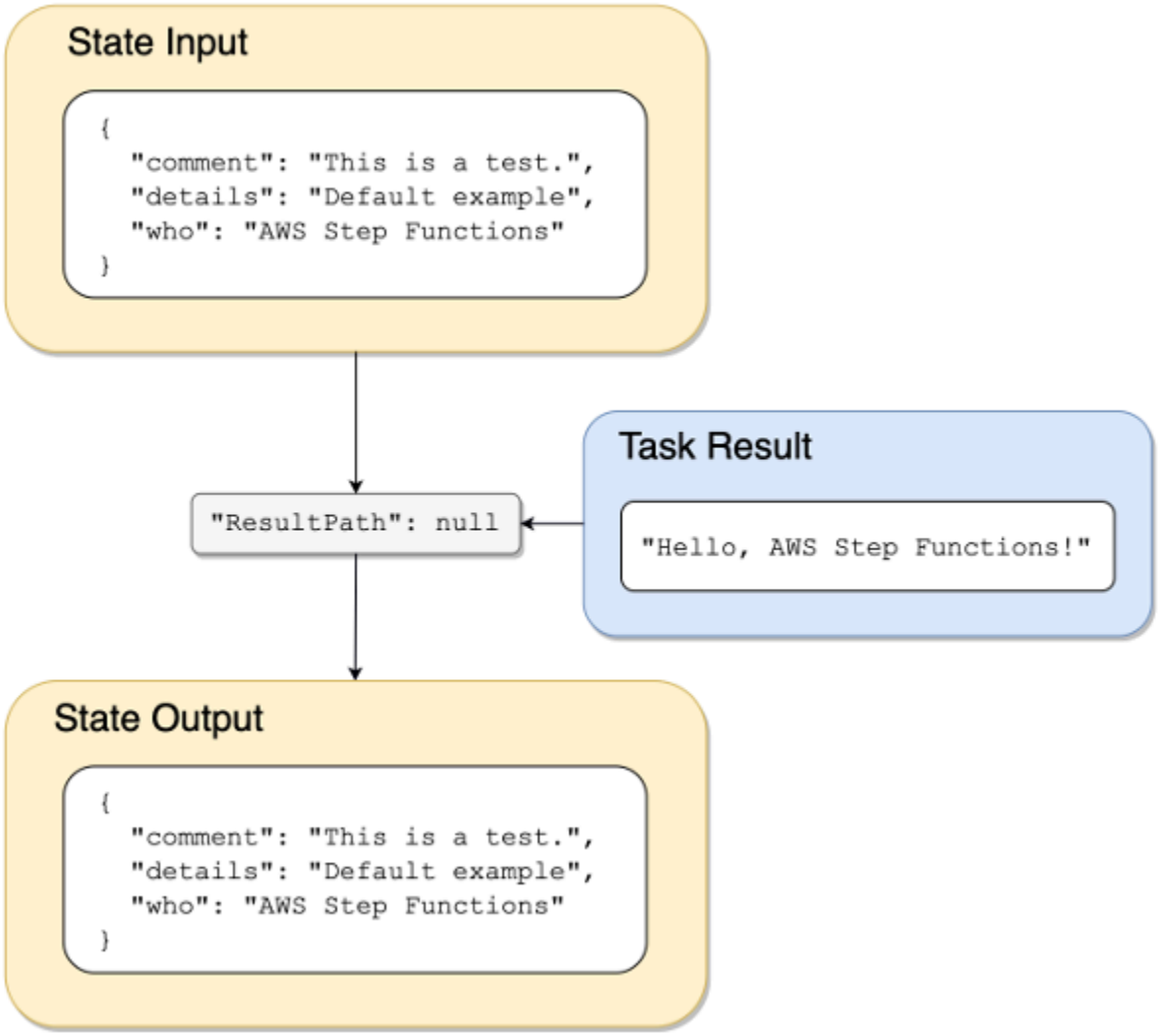
```
"Hello, AWS Step Functions!"
```

> **Note**
>
> `ResultPath` est utilisé pour inclure du contenu issu du résultat avec les données d'entrée, avant de les transmettre à la sortie. Toutefois, si `ResultPath` n'est pas spécifié, le comportement par défaut consiste à remplacer la totalité de l'entrée.

Supprimer le résultat et conserver l'entrée d'origine

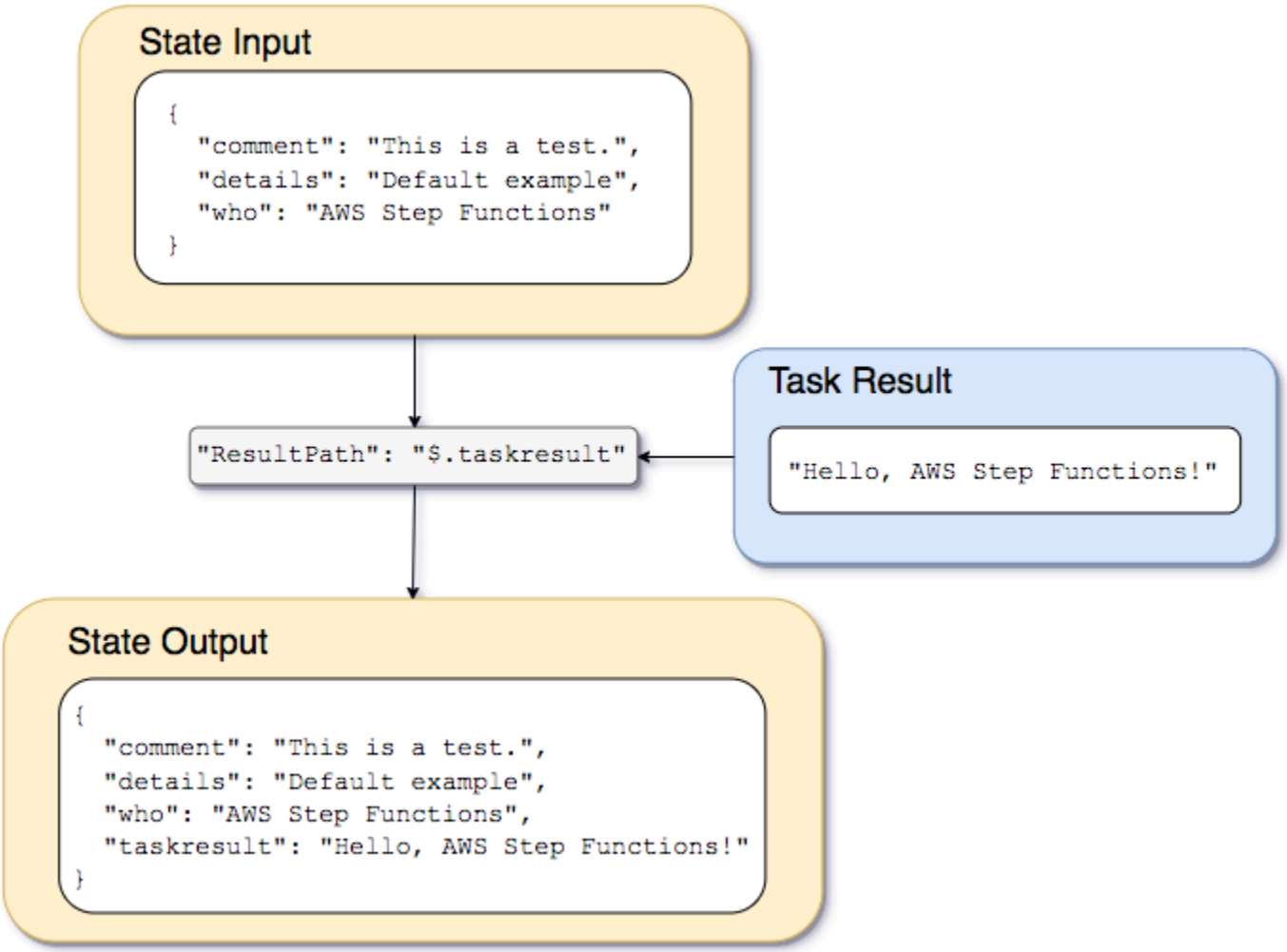
Si vous définissez `ResultPath` sur `null`, il transmet l'entrée d'origine à la sortie. À l'aide de `"ResultPath": null`, la charge utile d'entrée de l'état est copiée directement dans la sortie, sans tenir compte du résultat.

Le diagramme suivant montre comment un null `ResultPath` copie l'entrée directement dans la sortie.



ResultPath À utiliser pour inclure le résultat dans l'entrée

Le schéma suivant montre comment `ResultPath` peut inclure le résultat avec l'entrée.



En utilisant la machine à états et la fonction Lambda décrites dans le Création d'une machine d'état Step Functions utilisant Lambda didacticiel, nous pourrions transmettre l'entrée suivante.

```
{
  "comment": "This is a test of the input and output of a Task state.",
  "details": "Default example",
  "who": "AWS Step Functions"
}
```

Le résultat de la fonction Lambda est le suivant.

```
"Hello, AWS Step Functions!"
```

Pour préserver l'entrée, insérez le résultat de la fonction Lambda, puis passez le JSON combiné à l'état suivant, que nous pourrions `ResultPath` définir comme suit.

```
"ResultPath": "$.taskresult"
```

Cela inclut le résultat de la fonction Lambda avec l'entrée d'origine.

```
{
  "comment": "This is a test of input and output of a Task state.",
  "details": "Default behavior example",
  "who": "AWS Step Functions",
  "taskresult": "Hello, AWS Step Functions!"
}
```

La sortie de la fonction Lambda est ajoutée à l'entrée d'origine sous forme de valeur pour. `taskresult` L'entrée, y compris la nouvelle valeur insérée, est transmise à l'état suivant.

Vous pouvez également insérer le résultat dans un nœud enfant de l'entrée. Définissez `ResultPath` comme suit.

```
"ResultPath": "$.strings.lambdaresult"
```

Commencez une exécution avec l'entrée suivante.

```
{
  "comment": "An input comment.",
  "strings": {
    "string1": "foo",
    "string2": "bar",
    "string3": "baz"
  },
  "who": "AWS Step Functions"
}
```

Le résultat de la fonction Lambda est inséré en tant qu'enfant du `strings` nœud dans l'entrée.

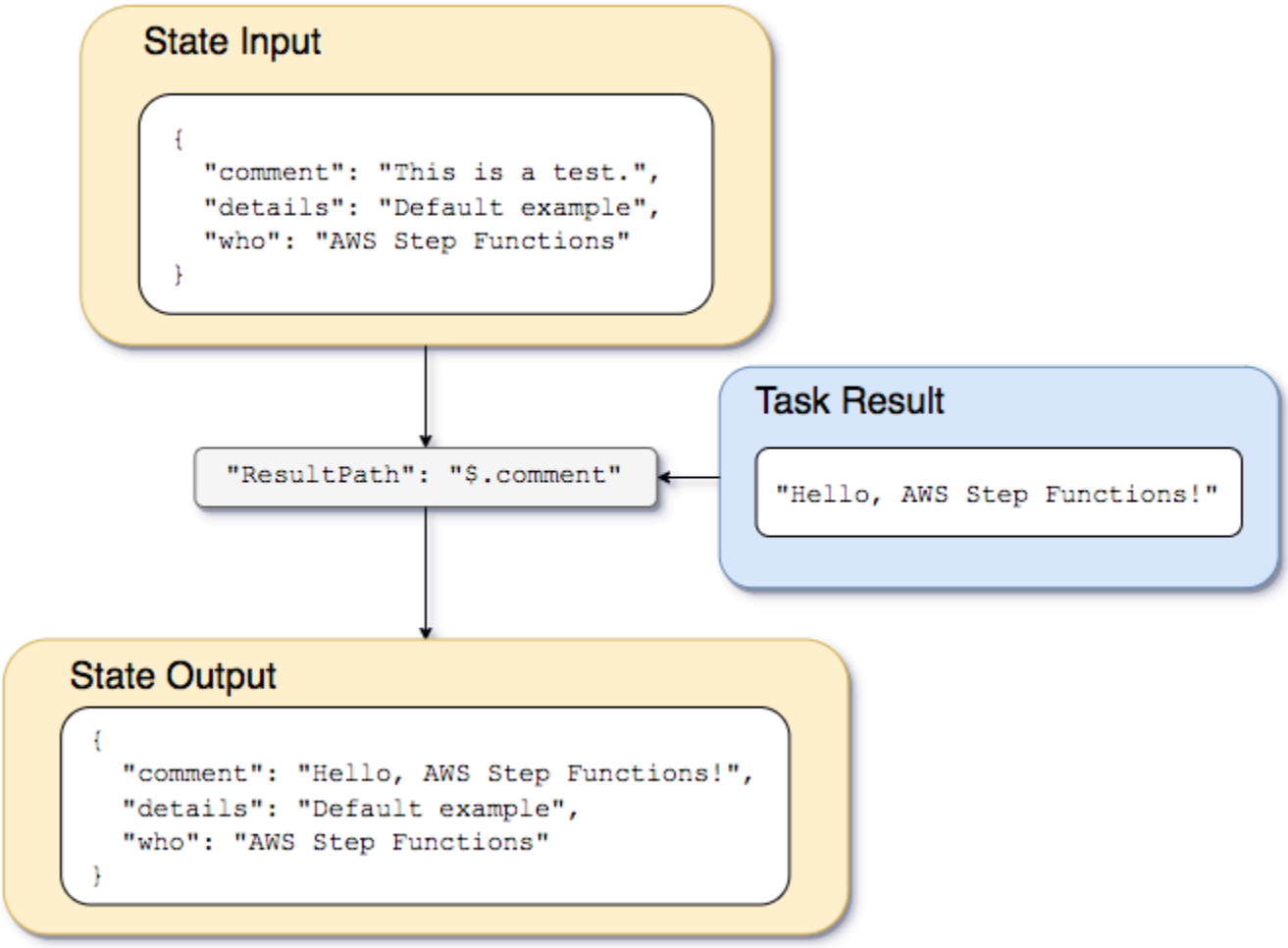
```
{
  "comment": "An input comment.",
  "strings": {
```

```
    "string1": "foo",
    "string2": "bar",
    "string3": "baz",
    "lambdaresult": "Hello, AWS Step Functions!"
  },
  "who": "AWS Step Functions"
}
```

La sortie de l'état inclut désormais le JSON d'entrée d'origine avec le résultat sous forme d'un nœud enfant.

ResultPath À utiliser pour mettre à jour un nœud dans l'entrée avec le résultat

Le schéma suivant montre comment `ResultPath` peut mettre à jour la valeur des nœuds JSON existants dans l'entrée avec des valeurs issues du résultat de la tâche.



En utilisant l'exemple de la machine à états et de la fonction Lambda décrits dans le Création d'une machine d'état Step Functions utilisant Lambda didacticiel, nous pourrions transmettre l'entrée suivante.

```
{
  "comment": "This is a test of the input and output of a Task state.",
  "details": "Default example",
  "who": "AWS Step Functions"
}
```

Le résultat de la fonction Lambda est le suivant.

```
Hello, AWS Step Functions!
```

Au lieu de préserver l'entrée et d'insérer le résultat en tant que nouveau nœud dans le fichier JSON, nous pouvons remplacer un nœud existant.

Par exemple, tout comme le fait d'omettre ou de paramétrer `"ResultPath": "$"` remplace l'ensemble du nœud, vous pouvez spécifier un nœud en particulier à remplacer par le résultat.

```
"ResultPath": "$.comment"
```

Comme le `comment` nœud existe déjà dans l'entrée d'état, le paramètre `ResultPath` to `"$.comment"` remplace ce nœud dans l'entrée par le résultat de la fonction Lambda. Sans filtrer davantage par `OutputPath`, les données suivantes sont transmises à la sortie.

```
{
  "comment": "Hello, AWS Step Functions!",
  "details": "Default behavior example",
  "who": "AWS Step Functions",
}
```

La valeur du `comment` nœud`"This is a test of the input and output of a Task state."`, est remplacée par le résultat de la fonction Lambda : `"Hello, AWS Step Functions!"` dans la sortie d'état.

ResultPath À utiliser pour inclure à la fois une erreur et une entrée dans un **`Catch`**

Le didacticiel Gestion des conditions d'erreur à l'aide d'une machine à états Step Functions explique comment utiliser une machine d'état pour intercepter une erreur. Dans certains cas, il se peut que

vous souhaitiez conserver l'entrée d'origine avec l'erreur. Utilisez `ResultPath` dans un champ `Catch` pour inclure l'erreur avec l'entrée d'origine, au lieu de la remplacer :

```
"Catch": [{
  "ErrorEquals": ["States.ALL"],
  "Next": "NextTask",
  "ResultPath": "$.error"
}]
```

Si l'instruction `Catch` précédente intercepte une erreur, elle inclut le résultat dans un nœud `error` avec l'entrée d'état. Par exemple, pour l'entrée suivante :

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

La sortie de l'état lors de l'interception de l'erreur est :

```
{
  "foo": "bar",
  "error": {
    "Error": "Error here"
  }
}
```

Pour plus d'informations sur la gestion des erreurs, voir :

- Gestion des erreurs dans Step Functions
- Gestion des conditions d'erreur à l'aide d'une machine à états Step Functions

OutputPath

`OutputPath` vous permet de sélectionner une portion de la sortie d'état pour passer au prochain état. Ceci vous permet de filtrer les informations indésirables et de passer uniquement la portion de JSON qui vous intéresse.

Si vous ne spécifiez pas une valeur `OutputPath`, la valeur par défaut est $. Cela transmet la totalité du nœud JSON (déterminé par l'état d'entrée, la tâche et les résultats `ResultPath`) à l'état suivant.

> **Tip**
>
> Utilisez le simulateur de flux de données de la console Step Functions pour tester la syntaxe des chemins JSON, pour mieux comprendre comment les données sont manipulées au sein d'un état et pour voir comment les données sont transmises entre les états.

Pour plus d'informations, consultez les ressources suivantes :

- Chemins dans la langue des États d'Amazon
- InputPath,ResultPath, et OutputPath exemples
- Transmettre du JSON statique en tant que paramètres
- Traitement des entrées et des sorties dans Step Functions

InputPath,ResultPath, et OutputPath exemples

Tout état autre qu'un Fail état ou un Succeed état peut inclure les champs de traitement d'entrée et de sortie, tels que `InputPathResultPath`, ou`OutputPath`. De plus, les Choice états Attente et ne prennent pas en charge `ResultPath` ce champ. Avec ces champs, vous pouvez utiliser a JsonPathpour filtrer les données JSON au fur et à mesure de leur évolution dans votre flux de travail.

Vous pouvez également utiliser le `Parameters` champ pour manipuler les données JSON au fur et à mesure de leur évolution dans votre flux de travail. Pour obtenir des informations sur l'utilisation d'`Parameters`, veuillez consulter InputPath, Paramètres et ResultSelector.

Par exemple, démarrez avec la fonction AWS Lambda et la machine d'état décrites dans le didacticiel Création d'une machine d'état Step Functions utilisant Lambda. Modifiez la machine d'état afin qu'elle comprenne les éléments `InputPath`, `ResultPath` et `OutputPath` suivants.

```
{
  "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
    }
  }
}
```

Commencez une exécution avec l'entrée suivante.

```
{
  "comment": "An input comment.",
  "data": {
    "val1": 23,
    "val2": 17
  },
  "extra": "foo",
  "lambda": {
    "who": "AWS Step Functions"
  }
}
```

Supposons que les `extra` nœuds `comment` et puissent être supprimés, mais que nous souhaitions inclure la sortie de la fonction Lambda et conserver les informations dans le nœud. `data`

Dans la machine d'état mise à jour, l'état `Task` est modifié pour traiter l'entrée pour la tâche.

```
"InputPath": "$.lambda",
```

Cette ligne dans la définition de la machine d'état limite l'entrée de tâche au nœud `lambda` de l'entrée d'état uniquement. La fonction Lambda reçoit uniquement l'objet JSON `{"who": "AWS Step Functions"}` en entrée.

```
"ResultPath": "$.data.lambdaresult",
```

Cela `ResultPath` indique à la machine d'état d'insérer le résultat de la fonction Lambda dans un nœud nommé`lambdaresult`, en tant qu'enfant du `data` nœud dans l'entrée de la machine d'état d'origine. Comme nous n'effectuons aucune autre manipulation sur l'entrée d'origine et sur le résultat en utilisant`OutputPath`, la sortie de l'état inclut désormais le résultat de la fonction Lambda avec l'entrée d'origine.

```
{
```

```
  "comment": "An input comment.",
  "data": {
    "val1": 23,
    "val2": 17,
    "lambdaresult": "Hello, AWS Step Functions!"
  },
  "extra": "foo",
  "lambda": {
    "who": "AWS Step Functions"
  }
}
```

Mais notre objectif était de ne conserver que le `data` nœud et d'inclure le résultat de la fonction Lambda. `OutputPath`filtre ce JSON combiné avant de le transmettre à la sortie d'état.

```
"OutputPath": "$.data",
```

Ainsi, seul le nœud `data` de l'entrée d'origine est sélectionné (y compris l'enfant `lambdaresult` inséré par `ResultPath`) comme devant être transmis à la sortie. La sortie de l'état est filtrée sur ce qui suit.

```
{
  "val1": 23,
  "val2": 17,
  "lambdaresult": "Hello, AWS Step Functions!"
}
```

Dans cet état `Task` :

1. `InputPath`envoie uniquement le `lambda` nœud de l'entrée à la fonction Lambda.
2. `ResultPath` insère le résultat en tant qu'enfant du nœud `data` dans l'entrée d'origine.
3. `OutputPath`filtre l'entrée d'état (qui inclut désormais le résultat de la fonction Lambda) afin qu'elle transmette uniquement le `data` nœud à la sortie d'état.

Example pour manipuler l'entrée, le résultat et la sortie finale de la machine à l'état d'origine en utilisant JsonPath

Prenons l'exemple de la machine d'État suivante qui vérifie l'identité et l'adresse d'un demandeur d'assurance.

> **Note**
>
> Pour voir l'exemple complet, voir Comment utiliser le chemin JSON dans Step Functions.

```
{
  "Comment": "Sample state machine to verify an applicant's ID and address",
  "StartAt": "Verify info",
  "States": {
    "Verify info": {
      "Type": "Parallel",
      "End": true,
      "Branches": [
        {
          "StartAt": "Verify identity",
          "States": {
            "Verify identity": {
              "Type": "Task",
              "Resource": "arn:aws:states:::lambda:invoke",
              "Parameters": {
                "Payload.$": "$",
                "FunctionName": "arn:aws:lambda:us-east-2:111122223333:function:check-
identity:$LATEST"
              },
              "End": true
            }
          }
        },
        {
          "StartAt": "Verify address",
          "States": {
            "Verify address": {
              "Type": "Task",
              "Resource": "arn:aws:states:::lambda:invoke",
              "Parameters": {
                "Payload.$": "$",
                "FunctionName": "arn:aws:lambda:us-east-2:111122223333:function:check-
address:$LATEST"
              },
              "End": true
            }
          }
```

```
        }
      ]
    }
  }
}
```

Si vous exécutez cette machine d'état à l'aide de l'entrée suivante, l'exécution échoue car les fonctions Lambda qui effectuent la vérification attendent uniquement les données qui doivent être vérifiées en tant qu'entrée. Par conséquent, vous devez spécifier les nœuds qui contiennent les informations à vérifier à l'aide d'un nœud appropriéJsonPath.

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

```
}
```

Pour spécifier le nœud que la fonction *check-identity* Lambda doit utiliser, utilisez le `InputPath` champ comme suit :

```
"InputPath": "$.data.identity"
```

Et pour spécifier le nœud que la fonction *check-address* Lambda doit utiliser, utilisez le `InputPath` champ comme suit :

```
"InputPath": "$.data.address"
```

Maintenant, si vous souhaitez enregistrer le résultat de la vérification dans l'entrée d'origine de la machine d'état, utilisez le `ResultPath` champ comme suit :

```
"ResultPath": "$.results"
```

Toutefois, si vous n'avez besoin que des résultats d'identité et de vérification et que vous ignorez la saisie d'origine, utilisez le `OutputPath` champ comme suit :

```
"OutputPath": "$.results"
```

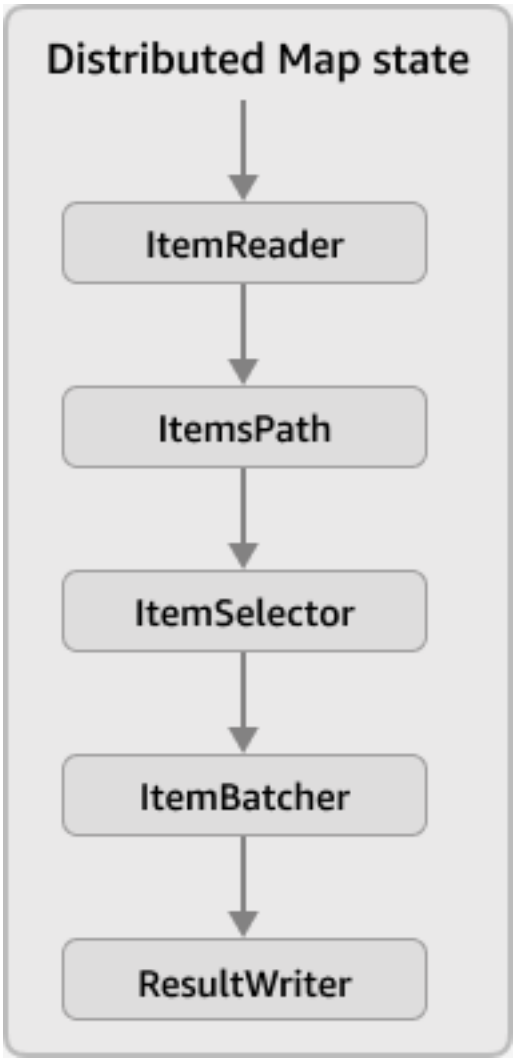
Pour plus d'informations, veuillez consulter Traitement des entrées et des sorties dans Step Functions.

Champs d'entrée et de sortie de l'état de la carte

`Map`les états parcourent simultanément un ensemble d'éléments d'un ensemble de données, tel qu'un tableau JSON, une liste d'objets Amazon S3 ou les lignes d'un fichier CSV dans un compartiment Amazon S3. Il répète une série d'étapes pour chaque élément de la collection. Vous pouvez configurer l'entrée que l'`Map`état reçoit et la sortie qu'il génère à l'aide de ces champs. Step Functions applique chaque champ de l'état de votre carte distribuée dans l'ordre indiqué dans la liste et l'illustration suivantes :

> **Note**
>
> Selon votre cas d'utilisation, il se peut que vous n'ayez pas à remplir tous ces champs.

1. ItemReader
2. ItemsPath
3. ItemSelector
4. ItemBatcher
5. ResultWriter



> **Note**
>
> Ces champs d'entrée et de sortie d'`Map`état ne sont actuellement pas disponibles dans le simulateur de flux de données de la console Step Functions.

ItemReader

Le`ItemReader`Un champ est un objet JSON qui spécifie un ensemble de données et son emplacement. UNÉtat de la carte distribuéeutilise cet ensemble de données comme entrée. Le jeu de données suivant montre la syntaxe du`ItemReader`champ si votre ensemble de données est un fichier CSV qui est stocké dans un compartiment Amazon S3.

```
"ItemReader": {
    "ReaderConfig": {
        "InputType": "CSV",
        "CSVHeaderLocation": "FIRST_ROW"
    },
    "Resource": "arn:aws:states:::s3:getObject",
    "Parameters": {
        "Bucket": "myBucket",
        "Key": "csvDataset/ratings.csv"
    }
}
```

> **Tip**
>
> Dans Workflow Studio, vous spécifiez le jeu de données et son emplacement dansSource de l'articlechamp.

Table des matières

Contenu du ItemReader champ

En fonction de votre ensemble de données, le contenu du`ItemReader`le champ varie. Par exemple, si votre ensemble de données est un tableau JSON transmis lors d'une étape précédente du flux de travail,`ItemReader`le champ est omis. Si votre ensemble de données est une source de données Amazon S3, ce champ contient les sous-champs suivants.

`ReaderConfig`

Un objet JSON qui spécifie les détails suivants :

- `InputType`

  Spécifie le type de source de données Amazon S3, tel qu'un fichier CSV, un objet, un fichier JSON ou une liste d'inventaire Amazon S3. Dans Workflow Studio, vous pouvez sélectionner un type d'entrée dansSource d'article Amazon S3liste déroulante située sous leSource de l'articlechamp.

- `CSVHeaderLocation`

  > **Note**
  >
  > Vous devez spécifier ce champ uniquement si vous utilisez un fichier CSV comme ensemble de données.

  Accepte l'une des valeurs suivantes pour spécifier l'emplacement de l'en-tête de colonne :

  > **Important**
  >
  > Step Functions prend actuellement en charge les en-têtes CSV d'une taille maximale de 10 Ko.

  - `FIRST_ROW`— Utilisez cette option si la première ligne du fichier est l'en-tête.
  - `GIVEN`— Utilisez cette option pour spécifier l'en-tête dans la définition de la machine à états. Par exemple, si votre fichier CSV contient les données suivantes.

    ```
    1,307,3.5,1256677221
    1,481,3.5,1256677456
    1,1091,1.5,1256677471
    ...
    ```

    Fournissez le tableau JSON suivant sous forme d'en-tête CSV.

    ```
    "ItemReader": {
        "ReaderConfig": {
    ```

```
        "InputType": "CSV",
        "CSVHeaderLocation": "GIVEN",
        "CSVHeaders": [
            "userId",
            "movieId",
            "rating",
            "timestamp"
        ]
    }
}
```

> **Tip**
>
> Dans Workflow Studio, vous pouvez trouver cette option sousConfiguration supplémentairedans leSource de l'articlechamp.

- `MaxItems`

  Limite le nombre d'éléments de données transmis au`Map`état. Supposons, par exemple, que vous fournissiez un fichier CSV contenant 1 000 lignes et que vous spécifiiez une limite de 100. Ensuite, l'interprète passeseulement100 lignes jusqu'au`Map`état. Le`Map`state traite les éléments dans un ordre séquentiel, en commençant après la ligne d'en-tête.

  Par défaut, le`Map`state itère sur tous les éléments de l'ensemble de données spécifié.

  > **Note**
  >
  > À l'heure actuelle, vous pouvez définir une limite maximale de 100 000 000. LeÉtat de la carte distribuéearrête de lire les éléments au-delà de cette limite.

  > **Tip**
  >
  > Dans Workflow Studio, vous pouvez trouver cette option sousConfiguration supplémentairedans leSource de l'articlechamp.

Vous pouvez également spécifier unechemin de référenceà une paire clé-valeur existante dans votreÉtat de la carte distribuéeentrée. Le chemin doit être résolu en un entier positif. Vous spécifiez le chemin de référence dans`MaxItemsPath`sous-champ.

> **Important**
>
> Vous pouvez spécifier soit le`MaxItems`ou le`MaxItemsPath`sous-champ, mais pas les deux.

`Resource`

L'action d'API Amazon S3 que Step Functions doit invoquer en fonction de l'ensemble de données spécifié.

`Parameters`

Objet JSON qui spécifie le nom du compartiment Amazon S3 et la clé d'objet dans lesquels l'ensemble de données est stocké.

> **Important**
>
> Assurez-vous que vos compartiments Amazon S3 se trouvent sous le mêmeCompte AWSetRégion AWScomme votre machine d'État.

Exemples de jeux de données

Vous pouvez spécifier l'une des options suivantes comme jeu de données :

- Tableau JSON d'une étape précédente
- Liste des objets Amazon S3
- Fichier JSON dans un compartiment Amazon S3
- Fichier CSV dans un compartiment Amazon S3
- Inventaire Amazon S3

> **Important**
>
> Step Functions a besoin des autorisations appropriées pour accéder aux ensembles de données Amazon S3 que vous utilisez. Pour plus d'informations sur les politiques IAM pour les jeux de données, voirPolitiques IAM pour les jeux de données.

Tableau JSON d'une étape précédente

UNÉtat de la carte distribuéepeut accepter une entrée JSON transmise à partir d'une étape précédente du flux de travail. Cette entrée doit être soit un tableau, soit contenir un tableau dans un nœud spécifique. Pour sélectionner un nœud contenant le tableau, vous pouvez utiliser`ItemsPath`champ.

Pour traiter les éléments individuels du tableau,État de la carte distribuéelance l'exécution d'un flux de travail enfant pour chaque élément du tableau. Les onglets suivants présentent des exemples d'entrées transmises au`Map`état et entrée correspondante pour l'exécution d'un flux de travail enfant.

> **Note**
>
> Step Functions omet le`ItemReader`champ lorsque votre ensemble de données est un tableau JSON issu d'une étape précédente.

Input passed to the Map state

Considérez le tableau JSON suivant de trois éléments.

```
"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

LeÉtat de la carte distribuéelance trois exécutions de flux de travail secondaires. Chaque exécution reçoit un élément de tableau en entrée. L'exemple suivant montre l'entrée reçue par l'exécution d'un flux de travail enfant.

```
{
  "verdict": "true",
  "statement_date": "6/11/2008",
  "statement_source": "speech"
}
```

Exemple d'objets Amazon S3

UNÉtat de la carte distribuéepeut itérer sur les objets stockés dans un compartiment Amazon S3. Lorsque l'exécution du flux de travail atteint le`Map`state, Step Functions invoque leListObjectsV2Action d'API, qui renvoie un tableau des métadonnées de l'objet Amazon S3. Dans ce tableau, chaque élément contient des données, telles queETagetClé, pour les données stockées dans le bucket.

Pour traiter les éléments individuels du tableau,État de la carte distribuéelance l'exécution d'un flux de travail enfant. Supposons par exemple que votre compartiment Amazon S3 contienne 100 images. Ensuite, le tableau est retourné après avoir invoqué le`ListObjectsV2`L'action de l'API contient 100 éléments. LeÉtat de la carte distribuéelance ensuite 100 exécutions de flux de travail enfants pour traiter chaque élément du tableau.

> **Note**
>
> - À l'heure actuelle, Step Functions inclut également un élément pour chaque dossier que vous créez dans un compartiment Amazon S3 à l'aide de la console Amazon S3. Cela entraîne l'exécution d'un flux de travail secondaire supplémentaire lancé par leÉtat de la carte distribuée. Pour éviter de créer une exécution de flux de travail secondaire

supplémentaire pour le dossier, nous vous recommandons d'utiliserAWS CLIpour créer des dossiers. Pour plus d'informations, veuillez consulter la rubriqueCommandes Amazon S3 de haut niveaudans leAWS Command Line InterfaceGuide d'utilisateur.

- Step Functions a besoin des autorisations appropriées pour accéder aux ensembles de données Amazon S3 que vous utilisez. Pour plus d'informations sur les politiques IAM pour les jeux de données, voirPolitiques IAM pour les jeux de données.

Les onglets suivants présentent des exemples de`ItemReader`syntaxe du champ et entrée transmise à l'exécution d'un flux de travail enfant pour cet ensemble de données.

ItemReader syntax

Dans cet exemple, vous avez organisé vos données, qui incluent des images, des fichiers JSON et des objets, dans un préfixe nommé`processData`dans un compartiment Amazon S3 nommé`myBucket`.

```
"ItemReader": {
    "Resource": "arn:aws:states:::s3:listObjectsV2",
    "Parameters": {
        "Bucket": "myBucket",
        "Prefix": "processData"
    }
}
```

Input passed to a child workflow execution

LeÉtat de la carte distribuéelance autant d'exécutions de flux de travail enfants que le nombre d'éléments présents dans le compartiment Amazon S3. L'exemple suivant montre l'entrée reçue par l'exécution d'un flux de travail enfant.

```
{
  "Etag": "\"05704fbdccb224cb01c59005bebbad28\"",
  "Key": "processData/images/n02085620_1073.jpg",
  "LastModified": 1668699881,
  "Size": 34910,
  "StorageClass": "STANDARD"
}
```

Fichier JSON dans un compartiment Amazon S3

UNÉtat de la carte distribuéepeut accepter un fichier JSON qui est stocké dans un compartiment Amazon S3 en tant que jeu de données. Le fichier JSON doit contenir un tableau.

Lorsque l'exécution du flux de travail atteint le`Map`state, Step Functions invoque leGetObjectAction d'API pour récupérer le fichier JSON spécifié. Le`Map`state itère ensuite chaque élément du tableau et lance l'exécution d'un flux de travail enfant pour chaque élément. Par exemple, si votre fichier JSON contient 1 000 éléments de tableau,`Map`state lance 1000 exécutions de flux de travail enfants.

> **Note**
>
> - L'entrée d'exécution utilisée pour démarrer l'exécution d'un flux de travail enfant ne peut pas dépasser 256 Ko. Step Functions permet toutefois de lire un élément d'une taille maximale de 8 Mo à partir d'un fichier CSV ou JSON si vous appliquez ensuite l'option facultative`ItemSelector`champ pour réduire la taille de l'article.
> - Actuellement, Step Functions prend en charge 10 Go comme taille maximale d'un fichier individuel dans un rapport d'inventaire Amazon S3. Step Functions peut toutefois traiter plus de 10 Go si la taille de chaque fichier est inférieure à 10 Go.
> - Step Functions a besoin des autorisations appropriées pour accéder aux ensembles de données Amazon S3 que vous utilisez. Pour plus d'informations sur les politiques IAM pour les jeux de données, voirPolitiques IAM pour les jeux de données.

Les onglets suivants présentent des exemples de`ItemReader`syntaxe du champ et entrée transmise à l'exécution d'un flux de travail enfant pour cet ensemble de données.

Dans cet exemple, imaginez que vous avez un fichier JSON nommé*`factcheck.json`*. Vous avez enregistré ce fichier dans un préfixe nommé*`jsonDataset`*dans un compartiment Amazon S3. Le jeu de données suivant est un exemple de jeu de données JSON.

```
[
  {
    "verdict": "true",
    "statement_date": "6/11/2008",
    "statement_source": "speech"
  },
  {
```

```
    "verdict": "false",
    "statement_date": "6/7/2022",
    "statement_source": "television"
  },
  {
    "verdict": "mostly-true",
    "statement_date": "5/18/2016",
    "statement_source": "news"
  },
  ...
]
```

ItemReader syntax

```
"ItemReader": {
    "Resource": "arn:aws:states:::s3:getObject",
    "ReaderConfig": {
        "InputType": "JSON"
    },
    "Parameters": {
        "Bucket": "myBucket",
        "Key": "jsonDataset/factcheck.json"
    }
}
```

Input to a child workflow execution

LeÉtat de la carte distribuéelance autant d'exécutions de flux de travail enfants que le nombre d'éléments du tableau présents dans le fichier JSON. L'exemple suivant montre l'entrée reçue par l'exécution d'un flux de travail enfant.

```
{
  "verdict": "true",
  "statement_date": "6/11/2008",
  "statement_source": "speech"
}
```

Fichier CSV dans un compartiment Amazon S3

UNÉtat de la carte distribuéepeut accepter un fichier CSV stocké dans un compartiment Amazon S3 en tant que jeu de données. Si vous utilisez un fichier CSV comme ensemble de données, vous

devez spécifier un en-tête de colonne CSV. Pour plus d'informations sur la spécification d'un en-tête CSV, voirContenu du ItemReader champ.

Comme il n'existe pas de format standardisé pour créer et gérer les données dans les fichiers CSV, Step Functions analyse les fichiers CSV selon les règles suivantes :

- Les virgules (,) sont des séparateurs qui séparent les champs individuels.
- Les nouvelles lignes sont un séparateur qui sépare les enregistrements individuels.
- Les champs sont traités comme des chaînes. Pour les conversions de types de données, utilisez`States.StringToJson`fonction intrinsèque dansItemSelector.
- Les guillemets doubles (» «) ne sont pas nécessaires pour placer les chaînes. Toutefois, les chaînes placées entre guillemets doubles peuvent contenir des virgules et des nouvelles lignes sans qu'elles servent de délimiteurs.
- Évitez les guillemets doubles en les répétant.
- Si le nombre de champs d'une ligne est inférieur au nombre de champs de l'en-tête, Step Functions fournit des chaînes vides pour les valeurs manquantes.
- Si le nombre de champs d'une ligne est supérieur au nombre de champs de l'en-tête, Step Functions ignore les champs supplémentaires.

Pour plus d'informations sur la façon dont Step Functions analyse un fichier CSV, voirExample of parsing an input CSV file.

Lorsque l'exécution du flux de travail atteint le`Map`state, Step Functions invoque leGetObjectAction d'API pour récupérer le fichier CSV spécifié. Le`Map`state itère ensuite chaque ligne du fichier CSV et lance l'exécution d'un flux de travail enfant pour traiter les éléments de chaque ligne. Supposons, par exemple, que vous fournissiez un fichier CSV contenant 100 lignes en entrée. Ensuite, l'interpréteur transmet chaque ligne au`Map`état. Le`Map`state traite les éléments par ordre de série, en commençant après la ligne d'en-tête.

> **Note**
>
> - L'entrée d'exécution utilisée pour démarrer l'exécution d'un flux de travail enfant ne peut pas dépasser 256 Ko. Step Functions permet toutefois de lire un élément d'une taille maximale de 8 Mo à partir d'un fichier CSV ou JSON si vous appliquez ensuite l'option facultative`ItemSelector`champ pour réduire la taille de l'article.

- Actuellement, Step Functions prend en charge 10 Go comme taille maximale d'un fichier individuel dans un rapport d'inventaire Amazon S3. Step Functions peut toutefois traiter plus de 10 Go si la taille de chaque fichier est inférieure à 10 Go.
- Step Functions a besoin des autorisations appropriées pour accéder aux ensembles de données Amazon S3 que vous utilisez. Pour plus d'informations sur les politiques IAM pour les jeux de données, voirPolitiques IAM pour les jeux de données.

Les onglets suivants présentent des exemples de`ItemReader`syntaxe du champ et entrée transmise à l'exécution d'un flux de travail enfant pour cet ensemble de données.

ItemReader syntax

Par exemple, supposons que vous avez un fichier CSV nommé*`ratings.csv`*. Ensuite, vous avez stocké ce fichier dans un préfixe nommé*`csvDataset`*dans un compartiment Amazon S3.

```
{
  "ItemReader": {
    "ReaderConfig": {
      "InputType": "CSV",
      "CSVHeaderLocation": "FIRST_ROW"
    },
    "Resource": "arn:aws:states:::s3:getObject",
    "Parameters": {
      "Bucket": "myBucket",
      "Key": "csvDataset/ratings.csv"
    }
  }
}
```

Input to a child workflow execution

LeÉtat de la carte distribuéelance autant d'exécutions de flux de travail enfant que le nombre de lignes présentes dans le fichier CSV, à l'exception de la ligne d'en-tête, si elle se trouve dans le fichier. L'exemple suivant montre l'entrée reçue par l'exécution d'un flux de travail enfant.

```
{
  "rating": "3.5",
  "movieId": "307",
  "userId": "1",
```

```
  "timestamp": "1256677221"
}
```

Exemple d'inventaire S3

UNÉtat de la carte distribuéepeut accepter un fichier de manifeste d'inventaire Amazon S3 qui est stocké dans un compartiment Amazon S3 sous la forme d'un ensemble de données.

Lorsque l'exécution du flux de travail atteint le`Map`state, Step Functions invoque leGetObjectAction d'API pour récupérer le fichier manifeste d'inventaire Amazon S3 spécifié. Le`Map`state effectue ensuite une itération sur les objets de l'inventaire pour renvoyer un tableau de métadonnées des objets d'inventaire Amazon S3.

> **Note**
>
> - Actuellement, Step Functions prend en charge 10 Go comme taille maximale d'un fichier individuel dans un rapport d'inventaire Amazon S3. Step Functions peut toutefois traiter plus de 10 Go si la taille de chaque fichier est inférieure à 10 Go.
> - Step Functions a besoin des autorisations appropriées pour accéder aux ensembles de données Amazon S3 que vous utilisez. Pour plus d'informations sur les politiques IAM pour les jeux de données, voirPolitiques IAM pour les jeux de données.

Le tableau suivant est un exemple de fichier d'inventaire au format CSV. Ce fich inclut les objets nommés`csvDataset`et`imageDataset`, stockés dans un compartiment Amazon S3 nommé`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**
>
> À l'heure actuelle, Step Functions ne prend pas en charge le rapport d'inventaire Amazon S3 défini par l'utilisateur sous forme de jeu de données. Vous devez également vous assurer que

le format de sortie de votre rapport d'inventaire Amazon S3 est CSV. Pour plus d'informations sur les stocks Amazon S3 et sur la façon de les configurer, consultezInventaire Amazon S3dans leGuide de l'utilisateur Amazon S3.

L'exemple suivant de fichier manifeste d'inventaire montre les en-têtes CSV des métadonnées des objets d'inventaire.

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

Les onglets suivants présentent des exemples de`ItemReader`syntaxe du champ et entrée transmise à l'exécution d'un flux de travail enfant pour cet ensemble de données.

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"
}
```

Selon les champs que vous avez sélectionnés lors de la configuration du rapport d'inventaire Amazon S3, le contenu de votre`manifest.json`le fichier peut être différent de l'exemple illustré.

Politiques IAM pour les jeux de données

Lorsque vous créez des flux de travail avec la console Step Functions, Step Functions peut générer automatiquement des politiques IAM en fonction des ressources figurant dans votre définition de flux de travail. Ces politiques incluent le minimum de privilèges nécessaires pour permettre au rôle de machine à états d'invoquer le`StartExecution`Action d'API pour leÉtat de la carte distribuée. Ces politiques incluent également le minimum de privilèges nécessaires pour accéder à Step Functions.AWSles ressources, telles que les compartiments et les objets Amazon S3 et les fonctions Lambda. Nous vous recommandons vivement de n'inclure que les autorisations nécessaires dans vos politiques IAM. Par exemple, si votre flux de travail inclut un`Map`en mode distribué, étendez vos politiques jusqu'au compartiment et au dossier Amazon S3 spécifiques qui contiennent votre ensemble de données.

> **Important**
>
> Si vous spécifiez un compartiment Amazon S3 et un objet, ou un préfixe, avecchemin de référenceà une paire clé-valeur existante dans votreÉtat de la carte distribuéesaisie, assurez-vous de mettre à jour les politiques IAM pour votre flux de travail. Élargissez les politiques jusqu'au bucket et aux noms d'objets auxquels le chemin aboutit au moment de l'exécution.

Les exemples de politique IAM suivants accordent le minimum de privilèges requis pour accéder à vos ensembles de données Amazon S3 à l'aide duListObjectsV2etGetObjectActions d'API.

Example Politique IAM pour les objets Amazon S3 en tant que jeu de données

L'exemple suivant montre une politique IAM qui accorde le moins de privilèges pour accéder aux objets organisés au sein de*processImages*dans un compartiment Amazon S3 nommé*myBucket*.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::myBucket"
            ],
            "Condition": {
                "StringLike": {
                    "s3:prefix": [
                        "processImages"
                    ]
                }
            }
        }
    ]
}
```

Example Politique IAM pour un fichier CSV en tant que jeu de données

L'exemple suivant montre une politique IAM qui accorde le moins de privilèges pour accéder à un fichier CSV nommé*ratings.csv*.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::myBucket/csvDataset/ratings.csv"
            ]
```

```
        }
    ]
}
```

Example Politique IAM pour un inventaire Amazon S3 sous forme de jeu de données

L'exemple suivant montre une politique IAM qui accorde le moins de privilèges pour accéder à un rapport d'inventaire Amazon S3.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::destination-prefix/source-bucket/config-ID/YYYY-MM-DDTHH-MMZ/manifest.json",
                "arn:aws:s3:::destination-prefix/source-bucket/config-ID/data/*"
            ]
        }
    ]
}
```

ItemsPath

Utilisez le `ItemsPath` champ pour sélectionner un tableau dans une entrée JSON fournie à un `Map` état. L'`Map`état répète un ensemble d'étapes pour chaque élément du tableau. Par défaut, l'`Map`état est défini sur`$`, ce qui permet de sélectionner la totalité de l'entrée. `ItemsPath` Si l'entrée de l'`Map`état est un tableau JSON, il exécute une itération pour chaque élément du tableau, en transmettant cet élément à l'itération en tant qu'entrée.

> **Note**
>
> Vous ne pouvez l'utiliser `ItemsPath` dans l'état Distributed Map que si vous utilisez une entrée JSON transmise à partir d'un état précédent dans le flux de travail.

Vous pouvez utiliser le `ItemsPath` champ pour spécifier un emplacement dans l'entrée qui pointe vers le tableau JSON utilisé pour les itérations. La valeur de `ItemsPath` doit être un chemin de référence, et ce chemin doit pointer vers un tableau JSON. Par exemple, imaginons l'entrée d'un état `Map` qui inclut deux tableaux, comme dans l'exemple suivant.

```
{
  "ThingsPiratesSay": [
    {
      "say": "Avast!"
    },
    {
      "say": "Yar!"
    },
    {
      "say": "Walk the Plank!"
    }
  ],
  "ThingsGiantsSay": [
    {
      "say": "Fee!"
    },
    {
      "say": "Fi!"
    },
    {
      "say": "Fo!"
    },
    {
      "say": "Fum!"
    }
  ]
}
```

Dans ce cas, vous pouvez spécifier le tableau à utiliser pour les itérations `Map` d'état en le sélectionnant avec`ItemsPath`. La définition de machine à états suivante spécifie le `ThingsPiratesSay` tableau dans l'entrée à l'`ItemsPath`aide de .It exécute ensuite une itération de l'état de `SayWord` passe pour chaque élément du `ThingsPiratesSay` tableau.

```
{
  "StartAt": "PiratesSay",
  "States": {
    "PiratesSay": {
```

```
      "Type": "Map",
      "ItemsPath": "$.ThingsPiratesSay",
      "ItemProcessor": {
         "StartAt": "SayWord",
         "States": {
           "SayWord": {
             "Type": "Pass",
             "End": true
           }
         }
      },
      "End": true
    }
  }
}
```

Lors du traitement d'une entrée, l'Mapétat s'applique `ItemsPath` ensuite `InputPath`. Il fonctionne sur l'entrée effective de l'état après avoir `InputPath` filtré l'entrée.

Pour plus d'informations sur les états `Map`, consultez les sections suivantes.

- État de la carte
- Modes de traitement de l'état des cartes
- Répéter une action à l'aide de l'état de la carte intégrée
- Traitement d'entrée et de sortie d'Mapétat en ligne

ItemSelector

Par défaut, l'entrée effective pour l'Mapétat est l'ensemble des éléments de données individuels présents dans l'entrée d'état brut. Le `ItemSelector` champ vous permet de remplacer les valeurs des éléments de données avant qu'elles ne soient transmises à l'Mapétat. Pour remplacer les valeurs, spécifiez une entrée JSON valide contenant une collection de paires clé-valeur. Ces paires peuvent être des valeurs statiques fournies dans la définition de votre machine d'état, des valeurs sélectionnées dans l'entrée d'état à l'aide d'un chemin ou des valeurs accessibles depuis l'objet de contexte.

Si vous spécifiez des paires clé-valeur à l'aide d'un chemin ou d'un objet de contexte, le nom de la clé doit se terminer par. `.$`

> **Note**
>
> Le `ItemSelector` champ remplace le `Parameters` champ au sein de l'`Map`état. Si vous utilisez le `Parameters` champ dans vos définitions d'`Map`état pour créer une entrée personnalisée, nous vous recommandons vivement de le remplacer par`ItemSelector`.

Vous pouvez spécifier le `ItemSelector` champ à la fois dans un état de carte en ligne et dans un état de carte distribuée.

Par exemple, considérez l'entrée JSON suivante qui contient un tableau de trois éléments au sein du `imageData` nœud. Pour chaque itération d'*Map*état, un élément de tableau est transmis à l'itération en tant qu'entrée.

```
[
  {
    "resize": "true",
    "format": "jpg"
  },
  {
    "resize": "false",
    "format": "png"
  },
  {
    "resize": "true",
    "format": "jpg"
  }
]
```

À l'aide du `ItemSelector` champ, vous pouvez définir une entrée JSON personnalisée pour remplacer l'entrée d'origine, comme indiqué dans l'exemple suivant. Step Functions transmet ensuite cette entrée personnalisée à chaque itération *Map* d'état. L'entrée personnalisée contient une valeur statique pour `size` et la valeur d'un objet de contexte, des données pour `Map` l'état. L'objet de `$$.Map.Item.Value` contexte contient la valeur de chaque élément de données individuel.

```
{
  "ItemSelector": {
    "size": 10,
    "value.$": "$$.Map.Item.Value"
  }
```

```
}
```

L'exemple suivant montre l'entrée reçue par une itération de l'état de la carte en ligne :

```
{
  "size": 10,
  "value": {
    "resize": "true",
    "format": "jpg"
  }
}
```

> **Tip**
>
> Pour un exemple complet d'état de carte distribuée utilisant le `ItemSelector` champ, reportez-vous à la sectionCommencer à utiliser Distributed Map State.

ItemBatcher

Le `ItemBatcher` champ est un objet JSON qui indique de traiter un groupe d'éléments lors d'une seule exécution de flux de travail enfant. Utilisez le traitement par lots lors du traitement de gros fichiers CSV ou de tableaux JSON, ou de grands ensembles d'objets Amazon S3.

L'exemple suivant montre la syntaxe du `ItemBatcher` champ. Dans la syntaxe suivante, le nombre maximum d'éléments que chaque exécution de flux de travail enfant doit traiter est défini sur 100.

```
{
  "ItemBatcher": {
    "MaxItemsPerBatch": 100
  }
}
```

Par défaut, chaque élément d'un ensemble de données est transmis en tant qu'entrée aux exécutions individuelles du flux de travail pour enfants. Supposons, par exemple, que vous indiquiez un fichier JSON en entrée contenant le tableau suivant :

```
[
  {
    "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"
  },
  ...
]
```

Pour l'entrée donnée, chaque exécution de flux de travail enfant reçoit un élément de tableau en tant qu'entrée. L'exemple suivant montre la saisie d'une exécution de flux de travail enfant :

```
{
  "verdict": "true",
  "statement_date": "6/11/2008",
  "statement_source": "speech"
}
```

Pour optimiser les performances et le coût de votre travail de traitement, sélectionnez une taille de lot qui équilibre le nombre d'articles par rapport au temps de traitement des articles. Si vous utilisez le traitement par lots, Step Functions ajoute les éléments à un tableau d'éléments. Il transmet ensuite le tableau en tant qu'entrée à chaque exécution de flux de travail enfant. L'exemple suivant montre un lot de deux éléments transmis en entrée à l'exécution d'un flux de travail enfant :

```
{
  "Items": [
    {
      "verdict": "true",
      "statement_date": "6/11/2008",
      "statement_source": "speech"
    },
    {
      "verdict": "false",
      "statement_date": "6/7/2022",
      "statement_source": "television"
```

```
    }
  ]
}
```

> **Tip**
>
> Pour en savoir plus sur l'utilisation du `ItemBatcher` champ dans vos flux de travail, essayez les didacticiels et l'atelier suivants :
>
> - Traiter un lot complet de données dans une fonction Lambda
> - Effectuez une itération sur les éléments d'un lot lors d'exécutions de flux de travail pour enfants
> - Parallélisation à grande échelle avec carte distribuée dans le module 14 - Traitement des données de l'atelier AWS Step Functions

Table des matières

Champs permettant de spécifier le regroupement des articles

Pour regrouper des articles, spécifiez le nombre maximum d'articles à regrouper, la taille maximale du lot, ou les deux. Vous devez spécifier l'une de ces valeurs pour regrouper les articles.

Nombre maximum d'articles par lot

Spécifie le nombre maximum d'éléments que chaque exécution de flux de travail enfant traite. L'interpréteur limite le nombre d'éléments groupés dans le `Items` tableau à cette valeur. Si vous spécifiez à la fois un numéro de lot et une taille, l'interpréteur réduit le nombre d'éléments d'un lot pour éviter de dépasser la limite de taille de lot spécifiée.

Si vous ne spécifiez pas cette valeur mais que vous fournissez une valeur pour la taille de lot maximale, Step Functions traite autant d'éléments que possible lors de chaque exécution de flux de travail enfant sans dépasser la taille de lot maximale en octets.

Par exemple, imaginez que vous exécutez une exécution avec un fichier JSON d'entrée contenant 1130 nœuds. Si vous spécifiez une valeur maximale d'éléments pour chaque lot de 100, Step Functions crée 12 lots. Parmi ceux-ci, 11 lots contiennent 100 articles chacun, tandis que le douzième lot contient les 30 articles restants.

Vous pouvez également spécifier le nombre maximum d'éléments pour chaque lot en tant que chemin de référence vers une paire clé-valeur existante dans votre entrée d'état de carte distribuée. Ce chemin doit être converti en un entier positif.

Par exemple, avec les données d'entrée suivantes :

```
{
  "maxBatchItems": 500
}
```

Vous pouvez spécifier le nombre maximum d'articles à regrouper à l'aide d'un chemin de référence comme suit :

```
{
  ...
  "Map": {
    "Type": "Map",
    "MaxConcurrency": 2000,
    "ItemBatcher": {
      "MaxItemsPerBatchPath": "$.maxBatchItems"
    }
    ...
    ...
  }
}
```

> **Important**
>
> Vous pouvez spécifier le champ `MaxItemsPerBatch` ou le `MaxItemsPerBatchPath` sous-champ, mais pas les deux.

Nombre maximal de Ko par lot

Spécifie la taille maximale d'un lot en octets, jusqu'à 256 Ko. Si vous spécifiez à la fois un numéro et une taille de lot maximaux, Step Functions réduit le nombre d'articles d'un lot afin d'éviter de dépasser la limite de taille de lot spécifiée.

Vous pouvez également spécifier la taille maximale du lot comme chemin de référence vers une paire clé-valeur existante dans votre entrée d'état de carte distribuée. Ce chemin doit être converti en un entier positif.

> **Note**
>
> Si vous utilisez le traitement par lots sans spécifier de taille de lot maximale, l'interpréteur traite autant d'éléments qu'il peut traiter jusqu'à 256 Ko lors de chaque exécution d'un flux de travail enfant.

Par exemple, avec les données d'entrée suivantes :

```
{
  "batchSize": 131072
}
```

Vous pouvez spécifier la taille maximale du lot à l'aide d'un chemin de référence comme suit :

```
{
  ...
  "Map": {
    "Type": "Map",
    "MaxConcurrency": 2000,
    "ItemBatcher": {
      "MaxInputBytesPerBatchPath": "$.batchSize"
    }
    ...
    ...
  }
}
```

> **Important**
>
> Vous pouvez spécifier le champ `MaxInputBytesPerBatch` ou le `MaxInputBytesPerBatchPath` sous-champ, mais pas les deux.

Entrée par lots

Vous pouvez également spécifier une entrée JSON fixe à inclure dans chaque lot transmis à chaque exécution de flux de travail enfant. Step Functions fusionne cette entrée avec celle de chaque exécution de flux de travail enfant individuelle. Par exemple, étant donné la saisie fixe suivante d'une date de vérification des faits sur un tableau d'éléments :

```
"ItemBatcher": {
    "BatchInput": {
        "factCheck": "December 2022"
    }
}
```

Chaque exécution d'un flux de travail enfant reçoit les informations suivantes en entrée :

```
{
  "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

Le `ResultWriter` champ est un objet JSON qui indique l'emplacement Amazon S3 où Step Functions écrit les résultats des exécutions du flux de travail enfant lancées par un état de carte distribuée. Par défaut, Step Functions n'exporte pas ces résultats.

> **Important**
>
> Assurez-vous que le compartiment Amazon S3 que vous utilisez pour exporter les résultats d'un Map Run se trouve sous le même Compte AWS emplacement Région AWS que votre machine d'état. Sinon, l'exécution de votre machine d'état échouera avec l'`States.ResultWriterFailed`erreur.

L'exportation des résultats vers un compartiment Amazon S3 est utile si la taille de votre charge utile de sortie dépasse 256 Ko. Step Functions consolide toutes les données d'exécution du flux de travail enfant, telles que les entrées et sorties d'exécution, l'ARN et le statut d'exécution. Il exporte ensuite les exécutions avec le même statut vers leurs fichiers respectifs à l'emplacement Amazon S3 spécifié. L'exemple suivant montre la syntaxe du `ResultWriter` champ si vous exportez les résultats de l'exécution du flux de travail enfant. Dans cet exemple, vous stockez les résultats dans un compartiment nommé `myOutputBucket` dans un préfixe appelé`csvProcessJobs`.

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

> **Tip**
>
> Dans Workflow Studio, vous pouvez exporter les résultats d'exécution du flux de travail enfant en sélectionnant Exporter les résultats de l'état de la carte vers Amazon S3. Indiquez ensuite le nom du compartiment Amazon S3 et le préfixe vers lesquels vous souhaitez exporter les résultats.

Step Functions a besoin des autorisations appropriées pour accéder au bucket et au dossier dans lesquels vous souhaitez exporter les résultats. Pour plus d'informations sur la politique IAM requise, consultezPolitiques IAM pour ResultWriter.

Si vous exportez les résultats de l'exécution du flux de travail enfant, l'exécution de l'état de la carte distribuée renvoie l'ARN Map Run et les données relatives au lieu d'exportation Amazon S3 au format suivant :

```
{
  "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 exporte les exécutions avec le même statut vers leurs fichiers respectifs. Par exemple, si les exécutions du flux de travail de votre enfant se sont soldées par 500 résultats réussis et 200 échecs, Step Functions crée deux fichiers à l'emplacement Amazon S3 spécifié pour les résultats de réussite et d'échec. Dans cet exemple, le fichier de résultats de réussite contient les 500 résultats de réussite, tandis que le fichier de résultats d'échec contient les 200 résultats d'échec.

Pour une tentative d'exécution donnée, Step Functions crée les fichiers suivants dans l'emplacement Amazon S3 spécifié en fonction du résultat de votre exécution :

- `manifest.json`— Contient les métadonnées Map Run, telles que l'emplacement d'exportation, l'ARN Map Run et des informations sur les fichiers de résultats.

  Si vous avez redrivenun Map Run, le `manifest.json` fichier contient des références à toutes les exécutions de flux de travail enfant réussies lors de toutes les tentatives d'exécution d'un Map Run. Toutefois, ce fichier contient des références aux exécutions échouées ou en attente pour une exécution spécifiqueredrive.
- `SUCCEEDED_n.json`— Contient les données consolidées pour toutes les exécutions réussies de flux de travail pour enfants. n représente le numéro d'index du fichier. Le numéro d'index commence à 0. Par exemple, `SUCCEEDED_1.json`.
- `FAILED_n.json`— Contient les données consolidées pour toutes les exécutions de flux de travail enfants ayant échoué, expiré ou abandonné. Utilisez ce fichier pour effectuer une restauration après un échec d'exécution. n représente l'index du fichier. Le numéro d'index commence à 0. Par exemple, `FAILED_1.json`.
- `PENDING_n.json`— Contient les données consolidées pour toutes les exécutions de flux de travail enfants qui n'ont pas été démarrées en raison de l'échec ou de l'abandon du Map Run. n représente l'index du fichier. Le numéro d'index commence à 0. Par exemple, `PENDING_1.json`.

Step Functions prend en charge des fichiers de résultats individuels d'une capacité maximale de 5 Go. Si la taille d'un fichier dépasse 5 Go, Step Functions crée un autre fichier pour écrire les résultats d'exécution restants et ajoute un numéro d'index au nom du fichier. Par exemple, si la taille du `Succeeded_0.json` fichier dépasse 5 Go, Step Functions crée un `Succeeded_1.json` fichier pour enregistrer les résultats restants.

Si vous n'avez pas spécifié d'exporter les résultats de l'exécution du flux de travail enfant, l'exécution de la machine d'état renvoie un tableau des résultats d'exécution du flux de travail enfant, comme indiqué dans l'exemple suivant :

> **Note**
>
> Si la taille de sortie renvoyée dépasse 256 Ko, l'exécution de la machine d'état échoue et renvoie une `States.DataLimitExceeded` erreur.

```
[
  {
    "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"
    }
  },
  ...
]
```

Politiques IAM pour ResultWriter

Lorsque vous créez des flux de travail avec la console Step Functions, Step Functions peut générer automatiquement des politiques IAM en fonction des ressources figurant dans votre définition de flux de travail. Ces politiques incluent le minimum de privilèges nécessaires pour permettre au rôle de machine d'état d'invoquer l'action d'`StartExecution`API pour l'état de la carte distribuée. Ces politiques incluent également le minimum de privilèges nécessaires aux Step Functions pour accéder aux AWS ressources, telles que les buckets et les objets Amazon S3 et les fonctions Lambda. Nous vous recommandons vivement de n'inclure que les autorisations nécessaires dans vos politiques

IAM. Par exemple, si votre flux de travail inclut un `Map` état en mode distribué, limitez vos politiques au compartiment et au dossier Amazon S3 spécifiques qui contiennent votre ensemble de données.

> **Important**
>
> Si vous spécifiez un compartiment et un objet Amazon S3, ou un préfixe, avec un chemin de référence vers une paire clé-valeur existante dans l'entrée d'état de votre carte distribuée, assurez-vous de mettre à jour les politiques IAM pour votre flux de travail. Élargissez les politiques jusqu'au bucket et aux noms d'objets auxquels le chemin aboutit au moment de l'exécution.

L'exemple de politique IAM suivant accorde le minimum de privilèges requis pour écrire les résultats de l'exécution du flux de travail de votre enfant dans un dossier nommé *CSVjobs* dans un compartiment Amazon S3 à l'aide de l'`PutObject`action API.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListMultipartUploadParts",
                "s3:AbortMultipartUpload"
            ],
            "Resource": [
                "arn:aws:s3:::resultBucket/csvJobs/*"
            ]
        }
    ]
}
```

Si le compartiment Amazon S3 dans lequel vous écrivez le résultat de l'exécution du flux de travail enfant est chiffré à l'aide d'une AWS Key Management Service (AWS KMS) clé, vous devez inclure les AWS KMS autorisations nécessaires dans votre politique IAM. Pour plus d'informations, consultez Autorisations IAM pour le compartiment Amazon S3 AWS KMS key chiffré.

Analyse d'un fichier CSV d'entrée

Comme il n'existe pas de format standardisé pour créer et gérer les données dans les fichiers CSV, Step Functions analyse les fichiers CSV selon les règles suivantes :

- Les virgules (,) sont des séparateurs qui séparent les champs individuels.
- Les nouvelles lignes sont un séparateur qui sépare les enregistrements individuels.
- Les champs sont traités comme des chaînes. Pour les conversions de types de données, utilisez le`States.StringToJson`fonction intrinsèque dansItemSelector.
- Les guillemets (» «) ne sont pas obligatoires pour entourer les chaînes. Toutefois, les chaînes placées entre guillemets doubles peuvent contenir des virgules et des nouvelles lignes sans qu'elles servent de délimiteurs.
- Évitez les guillemets doubles en les répétant.
- Si le nombre de champs d'une ligne est inférieur au nombre de champs de l'en-tête, Step Functions fournit des chaînes vides pour les valeurs manquantes.
- Si le nombre de champs d'une ligne est supérieur au nombre de champs de l'en-tête, Step Functions ignore les champs supplémentaires.

Example d'analyse d'un fichier CSV d'entrée

Supposons que vous ayez fourni un fichier CSV nommé*myCSVInput.csv*qui contient une ligne en entrée. Vous avez ensuite stocké ce fichier dans un compartiment Amazon S3 nommé*my-bucket*. Le fichier CSV est le suivant.

```
abc,123,"This string contains commas, a double quotation marks (""), and a newline (
)",{""MyKey"":""MyValue""},"[1,2,3]"
```

La machine à états suivante lit ce fichier CSV et utiliseItemSelectorpour convertir les types de données de certains champs.

```
{
  "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)"
      }
    }
  }
}
```

Lorsque vous exécutez cette machine d'état, elle renvoie le résultat suivant :

```
[
  {
    "MyNumbers": 123,
    "MyObject": {
      "MyKey": "MyValue"
    },
    "MyString": "This string contains commas, a double quote (\"), and a newline (\n)",
    "MyLetters": "abc",
    "MyArray": [
      1,
      2,
      3
    ]
  }
]
```

Objet Contexte

L'objet de contexte est une structure JSON interne disponible lors d'une exécution et contenant des informations sur votre machine à états et sur l'exécution. Cela permet à vos flux de travail d'accéder à des informations sur leur exécution spécifique. Vous pouvez accéder à l'objet de contexte à partir des champs suivants :

- `InputPath`
- `OutputPath`
- `ItemsPath`(dans les états de la carte)
- `Variable`(dans les États de Choice)
- `ResultSelector`
- `Parameters`
- Opérateurs de comparaison de variables à variables

Format d'objet de contexte

L'objet de contexte inclut des informations sur la machine d'état, l'état, l'exécution et la tâche. Cet objet JSON inclut des nœuds pour chaque type de données, et se trouve dans le format suivant.

```
{
    "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"
    }
}
```

Lors d'une exécution, l'objet de contexte est rempli avec des données pertinentes pour le champ `Parameters` à partir duquel il est accessible. La valeur d'un champ `Task` est null si le champ `Parameters` est en dehors d'un état de tâche.

La valeur de l'objet de `RedriveCount` contexte est 0, si vous n'avez pas encore effectué redrivend'exécution. De plus, l'objet de `RedriveTime` contexte n'est disponible que si vous avez redriven une exécution. Si c'est le cas redriven a Map Run, l'objet de `RedriveTime` contexte n'est disponible que pour les flux de travail enfants de type Standard. Pour un redriven Map Run avec des flux de travail enfants de type Express, `RedriveTime` ce n'est pas disponible.

Le contenu d'une exécution en cours inclut des détails dans le format suivant.

```
{
    "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**
>
> Pour les données d'objet de contexte liées aux états `Map`, consultez Données d'objet de contexte pour les états Map.

Accès à l'objet de contexte

Pour accéder à l'objet de contexte, vous devez d'abord spécifier le nom de paramètre en ajoutant `.$` à la fin, comme vous le faites lorsque vous sélectionnez des entrées d'état avec un chemin. Ensuite, pour accéder aux données de l'objet de contexte au lieu de l'entrée, préfixez le chemin d'accès par `$$.`. Cela indique AWS Step Functions d'utiliser le chemin pour sélectionner un nœud dans l'objet de contexte.

Les exemples suivants montrent comment accéder aux objets contextuels, tels que l'ID d'exécution, le nom, l'heure de début et le redrive nombre.

- Récupérez et transmettez l'ARN d'exécution à un service en aval
- Accédez à l'heure de début et au nom de l'exécution dans un état Pass
- Accédez au redrive décompte d'une exécution

Récupérez et transmettez l'ARN d'exécution à un service en aval

Cet exemple d'état de tâche utilise un chemin pour récupérer et transmettre le nom de ressource Amazon (ARN) d'exécution à un message Amazon Simple Queue Service (Amazon SQS).

```
{
  "Order Flight Ticket Queue": {
    "Type": "Task",
    "Resource": "arn:aws:states:::sqs:sendMessage",
    "Parameters": {
      "QueueUrl": "https://sqs.us-east-1.amazonaws.com/123456789012/flight-purchase",
      "MessageBody": {
        "From": "YVR",
        "To": "SEA",
        "Execution.$": "$$.Execution.Id"
      }
    },
    "Next": "NEXT_STATE"
  }
}
```

Pour plus d'informations sur l'utilisation du jeton de tâche lors de l'appel d'un service intégré, consultez Attendre un rappel avec le jeton de tâche.

Accédez à l'heure de début et au nom de l'exécution dans un état Pass

```
{
  "Comment": "Accessing context object in a state machine",
  "StartAt": "Get execution context data",
  "States": {
    "Get execution context data": {
      "Type": "Pass",
      "Parameters": {
        "startTime.$": "$$.Execution.StartTime",
        "execName.$": "$$.Execution.Name"
      },
      "ResultPath": "$.executionContext",
      "End": true
    }
  }
}
```

Accédez au redrive décompte d'une exécution

L'exemple suivant de définition d'état de tâche montre comment accéder au redrivenombre d'exécutions.

```
{
  "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
}
```

Données d'objet de contexte pour les états Map

Deux éléments supplémentaires sont disponibles dans l'objet de contexte lors du traitement d'un état Map : `Index` et `Value`. Pour chaque itération d'`Map`état, `Index` contient le numéro d'index de l'élément du tableau en cours de traitement, tandis que `Value` contient l'élément du tableau en cours de traitement. Dans un `Map` état, l'objet de contexte inclut les données suivantes :

```
"Map": {
   "Item": {
      "Index": Number,
      "Value": "String"
   }
}
```

Ils ne sont disponibles que dans un `Map` état et peuvent être spécifiés dans le `ItemSelector` champ.

> **Note**
>
> Vous devez définir des paramètres à partir de l'objet de contexte dans le bloc `ItemSelector` de l'état `Map` principal, et non dans les états inclus dans la section `ItemProcessor`.

Soit une machine d'état avec un état `Map` simple, nous pouvons injecter des informations à partir de l'objet de contexte comme suit.

```
{
  "StartAt": "ExampleMapState",
  "States": {
    "ExampleMapState": {
      "Type": "Map",
      "ItemSelector": {
        "ContextIndex.$": "$$.Map.Item.Index",
        "ContextValue.$": "$$.Map.Item.Value"
      },
      "ItemProcessor": {
        "ProcessorConfig": {
          "Mode": "INLINE"
        },
        "StartAt": "TestPass",
        "States": {
          "TestPass": {
            "Type": "Pass",
            "End": true
          }
        }
      },
      "End": true
    }
  }
}
```

Si vous exécutez la machine d'état précédente avec l'entrée suivante, les éléments `Index` et `Value` sont insérés dans la sortie.

```
[
  {
    "who": "bob"
  },
  {
    "who": "meg"
  },
  {
    "who": "joe"
  }
```

```
]
```

Le résultat de l'exécution renvoie les valeurs de `Index` et `Value` les éléments pour chacune des trois itérations, comme suit :

```
[
  {
    "ContextIndex": 0,
    "ContextValue": {
      "who": "bob"
    }
  },
  {
    "ContextIndex": 1,
    "ContextValue": {
      "who": "meg"
    }
  },
  {

    "ContextIndex": 2,
    "ContextValue": {
      "who": "joe"
    }
  }
]
```

Simulateur de flux de données

Vous pouvez concevoir, implémenter et déboguer des flux de travail dans la console Step Functions. Vous pouvez également contrôler le flux de données dans vos flux de travail grâce au traitement des données JsonPathd'entrée et de sortie. Le simulateur de flux de données vous permet de simuler l'ordre dans lequel les État de la tâche états de votre flux de travail traitent les données lors de l'exécution. À l'aide du simulateur, vous pouvez comprendre comment filtrer et manipuler les données lorsqu'elles passent d'un état à un autre. Il simule chacun des champs suivants que Step Functions utilise pour traiter et contrôler le flux de données JSON :

`InputPath`

Sélectionne la partie de l'ensemble de la charge utile d'entrée à utiliser comme entrée d'une tâche. Si vous spécifiez ce champ, Step Functions applique d'abord ce champ.

`Paramètres`

Spécifie à quoi doit ressembler l'entrée avant d'appeler la tâche. Ce `Parameters` champ vous permet de créer une collection de paires clé-valeur qui sont transmises en entrée à une Service AWSintégration, telle qu'une AWS Lambda fonction. Ces valeurs peuvent être statiques ou sélectionnées dynamiquement à partir de l'entrée d'état ou de l'objet de contexte du flux de travail.

`ResultSelector`

Détermine ce qu'il faut choisir parmi les résultats d'une tâche. Ce `ResultSelector` champ vous permet de créer une collection de paires clé-valeur qui remplacent le résultat d'un état et de transmettre cette collection à. `ResultPath`

`ResultPath`

Détermine où placer la sortie d'une tâche. Utilisez le `ResultPath` pour déterminer si la sortie d'un état est une copie de son entrée, du résultat qu'il produit ou une combinaison des deux.

`OutputPath`

Détermine CE qu'il faut envoyer à l'état suivant. Avec`OutputPath`, vous pouvez filtrer les informations indésirables et ne transmettre que la partie des données JSON qui vous intéresse.

Dans cette rubrique

Utilisation du simulateur de flux de données

Le simulateur fournit une side-by-side comparaison en temps réel de vos données avant et après l'application d'un champ de traitement des données d'entrée et de sortie. Pour utiliser le simulateur, spécifiez une entrée JSON. Ensuite, évaluez-le à travers chacun des champs de traitement d'entrée et de sortie. Le simulateur valide automatiquement votre entrée JSON et met en évidence les éventuelles erreurs de syntaxe.

Pour utiliser le simulateur de flux de données

Dans les étapes suivantes, vous devez fournir une entrée JSON et appliquer les `Paramètres` champs `InputPath` et. Vous pouvez également appliquer les autres champs disponibles et afficher leurs résultats.

1. Ouvrez la console Step Functions.
2. Dans le volet de navigation, sélectionnez Simulateur de flux de données.
3. Dans la zone de saisie State, remplacez les exemples de données JSON préremplis par les données JSON suivantes. Ensuite, choisissez Next (Suivant).

```
{
  "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. Pour InputPath, entrez **`$.data.address`** pour sélectionner le nœud d'adresse des données JSON d'entrée.

   La InputPath zone de saisie de l'état après affiche les résultats suivants.

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

5. Choisissez Suivant.
6. Appliquez le `Parameters` champ pour convertir les données JSON obtenues en chaîne. Pour convertir les données, procédez comme suit :

   - Dans la zone Paramètres, entrez le code suivant pour créer une chaîne appeléeaddressString.

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

7. Affichez le résultat de l'application `Parameters` sur le terrain dans la zone Entrée filtrée après paramètres.

Considérations relatives à l'utilisation du simulateur de flux de données

Avant d'utiliser le simulateur de flux de données, tenez compte de ses limites, notamment :

- Expressions de filtre non prises en charge

  Les expressions de filtre du simulateur se comportent différemment de celles du service Step Functions. Cela inclut les expressions qui utilisent la syntaxe suivante :`[?(expression)]`. La liste suivante répertorie les expressions non prises en charge. Si elles sont utilisées, ces expressions peuvent ne pas renvoyer le résultat attendu après leur évaluation.

  - `$..book[?(@.isInStock==true)]`
  - `$..book[?(@.price > 10.0)].title`
- JsonPathÉvaluation incorrecte pour les tableaux à éléments uniques

  Si vous filtrez vos données à l'aide JsonPath d'une expression renvoyant un seul élément du tableau, le simulateur renvoie l'élément sans le tableau. Par exemple, considérez le tableau de données suivant, appelé `items` :

```
{
  "items": [
    {
      "name": "shoe",
      "color": "blue",
      "comment": "nice shoe"
    },
    {
      "name": "hat",
      "color": "red"
    },
    {
```

```
      "name": "shirt",
      "color": "yellow"
    }
  ]
}
```

Compte tenu de ce `items` tableau, si vous le saisissez **`$..comment`** dans le InputPath champ, vous vous attendrez à obtenir le résultat suivant :

```
[
  "nice shoe"
]
```

Toutefois, le simulateur de flux de données renvoie plutôt la sortie suivante :

```
"nice shoe"
```

Pour JsonPath l'évaluation d'un tableau contenant plusieurs éléments, le simulateur renvoie la sortie attendue.

Gérez les déploiements continus avec des versions et des alias

Vous pouvez utiliser Step Functions pour gérer les déploiements continus de vos flux de travail via les versions et les alias des machines d'état. Une version est un instantané numéroté et immuable d'une machine d'état que vous pouvez exécuter. Un alias est un pointeur vers deux versions au maximum d'une machine à états.

Vous pouvez gérer plusieurs versions de vos machines d'état et gérer leur déploiement dans votre flux de travail de production. Les alias vous permettent d'acheminer le trafic entre différentes versions de flux de travail et de déployer progressivement ces flux de travail dans l'environnement de production.

En outre, vous pouvez démarrer les exécutions des machines d'état à l'aide d'une version ou d'un alias. Si vous n'utilisez pas de version ou d'alias lorsque vous démarrez l'exécution d'une machine d'état, Step Functions utilise la dernière révision de la définition de la machine d'état.

> **Révision de la machine d'État**
>
> Une machine d'état peut avoir une ou plusieurs révisions. Lorsque vous mettez à jour une machine d'état à l'aide de l'action d'UpdateStateMachineAPI, elle crée une nouvelle révision de machine d'état. Une révision est un instantané immuable en lecture seule de la définition et de la configuration d'une machine d'état. Vous ne pouvez pas démarrer l'exécution d'une machine d'état à partir d'une révision, et les révisions ne possèdent pas d'ARN. Les révisions ont un`revisionId`, qui est un identifiant unique universel (UUID).

Table des matières

Versions des machines à états

Une version est un instantané numéroté et immuable d'une machine à états. Vous publiez les versions de la dernière révision apportée à cette machine d'état. Chaque version possède un Amazon Resource Name (ARN) unique. Cet ARN est une combinaison de l'ARN de la machine à états et du numéro de version séparés par deux points (:). L'exemple suivant montre le format d'un ARN de version State Machine.

```
arn:partition:states:region:account-id:stateMachine:myStateMachine:1
```

Pour commencer à utiliser les versions State Machine, vous devez publier la première version. Après avoir publié une version, vous pouvez appeler l'action d'StartExecutionAPI avec l'ARN de la version. Vous ne pouvez pas modifier une version, mais vous pouvez mettre à jour une machine à états et publier une nouvelle version. Vous pouvez également publier plusieurs versions de votre machine d'état.

Lorsque vous publiez une nouvelle version de votre machine à états, Step Functions lui attribue un numéro de version. Les numéros de version commencent à 1 et augmentent de façon monotone pour chaque nouvelle version. Les numéros de version ne sont pas réutilisés pour une machine à états donnée. Si vous supprimez la version 10 de votre machine d'état puis publiez une nouvelle version, Step Functions la publie en tant que version 11.

Les propriétés suivantes sont les mêmes pour toutes les versions d'une machine à états :

- Toutes les versions d'une machine à états partagent le même type (Standard ou Express).
- Vous ne pouvez pas modifier le nom ou la date de création d'une machine à états entre les versions.
- Les balises s'appliquent globalement aux machines d'État. Vous pouvez gérer les balises pour les machines à états à l'aide des actions TagResourceet de UntagResourcel'API.

Les machines à états contiennent également des propriétés qui font partie de chaque versionrevision, mais ces propriétés peuvent différer entre deux versions ou révisions données. Ces propriétés incluent la définition de la machine State, le rôle IAM, la configuration du suivi et la configuration de la journalisation.

Table des matières

Publication d'une version de machine à états (console)

Vous pouvez publier jusqu'à 1 000 versions d'une machine à états. Pour demander une augmentation de cette limite souple, utilisez la page Support Center du AWS Management Console. Vous

pouvez supprimer manuellement les versions inutilisées depuis la console ou en appelant l'action DeleteStateMachineVersionAPI.

Pour publier une version d'une machine à états

1. Ouvrez la console Step Functions, puis choisissez une machine à états existante.
2. Sur la page détaillée de State machine, choisissez Modifier.
3. Modifiez la définition de la machine à états selon vos besoins, puis choisissez Enregistrer.
4. Choisissez Publier la version.
5. (Facultatif) Dans le champ Description de la boîte de dialogue qui apparaît, entrez une brève description de la version de la machine à états.
6. Choisissez Publish.

> **Note**
>
> Lorsque vous publiez une nouvelle version de votre machine à états, Step Functions lui attribue un numéro de version. Les numéros de version commencent à 1 et augmentent de façon monotone pour chaque nouvelle version. Les numéros de version ne sont pas réutilisés pour une machine à états donnée. Si vous supprimez la version 10 de votre machine d'état puis publiez une nouvelle version, Step Functions la publie en tant que version 11.

Gestion des versions avec les opérations de l'API Step Functions

Step Functions fournit les opérations d'API suivantes pour publier et gérer les versions de State Machine :

- PublishStateMachineVersion— Publie une version à partir de la version actuelle revision d'une machine à états.
- UpdateStateMachine— Publie une nouvelle version de machine à états si vous mettez à jour une machine à états et définissez le `publish` paramètre sur `true` dans la même demande.
- CreateStateMachine— Publie la première révision de la machine à états si vous définissez le `publish` paramètre sur`true`.
- ListStateMachineVersions— Répertorie les versions de l'ARN de la machine à états spécifié.
- DescribeStateMachine— Renvoie les détails de la version de la machine à états pour un ARN de version spécifié dans`stateMachineArn`.

- DeleteStateMachineVersion— Supprime une version de machine à états.

Pour publier une nouvelle version à partir de la révision en cours d'une machine à états appelée à l'*myStateMachine*aide du AWS Command Line Interface, utilisez la `publish-state-machine-version` commande :

```
aws stepfunctions publish-state-machine-version --state-machine-arn arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine
```

La réponse renvoie le `stateMachineVersionArn`. Par exemple, la commande précédente renvoie une réponse de`arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine:1`.

> **Note**
>
> Lorsque vous publiez une nouvelle version de votre machine à états, Step Functions lui attribue un numéro de version. Les numéros de version commencent à 1 et augmentent de façon monotone pour chaque nouvelle version. Les numéros de version ne sont pas réutilisés pour une machine à états donnée. Si vous supprimez la version 10 de votre machine d'état puis publiez une nouvelle version, Step Functions la publie en tant que version 11.

Exécution d'une version de machine à états depuis la console

Pour commencer à utiliser les versions de la machine à états, vous devez d'abord publier une version à partir de la machine à états actuellerevision. Pour publier une version, utilisez la console Step Functions ou appelez l'action PublishStateMachineVersionAPI. Vous pouvez également appeler l'action d'UpdateStateMachineAliasAPI avec un paramètre facultatif nommé `publish` pour mettre à jour une machine à états et publier sa version.

Vous pouvez démarrer l'exécution d'une version à l'aide de la console ou en appelant l'action d'StartExecutionAPI et en fournissant l'ARN de la version. Vous pouvez également utiliser un alias pour démarrer l'exécution d'une version. En fonction de sa configuration de routage, un alias achemine le trafic vers une version spécifique.

Si vous lancez une exécution par machine à états sans utiliser de version, Step Functions utilise la version la plus récente de la machine à états pour l'exécution. Pour plus d'informations sur la façon

dont Step Functions associe une exécution à une version, consultezAssocier des exécutions à une version ou à un alias.

Pour démarrer une exécution à l'aide d'une version de machine à états

1. Ouvrez la console Step Functions, puis choisissez une machine à états existante pour laquelle vous avez publié une ou plusieurs versions. Pour savoir comment publier une version, voirPublication d'une version de machine à états (console).
2. Sur la page détaillée de State machine, choisissez l'onglet Versions.
3. Dans la section Versions, procédez comme suit :
   a. Sélectionnez la version avec laquelle vous souhaitez démarrer l'exécution.
   b. Choisissez Start execution (Démarrer l'exécution).
4. (Facultatif) Dans la boîte de dialogue Démarrer l'exécution, entrez le nom de l'exécution.
5. (Facultatif), entrez l'entrée d'exécution, puis choisissez Démarrer l'exécution.

Alias de machine à états

Un alias est un pointeur pour un maximum de deux versions d'une même machine à états. Vous pouvez créer plusieurs alias pour vos machines d'état. Chaque alias possède un Amazon Resource Name (ARN) unique. L'ARN de l'alias est une combinaison de l'ARN de la machine à états et du nom de l'alias, séparés par deux points (:). L'exemple suivant montre le format d'un alias ARN de machine à états.

```
arn:partition:states:region:account-id:stateMachine:myStateMachine:aliasName
```

Vous pouvez utiliser un alias pour acheminer le trafic entre l'une des deux versions de machine à états. Vous pouvez également créer un alias pointant vers une version unique. Les alias ne peuvent pointer que vers les versions des machines à états. Vous ne pouvez pas utiliser un alias pour pointer vers un autre alias. Vous pouvez également mettre à jour un alias pour qu'il pointe vers une autre version de la machine à états.

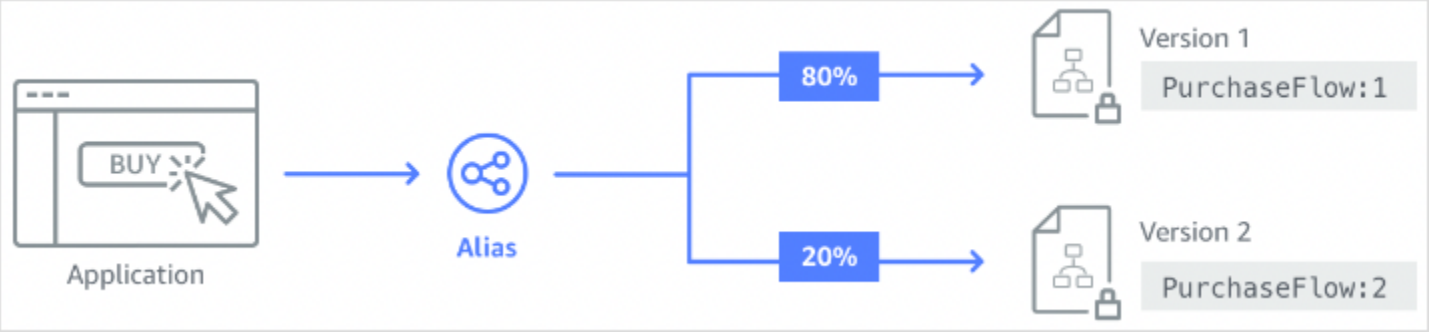


Table des matières

Création d'un alias de machine à états (console)

Vous pouvez créer jusqu'à 100 alias pour chaque machine à états à l'aide de la console Step Functions ou en appelant l'action CreateStateMachineAliasAPI. Pour demander une augmentation de cette limite souple, utilisez la page Support Center du AWS Management Console. Supprimez les alias non utilisés de la console ou en invoquant l'action DeleteStateMachineAliasAPI.

Pour créer un alias de machine à états

1. Ouvrez la console Step Functions, puis choisissez une machine à états existante.
2. Sur la page détaillée de State machine, choisissez l'onglet Alias.
3. Choisissez Créer un nouvel alias.
4. Sur la page Create alias (Créer un alias), procédez de la manière suivante :
    a. Entrez un nom d'alias.
    b. (Facultatif) Renseignez le champ Description de l'alias.
5. Pour configurer le routage sur l'alias, consultez la section Configuration du routage de l'alias.
6. Choisissez Créer un alias.

Gestion des alias avec les opérations de l'API Step Functions

Step Functions fournit les opérations d'API suivantes que vous pouvez utiliser pour créer et gérer des alias de machine à états ou obtenir des informations sur les alias :

- CreateStateMachineAlias— Crée un alias pour une machine à états.
- DescribeStateMachineAlias— Renvoie les informations relatives à un alias de machine à états.
- ListStateMachineAliases— Répertorie les alias pour l'ARN de la machine à états spécifié.
- UpdateStateMachineAlias— Met à jour la configuration d'un alias de machine à états existant en modifiant son `description` ou`routingConfiguration`.
- DeleteStateMachineAlias— Supprime une version de machine à états.

Pour créer un alias nommé *`PROD`* qui pointe vers la version 1 d'une machine à états nommée à *`myStateMachine`* l'aide de AWS Command Line Interface, utilisez la `create-state-machine-alias` commande.

```
aws stepfunctions create-state-machine-alias --name PROD  --routing-
configuration "[{\"stateMachineVersionArn\":\"arn:aws:states:us-
east-1:123456789012:stateMachine:myStateMachine:1\",\"weight\":100}]"
```

Configuration du routage d’alias

Vous pouvez utiliser un alias pour acheminer le trafic d'exécution entre deux versions d'une machine à états. Supposons, par exemple, que vous souhaitiez lancer une nouvelle version de votre machine d'état. Vous pouvez réduire les risques liés au déploiement de la nouvelle version en configurant le routage sur un alias. En configurant le routage, vous pouvez envoyer la majeure partie de votre trafic vers une version antérieure et testée de votre machine à états. La nouvelle version peut alors recevoir un pourcentage inférieur, jusqu'à ce que vous puissiez confirmer que vous pouvez la transférer en toute sécurité.

Pour définir la configuration du routage, assurez-vous de publier les deux versions de machine à états vers lesquelles pointe votre alias. Lorsque vous lancez une exécution à partir d'un alias, Step Functions choisit aléatoirement la version de la machine d'état à exécuter parmi les versions spécifiées dans la configuration de routage. Il base ce choix sur le pourcentage de trafic que vous attribuez à chaque version dans la configuration de routage des alias.

Pour configurer la configuration du routage sur un alias

- Sur la page Créer un alias, sous Configuration du routage, procédez comme suit :

  a. Pour Version, choisissez la première version de machine à états vers laquelle pointe l'alias.

  b. Cochez la case Répartir le trafic entre deux versions.

  > **Tip**
  >
  > Pour pointer vers une seule version, décochez la case Répartir le trafic entre deux versions.

  c. Dans Version, choisissez la deuxième version vers laquelle l'alias doit pointer.

  d. Dans les champs Pourcentage de trafic, spécifiez le pourcentage de trafic à acheminer vers chaque version. Par exemple, entrez **`60`** et **`40`** acheminez 60 % du trafic d'exécution vers la première version et 40 % du trafic vers la deuxième version.

     Les pourcentages de trafic combinés doivent être égaux à 100 %.

Exécution d'une machine à états à l'aide d'un alias (console)

Vous pouvez démarrer des exécutions de machines à états avec un alias depuis la console ou en appelant l'action d'StartExecutionAPI avec l'ARN de l'alias. Step Functions exécute ensuite la version spécifiée par l'alias. Par défaut, si vous ne spécifiez pas de version ou d'alias lorsque vous lancez l'exécution d'une machine à états, Step Functions utilise la version la plus récente.

Pour démarrer l'exécution d'une machine à états à l'aide d'un alias

1. Ouvrez la console Step Functions, puis choisissez une machine à états existante pour laquelle vous avez créé un alias. Pour plus d'informations sur la création d'un alias, consultezCréation d'un alias de machine à états (console).
2. Sur la page détaillée de State machine, choisissez l'onglet Alias.
3. Dans la section Alias, procédez comme suit :

   a. Sélectionnez l'alias avec lequel vous souhaitez démarrer l'exécution.

   b. Choisissez Start execution (Démarrer l'exécution).
4. (Facultatif) Dans la boîte de dialogue Démarrer l'exécution, entrez le nom de l'exécution.

5. Si nécessaire, entrez l'entrée d'exécution, puis choisissez Démarrer l'exécution.

Autorisation pour les versions et les alias

Pour invoquer les actions de l'API Step Functions avec une version ou un alias, vous devez disposer des autorisations appropriées. Pour autoriser une version ou un alias à invoquer une action d'API, Step Functions utilise l'ARN de la machine d'état au lieu d'utiliser l'ARN de version ou l'ARN d'alias. Vous pouvez également restreindre les autorisations pour une version ou un alias spécifique. Pour plus d'informations, veuillez consulter Réduire les autorisations.

Vous pouvez utiliser l'exemple de politique IAM suivant d'une machine d'état nommée *myStateMachine* pour invoquer l'action d'CreateStateMachineAliasAPI afin de créer un alias de machine d'état.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "states:CreateStateMachineAlias",
      "Resource": "arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine"
    }
  ]
}
```

Lorsque vous définissez des autorisations pour autoriser ou refuser l'accès aux actions de l'API à l'aide de versions ou d'alias de la machine d'état, tenez compte des points suivants :

- Si vous utilisez le `publish` paramètre des actions CreateStateMachineet de l'UpdateStateMachineAPI pour publier une nouvelle version de la machine d'état, vous devez également disposer de l'`ALLOW`autorisation relative à l'action d'PublishStateMachineVersionAPI.
- L'action d'DeleteStateMachineAPI supprime toutes les versions et tous les alias associés à une machine d'état.

Dans cette rubrique

- Réduire les autorisations pour une version ou un alias

Réduire les autorisations pour une version ou un alias

Vous pouvez utiliser un qualificatif pour affiner l'autorisation requise par une version ou un alias. Un qualificatif fait référence à un numéro de version ou à un nom d'alias. Vous utilisez le qualificatif pour qualifier une machine d'État. L'exemple suivant est un ARN de machine d'État qui utilise un alias nommé PROD comme qualificatif.

```
arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine:PROD
```

Pour plus d'informations sur les ARN qualifiés et non qualifiés, consultez. Associer des exécutions à une version ou à un alias

Vous pouvez définir les autorisations à l'aide de la clé de contexte facultative nommée `states:StateMachineQualifier` dans la `Condition` déclaration d'une politique IAM. Par exemple, la politique IAM suivante pour une machine d'état nommée `myStateMachine` refuse l'accès permettant d'invoquer l'action d'DescribeStateMachineAPI avec un alias nommé « PROD ou 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"
          ]
        }
      }
    }
  ]
}
```

La liste suivante indique les actions d'API pour lesquelles vous pouvez restreindre les autorisations à l'aide de la clé `StateMachineQualifier` contextuelle.

- CreateStateMachineAlias

- DeleteStateMachineAlias
- DeleteStateMachineVersion
- DescribeStateMachine
- DescribeStateMachineAlias
- ListExecutions
- ListStateMachineAliases
- StartExecution
- StartSyncExecution
- UpdateStateMachineAlias

Associer des exécutions par machine à états à une version ou à un alias

Step Functions associe une exécution à une version ou à un alias basé sur l'Amazon Resource Name (ARN) que vous utilisez pour appeler l'action d'StartExecutionAPI. Step Functions exécute cette action au début de l'exécution.

Vous pouvez démarrer l'exécution d'une machine à états à l'aide d'un ARN qualifié ou non qualifié.

- ARN qualifié — Fait référence à un ARN de machine à états suffixé par un numéro de version ou un nom d'alias.

  L'exemple d'ARN qualifié suivant fait référence à la version 3 d'une machine à états nomméemyStateMachine.

  ```
  arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine:3
  ```

  L'exemple d'ARN qualifié suivant fait référence à un alias nommé PROD d'une machine à états nomméemyStateMachine.

  ```
  arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine:PROD
  ```

- ARN non qualifié : fait référence à un ARN de machine à états sans numéro de version ni suffixe de nom d'alias.

  ```
  arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine
  ```

Par exemple, si votre ARN qualifié fait référence à une version3, Step Functions associe l'exécution à cette version. Il n'associe l'exécution à aucun alias pointant vers la version3.

Si votre ARN qualifié fait référence à un alias, Step Functions associe l'exécution à cet alias et à la version vers laquelle pointe l'alias. Une exécution ne peut être associée qu'à un seul alias.

> **Note**
>
> Si vous lancez une exécution avec un ARN non qualifié, Step Functions n'associe pas cette exécution à une version, même si la version utilise la même machine à revision états. Par exemple, si la version 3 utilise la dernière révision, mais que vous démarrez une exécution avec un ARN non qualifié, Step Functions n'associe pas cette exécution à la version 3.

Dans cette rubrique

Afficher les exécutions démarrées par une version ou un alias

Step Functions propose les méthodes suivantes pour visualiser les exécutions démarrées avec une version ou un alias :

Utilisation des actions d'API

Vous pouvez afficher toutes les exécutions associées à une version ou à un alias en invoquant les actions DescribeExecutionet ListExecutionsAPI. Ces actions d'API renvoient l'ARN de la version ou de l'alias utilisé pour démarrer l'exécution. Ces actions renvoient également d'autres informations, notamment le statut et l'ARN de l'exécution.

Vous pouvez également fournir un ARN d'alias de machine à états ou un ARN de version pour répertorier les exécutions associées à un alias ou à une version spécifique.

L'exemple de réponse suivant à l'action d'ListExecutionsAPI montre l'ARN de l'alias utilisé pour démarrer une exécution de machine à états nommée *myFirstExecution*.

Le texte *en italique* dans l'extrait de code suivant représente des informations spécifiques à la ressource.

```
{
```

```
    "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"
        }
    ]
}
```

Utilisation de la console Step Functions

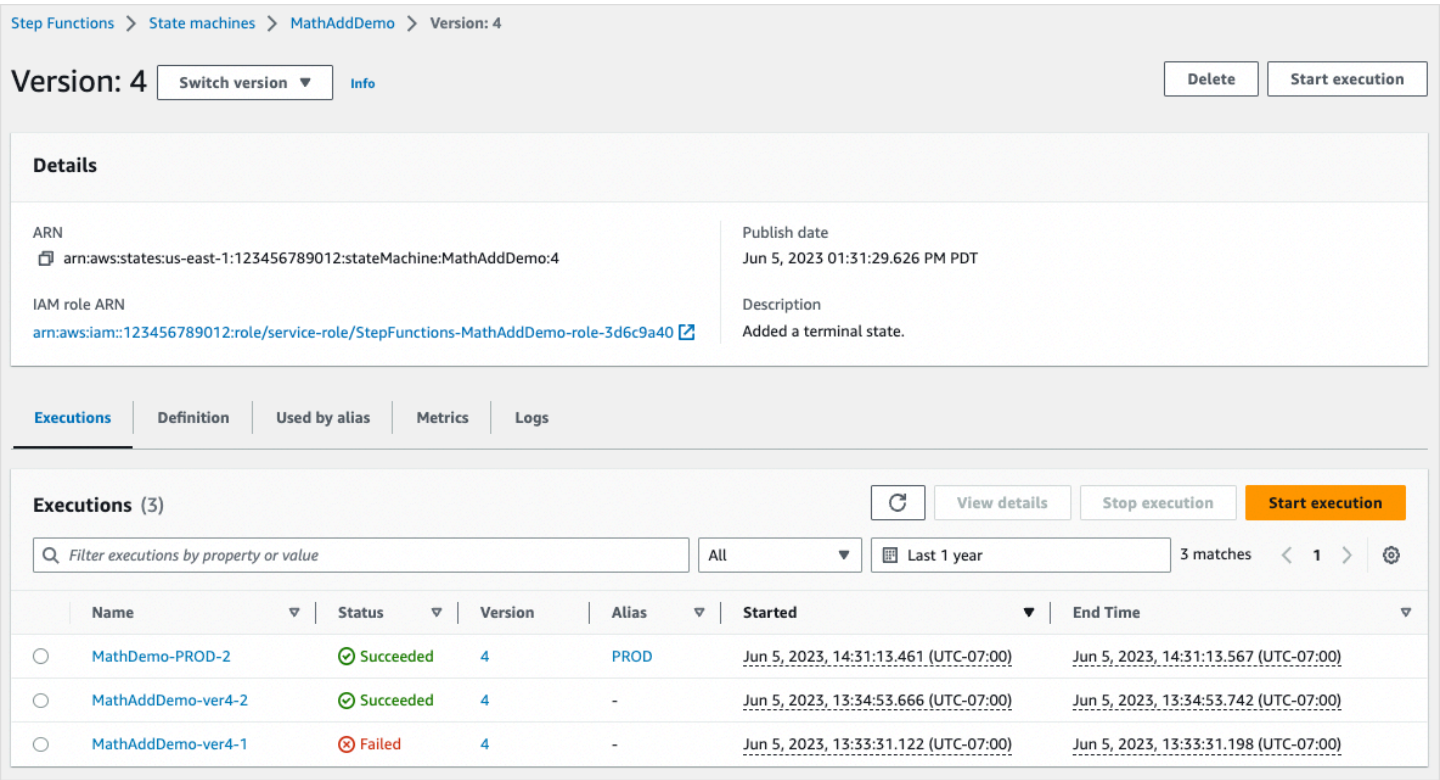
Vous pouvez également consulter les exécutions lancées par une version ou un alias depuis la console Step Functions. La procédure suivante indique comment visualiser les exécutions démarrées avec une version spécifique :

1. Ouvrez la console Step Functions, puis choisissez une machine à états existante pour laquelle vous avez publié une version ou créé un alias. Cet exemple montre comment afficher les exécutions démarrées avec une version de machine à états spécifique.
2. Choisissez l'onglet Versions, puis choisissez une version dans la liste des versions.

   > **Tip**
   >
   > Filtrez par propriété ou zone de valeur pour rechercher une version spécifique.

3. Sur la page Détails de la version, vous pouvez voir une liste de toutes les exécutions automatiques d'état en cours et passées lancées avec la version sélectionnée.

L'image suivante montre la page de console Détails de la version. Cette page répertorie les exécutions lancées par la version 4 d'une machine à états nomméeMathAddDemo. Cette liste affiche également une exécution démarrée par un alias nomméPROD. Cet alias a acheminé le trafic d'exécution vers la version 4.



Utilisation de CloudWatch métriques

Pour chaque exécution de machine à états que vous commencez par unQualified ARN, Step Functions émet des métriques supplémentaires portant le même nom et la même valeur que les métriques émises actuellement. Ces métriques supplémentaires contiennent des dimensions pour chaque identifiant de version et nom d'alias avec lesquels vous démarrez une exécution. Grâce à ces mesures, vous pouvez surveiller les exécutions des machines à états au niveau de la version et déterminer dans quels cas un scénario de restauration peut être nécessaire. Vous pouvez également créer des CloudWatch alarmes Amazon en fonction de ces statistiques.

Step Functions émet les métriques suivantes pour les exécutions que vous commencez par un alias ou une version :

- `ExecutionTime`
- `ExecutionsAborted`
- `ExecutionsFailed`
- `ExecutionsStarted`
- `ExecutionsSucceeded`
- `ExecutionsTimedOut`

Si vous avez démarré l'exécution avec une version ARN, Step Functions publie la métrique avec les `Version` dimensions `StateMachineArn` `StateMachineArn` et une seconde.

Si vous avez démarré l'exécution avec un alias ARN, Step Functions émet les métriques suivantes :

- Deux métriques pour l'ARN et la version non qualifiés.
- Une métrique avec les `Alias` dimensions `StateMachineArn` et.

Exemple de déploiement d'alias et de versions

L'exemple suivant de la technique de déploiement de Canary montre comment déployer une nouvelle version de machine à états avec leAWS Command Line Interface. Dans cet exemple, l'alias que vous créez achemine 20 % du trafic d'exécution vers la nouvelle version. Il achemine ensuite les 80 % restants vers la version précédente. Pour déployer une nouvelle version de la machine à états et déplacer le trafic d'exécution à l'aide d'un alias, procédez comme suit :

1. Publiez une version à partir de la révision de l'état actuel de la machine.

   Utilisez la publish-state-machine-version commande du AWS CLI pour publier une version à partir de la révision actuelle d'une machine d'état appelée *myStateMachine*:

   ```
   aws stepfunctions publish-state-machine-version --state-machine-arn
    arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine
   ```

   La réponse renvoie `stateMachineVersionArn` la version que vous avez publiée. Par exemple, `arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine:1`.

2. Créez un alias qui pointe vers la version de la machine d'État.

   Utilisez la create-state-machine-alias commande pour créer un alias nommé *PROD* qui pointe vers la version 1 de *myStateMachine* :

   ```
   aws stepfunctions create-state-machine-alias --name PROD --routing-
   configuration "[{\"stateMachineVersionArn\":\"arn:aws:states:us-
   east-1:123456789012:stateMachine:myStateMachine:1\",\"weight\":100}]"
   ```

3. Vérifiez que les exécutions démarrées par l'alias utilisent la version publiée correcte.

   Démarrez une nouvelle exécution de *myStateMachine* en fournissant l'ARN de l'alias **PROD** dans la start-execution commande :

```
aws stepfunctions start-execution
  --state-machine-arn arn:aws:states:us-
east-1:123456789012:stateMachineAlias:myStateMachine:PROD
  --input "{}"
```

Si vous fournissez l'ARN de la machine d'état dans la StartExecutiondemande, celle-ci utilise la version la plus récente revision de la machine d'état au lieu de la version spécifiée dans votre alias pour démarrer l'exécution.

4. Mettez à jour la définition de la machine d'état et publiez une nouvelle version.

   Mettez à jour *myStateMachine* et publiez sa nouvelle version. Pour cela, utilisez le `publish` paramètre optionnel de la update-state-machine commande :

   ```
   aws stepfunctions update-state-machine
       --state-machine-arn arn:aws:states:us-
   east-1:123456789012:stateMachine:myStateMachine
       --definition $UPDATED_STATE_MACHINE_DEFINITION
       --publish
   ```

   La réponse renvoie le `stateMachineVersionArn` pour la nouvelle version. Par exemple, `arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine:2`.

5. Mettez à jour l'alias pour qu'il pointe vers les deux versions et définissez la configuration de routage de l'alias.

   Utilisez la update-state-machine-alias commande pour mettre à jour la configuration de routage de l'aliasPROD. Configurez l'alias de sorte que 80 % du trafic d'exécution soit dirigé vers la version 1 et les 20 % restants vers la version 2 :

   ```
   aws stepfunctions update-state-machine-alias --state-machine-alias-arn
    arn:aws:states:us-east-1:123456789012:stateMachineAlias:myStateMachine:PROD
    --routing-configuration "[{\"stateMachineVersionArn\":
   \"arn:aws:states:us-east-1:123456789012:stateMachine:myStateMachine:1\",
   \"weight\":80}, {\"stateMachineVersionArn\":\"arn:aws:states:us-
   east-1:123456789012:stateMachine:myStateMachine:2\",\"weight\":20}]"
   ```

6. Remplacez la version 1 par la version 2.

Après avoir vérifié que la nouvelle version de votre machine d'état fonctionne correctement, vous pouvez déployer la nouvelle version de la machine d'état. Pour ce faire, mettez à nouveau l'alias à jour afin d'attribuer 100 % du trafic d'exécution à la nouvelle version.

Utilisez la update-state-machine-alias commande pour définir la configuration de routage de l'alias PROD à 100 % pour la version 2 :

```
aws stepfunctions update-state-machine-alias --state-machine-alias-arn
 arn:aws:states:us-east-1:123456789012:stateMachineAlias:myStateMachine:PROD
 --routing-configuration "[{\"stateMachineVersionArn\":\"arn:aws:states:us-
east-1:123456789012:stateMachine:myStateMachine:2\",\"weight\":100}]"
```

Tip

Pour annuler le déploiement de la version 2, modifiez la configuration de routage de l'alias afin de transférer 100 % du trafic vers la version récemment déployée.

```
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}]"
```

Vous pouvez utiliser des versions et des alias pour effectuer d'autres types de déploiements. Par exemple, vous pouvez effectuer un déploiement progressif d'une nouvelle version de votre machine d'État. Pour ce faire, augmentez progressivement le pourcentage pondéré dans la configuration de routage de l'alias qui pointe vers la nouvelle version.

Vous pouvez également utiliser des versions et des alias pour effectuer un déploiement bleu/vert. Pour ce faire, créez un alias nommé `green` qui exécute la version 1 actuelle de votre machine d'État. Créez ensuite un autre alias nommé `blue` qui exécute la nouvelle version, par exemple*2*. Pour tester la nouvelle version, envoyez le trafic d'exécution vers l'`blue`alias. Lorsque vous êtes sûr que votre nouvelle version fonctionne correctement, mettez à jour l'`green`alias pour qu'il pointe vers votre nouvelle version.

Procéder au déploiement progressif des versions des machines d'état

Un déploiement continu est une stratégie de déploiement qui remplace lentement les versions précédentes d'une application par de nouvelles versions d'une application. Pour effectuer un déploiement progressif d'une version de machine à états, envoyez progressivement une quantité croissante de trafic d'exécution vers la nouvelle version. La quantité de trafic et le taux d'augmentation sont des paramètres que vous configurez.

Vous pouvez effectuer le déploiement progressif d'une version à l'aide de l'une des options suivantes :

- Console Step Functions— Créez un alias pointant vers deux versions de la même machine à états. Pour cet alias, vous configurez la configuration du routage afin de transférer le trafic entre les deux versions. Pour plus d'informations sur l'utilisation de la console pour déployer des versions, voirVersionsetAlias.
- Scripts pourAWS CLIet SDK— Crée un script shell à l'aide duAWS CLIou leAWSSDK. Pour plus d'informations, consultez les sections suivantes concernant l'utilisationAWS CLIetAWSSDK.
- AWS CloudFormationmodèles— Utilisez le`AWS::StepFunctions::StateMachineVersion`et`AWS::StepFunctions::StateMachineAlias` ressources pour publier plusieurs versions de machines à états et créer un alias pointant vers une ou deux de ces versions.

Utilisez leAWS CLIpour déployer une nouvelle version de machine à états

L'exemple de script présenté dans cette section montre comment vous pouvez utiliserAWS CLIpour transférer progressivement le trafic d'une version précédente de la machine à État vers une nouvelle version de la machine à État. Vous pouvez utiliser cet exemple de script ou le mettre à jour selon vos besoins.

Ce script montre un déploiement Canary pour déployer une nouvelle version de machine à états à l'aide d'un alias. Les étapes suivantes décrivent les tâches exécutées par le script :

1. Si le`publish_revision`le paramètre est défini sur true, publiez le plus récentrevisioncomme prochaine version de la machine d'État. Cette version devient la nouvelle version active si le déploiement aboutit.

   Si vous définissez le`publish_revision`avec la valeur false, le script déploie la dernière version publiée de la machine d'état.

2. Créez un alias s'il n'existe pas encore. Si l'alias n'existe pas, dirigez 100 % du trafic de cet alias vers la nouvelle version, puis quittez le script.
3. Mettez à jour la configuration de routage de l'alias pour déplacer un faible pourcentage du trafic de la version précédente vers la nouvelle version. Vous définissez ce pourcentage de canaris à l'aide du`canary_percentage`paramètre.
4. Par défaut, surveillez le configurableCloudWatchalarmes toutes les 60 secondes. Si l'une de ces alarmes se déclenche, annulez immédiatement le déploiement en redirigeant 100 % du trafic vers la version précédente.

   Après chaque intervalle de temps, en secondes, défini dans`alarm_polling_interval`, continuez à surveiller les alarmes. Poursuivre la surveillance jusqu'à l'intervalle de temps défini dans`canary_interval_seconds`est passé.
5. Si aucune alarme n'a été déclenchée pendant`canary_interval_seconds`, transférez 100 % du trafic vers la nouvelle version.
6. Si la nouvelle version est déployée correctement, supprimez toutes les versions antérieures au numéro spécifié dans`history_max`paramètre.

```
#!/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!
```

Utilisez leAWSSDK pour déployer une nouvelle version de machine à états

L'exemple de script disponible suraws-stepfunctions-examplesmontre comment utiliser leAWSSDK pour Python permettant de transférer progressivement le trafic d'une version précédente vers une nouvelle version d'une machine d'État. Vous pouvez utiliser cet exemple de script ou le mettre à jour selon vos besoins.

Le script présente les stratégies de déploiement suivantes :

- Canari— Déplace le trafic en deux étapes.

  Dans le premier incrément, un faible pourcentage du trafic, par exemple 10 %, est transféré vers la nouvelle version. Au cours du deuxième incrément, avant qu'un intervalle de temps spécifié en secondes ne soit dépassé, le trafic restant est transféré vers la nouvelle version. Le passage à la nouvelle version pour le trafic restant n'a lieu que siCloudWatchles alarmes sont déclenchées pendant l'intervalle de temps spécifié.
- Linéaire ou roulant— Déplace le trafic vers la nouvelle version par incréments égaux avec un nombre égal de secondes entre chaque incrément.

Par exemple, si vous spécifiez le pourcentage d'incrémentation comme**20**avec un`--intervalde`**600**secondes, ce déploiement augmente le trafic de 20 % toutes les 600 secondes jusqu'à ce que la nouvelle version reçoive 100 % du trafic.

Ce déploiement annule immédiatement la nouvelle version, le cas échéant.CloudWatchles alarmes sont déclenchées.

- Tout à la fois ou bleu/vert— Transfère immédiatement 100 % du trafic vers la nouvelle version. Ce déploiement surveille la nouvelle version et la ramène automatiquement à la version précédente, le cas échéant.CloudWatchles alarmes sont déclenchées.

UtiliserAWS CloudFormationpour déployer une nouvelle version de machine à états

Ce qui suitCloudFormationun exemple de modèle publie deux versions d'une machine d'état nommée*MyStateMachine*. Il crée un alias nommé*PROD*, qui pointe vers ces deux versions, puis déploie la version2.

Dans cet exemple, 10 % du trafic est transféré vers la version2toutes les cinq minutes jusqu'à ce que cette version reçoive 100 % du trafic. Cet exemple montre également comment vous pouvez définirCloudWatchalarmes. Si l'une des alarmes que vous avez définies entre dans le`ALARM`état, le déploiement échoue et est annulé immédiatement.

```
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
```

Exécutions dans Step Functions

L'exécution de la machine d'état se produit lorsqu'une machine d'état AWS Step Functions s'exécute et effectue ses tâches. Chaque machine d'état Step Functions peut avoir plusieurs exécutions simultanées, que vous pouvez lancer depuis la console Step Functions, ou en utilisant les AWS SDK, les actions de l'API Step Functions ou le AWS Command Line Interface (AWS CLI). Une exécution reçoit des entrées JSON et produit une sortie JSON. Vous pouvez démarrer une exécution de Step Functions de différentes manières :

- Appelez l'action d'API StartExecution.
- Lancez une nouvelle exécution dans la console Step Functions.
- Utilisez Amazon EventBridge pour démarrer une exécution en réponse à un événement.
- Amazon EventBridge SchedulerÀ utiliser pour démarrer l'exécution d'une machine à états selon un calendrier.
- Démarrez une exécution avec Amazon API Gateway.

- Démarrez l'exécution d'un flux de travail imbriqué à partir d'un état de tâche.

Pour plus d'informations sur les différentes manières de travailler avec Step Functions, consultez la section Options de développement.

Démarrer les exécutions de flux de travail à partir d'un état de tâche

AWS Step Functions peut démarrer des exécutions de flux de travail directement à partir d'un état `Task` d'une machine d'état. Cela vous permet de diviser vos flux de travail en machines d'état plus petites et de démarrer l'exécution de ces machines d'état. En démarrant ces nouvelles exécutions de flux de travail, vous pouvez :

- Séparer un flux de travail de niveau supérieur des flux de travail de niveau inférieur et spécifiques aux tâches.
- Éviter les éléments répétitifs en appelant plusieurs fois une machine d'état distincte.
- Créer une bibliothèque de flux de travail modulaires réutilisables pour un développement plus rapide.
- Réduire la complexité et simplifier la modification et le dépannage des machines d'état.

Step Functions peut démarrer ces exécutions de flux de travail en appelant sa propre API en tant que service intégré. Il vous suffit d'appeler l'action d'API `StartExecution` à partir de votre état `Task` et de transmettre les paramètres nécessaires. Vous pouvez appeler l'API Step Functions à l'aide de n'importe quel modèle d'intégration de service.

> **Tip**
>
> Pour déployer un exemple de flux de travail imbriqué sur votreCompte AWS, consultez le Module 13 - Flux de travail Express imbriqués.

Pour démarrer une nouvelle exécution d'une machine à états, utilisez un `Task` état similaire à l'exemple suivant :

```
{
   "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
}
```

Cet état `Task` démarre une nouvelle exécution de la machine d'état `HelloWorld` et transmet le commentaire JSON en tant qu'entrée.

> **Note**
>
> Les quotas d'action d'API `StartExecution` peuvent réduire le nombre d'exécutions que vous pouvez démarrer. Utilisez `Retry` sur `StepFunctions.ExecutionLimitExceeded` pour vous assurer que votre exécution a démarré. Consultez les rubriques suivantes.
>
> - Quotas liés à la limitation des actions des API
> - Gestion des erreurs dans Step Functions

Exécution de flux de travail associés

Pour associer une exécution de flux de travail démarrée à l'exécution qui l'a démarrée, transmettez l'ID d'exécution de l' objet de contexte à l'entrée d'exécution. Vous pouvez accéder à l'ID depuis l'objet de contexte de votre état `Task` dans une exécution en cours d'exécution. Transmettez l'ID d'exécution en ajoutant `.$` au nom du paramètre et en référençant l'ID dans l'objet de contexte avec `$$.Execution.Id`.

```
"AWS_STEP_FUNCTIONS_STARTED_BY_EXECUTION_ID.$": "$$.Execution.Id"
```

Vous pouvez utiliser un paramètre spécial nommé `AWS_STEP_FUNCTIONS_STARTED_BY_EXECUTION_ID` lorsque vous démarrez une exécution. Si elle est incluse, cette association fournit des liens dans la section Détails des étapes de la console Step Functions. Dans ce cas, vous pouvez facilement suivre les exécutions de vos flux de travail, du démarrage à l'exécution. À l'aide de l'exemple précédent, associez l'ID d'exécution à l'exécution démarrée de la machine d'état `HelloWorld` comme suit.

```
{
   "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
}
```

Pour plus d'informations, consultez les ressources suivantes :

- Utilisation avec d'autres services
- Transmettre des paramètres à une API de service
- Accès à l'objet de contexte
- AWS Step Functions

Utilisation d'Amazon EventBridge Scheduler avec AWS Step Functions

Amazon EventBridge Scheduler est un planificateur sans serveur qui vous permet de créer, d'exécuter et de gérer des tâches à partir d'un service géré centralisé. Avec EventBridge Scheduler, vous pouvez créer des plannings à l'aide d'expressions cron et rate pour les modèles récurrents, ou configurer des appels ponctuels. Vous pouvez configurer des fenêtres de temps flexibles pour la livraison, définir des limites de nouvelles tentatives ainsi que la durée de conservation maximale pour les invocations d'API en échec.

Par exemple, avec EventBridge Scheduler, vous pouvez démarrer l'exécution d'une machine à états selon un calendrier lorsqu'un événement lié à la sécurité se produit ou pour automatiser une tâche de traitement de données.

Cette page explique comment utiliser le EventBridge Scheduler pour démarrer l'exécution d'une machine d'état Step Functions selon un calendrier.

Rubriques

Configurer le rôle d'exécution

Lorsque vous créez un nouveau calendrier, le EventBridge planificateur doit être autorisé à appeler son opération d'API cible en votre nom. Vous accordez ces autorisations au EventBridge Scheduler à l'aide d'un rôle d'exécution. La politique d'autorisation que vous associez au rôle d'exécution de votre planification définit les autorisations requises. Ces autorisations dépendent de l'API cible que le EventBridge Scheduler doit appeler.

Lorsque vous utilisez la console du EventBridge planificateur pour créer un calendrier, comme dans la procédure suivante, le EventBridge planificateur définit automatiquement un rôle d'exécution en fonction de la cible que vous avez sélectionnée. Si vous souhaitez créer un calendrier à l'aide de l'un des SDK du EventBridge planificateurAWS CloudFormation, vous devez disposer d'un rôle d'exécution existant qui accorde les autorisations dont le EventBridge planificateur a besoin pour appeler une cible. AWS CLI Pour plus d'informations sur la configuration manuelle d'un rôle d'exécution pour votre calendrier, consultez la section Configuration d'un rôle d'exécution dans le guide de l'utilisateur du EventBridge planificateur.

Créer une planification

Pour créer une planification à l'aide de la console

1. Ouvrez la console Amazon EventBridge Scheduler à l'adresse https://console.aws.amazon.com/scheduler/home.
2. Sur la page Planifications, choisissez Créer une planification.

3. Sur la page Spécifier le détail de la planification, dans la section Nom et description de la planification, procédez comme suit :

   a. Pour Nom de la planification, saisissez un nom à attribuer à votre planification. Par exemple, **`MyTestSchedule`**.

   b. (Facultatif) Dans le champ Description, saisissez une description de la planification. Par exemple, **`My first schedule`**.

   c. Pour Groupe de planifications, choisissez un groupe de planifications dans la liste déroulante. Si vous n'avez pas de groupe, choisissez par défaut. Pour créer un groupe de planifications, choisissez Crée votre propre planification.

      Vous utilisez des groupes de planifications pour leur ajouter des balises.

4. • Choisissez vos options de planification.

Occurrence	Faites ceci...
Planification ponctuelle Une planification ponctuelle n'invoque un objectif qu'une seule fois à la date et à l'heure que vous indiquez.	Pour Date et heure, procédez comme suit : • Entrez une date valide au format `YYYY/MM/DD` . • Entrez un horodatage au format `hh:mm` de 24 heures. • Dans le champ Fuseau horaire, choisissez le fuseau horaire.
Planification récurrente Une planification récurrente invoque un objectif à un taux que vous spécifiez à l'aide d'une expression cron ou d'une expression rate.	a. Pour Schedule type (Planifier le type), effectuez l'une des étapes suivantes : • Pour utiliser une expression cron afin de définir la planification, choisissez

Occurrence	Faites ceci...	
	Planification basée sur cron et entrez l'expression cron. • Pour utiliser une expression de rythme pour définir la planification, choisissez Planification basée sur le rythme. Pour plus d'informations sur les expressions cron et rate, consultez la section Types de planification sur EventBridge Scheduler dans le guide de l'utilisateur d'Amazon EventBridge Scheduler. b. Pour Fenêtre temporelle flexible, choisissez Désactivé pour désactiver cette option ou choisir l'une des fenêtres temporelles prédéfinies. Par exemple, si vous choisissez15 minutes et que vous définissez une planification récurrente pour invoquer son objectif une fois par heure, la planification s'exécute dans les 15	

Occurrence	Faites ceci...	
	minutes suivant le début de chaque heure.	

5. (Facultatif) Si vous avez choisi Planification récurrente à l'étape précédente, dans la section Délai, procédez comme suit :

   a. Dans le champ Fuseau horaire, choisissez un fuseau horaire.

   b. Pour Date et heure de début, entrez une date valide au format `YYYY/MM/DD`, puis spécifiez un horodatage au format `hh:mm` de 24 heures.

   c. Pour Date et heure de fin, entrez une date valide au format `YYYY/MM/DD`, puis spécifiez un horodatage au format `hh:mm` de 24 heures.

6. Choisissez Suivant.

7. Sur la page Sélectionner une cible, choisissez l'opération AWS d'API invoquée par le EventBridge planificateur :

   a. Choisissez AWS Step Functions StartExecution.

   b. Dans la StartExecutionsection, sélectionnez une machine à états ou choisissez Créer une nouvelle machine à états.

      À l'heure actuelle, vous ne pouvez pas exécuter de flux de travail Synchrous Express de manière planifiée.

   c. Entrez une charge utile JSON pour l'exécution. Même si votre machine d'état ne nécessite aucune charge utile JSON, vous devez toujours inclure les entrées au format JSON, comme indiqué dans l'exemple suivant.

```
{
    "Comment": "sampleJSONData"
}
```

8. Choisissez Suivant.

9. Sur la page Settings (Paramètres), procédez comme suit :

   a. Pour activer la planification, sous État de la planification, activez Activer la planification.

   b. Pour configurer une stratégie de nouvelles tentatives pour votre planification, sous Politique de nouvelle tentative et file d'attente de lettres mortes (DLQ), procédez comme suit :

- Activez Réessayer.
- Pour Âge maximal de l'événement, entrez le nombre maximum d'heures et de minutes pendant lequel le EventBridge planificateur doit conserver un événement non traité.
- La durée maximale est 24 heures.
- Pour Nombre maximum de tentatives, entrez le nombre maximum de fois que le EventBridge planificateur réessaie le calendrier si la cible renvoie une erreur.

  La valeur maximale est 185 nouvelles tentatives.

Avec les politiques de nouvelle tentative, si un calendrier ne parvient pas à invoquer sa cible, le EventBridge planificateur le réexécute. Si elle est configurée, vous devez définir la durée de rétention maximale et les nouvelles tentatives pour la planification.

c. Choisissez l'endroit où EventBridge Scheduler stocke les événements non livrés.

Option File d'attente de lettres mortes (DLQ)	Faites ceci...	
Ne stockez pas	Sélectionnez Aucun.	
Stocker l'événement dans le même Compte AWS où vous créez la planification	a. Choisissez Sélectionnez une file d'attente Amazon SQS dans mon Compte AWS en tant que DLQ. b. Choisissez l'Amazon Resource Name (ARN) de la file d'attente Amazon SQS.	

Option File d'attente de lettres mortes (DLQ)	Faites ceci...	
Stocker l'événement dans un autre Compte AWS que celui où vous créez la planification	a. Choisissez Spécifier une file d'attente Amazon SQS dans un autre Comptes AWS en tant que DLQ. b. Entrez l'Amazon Resource Name (ARN) de la file d'attente Amazon SQS.	

d. Pour utiliser une clé gérée par le client afin de chiffrer votre entrée cible, sous Chiffrement, choisissez Personnaliser les paramètres de chiffrement (avancé).

   Si vous choisissez cette option, entrez un ARN de clé KMS existant ou choisissez Créez un AWS KMS keypour accéder à la console AWS KMS. Pour plus d'informations sur la manière dont EventBridge Scheduler chiffre vos données au repos, consultez la section Chiffrement au repos dans le guide de l'utilisateur d'Amazon EventBridge Scheduler.

e. Pour que le EventBridge planificateur crée un nouveau rôle d'exécution pour vous, choisissez Créer un nouveau rôle pour ce calendrier. Ensuite, saisissez un nom pour Nom du rôle. Si vous choisissez cette option, le EventBridge planificateur associe au rôle les autorisations requises pour votre cible modélisée.

10. Choisissez Suivant.
11. Sur la page Examiner et créer une planification, examinez les détails de votre planification. Dans chaque section, choisissez Modifier pour revenir à cette étape et modifier ses détails.
12. Choisissez Créer une planification.

    Vous pouvez consulter la liste de vos planifications nouvelles et existantes sur la page Planifications. Sous la colonne État, vérifiez que votre nouvelle planification est activée.

Pour confirmer que EventBridge Scheduler a invoqué la machine à états, consultez les journaux Amazon CloudWatch de la machine à états.

Ressources connexes

Pour plus d'informations sur le EventBridge planificateur, consultez les rubriques suivantes :

- EventBridge Guide de l'utilisateur du planificateur
- EventBridge Référence de l'API Scheduler
- EventBridge Tarification du planificateur

Exécutions de flux de travail standard et express dans la console

Lorsque vous créez une machine à états, vous sélectionnez un type Standard ou Express. Le type par défaut pour les machines à états est Standard. Une machine à états dont le type est Standard est appelée flux de travail standard et une machine à états dont le type est Express est appelée flux de travail express.

Pour les flux de travail Standard et Express, vous définissez votre machine à états à l'aide duAmazon States Language. Les exécutions de vos machines à états se comporteront différemment selon le type que vous sélectionnez.

> **Important**
>
> Le type que vous choisissez ne peut pas être modifié une fois que vous avez créé la machine à états.

Pour plus d'informations sur les flux de travail Standard et Express, consultezFlux de travail standard ou express.

L'historique des exécutions de flux de travail standard est enregistré dans Step Functions, tandis que l'historique des exécutions de flux de travail Express n'est pas enregistré dans Step Functions. Pour enregistrer l'historique de l'exécution d'un flux de travail Express, vous devez le configurer pour envoyer des journaux à Amazon CloudWatch. Pour de plus amples informations, veuillez consulter Journalisation à l'aideCloudWatchJournaux.

Une fois la journalisation configurée dans un flux de travail Express, vous pouvez consulter ses exécutions dans la console Step Functions. L'expérience de la console permettant de visualiser les exécutions de flux de travail Express et les exécutions de flux de travail standard est similaire, à l'exception des différences et limitations suivantes.

> **Note**
>
> Les données d'exécution des flux de travail Express étant affichées à l'aide de CloudWatch Logs Insights, l'analyse des journaux entraîne des frais. Par défaut, votre groupe de journaux répertorie uniquement les exécutions effectuées au cours des trois dernières heures. Si vous spécifiez un intervalle de temps plus long incluant un plus grand nombre d'événements d'exécution, vos coûts augmenteront. Pour plus d'informations, consultez Vended Logs sous l'onglet Logs de la page CloudWatch Tarification etJournalisation à l'aideCloudWatchJournaux.

Table des matières

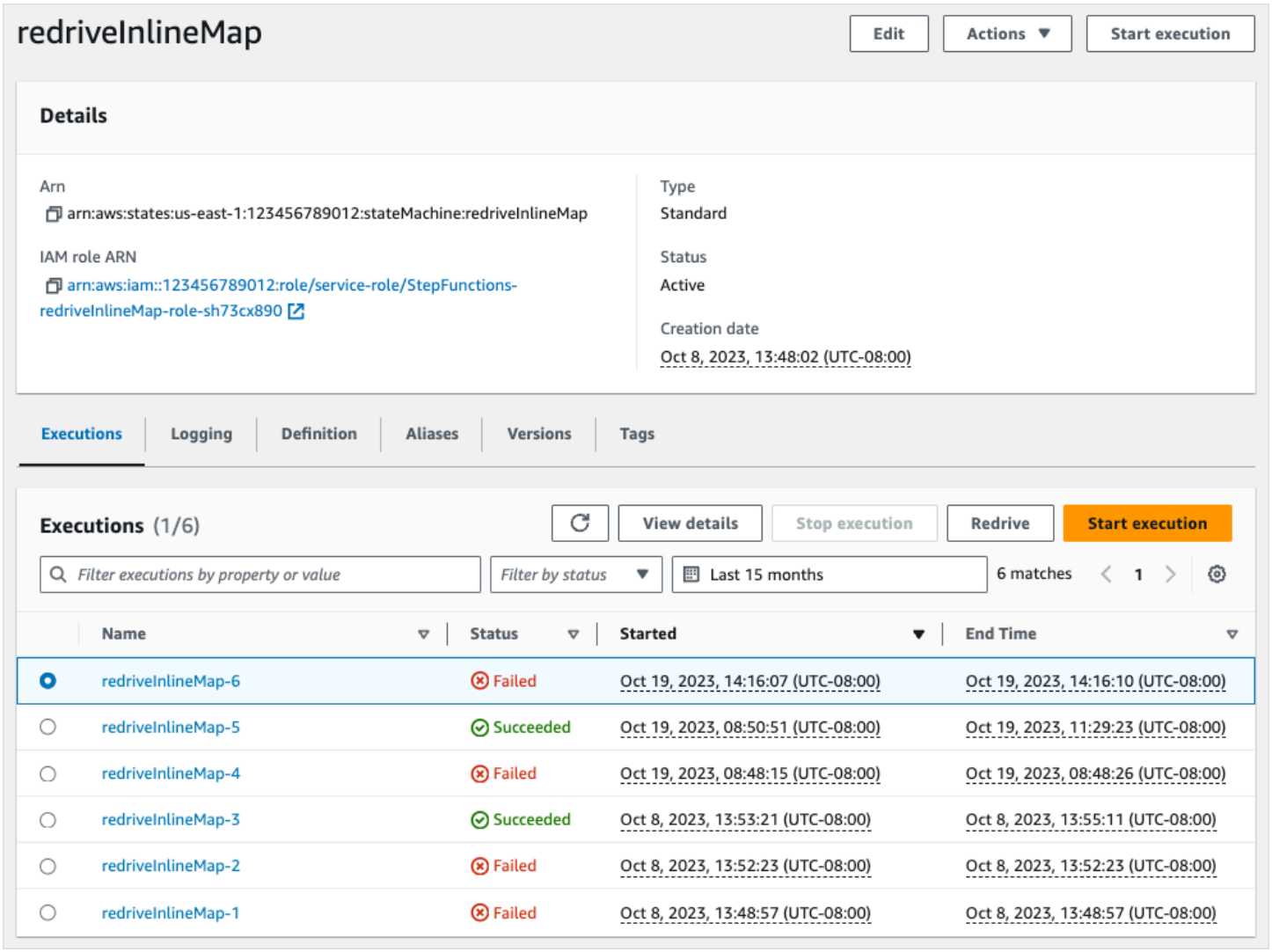
Différences d'expérience sur console

Pour tous les flux de travail Standard et Express, vous pouvez consulter les détails, tels que la machine à états et son rôle ARN IAM, sur la page détaillée de la machine à états de la console Step Functions.

Sur la page détaillée de la machine à états, vous pouvez également voir une liste des historiques d'exécution de votre machine à états sous l'onglet Exécutions. Utilisez la zone Filtrer les exécutions par propriété ou valeur pour rechercher une exécution, une version ou un alias spécifiques de la machine à états choisie. Utilisez le menu déroulant Tous pour filtrer les historiques d'exécution en fonction de leur statut. Vous pouvez également choisir un historique d'exécution et sélectionner le bouton Afficher les détails pour ouvrir sa page de détails d'exécution.

Flux de travail standard

Les historiques d'exécution des flux de travail standard sont toujours disponibles pour les exécutions effectuées au cours des 90 derniers jours.



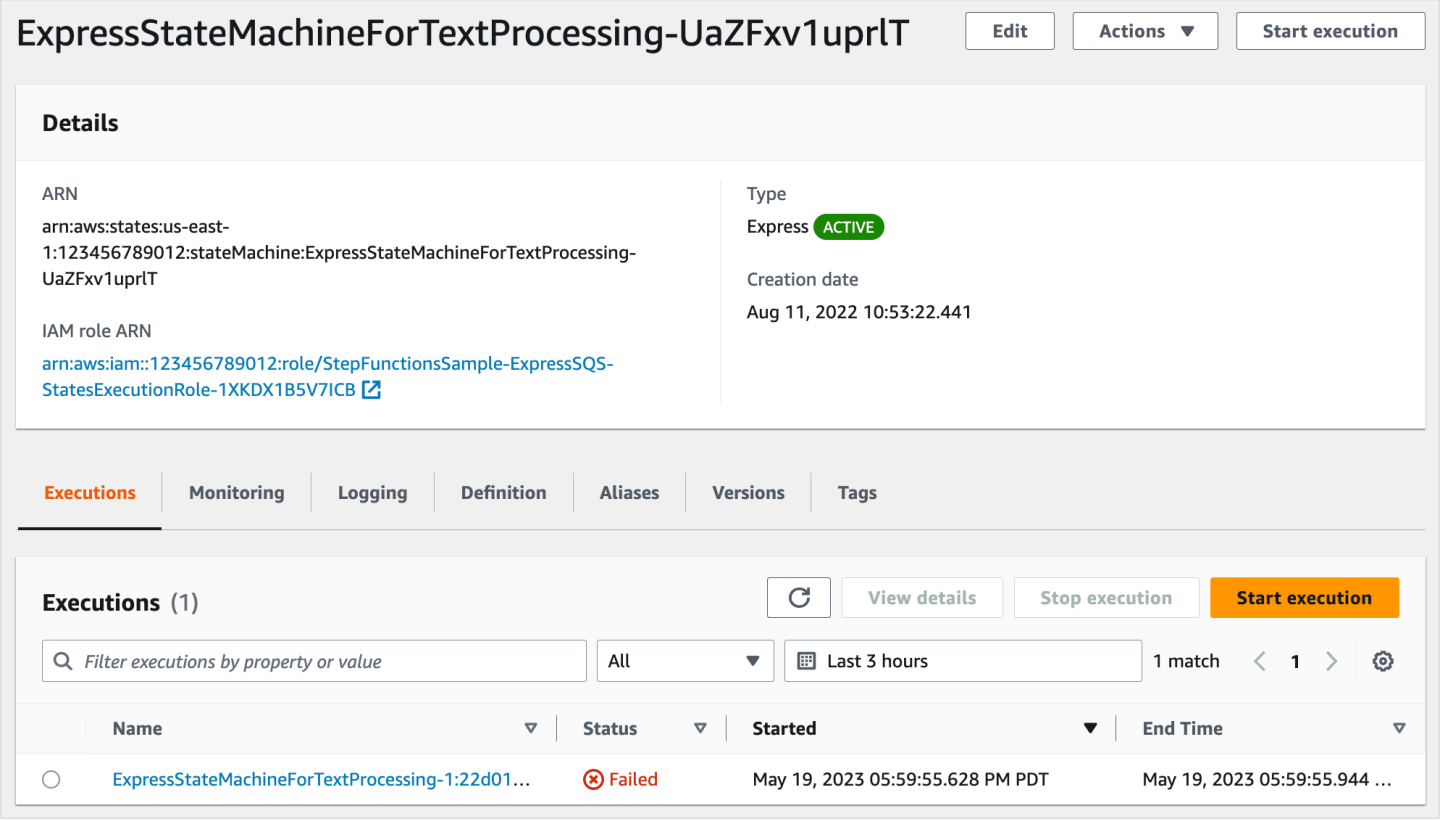
Flux de travail express

Pour afficher l'historique d'exécution des flux de travail Express, la console Step Functions récupère les données de journal collectées via un groupe de CloudWatch journaux de journaux.

Vous devez également activer la nouvelle expérience de console pour visualiser les exécutions de flux de travail Express. Pour ce faire, cliquez sur le bouton Activer affiché dans la bannière de l'onglet Exécutions. Une fois que vous aurez sélectionné ce bouton, il ne réapparaîtra plus.

> **Tip**
>
> Pour passer de l'activation à la désactivation de l'expérience console, utilisez le bouton Activer l'historique d'exécution express.

L'historique des exécutions effectuées au cours des trois dernières heures est disponible par défaut. Vous pouvez ajuster cette plage de temps ou définir une plage personnalisée. Si vous spécifiez un intervalle de temps plus long incluant un plus grand nombre d'événements d'exécution, le coût d'analyse des journaux augmentera. Pour plus d'informations, consultez Vended Logs sous l'onglet Logs de la page CloudWatch Tarification etJournalisation à l'aideCloudWatchJournaux.



Considérations et limites relatives à l'affichage des exécutions de flux de travail Express

Lorsque vous consultez les exécutions de flux de travail Express sur la console Step Functions, gardez à l'esprit les considérations et limites suivantes.

- La disponibilité des détails d'exécution du flux de travail Express repose sur Amazon CloudWatch Logs
- Les détails partiels de l'exécution du flux de travail Express sont disponibles si le niveau de journalisation est ERROR ou FATAL
- La définition de la machine à états d'une ancienne exécution ne peut pas être visualisée une fois qu'elle a été mise à jour

La disponibilité des détails d'exécution du flux de travail Express repose sur Amazon CloudWatch Logs

> **Note**
>
> Si vous n'activez pas la nouvelle expérience de console pour afficher les exécutions de flux de travail Express, les historiques d'exécution et les détails d'exécution correspondants ne sont pas disponibles dans la console Step Functions. Pour activer la nouvelle expérience de console, cliquez sur le bouton Activer affiché dans la bannière de l'onglet Exécutions.

Pour les flux de travail Express, leur historique d'exécution et les informations d'exécution détaillées sont collectés via CloudWatch Logs Insights. Ces informations sont conservées dans le groupe de CloudWatch journaux que vous spécifiez lorsque vous créez la machine à états. L'historique des exécutions de la machine à états est affiché sous l'onglet Executions de la console Step Functions. Des informations détaillées sur chaque exécution de la machine d'état sont affichées sur la page Détails de l'exécution pour l'exécution choisie.

> **Warning**
>
> Si vous supprimez les CloudWatch journaux d'un flux de travail Express, il ne sera pas répertorié sous l'onglet Exécutions.

Nous vous recommandons d'utiliser le niveau de journalisation par défaut ALL pour consigner tous les types d'événements d'exécution. Vous pouvez mettre à jour le niveau de journalisation selon les besoins de vos machines d'état existantes lorsque vous les modifiez. Pour plus d’informations, consultez Journalisation à l'aideCloudWatchJournaux et Niveaux de journalisation.

Les détails partiels de l'exécution du flux de travail Express sont disponibles si le niveau de journalisation est ERROR ou FATAL

Par défaut, le niveau de journalisation pour les exécutions de flux de travail Express est défini sur ALL. Si vous modifiez le niveau de journalisation, les historiques d'exécution et les détails d'exécution des exécutions terminées ne seront pas affectés. Cependant, toutes les nouvelles exécutions émettront des journaux en fonction du niveau de journal mis à jour. Pour plus d’informations, consultez Journalisation à l'aideCloudWatchJournaux et Niveaux de journalisation.

Par exemple, si vous modifiez le niveau de journalisation de ALL en ERROR ou FATAL, l'onglet Executions de la console Step Functions répertorie uniquement les exécutions ayant échoué. Dans l'onglet Affichage des événements, la console affiche uniquement les détails des événements relatifs aux étapes de la machine à états qui ont échoué.

Nous vous recommandons d'utiliser le niveau de journalisation par défaut ALL pour consigner tous les types d'événements d'exécution. Vous pouvez mettre à jour le niveau de journalisation selon les besoins de vos machines d'état existantes lorsque vous modifiez la machine d'état.

La définition de la machine à états d'une ancienne exécution ne peut pas être visualisée une fois qu'elle a été mise à jour

Les définitions de machines à états pour les exécutions passées ne sont pas stockées pour les flux de travail Express. Si vous modifiez la définition de la machine à états, vous ne pouvez consulter la définition de la machine à états que pour les exécutions utilisant la définition la plus récente.

Par exemple, si vous supprimez une ou plusieurs étapes de la définition de votre machine à états, Step Functions détecte une incompatibilité entre la définition et les événements d'exécution précédents. Comme les définitions précédentes ne sont pas stockées pour les flux de travail Express, Step Functions ne peut pas afficher la définition de machine à états pour les exécutions exécutées sur une version antérieure de la définition de machine à états. Par conséquent, les onglets Entrée et sortie d'exécution, Définition, Vue graphique et Vue tabulaire ne sont pas disponibles pour les exécutions exécutées sur des versions précédentes d'une définition de machine à états.

Affichage et débogage des exécutions sur la console Step Functions

La page Détails de l'exécution de la console Step Functions présente des informations sur les exécutions automatiques passées et en cours pour les flux de travail standard et express. Ces informations sont affichées sous forme de tableau de bord. Par exemple, vous pouvez trouver la définition du langage Amazon States Language de la machine à états, son statut d'exécution, son ARN et le nombre total de transitions d'états. Vous pouvez également consulter les détails d'exécution de n'importe quel état individuel dans la machine à états.

Table des matières

- Détails de l'étape
- Événements
- Tutoriel : Examen des exécutions par des machines à états à l'aide de la console Step Functions
  - Étape 1 : créer et tester les fonctions Lambda requises
  - Étape 2 : Création et exécution de la machine à états
  - Étape 3 : Afficher les détails de l'exécution de la machine à états
  - Étape 4 : Explorez les différents modes d'affichage

Page de détails d'exécution — Vue d'ensemble de l'interface

Vous trouverez les détails de toutes vos exécutions automatiques en cours et passées pour les flux de travail standard et express sur la page Détails de l'exécution. Si vous avez spécifié un ID d'exécution au début de votre exécution, cette page est intitulée avec cet ID d'exécution. Sinon, il est intitulé avec l'ID d'exécution unique que Step Functions génère automatiquement pour vous.

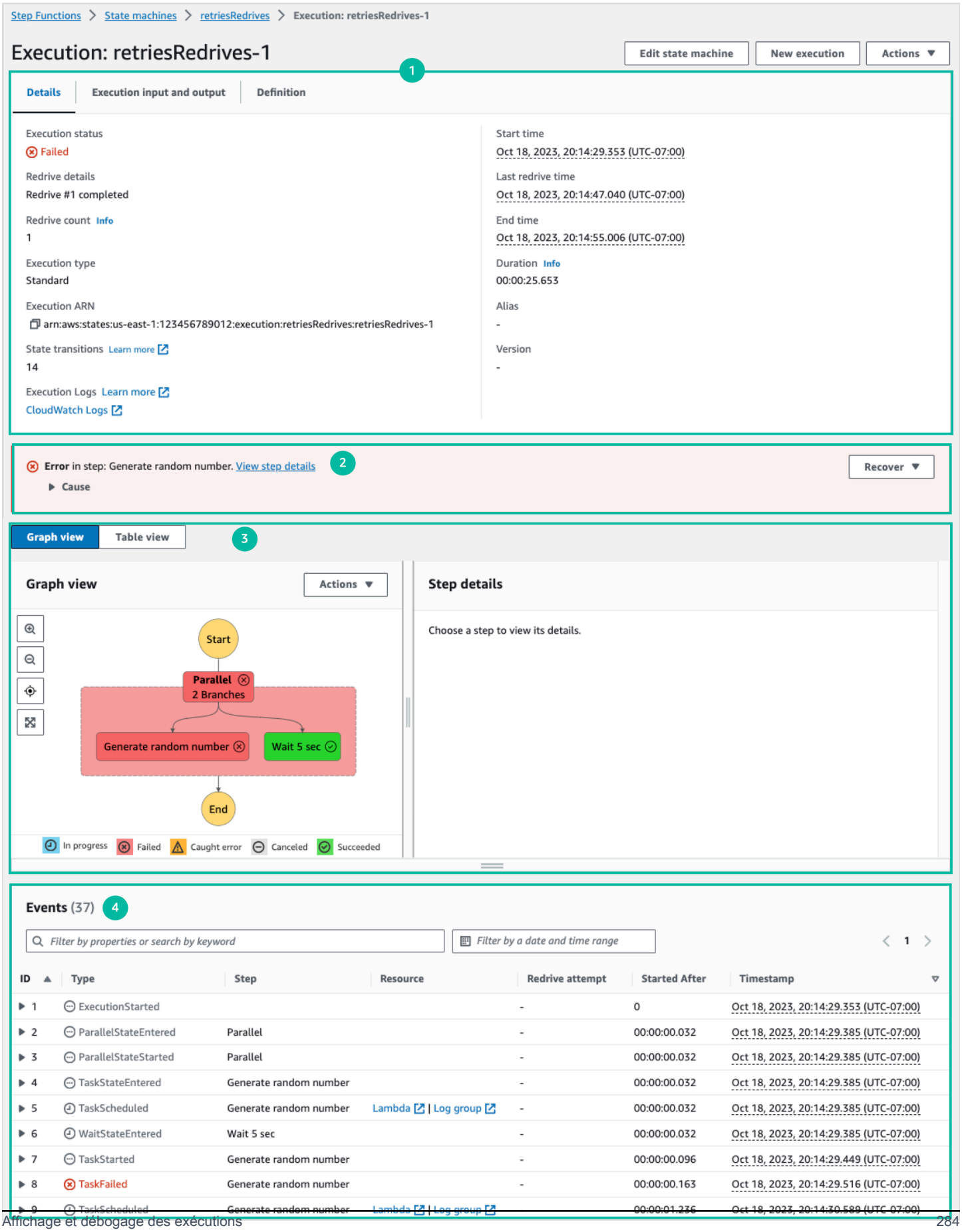
Outre les métriques d'exécution, la page Détails de l'exécution fournit les options suivantes pour gérer votre machine à états et son exécution :

Button	Cliquez sur ce bouton pour :
Modifier la machine à états	Modifiez la définition du langage Amazon States Language de votre machine à états.
Nouvelle exécution	Lancez une nouvelle exécution de votre machine d'état.
Actions	Propose les options suivantes parmi lesquelles choisir : • Arrêter l'exécution : arrête une exécution en cours. Cette option n'est pas disponible pour les exécutions terminées. • Redrive— des Redrive exécutions de flux de travail standard qui ne se sont pas terminées correctement au cours des 14 derniers jours. Il s'agit notamment des exécutions échouées,

Button	Cliquez sur ce bouton pour :
	abandonnées ou dont le délai imparti a expiré. Pour de plus amples informations, veuillez consulter Redrivingexécutions. • Exporter : exportez les détails de l'exécuti on au format JSON pour les partager avec d'autres personnes ou effectuer une analyse hors ligne. • Envoyer des commentaires : partagez des commentaires sur l'interface.

> **Afficher les exécutions démarrées par une version ou un alias**
>
> Vous pouvez également consulter les exécutions démarrées avec une version ou un alias dans la console Step Functions. Pour plus d'informations, consultez la section Répertorier les exécutions pour les versions et les alias.

La page de console Détails de l'exécution contient les sections suivantes :


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. Résumé de l'exécution
2. Error message (Message d’erreur)
3. Mode d'affichage
4. Détails de l'étape
5. Événements

Résumé de l'exécution

La section Résumé de l'exécution apparaît en haut de la page Détails de l'exécution. Cette section fournit une vue d'ensemble des détails d'exécution de votre flux de travail. Ces informations sont réparties entre les trois onglets suivants :

Détails

Affiche des informations, telles que l'état de l'exécution, l'ARN et les horodatages des heures de début et de fin de l'exécution. Vous pouvez également consulter le nombre total de transitions d'état survenues lors de l'exécution de l'exécution de la machine à états. Vous pouvez également consulter les liens vers X-Ray Trace Map et Amazon CloudWatch Execution Logs si vous avez activé le suivi ou les journaux pour votre machine à états.

Si l'exécution de votre machine à états a été lancée par une autre machine à états, vous pouvez consulter le lien vers la machine à états parent dans cet onglet.

Si votre machine à états a été exécutée redriven, cet onglet affiche des informations redrive connexes, par exemple le Redrivenombre.

Entrée et sortie d'exécution

Affiche l'entrée et la sortie de l'exécution de la machine à états side-by-side.

Définition

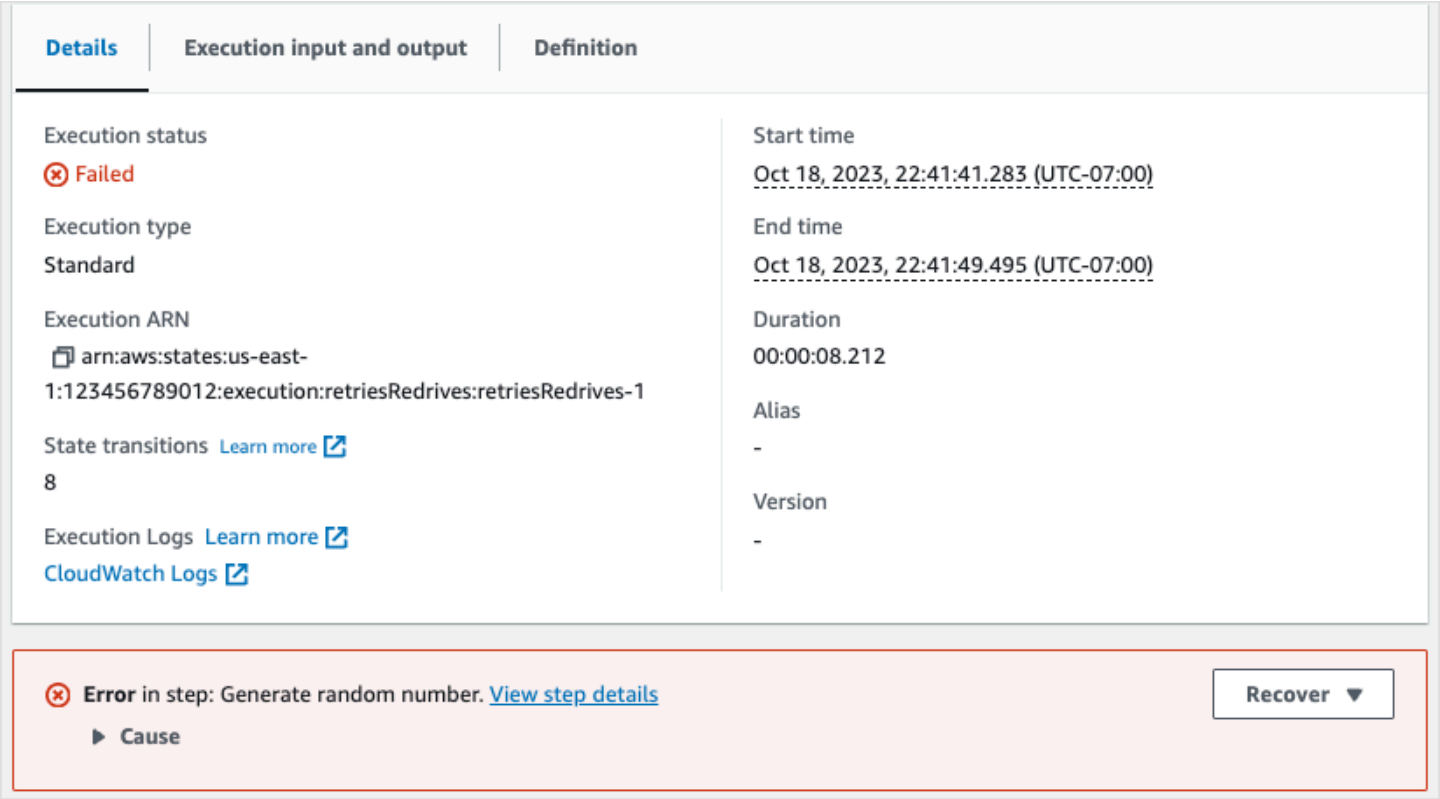
Affiche la définition du langage Amazon States de la machine à états.

Error message (Message d’erreur)

Si l'exécution de votre machine à états échoue, la page Détails de l'exécution affiche un message d'erreur. Choisissez Cause ou Afficher les détails de l'étape dans le message d'erreur pour voir la raison de l'échec de l'exécution ou l'étape à l'origine de l'erreur.

Si vous choisissez Afficher les détails de l'étape, Step Functions met en évidence l'étape à l'origine de l'erreur dans les onglets Détails de l'étape, Vue graphique et Vue sous forme de tableau. Si l'étape est une tâche, une carte ou un état parallèle pour lequel vous avez défini de nouvelles tentatives, le volet Détails de l'étape affiche l'onglet Réessayer correspondant à l'étape. De plus, si vous avez effectué redriven l'exécution, vous pouvez voir les nouvelles tentatives et les détails de redrive l'exécution dans l'redrivesonglet Rétentatives et tentatives du volet des détails de l'étape.

À partir du bouton déroulant Restaurer de ce message d'erreur, vous pouvez soit effectuer redrive vos exécutions infructueuses, soit démarrer une nouvelle exécution. Pour de plus amples informations, veuillez consulter Redrivingexécutions.



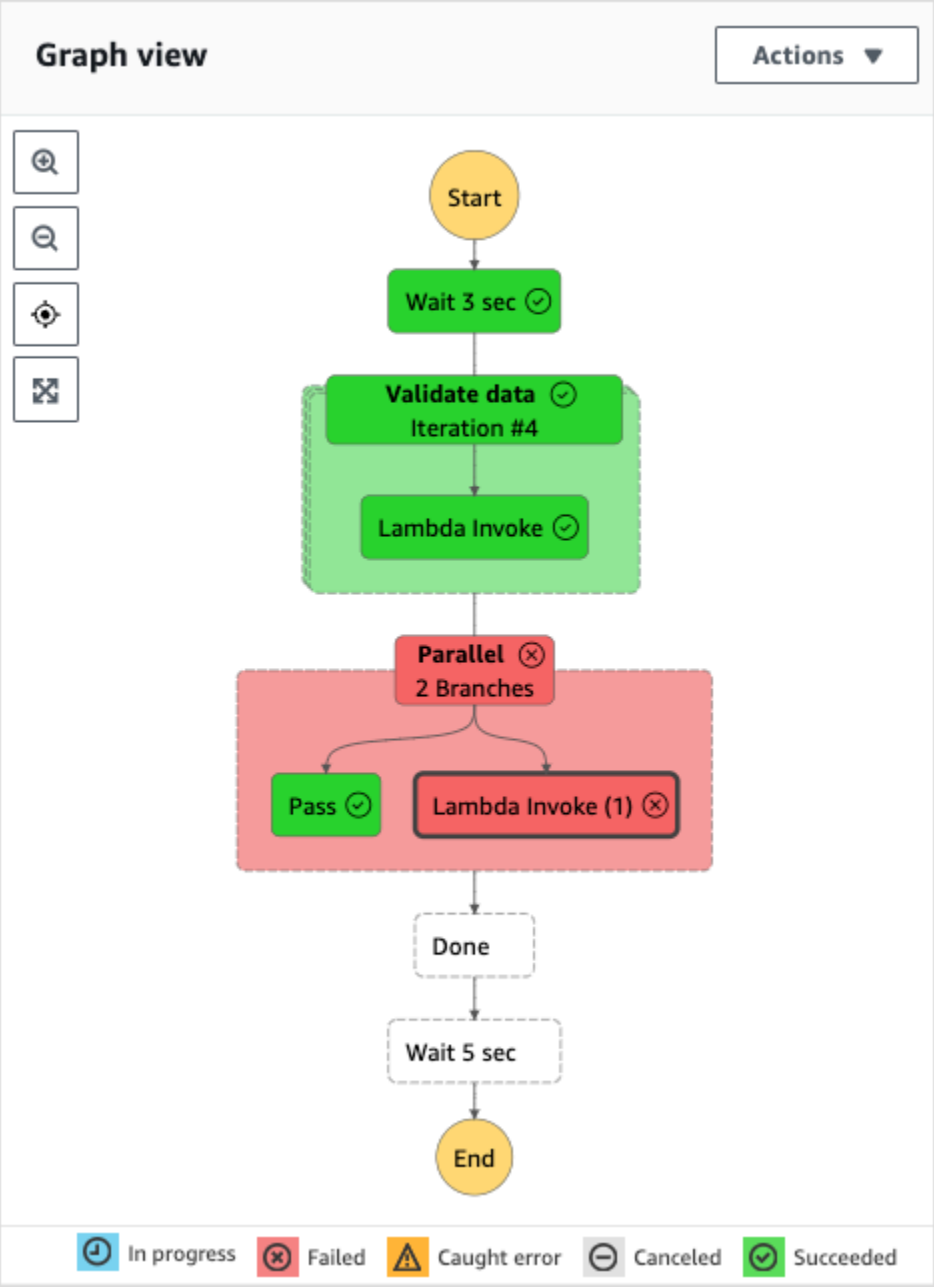
Mode d'affichage

La section Mode d'affichage contient deux visualisations différentes pour votre machine à états. Vous pouvez choisir d'afficher une représentation graphique du flux de travail, un tableau décrivant les états de votre flux de travail ou une liste des événements associés à l'exécution de votre machine à états :

> **Note**
>
> Choisissez un onglet pour afficher son contenu.

Graph view

Le mode d'affichage graphique affiche une représentation graphique de votre flux de travail. Une légende est incluse en bas qui indique l'état d'exécution de la machine à états. Il contient également des boutons qui vous permettent de zoomer, de dézoomer, d'aligner l'ensemble du flux de travail au centre ou d'afficher le flux de travail en mode plein écran.



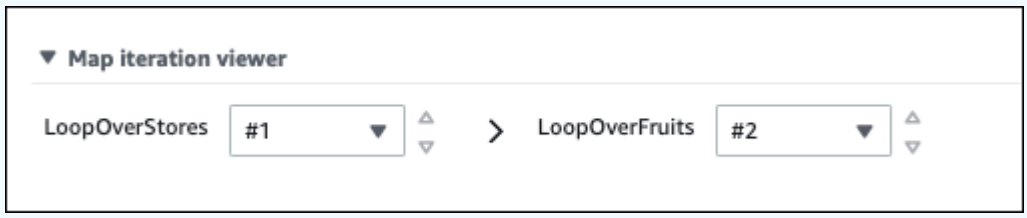
Dans cette vue, vous pouvez choisir n'importe quelle étape de votre flux de travail pour afficher les détails de son exécution dans le composant Détails de l'étape. Lorsque vous avez choisi une étape dans la vue graphique, la vue Tableau affiche également cette étape. Cela est également

vrai dans le sens inverse. Si vous choisissez une étape dans la vue Tableau, la vue Graphique montre la même étape.

Si votre machine d'états contient un `Map` état, un `Parallel` état ou les deux, vous pouvez afficher leurs noms dans le flux de travail dans la vue graphique. En outre, pour l'`Map`état, la vue graphique vous permet de passer d'une itération à une autre des données d'exécution de l'état de la carte. Par exemple, si l'état de votre carte comporte cinq itérations et que vous souhaitez afficher les données d'exécution pour les troisième et quatrième itérations, procédez comme suit :

1. Choisissez l'état de la carte dont vous souhaitez afficher les données d'itération.
2. Dans la visionneuse d'itérations cartographiques, choisissez #2 dans la liste déroulante pour la troisième itération. Cela est dû au fait que les itérations sont comptées à partir de zéro. De même, choisissez #3 dans la liste déroulante pour la quatrième itération de l'état de la carte.

   Vous pouvez également utiliser les
   ▽
   commandes
   △
   et pour passer d'une itération à une autre de l'état de la carte.

   > **Note**
   >
   > Si votre machine à états contient des `Map` états imbriqués, les listes déroulantes pour les itérations d'`Map`états parent et enfant seront affichées comme indiqué dans l'exemple suivant :
   >
   > 
   >
3. (Facultatif) Si une ou plusieurs itérations de votre état de carte ont échoué ou si l'exécution a été arrêtée, vous pouvez consulter ses données en choisissant les numéros d'itération sous Echec ou Abandonné dans la liste déroulante.

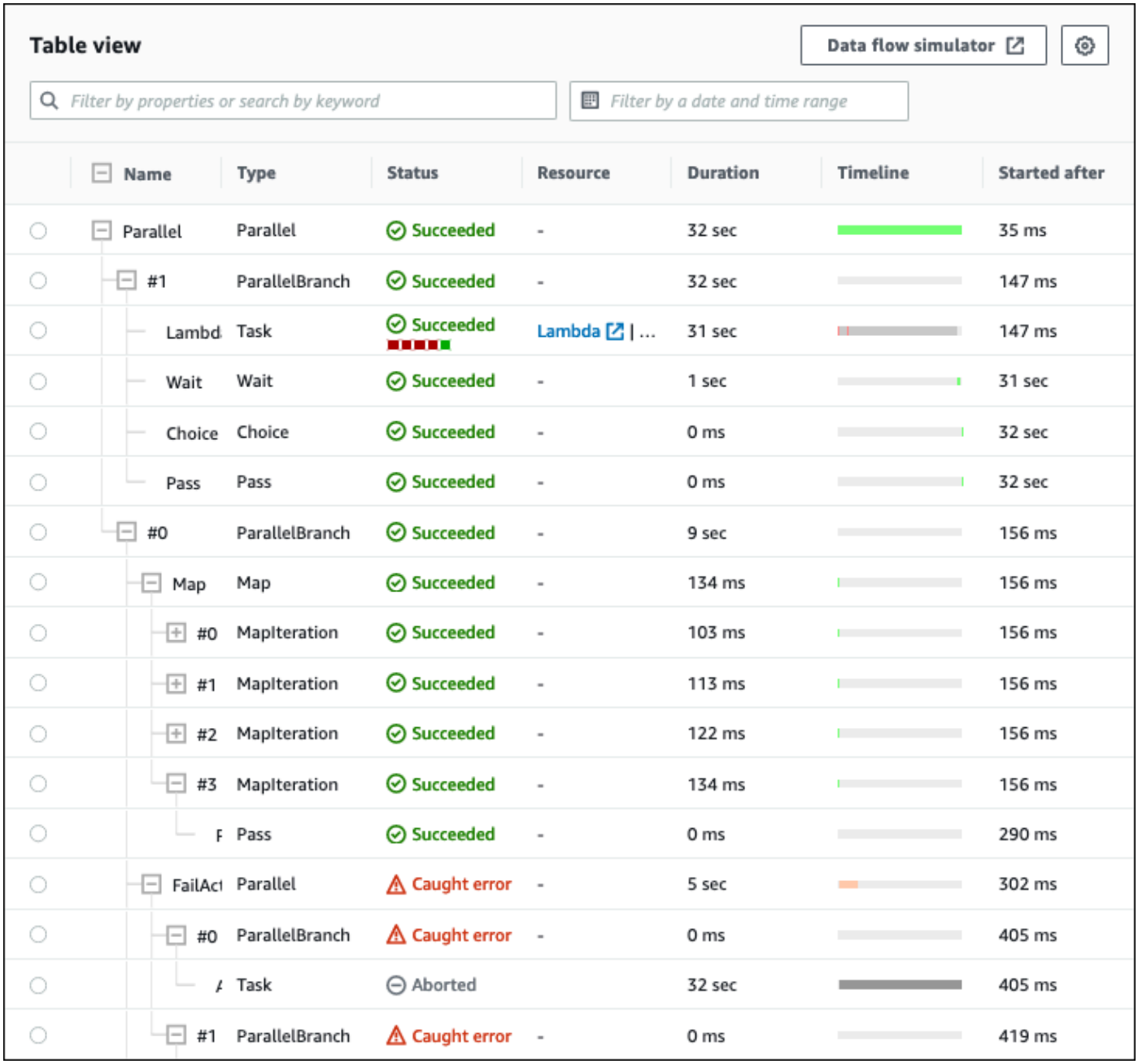
Enfin, vous pouvez utiliser les boutons Exporter et Disposition pour exporter le graphique du flux de travail sous forme d'image SVG ou PNG. Vous pouvez également passer d'une vue horizontale à une vue verticale de votre flux de travail.

Table view

Le mode d'affichage sous forme de tableau affiche une représentation tabulaire des états de votre flux de travail. Dans ce mode d'affichage, vous pouvez voir les détails de chaque état exécuté dans votre flux de travail, y compris son nom, le nom de toute ressource utilisée (telle qu'une AWS Lambda fonction) et si l'état a été exécuté correctement.

Dans cette vue, vous pouvez choisir n'importe quel état de votre flux de travail pour afficher les détails de son exécution dans le composant Détails de l'étape. Lorsque vous avez choisi une étape dans la vue Tableau, la vue Graphique affiche également cette étape. Cela est également vrai dans le sens inverse. Si vous choisissez une étape dans la vue graphique, la vue Tableau affiche la même étape.

Vous pouvez également limiter la quantité de données affichées en mode d'affichage sous forme de tableau en appliquant des filtres à l'affichage. Vous pouvez créer un filtre pour une propriété spécifique, telle que Status ou RedriveTentative. Pour de plus amples informations, veuillez consulter Tutoriel : Examen des exécutions par des machines à états à l'aide de la console Step Functions.

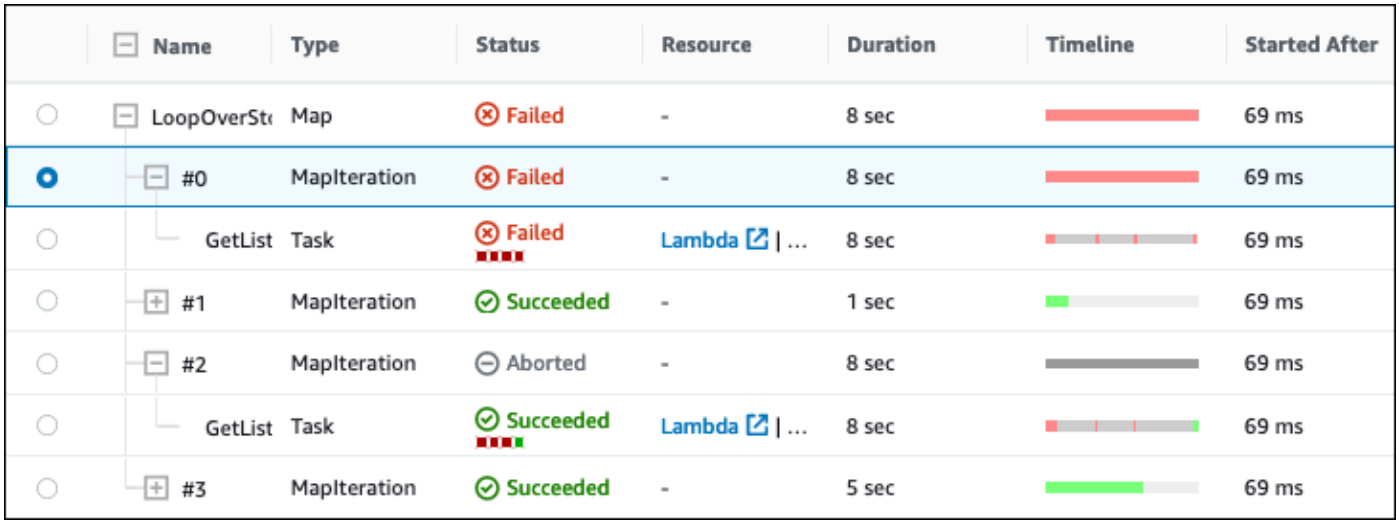


Par défaut, ce mode affiche les colonnes Nom, Type, État, Ressource et Commencé après. Vous pouvez configurer les colonnes que vous souhaitez afficher à l'aide de la boîte de dialogue Préférences. Les sélections que vous effectuez dans cette boîte de dialogue sont conservées pour les futures exécutions par State Machine jusqu'à ce qu'elles soient à nouveau modifiées.

Si vous ajoutez la colonne Chronologie, la durée d'exécution de chaque état est affichée par rapport au temps d'exécution pour l'ensemble de l'exécution. Ceci est affiché sous la forme d'une chronologie linéaire codée par couleur. Cela peut vous aider à identifier les problèmes liés aux performances liés à l'exécution d'un état spécifique. Les segments codés par couleur pour chaque

état de la chronologie vous aident à identifier le statut d'exécution de l'état, par exemple en cours, en échec ou abandonné.

Par exemple, si vous avez défini des tentatives d'exécution pour un état dans votre machine à états, ces tentatives sont affichées dans la chronologie. Les segments rouges représentent les `Retry` tentatives infructueuses, tandis que les segments gris clair représentent les `BackoffRate` intervalles entre chaque `Retry` tentative.



Si votre machine à états contient un `Map` état, un `Parallel` état ou les deux, vous pouvez afficher leurs noms dans le flux de travail en mode Tableau. Pour les `Parallel` états `Map` et, le mode d'affichage sous forme de tableau affiche les données d'exécution de leurs itérations et de leurs branches parallèles sous forme de nœuds dans une vue arborescente. Vous pouvez choisir chaque nœud dans ces états pour afficher ses détails individuels dans la section Détails de l'étape. Par exemple, vous pouvez consulter les données d'une itération d'état de carte spécifique à l'origine de l'échec de l'état. Développez le nœud pour l'état de la carte, puis visualisez le statut de chaque itération dans la colonne État.

Détails de l'étape

La section Détails de l'étape s'ouvre sur la droite lorsque vous choisissez un état dans la vue graphique ou dans la vue Tableau. Cette section contient les onglets suivants, qui fournissent des informations détaillées sur l'état sélectionné :

Entrée

Affiche les détails d'entrée de l'état sélectionné. S'il y a une erreur dans la saisie, elle est indiquée par un en-tête dans

l'⚠ onglet
De plus, vous pouvez voir la raison de l'erreur dans cet onglet.

Vous pouvez également cliquer sur le bouton d'affichage avancé pour voir le chemin de transfert des données d'entrée au fur et à mesure que les données passent par l'état sélectionné. Cela vous permet d'identifier la manière dont votre saisie a été traitée lorsqu'un ou plusieurs champs, tels que `InputPathParameters`,`ResultSelector`,`OutputPath`, et`ResultPath`, ont été appliqués aux données.

Sortie

Affiche le résultat de l'état sélectionné. S'il y a une erreur dans le résultat, elle est indiquée par un en-tête dans
l'⚠ onglet
De plus, vous pouvez voir la raison de l'erreur dans cet onglet.

Vous pouvez également cliquer sur le bouton d'affichage avancé pour voir le chemin de transfert des données de sortie au fur et à mesure que les données passent par l'état sélectionné. Cela vous permet d'identifier la manière dont votre saisie a été traitée lorsqu'un ou plusieurs champs, tels que `InputPathParameters`,`ResultSelector`,`OutputPath`, et`ResultPath`, ont été appliqués aux données.

Détails

Affiche des informations, telles que le type d'état, son statut d'exécution et sa durée d'exécution.

Pour `Task` les États qui utilisent une ressource, par exemple AWS Lambda, cet onglet fournit des liens vers la page de définition de la ressource et la page Amazon CloudWatch Logs pour l'invocation de la ressource. Il affiche également les valeurs, si elles sont spécifiées, pour l'`Task`état `TimeoutSeconds` et `HeartbeatSeconds` les champs.

Pour `Map` les états, cet onglet affiche des informations concernant le nombre total d'itérations d'un `Map` état. Les itérations sont classées dans les catégories suivantes : échec, abandon, réussite ou. InProgress

Définition

Affiche la définition de la langue Amazon States correspondant à l'état sélectionné.

Réessayer

> **Note**
>
> Cet onglet apparaît uniquement si vous avez défini un `Retry` champ dans votre machine à états `Task` ou dans votre `Parallel` état.

Affiche les tentatives de nouvelle tentative initiale et suivantes pour un état sélectionné lors de sa tentative d'exécution initiale. Pour la tentative initiale et toutes les tentatives infructueuses suivantes, cliquez à

côté de Type pour afficher la raison de l'échec qui apparaît dans une liste déroulante. Si la nouvelle tentative réussit, vous pouvez afficher le résultat qui apparaît dans une liste déroulante.

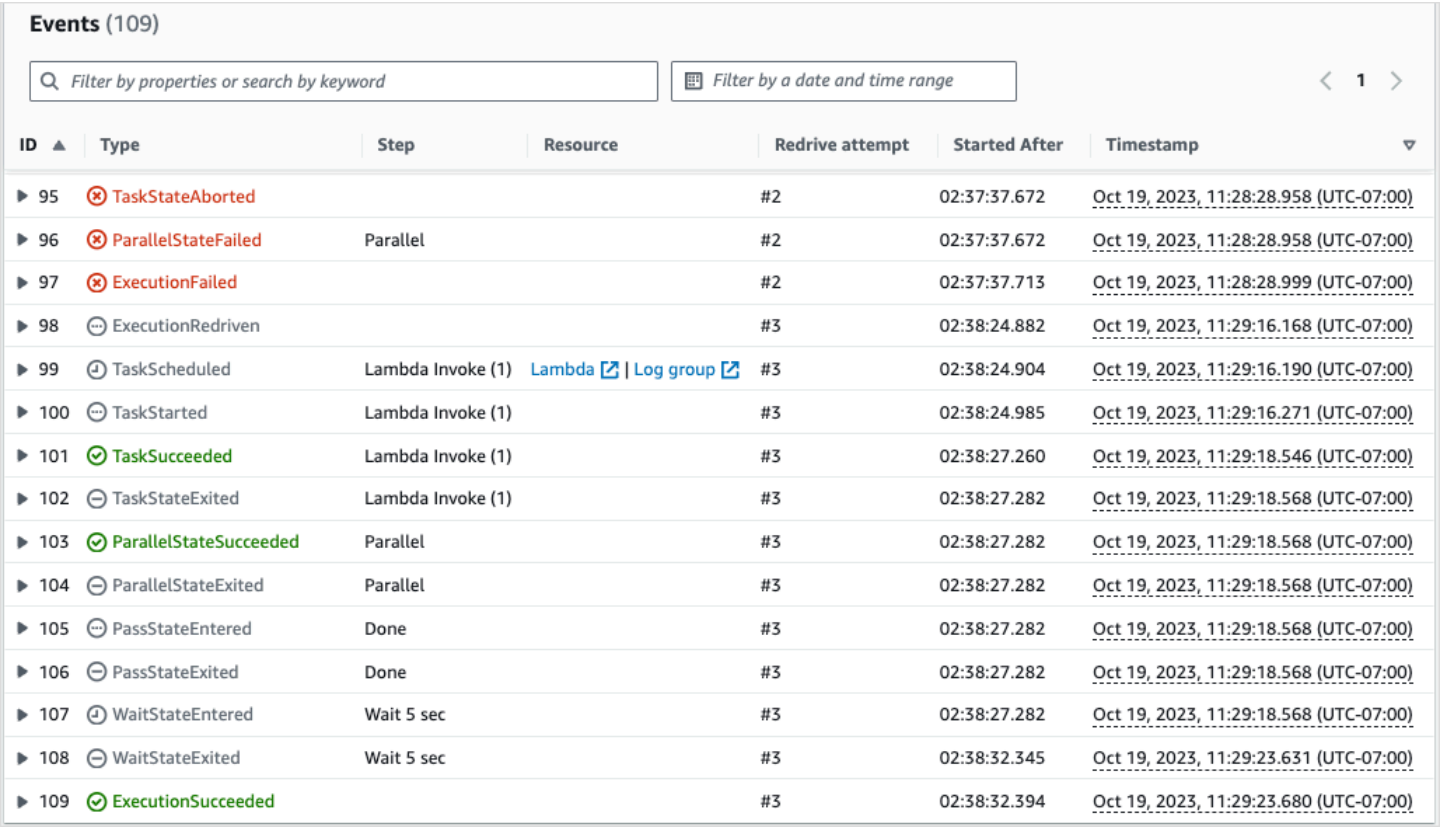
Si redriven votre exécution est terminée, cet en-tête d'onglet affiche le nom Rétentatives redrives et affiche les détails des tentatives pour chacune d'elles. redrive

Evénements

Affiche une liste filtrée des événements associés à l'état sélectionné dans une exécution. Les informations affichées dans cet onglet sont un sous-ensemble de l'historique complet des événements d'exécution que vous pouvez consulter dans le tableau Événements.

Événements

Le tableau Événements affiche l'historique complet de l'exécution sélectionnée sous la forme d'une liste d'événements s'étendant sur plusieurs pages. Chaque page contient jusqu'à 25 événements. Cette section affiche également le nombre total d'événements, ce qui peut vous aider à déterminer si vous avez dépassé le nombre maximum d'événements historiques de 25 000 événements.



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)

Par défaut, les résultats du tableau Événements sont affichés par ordre croissant en fonction de l'horodatage des événements. Vous pouvez modifier le tri de l'historique des événements d'exécution par ordre décroissant en cliquant sur l'en-tête de la colonne Horodatage.

Dans le tableau Événements, chaque événement est codé par couleur pour indiquer son état d'exécution. Par exemple, les événements qui ont échoué apparaissent en rouge. Pour afficher des informations supplémentaires sur un événement, cliquez sur le bouton à

côté de l'ID de l'événement. Une fois ouvert, les détails de l'événement indiquent l'entrée, la sortie et l'appel des ressources pour l'événement.

En outre, dans le tableau Événements, vous pouvez appliquer des filtres pour limiter les résultats de l'historique des événements d'exécution qui sont affichés. Vous pouvez choisir des propriétés telles que l'ID ou la Redrivetentative. Pour de plus amples informations, veuillez consulter Tutoriel : Examen des exécutions par des machines à états à l'aide de la console Step Functions.

Tutoriel : Examen des exécutions par des machines à états à l'aide de la console Step Functions

Dans ce didacticiel, vous allez apprendre à inspecter les informations d'exécution affichées sur la page Détails de l'exécution et à connaître la raison de l'échec de l'exécution. Vous apprendrez ensuite comment accéder aux différentes itérations d'une exécution d'`Map`état. Enfin, vous apprendrez à configurer les colonnes de la vue Tableau et à appliquer les filtres appropriés pour n'afficher que les informations qui vous intéressent.

Dans ce didacticiel, vous allez créer une machine à états de type standard, qui obtient le prix d'un ensemble de fruits. Pour ce faire, la machine à états utilise trois AWS Lambda fonctions qui renvoient une liste aléatoire de quatre fruits, le prix de chaque fruit et le coût moyen des fruits. Les fonctions Lambda sont conçues pour générer une erreur si le prix des fruits est inférieur ou égal à une valeur seuil.

> **Note**
>
> Bien que la procédure suivante contienne des instructions sur la façon d'examiner les détails d'une exécution de flux de travail standard, vous pouvez également examiner les détails des exécutions de flux de travail Express. Pour plus d'informations sur les différences entre les détails d'exécution pour les types de flux de travail Standard et Express, consultezExécutions de flux de travail standard et express dans la console.

Table des matières

Étape 1 : créer et tester les fonctions Lambda requises

1. Ouvrez la console Lambda, puis effectuez les étapes 1 à 4 de la Étape 1 : Créer une fonction Lambda section. Assurez-vous de nommer la fonction Lambda. **`GetListOfFruits`**
2. Après avoir créé votre fonction Lambda, copiez le nom de ressource Amazon (ARN) de la fonction affiché dans le coin supérieur droit de la page. Pour copier l'ARN, cliquez sur

le .

Voici un exemple d'ARN, où *function-name* est le nom de la fonction Lambda (dans ce cas, `GetListOfFruits`) :

```
arn:aws:lambda:us-east-1:123456789012:function:function-name
```

3. Copiez le code suivant pour la fonction Lambda dans la zone Code source de la GetListOfFruitspage.

```
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. Choisissez Deploy, puis choisissez Test, pour déployer les modifications et voir le résultat de votre fonction Lambda.
5. Créez deux fonctions Lambda supplémentaires, nommées **`GetFruitPrice`** et **`CalculateAverage`** respectivement, en procédant comme suit :

   a. Copiez le code suivant dans la zone Source du code de la fonction GetFruitPriceLambda :

```
exports.handler = async function(event, context) {

    const errorChance = 0;
    const waitTime = Math.floor( 100 * Math.random() );

    await new Promise( r => setTimeout(() => r(), waitTime));

    const num = Math.floor( 100 * Math.random() );
    if (num <= errorChance) {
        throw(new Error('Error'));
    }

    return Math.floor(Math.random()*100)/10;
};
```

   b. Copiez le code suivant dans la zone Source du code de la fonction CalculateAverageLambda :

```
function getRandomSubarray(arr, size) {
    var shuffled = arr.slice(0), i = arr.length, temp, index;
    while (i--) {
        index = Math.floor((i + 1) * Math.random());
        temp = shuffled[index];
        shuffled[index] = shuffled[i];
        shuffled[i] = temp;
    }
    return shuffled.slice(0, size);
}

const average = arr => arr.reduce( ( p, c ) => p + c, 0 ) / arr.length;

exports.handler = async function(event, context) {
        const errors = [
        "Error getting data from DynamoDB",
        "Error connecting to DynamoDB",
```

```
        "Network error",
        "MemoryError - Low memory"
        ]

    const errorChance = 0;

    const waitTime = Math.floor( 100 * Math.random() );

    await new Promise( r => setTimeout(() => r(), waitTime));

    const num = Math.floor( 100 * Math.random() );
    if (num <= errorChance) {
        throw(new Error(getRandomSubarray(errors, 1)[0]));
    }

    return average(event);
};
```

c. Assurez-vous de copier les ARN de ces deux fonctions Lambda, puis de les déployer et de les tester.

Étape 2 : Création et exécution de la machine à états

Utilisez la console Step Functions pour créer une machine à états qui invoque les fonctions Lambda que vous avez créées à l'étape 1. Dans cette machine à états, trois `Map` états sont définis. Chacun de ces `Map` états contient un `Task` état qui invoque l'une de vos fonctions Lambda. En outre, un `Retry` champ est défini dans chaque `Task` État avec un nombre de tentatives de nouvelle tentative défini pour chaque État. Si un `Task` état rencontre une erreur d'exécution, il est exécuté à nouveau dans la limite du nombre de tentatives défini pour cela`Task`.

1. Ouvrez la console Step Functions et choisissez Write your workflow in code.

> **Important**
>
> Assurez-vous que votre machine d'état se trouve sous le même AWS compte et dans la même région que la fonction Lambda que vous avez créée précédemment.

2. Pour Type, conservez la sélection par défaut Standard.

3. Copiez la définition Amazon States Language suivante et collez-la sous Definition. Assurez-vous de remplacer les ARN affichés par ceux des fonctions Lambda que vous avez créées précédemment.

```
{
    "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. Entrez un nom pour votre machine à états. Conservez les sélections par défaut pour les autres options de cette page et choisissez Create state machine.
5. Ouvrez la page intitulée avec le nom de votre machine à états. Effectuez les étapes 1 à 4 de la Étape 4 : Exécutez la machine d'état section, mais utilisez les données suivantes comme entrée d'exécution :

```
{
    "stores": [
      "Store A",
      "Store B",
      "Store C",
      "Store D"
    ]
}
```

Étape 3 : Afficher les détails de l'exécution de la machine à états

Sur la page intitulée avec votre identifiant d'exécution, vous pouvez consulter les résultats de votre exécution et corriger les erreurs éventuelles.

1. (Facultatif) Choisissez l'un des onglets affichés sur la page Détails de l'exécution pour voir les informations présentes dans chacun d'eux. Par exemple, pour afficher l'entrée de la machine à états et sa sortie d'exécution, choisissez Entrée et sortie d'exécution dans la section Résumé de l'exécution.
2. Si l'exécution de votre machine à états a échoué, choisissez Cause ou Afficher le détail de l'étape dans le message d'erreur. Les détails de l'erreur sont affichés dans la section Détails de l'étape. Notez que l'étape à l'origine de l'erreur, à savoir un `Task` état nommé GetListofFruits, est surlignée dans la vue graphique et dans la vue Tableau.

> **Note**
>
> Étant donné que l'GetListofFruitsétape est définie dans un **`Map`** état et qu'elle n'a pas réussi à s'exécuter, l'étape **`Map`** État de l'état est affichée comme ayant échoué.

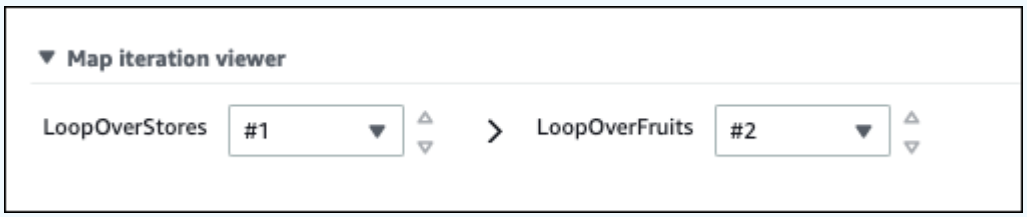
Étape 4 : Explorez les différents modes d'affichage

Vous pouvez choisir un mode préféré pour afficher le flux de travail de la machine à états ou l'historique des événements d'exécution. Certaines des tâches que vous pouvez effectuer dans ces modes d'affichage sont les suivantes :

Affichage graphique — Basculer entre les différentes itérations **`Map`** d'état

Si l'état de votre carte comporte cinq itérations et que vous souhaitez consulter les détails d'exécution des troisième et quatrième itérations, procédez comme suit :

1. Choisissez l'`Map`état pour lequel vous souhaitez afficher les données d'itération.
2. Dans le visualiseur d'itérations cartographiques, choisissez l'itération que vous souhaitez afficher. Les itérations sont comptées à partir de zéro. Pour choisir la troisième itération sur cinq, choisissez #2 dans la liste déroulante à côté du nom de l'état de la carte.

> **Note**
>
> Si votre machine à états contient des `Map` états imbriqués, Step Functions affiche les itérations des `Map` états parent et enfant sous forme de deux listes déroulantes distinctes :
>
> 
>

3. (Facultatif) Si une ou plusieurs de vos itérations d'`Map`état n'ont pas pu être exécutées ou ont été arrêtées alors qu'elles étaient abandonnées, vous pouvez consulter les détails de l'itération échouée. Pour voir ces détails, choisissez les numéros d'itération concernés sous Echec ou Abandonné dans la liste déroulante.

Affichage sous forme de tableau — Basculer entre les différentes itérations **`Map`** d'état

Si l'état de votre carte comporte cinq itérations et que vous souhaitez consulter les détails d'exécution des itérations numéro trois et quatre, procédez comme suit :

1. Choisissez l'`Map`état pour lequel vous souhaitez afficher les différentes données d'itération.
2. Dans l'affichage en arborescence des itérations d'`Map`état, choisissez la ligne d'itération nommée #2 pour l'itération numéro trois. De même, choisissez la ligne nommée #3 pour l'itération numéro quatre.

Affichage sous forme de tableau : configurez les colonnes à afficher

Sélectionnez

.

Ensuite, dans la boîte de dialogue Préférences, choisissez les colonnes que vous souhaitez afficher sous Sélectionner les colonnes visibles.

Par défaut, ce mode affiche les colonnes Nom, Type, État, Ressource et Commencé après.

Affichage sous forme de tableau — Filtrez les résultats

Limitez la quantité d'informations affichées en appliquant un ou plusieurs filtres basés sur une propriété, telle que le statut, ou une plage de dates et d'heures. Par exemple, pour afficher les étapes dont l'exécution a échoué, appliquez le filtre suivant :

1. Choisissez Filtrer par propriétés ou rechercher par mot clé, puis sélectionnez État sous Propriétés.
2. Sous Opérateurs, sélectionnez Status =.
3. Choisissez Status = Echec.
4. (Facultatif) Choisissez Effacer les filtres pour supprimer les filtres appliqués.

Affichage des événements — Filtrez les résultats

Limitez la quantité d'informations affichées en appliquant un ou plusieurs filtres basés sur une propriété, telle que le type, ou une plage de dates et d'heures. Par exemple, pour afficher les étapes `Task` d'état dont l'exécution a échoué, appliquez le filtre suivant :

1. Choisissez Filtrer par propriétés ou rechercher par mot clé, puis sélectionnez Type sous Propriétés.
2. Sous Opérateurs, sélectionnez Type =.
3. Choisissez Type = TaskFailed.
4. (Facultatif) Choisissez Effacer les filtres pour supprimer les filtres appliqués.

Vue des événements — Inspecter les détails d'un TaskFailedévénement

Cliquez à

côté de l'ID d'un TaskFailedévénement pour consulter ses détails, y compris les entrées, les sorties et l'appel de ressources qui apparaissent dans une liste déroulante.

Redrivingexécutions

Vous pouvez l'utiliser redrive pour redémarrer les exécutions de flux de travail standard qui ne se sont pas terminées correctement au cours des 14 derniers jours. Il s'agit notamment des exécutions échouées, abandonnées ou dont le délai imparti a expiré.

Lorsque vous effectuez redrive une exécution, elle poursuit l'exécution échouée à partir de l'étape infructueuse et utilise la même entrée. Step Functionspréserve les résultats et l'historique d'exécution

des étapes réussies, et ces étapes ne sont pas réexécutées lors redrive d'une exécution. Supposons, par exemple, que votre flux de travail contienne deux états : un Pass état suivi d'un État de la tâche état. Si l'exécution de votre flux de travail échoue à l'état Tâche et que vous redrive l'exécutez, l'exécution replanifie puis réexécute l'état Tâche.

Redrivenles exécutions utilisent la même définition de machine à états et le même ARN d'exécution que ceux utilisés lors de la tentative d'exécution initiale. Si votre tentative d'exécution initiale était associée à une version, à un alias ou aux deux, l'redrivenexécution est associée à la même version, au même alias, ou aux deux. Même si vous mettez à jour votre alias pour qu'il pointe vers une autre version, l'redrivenexécution continue d'utiliser la version associée à la tentative d'exécution initiale. Étant donné que les redriven exécutions utilisent la même définition de machine à états, vous devez démarrer une nouvelle exécution si vous mettez à jour la définition de votre machine à états.

Lors redrive d'une exécution, le délai d'expiration au niveau de la machine à états, s'il est défini, est remis à 0. Pour plus d'informations sur le délai d'expiration au niveau de la machine à états, consultez`TimeoutSeconds`.

redrivesLes exécutions sont considérées comme des transitions d'état. Pour plus d'informations sur l'impact des transitions entre États sur la facturation, consultez Step Functions Pricing.

Rubriques

- Redriveéligibilité en cas d'exécution infructueuse
- Redrivecomportement des différents États
- Autorisation IAM pour redrive une exécution
- Redrivingexécutions dans la console
- Redrivingexécutions à l'aide de l'API
- Examen des redriven exécutions
- Réessayer le comportement des exécutions redriven

Redriveéligibilité en cas d'exécution infructueuse

Vous pouvez exécuter redrive des exécutions si votre tentative d'exécution initiale répond aux conditions suivantes :

- Vous avez commencé l'exécution le 15 novembre 2023 ou après cette date. Les exécutions que vous avez commencées avant cette date ne sont pas éligiblesredrive.
- Le statut d'exécution ne l'est pas`SUCCEEDED`.

- L'exécution du flux de travail n'a pas dépassé le redrivable délai de 14 jours. RedrivableLa période fait référence au temps pendant lequel vous pouvez redrive effectuer une exécution donnée. Cette période commence le jour où une machine d'État termine son exécution.
- L'exécution du flux de travail n'a pas dépassé la durée d'ouverture maximale d'un an. Pour plus d'informations sur les quotas d'exécution des machines à états, consultezQuotas liés aux exécutions par les machines de l'État.
- L'historique des événements d'exécution est inférieur à 24 999. Redrivenles exécutions ajoutent leur historique des événements à l'historique des événements existant. Assurez-vous que l'exécution de votre flux de travail contient moins de 24 999 événements pour tenir compte de `ExecutionRedriven` l'événement historique et d'au moins un autre événement historique.

Redrivecomportement des différents États

En fonction de l'état qui a échoué dans votre flux de travail, le redrive comportement de tous les états d'échec varie. Le tableau suivant décrit le redrive comportement de tous les états.

Nom de l'État	Redrivecomportement d'exécution
Pass	Si une étape précédente échoue ou si le délai d'expiration de la machine à états expire, l'état Pass est quitté et n'est pas exécuté. redrive
État de la tâche	Planifie et recommence l'état de la tâche. Lorsque vous redrive effectuez une exécution qui réexécute un état de tâche, l'état `TimeoutSeconds` correspondant, s'il est défini, est remis à 0. Pour plus d'informations sur le délai d'expiration, consultez la section État des tâches.
Choice	Réévalue les règles de l'état Choice.
Attente	Si l'état indique `Timestamp` ou `Timestamp Path` fait référence à un horodatage antérieur, redrive provoque la sortie de l'état Wait et entre dans l'état spécifié dans le champ. `Next`

Nom de l'État	Redrivecomportement d'exécution
Succeed	N'indique pas redrive les exécutions automatiq ues qui passent à l'état Succeed.
Fail	Repasse à l'état Fail et échoue à nouveau.
Parallèle	Replanifie et redrives uniquement les branches qui ont échoué ou ont été abandonnées. Si l'état a échoué en raison d'une `States.DataLimitExceeded` erreur, l'état parallèle est réexécuté, y compris les branches qui ont réussi lors de la tentative d'exécution initiale.
État de la carte en ligne	Replanifie et redrives uniquement les itérations qui ont échoué ou ont été abandonnées. Si l'état a échoué en raison d'une `States.DataLimitExceeded` erreur, l'état Inline Map est réexécuté, y compris les itérations réussies lors de la tentative d'exécution initiale.
État de la carte distribuée	redrivesles exécutions infructueuses du flux de travail enfant dans un Map Run. Pour de plus amples informations, veuillez consulter RedrivingCartes parcourues. Si l'état a échoué en raison d'une `States.DataLimitExceeded` erreur, l'état de la carte distribuée est réexécuté. Cela inclut les flux de travail enfants qui ont réussi lors de la tentative d'exécution initiale.

Autorisation IAM pour redrive une exécution

Step Functions a besoin d'une autorisation appropriée pour redrive une exécution. L'exemple de politique IAM suivant accorde le minimum de privilèges requis à votre machine d'état pour redriving

une exécution. N'oubliez pas de remplacer le texte *en italique* par les informations spécifiques à votre ressource.

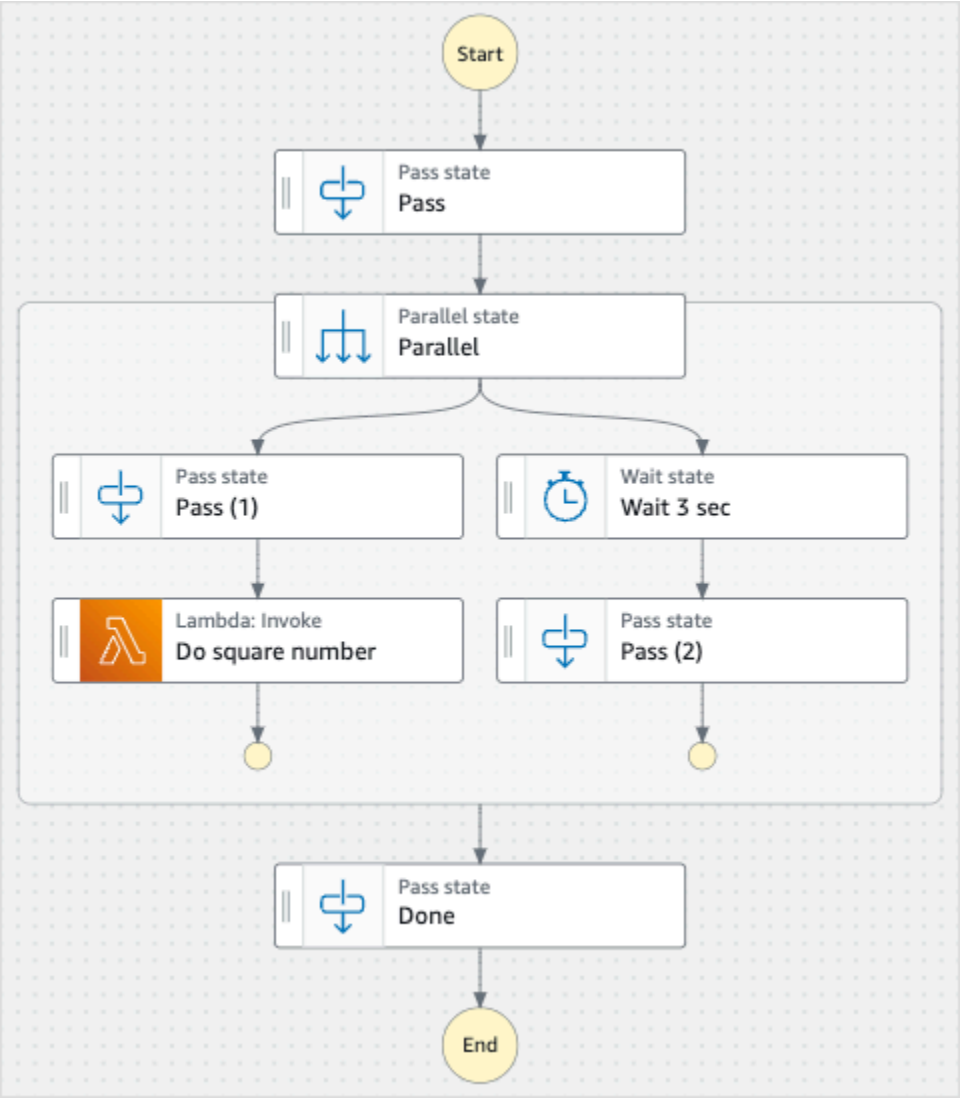
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "states:RedriveExecution"
            ],
            "Resource": "arn:aws:states:us-
east-2:123456789012:execution:myStateMachine:*"
        }
    ]
}
```

Pour un exemple de l'autorisation dont vous avez besoin pour exécuter redrive une carte, consultezExemple de politique IAM pour redriving une carte distribuée.

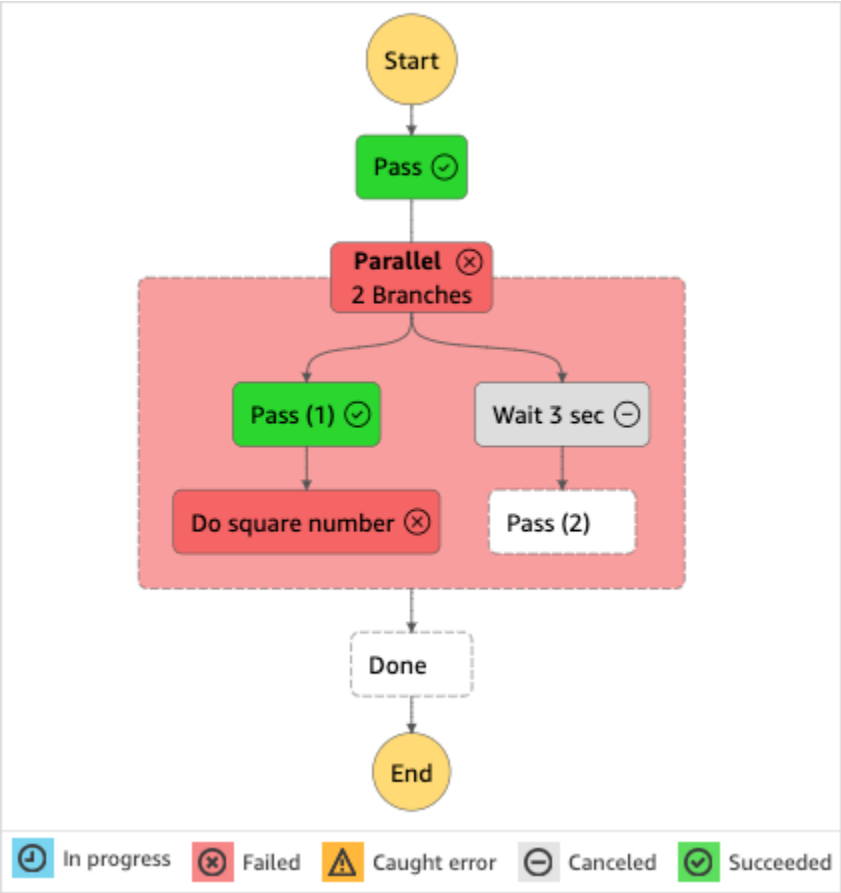
Redrivingexécutions dans la console

Vous pouvez redrive sélectionner des exécutions depuis la Step Functions console.

Supposons, par exemple, que l'image suivante représente le graphe du flux de travail de votre machine à états.



Imaginez que vous dirigez cette machine d'État. L'image suivante montre le graphique de l'exécution de la machine à états.



Comme le montre cette image, l'étape LambdaInvoke nommée Do square number inside the Parallel state a renvoyé une erreur. Cela a provoqué l'échec de l'état parallèle. Les branches dont l'exécution était en cours ou non démarrée sont arrêtées et l'exécution de la machine d'état échoue.

Vers redrive une exécution depuis la console

1. Ouvrez la console Step Functions, puis choisissez une machine à états existante dont l'exécution a échoué.
2. Sur la page détaillée de la machine d'état, sous Exécutions, choisissez une instance d'exécution ayant échoué.
3. Sélectionnez Redrive.
4. Dans la Redriveboîte de dialogue, sélectionnez RedriveExécution.

> **Tip**
>
> Si vous êtes sur la page Détails de l'exécution d'une exécution qui a échoué, effectuez l'une des opérations suivantes pour redrive l'exécution :

- Choisissez Restaurer, puis sélectionnez En cas Redrive d'échec.
- Choisissez Actions, puis sélectionnez Redrive.

Notez qu'il redrive utilise la même définition de machine à états et le même ARN. Il poursuit l'exécution à partir de l'étape qui a échoué lors de la tentative d'exécution initiale. Dans cet exemple, il s'agit de l'étape Do square number et de la branche Wait 3 sec dans l'état Parallel. Après avoir redémarré l'exécution de ces étapes infructueuses à l'état parallèle, l'exécution de l'étape Terminé se redrive poursuivra.

5. Choisissez l'exécution pour ouvrir la page Détails de l'exécution.

   Sur cette page, vous pouvez consulter les résultats de l'redrivenexécution. Par exemple, dans la Résumé de l'exécution section, vous pouvez voir le Redrivenombre, qui représente le nombre de fois qu'une exécution a eu lieuredriven. Dans la section Événements, vous pouvez voir les événements d'exécution redrive associés ajoutés aux événements de la tentative d'exécution initiale. Par exemple, l'`ExecutionRedrive`événement.

Redrivingexécutions à l'aide de l'API

Vous pouvez sélectionner redrive des exécutions à l'aide de l'RedriveExecutionAPI. Cette API redémarre les exécutions infructueuses des flux de travail standard à partir de l'étape qui a échoué, qui a été interrompue ou qui a expiré.

Dans le AWS Command Line Interface (AWS CLI), exécutez la commande suivante en cas redrive d'échec de l'exécution de la machine à états. N'oubliez pas de remplacer le texte *`en italique`* par les informations spécifiques à votre ressource.

```
aws stepfunctions redrive-execution --execution-arn arn:aws:states:us-
east-2:123456789012:execution:myStateMachine:foo
```

Examen des redriven exécutions

Vous pouvez examiner une redriven exécution dans la console ou à l'aide des API : GetExecutionHistoryet DescribeExecution.

Examiner redriven les exécutions sur console

1. Ouvrez la console Step Functions, puis choisissez une machine à états existante pour laquelle vous avez redriven une exécution.
2. Ouvrez la page Détails de l'exécution.

   Sur cette page, vous pouvez consulter les résultats de l'redrivenexécution. Par exemple, dans la Résumé de l'exécution section, vous pouvez voir le Redrivenombre, qui représente le nombre de fois qu'une exécution a eu lieuredriven. Dans la section Événements, vous pouvez voir les événements d'exécution redrive associés ajoutés aux événements de la tentative d'exécution initiale. Par exemple, l'`ExecutionRedrivenévénement`.

Examinez les redriven exécutions à l'aide d'API

Si vous avez redriven une exécution par machine à états, vous pouvez utiliser l'une des API suivantes pour afficher les détails de l'redrivenexécution. N'oubliez pas de remplacer le texte *en italique* par les informations spécifiques à votre ressource.

- GetExecutionHistory — Renvoie l'historique de l'exécution spécifiée sous forme de liste d'événements. Cette API renvoie également les informations relatives à la redrive tentative d'exécution, si elles sont disponibles.

  Dans le AWS CLI, exécutez la commande suivante.

  ```
  aws stepfunctions get-execution-history --execution-arn arn:aws:states:us-
  east-2:123456789012:execution:myStateMachine:foo
  ```

- DescribeExecution — Fournit des informations sur l'exécution d'une machine à états. Il peut s'agir de la machine d'état associée à l'exécution, des entrées et sorties d'exécution, des redrive détails de l'exécution, le cas échéant, et des métadonnées d'exécution pertinentes.

  Dans le AWS CLI, exécutez la commande suivante.

  ```
  aws stepfunctions describe-execution --execution-arn arn:aws:states:us-
  east-2:123456789012:execution:myStateMachine:foo
  ```

Réessayer le comportement des exécutions redriven

Si votre redriven exécution réexécute un état État de la tâcheParallèle, ou Inline Map, pour lequel vous avez défini de nouvelles tentatives, le nombre de tentatives pour ces états est remis à 0. Cela permet de maximiser le nombre de tentativesredrive. Pour une redriven exécution, vous pouvez suivre les tentatives de tentative individuelles de ces états à l'aide de la console.

Pour examiner les différentes tentatives de tentative dans la console

1. Sur la page Détails de l'exécution de la console Step Functions, choisissez un état qui a été réessayé. redrive
2. Cliquez sur l'redrivesonglet Rétentatives et.
3. Cliquez sur la
   ▶
   case à côté de chaque nouvelle tentative pour en afficher les détails. Si la nouvelle tentative a réussi, vous pouvez consulter les résultats dans la section Sortie qui apparaît dans une liste déroulante.

L'image suivante montre un exemple des nouvelles tentatives effectuées pour un état lors de la tentative d'exécution initiale et redrives de cette exécution. Dans cette image, trois nouvelles tentatives sont effectuées lors des tentatives d'origine et d'redriveexécution. L'exécution réussit à la quatrième redrive tentative et renvoie une sortie de 16.

Input | Output | Details | Definition | **Retries & redrives** | Events

Type	Status	Resource	Duration	Time
▶ Original execution	Failed	Logs | Lambda | Log group	00:00:00.151	
▶ Retry	Failed	Logs | Lambda | Log group	00:00:00.139	
▶ Retry	Failed	Logs | Lambda | Log group	00:00:00.164	
▶ Retry	Failed	Logs | Lambda | Log group	00:00:00.149	
▶ Redrive #1	Failed	Logs | Lambda | Log group	00:00:00.187	
▶ Retry	Failed	Logs | Lambda | Log group	00:00:00.147	
▶ Retry	Failed	Logs | Lambda | Log group	00:00:00.154	
▶ Retry	Failed	Logs | Lambda | Log group	00:00:00.170	
▶ Redrive #2	Failed	Logs | Lambda | Log group	00:00:00.206	
▶ Retry	Failed	Logs | Lambda | Log group	00:00:00.184	
▶ Retry	Failed	Logs | Lambda | Log group	00:00:00.188	
▶ Retry	Failed	Logs | Lambda | Log group	00:00:00.219	
▶ Redrive #3	Failed	Logs | Lambda | Log group	00:00:00.198	
▶ Retry	Failed	Logs | Lambda | Log group	00:00:00.142	
▶ Retry	Failed	Logs | Lambda | Log group	00:00:00.174	
▶ Retry	Failed	Logs | Lambda | Log group	00:00:00.208	
▼ Redrive #4	Succeeded	Logs | Lambda | Log group	00:00:00.195	

Output Learn more

```
1 ▾ {
2     "Squared": 16
3   }
```

Formatted

Examen de l'exécution cartographique d'une exécution de l'état d'une carte distribuée

Lorsque vous exécutez un `Map` état en mode distribué, Step Functions crée une ressource Map Run. Une exécution de carte fait référence à un ensemble d'exécutions de flux de travail enfants lancées par un état de carte distribuée, ainsi qu'aux paramètres d'exécution qui contrôlent ces exécutions. Step Functions attribue un Amazon Resource Name (ARN) à votre Map Run. Vous pouvez examiner un Map Run dans la console Step Functions. Vous pouvez également invoquer l'action d'`DescribeMapRun`API. Un Map Run envoie également des métriques à. CloudWatch

La console Step Functions fournit une page Map Run Details qui affiche toutes les informations relatives à l'exécution d'un état de carte distribuée. Par exemple, vous pouvez consulter le statut de l'exécution de l'état de la carte distribuée, l'ARN de la carte et le statut des éléments traités dans les exécutions du flux de travail enfant lancées par l'état de la carte distribuée. Vous pouvez également consulter la liste de toutes les exécutions de flux de travail enfants et accéder à leurs détails. De plus, si c'était le cas de votre Map Run redriven, vous pouvez consulter les redrive détails de cette exécution dans la Résumé de l'exécution de Map Run section. Par exemple, La dernière redrive fois. La console affiche ces informations sous forme de tableau de bord.

La page Map Run Details contient les sections suivantes :

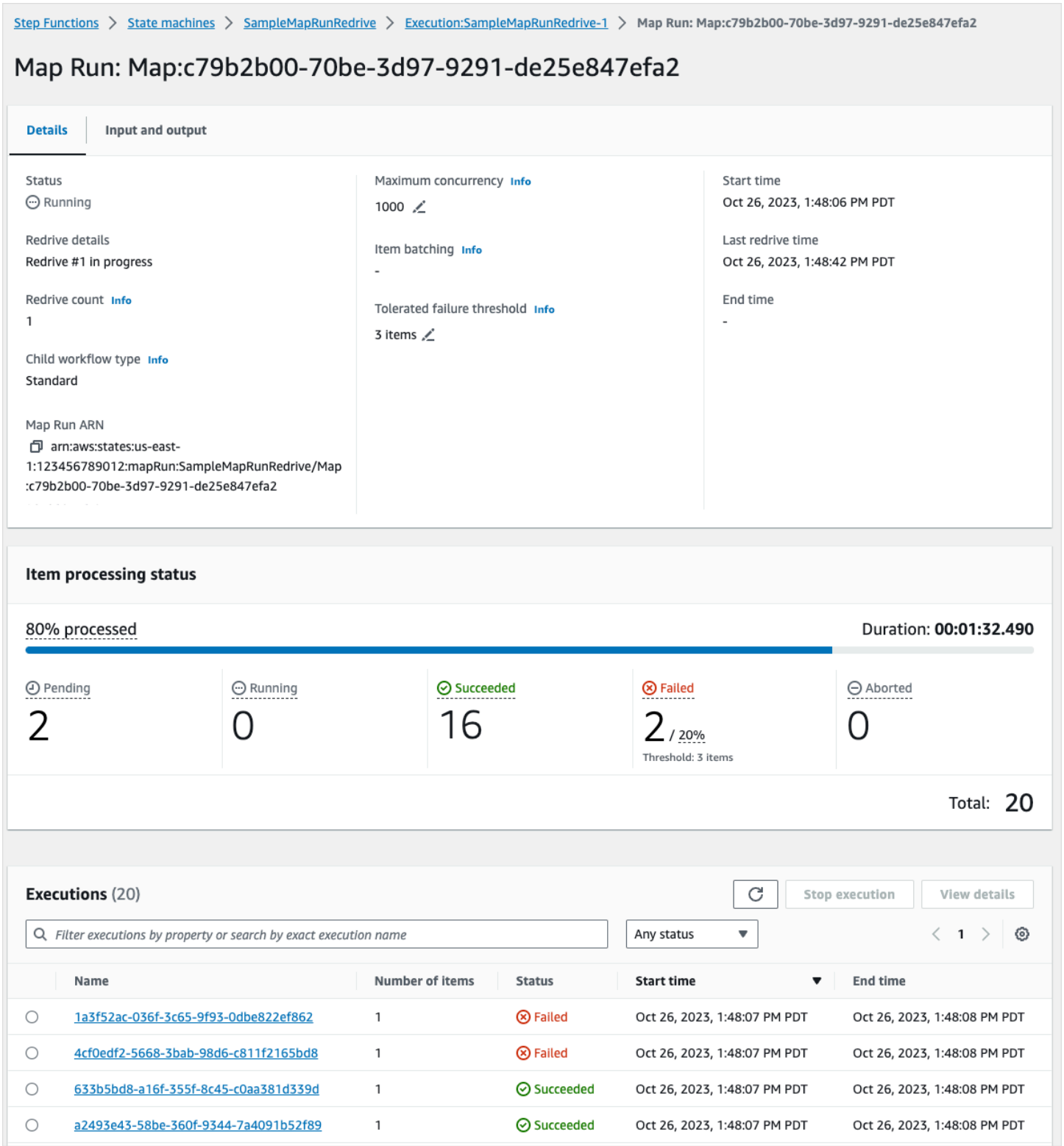


Table des matières

- État du traitement des articles
- Liste des exécutions
- RedrivingCartes parcourues
  - Redriveéligibilité aux flux de travail pour enfants dans un Map Run
  - redriveComportement d'exécution du workflow par enfant
  - Scénarios de saisie utilisés sur Map Run redrive
  - Autorisation IAM d'exécuter redrive une carte
  - RedrivingMap Run dans la console
  - RedrivingMap Run à l'aide de l'API

Résumé de l'exécution de Map Run

La section récapitulative de l'exécution du Map Run apparaît en haut de la page des détails du Map Run. Cette section fournit une vue d'ensemble des détails d'exécution de l'état de la carte distribuée. Ces informations sont réparties entre les onglets suivants :

Détails

Affiche des informations, telles que l'état d'exécution de l'état de la carte distribuée, l'ARN de la carte distribuée et le type des exécutions du flux de travail enfant lancées par l'état de la carte distribuée. Vous pouvez consulter des configurations supplémentaires, telles que le seuil d'échec toléré pour le Map Run et la simultanéité maximale spécifiée pour les exécutions de flux de travail enfants. Vous pouvez également modifier ces configurations.

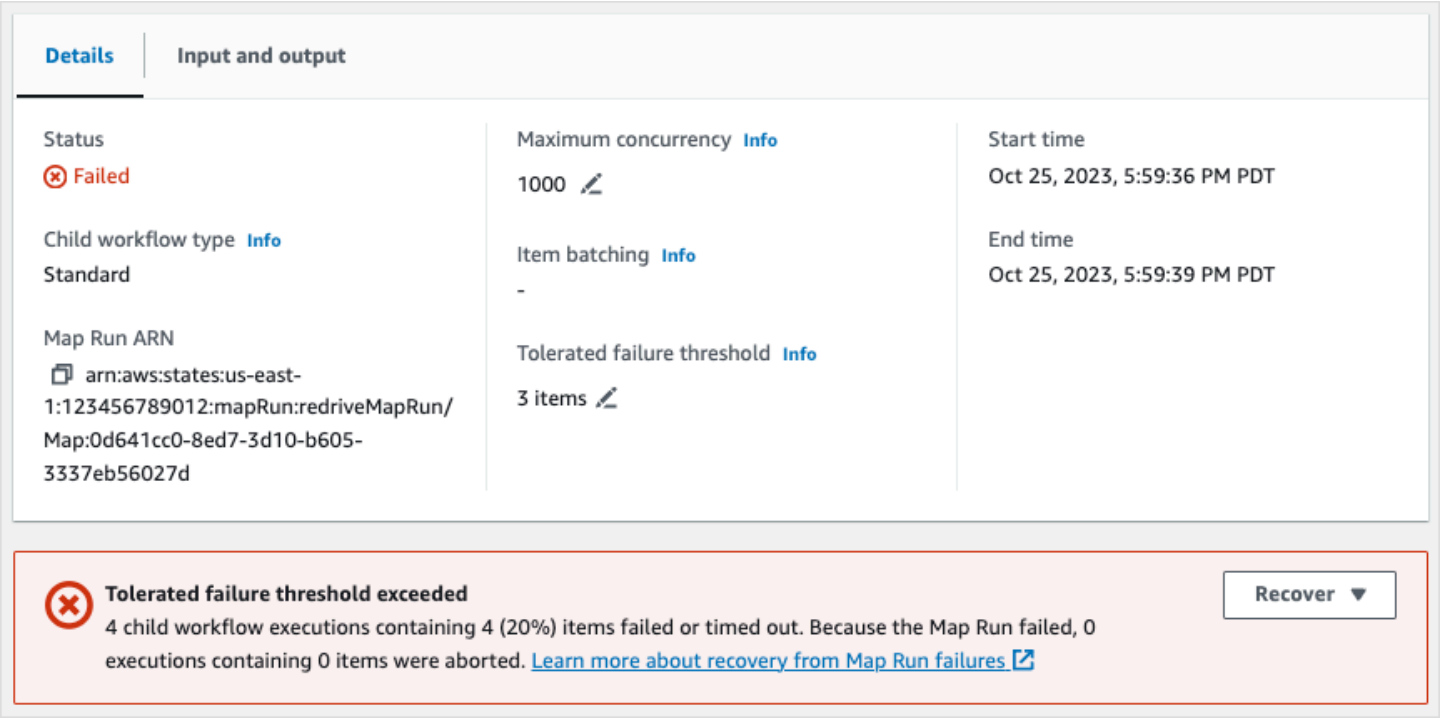
Entrée et sortie

Affiche l'entrée reçue par l'état de la carte distribuée et la sortie correspondante qu'il génère. Par exemple, vous pouvez afficher le jeu de données en entrée et son emplacement, ainsi que les filtres d'entrée appliqués aux éléments de données individuels de ce jeu de données. Si vous exportez le résultat de l'exécution de l'état de la carte distribuée, cet onglet indique le chemin d'accès au compartiment Amazon S3 qui contient les résultats de l'exécution. Sinon, il vous dirige vers la page Détails d'exécution du flux de travail parent pour afficher le résultat de l'exécution.

Error message (Message d’erreur)

Si votre exécution de carte a échoué, la page Détails de l'exécution de la carte affiche un message d'erreur indiquant la raison de l'échec.

À partir du bouton déroulant Restaurer de ce message d'erreur, vous pouvez soit annuler redrive les exécutions infructueuses du flux de travail enfant lancées par cette exécution de carte, soit démarrer une nouvelle exécution du flux de travail parent. Pour de plus amples informations, veuillez consulter RedrivingCartes parcourues.



État du traitement des articles

La section État du traitement des éléments affiche l'état des éléments traités lors d'une exécution cartographique. Par exemple, Pending indique que l'exécution d'un flux de travail enfant n'a pas encore commencé à traiter l'élément.

Le statut des éléments dépend de l'état des exécutions du flux de travail enfant traitant les éléments. Si l'exécution d'un flux de travail enfant échoue, expire ou si un utilisateur annule l'exécution, Step Functions ne reçoit aucune information sur le résultat du traitement des éléments contenus dans l'exécution de ce flux de travail enfant. Tous les éléments traités par cette exécution partagent le même statut que l'exécution du flux de travail enfant.

Supposons, par exemple, que vous souhaitiez traiter 100 éléments dans le cadre de deux exécutions de flux de travail secondaires, chaque exécution traitant un lot de 50 éléments. Si l'une des exécutions échoue et que l'autre réussit, vous aurez 50 objets réussis et 50 objets échoués.

Le tableau suivant explique les types de statuts de traitement disponibles pour tous les articles :

État	Description
En suspens	Indique un élément que l'exécution du flux de travail enfant n'a pas encore commencé à traiter. Si une exécution de carte s'arrête, échoue ou si un utilisateur annule l'exécuti on avant le début du traitement d'un élément, l'élément reste en attente. Par exemple, si une exécution de carte échoue alors que 10 éléments sont en attente de traitement, ces 10 éléments restent dans le statut En attente.
En cours d'exécution	Indique un élément en cours de traitement par l'exécution du flux de travail enfant.
Réussi	Indique que l'exécution du flux de travail enfant a correctement traité l'élément. L'exécution réussie d'un flux de travail enfant ne peut entraîner aucun échec. Si un élément de l'ensemble de données échoue pendant l'exécution, l'exécution complète du flux de travail enfant échoue.
Échec	Indique que l'exécution du flux de travail enfant n'a pas réussi à traiter l'élément ou que le délai d'exécution a expiré. Si l'exécution d'un élément traité par un flux de travail enfant échoue, l'exécution complète du flux de travail enfant échoue. Imaginons, par exemple, l'exécution d'un flux de travail enfant qui a traité 1 000 éléments. Si un élément de cet ensemble de données échoue pendant l'exécution, Step Functions

État	Description
	considère que l'exécution complète du flux de travail enfant a échoué. Lorsque vous lancez redriveune carte, le nombre d'objets ayant ce statut est remis à 0.
Annulé	Indique que l'exécution du flux de travail enfant a commencé à traiter l'élément, mais que soit l'utilisateur a annulé l'exécution, soit Step Functions l'a arrêtée en raison de l'échec du Map Run. Prenons l'exemple d'une exécution de flux de travail Running Child qui traite 50 éléments. Si l'exécution de la carte s'arrête en raison d'un échec ou parce qu'un utilisateur a annulé l'exécution, l'exécution du flux de travail enfant et le statut des 50 éléments passent à Abandonné. Si vous utilisez une exécution de flux de travail enfant de type Express, vous ne pouvez pas arrêter l'exécution. Lorsque vous lancez redriveune exécution de flux de travail enfant de type Express sur une carte, le nombre d'éléments présentant ce statut est remis à 0. Cela est dû au fait que les flux de travail Express Child sont redémarrés à l'aide de l'action de l'StartExecutionAPI au lieu de l'êtreredriven.

Liste des exécutions

La section Exécutions répertorie toutes les exécutions de flux de travail enfants pour un Map Run spécifique. Utilisez le champ Rechercher par nom d'exécution exact pour rechercher une exécution

de flux de travail enfant spécifique. Vous pouvez également utiliser le menu déroulant N'importe quel statut pour filtrer les historiques d'exécution des flux de travail des enfants en fonction de leur statut. Pour voir les détails d'une exécution spécifique, sélectionnez une exécution de flux de travail enfant dans la liste et cliquez sur le bouton Afficher les détails pour ouvrir sa page de détails d'exécution.

> **Important**
>
> La politique de rétention pour les exécutions de flux de travail pour enfants est de 90 jours. Les exécutions de flux de travail enfant terminées qui sont antérieures à cette période de rétention ne sont pas affichées dans le tableau Exécutions. Cela est vrai même si l'état de la carte distribuée ou le flux de travail parent continue de s'exécuter au-delà de la période de rétention. Vous pouvez consulter les détails d'exécution, y compris les résultats, de ces exécutions de flux de travail enfants si vous exportez la sortie d'état de la carte distribuée vers un compartiment Amazon S3 à l'aide de`ResultWriter`.

> **Tip**
>
> Cliquez sur le bouton
>
> 
>
> d'actualisation pour afficher la liste la plus récente de toutes les exécutions de flux de travail enfants.

RedrivingCartes parcourues

Vous pouvez redémarrer les exécutions infructueuses d'un flux de travail enfant dans un flux de travail Map Run par redrivingvotre flux de travail parent. Un flux de travail redriven parent redrives contenant tous les états infructueux, y compris la carte distribuée. Un flux de travail parent réactive les états ayant échoué s'il n'existe aucun `<stateType>Exited` événement correspondant à l'`<stateType>Entered`événement correspondant à un état lorsque le flux de travail parent a terminé son exécution. Par exemple, si l'historique des événements ne contient pas l'événement correspondant à un `MapStateExited` `MapStateEntered` événement, vous pouvez attribuer redrive le flux de travail parent à redrive toutes les exécutions de flux de travail enfant infructueuses dans Map Run.

Une exécution de carte n'est pas démarrée ou échoue lors de la tentative d'exécution initiale lorsque la machine d'état n'a pas l'autorisation requise pour accéder à ItemReaderResultWriter, ou aux deux. Si le Map Run n'a pas été lancé lors de la tentative d'exécution initiale du flux de travail parent, redriving le flux de travail parent démarre le Map Run pour la première fois. Pour résoudre ce problème, ajoutez les autorisations requises à votre rôle de machine à états, puis redrive au flux de travail parent. Si vous êtes redrive le flux de travail parent sans ajouter les autorisations requises, il tente de démarrer une nouvelle exécution de Map Run qui échouera à nouveau. Pour plus d'informations sur les autorisations dont vous pourriez avoir besoin, consultezPolitiques IAM pour l'utilisation de l'état de la carte distribuée.

Rubriques

- Redriveéligibilité aux flux de travail pour enfants dans un Map Run
- redriveComportement d'exécution du workflow par enfant
- Scénarios de saisie utilisés sur Map Run redrive
- Autorisation IAM d'exécuter redrive une carte
- RedrivingMap Run dans la console
- RedrivingMap Run à l'aide de l'API

Redriveéligibilité aux flux de travail pour enfants dans un Map Run

Vous pouvez redrive exécuter un flux de travail enfant infructueux dans un Map Run si les conditions suivantes sont remplies :

- Vous avez commencé l'exécution du flux de travail parent le 15 novembre 2023 ou après cette date. Les exécutions que vous avez commencées avant cette date ne sont pas éligiblesredrive.
- Vous n'avez pas dépassé la limite stricte redrives de 1 000 pour une course de carte donnée. Si vous avez dépassé cette limite, vous recevrez le `States.Runtime` message d'erreur.
- Le flux de travail parent estredrivable. Si le flux de travail parent ne l'est pasredrivable, vous ne pouvez pas exécuter redrive le flux de travail enfant dans un Map Run. Pour plus d'informations sur l'redriveéligibilité d'un flux de travail, consultezRedriveéligibilité en cas d'exécution infructueuse.
- Les exécutions de flux de travail enfants de type Standard dans votre Map Run n'ont pas dépassé la limite de 25 000 événements d'exécution dans l'historique des événements. Les exécutions de flux de travail des enfants qui ont dépassé la limite d'historique des événements sont prises en compte dans le calcul du seuil d'échec toléré et considérées comme ayant échoué. Pour plus

d'informations sur l'redriveéligibilité d'une exécution, consultezRedriveéligibilité en cas d'exécution infructueuse.

Une nouvelle exécution de carte est lancée et l'exécution de carte existante ne l'est pas redriven dans les cas suivants, même si l'exécution de la carte a échoué lors de la tentative d'exécution initiale :

- Map Run a échoué à cause de `States.DataLimitExceeded` cette erreur.
- L'exécution de la carte a échoué en raison de l'erreur d'interpolation des données JSON,`States.Runtime`. Par exemple, vous avez sélectionné un nœud JSON inexistant dansOutputPath.

Une exécution cartographique peut continuer à s'exécuter même après l'arrêt ou l'expiration du flux de travail parent. Dans ces scénarios, redrive cela ne se produit pas immédiatement :

- Map Run est peut-être toujours en train d'annuler les exécutions de flux de travail enfant en cours de type Standard ou d'attendre que les exécutions de flux de travail enfants de type Express soient terminées.
- Map Run est peut-être toujours en train d'écrire des résultats dans leResultWriter, si vous l'avez configuré pour exporter les résultats.

Dans ces cas, le Map Run en cours d'exécution termine ses opérations avant de tenter de le faireredrive.

redriveComportement d'exécution du workflow par enfant

Les exécutions du flux de travail redriven enfant dans un Map Run présentent le comportement décrit dans le tableau suivant.

Flux de travail Express pour enfants	Flux de travail standard pour enfants
Toutes les exécutions de flux de travail enfant qui ont échoué ou ont expiré lors de la tentative d'exécution initiale sont lancées à l'aide de l'action StartExecutionAPI. Le premier état entré ItemProcessor est exécuté en premier.	Toutes les exécutions de flux de travail enfant qui ont échoué, ont expiré ou ont été annulées lors de la tentative d'exécution initiale redriven utilisent l'action RedriveExecutionAPI. Ces flux de travail enfants redriven datent du dernier

Flux de travail Express pour enfants	Flux de travail standard pour enfants
	état ItemProcessor qui a entraîné l'échec de leur exécution.
Des exécutions infructueuses peuvent toujours l'êtreredriven. Cela est dû au fait que les exécutions du flux de travail Express Child sont toujours démarrées en tant que nouvelle exécution à l'aide de l'action StartExecution API.	Les exécutions de flux de travail standard pour enfants ne peuvent pas toujours échouerre driven. Si aucune exécution ne l'estredrivable, elle ne sera pas tentée à nouveau. La dernière erreur ou sortie de l'exécution est permanente. Cela est possible lorsqu'une exécution dépasse 25 000 événements historiques ou que sa redrivable période de 14 jours a expiré. Une exécution de flux de travail enfant standard peut ne pas se redrivable produire si l'exécution du flux de travail parent s'est terminée dans les 14 jours, mais si l'exécution du flux de travail enfant s'est terminée avant 14 jours.
Les exécutions de flux de travail Express Child utilisent le même ARN d'exécution que la tentative d'exécution initiale, mais vous ne pouvez pas les identifier clairement. redrives	Les exécutions de flux de travail enfant standard utilisent le même ARN d'exécuti on que la tentative d'exécution initiale. Vous pouvez identifier clairement l'individu redrives dans la console et à l'aide d'API, telles que GetExecutionHistoryet DescribeExecution. Pour plus d'informations, consultez Examen des redriven exécutions.

Si vous avez redriven une exécution de carte et que celle-ci a atteint sa limite de simultanéité, les exécutions du flux de travail enfant pendant cette exécution de carte passent à l'état en attente. L'état d'exécution de Map Run passe également à l'redriveétat En attente. Jusqu'à ce que la limite de simultanéité spécifiée permette l'exécution d'un plus grand nombre d'exécutions de flux de travail enfants, l'exécution reste à l'redriveétat En attente.

Supposons, par exemple, que la limite de simultanéité de la carte distribuée dans votre flux de travail soit de 3 000 et que le nombre de flux de travail enfants à réexécuter soit de 6 000. Cela entraîne

l'exécution en parallèle de 3 000 flux de travail enfants, tandis que les 3 000 flux de travail restants restent dans l'état Pending redrive. Une fois que le premier lot de 3 000 flux de travail enfants a terminé son exécution, les 3 000 flux de travail enfants restants sont exécutés.

Lorsqu'une exécution de carte est terminée ou est abandonnée, le nombre d'exécutions de flux de travail enfants à l'redriveétat En attente est remis à 0.

Scénarios de saisie utilisés sur Map Run redrive

Selon la manière dont vous avez fourni une entrée à la carte distribuée lors de la tentative d'exécution initiale, une exécution de redriven carte utilisera l'entrée comme décrit dans le tableau suivant.

Entrée lors de la tentative d'exécution initiale	Entrée utilisée sur Map Run redrive
Entrée transmise depuis un état précédent ou depuis l'entrée d'exécution.	Le redriven Map Run utilise la même entrée.
L'entrée a été transmise en utilisant ItemReader et le Map Run n'a pas démarré les exécutions du flux de travail enfant car l'une des conditions suivantes est vraie : • Map Run a échoué avec cette `States.ItemReaderFailed` erreur. • Map Run a échoué avec cette `States.ResultWriterFailed` erreur. • L'exécution du flux de travail parent a expiré ou a été annulée avant le début de Map Run.	Le redriven Map Run utilise l'entrée du compartiment Amazon S3.
Entrée transmise à l'aide de ItemReader. L'exécution de la carte a échoué après le démarrage ou la tentative de lancement d'exécutions de flux de travail enfants.	Le redriven Map Run utilise la même entrée que celle fournie lors de la tentative d'exécution initiale.

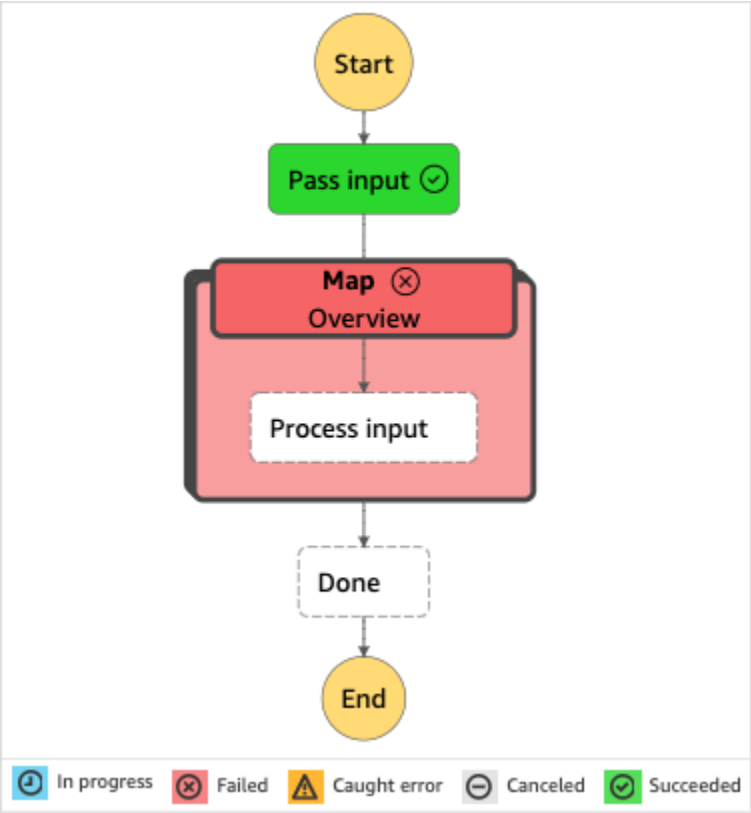
Autorisation IAM d'exécuter redrive une carte

Step Functions a besoin des autorisations appropriées pour exécuter redrive un Map Run. L'exemple de politique IAM suivant accorde le minimum de privilèges requis à votre machine d'état pour redriving une exécution cartographique. N'oubliez pas de remplacer le texte *en italique* par les informations spécifiques à votre ressource.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:RedriveExecution"
      ],
      "Resource": "arn:aws:states:us-
east-2:123456789012:execution:myStateMachine/myMapRunLabel:*"
    }
  ]
}
```

RedrivingMap Run dans la console

L'image suivante montre le graphe d'exécution d'une machine à états contenant une carte distribuée. Cette exécution a échoué car le Map Run a échoué. Pour redrive exécuter Map Run, vous devez utiliser redrive le flux de travail parent.



Vers redrive une exécution cartographique depuis la console

1. Ouvrez la console Step Functions, puis choisissez une machine à états existante contenant une carte distribuée dont l'exécution a échoué.
2. Sur la page détaillée de la machine à états, sous Exécutions, choisissez une instance d'exécution ayant échoué de cette machine à états.
3. Sélectionnez Redrive.
4. Dans la Redriveboîte de dialogue, sélectionnez RedriveExécution.

> **Tip**
>
> Vous pouvez également effectuer redrive une exécution cartographique à partir de la page Détails de l'exécution ou Détails de l'exécution de la carte.
> Si vous êtes sur la page Détails de l'exécution, effectuez l'une des opérations suivantes pour redrive l'exécution :
>
> - Choisissez Restaurer, puis sélectionnez En cas Redrive d'échec.
> - Choisissez Actions, puis sélectionnez Redrive.

Si vous êtes sur la page Map Run Details, choisissez Recover, puis sélectionnez RedriveFrom failure.

Notez qu'il redrive utilise la même définition de machine à états et le même ARN. Il poursuit l'exécution à partir de l'étape qui a échoué lors de la tentative d'exécution initiale. Dans cet exemple, il s'agit de l'étape Carte distribuée nommée Carte et de l'étape de saisie du processus qui s'y trouve. Après le redémarrage du flux de travail enfant infructueux, les exécutions de Map Run se redrive poursuivront pour l'étape Terminé.

5. Sur la page Détails de l'exécution, choisissez Map Run pour voir les détails de l'redrivenexécution.

   Sur cette page, vous pouvez consulter les résultats de l'redrivenexécution. Par exemple, dans la Résumé de l'exécution de Map Run section, vous pouvez voir le Redrivenombre, qui représente le nombre de fois que la Map Run a été exécutéeredriven. Dans la section Événements, vous pouvez voir les événements d'exécution redrive associés ajoutés aux événements de la tentative d'exécution initiale. Par exemple, l'`MapRunRedrive`événement.

Après avoir exécuté redriven un Map Run, vous pouvez en examiner les redrive détails dans la console ou à l'aide des actions de l'DescribeExecutionAPI GetExecutionHistoryet. Pour plus d'informations sur l'examen d'une redriven exécution, consultezExamen des redriven exécutions.

RedrivingMap Run à l'aide de l'API

Vous pouvez exécuter redrive une carte éligible à l'aide de l'RedriveExecutionAPI du flux de travail parent. Cette API redémarre les exécutions infructueuses du flux de travail enfant dans un Map Run.

Dans le AWS Command Line Interface (AWS CLI), exécutez la commande suivante en cas redrive d'échec de l'exécution de la machine à états. N'oubliez pas de remplacer le texte *`en italique`* par les informations spécifiques à votre ressource.

```
aws stepfunctions redrive-execution --execution-arn arn:aws:states:us-
east-2:123456789012:execution:myStateMachine:foo
```

Après avoir exécuté redriven un Map Run, vous pouvez en examiner les redrive détails dans la console ou à l'aide de l'action de l'DescribeMapRunAPI. Pour examiner les redrive détails

des exécutions de flux de travail standard dans un Map Run, vous pouvez utiliser l'action DescribeExecutionAPI GetExecutionHistoryor. Pour plus d'informations sur l'examen d'une redriven exécution, consultezthe section called “Examen des redriven exécutions”.

Vous pouvez examiner les redrive détails des exécutions de flux de travail Express dans un Map Run sur la console Step Functions si vous avez activé la connexion au flux de travail parent. Pour plus d'informations, voir Journalisation à l'aideCloudWatchJournaux.

Gestion des erreurs dans Step Functions

Tous les états, à l'exception `Pass` des `Wait` états, peuvent rencontrer des erreurs d'exécution. Des erreurs peuvent se produire pour différentes raisons, comme dans les exemples suivants :

- Problèmes de définition de la machine d'état (par exemple, aucune règle correspondante dans un état `Choice`)
- Echecs de tâche (par exemple, une exception dans une fonction AWS Lambda)
- Problèmes temporaires (par exemple, les événements de partition du réseau)

Par défaut, lorsqu'un état signale une erreur, AWS Step Functions entraîne l'échec complet de l'exécution.

> **Tip**
>
> Pour déployer un exemple de flux de travail incluant la gestion des erreurs sur votre ordinateurCompte AWS, consultez le Module 8 - Gestion des erreurs dans l'AWS Step Functionsatelier.

Rubriques

- Noms des erreurs
- Réessayer après une erreur
- États de repli
- Exemples de machines à états utilisant Retry et Catch

Noms des erreurs

Step Functions identifie les erreurs dans l'Amazon States Language à l'aide de chaînes distinguant majuscules et minuscules, appelées noms d'erreur. L'Amazon States Language définit un ensemble de chaînes intégrées qui nomment les erreurs connues, en commençant toutes par le `States.` préfixe.

`States.ALL`

Un caractère générique qui correspond à n'importe quel nom d'erreur connu.

> **Note**
>
> Ce type d'erreur ne peut pas détecter le type d'erreur du `States.DataLimitExceeded` terminal et les types d'erreur d'exécution. Pour plus d'informations sur ces types d'erreur, reportez-vous `States.DataLimitExceeded`aux sections et `States.Runtime`.

`States.DataLimitExceeded`

Step Functions signale une `States.DataLimitExceeded` exception dans les conditions suivantes :

- Lorsque la sortie d'un connecteur est supérieure au quota de charge utile.
- Lorsque le résultat d'un état est supérieur au quota de taille de charge utile.
- Lorsque, après le `Parameters` traitement, l'entrée d'un état est supérieure au quota de taille de charge utile.

Pour plus d'informations sur les quotas, voirQuotas.

> **Note**
>
> Il s'agit d'une erreur de terminal qui ne peut pas être détectée par le type `States.ALL` d'erreur.

`States.ExceedToleratedFailureThreshold`

Un `Map` état a échoué car le nombre d'éléments défaillants a dépassé le seuil spécifié dans la définition de la machine à états. Pour en savoir plus, consultez Seuil de défaillance toléré pour l'état de la carte distribuée.

`States.HeartbeatTimeout`

Un `Task` état n'a pas réussi à envoyer un battement de cœur pendant une période supérieure à la `HeartbeatSeconds` valeur.

> **Note**
>
> Cette erreur n'est disponible que dans les `Retry` champs `Catch` et.

`States.IntrinsicFailure`

Une tentative d'appel d'une fonction intrinsèque dans un modèle de charge utile a échoué.

`States.ItemReaderFailed`

Un `Map` état a échoué car il n'a pas pu lire à partir de la source de l'élément spécifiée dans le `ItemReader` champ. Pour en savoir plus, consultez `ItemReader`.

`States.NoChoiceMatched`

Un `Choice` état n'a pas réussi à faire correspondre l'entrée aux conditions définies dans la règle de choix et aucune transition par défaut n'est spécifiée.

`States.ParameterPathFailure`

Une tentative de remplacement d'un champ, dans le champ d'un État, dont `Parameters` le nom se termine par l'`.$`utilisation d'un chemin échoue.

`States.Permissions`

Un `Task` état a échoué car il ne disposait pas de privilèges suffisants pour exécuter le code spécifié.

`States.ResultPathMatchFailure`

Step Functions n'a pas réussi à appliquer `ResultPath` le champ d'un état à l'entrée que l'état a reçue.

`States.ResultWriterFailed`

Un `Map` état a échoué car il n'a pas pu écrire les résultats vers la destination spécifiée dans le `ResultWriter` champ. Pour en savoir plus, consultez `ResultWriter`.

`States.Runtime`

Une exécution a échoué en raison d'une exception qu'elle n'a pas pu traiter. Souvent, ces erreurs sont causées par des erreurs lors de l'exécution, telles que la tentative d'application `InputPath` ou `OutputPath` sur une charge utile JSON nulle. Une `States.Runtime` erreur n'est pas récupérable et entraîne toujours l'échec de l'exécution. Une nouvelle tentative ou un catch on `States.ALL` ne détectera pas `States.Runtime` les erreurs.

`States.TaskFailed`

L'état `Task` a échoué au cours de l'exécution. Lorsqu'il est utilisé dans un retry ou un catch, `States.TaskFailed` agit comme un caractère générique qui correspond à n'importe quel nom d'erreur connu, à l'exception de. `States.Timeout`

`States.Timeout`

Un état `Task` a été exécuté plus longtemps que la valeur `TimeoutSeconds` ou n'est pas parvenu à envoyer une pulsation pendant une période plus longue que la valeur `HeartbeatSeconds`.

De plus, si une machine à états fonctionne plus longtemps que la `TimeoutSeconds` valeur spécifiée, l'exécution échoue avec une `States.Timeout` erreur.

Les états peuvent signaler les erreurs avec d'autres noms. Toutefois, ces noms d'erreur ne peuvent pas commencer par le `States.` préfixe.

Une bonne pratique consiste à s'assurer que le code de production puisse gérer les exceptions de service AWS Lambda (`Lambda.ServiceException` et `Lambda.SdkClientException`). Pour en savoir plus, consultez Gérer les exceptions du service Lambda.

> **Note**
>
> Les erreurs non gérées dans Lambda sont signalées `Lambda.Unknown` comme dans le résultat d'erreur. Il s'agit notamment out-of-memory des erreurs et des délais d'expiration des fonctions. Vous pouvez effectuer une correspondance ou `States.TaskFailed` pour gérer ces erreurs. `Lambda.Unknown` `States.ALL` Lorsque Lambda atteint le nombre maximum d'appels, l'erreur est. `Lambda.TooManyRequestsException` Pour plus d'informations sur

les erreurs liées aux fonctions Lambda, consultez la section Gestion des erreurs et tentatives automatiques dans le Guide du AWS Lambda développeur.

Réessayer après une erreur

`TaskParallel`, et `Map` les états peuvent avoir un champ nomméRetry, dont la valeur doit être un tableau d'objets appelés retriers. Un réessayeur représente un certain nombre de nouvelles tentatives, généralement à intervalles de temps croissants.

Lorsque l'un de ces états signale une erreur et qu'un `Retry` champ apparaît, Step Functions parcourt les récupérateurs dans l'ordre indiqué dans le tableau. Lorsque le nom de l'erreur apparaît dans la valeur du `ErrorEquals` champ d'un récupérateur, la machine à états effectue de nouvelles tentatives comme indiqué dans le `Retry` champ.

Si votre redriven exécution réexécute un état État de la tâcheParallèle, ou Inline Map, pour lequel vous avez défini de nouvelles tentatives, le nombre de tentatives pour ces états est remis à 0 pour tenir compte du nombre maximum de tentatives. redrive Pour une redriven exécution, vous pouvez suivre les tentatives de tentative individuelles de ces états à l'aide de la console. Pour plus d'informations, consultez Réessayer le comportement des exécutions redriven dans Redrivingexécutions.

Un réessayeur contient les champs suivants :

> **Note**
>
> Les nouvelles tentatives sont traitées comme des transitions d'état. Pour plus d'informations sur l'impact des transitions entre États sur la facturation, consultez Step Functions Pricing.

`ErrorEquals` (Obligatoire)

Tableau non vide de chaînes qui correspondent aux noms d'erreur. Lorsqu'un état signale une erreur, Step Functions analyse les récupérateurs. Lorsque le nom de l'erreur apparaît dans ce tableau, il met en œuvre la stratégie de nouvelle tentative décrite dans ce réessayeur.

`IntervalSeconds` (facultatif)

Un entier positif qui représente le nombre de secondes avant la première tentative (1par défaut). `IntervalSeconds`a une valeur maximale de 99999999.

`MaxAttempts` (facultatif)

Nombre entier positif qui représente le nombre maximum de nouvelles tentatives (`3` par défaut). Si l'erreur se produit un nombre de fois supérieur à la valeur spécifiée, les nouvelles tentatives cessent et la gestion normale des erreurs reprend. La valeur de `0` indique que l'erreur n'est jamais réessayée. `MaxAttempts`a une valeur maximale de 99999999.

`BackoffRate` (facultatif)

Multiplicateur par lequel l'intervalle de tentatives indiqué par `IntervalSeconds` augmente après chaque nouvelle tentative. Par défaut, la `BackoffRate` valeur augmente de`2.0`.

Par exemple, disons `IntervalSeconds` que vous êtes `MaxAttempts` 3, 3 et `BackoffRate` 2. La première tentative a lieu trois secondes après l'apparition de l'erreur. La deuxième tentative a lieu six secondes après la première tentative. Alors que la troisième tentative a lieu 12 secondes après la deuxième tentative.

`MaxDelaySeconds` (facultatif)

Nombre entier positif qui définit la valeur maximale, en secondes, jusqu'à laquelle un intervalle de nouvelles tentatives peut être augmenté. Ce champ est utile à utiliser avec le `BackoffRate` champ. La valeur que vous spécifiez dans ce champ limite les temps d'attente exponentiels résultant du multiplicateur du taux de ralentissement appliqué à chaque nouvelle tentative consécutive. Vous devez spécifier une valeur supérieure à 0 et inférieure à 31622401 pour. `MaxDelaySeconds`

Si vous ne spécifiez pas cette valeur, Step Functions ne limite pas les temps d'attente entre les nouvelles tentatives.

`JitterStrategy` (facultatif)

Chaîne qui détermine s'il faut ou non inclure la gigue dans les temps d'attente entre deux tentatives consécutives. Jitter réduit le nombre de tentatives simultanées en les répartissant sur un intervalle de temporisation aléatoire. Cette chaîne accepte `FULL` ou `NONE` comme valeurs. La valeur par défaut est `NONE`.

Supposons, par exemple, que vous ayez défini `MaxAttempts` `IntervalSeconds` comme 3, comme 2 et `BackoffRate` comme 2. La première tentative a lieu deux secondes après l'apparition de l'erreur. La deuxième tentative a lieu quatre secondes après la première tentative et la troisième tentative huit secondes après la deuxième tentative. Si vous définissez `JitterStrategy` comme`FULL`, le premier intervalle de tentatives est randomisé entre 0 et 2

secondes, le deuxième intervalle entre 0 et 4 secondes, et le troisième intervalle entre 0 et 8 secondes.

Exemples de champs à réessayer

Cette section inclut les exemples de `Retry` champs suivants.

- Retry with BackoffRate
- Retry with MaxDelaySeconds
- Retry all errors except States.Timeout
- Complex retry scenario

> **Tip**
>
> Pour déployer un exemple de flux de travail de gestion des erreurs sur votre ordinateurCompte AWS, consultez le module de gestion des erreurs de The AWS Step Functions Workshop.

Exemple 1 — Réessayez avec BackoffRate

L'exemple suivant montre `Retry` deux tentatives de tentative, la première ayant lieu après trois secondes d'attente. En fonction de ce `BackoffRate` que vous spécifiez, Step Functions augmente l'intervalle entre chaque nouvelle tentative jusqu'à ce que le nombre maximum de tentatives soit atteint. Dans l'exemple suivant, la deuxième tentative commence après une attente de trois secondes après la première tentative.

```
"Retry": [ {
   "ErrorEquals": [ "States.Timeout" ],
   "IntervalSeconds": 3,
   "MaxAttempts": 2,
   "BackoffRate": 1
} ]
```

Exemple 2 — Réessayez avec MaxDelaySeconds

L'exemple suivant effectue trois tentatives de nouvelle tentative et limite le temps d'attente `BackoffRate` à 5 secondes. La première tentative a lieu après trois secondes d'attente. Les

deuxième et troisième tentatives ont lieu après une attente de cinq secondes après la tentative précédente en raison de la limite de temps d'attente maximale définie par. `MaxDelaySeconds`

```
"Retry": [ {
    "ErrorEquals": [ "States.Timeout" ],
    "IntervalSeconds": 3,
    "MaxAttempts": 3,
    "BackoffRate":2,
    "MaxDelaySeconds": 5,
    "JitterStrategy": "FULL"
} ]
```

Dans le cas contraire`MaxDelaySeconds`, la deuxième tentative aurait lieu six secondes après la première tentative, et la troisième tentative aurait lieu 12 secondes après la deuxième tentative.

Exemple 3 — Réessayez toutes les erreurs sauf States.Timeout

Le nom réservé `States.ALL` qui apparaît dans le champ `ErrorEquals` du Réessayeur est une valeur générique correspondant à n'importe quel nom d'erreur. Il doit apparaître seul dans le tableau `ErrorEquals` et doit figurer dans le dernier réessayeur du tableau `Retry`. Le nom agit `States.TaskFailed` également comme un caractère générique et correspond à toutes les erreurs, à l'exception de`States.Timeout`.

Dans l'exemple de `Retry` champ suivant, une nouvelle tentative d'erreur est effectuée, sauf`States.Timeout`.

```
"Retry": [ {
   "ErrorEquals": [ "States.Timeout" ],
   "MaxAttempts": 0
}, {
   "ErrorEquals": [ "States.ALL" ]
} ]
```

Exemple 4 — Scénario de nouvelle tentative complexe

Les paramètres d'un réessayeur s'appliquent à toutes les visites de ce réessayeur dans le cadre d'une seule exécution d'état.

Prenons l'exemple de l'état `Task` suivant :

```
"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"
   } ]
}
```

Cette tâche échoue quatre fois de suite et affiche les noms d'erreur suivants :`ErrorA`, `ErrorBErrorC`, et`ErrorB`. Voici ce qui se produit ensuite :

- Les deux premières erreurs correspondent au premier retrier et entraînent des temps d'attente d'une ou deux secondes.
- La troisième erreur correspond au deuxième retrier et entraîne une attente de cinq secondes.
- La quatrième erreur correspond également au premier retrier. Cependant, il a déjà atteint son maximum de deux tentatives (`MaxAttempts`) pour cette erreur particulière. Par conséquent, ce récupérateur échoue et l'exécution redirige le flux de travail vers l'Zétat via le `Catch` champ.

États de repli

`Task`, `Map` et `Parallel` les états peuvent chacun avoir un champ nomméCatch. La valeur de ce champ doit être un tableau d'objets, nommés receveurs.

Un receveur contient les champs suivants.

`ErrorEquals` (Obligatoire)

Un tableau non vide de chaînes qui correspondent à des noms d'erreur, spécifiées exactement comme dans le champ de réessayeur du même nom.

`Next` (Obligatoire)

Une chaîne qui doit correspondre exactement à l'un des noms d'état de la machine d'état.

`ResultPath` (facultatif)

Un chemin qui détermine l'entrée que le receveur envoie à l'état spécifié dans le `Next` champ.

Lorsqu'un État signale une erreur et qu'il n'y a aucun `Retry` champ, ou si les tentatives échouent à résoudre l'erreur, Step Functions analyse les capteurs dans l'ordre indiqué dans le tableau. Lorsque le nom de l'erreur apparaît dans la valeur du champ `ErrorEquals` d'un receveur, la machine d'état passe à l'état nommé dans le champ `Next`.

Le nom réservé `States.ALL` qui apparaît dans le champ `ErrorEquals` du receveur est une valeur générique correspondant à n'importe quel nom d'erreur. Il doit apparaître seul dans le tableau `ErrorEquals` et doit figurer dans le dernier receveur du tableau `Catch`. Le nom agit `States.TaskFailed` également comme un caractère générique et correspond à toutes les erreurs, à l'exception de`States.Timeout`.

L'exemple suivant d'un `Catch` champ passe à l'état nommé `RecoveryState` lorsqu'une fonction Lambda génère une exception Java non gérée. Sinon, le champ passe à l'état `EndState`.

```
"Catch": [ {
   "ErrorEquals": [ "java.lang.Exception" ],
   "ResultPath": "$.error-info",
   "Next": "RecoveryState"
}, {
   "ErrorEquals": [ "States.ALL" ],
   "Next": "EndState"
} ]
```

> **Note**
>
> Chaque receveur peut spécifier plusieurs erreurs à gérer.

Sortie d'erreur

Lorsque Step Functions passe à l'état spécifié dans un catch name, l'objet contient généralement le champ`Cause`. La valeur de ce champ est une description lisible de l'erreur. Cet objet est connu sous le nom de sortie de l'erreur.

Dans cet exemple, le premier receveur contient un champ `ResultPath`. Cela fonctionne de la même manière qu'un champ `ResultPath` au niveau supérieur d'un état, ce qui se traduit par deux possibilités :

- Il prend les résultats de l'exécution de cet état et remplace la totalité ou une partie des entrées de l'état.
- Il prend les résultats et les ajoute à l'entrée. Dans le cas d'une erreur gérée par un catcher, le résultat de l'exécution de l'état est la sortie d'erreur.

Ainsi, pour le premier capteur de l'exemple, le capteur ajoute la sortie d'erreur à l'entrée sous forme de champ nommé `error-info` s'il n'existe pas déjà de champ portant ce nom dans l'entrée. Ensuite, le receveur envoie l'intégralité de l'entrée à`RecoveryState`. Pour le second capteur, la sortie d'erreur remplace l'entrée et le capteur envoie uniquement la sortie d'erreur à. `EndState`

> **Note**
>
> Lorsque la valeur du champ `ResultPath` n'est pas spécifiée, c'est $ qui est utilisée par défaut afin de sélectionner, et donc de remplacer, toute l'entrée.

Lorsqu'un état possède à la fois `Retry` des `Catch` champs et, Step Functions utilise d'abord les récupérateurs appropriés. Si la politique de nouvelles tentatives ne parvient pas à résoudre l'erreur, Step Functions applique la transition de capture correspondante.

Provoquer des charges utiles et des intégrations de services

Un catcher renvoie une charge utile sous forme de chaîne en sortie. Lorsque vous travaillez avec des intégrations de services telles qu'Amazon Athena AWS CodeBuild ou, vous souhaiterez peut-être convertir `Cause` la chaîne en JSON. L'exemple suivant d'un `Pass` état avec des fonctions intrinsèques montre comment convertir une `Cause` chaîne en JSON.

```
"Handle escaped JSON with JSONtoString": {
  "Type": "Pass",
  "Parameters": {
    "Cause.$": "States.StringToJson($.Cause)"
  },
  "Next": "Pass State with Pass Processing"
},
```

Exemples de machines à états utilisant Retry et Catch

Les machines à états définies dans les exemples suivants supposent l'existence de deux fonctions Lambda : l'une qui échoue toujours et l'autre qui attend suffisamment longtemps pour qu'un délai défini dans la machine à états se produise.

Il s'agit de la définition d'une fonction Lambda Node.js qui échoue toujours et renvoie le message. `error` Dans les exemples de machine à états suivants, cette fonction Lambda est nommée. `FailFunction` Pour plus d'informations sur la création d'une fonction Lambda, reportez-vous Étape 1 : Créer une fonction Lambda à la section.

```
exports.handler = (event, context, callback) => {
    callback("error");
};
```

Il s'agit de la définition d'une fonction Lambda de Node.js qui est en veille pendant 10 secondes. Dans les exemples de machine à états suivants, cette fonction Lambda est nommée. `sleep10`

> **Note**
>
> Lorsque vous créez cette fonction Lambda dans la console Lambda, n'oubliez pas de modifier la valeur du délai d'expiration dans la section Paramètres avancés de 3 secondes (par défaut) à 11 secondes.

```
exports.handler = (event, context, callback) => {
    setTimeout(function(){
    }, 11000);
};
```

Gérer un échec à l'aide de Retry

Cette machine d'état utilise un champ `Retry` pour réessayer une fonction qui échoue et génère le nom d'erreur `HandledError`. Il réessaie cette fonction deux fois avec un décalage exponentiel entre les tentatives.

```
{
   "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
      }
   }
}
```

Cette variante utilise le code d'erreur prédéfini`States.TaskFailed`, qui correspond à toute erreur générée par une fonction Lambda.

```
{
   "Comment": "A Hello World example of the Amazon States Language using an AWS Lambda
 function",
   "StartAt": "HelloWorld",
   "States": {
      "HelloWorld": {
         "Type": "Task",
         "Resource": "arn:aws:lambda:us-east-1:123456789012:function:FailFunction",
         "Retry": [ {
            "ErrorEquals": ["States.TaskFailed"],
            "IntervalSeconds": 1,
            "MaxAttempts": 2,
            "BackoffRate": 2.0
         } ],
         "End": true
      }
   }
}
```

> **Note**
>
> Il est recommandé que les tâches qui font référence à une fonction Lambda gèrent les exceptions de service Lambda. Pour en savoir plus, consultez Gérer les exceptions du service Lambda.

Gérer une panne à l'aide de Catch

Cet exemple utilise un champ `Catch`. Lorsqu'une fonction Lambda génère une erreur, elle détecte l'erreur et la machine à états passe à l'`fallback`état.

```
{
   "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
      }
   }
}
```

Cette variante utilise le code d'erreur prédéfini`States.TaskFailed`, qui correspond à toute erreur générée par une fonction Lambda.

```
{
   "Comment": "A Hello World example of the Amazon States Language using an AWS Lambda
 function",
   "StartAt": "HelloWorld",
```

```
   "States": {
      "HelloWorld": {
         "Type": "Task",
         "Resource": "arn:aws:lambda:us-east-1:123456789012:function:FailFunction",
         "Catch": [ {
            "ErrorEquals": ["States.TaskFailed"],
            "Next": "fallback"
         } ],
      "End": true
      },
      "fallback": {
         "Type": "Pass",
         "Result": "Hello, AWS Step Functions!",
         "End": true
      }
   }
}
```

Gestion d'un délai d'attente à l'aide de Retry

Cette machine à états utilise un `Retry` champ pour réessayer un `Task` état expirant, en fonction de la valeur de délai spécifiée dans. `TimeoutSeconds` Step Functions réessaie deux fois d'invoquer la fonction Lambda dans `Task` cet état, avec un décalage exponentiel entre les tentatives.

```
{
   "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
      }
   }
```

```
}
```

Gérer un délai d'attente à l'aide de Catch

Cet exemple utilise un champ `Catch`. Lorsqu'un délai arrive à expiration, la machine d'état passe à l'état `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**
>
> Vous pouvez conserver l'entrée d'origine avec l'erreur, à l'aide de l'option `ResultPath`. Consultez ResultPath À utiliser pour inclure à la fois une erreur et une entrée dans un `Catch`.

InvoquerAWS Step Functionsà partir d'autres services

Vous pouvez configurer plusieurs autres services pour appeler des machines d'État. Sur la base de la machine d'Étattype de flux de travail, vous pouvez appeler des machines d'État de

manière asynchrone ou synchrone. Pour appeler des machines d'état de manière synchrone, utilisezStartSyncExecutionAppel d'API ou intégration d'Amazon API Gateway à Express Workflows. En cas d'invocation asynchrone, Step Functions suspend l'exécution du flux de travail jusqu'à ce qu'un jeton de tâche soit renvoyé. Toutefois, l'attente d'un jeton de tâche rend le flux de travail synchrone.

Les services que vous pouvez configurer pour appeler Step Functions incluent :

- AWS Lambda, à l'aide duStartExecutionappel.
- Amazon API Gateway
- Amazon EventBridge
- AWS CodePipeline
- AWS IoTMoteur de règles
- AWS Step Functions

Les invocations de Step Functions sont régies parStartExecutionquota. Pour plus d'informations, reportez-vous à :

- Quotas

Cohérence de lecture dans Step Functions

Les mises à jour de machine d'état dans AWS Step Functionssont cohérentes à terme. Dans quelques secondes, tous les `StartExecution` appels utiliseront la définition mise à jour et `roleArn` (le nom de ressource Amazon pour le rôle IAM). Les exécutions démarrées immédiatement après `UpdateStateMachine` peuvent utiliser la définition de machine d'état précédente et `roleArn`.

Pour plus d'informations, consultez les ressources suivantes :

- `UpdateStateMachine` dans la Référence d'API AWS Step Functions
- Mettre à jour un flux de travail dansDémarrer avec AWS Step Functions.

Fonctions de balisage en étapes

AWS Step Functions prend en charge le marquage des machines d'état (standard et express) et des activités. Cela peut vous aider à suivre et à gérer les coûts associés à vos ressources et à renforcer la sécurité de vos politiques AWS Identity and Access Management (IAM). Le balisage des ressources de Step Functions permet de les gérer parAWS Resource Groups. Pour plus d'informations sur les groupes de ressources, consultez le Guide de AWS Resource Groups l'utilisateur.

Pour l'autorisation basée sur des balises, les ressources d'exécution de la machine d'état, comme indiqué dans l'exemple suivant, héritent des balises associées à une machine d'état.

```
arn:<partition>:states:<Region>:<account-id>:execution:<StateMachineName>:<ExecutionId>
```

Lorsque vous appelez DescribeExecutionou d'autres API dans lesquelles vous spécifiez l'ARN de la ressource d'exécution, Step Functions utilise des balises associées à la machine d'état pour accepter ou refuser la demande tout en effectuant une autorisation basée sur des balises. Cela vous permet d'autoriser ou de refuser l'accès aux exécutions de la machine d'État au niveau de la machine d'État.

Pour passer en revue les restrictions liées au balisage de ressources, consultez Restrictions liées au balisage.

Rubriques

- Balisage de répartition des coûts
- Balisage pour la sécurité
- Affichage et gestion des balises dans la console Step Functions
- Gérez les balises avec les actions de l'API Step Functions

Balisage de répartition des coûts

Pour organiser et identifier vos ressources Step Functions en vue de la répartition des coûts, vous pouvez ajouter des balises de métadonnées qui identifient l'objectif d'une machine ou d'une activité d'état. Cette approche est utile lorsque vous avez un grand nombre de ressources. Vous pouvez utiliser des identifications d'allocation des coûts pour organiser votre facture AWS afin de refléter votre propre structure de coût. Pour ce faire, inscrivez-vous pour que votre facture de compte AWS inclut les clés et valeurs de balise. Pour plus d'informations, consultez Configuration du rapport de répartition des coûts mensuel dans le Guide d'utilisateur AWS Billing.

Par exemple, vous pouvez ajouter des balises qui représentent le centre de coûts et l'objectif de vos ressources Step Functions, comme suit.

Ressource	Clé	Valeur
StateMachine1	Cost Center	34567
	Application	Image processing
StateMachine2	Cost Center	34567
	Application	Rekognition processing
Activity1	Cost Center	12345
	Application	Legacy database

Cette méthode de balisage vous permet de regrouper les deux machines d'état effectuant les tâches connexes dans le même centre de coûts, tout en balisant une activité indépendante avec une autre balise de répartition des coûts.

Balisage pour la sécurité

IAM prend en charge le contrôle de l'accès aux ressources en fonction des balises. Pour contrôler l'accès en fonction des balises, fournissez des informations sur vos balises de ressources dans l'élément condition d'une politique IAM.

Par exemple, vous pouvez restreindre l'accès à toutes les ressources Step Functions qui incluent une balise avec la clé `environment` et la valeur`production`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "states:TagResource",
                "states:DeleteActivity",
```

```
                "states:DeleteStateMachine",
                "states:StopExecution"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {"aws:ResourceTag/environment": "production"}
            }
        }
    ]
}
```

Pour plus d'informations, consultez Contrôle de l'accès à l'aide d'identifications dans le Guide de l'utilisateur IAM.

Affichage et gestion des balises dans la console Step Functions

Step Functions vous permet de visualiser et de gérer les balises de vos machines d'état dans la console Step Functions. Sur la page Détails d'une machine d'état, sélectionnez Tags. Ici, vous pouvez consulter les balises existantes associées à votre machine d'état.

> **Note**
>
> Pour gérer les balises d'activités, consultez Gérez les balises avec les actions de l'API Step Functions.

Pour ajouter ou supprimer des balises associées à votre machine d'état, sélectionnez le bouton Manage Tags (Gérer les balises).

1. Naviguez vers la page de détails d'une machine d'état.
2. Sélectionnez Tags (Balises) à côté de Executions (Exécutions) et Definition (Définition).
3. Choisissez Manage tags (Gérer les balises).
   - Pour modifier des balises existantes, modifiez la Clé et la Valeur.
   - Pour supprimer des balises existantes, choisissez Supprimer la balise.
   - Pour ajouter une nouvelle balise, sélectionnez Ajouter une balise et saisissez une Clé et une Valeur.
4. Choisissez Save (Enregistrer).

Gérez les balises avec les actions de l'API Step Functions

Pour gérer les balises à l'aide de l'API Step Functions, utilisez les actions d'API suivantes :

- `ListTagsForResource`
- `TagResource`
- `UntagResource`

AWS Step Functions Studio de flux de travail

Workflow Studio for AWS Step Functions est un concepteur visuel de flux de travail low-code qui vous permet de créer des flux de travail sans serveur en AWS orchestrant des services. À l'aide de sa drag-and-drop fonctionnalité ou de l'éditeur de code intégré, vous pouvez créer et modifier des flux de travail, contrôler la manière dont les entrées et les sorties sont filtrées ou transformées pour chaque état, et configurer la gestion des erreurs. Lorsque vous glissez et déposez des états pour créer votre flux de travail, Workflow Studio valide votre travail et génère automatiquement du code. Vous pouvez consulter le code généré ou mettre à jour la définition de la machine à états dans l'éditeur de code. Lorsque vous avez terminé, vous pouvez enregistrer votre flux de travail, l'exécuter, puis examiner les résultats dans la console Step Functions. Vous pouvez ajouter et modifier visuellement des flux de travail pour orchestrer les multiples services de votre application.

Pour utiliser Step Functions Workflow Studio, vous aurez besoin d'un Compte AWS identifiant et d'informations d'identification fournissant les autorisations appropriées pour toutes les ressources que vous souhaitez utiliser. Pour plus d’informations, consultez Conditions préalables pour démarrer avec AWS Step Functions.

> **Note**
>
> Workflow Studio ne prend pas en charge Internet Explorer 11. Si vous utilisez Internet Explorer 11 et que vous rencontrez des problèmes lors de l'utilisation de Workflow Studio, essayez d'utiliser un autre navigateur.

Vous pouvez accéder à Workflow Studio depuis la console Step Functions, lorsque vous créez ou modifiez un flux de travail dans Step Functions. Vous pouvez également accéder à Workflow Studio depuisComposeur d'applications AWS. Workflow Studio in Application Composer fournit un environnement IaC visuel qui vous permet d'intégrer facilement des flux de travail dans vos applications sans serveur créées à l'aide d'outils IaC, tels que des AWS CloudFormation modèles. À l'aide de Workflow StudioApplication Composer, vous pouvez créer des flux de travail à l'aide AWS CloudFormation de modèles. Vous pouvez y ajouter un nouveau flux de travail, modifier un flux de travail existant et connecter des étapes de flux de travail individuelles à d'autres ressources de l'application. Application Composer Application Composercrée et met à jour automatiquement les CloudFormation ressources et les configurations nécessaires. Cela vous permet de créer et de gérer toutes les ressources utilisées dans vos flux de travail en un seul endroit. Cela vous permet également d'accélérer le passage du prototypage de flux de travail au déploiement en production.

Lorsque vous utilisez Workflow Studio dansApplication Composer, vous pouvez également vous connecter directement à votre projet local. Cela vous permet de travailler dans votre environnement de développement intégré (IDE) parallèlement au canevas visuel. Pour plus d'informations, consultez Utilisation de Workflow Studio dans Application Composer.

Rubriques

Présentation de l'interface

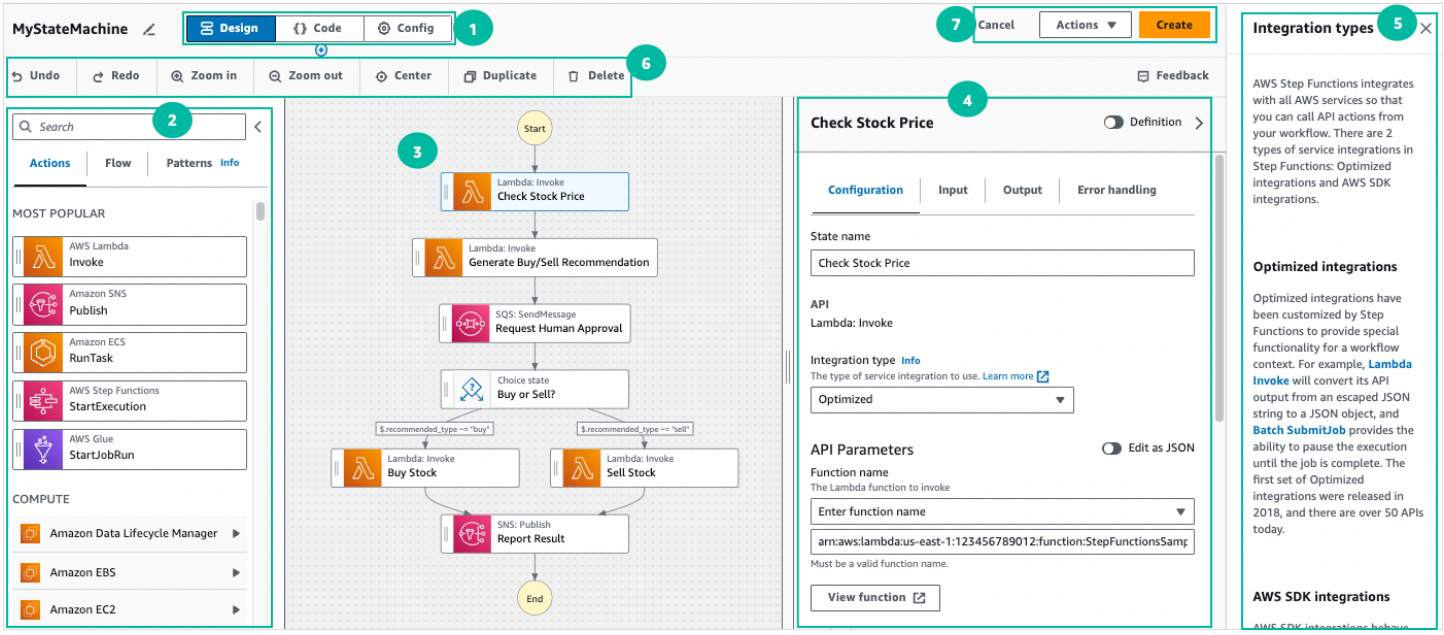
Workflow Studio for AWS Step Functions est un concepteur de flux de travail visuel à faible code qui vous permet de créer des flux de travail sans serveur en les orchestrant. Services AWS Grâce à sa fonction glisser-déposer, Workflow Studio vous permet de créer, de modifier et de visualiser facilement vos prototypes de flux de travail. Workflow Studio propose également un éditeur de code intégré pour écrire et modifier vos définitions de flux de travail en utilisant Amazon States Language (ASL) dans la console Step Functions.

Pour vous aider à créer et à visualiser vos flux de travail, à modifier leurs définitions et à gérer leur configuration, Workflow Studio propose trois modes : Design, Code et Config. Les sections suivantes décrivent ces modes en détail.

Dans cette rubrique

Mode de conception

Le mode Design de Workflow Studio fournit une interface graphique permettant de visualiser vos flux de travail au fur et à mesure que vous créez leurs prototypes. L'image suivante montre les différents composants disponibles en mode Design.



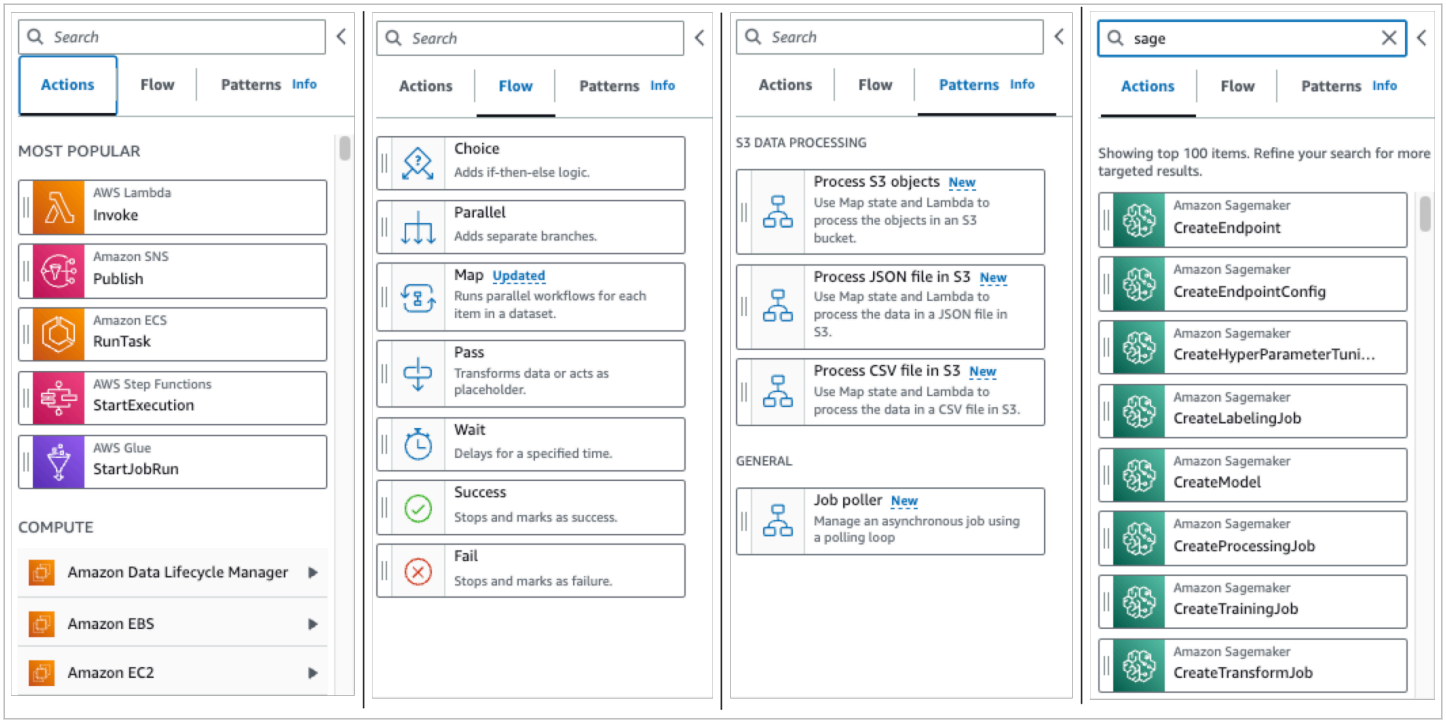
1. Boutons de mode — Basculez entre les modes Design, Code et Config du Workflow Studio à l'aide des boutons de mode. Vous ne pouvez pas changer de mode si le JSON de la définition ASL de votre flux de travail n'est pas valide.
2. Navigateur d'étatsIl contient les trois onglets suivants :
   - L'onglet Actions fournit une liste d' AWS API que vous pouvez glisser-déposer dans le graphique de votre flux de travail dans le canevas. Chaque action représente un État de la tâche état.
   - L'onglet Flux fournit une liste d'états de flux que vous pouvez glisser-déposer dans le graphique de votre flux de travail dans le canevas.
   - L'onglet Patterns fournit plusieurs ready-to-use blocs de construction réutilisables que vous pouvez utiliser pour divers cas d'utilisation. Par exemple, vous pouvez utiliser ces modèles pour traiter les données de manière itérative dans un compartiment Amazon S3.
3. CanvasC'est là que vous glissez et déposez les états dans votre graphe de flux de travail, que vous modifiez l'ordre des états et que vous sélectionnez les états à configurer ou à afficher.
4. Le Inspector panneau est l'endroit où vous pouvez afficher et modifier les propriétés de n'importe quel état que vous avez sélectionné sur le canevas. Activez le bouton Définition pour afficher

le code de langue des États Amazon correspondant à votre flux de travail, l'état actuellement sélectionné étant surligné.

5. Les liens Informations ouvrent un panneau contenant des informations contextuelles lorsque vous avez besoin d'aide. Ces panneaux incluent également des liens vers des rubriques connexes dans la documentation Step Functions.
6. Barre d'outils de conception : contient un ensemble de boutons permettant d'effectuer des actions courantes, telles que l'annulation, la suppression et le zoom avant.
7. Boutons utilitaires : ensemble de boutons permettant d'effectuer des tâches, telles que l'enregistrement de vos flux de travail ou l'exportation de leurs définitions ASL dans un fichier JSON ou YAML.

Navigateur d'états

Le navigateur d'états vous permet de sélectionner les états à glisser-déposer dans votre graphe de flux de travail. L'onglet Actions fournit une liste d' AWS API, tandis que l'onglet Flow fournit une liste des états de flux. L'onglet Patterns fournit plusieurs ready-to-use blocs de construction réutilisables que vous pouvez utiliser pour divers cas d'utilisation. Vous pouvez rechercher tous les états dans le navigateur d'états à l'aide de la zone de recherche en haut.



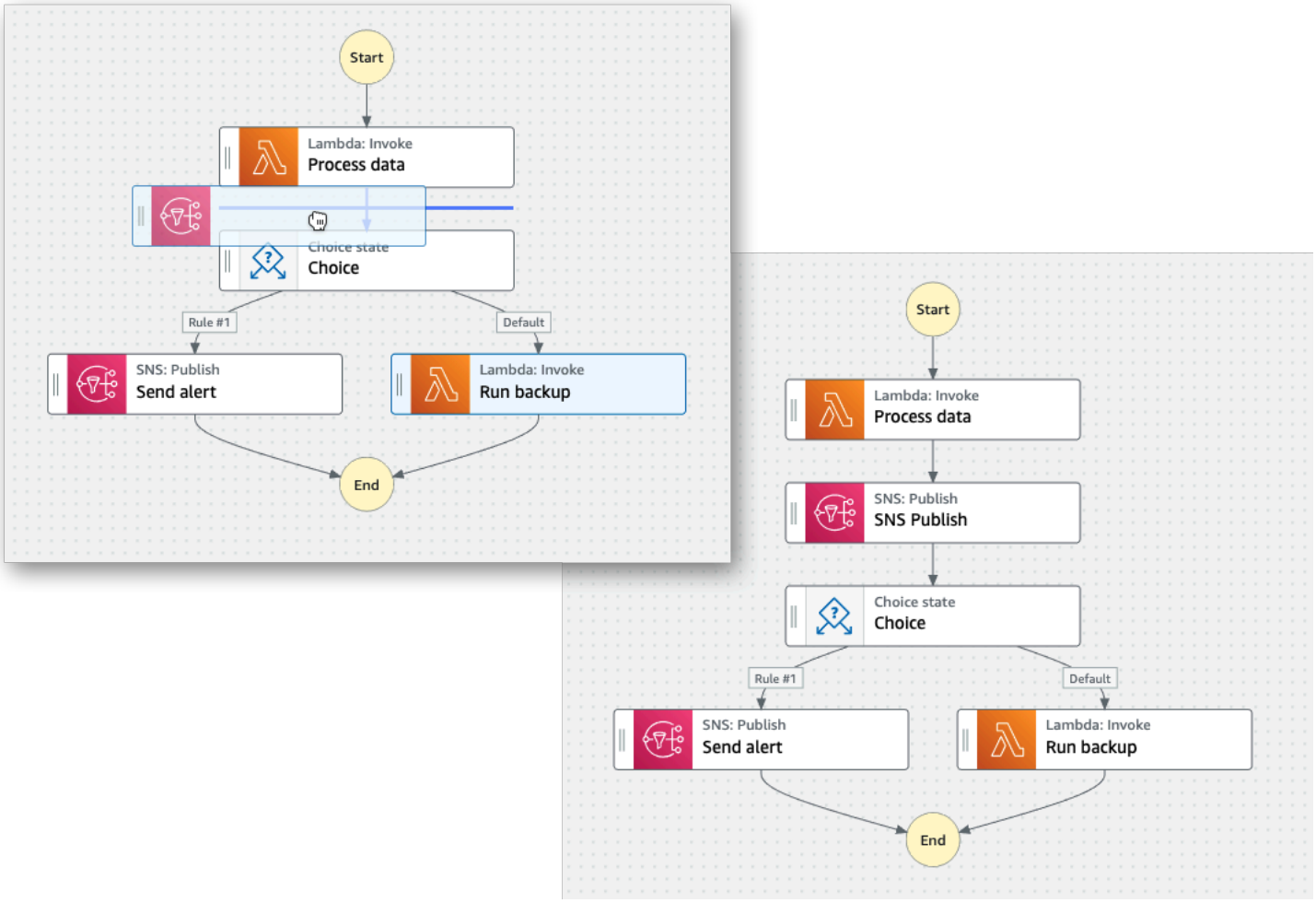
Il existe sept états de flux que vous pouvez utiliser pour diriger et contrôler votre flux de travail. Ils prennent tous des entrées de l'état précédent, et beaucoup vous permettent de filtrer les entrées de l'état précédent, et la sortie vers l'état suivant. Les états du flux sont les suivants :

- Choice: Ajoutez un choix entre les branches d'exécution à votre flux de travail. Dans l'onglet Configuration de l'Inspector, vous pouvez configurer des règles pour déterminer l'état vers lequel le flux de travail doit passer.
- Parallèle: Ajoutez des branches d'exécution parallèles à votre flux de travail.
- Map: itère dynamiquement les étapes pour chaque élément d'un tableau d'entrée. Contrairement à un état de `Parallel` flux, un `Map` état exécutera les mêmes étapes pour plusieurs entrées d'un tableau dans l'entrée d'état.
- Pass: Permet de passer son entrée à sa sortie. (Facultatif) Vous pouvez ajouter des données fixes dans la sortie.
- Attente: Faites en sorte que votre flux de travail soit suspendu pendant un certain temps ou jusqu'à une heure ou une date spécifiée.
- Succeed: arrête votre flux de travail avec succès.
- Fail: arrête votre flux de travail en cas d'échec.

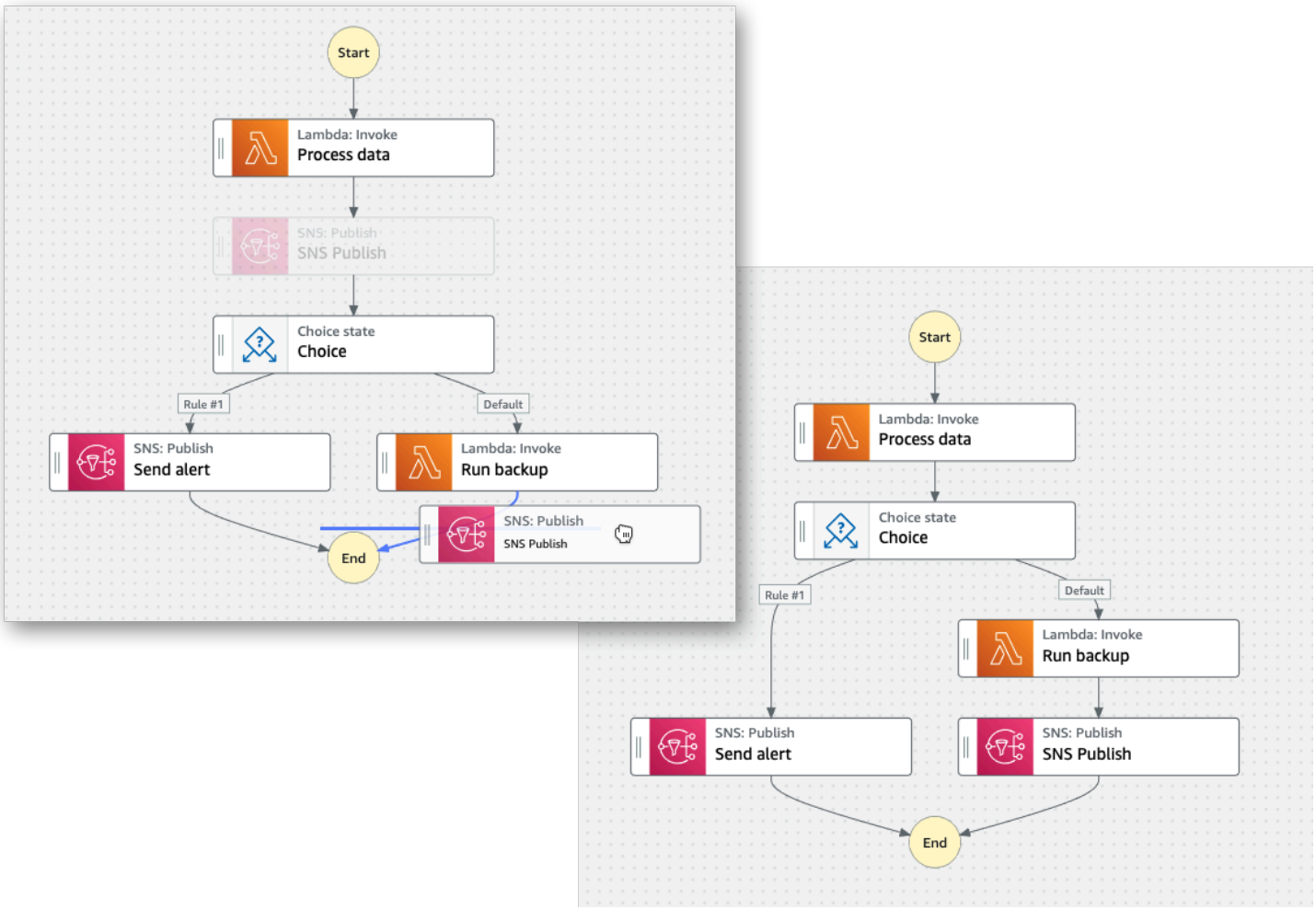
Canvas

Après avoir choisi un état à ajouter à votre flux de travail, faites-le glisser vers le canevas et déposez-le dans votre graphe de flux de travail. Vous pouvez également glisser-déposer des états pour les déplacer à différents endroits de votre flux de travail. Si votre flux de travail est complexe, il se peut que vous ne puissiez pas tout afficher dans le panneau du canevas. Utilisez les commandes situées en haut du canevas pour zoomer ou dézoomer. Pour afficher les différentes parties d'un graphique de flux de travail, vous pouvez faire glisser le graphique de flux de travail dans le canevas.

Faites glisser un état de flux de travail depuis l'onglet Actions ou Flux et déposez-le dans votre flux de travail. Une ligne indique où il sera placé dans votre flux de travail. Le nouvel état du flux de travail a été ajouté à votre flux de travail et son code est généré automatiquement.

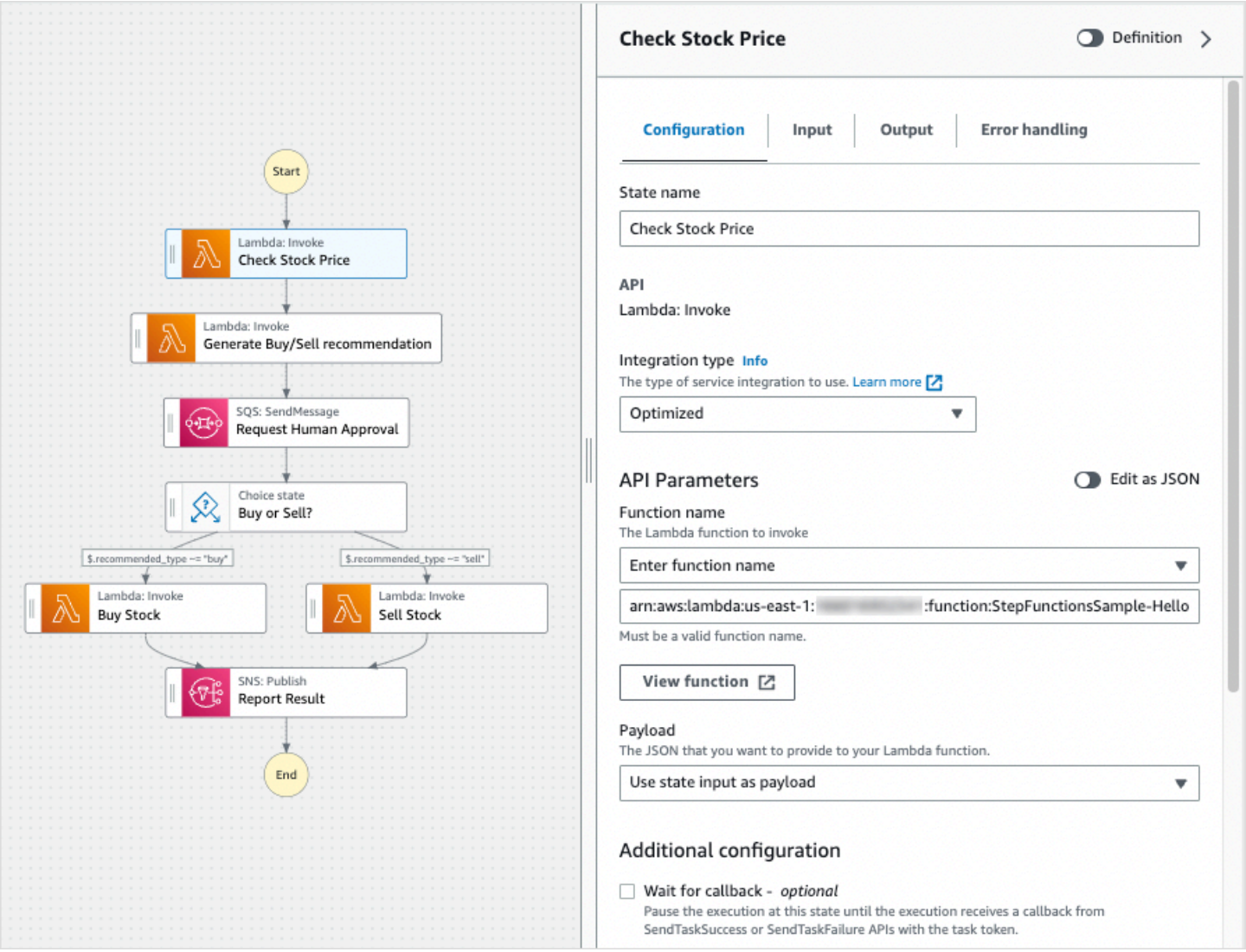


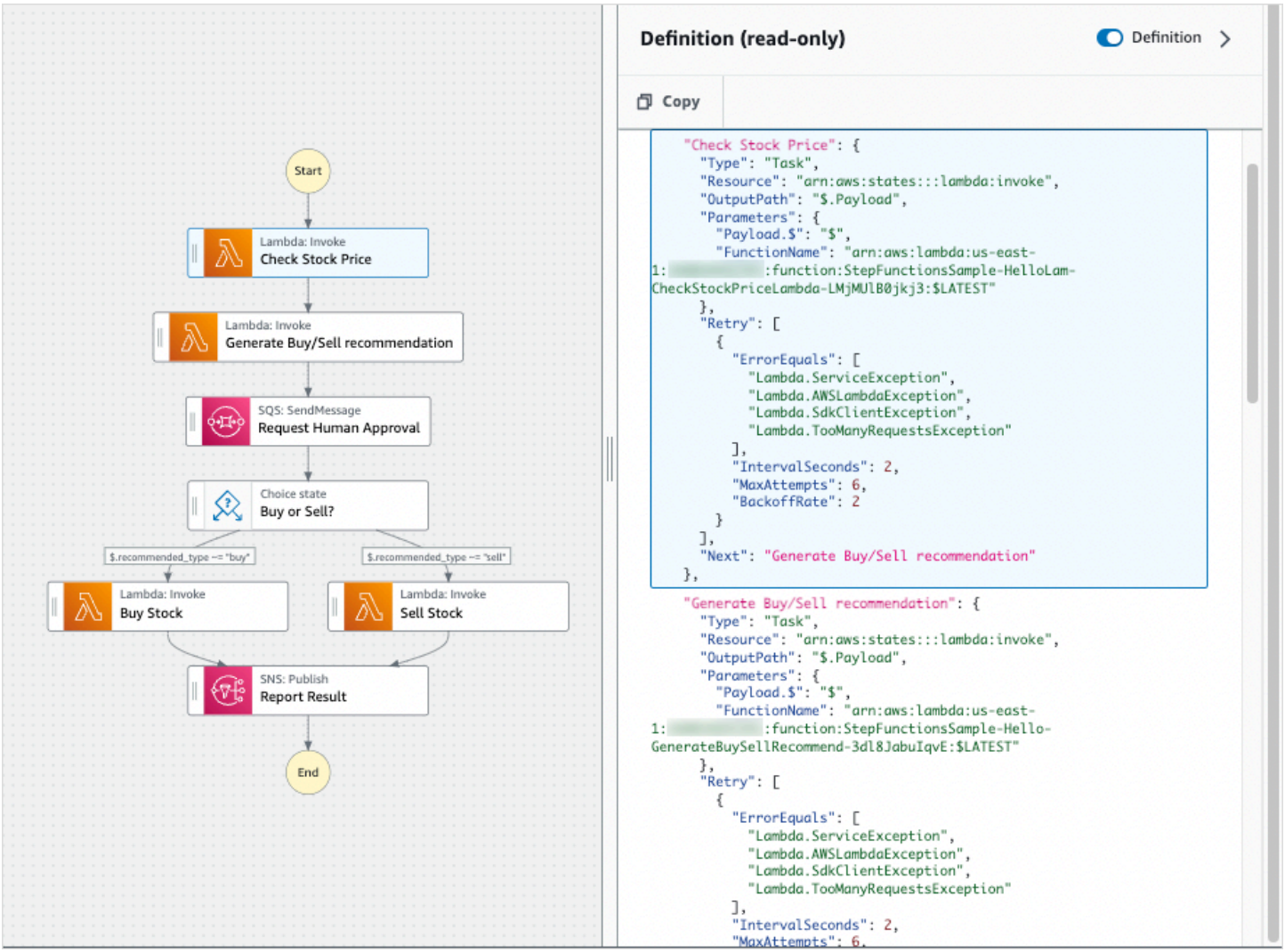
Pour modifier l'ordre d'un état, vous pouvez le faire glisser vers un autre endroit de votre flux de travail.



Inspector

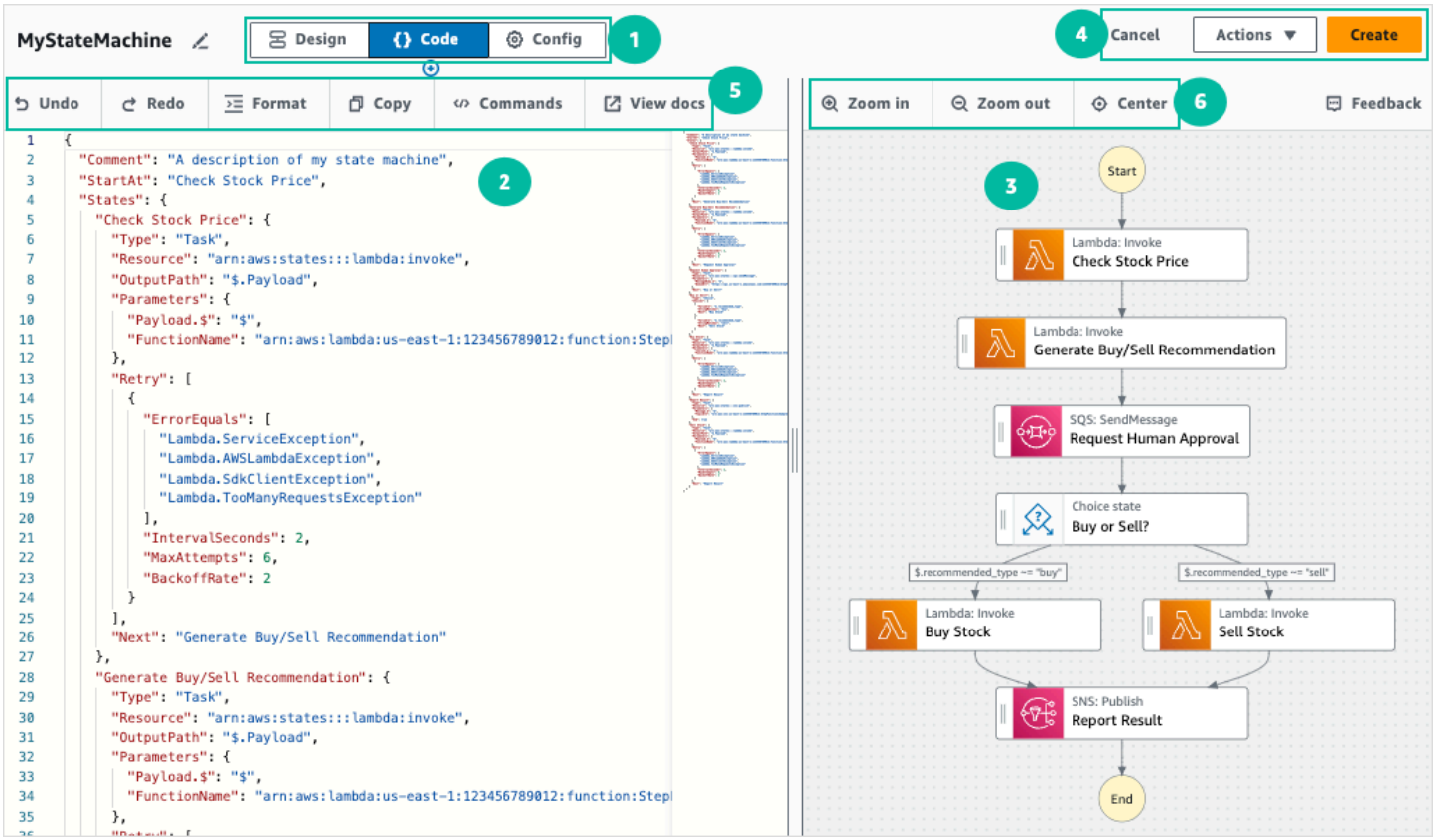
Vous pouvez configurer tous les états que vous ajoutez à votre flux de travail. Choisissez l'état que vous souhaitez configurer et vous verrez ses options de configuration dans le panneau Inspector. Pour voir la définition ASL générée automatiquement pour votre code de flux de travail, activez le bouton Définition. La définition ASL associée à l'état que vous avez sélectionné apparaîtra surlignée.


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.



Mode code

Le mode Code de Workflow Studio fournit un éditeur de code intégré pour afficher, écrire et modifier la définition Amazon States Language (ASL) de vos flux de travail dans la console Step Functions. L'image suivante montre les différents composants disponibles en mode Code.



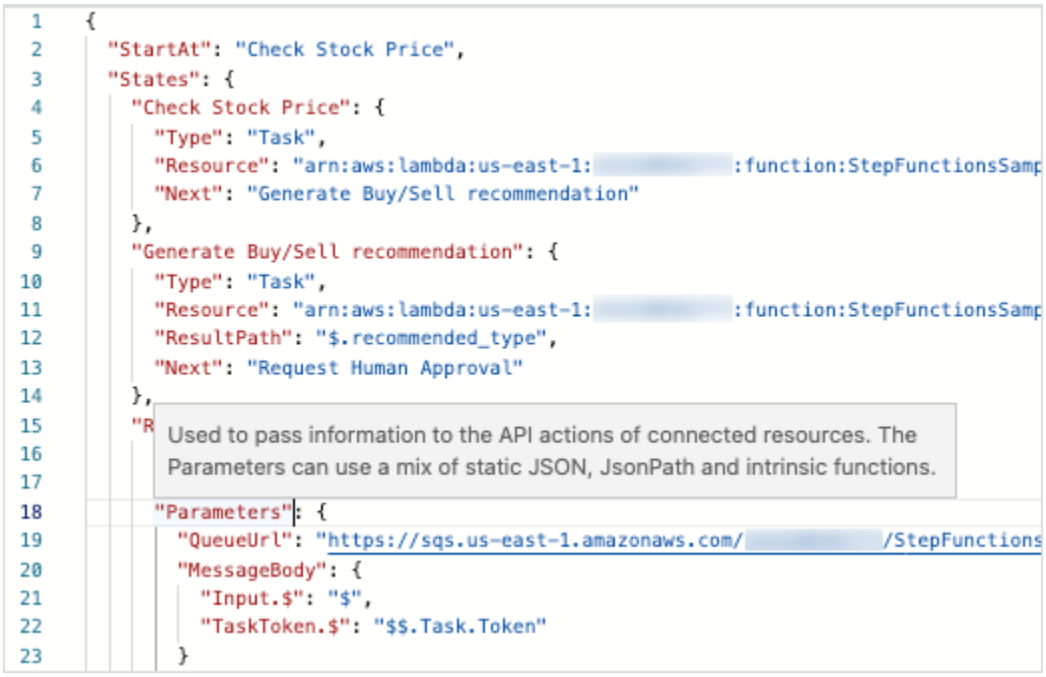
1. Boutons de mode — Basculez entre les modes Design, Code et Config du Workflow Studio à l'aide des boutons de mode. Vous ne pouvez pas changer de mode si le JSON de la définition ASL de votre flux de travail n'est pas valide.
2. Éditeur de codeC'est ici que vous écrivez et modifiez la définition ASL de vos flux de travail dans Workflow Studio. L'éditeur de code fournit également des fonctionnalités telles que la mise en évidence de la syntaxe et la saisie automatique.
3. Volet de visualisation de graphes— Affiche une visualisation graphique en temps réel de votre flux de travail.
4. Boutons utilitaires : ensemble de boutons permettant d'effectuer des tâches, telles que l'enregistrement de vos flux de travail ou l'exportation de leurs définitions ASL dans un fichier JSON ou YAML.
5. Barre d'outils de code : contient un ensemble de boutons permettant d'effectuer des actions courantes, telles que l'annulation d'une action ou le formatage du code.
6. Barre d'outils du graphique : contient un ensemble de boutons permettant d'effectuer des actions courantes, telles que le zoom avant ou arrière sur le graphe du flux de travail.

Éditeur de code

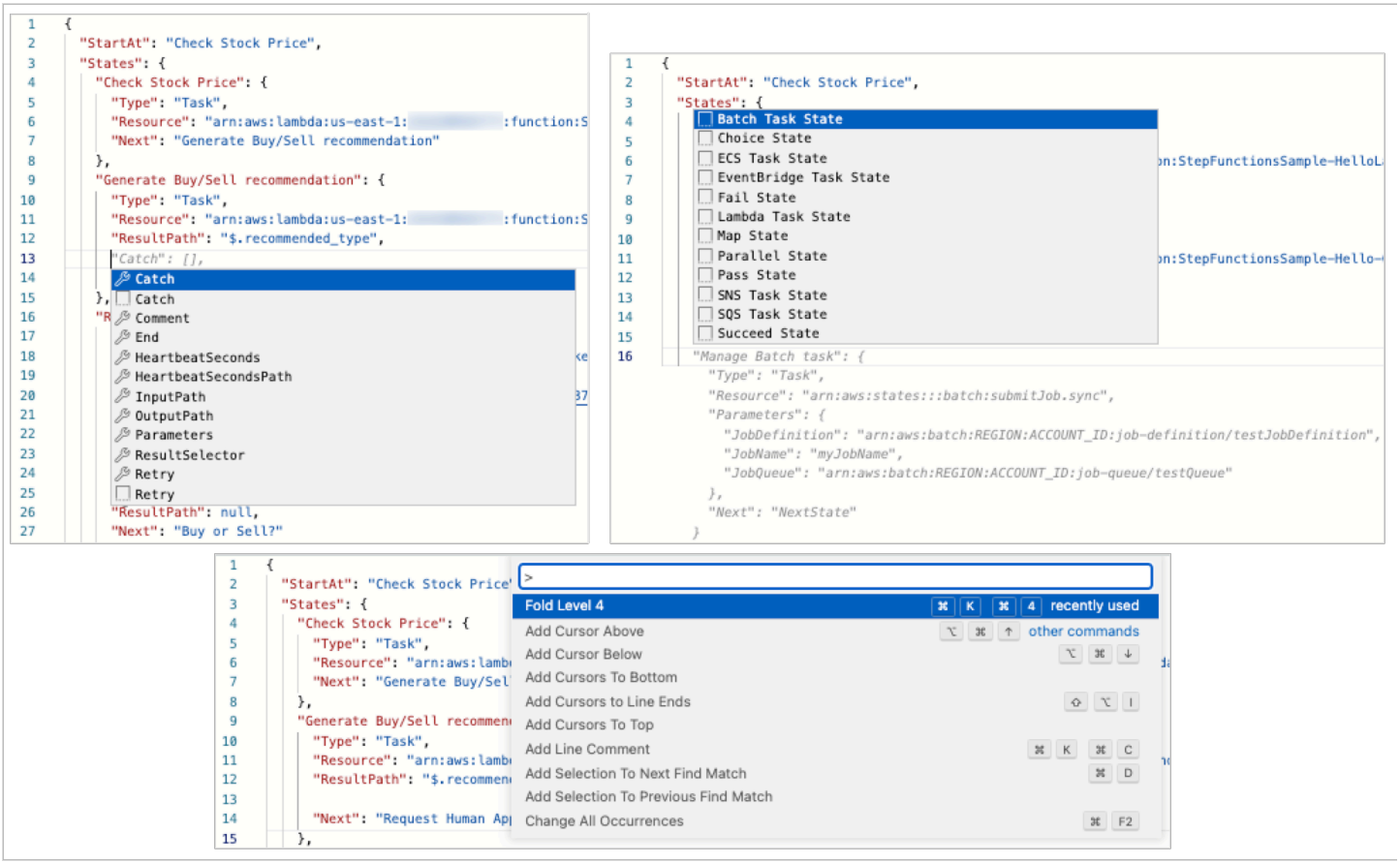
L'éditeur de code fournit une expérience de type IDE pour écrire et modifier vos définitions de flux de travail à l'aide de JSON dans Workflow Studio. L'éditeur de code inclut plusieurs fonctionnalités, telles que la mise en évidence de la syntaxe, les suggestions de saisie automatique, la validation des définitions ASL et l'affichage de l'aide contextuelle. Lorsque vous mettez à jour la définition de votre flux de travail, un graphique en temps réel de votre flux de travail Volet de visualisation de graphes s'affiche. Vous pouvez également consulter le graphique du flux de travail mis à jour dans leMode de conception.

Si vous sélectionnez un état dans le volet Mode de conception ou dans le volet de visualisation du graphe, la définition ASL de cet état apparaît surlignée dans l'éditeur de code. La définition ASL de votre flux de travail est automatiquement mise à jour si vous réorganisez, supprimez ou ajoutez un état en mode Conception ou dans le volet de visualisation du graphique.

Placez le curseur sur n'importe quel champ de la définition du flux de travail pour afficher son aide contextuelle sous forme d'infobulle.



Les suggestions de saisie automatique affichent des extraits de code pour les champs ou les états que vous pouvez inclure dans vos flux de travail. Pour voir la liste des champs que vous pouvez inclure dans un état spécifique, appuyez sur**`Ctrl+Space`**. Pour générer un extrait de code pour un nouvel état de votre flux de travail, appuyez **`Ctrl+Space`** après la définition de l'état actuel. Vous pouvez également appuyer sur **`F1`** cette touche pour afficher la liste des commandes disponibles.



Volet de visualisation de graphes

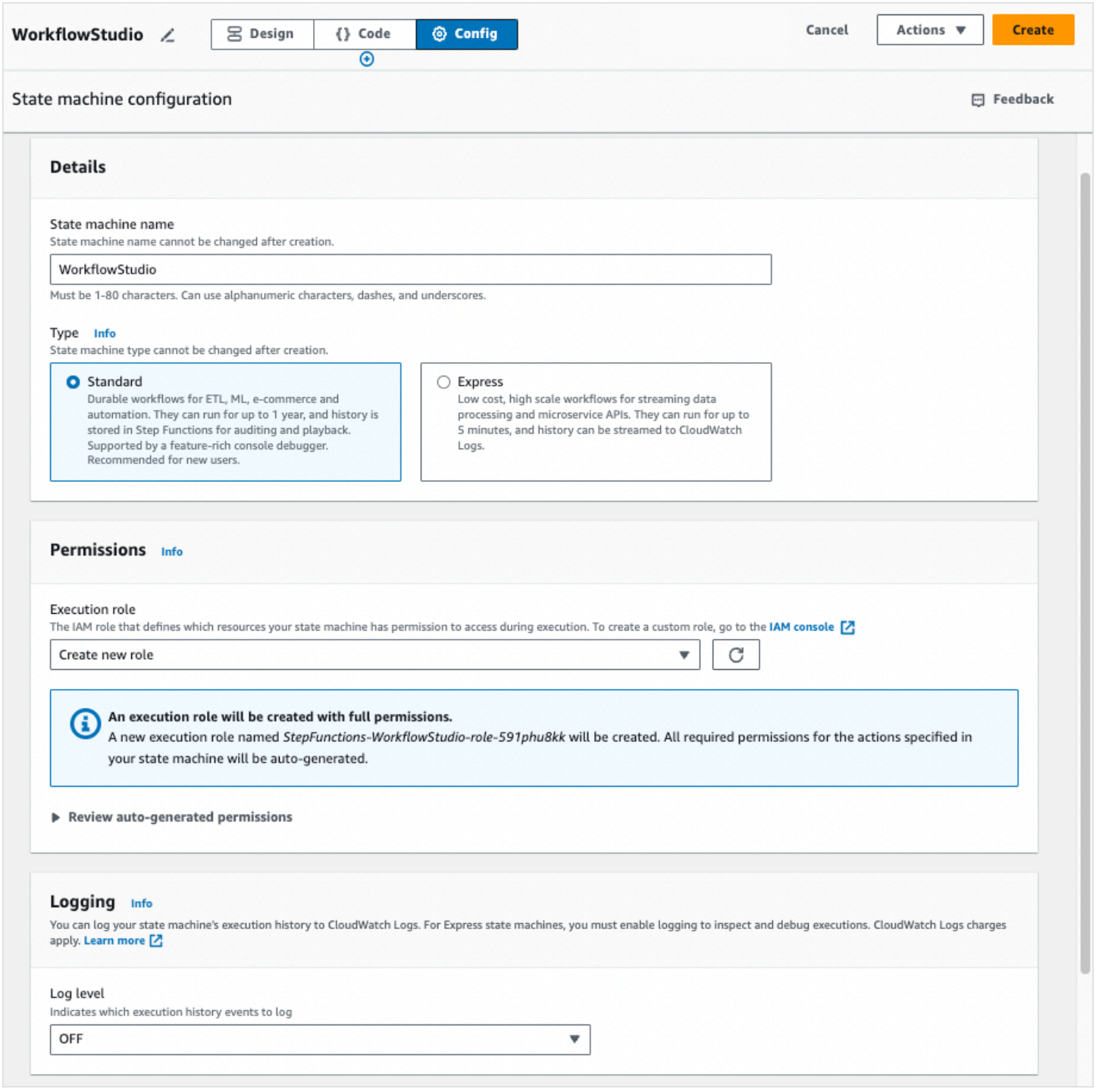
Les visualisations graphiques vous permettent de voir à quoi ressemble votre flux de travail sous forme graphique. Lorsque vous rédigez vos définitions de flux Éditeur de code de travail dans Workflow Studio, le volet de visualisation graphique affiche un graphique en temps réel de votre flux de travail. Lorsque vous réorganisez, supprimez ou dupliquez un état dans le volet de visualisation du graphique, la définition du flux de travail dans l'éditeur de code est automatiquement mise à jour. De même, lorsque vous mettez à jour vos définitions de flux de travail, que vous réorganisez, supprimez ou ajoutez un état dans l'éditeur de code, la visualisation est automatiquement mise à jour.

Si le JSON contenu dans la définition ASL de votre flux de travail n'est pas valide, le volet de visualisation du graphique interrompt le rendu et affiche un message d'état en bas du volet.

Mode Config

Le mode Config de Workflow Studio vous permet de gérer la configuration de vos machines d'état. Dans ce mode, vous pouvez spécifier des détails, tels que le nom de la machine d'état et son type, les autorisations IAM et la configuration de journalisation pour la machine d'état. Les autres

configurations supplémentaires que vous pouvez spécifier dans ce mode incluent l'activation du AWS X-Ray suivi et la publication d'une version lorsque vous créez la machine à états. Une fois que vous avez créé la machine d'état, vous pouvez modifier toutes les options de configuration de la machine d'état à l'exception du nom et du type de machine d'état. L'image suivante montre certaines des configurations que vous pouvez spécifier en mode Config.



Gérer la configuration de la machine à états

Pour gérer la configuration de votre machine à états, procédez comme suit :

1. Entrez le nom de votre machine à états dans le champ Nom de la machine à états.

> **Tip**
>
> Vous pouvez également choisir l'icône d'édition à côté du nom de la machine à états par défaut de MyStateMachine. Ensuite, sous Configuration de la machine d'état, spécifiez un nom.

> **Important**
>
> Vous ne pouvez pas modifier le nom de la machine à états une fois que vous l'avez créée.

2. Dans Type, choisissez un type de machine à états Standard ou Express. Pour plus d'informations sur les types de machines à états, consultezFlux de travail standard ou express.

> **Important**
>
> Vous ne pouvez pas modifier le type de machine à états une fois que vous l'avez créée.

3. Dans Autorisations, sélectionnez le rôle IAM à utiliser comme rôle d'exécution pour la machine d'état.
   - Créer un nouveau rôle (recommandé) : Si vous sélectionnez cette option, Step Functions crée automatiquement un rôle d'exécution pour vos machines d'état avec le moins de privilèges requis lorsque vous créez les machines d'état. Ces rôles IAM générés automatiquement sont valides pour la machine à états Région AWS dans laquelle vous créez la machine à états.

> **Tip**
>
> Pour vérifier les autorisations que Step Functions générera automatiquement pour votre machine à états, choisissez Revoir les autorisations générées automatiquement.

> **Note**
>
> Si vous supprimez le rôle IAM créé par Step Functions, Step Functions ne pourra pas le recréer ultérieurement. De même, si vous modifiez le rôle (par exemple, en supprimant Step Functions des principes de la politique IAM), Step Functions ne pourra pas restaurer ses paramètres d'origine ultérieurement.

- Choisissez un rôle existant : créez votre propre rôle IAM pour la machine d'état, puis choisissez-le parmi les options répertoriées ci-dessous Choisissez un rôle existant. Assurez-vous que la politique du rôle inclut les autorisations que vous souhaitez que la machine à états assume.

  Pour obtenir des informations sur la création de politiques IAM, veuillez consulter Création de politiques IAM dans le Guide de l'utilisateur IAM.

- Entrez un ARN de rôle : Spécifiez le nom de ressource Amazon (ARN) d'un rôle IAM existant à utiliser pour cette machine à états. Par exemple, `arn:aws:iam::123456789012:role/service-role/StepFunctions-WorkflowStudio-role-777f4027`.

4. Dans Logging, définissez le niveau de journalisation de votre machine d'état. Step Functions enregistre les événements de l'historique d'exécution en fonction de votre sélection. Vous pouvez sélectionner l'une des options suivantes :

   - TOUS : Tous les types d'événements sont enregistrés.
   - ERREUR : tous les types d'événements d'erreur sont enregistrés, tels que TaskFailed et ExecutionFailed.
   - FATAL : tous les types d'événements d'erreur fatale sont enregistrés, tels que ExecutionAborted et ExecutionFailed.
   - OFF : aucun type d'événement n'est enregistré.

   Pour plus d'informations sur les niveaux de journalisation, consultezNiveaux de journalisation.

5. Dans Configuration supplémentaire, définissez une ou plusieurs des configurations facultatives suivantes :

   - Activer le X-Ray suivi : cochez cette case pour que Step Functions envoie des traces X-Ray pour les exécutions par State Machine, même si aucun identifiant de trace n'est transmis par un service en amont. Pour plus d'informations, consultez AWS X-Ray et Step Functions.
   - Publier la version lors de sa création : une version est un instantané numéroté et immuable d'une machine à états que vous pouvez exécuter. Cochez cette case pour publier une version

de votre machine à états lors de la création de la machine à états. Step Functions publie la version 1 en tant que première révision de la machine à états.

Pour plus d'informations sur les versions, consultez Versions des machines à états.

- Ajouter un nouveau tag : cochez cette case pour ajouter des tags à votre machine d'état. L'ajout de balises peut vous aider à suivre et à gérer les coûts associés à vos ressources et à renforcer la sécurité de vos politiques IAM. Pour en savoir plus sur les identifications, consultez Fonctions de balisage en étapes.

6. Choisissez Créer.
7. Dans la boîte de dialogue Confirmer la création du rôle, choisissez Confirmer pour continuer.

Vous pouvez également choisir Afficher la configuration des rôles pour revenir au mode Config.

Raccourcis clavier

Workflow Studio prend en charge les raccourcis clavier suivants :

Raccourci clavier	Fonction
Shortcuts for the Code mode	
Ctrl+space	Auto-complete suggestions
F1	Display a list of available commands
Common shortcuts for the Design and Code modes	
Ctrl+Z	Undo the last operation
Ctrl+Shift+Z	Redo the last operation
Alt+C	Center the workflow in the canvas
Backspace	Remove all selected states
Delete	Remove all selected states
Ctrl+D	Duplicate selected state

Utilisation de Workflow Studio

Apprenez à créer, modifier et exécuter des flux de travail à l'aide de Step Functions Workflow Studio. Une fois que votre flux de travail est prêt, vous pouvez l'exporter. Vous pouvez également utiliser Workflow Studio pour le prototypage rapide.

Dans cette rubrique

Création d'un flux de travail

Dans Workflow Studio, vous pouvez soit choisir un modèle de démarrage, soit choisir un modèle vierge pour créer un flux de travail à partir de zéro. Pour les modèles vierges, vous pouvez utiliser le mode Design ou Code pour créer votre flux de travail.

Un modèle de démarrage est un ready-to-run exemple de projet qui crée automatiquement le prototype et la définition du flux de travail, et déploie toutes les AWS ressources connexes dont votre projet a besoin sur votre compte. Compte AWS Vous pouvez utiliser ces modèles de démarrage pour les déployer et les exécuter tels quels, ou utiliser les prototypes de flux de travail pour les exploiter. Pour plus d'informations sur les modèles de démarrage, consultezExemples de projets pour Step Functions.

Création d'un flux de travail à l'aide de modèles de démarrage

1. Ouvrez la console Step Functions et choisissez Create state machine.
2. Dans la boîte de dialogue Choisir un modèle, effectuez l'une des opérations suivantes pour choisir un exemple de projet, par exemple l'exemple de projet Task Timer :

   - Tapez **`Task Timer`** dans la zone Rechercher par mot clé, puis choisissez Task Timer dans les résultats de recherche renvoyés.

- Parcourez les exemples de projets répertoriés sous Tous dans le volet droit, puis choisissez Task Timer.

3. Choisissez Next (Suivant) pour continuer.
4. Step Functions répertorie les Services AWS éléments utilisés dans l'exemple de projet que vous avez sélectionné. Il montre également un graphique de flux de travail pour l'exemple de projet. Déployez ce projet sur votre site Compte AWS ou utilisez-le comme point de départ pour créer vos propres projets. Selon la façon dont vous souhaitez procéder, choisissez Exécuter une démo ou Construire à partir de celle-ci.
5. Choisissez Utiliser le modèle pour poursuivre votre sélection.
6. Effectuez l'une des actions suivantes :

   - Si vous avez sélectionné Build on it, Step Functions crée le prototype de flux de travail pour l'exemple de projet que vous avez sélectionné. Step Functions ne déploie pas les ressources répertoriées dans la définition du flux de travail. DansMode de conception, glissez et déposez les états depuis le Navigateur d'états pour continuer à créer votre prototype de flux de travail. Ou passez au Mode code pour mettre à jour la définition Amazon States Language (ASL) de votre flux de travail.

     > **Important**
     >
     > N'oubliez pas de mettre à jour l'espace réservé Amazon Resource Name (ARN) pour les ressources utilisées dans l'exemple de projet avant d'exécuter votre flux de travail.

   - Si vous avez sélectionné Run a demo, Step Functions crée un exemple de projet en lecture seule qui utilise un AWS CloudFormation modèle pour déployer les AWS ressources répertoriées dans ce modèle sur votre. Compte AWS

     > **Tip**
     >
     > Pour afficher la définition de la machine à états de l'exemple de projet, choisissez Code.

     Lorsque vous êtes prêt, choisissez Déployer et exécuter pour déployer l'exemple de projet et créer les ressources.

> **Note**
>
> La création de ces ressources et des autorisations IAM associées peut prendre jusqu'à 10 minutes. Pendant le déploiement de vos ressources, vous pouvez ouvrir le lien CloudFormation Stack ID pour voir quelles ressources sont mises en service.

> **Important**
>
> Les frais standard s'appliquent pour chaque service utilisé dans le CloudFormation modèle.

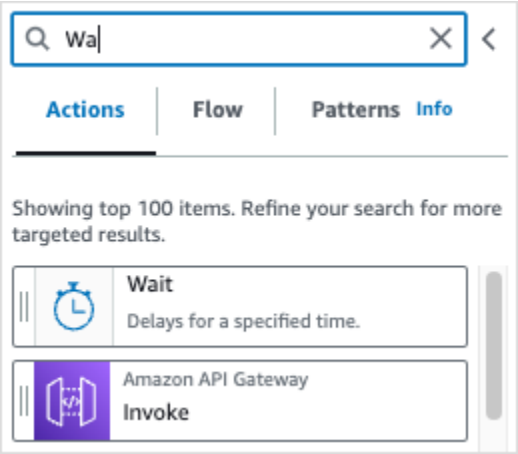
Création d'un flux de travail à l'aide d'un modèle vierge

1. Ouvrez la console Step Functions.
2. Choisissez Create state machine (Créer une machine d'état).
3. Dans la boîte de dialogue Choisir un modèle, sélectionnez Vide.
4. Choisissez Select (Sélectionner). Cela ouvre Workflow Studio dansMode de conception.

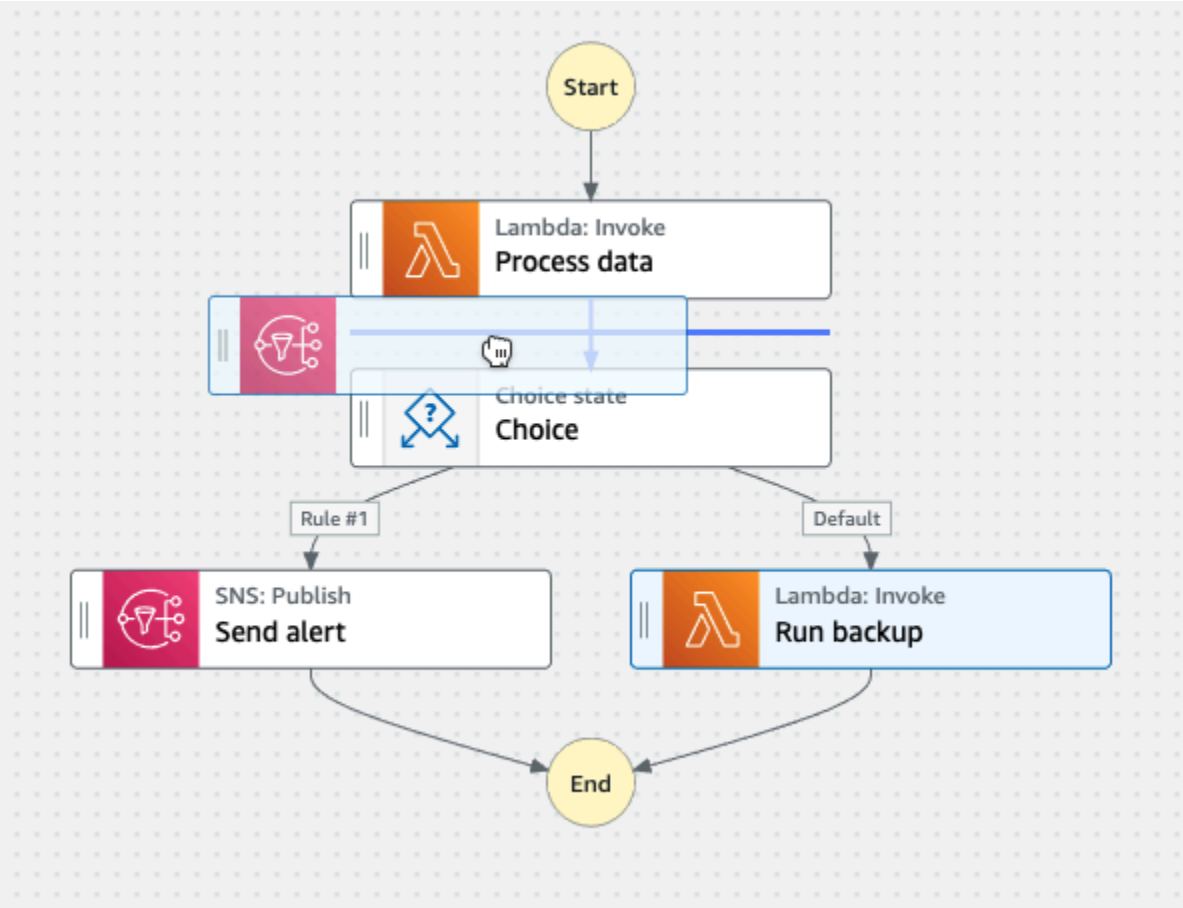
   Vous pouvez désormais commencer à concevoir votre flux de travail Mode de conception ou à écrire votre définition de flux de travail dansMode code.
5. Choisissez Config pour gérer la configuration de votre flux de travail dans leMode Config. Par exemple, donnez un nom à votre flux de travail et choisissez son type.

Concevoir un flux de travail

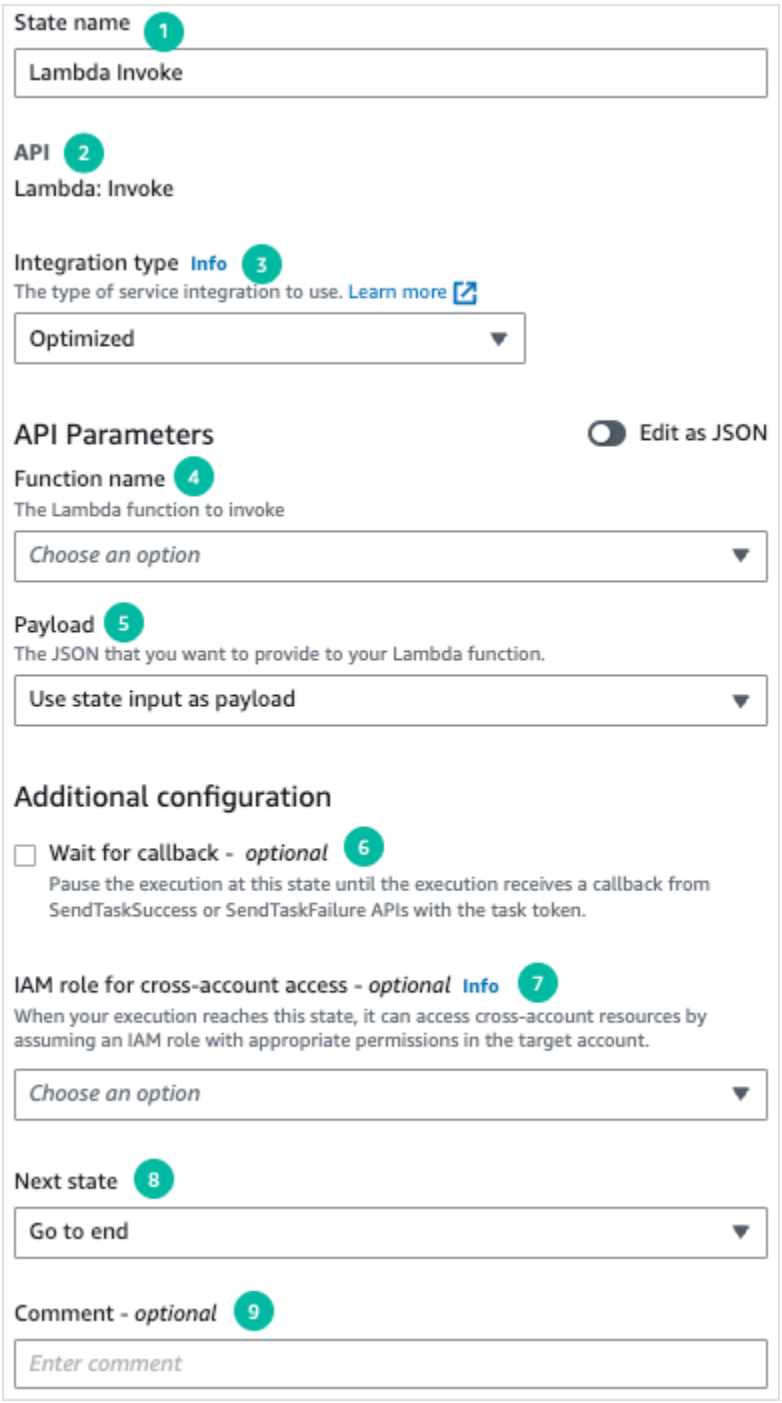
Si vous connaissez le nom de l'état que vous souhaitez ajouter, utilisez le champ de recherche en haut du Navigateur d'états pour trouver cet état dans les onglets Actions et Flux duMode de conception.



Sinon, choisissez un état dans le navigateur d'états et faites-le glisser sur le canevas, en le plaçant où vous le souhaitez dans votre flux de travail. Vous pouvez également réorganiser les états de votre flux de travail en les faisant glisser vers un autre emplacement de votre flux de travail. Lorsque vous faites glisser un état sur le canevas, une ligne apparaît à l'endroit où vous pouvez le déposer dans votre flux de travail. Une fois qu'un état est déposé sur le canevas, son code est généré automatiquement et ajouté dans la définition de votre flux de travail. Pour voir la définition, activez le bouton Définition dans le panneau Inspector. Pour apporter des modifications à la définition de votre flux de travail, choisissez celui Mode code qui propose un éditeur de code intégré.



Après avoir déposé un état sur le canevas, vous pouvez le configurer dans le Inspector panneau de droite. Ce panneau contient les onglets Configuration, Entrée, Sortie et Gestion des erreurs pour chaque état ou action d'API que vous placez sur le canevas. Vous configurez les états que vous incluez dans vos flux de travail dans l'onglet Configuration. Par exemple, l'onglet Configuration de l'action d'API Lambda Invoke comprend les options suivantes :



1. Le nom de l'État identifie l'État. Vous pouvez utiliser votre propre nom ou accepter le nom généré par défaut.
2. L'API montre l'action d'API utilisée par l'État.
3. La liste déroulante des types d'intégration propose des options permettant de choisir le type d'intégrations de services disponibles dans Step Functions. Le type d'intégration que vous

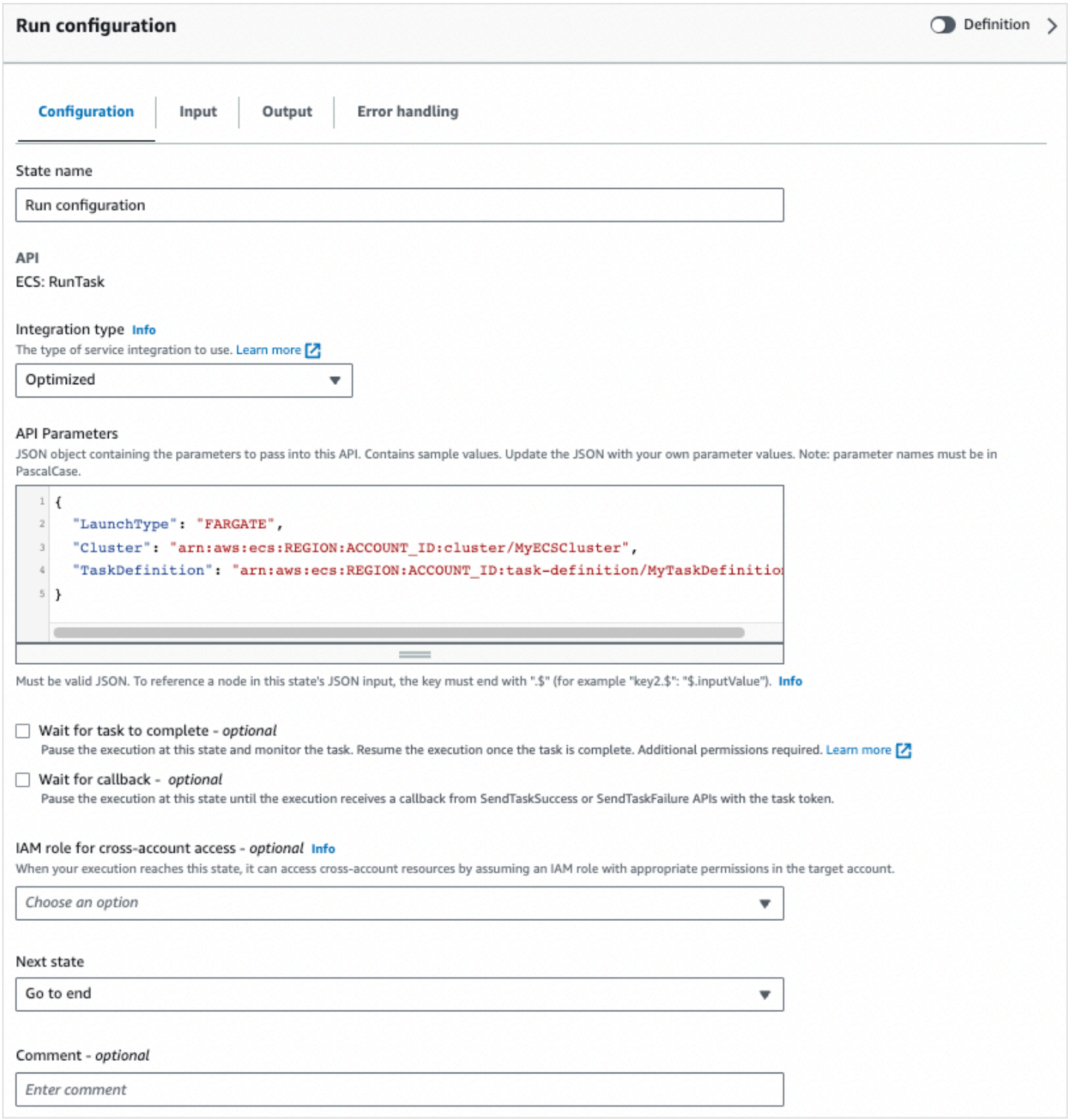
choisissez est utilisé pour appeler les actions d'API d'un élément spécifique Service AWS de votre flux de travail.

4. Le nom de la fonction fournit des options permettant de :
   - Entrez le nom de la fonction : vous pouvez saisir le nom de votre fonction ou son ARN.
   - Obtenir le nom de la fonction au moment de l'exécution à partir de l'entrée d'état : vous pouvez utiliser cette option pour obtenir dynamiquement le nom de la fonction à partir de l'entrée d'état en fonction du chemin que vous spécifiez.
   - Sélectionnez le nom de la fonction : vous pouvez sélectionner directement l'une des fonctions disponibles dans votre compte et dans votre région.
5. La charge utile vous permet de sélectionner l'une des options suivantes :
   - Utiliser l'entrée d'état comme charge utile : vous pouvez utiliser cette option pour transmettre l'entrée de l'état en tant que charge utile fournie à votre fonction Lambda.
   - Entrez votre propre charge utile : vous pouvez utiliser cette option pour créer un objet JSON à transmettre en tant que charge utile à votre fonction Lambda. Ce JSON peut inclure à la fois des valeurs statiques et des valeurs sélectionnées à partir de l'entrée d'état.
   - Aucune charge utile : vous pouvez utiliser cette option si vous ne souhaitez pas transmettre de charge utile à votre fonction Lambda.
6. (Facultatif) Certains États auront la possibilité de sélectionner Attendre la fin de la tâche ou Attendre le rappel. Lorsqu'elles sont disponibles, ces options sélectionnent l'un des modèles d'intégration de services suivants :
   - Aucune option sélectionnée : Step Functions utilisera le modèle Réponse à la requête d'intégration. Step Functions attendra une réponse HTTP, puis passera à l'état suivant. Step Functions n'attendra pas qu'une tâche soit terminée. Lorsqu'aucune option n'est disponible, l'État utilise ce modèle.
   - Attendez que la tâche soit terminée : Step Functions utilisera le modèle Exécuter une tâche (.sync) d'intégration.
   - Attendez le rappel : Step Functions utilisera le modèle Attendre un rappel avec le jeton de tâche d'intégration.
7. (Facultatif) Pour accéder aux ressources configurées de différentes manières au Comptes AWS sein de vos flux de travail, Step Functions fournit un accès entre comptes. Le rôle IAM pour l'accès entre comptes fournit des options pour :
   - Fournir l'ARN du rôle IAM : Spécifiez le rôle IAM qui contient les autorisations d'accès aux ressources appropriées. Ces ressources sont disponibles sur un compte cible, qui est un compte Compte AWS vers lequel vous passez des appels entre comptes.

- Obtenir l'ARN du rôle IAM au moment de l'exécution à partir de l'entrée d'état : Spécifiez un chemin de référence vers une paire clé-valeur existante dans l'entrée JSON de l'État qui contient le rôle IAM.

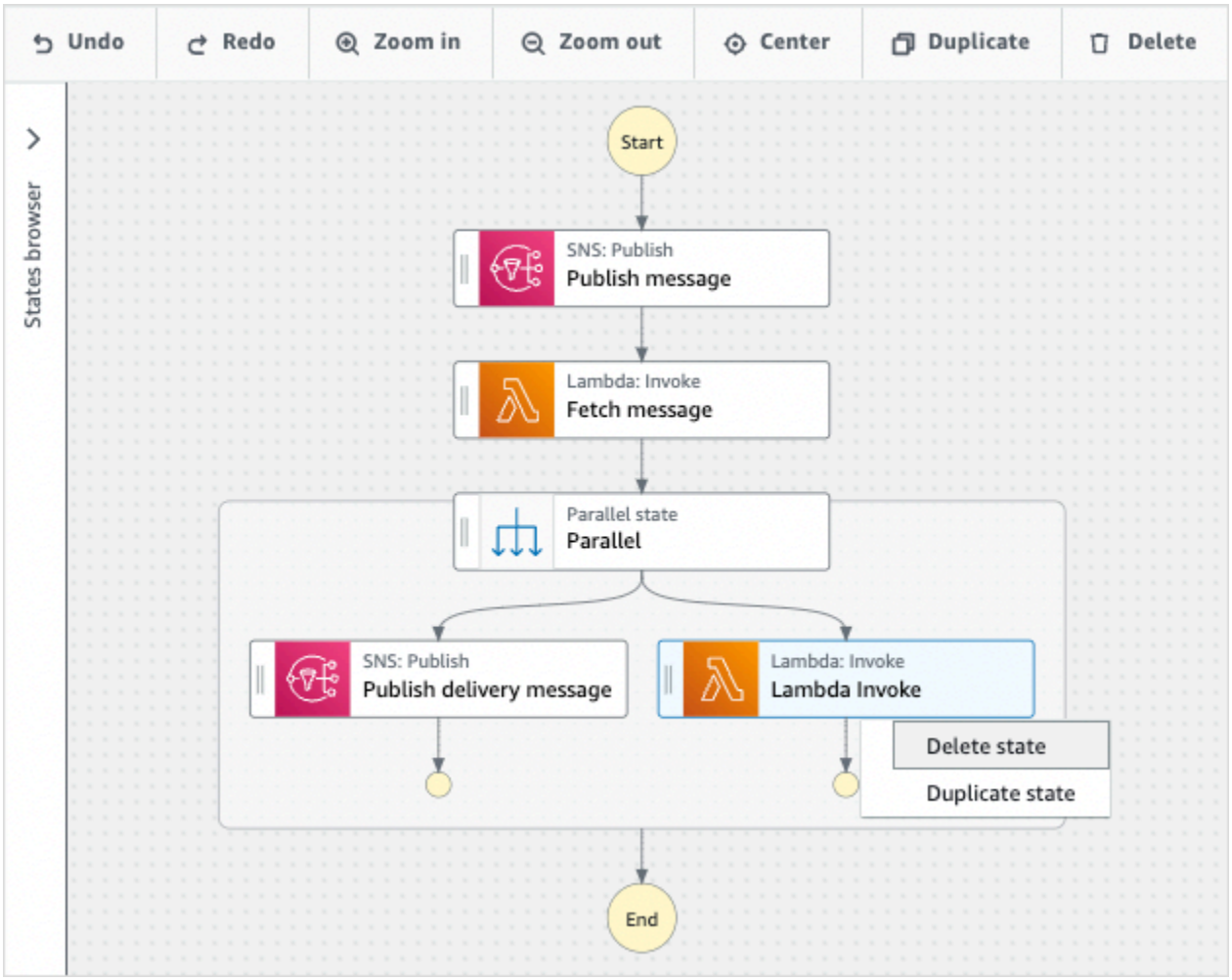
8. L'état suivant vous permet de sélectionner l'état vers lequel vous souhaitez passer au suivant.
9. (Facultatif) Le champ Commentaire peut être utilisé pour ajouter votre propre commentaire. Cela n'affectera pas le flux de travail, mais peut être utilisé pour annoter votre flux de travail.

Certains États auront des options de configuration plus génériques. Par exemple, la configuration de `RunTask` l'état d'Amazon ECS contient un `API Parameters` champ rempli de valeurs d'espace réservé.



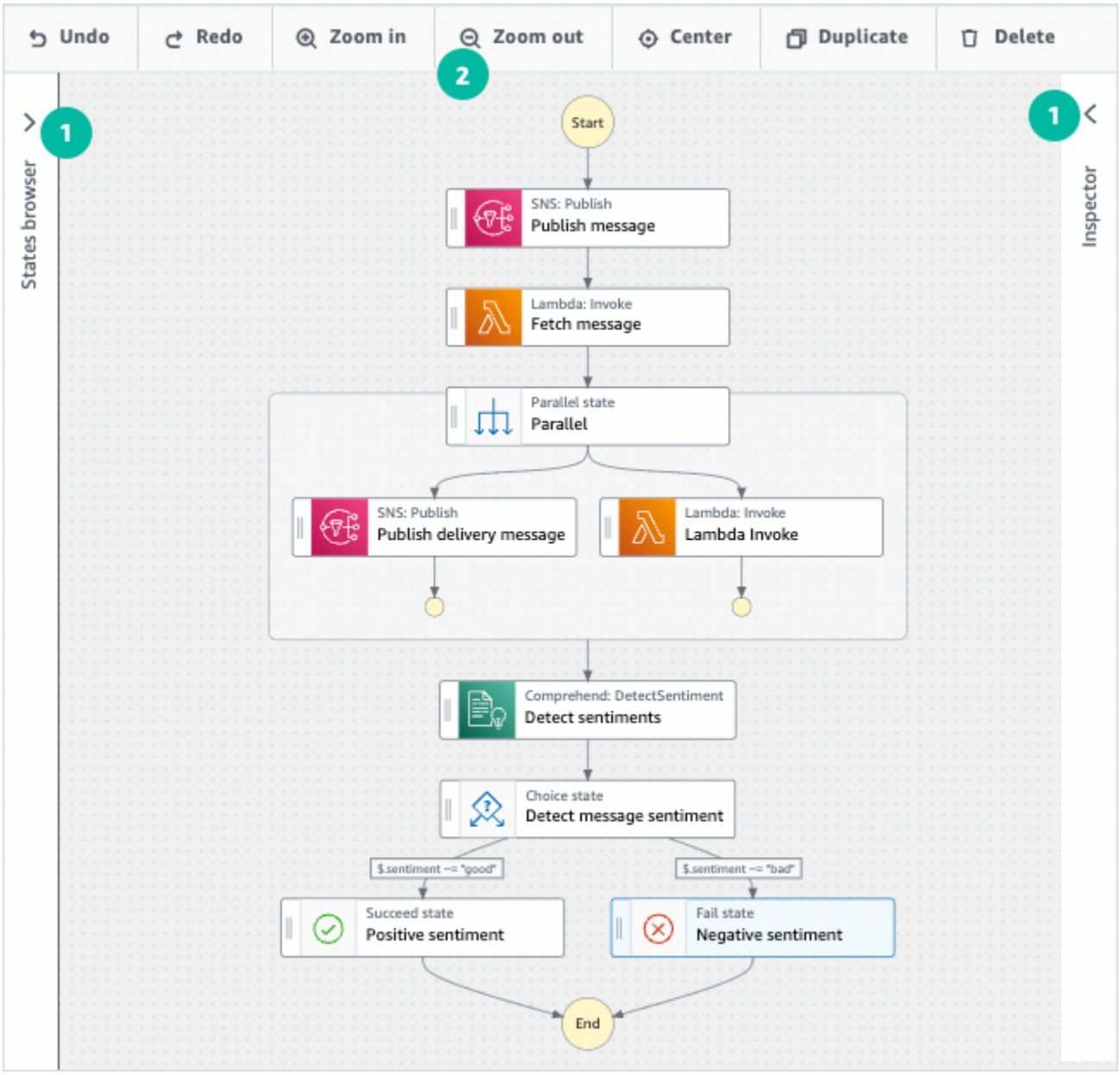
Pour ces états, vous pouvez remplacer les valeurs d'espace réservé par des configurations adaptées à vos besoins.

Pour supprimer un état, vous pouvez utiliser le backspace, cliquer avec le bouton droit de la souris et choisir Supprimer l'état, ou choisir Supprimer dans la barre d'outils de conception.



Au fur et à mesure que votre flux de travail se développe, il se peut qu'il ne rentre pas dans le canevas. Vous pouvez :

1. Utilisez les commandes situées sur les panneaux latéraux pour redimensionner ou fermer les panneaux.
2. Utilisez les commandes de la barre d'outils Design situées en haut du Canvas pour zoomer ou dézoomer le graphique du flux de travail.



Exécutez votre flux de travail

Après avoir créé ou modifié votre flux de travail avec Workflow Studio, vous pouvez l'exécuter et visualiser son exécution dans la console Step Functions.

Pour exécuter un flux de travail dans Workflow Studio

1. En mode Design, Code ou Config, choisissez Execute.

   La boîte de dialogue Démarrer l'exécution s'ouvre dans un nouvel onglet.
2. Dans la boîte de dialogue Démarrer l'exécution, procédez comme suit :

1. (Facultatif) Pour identifier votre exécution, vous pouvez lui donner un nom dans le champ Nom. Par défaut, Step Functions génère automatiquement un nom d'exécution unique.

   > **Note**
   >
   > Step Functions vous permet de créer des noms pour les machines d'état, les exécutions et les activités, ainsi que des étiquettes contenant des caractères non ASCII. Ces noms non ASCII ne fonctionnent pas avec Amazon. CloudWatch Pour être sûr de pouvoir suivre CloudWatch les métriques, choisissez un nom qui utilise uniquement des caractères ASCII.

2. (Facultatif) Dans la zone de saisie, entrez les valeurs d'entrée au format JSON pour exécuter votre flux de travail.
3. Choisissez Start execution (Démarrer l'exécution).
4. La console Step Functions vous dirige vers une page intitulée avec votre ID d'exécution. Cette page est connue sous le nom de page Détails de l'exécution. Sur cette page, vous pouvez consulter les résultats de l'exécution au fur et à mesure que l'exécution progresse ou une fois celle-ci terminée.

   Pour consulter les résultats de l'exécution, choisissez des états individuels dans la vue graphique, puis choisissez les onglets individuels du Détails de l'étape volet pour afficher les détails de chaque état, y compris les entrées, les sorties et la définition respectivement. Pour plus de détails sur les informations d'exécution que vous pouvez consulter sur la page Détails de l'exécution, voirPage de détails d'exécution — Vue d'ensemble de l'interface.

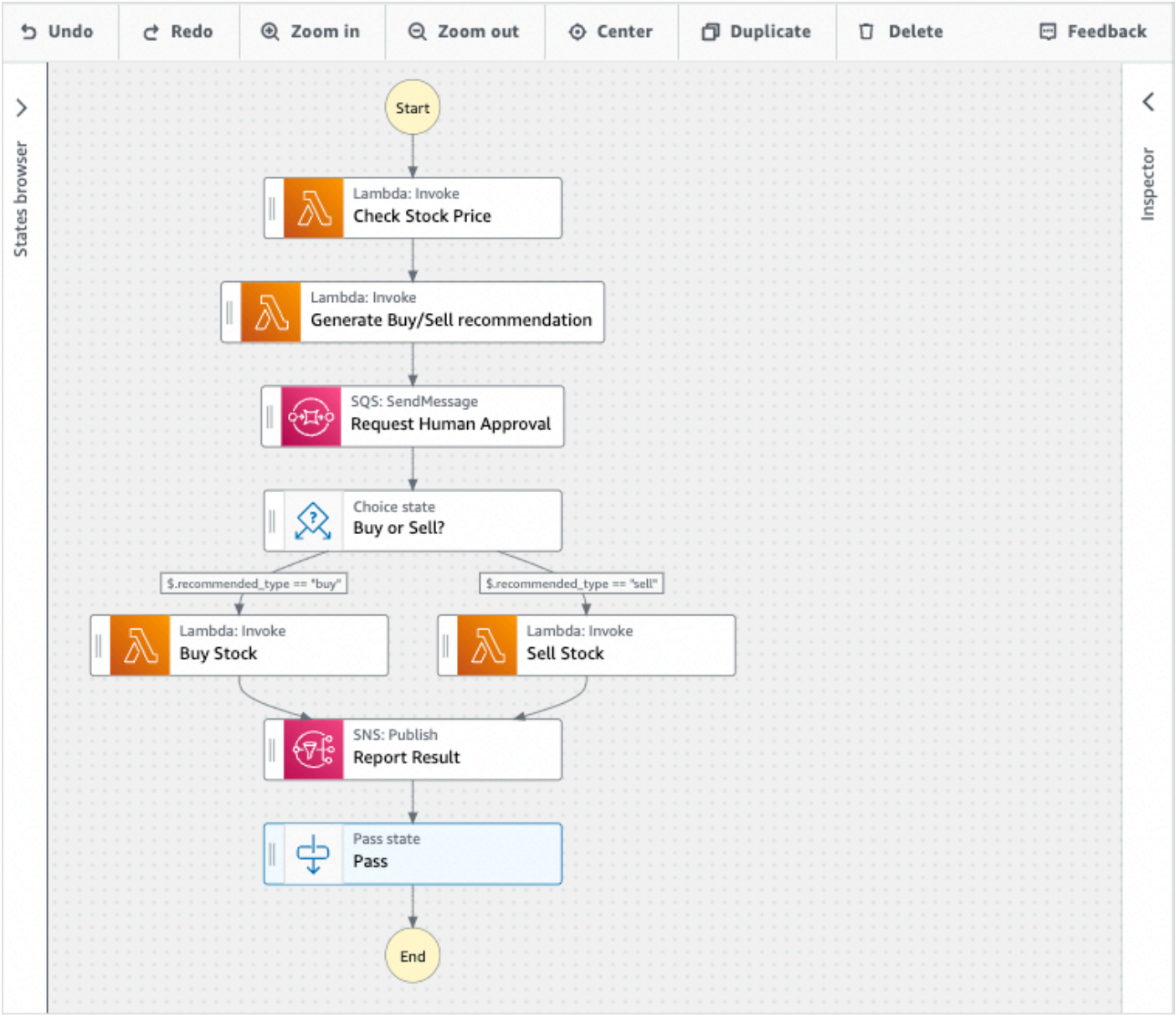
Modifiez votre flux de travail

Vous pouvez modifier visuellement un flux de travail existant dans Mode de conception Workflow Studio. Vous pouvez également modifier la définition du flux Mode code de travail dans Workflow Studio.

Pour modifier un flux de travail existant :

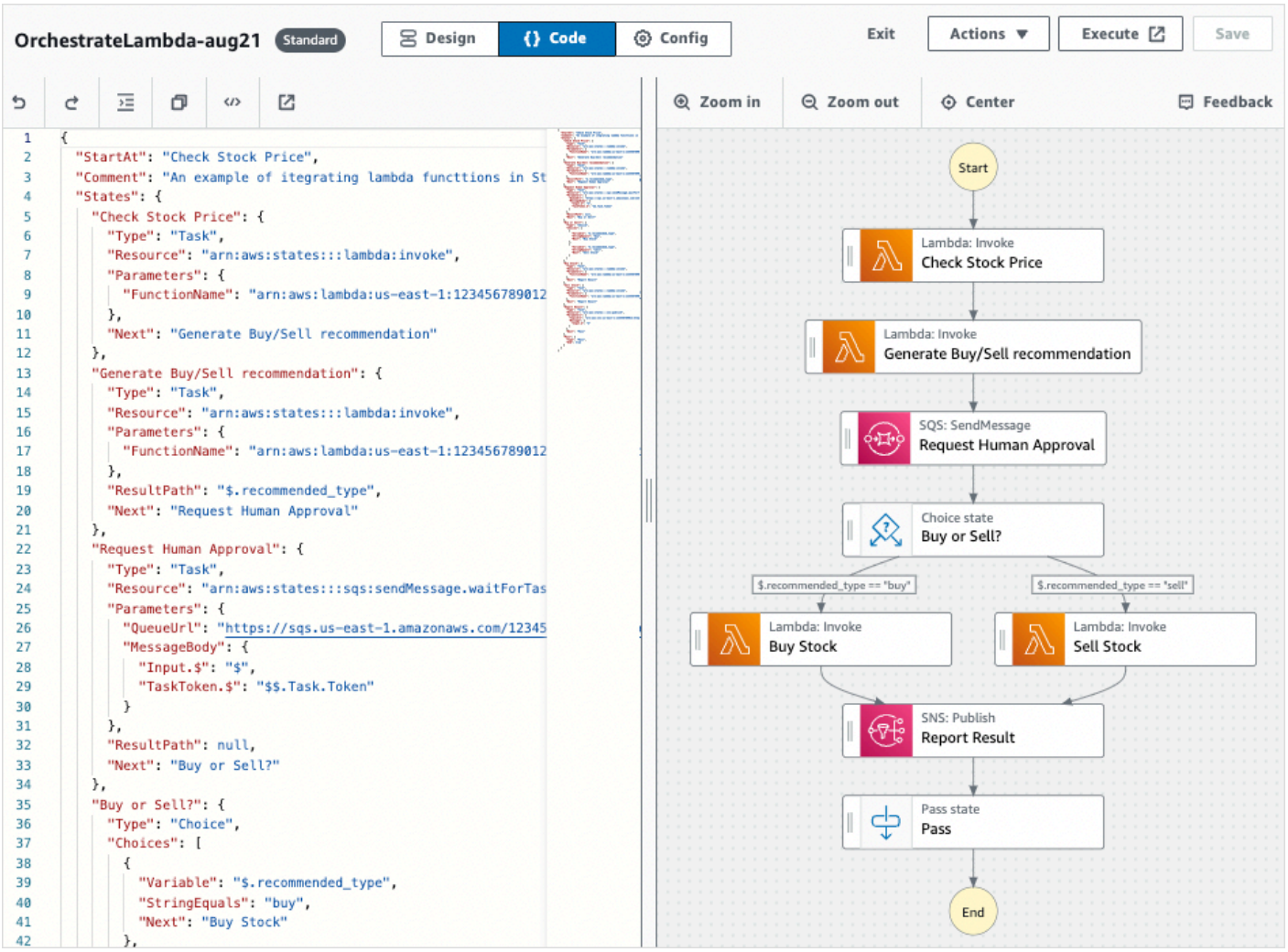
1. Ouvrez la console Step Functions.
2. Sur la page State machines, choisissez le flux de travail que vous souhaitez modifier.
3. Sur la page détaillée de State machine, choisissez Modifier.

4. Le flux de travail s'ouvre en mode Design de Workflow Studio. Modifiez le flux de travail selon vos besoins.



> **Note**
>
> Si vous constatez des erreurs dans votre flux de travail, vous devez les corriger en mode Création. Vous ne pouvez pas passer en mode Code ou Config en cas d'erreur dans votre flux de travail.

5. (Facultatif) Cliquez sur le bouton Code pour afficher ou modifier la définition du flux de travail dans Workflow Studio.



6. Lorsque vous avez terminé, choisissez Enregistrer pour enregistrer votre flux de travail mis à jour.
7. (Facultatif) Pour exécuter votre flux de travail mis à jour, choisissez Execute. La boîte de dialogue Démarrer l'exécution s'ouvre dans un nouvel onglet.

Exportez votre flux de travail

Vous pouvez exporter la définition de votre flux de travail Amazon States Language (ASL) et le graphique de votre flux de travail :

1. Choisissez votre flux de travail dans la console Step Functions.
2. Sur la page détaillée de State machine, choisissez Modifier.
3. (Facultatif) Votre flux de travail s'ouvre en mode Design de Workflow Studio. Modifiez votre flux de travail en mode Design ou passez en mode Code.
4. Cliquez sur le bouton déroulant Actions, puis effectuez l'une des opérations suivantes ou les deux :

- Pour exporter le graphique du flux de travail vers un fichier SVG ou PNG, sous Exporter le graphique, sélectionnez le format souhaité.
- Pour exporter la définition du flux de travail sous forme de fichier JSON ou YAML, sous Définition d'exportation, sélectionnez le format souhaité.

Créez votre prototype de flux de travail

Vous pouvez utiliser Workflow Studio pour créer des prototypes de nouveaux flux de travail contenant des ressources réservées. Vous pouvez également créer vos flux de travail à l'aide de Workflow Studio dans Application Composer. Pour créer un prototype :

1. Connectez-vous à la console Step Functions.
2. Choisissez Create state machine (Créer une machine d'état).
3. Dans la boîte de dialogue Choisir un modèle, sélectionnez Vide.
4. Choisissez Select (Sélectionner). Cela ouvre Workflow Studio dansMode de conception.
5. Le mode Design de Workflow Studio s'ouvre. Concevez votre flux de travail dans Workflow Studio. Pour inclure des ressources réservées :
   a. Choisissez l'état pour lequel vous souhaitez inclure une ressource d'espace réservé, puis dans Configuration :
      - Pour les états Lambda Invoke, choisissez le nom de la fonction, puis entrez le nom de la fonction. Vous pouvez également saisir un nom personnalisé pour votre fonction.
      - Pour les états Amazon SQS Send Message, choisissez URL de file d'attente, puis choisissez Enter queue URL. Entrez une URL de file d'attente fictive.
      - Pour les états Amazon SNS Publish, dans Rubrique, choisissez un ARN de rubrique.
      - Pour tous les autres états répertoriés sous Actions, vous pouvez utiliser la configuration par défaut.

      > **Note**
      >
      > Si vous constatez des erreurs dans votre flux de travail, vous devez les corriger en mode Création. Vous ne pouvez pas passer en mode Code ou Config en cas d'erreur dans votre flux de travail.

   b. (Facultatif) Pour afficher la définition ASL générée automatiquement de votre flux de travail, choisissez Definition.

c. (Facultatif) Pour mettre à jour la définition du flux de travail dans Workflow Studio, cliquez sur le bouton Code.

> **Note**
>
> Si vous constatez des erreurs dans la définition de votre flux de travail, vous devez les corriger en mode Code. Vous ne pouvez pas passer en mode Design ou Config si des erreurs existent dans la définition de votre flux de travail.

6. (Facultatif) Pour modifier le nom de la machine d'état, cliquez sur l'icône d'édition à côté du nom de la machine d'état par défaut de MyStateMachineet spécifiez un nom dans le champ Nom de la machine d'état.

   Vous pouvez également passer au Mode Config pour modifier le nom de la machine à états par défaut.

7. Spécifiez les paramètres de votre flux de travail, tels que le type de machine à états et son rôle d'exécution.
8. Choisissez Créer.

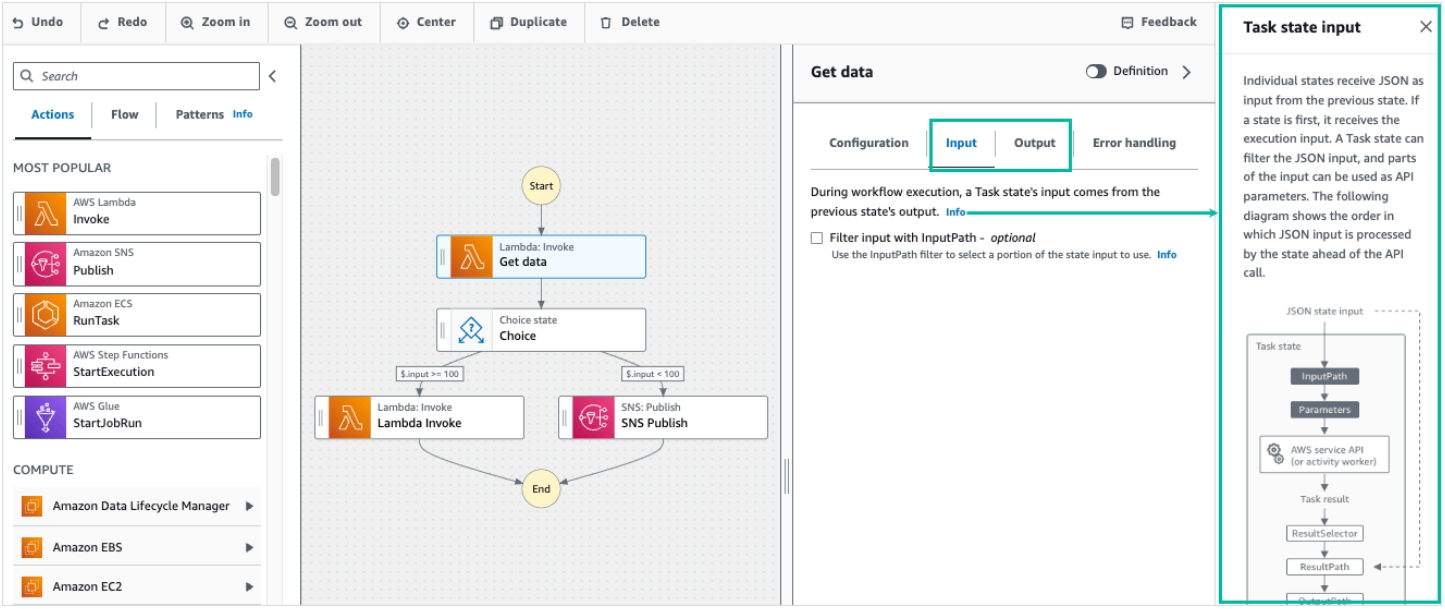
Vous venez de créer un nouveau flux de travail avec des ressources réservées qui peuvent être utilisées pour le prototypage. Vous pouvez exporter la définition de votre flux de travail et le graphique du flux de travail.

- Pour exporter la définition de votre flux de travail sous forme de fichier JSON ou YAML, en mode Design ou Code, cliquez sur le bouton déroulant Actions. Ensuite, sous Définition de l'exportation, sélectionnez le format que vous souhaitez exporter. Vous pouvez utiliser cette définition exportée comme point de départ pour le développement local avec le AWS Toolkit for Visual Studio Code.
- Pour exporter votre graphique de flux de travail vers un fichier SVG ou PNG, en mode Design ou Code, cliquez sur le bouton déroulant Actions. Ensuite, sous Définition de l'exportation, sélectionnez le format souhaité.

Configurez les entrées et les sorties pour vos états

Chaque État prend une décision ou exécute une action en fonction des informations qu'il reçoit. Dans la plupart des cas, il transmet ensuite la sortie à d'autres états. Dans Workflow Studio, vous pouvez configurer la manière dont un état filtre et manipule ses données d'entrée et de sortie

dans les onglets Entrée et Sortie du Inspector panneau. Utilisez les liens Info pour accéder à l'aide contextuelle lors de la configuration des entrées et des sorties.



Pour des informations détaillées sur la façon dont Step Functions traite les entrées et les sorties, consultezTraitement des entrées et des sorties dans Step Functions.

Configuration de l'entrée dans un état

Chaque état reçoit les entrées de l'état précédent au format JSON. Si vous souhaitez filtrer l'entrée, vous pouvez utiliser le `InputPath` filtre sous l'onglet Entrée du Inspector panneau. `InputPath`Il s'agit d'une chaîne commençant par$, qui identifie un nœud JSON spécifique. Ils sont appelés chemins de référence, et ils suivent JsonPath la syntaxe.

Pour filtrer l'entrée, procédez comme suit :

- Choisissez Filtrer l'entrée avec InputPath.

- Entrez une valeur valide JsonPathpour le `InputPath` filtre. Par exemple, **`$.data`**.

Votre `InputPath` filtre sera ajouté à votre flux de travail.

Example Exemple 1 : utilisation d'un InputPath filtre dans Workflow Studio

Supposons que l'entrée de votre état inclut les données JSON suivantes.

```
{
  "comment": "Example for InputPath",
  "dataset1": {
    "val1": 1,
    "val2": 2,
    "val3": 3
  },
  "dataset2": {
    "val1": "a",
    "val2": "b",
    "val3": "c"
  }
}
```

Pour appliquer le `InputPath` filtre, choisissez Filtrer l'entrée avec InputPath, puis entrez un chemin de référence approprié. Si vous entrez**`$.dataset2.val1`**, le JSON suivant est transmis en entrée à l'état.

```
{"a"}
```

Un chemin de référence peut également comporter une sélection de valeurs. Si les données auxquelles vous faites référence le sont `{ "a": [1, 2, 3, 4] }` et que vous appliquez le chemin de référence `$.a[0:2]` comme `InputPath` filtre, voici le résultat.

```
[ 1, 2 ]
```

Parallèle,Map, et les états de Pass flux disposent d'une option de filtrage d'entrée supplémentaire appelée `Parameters` sous leur onglet Entrée. Ce filtre prend effet après le InputPath filtre et peut être utilisé pour créer un objet JSON personnalisé composé d'une ou de plusieurs paires clé-valeur. Les valeurs de chaque paire peuvent être soit des valeurs statiques, soit être sélectionnées à partir de l'entrée, soit être sélectionnées à partir du Objet Contexte chemin.

> **Note**
>
> Pour spécifier qu'un paramètre utilise un chemin de référence pour pointer vers un nœud JSON en entrée, le nom du paramètre doit se terminer par `.$`.

Example Exemple 2 : créer une entrée JSON personnalisée pour l'état parallèle

Supposons que les données JSON suivantes soient l'entrée d'un état parallèle.

```
{
  "comment": "Example for Parameters",
  "product": {
    "details": {
      "color": "blue",
      "size": "small",
      "material": "cotton"
    },
    "availability": "in stock",
    "sku": "2317",
    "cost": "$23"
  }
}
```

Pour sélectionner une partie de cette entrée et transmettre des paires clé-valeur supplémentaires avec une valeur statique, vous pouvez spécifier ce qui suit dans le champ Paramètres, sous l'onglet Entrée de l'état parallèle.

```
{
 "comment": "Selecting what I care about.",
 "MyDetails": {
    "size.$": "$.product.details.size",
    "exists.$": "$.product.availability",
    "StaticValue": "foo"
    }
 }
```

Les données JSON suivantes seront le résultat.

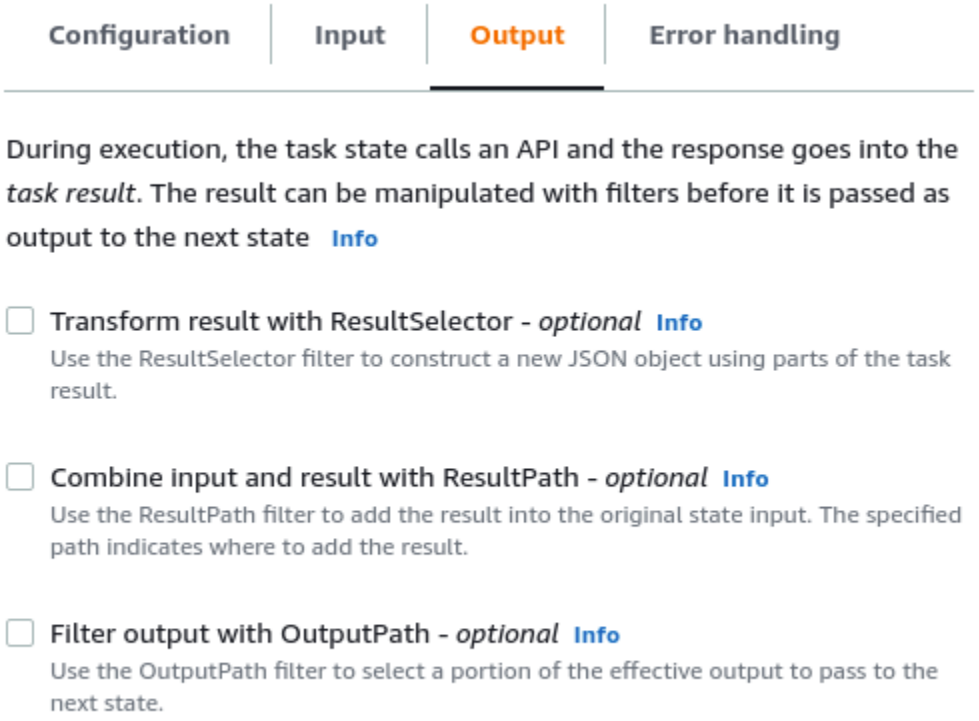
```
{
```

```
  "comment": "Selecting what I care about.",
  "MyDetails": {
    "size": "small",
    "exists": "in stock",
    "StaticValue": "foo"
  }
}
```

Configuration de la sortie d'un état

Chaque état produit une sortie JSON qui peut être filtrée avant de passer à l'état suivant. Plusieurs filtres sont disponibles, et chacun affecte le résultat d'une manière différente. Les filtres de sortie disponibles pour chaque état sont répertoriés sous l'onglet Output du panneau Inspector. Pour État de la tâche les états, tous les filtres de sortie que vous sélectionnez sont traités dans cet ordre :

1. `ResultSelector`: utilisez ce filtre pour manipuler le résultat de l'état. Vous pouvez créer un nouvel objet JSON avec des parties du résultat.
2. `ResultPath`: utilisez ce filtre pour sélectionner une combinaison entre l'entrée d'état et le résultat de la tâche à transmettre à la sortie.
3. `OutputPath`: utilisez ce filtre pour filtrer la sortie JSON afin de choisir les informations du résultat qui seront transmises à l'état suivant.



Utiliser ResultSelector

`ResultSelector`est un filtre de sortie optionnel pour les états suivants :

- État de la tâcheétats, qui sont tous les états répertoriés dans l'onglet Actions duNavigateur d'états.
- Mapétats, dans l'onglet Flow du navigateur States.
- Parallèleétats, dans l'onglet Flow du navigateur States.

`ResultSelector`peut être utilisé pour construire un objet JSON personnalisé composé d'une ou de plusieurs paires clé-valeur. Les valeurs de chaque paire peuvent être des valeurs statiques ou sélectionnées à partir du résultat de l'état avec un chemin.

> **Note**
>
> Pour spécifier qu'un paramètre utilise un chemin pour référencer un nœud JSON dans le résultat, le nom du paramètre doit se terminer par `.$`.

Example Exemple d'utilisation du ResultSelector filtre

Dans cet exemple, vous pouvez `ResultSelector` manipuler la réponse de l'appel d' CreateCluster API Amazon EMR pour un état Amazon EMR. `CreateCluster` Voici le résultat de l'appel d'`CreateCluster`API Amazon EMR.

```
{
  "resourceType": "elasticmapreduce",
  "resource": "createCluster.sync",
  "output": {
    "SdkHttpMetadata": {
      "HttpHeaders": {
        "Content-Length": "1112",
        "Content-Type": "application/x-amz-JSON-1.1",
        "Date": "Mon, 25 Nov 2019 19:41:29 GMT",
        "x-amzn-RequestId": "1234-5678-9012"
      },
      "HttpStatusCode": 200
    },
    "SdkResponseMetadata": {
      "RequestId": "1234-5678-9012"
    },
    "ClusterId": "AKIAIOSFODNN7EXAMPLE"
  }
}
```

Pour sélectionner une partie de ces informations et transmettre une paire clé-valeur supplémentaire avec une valeur statique, spécifiez ce qui suit dans le ResultSelectorchamp, sous l'onglet Sortie de l'État.

```
{
 "result": "found",
 "ClusterId.$": "$.output.ClusterId",
 "ResourceType.$": "$.resourceType"
 }
```

L'utilisation `ResultSelector` produit le résultat suivant.

```
{
 "result": "found",
 "ClusterId": "AKIAIOSFODNN7EXAMPLE",
```

```
 "ResourceType": "elasticmapreduce"
}
```

Utiliser ResultPath

La sortie d'un état peut être une copie de son entrée, le résultat qu'il produit ou une combinaison de son entrée et de son résultat. Utilisez `ResultPath` pour contrôler les combinaisons de ces stratégies transmises à la sortie de l'état. Pour d'autres cas d'utilisation de`ResultPath`, voirResultPath.

`ResultPath`est un filtre de sortie optionnel pour les états suivants :

- État de la tâcheétats, qui sont tous les états répertoriés dans l'onglet Actions du navigateur d'états.
- Mapétats, dans l'onglet Flow du navigateur States.
- Parallèleétats, dans l'onglet Flow du navigateur States.
- Passétats, dans l'onglet Flow du navigateur States.

`ResultPath`peut être utilisé pour ajouter le résultat dans l'entrée d'état d'origine. Le chemin spécifié indique où ajouter le résultat.

Example Exemple d'utilisation du ResultPath filtre

Supposons que ce qui suit soit l'entrée d'un état de tâche.

```
{
  "details": "Default example",
  "who": "AWS Step Functions"
}
```

Le résultat de l'état de la tâche est le suivant.

```
Hello, AWS Step Functions
```

Vous pouvez ajouter ce résultat à l'entrée de l'état en appliquant `ResultPath` et en saisissant un chemin de référence indiquant où ajouter le résultat, tel que `$.taskresult` :

Avec cela`ResultPath`, voici le JSON qui est transmis comme sortie de l'état.

```
{
```

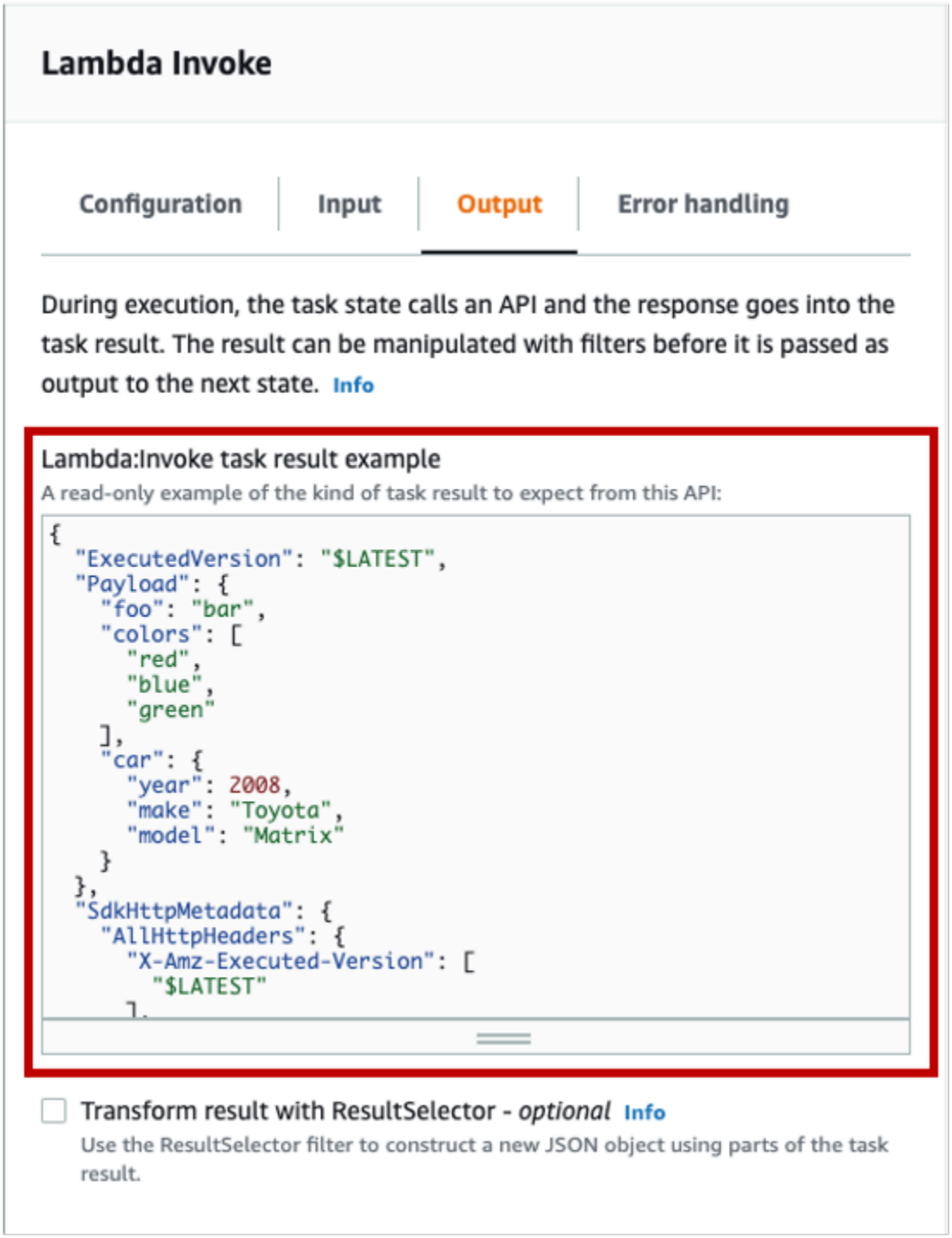
```
  "details": "Default example",
  "who": "AWS Step Functions",
  "taskresult": "Hello, AWS Step Functions!"
}
```

Utiliser OutputPath

Le `OutputPath` filtre vous permet de filtrer les informations indésirables et de ne transmettre que la partie du JSON qui vous intéresse. `OutputPath`Il s'agit d'une chaîne, commençant par$, qui identifie les nœuds dans le texte JSON.

Example Exemple d'utilisation du OutputPath filtre

Un appel d'API Lambda Invoke renvoie des métadonnées en plus de la charge utile, qui est le résultat de la fonction Lambda. Un exemple de réponse à cet appel d'API est affiché sous l'onglet Output de l'état.



Vous pouvez l'`OutputPath`utiliser pour filtrer les métadonnées supplémentaires. Par défaut, la valeur du OutputPathfiltre pour les états Lambda Invoke créés via le Workflow Studio est. `$.Payload` Cette valeur par défaut supprime les métadonnées supplémentaires et renvoie une sortie équivalente à l'exécution directe de la fonction Lambda.

L'exemple de résultat de la tâche Lambda Invoke et la valeur de `$.Payload` pour le filtre de sortie transmettent les données JSON suivantes en sortie.

```
{
 "foo": "bar",
 "colors": [
      "red",
      "blue",
      "green"
 ],
 "car": {
      "year": 2008,
      "make": "Toyota",
      "model": "Matrix"
 }
}
```

> **Note**
>
> Le `OutputPath` filtre étant le dernier filtre de sortie à prendre effet, si vous utilisez des filtres de sortie supplémentaires tels que `ResultSelector` ou`ResultPath`, vous devez modifier la valeur par défaut du `$.Payload` `OutputPath` filtre en conséquence.

Rôles d'exécution dans Workflow Studio

Chaque machine d'Step Functionsétat nécessite un rôle AWS Identity and Access Management (IAM) qui lui accorde l'autorisation d'effectuer des actions sur Services AWS des ressources ou d'appeler des API tierces. Ce rôle est appelé rôle d'exécution. Ce rôle doit contenir des IAM politiques pour chaque action, par exemple des politiques qui permettent à la machine d'état d'appeler une AWS Lambda fonction, d'exécuter une AWS Batch tâche ou d'appeler l'API Stripe. Step Functionsvous oblige à fournir un rôle d'exécution dans les cas suivants :

- Vous créez une machine à états dans la console, dans les AWS SDK ou à AWS CLI l'aide de l'CreateStateMachineAPI.
- Vous testez un état dans la console, dans AWS les SDK ou à AWS CLI l'aide de l'TestStateAPI.

Workflow Studio possède des fonctionnalités qui facilitent la gestion des rôles d'exécution de vos flux de travail.

Rubriques

À propos des rôles générés automatiquement

Lorsque vous créez une machine à états dans la Step Functions console, Workflow Studio peut automatiquement créer pour vous un rôle d'exécution contenant les IAM politiques nécessaires. Workflow Studio analyse la définition de votre machine à états et génère des politiques avec le moins de privilèges nécessaires pour exécuter votre flux de travail.

Workflow Studio peut générer des IAM politiques pour les éléments suivants :

- Tâches HTTP qui appellent des API tierces.
- États de tâches qui en appellent d'autres Services AWS à l'aide d'intégrations optimisées, telles que LambdaInvoke ou AWS Glue StartJobRun. DynamoDB GetItem
- États de tâches qui exécutent des flux de travail imbriqués.
- États cartographiques distribués, y compris les politiques permettant de démarrer l'exécution des flux de travail pour enfants, de répertorier les Amazon S3 compartiments et de lire ou d'écrire des objets S3.
- X-Raytraçage. Chaque rôle généré automatiquement dans Workflow Studio contient une politique qui accorde des autorisations à la machine d'état pour envoyer X-Ray des traces.
- Journalisation à l'aideCloudWatchJournauxlorsque la journalisation est activée sur la machine d'état.

Workflow Studio ne peut pas générer de IAM politiques pour les états de tâches qui en appellent d'autres à Services AWS l'aide des intégrations du AWS SDK.

Génération automatique de rôles

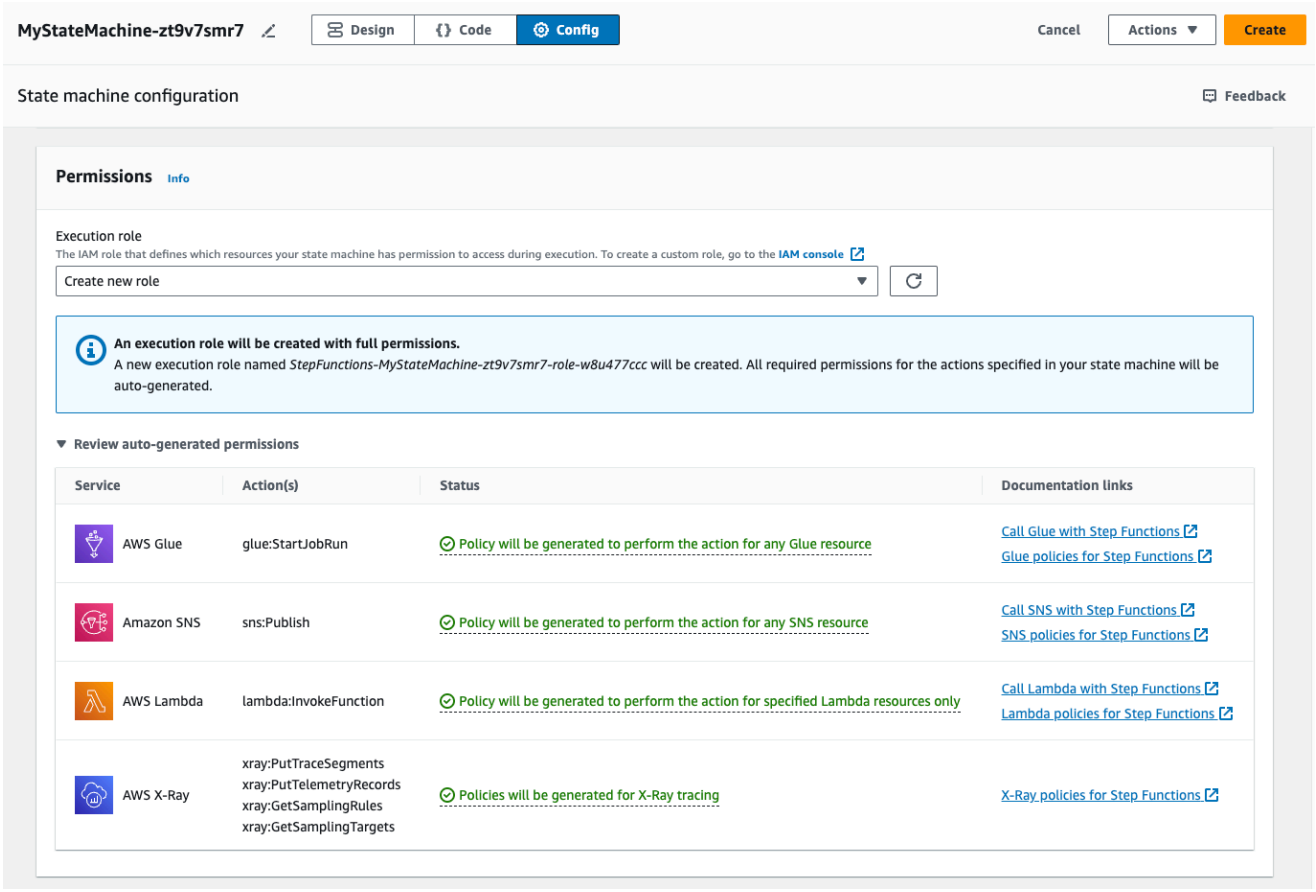
1. Ouvrez la console Step Functions et choisissez Create state machine.

   Vous pouvez également mettre à jour une machine à états existante. Reportez-vous à l'étape 4 si vous mettez à jour une machine à états.

2. Dans la boîte de dialogue Choisir un modèle, sélectionnez Vide.
3. Choisissez Select (Sélectionner). Cela ouvre Workflow Studio dansMode de conception.
4. Choisissez l'onglet Config.
5. Faites défiler la page jusqu'à la section Autorisations, puis procédez comme suit :

   a. Pour le rôle d'exécution, assurez-vous de conserver la sélection par défaut de Créer un nouveau rôle.

   Workflow Studio génère automatiquement toutes les IAM politiques requises pour chaque état valide dans la définition de votre machine à états. Il affiche une bannière avec le message « Un rôle d'exécution sera créé avec toutes les autorisations ».



> **Tip**
>
> Pour vérifier les autorisations générées automatiquement par Workflow Studio pour votre machine d'état, choisissez Vérifier les autorisations générées automatiquement.

> **Note**
>
> Si vous supprimez le rôle IAM créé par Step Functions, Step Functions ne pourra pas le recréer ultérieurement. De même, si vous modifiez le rôle (par exemple, en supprimant Step Functions des principes de la politique IAM), Step Functions ne pourra pas restaurer ses paramètres d'origine ultérieurement.

Si Workflow Studio ne parvient pas à générer toutes les IAM politiques requises, il affiche une bannière avec le message Les autorisations pour certaines actions ne peuvent pas être générées automatiquement. Un IAM rôle sera créé avec des autorisations partielles uniquement. Pour plus d'informations sur la façon d'ajouter les autorisations manquantes, consultezRésoudre les problèmes de génération de rôles.

b. Choisissez Create si vous créez une machine à états. Sinon, choisissez Save (Enregistrer).

c. Choisissez Confirmer dans la boîte de dialogue qui apparaît.

Workflow Studio enregistre votre machine d'état et crée le nouveau rôle d'exécution.

Résoudre les problèmes de génération de rôles

Workflow Studio ne peut pas générer automatiquement un rôle d'exécution avec toutes les autorisations requises dans les cas suivants :

- Il y a des erreurs dans votre machine à états. Assurez-vous de résoudre toutes les erreurs de validation dans Workflow Studio. Assurez-vous également de corriger toutes les erreurs côté serveur que vous rencontrez au cours de l'enregistrement.
- Votre machine à états contient des tâches utilisant des intégrations de AWS SDK. Workflow Studio ne peut pas générer automatiquement de IAM politiques dans ce cas. Workflow Studio affiche une bannière avec le message « Les autorisations pour certaines actions ne peuvent pas être générées

automatiquement ». Un IAM rôle sera créé avec des autorisations partielles uniquement. Dans le tableau des autorisations générées automatiquement par Review, choisissez le contenu dans Status pour plus d'informations sur les politiques manquantes à votre rôle d'exécution. Workflow Studio peut toujours générer un rôle d'exécution, mais ce rôle ne contiendra pas de IAM politiques pour toutes les actions. Consultez les liens sous Liens de documentation pour rédiger vos propres politiques et les ajouter au rôle une fois celui-ci généré. Ces liens sont disponibles même après avoir enregistré la machine d'état.

Rôle pour tester les tâches HTTP dans Workflow Studio

Vous avez besoin d'un rôle d'exécution pour tester l'état d'une tâche HTTP. Si vous ne disposez pas d'un rôle doté d'autorisations suffisantes, utilisez l'une des options suivantes pour créer un rôle :

- Générer automatiquement un rôle avec Workflow Studio (recommandé) : il s'agit de l'option sécurisée. Fermez la boîte de dialogue État du test et suivez les instructions indiquées dansGénération automatique de rôles. Cela vous obligera à créer ou à mettre à jour votre machine d'état d'abord, puis à retourner dans Workflow Studio pour tester votre état.
- Utiliser un rôle doté d'un accès administrateur : si vous êtes autorisé à créer un rôle avec un accès complet à tous les services et ressources AWS, vous pouvez utiliser ce rôle pour tester n'importe quel type d'état dans votre flux de travail. Pour ce faire, vous pouvez créer un rôle de Step Functions service et y ajouter la AdministratorAccess politique dans la IAM console https://console.aws.amazon.com/iam/.

Rôle pour tester une intégration de service optimisée dans Workflow Studio

Vous avez besoin d'un rôle d'exécution pour les états de tâches qui appellent des intégrations de services optimisées. Si vous ne disposez pas d'un rôle doté d'autorisations suffisantes, utilisez l'une des options suivantes pour créer un rôle :

- Utilisez les liens de documentation dans Workflow Studio pour rédiger vos propres IAM politiques (recommandé). Il s'agit de l'option sécurisée. Fermez la boîte de dialogue État du test et suivez les instructions indiquées dansGénération automatique de rôles. Cela vous obligera à créer ou à mettre à jour votre machine d'état d'abord, puis à retourner dans Workflow Studio pour tester votre état.
- Utiliser un rôle doté d'un accès administrateur : si vous êtes autorisé à créer un rôle avec un accès complet à tous les services et ressources AWS, vous pouvez utiliser ce rôle pour tester

n'importe quel type d'état dans votre flux de travail. Pour ce faire, vous pouvez créer un rôle de Step Functions service et y ajouter la AdministratorAccess politique dans la IAM console https://console.aws.amazon.com/iam/.

Rôle pour tester l'intégration d'un service AWS SDK dans Workflow Studio

Vous avez besoin d'un rôle d'exécution pour les états de tâches qui appellent des intégrations de AWS SDK. Si vous ne disposez pas d'un rôle doté d'autorisations suffisantes, utilisez l'une des options suivantes pour créer un rôle :

- Utilisez les liens de documentation dans Workflow Studio pour rédiger vos propres IAM politiques (recommandé). Il s'agit de l'option sécurisée. Fermez la boîte de dialogue État du test et suivez les instructions indiquées dansGénération automatique de rôles. Cela vous obligera à créer ou à mettre à jour votre machine d'état d'abord, puis à retourner dans Workflow Studio pour tester votre état. Procédez comme suit :
  1. Fermez la boîte de dialogue État du test
  2. Choisissez l'onglet Config pour afficher le mode Config.
  3. Faites défiler la page vers le bas jusqu'à la section Autorisations.
  4. Workflow Studio affiche une bannière avec le message « Les autorisations pour certaines actions ne peuvent pas être générées automatiquement ». Un IAM rôle sera créé avec des autorisations partielles uniquement. Choisissez Vérifier les autorisations générées automatiquement.
  5. Le tableau des autorisations généré automatiquement par Review affiche une ligne qui indique l'action correspondant à l'état de la tâche que vous souhaitez tester. Consultez les liens sous Liens de documentation pour écrire vos propres IAM politiques dans un rôle personnalisé.
- Utiliser un rôle doté d'un accès administrateur : si vous êtes autorisé à créer un rôle avec un accès complet à tous les services et ressources AWS, vous pouvez utiliser ce rôle pour tester n'importe quel type d'état dans votre flux de travail. Pour ce faire, vous pouvez créer un rôle de Step Functions service et y ajouter la AdministratorAccess politique dans la IAM console https://console.aws.amazon.com/iam/.

Rôle pour tester les états des flux dans Workflow Studio

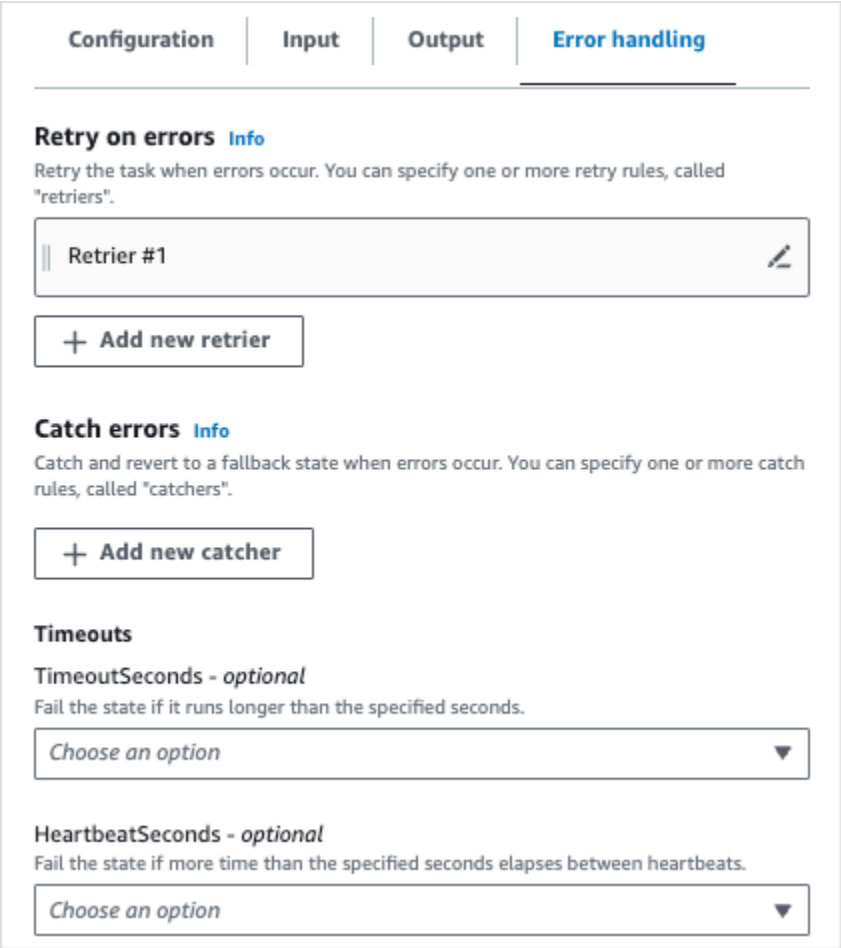
Vous avez besoin d'un rôle d'exécution pour tester les états de flux dans Workflow Studio. Les états de flux sont les états qui dirigent le flux d'exécution ChoiceParallèle, tels

queMap,Pass,Attente,Succeed, ouFail. L'TestStateAPI ne fonctionne pas avec les états Map ou Parallel. Utilisez l'une des options suivantes pour créer un rôle afin de tester l'état d'un flux :

- Utilisez n'importe quel rôle dans votre Compte AWS (recommandé) — Les états de flux ne nécessitent aucune IAM politique spécifique, car ils n'appellent ni AWS actions ni ressources. Par conséquent, vous pouvez utiliser n'importe quel IAM rôle dans votre Compte AWS.
  1. Dans la boîte de dialogue État du test, sélectionnez un rôle dans la liste déroulante Rôle d'exécution.
  2. Si aucun rôle n'apparaît dans la liste déroulante, procédez comme suit :
     a. Dans la IAM console https://console.aws.amazon.com/iam/, choisissez Roles.
     b. Choisissez un rôle dans la liste et copiez son ARN depuis la page de détails du rôle. Vous devrez fournir cet ARN dans la boîte de dialogue Test state.
     c. Dans la boîte de dialogue État du test, sélectionnez Entrer un ARN de rôle dans la liste déroulante Rôle d'exécution.
     d. Collez l'ARN dans Role ARN.
- Utiliser un rôle doté d'un accès administrateur : si vous êtes autorisé à créer un rôle avec un accès complet à tous les services et ressources AWS, vous pouvez utiliser ce rôle pour tester n'importe quel type d'état dans votre flux de travail. Pour ce faire, vous pouvez créer un rôle de Step Functions service et y ajouter la AdministratorAccess politique dans la IAM console https://console.aws.amazon.com/iam/.

Gestion des erreurs

Par défaut, lorsqu'un état signale une erreur, Step Functions fait échouer complètement l'exécution du flux de travail. Pour les actions et certains états de flux, vous pouvez configurer la façon dont Step Functions gère les erreurs. Même si vous avez configuré la gestion des erreurs, certaines erreurs peuvent toujours entraîner l'échec de l'exécution d'un flux de travail. Pour plus d’informations, consultez Gestion des erreurs dans Step Functions. Dans Workflow Studio, configurez la gestion des erreurs dans l'onglet Gestion des erreurs du Inspector panneau.



Réessayer en cas d'erreur

Vous pouvez ajouter une ou plusieurs règles aux états d'action et à l'état de Parallèle flux pour réessayer la tâche en cas d'erreur. Ces règles sont appelées retriers. Pour ajouter un retrier, choisissez l'icône d'édition dans la case Retrier #1, puis configurez ses options :

- (Facultatif) Dans le champ Commentaire, ajoutez votre commentaire. Cela n'affectera pas le flux de travail, mais peut être utilisé pour annoter votre flux de travail.
- Placez le curseur dans le champ Erreurs et choisissez une erreur qui déclenchera le récupérateur, ou entrez un nom d'erreur personnalisé. Vous pouvez sélectionner ou ajouter plusieurs erreurs.
- (Facultatif) Définissez un intervalle. Il s'agit du délai en secondes avant que Step Functions n'effectue sa première tentative. D'autres tentatives suivront à des intervalles que vous pouvez configurer avec le nombre maximum de tentatives et le taux de rétroactivité.
- (Facultatif) Définissez le nombre maximum de tentatives. Il s'agit du nombre maximum de tentatives avant que Step Functions ne fasse échouer l'exécution.

- (Facultatif) Réglez le taux de rétrogradation. Il s'agit d'un multiplicateur qui détermine l'augmentation de l'intervalle entre les tentatives à chaque tentative.

> **Note**
>
> Les options de gestion des erreurs ne sont pas toutes disponibles pour tous les États. Lambda Invoke possède un récupérateur configuré par défaut.

Détectez les erreurs

Vous pouvez ajouter une ou plusieurs règles aux états action Parallèle et Map flow pour détecter une erreur. Ces règles sont appelées catchers. Pour ajouter un capteur, choisissez Ajouter un nouveau capteur, puis configurez ses options :

- (Facultatif) Dans le champ Commentaire, ajoutez votre commentaire. Cela n'affectera pas le flux de travail, mais peut être utilisé pour annoter votre flux de travail.
- Placez le curseur dans le champ Erreurs et choisissez une erreur qui déclenchera le capteur, ou entrez un nom d'erreur personnalisé. Vous pouvez sélectionner ou ajouter plusieurs erreurs.
- Dans le champ État de secours, choisissez un état de secours. Il s'agit de l'état dans lequel le flux de travail passera ensuite après la détection d'une erreur.
- (Facultatif) Dans le ResultPathchamp, ajoutez un `ResultPath` filtre pour ajouter l'erreur à l'entrée d'état d'origine. `ResultPath`Il doit s'agir d'un code valide JsonPath. Cela sera envoyé à l'état de secours.

Délais

Vous pouvez configurer un délai d'expiration pour les états d'action afin de définir le nombre maximal de secondes pendant lequel votre état peut s'exécuter avant qu'il n'échoue. Utilisez des délais d'attente pour empêcher les exécutions bloquées. Pour configurer un délai d'attente, entrez le nombre de secondes que votre état doit attendre avant que l'exécution échoue. Pour plus d'informations sur les délais d'expiration, voir `TimeoutSeconds` in État de la tâche state.

HeartbeatSeconds

Vous pouvez configurer un rythme cardiaque ou une notification périodique envoyée par votre tâche. Si vous définissez un intervalle entre les pulsations et que votre État n'envoie pas de notifications de

battements cardiaques dans les intervalles définis, la tâche est marquée comme ayant échoué. Pour configurer un rythme cardiaque, définissez un nombre entier de secondes positif différent de zéro. Pour plus d'informations, voir `HeartBeatSeconds` en État de la tâche état.

Tutoriel : Apprenez à utiliser le AWS Step Functions Workflow Studio

Dans ce didacticiel, vous découvrirez les bases de l'utilisation de Workflow Studio pour AWS Step Functions. Dans Mode de conception Workflow Studio, vous allez créer une machine à états contenant plusieurs états`Pass`, notamment`Choice`,`Fail`,`Wait`, et`Parallel`. Vous allez utiliser la fonction glisser-déposer pour rechercher, sélectionner et configurer ces états. Vous verrez ensuite la définition générée automatiquement Amazon States Language (ASL) de votre flux de travail. Vous utiliserez également Workflow Studio pour modifier la définition du flux de travail. Mode code Ensuite, vous quitterez Workflow Studio, exécuterez la machine d'état et passerez en revue les détails de l'exécution.

Dans ce didacticiel, vous apprendrez également à mettre à jour la machine à états et à visualiser les modifications apportées au résultat de l'exécution. Enfin, vous allez effectuer une étape de nettoyage et supprimer votre machine à états.

Après avoir terminé ce didacticiel, vous saurez comment utiliser Workflow Studio pour créer et configurer un flux de travail en utilisant à la fois les modes Design et Code. Vous saurez également comment mettre à jour, exécuter et supprimer votre machine d'état.

> **Note**
>
> Avant de commencer, assurez-vous de remplir les conditions requises pour ce didacticiel.

Rubriques

Étape 1 : Accédez à Workflow Studio

1. Ouvrez la console Step Functions et choisissez Create state machine.
2. Dans la boîte de dialogue Choisir un modèle, sélectionnez Vide.
3. Choisissez Select (Sélectionner). Cela ouvre Workflow Studio dansMode de conception.

Étape 2 : Création d'une machine à états

Dans Workflow Studio, une machine à états est une représentation graphique de votre flux de travail. Avec Workflow Studio, vous pouvez définir, configurer et examiner les différentes étapes de votre flux de travail. Dans les étapes suivantes, vous utiliserez Mode de conception Workflow Studio pour créer votre machine à états.

Pour créer une machine d'état

1. Assurez-vous que vous êtes en mode Design de Workflow Studio.
2. Navigateur d'étatsSur la gauche, choisissez l'onglet Flow. Faites ensuite glisser un état Pass vers l'état vide intitulé Drag first state here.
3. Faites glisser un état Choice depuis l'onglet Flow et déposez-le en dessous de l'état Pass.
4. Pour le nom de l'État, remplacez le nom par défaut de Choice. Pour ce tutoriel, utilisez le nom **`IsHelloWorldExample`**.
5. Faites glisser un autre état Pass et déposez-le sur une branche de l'IsHelloWorldExampleétat. Faites ensuite glisser un état Fail et déposez-le sous l'autre branche de l'IsHelloWorldExampleétat.
6. Choisissez l'état Pass (1) et renommez-le en**`Yes`**. Renommez l'état Fail en**`No`**.
7. Spécifiez la logique de branchement de IsHelloWorldExamplel'état à l'aide de la variable booléenne. `IsHelloWorldExample`

   Si tel `IsHelloWorldExample` est le cas`False`, le flux de travail passera à l'état Non. Dans le cas contraire, le flux de travail poursuivra son flux d'exécution à l'état Oui.

   Pour définir la logique de branchement, procédez comme suit :

a. Choisissez l'IsHelloWorldExampleétat sur leCanvas, puis sous Règles de choix, cliquez sur l'icône d'édition dans la zone Règle #1 pour définir la règle de premier choix.

b. Sélectionnez Ajouter des conditions.

c. Dans la boîte de dialogue Conditions pour la règle #1, entrez dans **`$.IsHelloWorldExample`** Variable.

d. Choisir est égal à sous Opérateur.

e. Choisissez Constante booléenne sous Valeur, puis sélectionnez vrai dans la liste déroulante.

f. Choisissez Enregistrer les conditions.

g. Assurez-vous que l'état suivant est : est sélectionné dans la liste déroulante « Oui ».

h. Choisissez Ajouter une nouvelle règle de choix, puis sélectionnez Ajouter des conditions.

i. Dans la zone Règle #2, définissez la règle du second choix lorsque la valeur de la `IsHelloWorldExample` variable est fausse en répétant les sous-étapes 7.c à 7.f. Pour l'étape 7.e, choisissez false au lieu de true.

j. Dans la zone Règle #2, choisissez Non dans la liste déroulante Then next state is :.

k. Dans la zone Règle par défaut, cliquez sur l'icône de modification pour définir la règle de choix par défaut, puis choisissez Oui dans la liste déroulante.

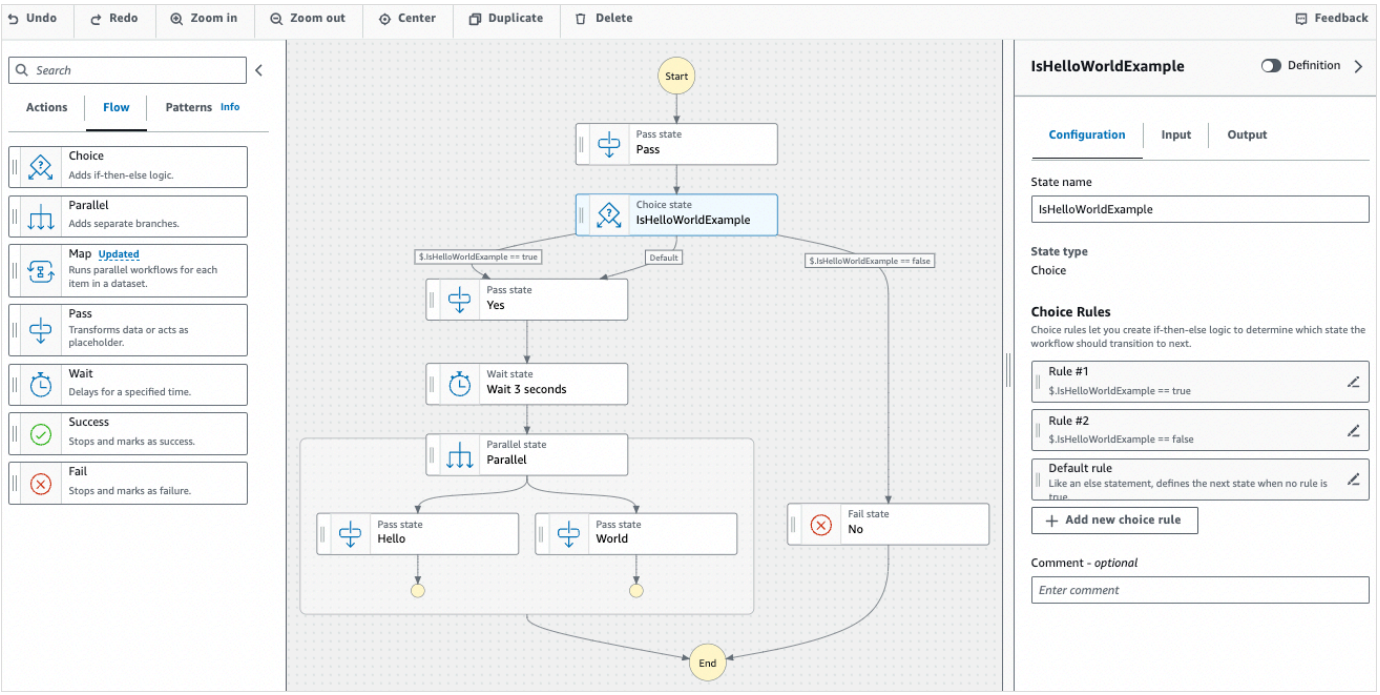
8. Ajoutez un état d'attente après l'état Oui et nommez-le**`Wait 3 sec`**. Configurez ensuite le temps d'attente de trois secondes en procédant comme suit :

   a. Sous Options, conservez la sélection par défaut de Attendre pendant un intervalle fixe.

   b. Sous Secondes, assurez-vous que l'option Enter seconds est sélectionnée, puis entrez **`3`** dans le champ.

9. Après l'état Attendre 3 secondes, ajoutez un état parallèle. Ajoutez deux états Pass dans ses deux branches. Nommez le premier État du Pass**`Hello`**. Nommez le deuxième état du Pass**`World`**.

   Le flux de travail terminé ressemblera à ceci :



Étape 3 : Vérifiez la définition de la langue Amazon States générée automatiquement

Lorsque vous glissez et déposez des états de l'onglet Flow sur le canevas, Workflow Studio compose automatiquement la définition Amazon States Language (ASL) de votre flux de travail en temps réel. Dans le Inspector panneau, cliquez sur le bouton Définition pour afficher cette définition ou passez au bouton Mode code pour modifier cette définition selon vos besoins. Pour plus d'informations sur la modification de la définition du flux de travail, reportez-vous à l'étape 4 de ce didacticiel.

- (Facultatif) Choisissez Definition dans le panneau Inspector et visualisez le flux de travail de la machine à états.

  L'exemple de code suivant montre la définition du langage Amazon States générée automatiquement pour la machine à `IsHelloWorldExample` états. L'`Choice`état que vous avez ajouté dans Workflow Studio est utilisé pour déterminer le flux d'exécution en fonction de la logique de branchement que vous avez définie à l'étape 2.

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

```
   "Pass": {
     "Type": "Pass",
     "Next": "IsHelloWorldExample",
     "Comment": "A Pass state passes its input to its output, without performing
work. Pass states are useful when constructing and debugging state machines."
   },
   "IsHelloWorldExample": {
     "Type": "Choice",
     "Comment": "A Choice state adds branching logic to a state machine. Choice
rules can implement 16 different comparison operators, and can be combined using
And, Or, and Not\"",
     "Choices": [
       {
         "Variable": "$.IsHelloWorldExample",
         "BooleanEquals": false,
         "Next": "No"
       },
       {
         "Variable": "$.IsHelloWorldExample",
         "BooleanEquals": true,
         "Next": "Yes"
       }
     ],
     "Default": "Yes"
   },
   "No": {
     "Type": "Fail",
     "Cause": "Not Hello World"
   },
   "Yes": {
     "Type": "Pass",
     "Next": "Wait 3 sec"
   },
   "Wait 3 sec": {
     "Type": "Wait",
     "Seconds": 3,
     "Next": "Parallel"
   },
   "Parallel": {
     "Type": "Parallel",
     "End": true,
     "Branches": [
       {
         "StartAt": "Hello",
```

```
          "States": {
            "Hello": {
              "Type": "Pass",
              "End": true
            }
          }
        },
        {
          "StartAt": "World",
          "States": {
            "World": {
              "Type": "Pass",
              "End": true
            }
          }
        }
      ]
    }
  }
}
```

Étape 4 : Modifier la définition du flux de travail en mode Code

Le mode Code de Workflow Studio fournit un éditeur de code intégré pour afficher et modifier la définition ASL de vos flux de travail.

1. Choisissez Code pour passer en mode Code.
2. Après la définition de l'état parallèle, placez le curseur et appuyez sur**Enter**.
3. Appuyez **Ctrl+space** pour voir la liste des états que vous pouvez ajouter après l'état parallèle.
4. Choisissez Pass State dans la liste des options. L'éditeur de code ajoute un code standard pour le Pass State.
5. L'ajout de cet état entraîne des erreurs dans la définition de votre flux de travail. Dans la définition de l'état parallèle, remplacez `"End": true` par**`"Next": "PassState"`**.
6. Dans la définition de Pass State que vous avez ajoutée, apportez les modifications suivantes :
   a. Supprimez le nœud Result.
   b. Supprimer `"ResultPath": "$.result"`, et`"Next": "NextState"`.
   c. Après`"Type": "Pass",`, entrez**`"End": true`**.

d. Ajoutez un **,** après la définition de Pass State.

La définition de votre flux de travail doit désormais ressembler à la définition suivante.

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

Étape 5 : Enregistrez la machine à états

1. Choisissez le mode Config ou cliquez sur l'icône d'édition à côté du nom de la machine à états par défaut de MyStateMachine. Dans Configuration de la machine State, spécifiez un nom. Par exemple, saisissez **`HelloWorld`**.
2. (Facultatif) Spécifiez d'autres paramètres de flux de travail, tels que le type de machine à états et son rôle d'exécution. Pour ce didacticiel, conservez toutes les sélections par défaut dans la configuration State Machine.
3. Choisissez Créer.
4. Dans la boîte de dialogue Confirmer la création du rôle, choisissez Confirmer pour continuer.

   Vous pouvez également choisir Afficher la configuration des rôles pour revenir au mode Config.

Pour plus d'informations sur le mode Config, voir Mode Config de Workflow Studio.

Étape 6 : Exécutez la machine d'état

Les exécutions par State Machine sont des instances dans lesquelles vous exécutez votre flux de travail pour effectuer des tâches.

1. Sur la page State machines, choisissez l'HelloWorldétat machine.
2. Sur la HelloWorldpage, choisissez Démarrer l'exécution.
3. (Facultatif) Pour identifier votre exécution, vous pouvez lui donner un nom dans le champ Nom. Par défaut, Step Functions génère automatiquement un nom d'exécution unique.

   > **Note**
   >
   > Step Functions vous permet de créer des noms pour les machines d'état, les exécutions et les activités, ainsi que des étiquettes contenant des caractères non ASCII. Ces noms non ASCII ne fonctionnent pas avec Amazon. CloudWatch Pour être sûr de pouvoir suivre CloudWatch les métriques, choisissez un nom qui utilise uniquement des caractères ASCII.

4. Dans le champ Entrée, entrez les valeurs d'entrée pour votre exécution au format JSON. En fonction de votre saisie, la `IsHelloWorldExample` variable détermine le flux de machine à états qui sera exécuté. Pour le moment, utilisez la valeur d'entrée suivante :

   ```
   {
      "IsHelloWorldExample": true
   }
   ```

   > **Note**
   >
   > Bien que la spécification d'une entrée d'exécution soit facultative, dans ce didacticiel, il est obligatoire de spécifier une entrée d'exécution similaire à l'exemple d'entrée ci-dessus. Cette valeur d'entrée est référencée dans l'`Choice`état lorsque vous exécutez la machine d'état.

5. Choisissez Start execution (Démarrer l'exécution).
6. La console Step Functions vous dirige vers une page intitulée avec votre ID d'exécution. Cette page est connue sous le nom de page Détails de l'exécution. Sur cette page, vous pouvez

consulter les résultats de l'exécution au fur et à mesure que l'exécution progresse ou une fois celle-ci terminée.

Pour consulter les résultats de l'exécution, choisissez des états individuels dans la vue graphique, puis choisissez les onglets individuels du Détails de l'étape volet pour afficher les détails de chaque état, y compris les entrées, les sorties et la définition respectivement. Pour plus de détails sur les informations d'exécution que vous pouvez consulter sur la page Détails de l'exécution, voirPage de détails d'exécution — Vue d'ensemble de l'interface.

Pour ce didacticiel, si vous avez saisi une valeur d'entrée de`"IsHelloWorldExample": true`, vous devriez voir le résultat suivant :

```
{
   "IsHelloWorldExample": true
},
{
   "IsHelloWorldExample": true
}
```

Étape 7 : Mettez à jour votre machine d'état

Lorsque vous mettez à jour une machine à états, vos mises à jour sont finalement cohérentes. Après un court laps de temps, toutes les nouvelles exécutions refléteront la définition mise à jour de votre machine à états. Toutes les exécutions en cours s'exécuteront jusqu'à leur fin selon la définition précédente.

Au cours de cette étape, vous allez mettre à jour votre machine d'état en Mode de conception mode Workflow Studio. Vous allez ajouter un `Result` champ intitulé World dans l'état du Pass.

1. Sur la page intitulée avec votre ID d'exécution, choisissez Modifier la machine à états.
2. Assurez-vous que vous êtes en mode Design.
3. Choisissez l'état Pass nommé World sur le canevas, puis choisissez Output.
4. Dans le champ Résultat, entrez**`"World has been updated!"`**.
5. Choisissez Enregistrer.
6. (Facultatif) Dans la zone Définition, consultez la définition Amazon States Language mise à jour de votre flux de travail.

```
{
      "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. Sélectionnez Execute (Exécuter).
8. Dans la boîte de dialogue Démarrer l'exécution qui s'ouvre dans un nouvel onglet, saisissez l'entrée d'exécution suivante.

```
{
   "IsHelloWorldExample": true
}
```

9. Choisissez Démarrer une exécution.
10. (Facultatif) Dans la vue graphique, choisissez l'étape Monde, puis choisissez Sortie. Le résultat est que World a été mis à jour !

Étape 8 : Nettoyer

Pour supprimer votre machine à états

1. Dans le menu de navigation, choisissez State machines.
2. Sur la page State machines, sélectionnez HelloWorld, puis choisissez Supprimer.
3. Dans la boîte de dialogue Supprimer la machine à états, tapez **`delete`** pour confirmer la suppression.
4. Sélectionnez Delete (Supprimer).

   Si la suppression est réussie, une barre d'état verte apparaît en haut de votre écran. La barre d'état verte indique que votre machine à états est marquée pour être supprimée. Votre machine d'état sera supprimée lorsque toutes ses exécutions en cours cesseront de fonctionner.

Pour supprimer votre rôle d'exécution

1. Ouvrez la page Rôles pour IAM.
2. Choisissez le rôle IAM que Step Functions a créé pour vous. Par exemple, StepFunctions-HelloWorld -Role-example.
3. Choisissez Supprimer le rôle.
4. Choisissez Oui, supprimer.

Tutoriels pour Step Functions

Les didacticiels de cette section vous aident à comprendre les différents aspects de l'utilisation d' AWS Step Functions.

Pour suivre ces didacticiels, vous avez besoin d'un AWS compte. Si vous n'avez pas de AWS compte, rendez-vous sur https://aws.amazon.com/ et choisissez Créer un AWS compte.

Rubriques

Création d'une machine d'état Step Functions utilisant Lambda

Dans ce didacticiel, vous allez créer un flux de travail en une seule étape en utilisant AWS Step Functions pour appeler une AWS Lambda fonction.

> **Note**
>
> Step Functions est basé sur des machines à états et des tâches. Dans Step Functions, les machines à états sont appelées flux de travail, qui sont une série d'étapes pilotées par des événements. Chaque étape d'un flux de travail est appelée un état. Par exemple, un état de tâche représente une unité de travail exécutée par un autre AWS service, comme l'appel d'un autre service Service AWS ou d'une API.
> Pour plus d'informations, consultez :
>
> - Qu'est-ce que c'est AWS Step Functions ?
> - Appelez d'autres AWS services

Lambda convient parfaitement aux `Task` états, car les fonctions Lambda sont sans serveur et faciles à écrire. Vous pouvez écrire du code dans l'éditeur AWS Management Console ou dans votre éditeur préféré. AWS gère les détails relatifs à la mise à disposition d'un environnement informatique pour votre fonction et à son exécution.

Dans cette rubrique :

Étape 1 : Créer une fonction Lambda

Votre fonction Lambda reçoit les données des événements et renvoie un message d'accueil.

> **Important**
>
> Assurez-vous que votre fonction Lambda se trouve sous le même AWS compte et dans la même AWS région que votre machine à états.

1. Ouvrez la console Lambda et choisissez Create function.
2. Sur la page Create function, sélectionnez Author from scratch.

3. Sous Nom de la fonction, saisissez `HelloFunction`.
4. Conservez les sélections par défaut pour toutes les autres options, puis choisissez Créer une fonction.
5. Une fois votre fonction Lambda créée, copiez le nom de ressource Amazon (ARN) de la fonction affiché dans le coin supérieur droit de la page. Pour copier l'ARN, cliquez sur
   Voici un exemple d'ARN :

   ```
   arn:aws:lambda:us-east-1:123456789012:function:HelloFunction
   ```

6. Copiez le code suivant pour la fonction Lambda dans la section Code source de la ***HelloFunction***page.

   ```
   export const handler = async(event, context, callback) => {
       callback(null, "Hello from " + event.who + "!");
   };
   ```

   Ce code crée une salutation à l'aide du champ `who`des données d'entrée, qui sont fournies par l'objet `event` transmis à votre fonction. Vous ajouterez les données d'entrée de cette fonction ultérieurement, lorsque vous commencerez une nouvelle exécution. La méthode `callback` renvoie la salutation créée à partir de votre fonction.
7. Choisissez Deploy (Déployer).

Étape 2 : tester la fonction Lambda

Testez votre fonction Lambda pour voir si elle fonctionne.

1. Sélectionnez Tester).
2. Dans Event name (Nom de l'événement), saisissez `HelloEvent`.
3. Remplacez les données JSON de l'événement par les données suivantes.

   ```
   {
       "who": "AWS Step Functions"
   }
   ```

L'`"who"`entrée correspond au `event.who` champ de votre fonction Lambda, qui complète le message d'accueil. Vous saisirez les mêmes données d'entrée lorsque vous exécuterez votre machine à états.

4. Choisissez Enregistrer, puis sélectionnez Test.
5. Pour examiner les résultats du test, sous Execution result (Résultat de l'exécution), développez Details (Détails).

Étape 3 : Création d'une machine à états

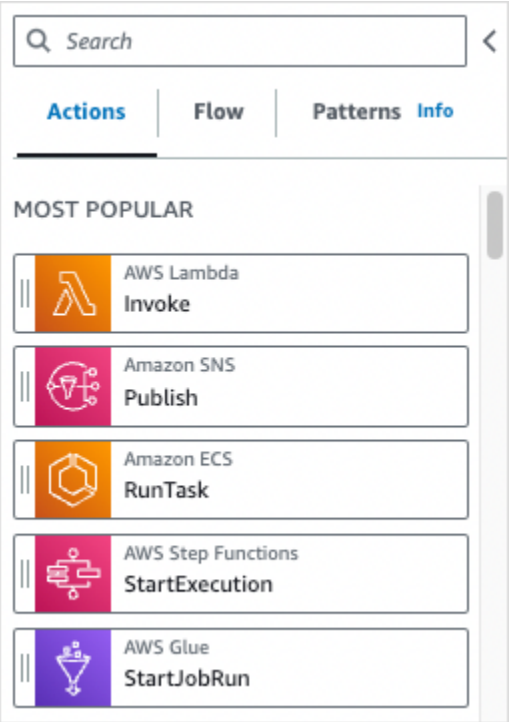
Utilisez la console Step Functions pour créer une machine à états qui invoque la fonction Lambda que vous avez créée à l'étape 1.

1. Ouvrez la console Step Functions et choisissez Create state machine.

> **Important**
>
> Assurez-vous que votre machine d'état est associée au même AWS compte et à la même région que la fonction Lambda que vous avez créée précédemment.

2. Dans la boîte de dialogue Choisir un modèle, sélectionnez Vide.
3. Choisissez Select (Sélectionner). Cela ouvre Workflow Studio dansMode de conception.
4. Dans le navigateur States sur la gauche, assurez-vous d'avoir choisi l'onglet Actions. Ensuite, procédez comme suit :

   - Glissez et déposez l'API AWS Lambda Invoke dans l'état vide intitulé Drag first state here.



5. Dans le panneau Inspector sur la droite, configurez la fonction Lambda :

   a. Dans la section Paramètres de l'API, choisissez la fonction Lambda que vous avez créée précédemment dans la liste déroulante Nom de la fonction.

   b. Conservez la sélection par défaut dans la liste déroulante Charge utile.

6. (Facultatif) Choisissez Definition pour afficher la définition de la machine à états Amazon States Language (ASL), qui est automatiquement générée en fonction de vos sélections dans l'onglet Actions et dans le panneau Inspector.

7. Spécifiez un nom pour votre machine à états. Pour ce faire, cliquez sur l'icône d'édition à côté du nom de la machine à états par défaut de MyStateMachine. Ensuite, dans Configuration de la machine d'état, spécifiez un nom dans le champ Nom de la machine d'état.

   Par exemple, saisissez le nom **`LambdaStateMachine`**.

   > **Note**
   >
   > Les noms des machines d'état, des exécutions et des tâches d'activité ne doivent pas dépasser 80 caractères. Ces noms doivent être uniques pour votre compte et votre AWS région, et ne doivent contenir aucun des éléments suivants :
   >
   > - Espace blanc

- Caractères génériques () ? *
- Caractères entre crochets (< > { } [ ])
- Caractères spéciaux (" # % \ ^ | ~ ` $ & , ; : /)
- Caractères de contrôle (\\u0000- \\u001f ou \\u007f -\\u009f).

Si votre machine à états est de type Express, vous pouvez attribuer le même nom à plusieurs exécutions de la machine à états. Step Functions génère un ARN d'exécution unique pour chaque exécution automatique d'Express State, même si plusieurs exécutions portent le même nom.
Step Functions vous permet de créer des noms pour les machines d'état, les exécutions et les activités, ainsi que des étiquettes contenant des caractères non ASCII. Ces noms non ASCII ne fonctionnent pas avec Amazon. CloudWatch Pour être sûr de pouvoir suivre CloudWatch les métriques, choisissez un nom qui utilise uniquement des caractères ASCII.

8. (Facultatif) Dans Configuration de la machine à états, spécifiez d'autres paramètres de flux de travail, tels que le type de machine à états et son rôle d'exécution.

   Pour ce didacticiel, conservez toutes les sélections par défaut dans les paramètres State Machine.

9. Choisissez Créer.

10. Dans la boîte de dialogue Confirmer la création du rôle, choisissez Confirmer pour continuer.

    Vous pouvez également choisir Afficher les paramètres des rôles pour revenir à la configuration de la machine State.

> **Note**
>
> Si vous supprimez le rôle IAM créé par Step Functions, Step Functions ne pourra pas le recréer ultérieurement. De même, si vous modifiez le rôle (par exemple, en supprimant Step Functions des principes de la politique IAM), Step Functions ne pourra pas restaurer ses paramètres d'origine ultérieurement.

Étape 4 : Exécutez la machine d'état

Après avoir créé votre machine d'état, vous pouvez l'exécuter.

1. Sur la page State machines, choisissez LambdaStateMachine.
2. Choisissez Start execution (Démarrer l'exécution).

   La boîte de dialogue Démarrer l'exécution s'affiche.
3. (Facultatif) Pour identifier votre exécution, vous pouvez lui donner un nom dans le champ Nom. Par défaut, Step Functions génère automatiquement un nom d'exécution unique.

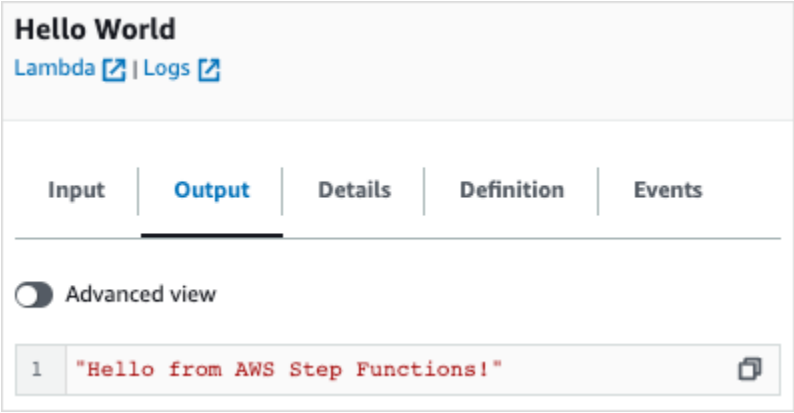
   > **Note**
   >
   > Step Functions vous permet de créer des noms pour les machines d'état, les exécutions et les activités, ainsi que des étiquettes contenant des caractères non ASCII. Ces noms non ASCII ne fonctionnent pas avec Amazon. CloudWatch Pour être sûr de pouvoir suivre CloudWatch les métriques, choisissez un nom qui utilise uniquement des caractères ASCII.
4. Dans la zone de saisie, remplacez les exemples de données d'exécution par les données suivantes.

   ```
   {
       "who" : "AWS Step Functions"
   }
   ```

   `"who"`est le nom clé que votre fonction Lambda utilise pour obtenir le nom de la personne à saluer.
5. Choisissez Démarrer une exécution.

   L'exécution de votre machine d'état démarre et une nouvelle page indiquant votre exécution en cours s'affiche.
6. La console Step Functions vous dirige vers une page intitulée avec votre ID d'exécution. Cette page est connue sous le nom de page Détails de l'exécution. Sur cette page, vous pouvez consulter les résultats de l'exécution au fur et à mesure que l'exécution progresse ou une fois celle-ci terminée.

Pour consulter les résultats de l'exécution, choisissez des états individuels dans la vue graphique, puis choisissez les onglets individuels du Détails de l'étape volet pour afficher les détails de chaque état, y compris les entrées, les sorties et la définition respectivement. Pour plus de détails sur les informations d'exécution que vous pouvez consulter sur la page Détails de l'exécution, voirPage de détails d'exécution — Vue d'ensemble de l'interface.



> **Note**
>
> Vous pouvez également transmettre des charges utiles lorsque vous invoquez Lambda depuis une machine à états. Pour plus d'informations et des exemples sur l'invocation de Lambda en transmettant une charge utile dans `Parameters` le champ, consultez. Invoquez Lambda avec Step Functions

Gestion des conditions d'erreur à l'aide d'une machine à états Step Functions

Dans ce didacticiel, vous allez créer une machine à AWS Step Functions états avec un États de repli champ. Le `Catch` champ utilise une AWS Lambda fonction pour répondre selon une logique conditionnelle basée sur le type de message d'erreur. Il s'agit d'une technique appelée gestion des erreurs de fonction.

Pour plus d'informations, consultez AWS Lambda la section Erreurs de fonctionnement dans Node.js dans le manuel du AWS Lambda développeur.

> **Note**
>
> Vous pouvez également créer des machines à états qui réessayent en cas d'expiration du délai ou qui permettent de passer `Catch` à un état spécifique en cas d'erreur ou de délai d'expiration. Pour accéder à des exemples de ces techniques de gestion des erreurs, consultez Exemples utilisant Retry et utilisant Catch.

Dans cette rubrique :

Étape 1 : créer une fonction Lambda qui échoue

Utilisez une fonction Lambda pour simuler une condition d'erreur.

> **Important**
>
> Assurez-vous que votre fonction Lambda se trouve sous le même AWS compte et dans la même AWS région que votre machine à états.

1. Ouvrez la AWS Lambda console à l'adresse https://console.aws.amazon.com/lambda/.
2. Choisissez Créer une fonction.
3. Choisissez Utiliser un plan, entrez `step-functions` dans le champ de recherche, puis choisissez l'option Lancer un plan d'erreur personnalisé.
4. Sous Nom de la fonction, saisissez `FailFunction`.
5. Pour Rôle, conservez la sélection par défaut (Créez un nouveau rôle avec des autorisations Lambda de base).
6. Le code suivant s'affiche dans le volet des codes de fonction Lambda.

```
exports.handler = async (event, context) => {
    function CustomError(message) {
        this.name = 'CustomError';
```

```
        this.message = message;
    }
    CustomError.prototype = new Error();

    throw new CustomError('This is a custom error!');
};
```

L'objet `context` renvoie le message d'erreur `This is a custom error!`.

7. Choisissez Créer une fonction.
8. Une fois votre fonction Lambda créée, copiez le nom de ressource Amazon (ARN) de la fonction affiché dans le coin supérieur droit de la page. Pour copier l'ARN, cliquez sur
   Voici un exemple d'ARN :

```
arn:aws:lambda:us-east-1:123456789012:function:FailFunction
```

9. Choisissez Deploy (Déployer).

Étape 2 : tester la fonction Lambda

Testez votre fonction Lambda pour voir si elle fonctionne.

1. Sur la FailFunctionpage, choisissez l'onglet Test, puis sélectionnez Test. Il n'est pas nécessaire de créer un événement de test.
2. Pour consulter les résultats du test (erreur simulée), sous Résultat de l'exécution, développez Détails.

Étape 3 : Création d'une machine à états avec un champ Catch

Utilisez la console Step Functions pour créer une machine à états qui utilise un État de la tâche état avec un `Catch` champ. Ajoutez une référence à votre fonction Lambda dans l'état Task. La machine à états invoque la fonction Lambda, qui échoue lors de l'exécution. Step Functions réessaie la fonction deux fois en utilisant un décalage exponentiel entre les tentatives.

1. Ouvrez la console Step Functions et choisissez Create state machine.
2. Dans la boîte de dialogue Choisir un modèle, sélectionnez Vide.
3. Choisissez Select (Sélectionner). Cela ouvre Workflow Studio dansMode de conception.

4. Choisissez Code pour ouvrir l'éditeur de code. Dans l'éditeur de code, vous écrivez et modifiez la définition Amazon States Language (ASL) de vos flux de travail.

5. Collez le code suivant, mais remplacez l'ARN de la fonction Lambda que vous avez créée précédemment dans le `Resource` champ.

```
{
   "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
      }
   }
}
```

Voici une description de votre machine à états utilisant le langage Amazon States. Elle décrit un seul état `Task` nommé `CreateAccount`. Pour plus d'informations, consultez Structure de la machine d'état.

Pour plus d'informations sur la syntaxe du champ `Retry`, consultez Exemples de machines à états utilisant Retry et Catch.

> **Note**
>
> Les erreurs non gérées dans Lambda sont signalées `Lambda.Unknown` comme dans le résultat d'erreur. Il s'agit notamment out-of-memory des erreurs et des délais d'expiration des fonctions. Vous pouvez effectuer une correspondance ou `States.TaskFailed` pour gérer ces erreurs. `Lambda.Unknown` `States.ALL` Lorsque Lambda atteint le nombre maximum d'appels, l'erreur est. `Lambda.TooManyRequestsException` Pour plus d'informations sur les erreurs liées aux fonctions Lambda, consultez la section Gestion des erreurs et tentatives automatiques dans le Guide du AWS Lambda développeur.

6. (Facultatif) Dans leVolet de visualisation de graphes, consultez la visualisation graphique en temps réel de votre flux de travail.
7. Spécifiez un nom pour votre machine à états. Pour ce faire, cliquez sur l'icône d'édition à côté du nom de la machine à états par défaut de MyStateMachine. Ensuite, dans Configuration de la machine d'état, spécifiez un nom dans le champ Nom de la machine d'état.

   Dans le cadre de ce didacticiel, entrez **`Catchfailure`**.
8. (Facultatif) Dans Configuration de la machine à états, spécifiez d'autres paramètres de flux de travail, tels que le type de machine à états et son rôle d'exécution.

   Pour ce didacticiel, conservez toutes les sélections par défaut dans les paramètres State Machine.
9. Dans la boîte de dialogue Confirmer la création du rôle, choisissez Confirmer pour continuer.

   Vous pouvez également choisir Afficher les paramètres des rôles pour revenir à la configuration de la machine State.

> **Note**
>
> Si vous supprimez le rôle IAM créé par Step Functions, Step Functions ne pourra pas le recréer ultérieurement. De même, si vous modifiez le rôle (par exemple, en supprimant Step Functions des principes de la politique IAM), Step Functions ne pourra pas restaurer ses paramètres d'origine ultérieurement.

Étape 4 : Exécutez la machine d'état

Après avoir créé votre machine d'état, vous pouvez l'exécuter.

1. Sur la page State machines, choisissez Catchfailure.
2. Sur la page Catchfailure, choisissez Démarrer l'exécution. La boîte de dialogue Démarrer l'exécution s'affiche.
3. Dans la boîte de dialogue Démarrer l'exécution, procédez comme suit :

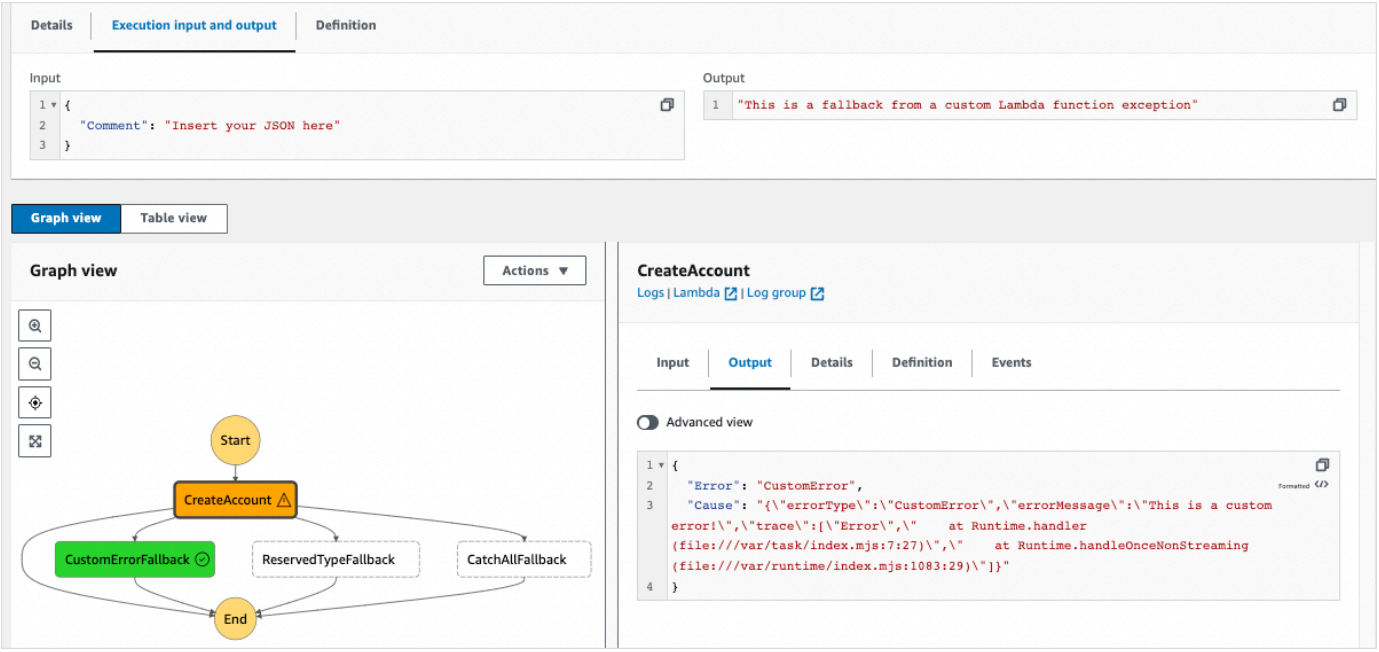
    1. (Facultatif) Pour identifier votre exécution, vous pouvez lui donner un nom dans le champ Nom. Par défaut, Step Functions génère automatiquement un nom d'exécution unique.

        > **Note**
        >
        > Step Functions vous permet de créer des noms pour les machines d'état, les exécutions et les activités, ainsi que des étiquettes contenant des caractères non ASCII. Ces noms non ASCII ne fonctionnent pas avec Amazon. CloudWatch Pour être sûr de pouvoir suivre CloudWatch les métriques, choisissez un nom qui utilise uniquement des caractères ASCII.

    2. (Facultatif) Dans la zone de saisie, entrez les valeurs d'entrée au format JSON pour exécuter votre flux de travail.
    3. Choisissez Start execution (Démarrer l'exécution).
    4. La console Step Functions vous dirige vers une page intitulée avec votre ID d'exécution. Cette page est connue sous le nom de page Détails de l'exécution. Sur cette page, vous pouvez consulter les résultats de l'exécution au fur et à mesure que l'exécution progresse ou une fois celle-ci terminée.

Pour consulter les résultats de l'exécution, choisissez des états individuels dans la vue graphique, puis choisissez les onglets individuels du Détails de l'étape volet pour afficher les détails de chaque état, y compris les entrées, les sorties et la définition respectivement. Pour plus de détails sur les informations d'exécution que vous pouvez consulter sur la page Détails de l'exécution, voirPage de détails d'exécution — Vue d'ensemble de l'interface.

Par exemple, pour afficher votre message d'erreur personnalisé, choisissez l'CreateAccountétape dans la vue graphique, puis choisissez l'onglet Sortie.



> **Note**
>
> Vous pouvez conserver l'entrée de l'état avec l'erreur, à l'aide de l'option `ResultPath`. veuillez consulter ResultPath À utiliser pour inclure à la fois une erreur et une entrée dans un `Catch`.

Utilisation de l'état de la carte intégrée pour répéter une action

Ce didacticiel vous aide à commencer à utiliser l'`Map`état en mode Inline. Vous utilisez l'état Inline Map dans vos flux de travail pour effectuer une action à plusieurs reprises. Pour plus d'informations sur le mode Inline, voir État de la carte en mode Inline.

Dans ce didacticiel, vous allez utiliser l'état de la carte intégrée pour générer à plusieurs reprises des identifiants uniques universels de version 4 (UUID v4). Vous commencez par créer un flux de travail contenant deux Pass états et un état de carte intégrée dans Workflow Studio. Ensuite, vous configurez l'entrée et la sortie, y compris le tableau JSON d'entrée pour l'`Map`état. L'`Map`état renvoie un tableau de sortie qui contient les UUID v4 générés pour chaque élément du tableau d'entrée.

Table des matières

Étape 1 : Création du prototype de flux de travail

Au cours de cette étape, vous allez créer le prototype de votre flux de travail à l'aide de Workflow Studio. Workflow Studio est un concepteur visuel de flux de travail disponible dans la console Step Functions. Vous allez choisir les états requis dans l'onglet Flow et utiliser la fonction glisser-déposer de Workflow Studio pour créer le prototype du flux de travail.

1. Ouvrez la console Step Functions et choisissez Create state machine.
2. Dans la boîte de dialogue Choisir un modèle, sélectionnez Vide.
3. Choisissez Select (Sélectionner). Cela ouvre Workflow Studio dansMode de conception.
4. Dans l'onglet Flux, faites glisser un état Pass et déposez-le vers l'état vide intitulé Drag first state here.
5. Faites glisser un état de la carte et déposez-le en dessous de l'état Pass. Renommez l'état de la carte en. **`Map demo`**
6. Faites glisser un deuxième état Pass et déposez-le dans l'état de démonstration de la carte.
7. Renommez le deuxième état Pass en. **`Generate UUID`**

Étape 2 : Configuration de l'entrée et de la sortie

Au cours de cette étape, vous configurez l'entrée et la sortie pour tous les états de votre prototype de flux de travail. Tout d'abord, vous injectez des données fixes dans le flux de travail en utilisant le premier état Pass. Cet état Pass transmet ces données en entrée à l'état de démonstration de la

carte. Dans cette entrée, vous spécifiez le nœud qui contient le tableau d'entrée sur lequel l'état de démonstration de la carte doit être itéré. Vous définissez ensuite l'étape que l'état de démonstration de la carte doit répéter pour générer les UUID v4. Enfin, vous configurez le résultat à renvoyer à chaque répétition.

1. Choisissez le premier état Pass dans votre prototype de flux de travail. Dans l'onglet Sortie, entrez ce qui suit sous Résultat :

```
{
  "foo": "bar",
  "colors": [
    "red",
    "green",
    "blue",
    "yellow",
    "white"
  ]
}
```

2. Choisissez l'état de démonstration de la carte et dans l'onglet Configuration, procédez comme suit :

   a. Choisissez Fournir un chemin vers le tableau d'éléments.

   b. Spécifiez le chemin de référence suivant pour sélectionner le nœud qui contient le tableau d'entrée :

```
$.colors
```

3. Choisissez l'état Generate UUID et dans l'onglet Input, procédez comme suit :

   a. Choisissez Transformer l'entrée avec paramètres.

   b. Entrez l'entrée JSON suivante pour générer les UUID v4 pour chacun des éléments du tableau d'entrée. Vous utilisez la fonction `States.UUID` intrinsèque pour générer les UUID.

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

4. Pour l'état Generate UUID, choisissez l'onglet Output et procédez comme suit :

   a. Choisissez Filtrer la sortie avec OutputPath.

b. Entrez le chemin de référence suivant pour sélectionner le nœud JSON qui contient les éléments du tableau de sortie :

```
$.uuid
```

Étape 3 : Vérifiez la définition du langage Amazon States générée automatiquement et enregistrez le flux de travail

Lorsque vous glissez et déposez des états du panneau Flow vers le canevas, Workflow Studio compose automatiquement la définition Amazon States Language (ASL) de votre flux de travail en temps réel. Vous pouvez modifier cette définition selon vos besoins.

1. (Facultatif) Choisissez Definition dans le Inspector panneau pour afficher la définition Amazon States Language générée automatiquement de votre flux de travail.

> **Tip**
>
> Vous pouvez également consulter la définition ASL dans Workflow Studio. Éditeur de code Dans l'éditeur de code, vous pouvez également modifier la définition ASL de votre flux de travail.

L'exemple suivant montre la définition du langage Amazon States générée automatiquement pour votre flux de travail.

```
{
    "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. Spécifiez un nom pour votre machine à états. Pour ce faire, cliquez sur l'icône d'édition à côté du nom de la machine à états par défaut de MyStateMachine. Ensuite, dans Configuration de la machine d'état, spécifiez un nom dans le champ Nom de la machine d'état.

   Pour ce didacticiel, saisissez le nom **`InlineMapDemo`**.

3. (Facultatif) Dans Configuration de la machine à états, spécifiez d'autres paramètres de flux de travail, tels que le type de machine à états et son rôle d'exécution.

   Pour ce didacticiel, conservez toutes les sélections par défaut dans la configuration State Machine.

4. Dans la boîte de dialogue Confirmer la création du rôle, choisissez Confirmer pour continuer.

Vous pouvez également choisir Afficher les paramètres des rôles pour revenir à la configuration de la machine State.

> **Note**
>
> Si vous supprimez le rôle IAM créé par Step Functions, Step Functions ne pourra pas le recréer ultérieurement. De même, si vous modifiez le rôle (par exemple, en supprimant Step Functions des principes de la politique IAM), Step Functions ne pourra pas restaurer ses paramètres d'origine ultérieurement.

Étape 4 : Exécutez la machine d'état

Les exécutions par State Machine sont des instances dans lesquelles vous exécutez votre flux de travail pour effectuer des tâches.

1. Sur la InlineMapDemopage, choisissez Démarrer l'exécution.
2. Dans la boîte de dialogue Démarrer l'exécution, procédez comme suit :

    1. (Facultatif) Pour identifier votre exécution, vous pouvez lui donner un nom dans le champ Nom. Par défaut, Step Functions génère automatiquement un nom d'exécution unique.

        > **Note**
        >
        > Step Functions vous permet de créer des noms pour les machines d'état, les exécutions, les activités et les étiquettes contenant des caractères non ASCII. Ces noms non ASCII ne fonctionnent pas avec Amazon. CloudWatch Pour être sûr de pouvoir suivre CloudWatch les métriques, choisissez un nom qui utilise uniquement des caractères ASCII.

    2. (Facultatif) Dans la zone de saisie, entrez les valeurs d'entrée au format JSON pour exécuter votre flux de travail.
    3. Choisissez Start execution (Démarrer l'exécution).
    4. La console Step Functions vous dirige vers une page intitulée avec votre ID d'exécution. Cette page est connue sous le nom de page Détails de l'exécution. Sur cette page, vous pouvez consulter les résultats de l'exécution au fur et à mesure que l'exécution progresse ou une fois celle-ci terminée.

Pour consulter les résultats de l'exécution, choisissez des états individuels dans la vue graphique, puis choisissez les onglets individuels du Détails de l'étape volet pour afficher les détails de chaque état, y compris les entrées, les sorties et la définition respectivement. Pour plus de détails sur les informations d'exécution que vous pouvez consulter sur la page Détails de l'exécution, voirPage de détails d'exécution — Vue d'ensemble de l'interface.

Pour afficher les entrées et sorties d'exécution side-by-side, choisissez Entrée et sortie d'exécution. Sous Sortie, consultez le tableau de sortie renvoyé par l'`Map`état. Voici un exemple de tableau de sortie :

```
[
  "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"
]
```

Copie de données CSV à grande échelle à l'aide d'une carte distribuée

Ce didacticiel vous aide à commencer à utiliser l'`Map`état en mode distribué. Un `Map` état défini sur Distribué est appelé état de carte distribuée. Vous utilisez l'état de la carte distribuée dans vos flux de travail pour itérer sur des sources de données Amazon S3 à grande échelle. L'`Map`état exécute chaque itération en tant qu'exécution d'un flux de travail secondaire, ce qui permet une simultanéité élevée. Pour plus d'informations sur le mode distribué, consultez la section État de la carte en mode distribué.

Dans ce didacticiel, vous allez utiliser l'état de la carte distribuée pour itérer sur un fichier CSV dans un compartiment Amazon S3. Vous renvoyez ensuite son contenu, ainsi que l'ARN d'une exécution de flux de travail enfant, dans un autre compartiment Amazon S3. Vous commencez par créer un prototype de flux de travail dans le Workflow Studio. Ensuite, vous définissez le mode de traitement de `Map` l'état sur Distribué, vous spécifiez le fichier CSV comme ensemble de données et vous indiquez son emplacement à l'`Map`État. Vous spécifiez également le type de flux de travail pour les exécutions de flux de travail enfants dont l'état de carte distribuée commence par Express.

Outre ces paramètres, vous spécifiez également d'autres configurations, telles que le nombre maximum d'exécutions simultanées de flux de travail enfant et l'emplacement d'exportation du `Map` résultat, pour l'exemple de flux de travail utilisé dans ce didacticiel.

Table des matières

Prérequis

- Chargez un fichier CSV dans un compartiment Amazon S3. Vous devez définir une ligne d'en-tête dans votre fichier CSV. Pour plus d'informations sur les limites de taille imposées au fichier CSV et sur la manière de spécifier la ligne d'en-tête, consultezFichier CSV dans un compartiment Amazon S3.
- Créez un autre compartiment Amazon S3 et un dossier dans ce compartiment vers lequel exporter le résultat de `Map` l'état.

> **Important**
>
> Assurez-vous que vos compartiments Amazon S3 se trouvent sous le même Compte AWS emplacement Région AWS que votre machine à états.

Étape 1 : Création du prototype de flux de travail

Au cours de cette étape, vous allez créer le prototype de votre flux de travail à l'aide de Workflow Studio. Workflow Studio est un concepteur visuel de flux de travail disponible dans la console

Step Functions. Vous choisissez l'état et l'action d'API requis dans les onglets Flux et Actions respectivement. Vous allez utiliser la fonction glisser-déposer de Workflow Studio pour créer le prototype du flux de travail.

1. Ouvrez la console Step Functions et choisissez Create state machine.
2. Dans la boîte de dialogue Choisir un modèle, sélectionnez Vide.
3. Choisissez Select (Sélectionner). Cela ouvre Workflow Studio dansMode de conception.
4. Dans l'onglet Flux, faites glisser un état de la carte et déposez-le dans l'état vide intitulé Drag first state here.
5. Dans l'onglet Configuration, saisissez le nom de l'État**`Process data`**.
6. Dans l'onglet Actions, faites glisser une action de l'API AWS Lambda Invoke et déposez-la dans l'état des données du processus.
7. Renommez l'état AWS Lambda Invoke en. **`Process CSV data`**

Étape 2 : Configuration des champs obligatoires pour l'état de la carte

Au cours de cette étape, vous configurez les champs obligatoires suivants de l'état de la carte distribuée :

- ItemReader— Spécifie l'ensemble de données et son emplacement à partir duquel l'`Map`état peut lire les entrées.
- ItemProcessor— Spécifie les valeurs suivantes :
  - `ProcessorConfig`— Définissez `EXPRESS` respectivement `ExecutionType` les `Mode` `DISTRIBUTED` et et. Cela définit le mode de traitement de `Map` l'état et le type de flux de travail pour les exécutions de flux de travail enfants lancées par l'état Distributed Map.
  - `StartAt`— Le premier état du flux de travail cartographique.
  - `States`— Définit le flux de travail Map, qui est un ensemble d'étapes à répéter lors de l'exécution de chaque flux de travail enfant.
- ResultWriter— Spécifie l'emplacement Amazon S3 où Step Functions écrit les résultats de l'état de la carte distribuée.

> **Important**
>
> Assurez-vous que le compartiment Amazon S3 que vous utilisez pour exporter les résultats d'un Map Run se trouve sous le même Compte AWS emplacement Région

AWS que votre machine d'état. Sinon, l'exécution de votre machine d'état échouera avec l'`States.ResultWriterFailed`erreur.

Pour configurer les champs obligatoires :

1. Choisissez l'état des données de processus et, dans l'onglet Configuration, procédez comme suit :
   a. Pour le mode de traitement, choisissez Distribué.
   b. Pour Source de l'article, choisissez Amazon S3, puis choisissez le fichier CSV dans S3 dans la liste déroulante des sources de l'article S3.
   c. Procédez comme suit pour spécifier l'emplacement Amazon S3 de votre fichier CSV :
      i. Pour l'objet S3, sélectionnez Enter bucket and key dans la liste déroulante.
      ii. Pour Bucket, entrez le nom du compartiment Amazon S3, qui contient le fichier CSV. Par exemple, **`sourceBucket`**.
      iii. Pour Key, entrez le nom de l'objet Amazon S3 dans lequel vous avez enregistré le fichier CSV. Vous devez également indiquer le nom du fichier CSV dans ce champ. Par exemple, **`csvDataset/ratings.csv`**.
   d. Pour les fichiers CSV, vous devez également spécifier l'emplacement de l'en-tête de colonne. Pour ce faire, choisissez Configuration supplémentaire, puis pour l'emplacement de l'en-tête CSV, conservez la sélection par défaut de Première ligne si la première ligne de votre fichier CSV est l'en-tête. Sinon, choisissez Given pour spécifier l'en-tête dans la définition de la machine à états. Pour plus d'informations, consultez `ReaderConfig`.
   e. Pour le type d'exécution Child, choisissez Express.
2. Dans Emplacement d'exportation, pour exporter les résultats de Map Run vers un emplacement Amazon S3 spécifique, choisissez Exporter la sortie de l'état de la carte vers Amazon S3.
3. Procédez comme suit :
   a. Pour le compartiment S3, choisissez Enter bucket name and prefix (Enter bucket name and prefix) dans la liste déroulante.
   b. Pour Bucket, entrez le nom du compartiment Amazon S3 vers lequel vous souhaitez exporter les résultats. Par exemple, **`mapOutputs`**.
   c. Dans Préfixe, entrez le nom du dossier dans lequel vous souhaitez enregistrer les résultats. Par exemple, **`resultData`**.

Étape 3 : Configuration des options supplémentaires

Outre les paramètres requis pour l'état d'une carte distribuée, vous pouvez également définir d'autres options. Il peut s'agir du nombre maximum d'exécutions simultanées d'un flux de travail enfant et de l'emplacement vers lequel exporter le résultat de `Map` l'état.

1. Choisissez l'état des données du processus. Ensuite, dans Source de l'article, sélectionnez Configuration supplémentaire.
2. Procédez comme suit :

   a. Choisissez Modifier les éléments avec ItemSelector pour spécifier une entrée JSON personnalisée pour chaque exécution de flux de travail enfant.

   b. Entrez l'entrée JSON suivante :

   ```
   {
     "index.$": "$$.Map.Item.Index",
     "value.$": "$$.Map.Item.Value"
   }
   ```

   Pour plus d'informations sur la création d'une entrée personnalisée, consultez`ItemSelector`.
3. Dans les paramètres d'exécution, pour Limite de simultanéité, spécifiez le nombre d'exécutions simultanées de flux de travail enfants que l'état de carte distribuée peut démarrer. Par exemple, saisissez **`100`**.
4. Ouvrez une nouvelle fenêtre ou un nouvel onglet sur votre navigateur et terminez la configuration de la fonction Lambda que vous utiliserez dans ce flux de travail, comme expliqué dans. Étape 4 : Configuration de la fonction Lambda

Étape 4 : Configuration de la fonction Lambda

> **Important**
>
> Assurez-vous que votre fonction Lambda est identique à celle de votre Région AWS machine à états.

1. Ouvrez la console Lambda et choisissez Create function.

2. Sur la page Create function, sélectionnez Author from scratch.
3. Dans la section Informations de base, configurez votre fonction Lambda :
   a. Sous Nom de la fonction, saisissez **`distributedMapLambda`**.
   b. Pour Exécution, choisissez Node.js 16.x.
   c. Conservez toutes les sélections par défaut et choisissez Créer une fonction.
   d. Après avoir créé votre fonction Lambda, copiez le nom de ressource Amazon (ARN) de la fonction affiché dans le coin supérieur droit de la page. Vous devrez le fournir dans votre prototype de flux de travail. Pour copier l'ARN, cliquez sur
   Voici un exemple d'ARN :

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

4. Copiez le code suivant pour la fonction Lambda et collez-le dans la section Code source de la distributedMapLambdapage.

   ```
   exports.handler = async function(event, context) {
       console.log("Received Input:\n", event);

       return {
           'statusCode' : 200,
           'inputReceived' : event //returns the input that it received
       }
   };
   ```

5. Choisissez Deploy (Déployer). Une fois votre fonction déployée, choisissez Test pour voir le résultat de votre fonction Lambda.

Étape 5 : Mettre à jour le prototype du flux de travail

Dans la console Step Functions, vous allez mettre à jour votre flux de travail pour ajouter l'ARN de la fonction Lambda.

1. Retournez à l'onglet ou à la fenêtre dans lequel vous avez créé le prototype de flux de travail.
2. Choisissez l'étape Traiter les données CSV, puis dans l'onglet Configuration, procédez comme suit :

a. Pour le type d'intégration, choisissez Optimisé.

b. Pour Nom de la fonction, commencez par saisir le nom de votre fonction Lambda. Choisissez la fonction dans la liste déroulante qui apparaît, ou choisissez Enter function name (Entrez le nom de la fonction et indiquez l'ARN de la fonction Lambda).

Étape 6 : Vérifiez la définition du langage Amazon States générée automatiquement et enregistrez le flux de travail

Lorsque vous glissez et déposez les états des onglets Action et Flow sur le canevas, Workflow Studio compose automatiquement la définition du langage Amazon States de votre flux de travail en temps réel. Vous pouvez modifier cette définition selon vos besoins.

1. (Facultatif) Choisissez Definition dans le Inspector panneau et visualisez la définition de la machine à états.

> **Tip**
>
> Vous pouvez également consulter la définition ASL dans Workflow Studio. Éditeur de code Dans l'éditeur de code, vous pouvez également modifier la définition ASL de votre flux de travail.

L'exemple de code suivant montre la définition du langage Amazon States générée automatiquement pour votre flux de travail.

```
{
  "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. Spécifiez un nom pour votre machine à états. Pour ce faire, cliquez sur l'icône d'édition à côté du nom de la machine à états par défaut de MyStateMachine. Ensuite, dans Configuration de la machine d'état, spécifiez un nom dans le champ Nom de la machine d'état.

   Pour ce didacticiel, saisissez le nom **`DistributedMapDemo`**.

3. (Facultatif) Dans Configuration de la machine à états, spécifiez d'autres paramètres de flux de travail, tels que le type de machine à états et son rôle d'exécution.

   Pour ce didacticiel, conservez toutes les sélections par défaut dans la configuration State Machine.

4. Dans la boîte de dialogue Confirmer la création du rôle, choisissez Confirmer pour continuer.

   Vous pouvez également choisir Afficher les paramètres des rôles pour revenir à la configuration de la machine State.

   > **Note**
   >
   > Si vous supprimez le rôle IAM créé par Step Functions, Step Functions ne pourra pas le recréer ultérieurement. De même, si vous modifiez le rôle (par exemple, en supprimant Step Functions des principes de la politique IAM), Step Functions ne pourra pas restaurer ses paramètres d'origine ultérieurement.

Étape 7 : Exécutez la machine d'état

Une exécution est une instance de votre machine à états dans laquelle vous exécutez votre flux de travail pour effectuer des tâches.

1. Sur la DistributedMapDemopage, choisissez Démarrer l'exécution.
2. Dans la boîte de dialogue Démarrer l'exécution, procédez comme suit :

   1. (Facultatif) Pour identifier votre exécution, vous pouvez lui donner un nom dans le champ Nom. Par défaut, Step Functions génère automatiquement un nom d'exécution unique.

      > **Note**
      >
      > Step Functions vous permet de créer des noms pour les machines d'état, les exécutions, les activités et les étiquettes contenant des caractères non ASCII. Ces

noms non ASCII ne fonctionnent pas avec Amazon. CloudWatch Pour être sûr de pouvoir suivre CloudWatch les métriques, choisissez un nom qui utilise uniquement des caractères ASCII.

2. (Facultatif) Dans la zone de saisie, entrez les valeurs d'entrée au format JSON pour exécuter votre flux de travail.
3. Choisissez Start execution (Démarrer l'exécution).
4. La console Step Functions vous dirige vers une page intitulée avec votre ID d'exécution. Cette page est connue sous le nom de page Détails de l'exécution. Sur cette page, vous pouvez consulter les résultats de l'exécution au fur et à mesure que l'exécution progresse ou une fois celle-ci terminée.

   Pour consulter les résultats de l'exécution, choisissez des états individuels dans la vue graphique, puis choisissez les onglets individuels du Détails de l'étape volet pour afficher les détails de chaque état, y compris les entrées, les sorties et la définition respectivement. Pour plus de détails sur les informations d'exécution que vous pouvez consulter sur la page Détails de l'exécution, voirPage de détails d'exécution — Vue d'ensemble de l'interface.

Par exemple, choisissez l'`Map`état, puis choisissez Map Run pour ouvrir la page Map Run Details. Sur cette page, vous pouvez consulter tous les détails d'exécution relatifs à l'état de la carte distribuée et aux exécutions du flux de travail enfant qu'elle a lancées. Pour plus d'informations sur cette page, consultezExamen de Map Run.

Traitement d'un lot complet de données avec une fonction Lambda

Dans ce didacticiel, vous allez utiliser le ItemBatcher champ d'état de la carte distribuée pour traiter un lot complet d'éléments dans une fonction Lambda. Chaque lot contient un maximum de trois articles. L'état de la carte distribuée lance quatre exécutions de flux de travail enfants, où chaque exécution traite trois éléments, tandis qu'une exécution traite un seul élément. Chaque exécution d'un flux de travail enfant invoque une fonction Lambda qui itère sur les éléments individuels présents dans le lot.

Vous allez créer une machine à états qui effectue la multiplication sur un tableau d'entiers. Supposons que le tableau d'entiers que vous fournissez en entrée est `[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]` et que le facteur de multiplication est7. Ensuite, le tableau résultant formé après avoir multiplié ces entiers par un facteur de 7 sera. `[7, 14, 21, 28, 35, 42, 49, 56, 63, 70]`

Rubriques

Étape 1 : Création de la machine à états

Au cours de cette étape, vous créez le prototype de flux de travail de la machine à états qui transmet un lot complet de données à la fonction Lambda que vous allez créer à l'étape 2.

- Utilisez la définition suivante pour créer une machine à états à l'aide de la console Step Functions. Pour plus d'informations sur la création d'une machine à états, consultez Étape 1 : Création du prototype de flux de travail le didacticiel Getting started with using Distributed Map state.

  Dans cette machine à états, vous définissez un état de carte distribuée qui accepte un tableau de 10 entiers en entrée et transmet ce tableau à une fonction Lambda par lots de. 3 La fonction Lambda itère sur les éléments individuels présents dans le lot et renvoie un tableau de sortie nommé. `multiplied` Le tableau de sortie contient le résultat de la multiplication effectuée sur les éléments passés dans le tableau d'entrée.

  > **Important**
  >
  > Assurez-vous de remplacer l'Amazon Resource Name (ARN) de la fonction Lambda dans le code suivant par l'ARN de la fonction que vous allez créer à l'étape 2.

```
{
  "StartAt": "Pass",
  "States": {
    "Pass": {
      "Type": "Pass",
      "Next": "Map",
      "Result": {
        "MyMultiplicationFactor": 7,
        "MyItems": [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
      }
    },
```

```
    "Map": {
      "Type": "Map",
      "ItemProcessor": {
        "ProcessorConfig": {
          "Mode": "DISTRIBUTED",
          "ExecutionType": "STANDARD"
        },
        "StartAt": "Lambda Invoke",
        "States": {
          "Lambda Invoke": {
            "Type": "Task",
            "Resource": "arn:aws:states:::lambda:invoke",
            "OutputPath": "$.Payload",
            "Parameters": {
              "Payload.$": "$",
              "FunctionName": "arn:aws:lambda:us-
east-1:123456789012:function:functionName"
            },
            "Retry": [
              {
                "ErrorEquals": [
                  "Lambda.ServiceException",
                  "Lambda.AWSLambdaException",
                  "Lambda.SdkClientException",
                  "Lambda.TooManyRequestsException"
                ],
                "IntervalSeconds": 2,
                "MaxAttempts": 6,
                "BackoffRate": 2
              }
            ],
            "End": true
          }
        }
      },
      "End": true,
      "Label": "Map",
      "MaxConcurrency": 1000,
      "ItemBatcher": {
        "MaxItemsPerBatch": 3,
        "BatchInput": {
          "MyMultiplicationFactor.$": "$.MyMultiplicationFactor"
        }
      },
```

```
      "ItemsPath": "$.MyItems"
    }
  }
}
```

Étape 2 : Création de la fonction Lambda

Au cours de cette étape, vous créez la fonction Lambda qui traite tous les éléments transmis dans le lot.

> **Important**
>
> Assurez-vous que votre fonction Lambda est identique à celle de votre Région AWS machine à états.

Pour créer la fonction Lambda

1. Utilisez la console Lambda pour créer une fonction Lambda Python 3.9 nommée. **`ProcessEntireBatch`** Pour plus d'informations sur la création d'une fonction Lambda, voir Étape 4 : Configuration de la fonction Lambda dans le didacticiel Getting started with using Distributed Map state.
2. Copiez le code suivant pour la fonction Lambda et collez-le dans la section Code source de votre fonction Lambda.

```
import json

def lambda_handler(event, context):
    multiplication_factor = event['BatchInput']['MyMultiplicationFactor']
    items = event['Items']

    results = [multiplication_factor * item for item in items]

    return {
        'statusCode': 200,
        'multiplied': results
    }
```

3. Après avoir créé votre fonction Lambda, copiez l'ARN de la fonction affiché dans le coin supérieur droit de la page. Pour copier l'ARN, cliquez sur

le
.

Voici un exemple d'ARN, où *function-name*est le nom de la fonction Lambda (dans ce cas,`ProcessEntireBatch`) :

```
arn:aws:lambda:us-east-1:123456789012:function:function-name
```

Vous devrez fournir la fonction ARN dans la machine à états que vous avez créée à l'étape 1.

4. Choisissez Déployer pour déployer les modifications.

Étape 3 : Exécutez la machine d'état

Lorsque vous exécutez la machine à états, l'état de la carte distribuée lance quatre exécutions de flux de travail enfants, où chaque exécution traite trois éléments, tandis qu'une exécution traite un seul élément.

L'exemple suivant montre les données transmises à la `ProcessEntireBatch`fonction par l'une des exécutions du flux de travail enfant.

```
{
  "BatchInput": {
    "MyMultiplicationFactor": 7
  },
  "Items": [1, 2, 3]
}
```

Compte tenu de cette entrée, l'exemple suivant montre le tableau de sortie nommé `multiplied` renvoyé par la fonction Lambda.

```
{
  "statusCode": 200,
  "multiplied": [7, 14, 21]
}
```

La machine d'état renvoie la sortie suivante qui contient quatre tableaux nommés d'après `multiplied` les quatre exécutions de flux de travail enfants. Ces tableaux contiennent les résultats de multiplication des éléments d'entrée individuels.

```
[
```

```
  {
    "statusCode": 200,
    "multiplied": [7, 14, 21]
  },
  {
    "statusCode": 200,
    "multiplied": [28, 35, 42]
  },
  {
    "statusCode": 200,
    "multiplied": [49, 56, 63]
  },
  {
    "statusCode": 200,
    "multiplied": [70]
  }
]
```

Pour combiner tous les éléments du tableau renvoyés dans un seul tableau de sortie, vous pouvez utiliser le ResultSelector champ. Définissez ce champ dans l'état de la carte distribuée pour rechercher tous les `multiplied` tableaux, extraire tous les éléments contenus dans ces tableaux, puis les combiner en un seul tableau de sortie.

Pour utiliser le `ResultSelector` champ, mettez à jour la définition de votre machine à états comme indiqué dans l'exemple suivant.

```
{
  "StartAt": "Pass",
  "States": {
    ...
    ...
    "Map": {
      "Type": "Map",
      ...
      ...
      "ItemsPath": "$.MyItems",
      "ResultSelector": {
        "multiplied.$": "$..multiplied[*]"
      }
    }
  }
}
```

La machine d'état mise à jour renvoie un tableau de sortie consolidé, comme indiqué dans l'exemple suivant.

```
{
  "multiplied": [7, 14, 21, 28, 35, 42, 49, 56, 63, 70]
}
```

Traitement d'éléments de données individuels à l'aide d'une fonction Lambda

Dans ce didacticiel, vous allez utiliser le ItemBatcher champ d'état de la carte distribuée pour itérer sur les éléments individuels présents dans un lot à l'aide d'une fonction Lambda. L'état de la carte distribuée lance quatre exécutions de flux de travail secondaires. Chacun de ces flux de travail enfants exécute un état de carte intégrée. À chaque itération, l'état Inline Map invoque une fonction Lambda et transmet un seul élément du lot à la fonction. La fonction Lambda traite ensuite l'élément et renvoie le résultat.

Vous allez créer une machine à états qui effectue la multiplication sur un tableau d'entiers. Supposons que le tableau d'entiers que vous fournissez en entrée est `[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]` et que le facteur de multiplication est7. Ensuite, le tableau résultant formé après avoir multiplié ces entiers par un facteur de 7 sera. `[7, 14, 21, 28, 35, 42, 49, 56, 63, 70]`

Rubriques

- Étape 1 : Création de la machine à états
- Étape 2 : Création de la fonction Lambda
- Étape 3 : Exécutez la machine d'état

Étape 1 : Création de la machine à états

Au cours de cette étape, vous créez le prototype de flux de travail de la machine à états qui transmet un seul élément d'un lot d'éléments à chaque appel de la fonction Lambda que vous allez créer à l'étape 2.

- Utilisez la définition suivante pour créer une machine à états à l'aide de la console Step Functions. Pour plus d'informations sur la création d'une machine à états, consultez Étape 1 : Création du prototype de flux de travail le didacticiel Getting started with using Distributed Map state.

Dans cette machine à états, vous définissez un état de carte distribuée qui accepte un tableau de 10 entiers en entrée et transmet ces éléments du tableau aux flux de travail enfants exécutés par lots. Chaque exécution d'un flux de travail enfant reçoit un lot de trois éléments en entrée et exécute un état Inline Map. Chaque itération de l'état Inline Map appelle une fonction Lambda et transmet un élément du lot à la fonction. Cette fonction multiplie ensuite l'élément par un facteur de 7 et renvoie le résultat.

La sortie de chaque exécution de flux de travail enfant est un tableau JSON qui contient le résultat de la multiplication pour chacun des éléments transmis.

> **Important**
>
> Assurez-vous de remplacer l'Amazon Resource Name (ARN) de la fonction Lambda dans le code suivant par l'ARN de la fonction que vous allez créer à l'étape 2.

```
{
  "StartAt": "Pass",
  "States": {
    "Pass": {
      "Type": "Pass",
      "Next": "Map",
      "Result": {
        "MyMultiplicationFactor": 7,
        "MyItems": [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
      }
    },
    "Map": {
      "Type": "Map",
      "ItemProcessor": {
        "ProcessorConfig": {
          "Mode": "DISTRIBUTED",
          "ExecutionType": "STANDARD"
        },
        "StartAt": "InnerMap",
        "States": {
          "InnerMap": {
            "Type": "Map",
            "ItemProcessor": {
              "ProcessorConfig": {
```

```
                "Mode": "INLINE"
              },
              "StartAt": "Lambda Invoke",
              "States": {
                "Lambda Invoke": {
                  "Type": "Task",
                  "Resource": "arn:aws:states:::lambda:invoke",
                  "OutputPath": "$.Payload",
                  "Parameters": {
                    "Payload.$": "$",
                    "FunctionName": "arn:aws:lambda:us-
east-1:123456789012:function:functionName"
                  },
                  "Retry": [
                    {
                      "ErrorEquals": [
                        "Lambda.ServiceException",
                        "Lambda.AWSLambdaException",
                        "Lambda.SdkClientException",
                        "Lambda.TooManyRequestsException"
                      ],
                      "IntervalSeconds": 2,
                      "MaxAttempts": 6,
                      "BackoffRate": 2
                    }
                  ],
                  "End": true
                }
              }
            },
            "End": true,
            "ItemsPath": "$.Items",
            "ItemSelector": {
              "MyMultiplicationFactor.$": "$.BatchInput.MyMultiplicationFactor",
              "MyItem.$": "$$.Map.Item.Value"
            }
          }
        }
      },
      "End": true,
      "Label": "Map",
      "MaxConcurrency": 1000,
      "ItemsPath": "$.MyItems",
      "ItemBatcher": {
```

```
        "MaxItemsPerBatch": 3,
        "BatchInput": {
          "MyMultiplicationFactor.$": "$.MyMultiplicationFactor"
        }
      }
    }
  }
}
```

Étape 2 : Création de la fonction Lambda

Au cours de cette étape, vous créez la fonction Lambda qui traite chaque article transmis par le lot.

> **Important**
>
> Assurez-vous que votre fonction Lambda est identique à celle de votre Région AWS machine à états.

Pour créer la fonction Lambda

1. Utilisez la console Lambda pour créer une fonction Lambda Python 3.9 nommée. **`ProcessSingleItem`** Pour plus d'informations sur la création d'une fonction Lambda, voir Étape 4 : Configuration de la fonction Lambda dans le didacticiel Getting started with using Distributed Map state.
2. Copiez le code suivant pour la fonction Lambda et collez-le dans la section Code source de votre fonction Lambda.

```
import json

def lambda_handler(event, context):

    multiplication_factor = event['MyMultiplicationFactor']
    item = event['MyItem']

    result = multiplication_factor * item

    return {
        'statusCode': 200,
        'multiplied': result
```

```
    }
```

3. Après avoir créé votre fonction Lambda, copiez l'ARN de la fonction affiché dans le coin supérieur droit de la page. Pour copier l'ARN, cliquez sur le
   Voici un exemple d'ARN, où *function-name*est le nom de la fonction Lambda (dans ce cas,`ProcessSingleItem`) :

   ```
   arn:aws:lambda:us-east-1:123456789012:function:function-name
   ```

   Vous devrez fournir la fonction ARN dans la machine à états que vous avez créée à l'étape 1.
4. Choisissez Déployer pour déployer les modifications.

Étape 3 : Exécutez la machine d'état

Lorsque vous exécutez la machine à états, l'état de la carte distribuée lance quatre exécutions de flux de travail secondaires, chaque exécution traitant trois éléments, tandis qu'une exécution traite un seul élément.

L'exemple suivant montre les données transmises à l'une des invocations de `ProcessSingleItem`fonction dans le cadre de l'exécution d'un flux de travail enfant.

```
{
  "MyMultiplicationFactor": 7,
  "MyItem": 1
}
```

Compte tenu de cette entrée, l'exemple suivant montre la sortie renvoyée par la fonction Lambda.

```
{
  "statusCode": 200,
  "multiplied": 7
}
```

L'exemple suivant montre le tableau JSON de sortie pour l'une des exécutions du flux de travail enfant.

```
[
```

```
  {
    "statusCode": 200,
    "multiplied": 7
  },
  {
    "statusCode": 200,
    "multiplied": 14
  },
  {
    "statusCode": 200,
    "multiplied": 21
  }
]
```

La machine d'état renvoie la sortie suivante qui contient quatre tableaux pour les quatre exécutions de flux de travail secondaires. Ces tableaux contiennent les résultats de multiplication des éléments d'entrée individuels.

Enfin, la sortie de la machine à états est un tableau nommé `multiplied` qui combine tous les résultats de multiplication renvoyés pour les quatre exécutions de flux de travail secondaires.

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

Pour combiner tous les résultats de multiplication renvoyés par les exécutions du flux de travail enfant dans un seul tableau de sortie, vous pouvez utiliser le ResultSelector champ. Définissez ce champ dans l'état de la carte distribuée pour rechercher tous les résultats, extraire les résultats individuels, puis les combiner dans un seul tableau de sortie nommé`multiplied`.

Pour utiliser le `ResultSelector` champ, mettez à jour la définition de votre machine à états comme indiqué dans l'exemple suivant.

```
{
  "StartAt": "Pass",
  "States": {
    ...
    ...
    "Map": {
```

```
      "Type": "Map",
      ...
      ...
      "ItemBatcher": {
        "MaxItemsPerBatch": 3,
        "BatchInput": {
          "MyMultiplicationFactor.$": "$.MyMultiplicationFactor"
        }
      },
      "ItemsPath": "$.MyItems",
      "ResultSelector": {
        "multiplied.$": "$..multiplied"
      }
    }
  }
}
```

La machine d'état mise à jour renvoie un tableau de sortie consolidé, comme indiqué dans l'exemple suivant.

```
{
  "multiplied": [7, 14, 21, 28, 35, 42, 49, 56, 63, 70]
}
```

Démarrage d'une exécution State Machine en réponse à des événements Amazon S3

Vous pouvez exécuter une machine à AWS Step Functions états en réponse à une EventBridge règle Amazon.

Ce didacticiel explique comment configurer une machine à états en tant que cible pour une EventBridge règle Amazon. Cette règle lancera une exécution automatique lorsque des fichiers sont ajoutés à un bucket Amazon Simple Storage Service (Amazon S3).

Pour une application pratique, vous pouvez lancer une machine à états qui exécute des opérations sur les fichiers que vous ajoutez au bucket, telles que la création de vignettes ou l'exécution d'une analyse Amazon Rekognition sur des fichiers image et vidéo.

Dans ce didacticiel, vous lancez l'exécution d'une machine à `Helloworld` états en téléchargeant un fichier dans un compartiment Amazon S3. Vous examinez ensuite l'exemple d'entrée de cette

exécution pour identifier les informations incluses dans l'entrée de la notification d'événement Amazon S3 envoyée à EventBridge.

Rubriques

Prérequis : Création d'une machine d'état

Avant de pouvoir configurer une machine à états en tant que EventBridge cible Amazon, vous devez créer la machine à états.

- Pour créer une machine à états de base, utilisez le didacticiel Création d'une machine à états utilisant une fonction Lambda.
- Si vous disposez déjà d'une machine d'état `Helloworld`, passez à l'étape suivante.

Étape 1 : créer un compartiment dans Amazon S3

Maintenant que vous disposez d'une machine à `Helloworld` états, vous devez créer un compartiment Amazon S3 qui stocke vos fichiers. À l'étape 3 de ce didacticiel, vous avez défini une règle de telle sorte que lorsqu'un fichier est téléchargé dans ce compartiment, cela EventBridge déclenche l'exécution de votre machine à états.

1. Accédez à la console Amazon S3, puis choisissez Create bucket pour créer le compartiment dans lequel vous souhaitez stocker vos fichiers et déclencher une règle d'événement Amazon S3.
2. Entrez un Nom de compartiment, tel que *`username`*`-sfn-tutorial`.

> **Note**
>
> Les noms de compartiment doivent être uniques parmi tous les noms de compartiment existants dans toutes les AWS régions d'Amazon S3. Utilisez votre propre *nom*

d'utilisateur pour rendre ce nom unique. Vous devez créer toutes les ressources dans la même région AWS.

3. Conservez toutes les sélections par défaut sur la page, puis choisissez Create bucket.

Étape 2 : activer la notification d'événements Amazon S3 avec EventBridge

Après avoir créé le compartiment Amazon S3, configurez-le pour envoyer des événements EventBridge chaque fois que certains événements se produisent dans votre compartiment S3, tels que les téléchargements de fichiers.

1. Accédez à la console Amazon S3.
2. Dans la liste Buckets (Compartiments), choisissez le nom du compartiment pour lequel vous souhaitez activer les événements.
3. Choisissez Propriétés.
4. Faites défiler la page vers le bas pour afficher la section Notifications d'événements, puis choisissez Modifier dans la EventBridge sous-section Amazon.
5. Sous Envoyer des notifications à Amazon EventBridge pour tous les événements de ce compartiment, choisissez Activé.
6. Choisissez Save Changes (Enregistrer les modifications).

> **Note**
>
> Après l'activation EventBridge, il faut environ cinq minutes pour que les modifications prennent effet.

Étape 3 : créer une EventBridge règle Amazon

Une fois que vous disposez d'une machine à états, que vous avez créé le compartiment Amazon S3 et que vous l'avez configuré pour envoyer des notifications d'événements EventBridge, créez une EventBridge règle.

> **Note**
>
> Vous devez configurer la EventBridge règle dans la même AWS région que le compartiment Amazon S3.

Pour créer la règle

1. Accédez à la EventBridge console Amazon, puis choisissez Create rule.

   > **Tip**
   >
   > Dans le volet de navigation de la EventBridge console, vous pouvez également choisir Règles sous Bus, puis Créer une règle.

2. Entrez un nom pour votre règle (par exemple,*S3Step Functions*) et entrez éventuellement une description pour la règle.
3. Pour le bus d'événements et le type de règle, conservez les sélections par défaut.
4. Choisissez Suivant. Cela ouvre la page Créer un modèle d'événement.
5. Faites défiler la page jusqu'à la section Modèle d'événement, puis procédez comme suit :

   a. Pour Source d'événements, conservez la sélection par défaut d'AWSévénements ou d'événements EventBridge partenaires.

   b. Pour le AWSservice, choisissez Simple Storage Service (S3).

   c. Pour Type d'événement, choisissez Amazon S3 Event Notification.

   d. Choisissez Événement (s) spécifique (s), puis Object Created.

   e. Choisissez un ou plusieurs compartiments spécifiques par nom et entrez le nom du compartiment que vous avez créé à l'étape 1 (*username*`-sfn-tutorial`) pour stocker vos fichiers.

   f. Choisissez Suivant. Cela ouvre la page Sélectionner une ou plusieurs cibles.

Pour créer la cible

1. Dans Target 1, conservez la sélection de AWSservice par défaut.
2. Dans la liste déroulante Select a target, sélectionnez Step Functions state machine.

3. Dans la liste des machines à états, sélectionnez la machine à états que vous avez créée précédemment (par exemple,`Helloworld`).
4. Conservez toutes les sélections par défaut sur la page, puis choisissez Next. Cela ouvre la page Configurer les balises.
5. Choisissez Suivant à nouveau. Cela ouvre la page Réviser et créer.
6. Consultez les détails de la règle et choisissez Create rule (Créer une règle).

   La règle est créée et la page Règles s'affiche, répertoriant toutes vos EventBridge règles Amazon.

Étape 4 : Test de la règle

Maintenant que tout est en place, testez l'ajout d'un fichier au compartiment Amazon S3, puis examinez l'entrée de l'exécution de la machine à états qui en résulte.

1. Ajoutez un fichier à votre compartiment Amazon S3.

   Accédez à la console Amazon S3, choisissez le compartiment que vous avez créé pour stocker les fichiers (*`username`*`-sfn-tutorial`), puis choisissez Upload.
2. Ajoutez un fichier, par exemple*`test.png`*, puis choisissez Upload.

   Cela lance une exécution de votre machine d'état, en transmettant les informations depuis AWS CloudTrail comme entrée.
3. Vérifiez l'exécution de votre machine d'état.

   Accédez à la console Step Functions et sélectionnez la machine à états utilisée dans votre EventBridge règle Amazon (`Helloworld`).
4. Sélectionnez l'exécution la plus récente de cette machine à états et développez la section Execution Input.

   Cette section inclut des informations telles que le nom du compartiment et le nom d'objet. Dans un cas d'utilisation du monde réel, une machine d'état peut utiliser ces informations pour effectuer des actions sur cet objet.

Exemple d'entrée d'exécution

L'exemple suivant montre une entrée typique lors de l'exécution de la machine à états.

```
{
    "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"
    }
  }
```

Création d'une API Step Functions à l'aide d'API Gateway

Vous pouvez utiliser Amazon API Gateway pour associer vos AWS Step Functions API aux méthodes d'une API API Gateway. Lorsqu'une requête HTTPS est envoyée à une méthode d'API, API Gateway appelle vos actions d'API Step Functions.

Ce didacticiel vous montre comment créer une API qui utilise une ressource et la méthode `POST` pour communiquer avec l'action d'API `StartExecution`. Vous allez utiliser la console AWS Identity and Access Management (IAM) pour créer un rôle pour API Gateway. Vous allez ensuite utiliser la console API Gateway pour créer une API API Gateway, créer une ressource et une méthode, et associer la méthode à l'action de l'`StartExecution`API. En dernier lieu, vous déploierez et testerez votre API.

> **Note**
>
> Bien qu'Amazon API Gateway puisse démarrer une exécution de Step Functions en appelant`StartExecution`, vous devez appeler `DescribeExecution` pour obtenir le résultat.

Rubriques

Étape 1 : créer un rôle IAM pour API Gateway

Avant de créer votre API API Gateway, vous devez autoriser API Gateway à appeler les actions de l'API Step Functions.

Pour configurer les autorisations pour API Gateway

1. Connectez-vous à la console IAM et choisissez Rôles, Créer un rôle.
2. Sur la page Select trusted entity (Sélectionner une entité de confiance), procédez comme suit :
   a. Pour le type d'entité fiable, conservez la sélection par défaut de Service AWS.
   b. Dans le cas d'utilisation, choisissez API Gateway dans la liste déroulante.
3. Sélectionnez API Gateway, puis Next.
4. Sur la page Ajouter des autorisations, sélectionnez Suivant.
5. (Facultatif) Sur la page Nom, révision et création, entrez des informations, telles que le nom du rôle. Par exemple, saisissez **`APIGatewayToStepFunctions`**.
6. Sélectionnez Créer un rôle.

   Le rôle IAM apparaît dans la liste des rôles.
7. Choisissez le nom de votre rôle et notez l' ARN de rôle, comme illustré dans l'exemple suivant.

```
arn:aws:iam::123456789012:role/APIGatewayToStepFunctions
```

Pour associer une politique au rôle IAM

1. Sur la page Rôles, recherchez votre rôle (`APIGatewayToStepFunctions`), puis choisissez le rôle.
2. Dans l'onglet Autorisations, choisissez Ajouter des autorisations, puis choisissez Joindre des politiques.
3. Sur la page Joindre une politique, recherchez`AWSStepFunctionsFullAccess`, choisissez la politique, puis choisissez Ajouter des autorisations.

Étape 2 : Création de votre API API Gateway

Après avoir créé votre rôle IAM, vous pouvez créer votre API API Gateway personnalisée.

Pour créer l’API

1. Ouvrez la console Amazon API Gateway, puis choisissez Create API.
2. Sur la page Choisissez un type d'API, dans le volet API REST, choisissez Build.
3. Sur la page Créer une API REST, sélectionnez Nouvelle API, puis entrez ***`StartExecutionAPI`*** pour le nom de l'API.
4. Conservez le type de point de terminaison d'API régional, puis choisissez Create API.

Pour créer une ressource

1. Sur la page Ressources de l'***`StartExecutionAPI`***, choisissez Create resource.
2. Sur la page Créer une ressource, entrez le nom **`execution`** de la ressource, puis choisissez Créer une ressource.

Pour créer une méthode POST

1. Choisissez la ressource /execution, puis choisissez Create method.
2. Dans Type de méthode, sélectionnez`POST`.
3. Dans Type d'intégration, sélectionnez AWS service.
4. Pour Région AWS, choisissez une région dans la liste.

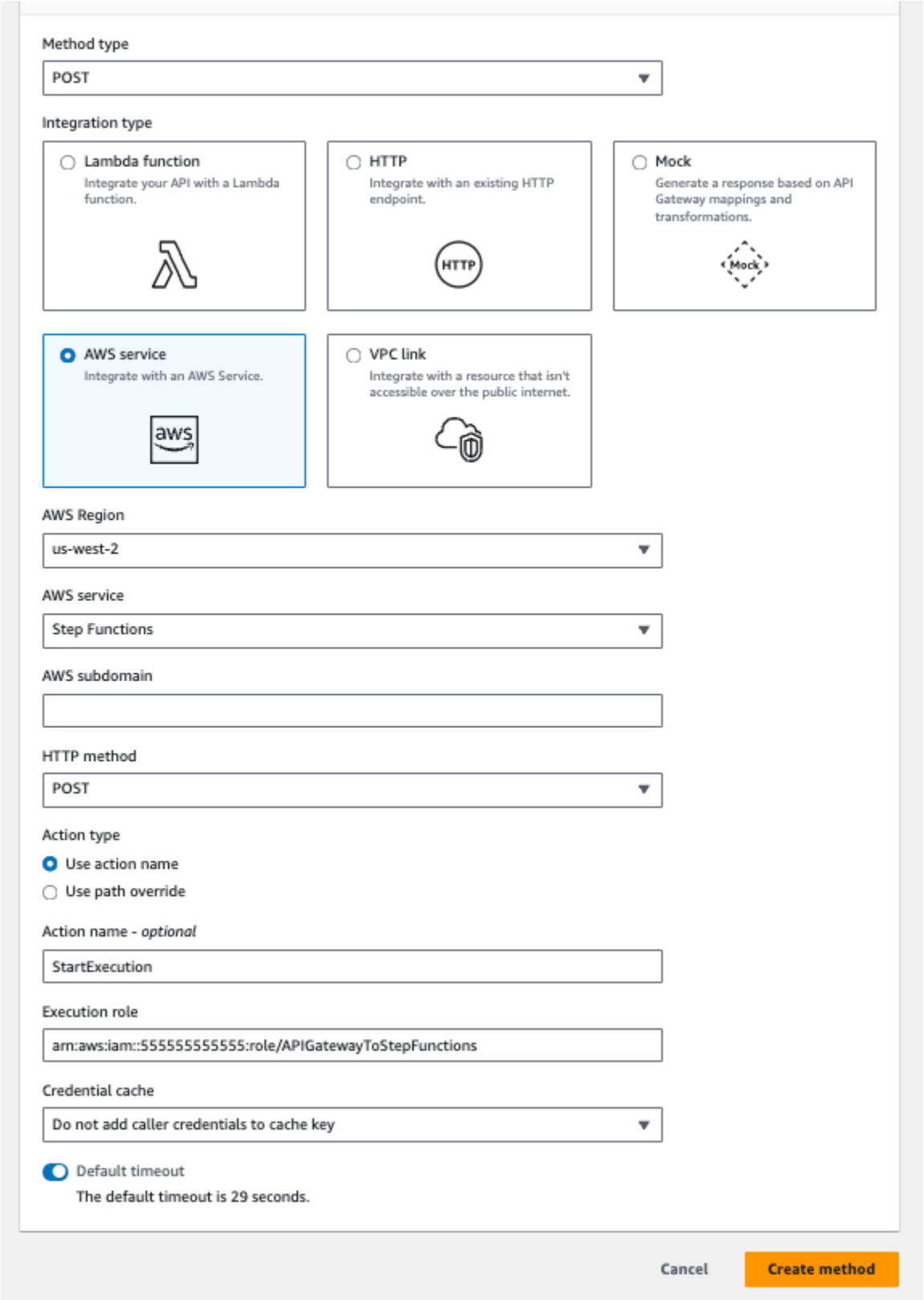
> **Note**
>
> Pour les régions qui prennent actuellement en charge Step Functions, consultez la section Régions prises en charge.

5. Pour Service AWS, choisissez Step Functions dans la liste.
6. Laissez Sous-domaine AWS vide.
7. Pour la méthode HTTP, choisissez POST dans la liste.

> **Note**
>
> Toutes les actions de l'API Step Functions utilisent la `POST` méthode HTTP.

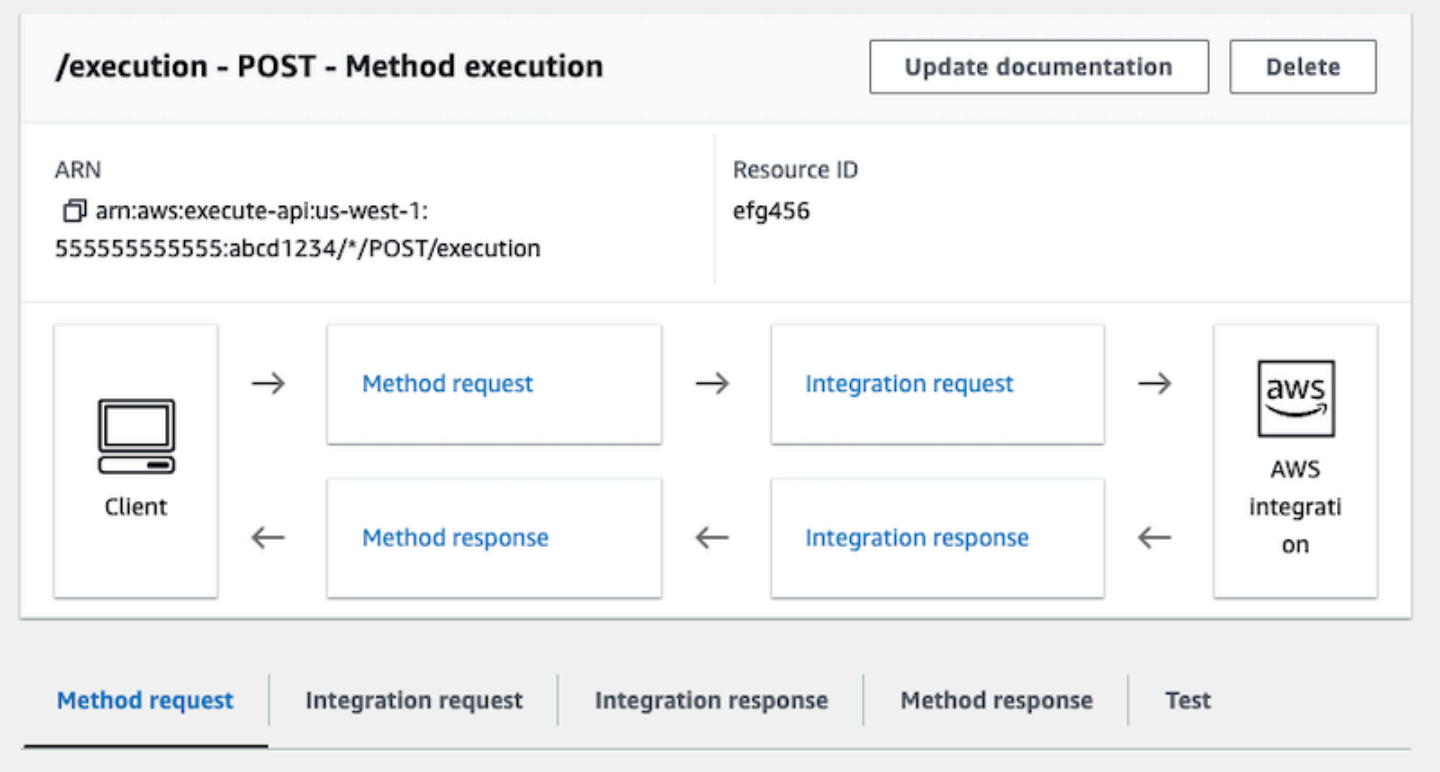
8. Pour Type d’action, sélectionnez Utiliser un nom d’action.
9. Pour Nom de l’action, saisissez **`StartExecution`**.
10. Pour le rôle d'exécution, entrez l'ARN du rôle IAM que vous avez créé précédemment, comme indiqué dans l'exemple suivant.

```
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. Conservez les options par défaut pour le cache des informations d'identification et le délai d'expiration par défaut, puis choisissez Enregistrer.

Le mappage visuel entre API Gateway et Step Functions est affiché sur la page d'exécution de la méthode /execution - POST.



Étape 3 : tester et déployer l'API API Gateway

Une fois que vous avez créé l'API, testez-la et déployez-la.

Pour tester la communication entre API Gateway et Step Functions

1. Sur la page /execution - POST - Method Execution, choisissez l'onglet Test. Vous devrez peut-être choisir la flèche droite pour afficher l'onglet.
2. Dans l'onglet /execution - POST - Method Test, copiez les paramètres de demande suivants dans la section Corps de la requête en utilisant l'ARN d'une machine à états existante (ou créez une nouvelle machine à états utilisant une fonction Lambda), puis choisissez Test.

```
{
   "input": "{}",
   "name": "MyExecution",
```

```
   "stateMachineArn": "arn:aws:states:us-
east-1:123456789012:stateMachine:HelloWorld"
}
```

Pour plus d'informations, consultez la syntaxe des `StartExecution` demandes dans la référence de l'AWS Step Functions API.

> **Note**
>
> Si vous ne souhaitez pas inclure l'ARN de votre machine d'état dans le corps de votre appel API Gateway, vous pouvez configurer un modèle de mappage dans l'onglet Demande d'intégration, comme indiqué dans l'exemple suivant.
>
> ```
> {
>     "input": "$util.escapeJavaScript($input.json('$'))",
>     "stateMachineArn": "$util.escapeJavaScript($stageVariables.arn)"
> }
> ```
>
> Avec cette approche, vous pouvez spécifier les ARN de différentes machines à états en fonction de votre stade de développement (par exemple, `devtest`, et`prod`). Pour plus d'informations sur la spécification de variables d'étape dans un modèle de mappage, consultez `$stageVariables`le guide du développeur d'API Gateway.

3. L'exécution démarre et l'ARN d'exécution ainsi que sa date d'époque sont affichés sous le corps de la réponse.

```
{
   "executionArn": "arn:aws:states:us-
east-1:123456789012:execution:HelloWorld:MyExecution",
   "startDate": 1486768956.878
}
```

> **Note**
>
> Vous pouvez consulter l'exécution en choisissant votre machine d'état sur la console AWS Step Functions.

Pour déployer votre API

1. Sur la page Ressources de l'***StartExecutionAPI***, choisissez Deploy API.
2. Pour Étape, sélectionnez Nouvelle étape.
3. Sous Stage name (Nom de l'étape), entrez **alpha**.
4. (Facultatif) Sous Description, entrez une description.
5. Choisissez Deploy (Déployer).

Pour tester votre déploiement

1. Sur la page Stages de ***StartExecutionl'API***, développez alpha,/, /execution, POST, puis choisissez la méthode POST.
2. Sous Method overrides, choisissez l'icône de copie pour copier l'URL d'appel de votre API. L'URL complète doit ressembler à l'exemple suivant.

```
https://a1b2c3d4e5.execute-api.us-east-1.amazonaws.com/alpha/execution
```

3. À partir de la ligne de commande, exécutez la commande `curl` à l'aide de l'ARN de votre machine d'état, puis appelez l'URL de votre déploiement, comme illustré dans l'exemple suivant :

```
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
```

L'ARN d'exécution et sa date d'époque sont renvoyés, comme illustré dans l'exemple suivant.

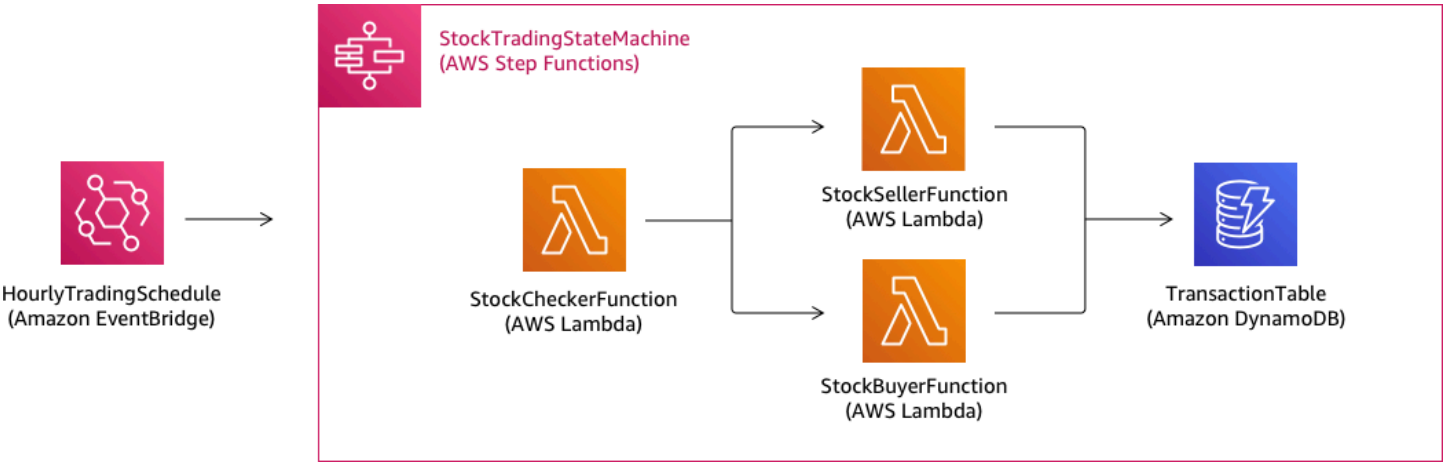
```
{"executionArn":"arn:aws:states:us-
east-1:123456789012:execution:HelloWorld:MyExecution","startDate":1.486772644911E9}
```

> **Note**
>
> Si le message d'erreur « jeton d'authentification manquant » s'affiche, assurez-vous que l'URL d'appel se termine par /execution.

Créez une machine à états Step Functions à l'aideAWS SAM

Dans ce guide, vous téléchargez, créez et déployez un exemple d'application AWS SAM contenant une machine d'état AWS Step Functions. Cette application crée un flux de travail simulé de transactions boursières qui s'exécute selon un calendrier prédéfini (veuillez noter que le calendrier est désactivé par défaut pour éviter d'encourir des frais).

Le diagramme suivant montre les composants de cette application :



Voici un aperçu des commandes que vous exécutez pour créer votre exemple d'application. Pour plus d'informations sur chacune de ces commandes, consultez les sections plus loin dans cette page

```
# 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
```

Prérequis

Ce guide suppose que vous avez complété les étapes de la section Installation de l'interface de ligne de commande AWS SAM pour votre système d'exploitation. Il suppose que ce qui suit a été effectué :

1. A créé unAWScompte.

2. Permissions IAM configurées.
3. Homebrew installé. Remarque : Homebrew est uniquement un prérequis pour Linux et macOS.
4. Installez l'interface de ligne de commande AWS SAM. Remarque : assurez-vous d'avoir la version 0.52.0 ou ultérieure. Vous pouvez vérifier la version dont vous disposez en exécutant la commande `sam --version`.

Étape 1 : Téléchargez un exemple d'application AWS SAM

Commande à exécuter :

```
sam init
```

Suivez les invites à l'écran pour sélectionner les éléments suivants :

1. Modèle : AWSModèles Quick Start
2. Langage : Python, Ruby, NodeJS, Go, Java ou .NET
3. Nom du projet : (nom de votre choix - par défaut : `sam-app`)
4. Application à démarrage rapide :Un flux de travail en plusieurs étapes

Ce que fait AWS SAM :

Cette commande crée un répertoire avec le nom que vous avez fourni pour l'invite 'Nom du projet' (par défaut : `sam-app`). Le contenu spécifique du répertoire dépendra du langage choisi.

Voici le contenu du répertoire lorsque vous choisissez l'un des runtimes Python :

```
### README.md
### functions
#   ### __init__.py
#   ### stock_buyer
#   #   ### __init__.py
#   #   ### app.py
#   #   ### requirements.txt
#   ### stock_checker
#   #   ### __init__.py
#   #   ### app.py
#   #   ### requirements.txt
#   ### stock_seller
```

```
#       ### __init__.py
#       ### app.py
#       ### requirements.txt
### statemachine
#   ### stock_trader.asl.json
### template.yaml
### tests
    ### unit
        ### __init__.py
        ### test_buyer.py
        ### test_checker.py
        ### test_seller.py
```

Il y a deux fichiers particulièrement intéressants que vous pouvez examiner :

- `template.yaml` : contient le modèle AWS SAM qui définit les ressources de votre application AWS.
- `statemachine/stockTrader.asl.json` : contient la définition de la machine d'état de l'application, qui est écrite dans Amazon States Language.

Vous pouvez voir l'entrée suivante dans le fichier `template.yaml`, qui pointe vers le fichier de définition de la machine d'état :

```
    Properties:
      DefinitionUri: statemachine/stock_trader.asl.json
```

Il peut être utile de conserver la définition de la machine à états sous forme de fichier distinct au lieu de l'intégrer dansAWS SAMmodèle. Par exemple, le suivi des modifications apportées à la définition de la machine à états est plus facile si vous n'incluez pas la définition dans le modèle. Vous pouvez utiliser le Workflow Studio pour créer et gérer la définition de la machine à états, et exporter la définition de la console directement vers le fichier de spécification Amazon States Language sans la fusionner dans le modèle.

Pour plus d'informations sur l'application exemple, consultez le fichier `README.md` dans le répertoire du projet.

Étape 2 : Créer votre application

Commande à exécuter :

Changez d'abord dans le répertoire du projet (c'est-à-dire le répertoire où se trouve le fichier `template.yaml` de l'application exemple ; par défaut : `sam-app`), puis exécutez cette commande :

```
sam build
```

Exemple de sortie :

```
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
```

Ce que fait AWS SAM :

LeAWS SAMLa CLI est fournie avec des abstractions pour un certain nombre d'exécutions Lambda, afin de créer vos dépendances. Elle copie tous les artefacts intermédiaires, afin que tout soit prêt à être empaqueté et déployé. La commande `sam build` crée toutes les dépendances de votre application et copie les artefacts de création dans les dossiers sous `.aws-sam/build`.

Étape 3 : Déployez votre application surAWSNuage

Commande à exécuter :

```
sam deploy --guided
```

Suivez les invites à l'écran. Vous pouvez simplement répondre avec `Enter` pour accepter les options par défaut fournies dans l'expérience interactive.

Ce que fait AWS SAM :

Cette commande est déployée votre application dansAWSnuage. Il faut utiliser les artefacts de déploiement que vous créez à l'aide du`sam build`commande, empaqueté et empaqueté dans un compartiment Amazon S3 créé parAWS SAMCLI, et déploie l'application à l'aide deAWS

CloudFormation. Dans la sortie de la commande de déploiement, vous pouvez voir les modifications apportées à votre pile AWS CloudFormation.

Vous pouvez vérifier que l'exemple de machine d'état Step Functions a été déployé avec succès en suivant ces étapes :

1. Connectez-vous auAWS Management Consoleet ouvrez la console Step Functions à l'adressehttps://console.aws.amazon.com/states/.
2. Dans le panneau de navigation de gauche, choisissez Machines d'état.
3. Recherchez et choisissez dans la liste votre nouvelle machine d'état. Il sera nommé StockTradingStateMachine-*<unique-hash>*.
4. Choisissez l'onglet Définition.

Vous devriez maintenant disposer d'une représentation visuelle de votre machine d'état. Vous pouvez vérifier que la représentation visuelle correspond à la définition de la machine d'état trouvée dans le fichier `statemachine/stockTrader.asl.json` du répertoire de votre projet.

Résolution des problèmes

Erreur de l'interface de ligne de commande SAM : « aucune option de ce type : — guidé »

Lors de l'exécution de `sam deploy`, l'erreur suivante s'affiche :

```
Error: no such option: --guided
```

Cela signifie que vous utilisez une ancienne version de l'interface de ligne de commande AWS SAM qui ne prend pas en charge le paramètre `--guided`. Pour résoudre ce problème, vous pouvez soit mettre à jour votre version de l'interface de ligne de commande AWS SAM vers 0.33.0 ou une version ultérieure, soit omettre le paramètre `--guided` de la commande `sam deploy`.

Erreur de l'interface de ligne de commande SAM : « Impossible de créer les ressources gérées : impossible de localiser les informations d'identification »

Lors de l'exécution de `sam deploy`, l'erreur suivante s'affiche :

```
Error: Failed to create managed resources: Unable to locate credentials
```

Cela signifie que vous n'avez pas configuré les informations d'identification AWS pour permettre à la CLI AWS SAM d'effectuer des appels de service AWS. Pour résoudre ce problème, vous devez configurer les informations d'identification AWS. Pour plus d'informations, veuillez consulter la rubriqueConfigurationAWSInformations d'identificationdans leAWS Serverless Application ModelGuide du développeur.

Nettoyage

Si vous n'avez plus besoin duAWSles ressources que vous avez créées en exécutant ce didacticiel, vous pouvez les supprimer en supprimantAWS CloudFormationpile que vous avez déployée.

Pour supprimer la pile AWS CloudFormation créée avec ce didacticiel à l'aide de la AWS Management Console, procédez comme suit :

1. Connectez-vous à AWS Management Console et ouvrez la console AWS CloudFormation à l'adresse https://console.aws.amazon.com/cloudformation.
2. Dans le volet de navigation de gauche, choisissez Stacks (Piles).
3. Dans la liste des piles, choisissez sam-app (ou le nom de la pile que vous avez créée).
4. Choisissez Delete (Supprimer).

L'état de la pile passe alors à DELETE_COMPLETE.

Vous pouvez également supprimer la pile AWS CloudFormation en exécutant la commande de l’AWS CLI suivante :

```
aws cloudformation delete-stack --stack-name sam-app --region region
```

Vérifier la pile supprimée

Pour les deux méthodes de suppression duAWS CloudFormationpile, vous pouvez vérifier qu'elle a été supprimée en vous rendant danshttps://console.aws.amazon.com/cloudformation, en choisissantPilesdans le volet de navigation gauche, et en choisissantSupprimédans le menu déroulant à droite de la zone de texte de recherche. Vous devriez voir le nom de votre pile sam-app (ou le nom de la pile que vous avez créée) dans la liste des piles supprimées.

Création d'une machine à états d'activité à l'aide de Step Functions

Ce didacticiel présente la création d'une machine d'état basée sur les activités à l'aide de Java et d' AWS Step Functions. Les activités vous permettent de contrôler le code de travail qui s'exécute ailleurs depuis votre machine d'état. Pour accéder à une présentation, consultez Activités dans Comment fonctionne Step Functions.

Pour suivre ce didacticiel, vous aurez besoin des éléments suivants :

- Le kit SDK pour Java. L'exemple d'activité présenté dans ce didacticiel est une application Java qui utilise le AWS SDK for Java pour communiquer avec AWS.
- AWS informations d'identification dans l'environnement ou dans le fichier AWS de configuration standard. Pour plus d'informations, consultez la section Configurer vos AWS informations d'identification dans le guide du AWS SDK for Java développeur.

Rubriques

Étape 1 : Créer une activité

Vous devez informer Step Functions de l'activité dont vous souhaitez créer le worker (un programme). Step Functions répond avec un Amazon Resource Name (ARN) qui établit l'identité de l'activité. Utilisez cette identité pour coordonner les informations transmises entre votre machine d'état et l'application de travail.

> **Important**
>
> Assurez-vous que votre tâche d'activité est enregistrée sur le même AWS compte que votre machine d'état.

1. Dans la console Step Functions, dans le volet de navigation de gauche, choisissez Activities.
2. Choisissez Create activity (Créer une activité).
3. Entrez un nom pour l'activité, par exemple*get-greeting*, puis choisissez Créer une activité.
4. Lorsque votre tâche d'activité est créée, notez son ARN, comme illustré dans l'exemple suivant.

```
arn:aws:states:us-east-1:123456789012:activity:get-greeting
```

Étape 2 : Création d'une machine à états

Créez une machine d'état qui détermine à quel moment votre activité est appelée et à quel moment votre application de travail doit effectuer son travail principal, collecter ses résultats et les renvoyer. Pour créer la machine à états, vous allez utiliser Workflow Studio. Éditeur de code

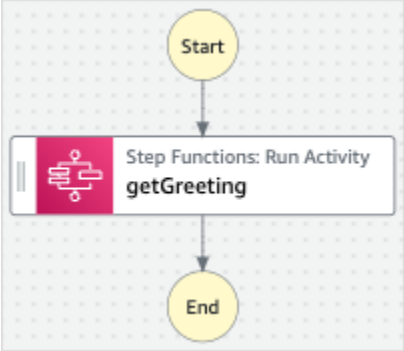
1. Dans la console Step Functions, dans le volet de navigation de gauche, choisissez State machines.
2. Sur la page State machines, choisissez Create state machine.
3. Dans la boîte de dialogue Choisir un modèle, sélectionnez Vide.
4. Choisissez Select (Sélectionner). Cela ouvre Workflow Studio dansMode de conception.
5. Pour ce didacticiel, vous allez écrire la définition Amazon States Language (ASL) de votre machine à états dans l'éditeur de code. Pour ce faire, sélectionnez Code.
6. Supprimez le code standard existant et collez le code suivant. N'oubliez pas de remplacer l'exemple d'ARN dans ce code par l'ARN de la tâche d'activité que vous avez créée précédemment dans le `Resource` champ.

```
{
  "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
    }
  }
```

```
}
```

Il s'agit d'une description de votre machine à états utilisant le Amazon States Language (ASL). Elle décrit un seul état `Task` nommé `getGreeting`. Pour plus d'informations, consultez Structure de la machine d'état.

7. Sur leVolet de visualisation de graphes, assurez-vous que le graphique du flux de travail correspondant à la définition ASL que vous avez ajoutée ressemble au graphique suivant.



8. Spécifiez un nom pour votre machine à états. Pour ce faire, cliquez sur l'icône d'édition à côté du nom de la machine à états par défaut de MyStateMachine. Ensuite, dans Configuration de la machine d'état, spécifiez un nom dans le champ Nom de la machine d'état.

   Pour ce didacticiel, saisissez le nom **`ActivityStateMachine`**.

9. (Facultatif) Dans Configuration de la machine à états, spécifiez d'autres paramètres de flux de travail, tels que le type de machine à états et son rôle d'exécution.

   Pour ce didacticiel, conservez toutes les sélections par défaut dans les paramètres State Machine.

   Si vous avez déjà créé un rôle IAM avec les autorisations appropriées pour votre machine d'état et que vous souhaitez l'utiliser, dans Autorisations, sélectionnez Choisir un rôle existant, puis sélectionnez un rôle dans la liste. Vous pouvez également sélectionner Entrer un ARN de rôle, puis fournir un ARN pour ce rôle IAM.

10. Dans la boîte de dialogue Confirmer la création du rôle, choisissez Confirmer pour continuer.

    Vous pouvez également choisir Afficher les paramètres des rôles pour revenir à la configuration de la machine State.

> **Note**
>
> Si vous supprimez le rôle IAM créé par Step Functions, Step Functions ne pourra pas le recréer ultérieurement. De même, si vous modifiez le rôle (par exemple, en supprimant Step Functions des principes de la politique IAM), Step Functions ne pourra pas restaurer ses paramètres d'origine ultérieurement.

Étape 3 : Implémentation d'une application de travail

Créez une application de travail. Une application de travail est un programme qui a en charge les opérations suivantes :

- Polling Step Functions pour les activités utilisant l'action `GetActivityTask` API.
- Exécution du travail de l'activité à l'aide de votre code (par exemple, la méthode `getGreeting()` dans le code ci-dessous).
- Renvoi des résultats à l'aide des actions d'API `SendTaskSuccess`, `SendTaskFailure` et `SendTaskHeartbeat`.

> **Note**
>
> Pour obtenir un exemple plus complet d'un travail d'activité, consultez Exemple d'activité de travail en Ruby. Cet exemple fournit une implémentation basée sur les bonnes pratiques, qui peut être utilisée comme référence pour l'application de travail de votre activité. Le code implémente un modèle consommateur-producteur avec un nombre configurable de threads pour les observateurs et les travaux d'activité.

Pour implémenter l'application de travail

1. Créez un fichier nommé `GreeterActivities.java`.
2. Ajoutez-lui le code suivant.

```
import com.amazonaws.ClientConfiguration;
import com.amazonaws.auth.EnvironmentVariableCredentialsProvider;
import com.amazonaws.regions.Regions;
```

```
import com.amazonaws.services.stepfunctions.AWSStepFunctions;
import com.amazonaws.services.stepfunctions.AWSStepFunctionsClientBuilder;
import com.amazonaws.services.stepfunctions.model.GetActivityTaskRequest;
import com.amazonaws.services.stepfunctions.model.GetActivityTaskResult;
import com.amazonaws.services.stepfunctions.model.SendTaskFailureRequest;
import com.amazonaws.services.stepfunctions.model.SendTaskSuccessRequest;
import com.amazonaws.util.json.Jackson;
import com.fasterxml.jackson.databind.JsonNode;
import java.util.concurrent.TimeUnit;


public class GreeterActivities {

    public String getGreeting(String who) throws Exception {
        return "{\"Hello\": \"" + who + "\"}";
    }

    public static void main(final String[] args) throws Exception {
        GreeterActivities greeterActivities = new GreeterActivities();
        ClientConfiguration clientConfiguration = new ClientConfiguration();
        clientConfiguration.setSocketTimeout((int)TimeUnit.SECONDS.toMillis(70));

        AWSStepFunctions client = AWSStepFunctionsClientBuilder.standard()
                .withRegion(Regions.US_EAST_1)
                .withCredentials(new EnvironmentVariableCredentialsProvider())
                .withClientConfiguration(clientConfiguration)
                .build();

        while (true) {
            GetActivityTaskResult getActivityTaskResult =
                    client.getActivityTask(
                            new
 GetActivityTaskRequest().withActivityArn(ACTIVITY_ARN));

            if (getActivityTaskResult.getTaskToken() != null) {
                try {
                    JsonNode json =
 Jackson.jsonNodeOf(getActivityTaskResult.getInput());
                    String greetingResult =

 greeterActivities.getGreeting(json.get("who").textValue());
                    client.sendTaskSuccess(
                            new SendTaskSuccessRequest().withOutput(
```

```
 greetingResult).withTaskToken(getActivityTaskResult.getTaskToken()));
                } catch (Exception e) {
                    client.sendTaskFailure(new
 SendTaskFailureRequest().withTaskToken(
                            getActivityTaskResult.getTaskToken()));
                }
            } else {
                Thread.sleep(1000);
            }
        }
    }
}
```

> **Note**
>
> La classe `EnvironmentVariableCredentialsProvider` de cet exemple suivant suppose que les variables d'environnement `AWS_ACCESS_KEY_ID` (ou `AWS_ACCESS_KEY`) et `AWS_SECRET_KEY` (ou `AWS_SECRET_ACCESS_KEY`) sont définies. Pour plus d'informations sur la fourniture des informations d'identification requises à l'usine, reportez-vous AWSCredentialsProviderà la section Référence des AWS SDK for Java API et à la section Configurer les AWS informations d'identification et la région pour le développement dans le guide du AWS SDK for Java développeur. Par défaut, le AWS SDK attendra jusqu'à 50 secondes pour recevoir les données du serveur pour toute opération. `GetActivityTask` est une opération d'attente active de longue durée qui attendra la prochaine tâche disponible pendant 60 secondes au maximum. Pour éviter de recevoir une `SocketTimeoutException` erreur, réglez SocketTimeout ce paramètre sur 70 secondes.

3. Dans la liste des paramètres du constructeur `GetActivityTaskRequest().withActivityArn()`, remplacez la valeur `ACTIVITY_ARN` par l'ARN de la tâche d'activité que vous avez créée précédemment.

Étape 4 : Exécutez la machine d'état

Lorsque vous lancez l'exécution de la machine à états, votre assistant interroge Step Functions pour connaître les activités, effectue son travail (en utilisant les entrées que vous fournissez) et renvoie ses résultats.

1. Sur la ***ActivityStateMachine***page, choisissez Démarrer l'exécution.

   La boîte de dialogue Démarrer l'exécution s'affiche.

2. Dans la boîte de dialogue Démarrer l'exécution, procédez comme suit :

   a. (Facultatif) Pour identifier votre exécution, vous pouvez lui donner un nom dans le champ Nom. Par défaut, Step Functions génère automatiquement un nom d'exécution unique.

   > **Note**
   >
   > Step Functions vous permet de créer des noms pour les machines d'état, les exécutions et les activités, ainsi que des étiquettes contenant des caractères non ASCII. Ces noms non ASCII ne fonctionnent pas avec Amazon. CloudWatch Pour être sûr de pouvoir suivre CloudWatch les métriques, choisissez un nom qui utilise uniquement des caractères ASCII.

   b. Dans le champ Entrée, entrez l'entrée JSON suivante pour exécuter votre flux de travail.

   ```
   {
     "who": "AWS Step Functions"
   }
   ```

   c. Choisissez Start execution (Démarrer l'exécution).

   d. La console Step Functions vous dirige vers une page intitulée avec votre ID d'exécution. Cette page est connue sous le nom de page Détails de l'exécution. Sur cette page, vous pouvez consulter les résultats de l'exécution au fur et à mesure que l'exécution progresse ou une fois celle-ci terminée.

      Pour consulter les résultats de l'exécution, choisissez des états individuels dans la vue graphique, puis choisissez les onglets individuels du Détails de l'étape volet pour afficher les détails de chaque état, y compris les entrées, les sorties et la définition respectivement. Pour plus de détails sur les informations d'exécution que vous pouvez consulter sur la page Détails de l'exécution, voirPage de détails d'exécution — Vue d'ensemble de l'interface.

Étape 5 : Exécuter et arrêter l'application de travail

Pour que l'application de travail interroge les activités de votre machine d'état, vous devez exécuter l'application de travail.

1. Sur la ligne de commande, accédez au répertoire dans lequel vous avez créé `GreeterActivities.java`.
2. Pour utiliser le AWS SDK, ajoutez le chemin complet des `third-party` répertoires `lib` et aux dépendances de votre fichier de compilation et à votre Java`CLASSPATH`. Pour plus d'informations, consultez la section Téléchargement et extraction du SDK dans le guide du AWS SDK for Java développeur.
3. Compilez le fichier.

```
$ javac GreeterActivities.java
```

4. Exécutez le fichier .

```
$ java GreeterActivities
```

5. Sur la console Step Functions, accédez à la page Détails de l'exécution.
6. Lorsque l'exécution est terminée, examinez les résultats de votre exécution.
7. Arrêtez l'application de travail.

Itérer une boucle avec Lambda

Dans ce didacticiel, vous implémentez un modèle de conception qui utilise une machine d'état et une fonction AWS Lambda pour itérer une boucle un certain nombre de fois.

Utilisez ce modèle de conception chaque fois que vous avez besoin d'effectuer le suivi du nombre de boucles dans une machine d'état. Cette implémentation peut vous aider à diviser des tâches volumineuses ou des exécutions de longue durée en fragments plus petits, ou d'arrêter une exécution après un nombre spécifique d'événements. Vous pouvez utiliser une implémentation similaire pour arrêter et redémarrer périodiquement une exécution de longue durée afin d'éviter de dépasser les quotas de service pour AWS Step Functions AWS Lambda, ou pour d'autres AWS services.

Avant de commencer, suivez le Création d'une machine d'état Step Functions utilisant Lambda didacticiel pour vous assurer que vous êtes familiarisé avec l'utilisation conjointe de Lambda et de Step Functions.

Étape 1 : créer une fonction Lambda pour itérer un décompte

En utilisant une fonction Lambda, vous pouvez suivre le nombre d'itérations d'une boucle dans votre machine à états. La fonction Lambda suivante reçoit les valeurs d'entrée pour `countindex`, et.

`step` Puis, elle retourne ces valeurs avec un `index` mis à jour et une valeur booléenne nommée `continue`. La fonction Lambda est définie `continue` sur `true` si le `index` est inférieur à. `count`

Votre machine d'état implémente alors un état `Choice` qui exécute une certaine logique d'application si `continue` a la valeur `true`, ou s'arrête si la valeur est `false`.

Pour créer la fonction Lambda

1. Connectez-vous à la console Lambda, puis choisissez Create function.
2. Sur la page Create function, sélectionnez Author from scratch.
3. Dans la section Informations de base, configurez votre fonction Lambda comme suit :
   a. Sous Nom de la fonction, saisissez `Iterator`.
   b. Dans le champ Runtime, sélectionnez Node.js.
   c. Dans Modifier le rôle d'exécution par défaut, choisissez Créer un nouveau rôle avec des autorisations Lambda de base.
   d. Choisissez Créer une fonction.
4. Copiez le code suivant pour la fonction Lambda dans le code source.

```
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
  })
}
```

   Le code accepte les valeurs d'entrée pour `count`, `index` et `step`. Il incrémente `index` de la valeur de `step` et retourne ces valeurs, et la valeur booléenne `continue`. La valeur de `continue` est `true` si `index` est inférieur à `count`.
5. Choisissez Deploy (Déployer).

Étape 2 : tester la fonction Lambda

Exécutez votre fonction Lambda avec des valeurs numériques pour la voir fonctionner. Vous pouvez fournir des valeurs d'entrée pour votre fonction Lambda qui imitent une itération.

Pour tester votre fonction Lambda

1. Sélectionnez Tester).
2. Dans la boîte de dialogue Configurer l'événement de test, entrez `TestIterator` le nom de l'événement.
3. Remplacez les données de l'exemple par les suivantes.

   ```
   {
     "Comment": "Test my Iterator function",
     "iterator": {
       "count": 10,
       "index": 5,
       "step": 1
     }
   }
   ```

   Ces valeurs simulent ce qui provient de votre machine d'état au cours d'une itération. La fonction Lambda incrémente l'indice et renvoie `continue` lorsque `true` l'indice est inférieur à. `count` Pour ce test, l'index a déjà été incrémenté à `5`. Le test sera incrémenté `index` `6` et réglé sur`continue`. `true`
4. Choisissez Créer.
5. Choisissez Test pour tester votre fonction Lambda.

   Les résultats du test sont affichés dans l'onglet Résultats de l'exécution.
6. Cliquez sur l'onglet Résultats de l'exécution pour voir le résultat.

   ```
   {
     "index": 6,
     "step": 1,
     "count": 10,
     "continue": true
   }
   ```

> **Note**
>
> Si vous réglez `index` 9 et testez à nouveau, les `index` incréments de `10` et `continue` serontfalse.

Étape 3 : Création d'une machine d'état

> **Avant de quitter la console Lambda...**
>
> Copiez l'ARN de la fonction Lambda. Collez-le dans une note. Vous en aurez besoin à l'étape suivante.

Ensuite, vous allez créer une machine à états avec les états suivants :

- `ConfigureCount`— Définit les valeurs par défaut pour `count`index, et`step`.
- `Iterator`— Fait référence à la fonction Lambda que vous avez créée précédemment, en transmettant les valeurs configurées dans. `ConfigureCount`
- `IsCountReached`— Un état de choix qui continue la boucle ou passe à `Done` l'état, en fonction de la valeur renvoyée par votre `Iterator` fonction.
- `ExampleWork`— Un talon pour les travaux à effectuer. Dans cet exemple, le flux de travail possède un `Pass` état, mais dans une solution réelle, vous utiliseriez probablement un`Task`.
- `Done`— État final de votre flux de travail.

Pour créer la machine à états dans la console :

1. Ouvrez la console Step Functions, puis choisissez Create a state machine.

   > **Important**
   >
   > Votre machine d'état doit se trouver dans le même AWS compte et dans la même région que votre fonction Lambda.

2. Sélectionnez le modèle vide.

3. Dans le volet Code, collez le code JSON suivant qui définit la machine à états.

   Pour plus d'informations sur le langage Amazon States, consultez 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. Remplacez le `Iterator Resource` champ par l'ARN de votre fonction `Iterator` Lambda que vous avez créée précédemment.
5. Sélectionnez Config, puis entrez un nom pour votre machine d'état, tel que*`IterateCount`*.

> **Note**
>
> Les noms des machines d'état, des exécutions et des tâches d'activité ne doivent pas dépasser 80 caractères. Ces noms doivent être uniques pour votre compte et votre AWS région, et ne doivent contenir aucun des éléments suivants :
>
> - Espace blanc
> - Caractères génériques () ? *
> - Caractères entre crochets (< > { } [ ])
> - Caractères spéciaux (" # % \ ^ | ~ ` $ & , ; : /)
> - Caractères de contrôle (\\u0000- \\u001f ou \\u007f -\\u009f).
>
> Si votre machine à états est de type Express, vous pouvez attribuer le même nom à plusieurs exécutions de la machine à états. Step Functions génère un ARN d'exécution unique pour chaque exécution automatique d'Express State, même si plusieurs exécutions portent le même nom.
> Step Functions vous permet de créer des noms pour les machines d'état, les exécutions et les activités, ainsi que des étiquettes contenant des caractères non ASCII. Ces noms non ASCII ne fonctionnent pas avec Amazon. CloudWatch Pour être sûr de pouvoir suivre CloudWatch les métriques, choisissez un nom qui utilise uniquement des caractères ASCII.

6. Pour Type, acceptez la valeur par défaut Standard. Pour Autorisations, choisissez Créer un nouveau rôle.

7. Choisissez Créer, puis Confirmez les créations de rôles.

Étape 4 : Démarrage d'une nouvelle exécution

Une fois que vous avez créé votre machine d'état, vous pouvez lancer une exécution.

1. Sur la IterateCountpage, choisissez Démarrer l'exécution.
2. (Facultatif) Pour identifier votre exécution, vous pouvez lui donner un nom dans le champ Nom. Par défaut, Step Functions génère automatiquement un nom d'exécution unique.

   > **Note**
   >
   > Step Functions vous permet de créer des noms pour les machines d'état, les exécutions et les activités, ainsi que des étiquettes contenant des caractères non ASCII. Ces noms non ASCII ne fonctionnent pas avec Amazon. CloudWatch Pour être sûr de pouvoir suivre CloudWatch les métriques, choisissez un nom qui utilise uniquement des caractères ASCII.

3. Choisissez Démarrer une exécution.

   Une nouvelle exécution de votre machine d'état commence, montrant votre exécution en cours.
   Vue graphique de la machine d'état, montrant l'état de l'itérateur en bleu pour indiquer l'état en cours.

   L'exécution s'incrémente par étapes, en suivant le nombre à l'aide de votre fonction Lambda. Sur chaque itération, elle exécute l'exemple de travail référencé dans l'état `ExampleWork` de votre machine d'état.

   Une fois que le compte atteint le nombre configuré dans l'état `ConfigureCount` de votre machine d'état, l'exécution quitte l'itération et prend fin.
   Vue graphique de la machine à états, montrant l'état de l'itérateur et l'état terminé en vert pour indiquer que les deux ont réussi.

Poursuite des exécutions de flux de travail de longue durée en tant que nouvelle exécution

AWS Step Functions est conçu pour exécuter les flux de travail qui ont une durée et un nombre d'étapes finis. Les exécutions ont une durée maximale d'un an et un maximum de 25 000 événements (voir Quotas).

Pour les exécutions de longue durée, afin d'éviter d'atteindre le quota strict de 25 000 entrées dans l'historique des événements d'exécution, nous vous recommandons de démarrer une nouvelle exécution de flux de travail directement à partir de l'`Task`état d'une machine d'état. Cela vous permet de diviser vos flux de travail en machines à états plus petites et de poursuivre votre travail en cours dans le cadre d'une nouvelle exécution. Pour démarrer ces exécutions de flux de travail, appelez l'action d'`StartExecution`API depuis votre `Task` état et transmettez les paramètres nécessaires.

Vous pouvez également implémenter un modèle qui utilise une fonction Lambda pour démarrer une nouvelle exécution de votre machine d'état afin de répartir le travail en cours entre plusieurs exécutions de flux de travail.

Ce didacticiel présente les deux approches permettant de poursuivre l'exécution du flux de travail sans dépasser les quotas de service.

Rubriques

- Utiliser une action de l'API Step Functions pour poursuivre une nouvelle exécution (recommandé)
- Utilisation d'une fonction Lambda pour poursuivre une nouvelle exécution

Utiliser une action de l'API Step Functions pour poursuivre une nouvelle exécution (recommandé)

Step Functions peut lancer des exécutions de flux de travail en appelant sa propre API en tant que service intégré. Nous vous recommandons d'utiliser cette approche pour éviter de dépasser les quotas de service lors d'exécutions de longue durée.

Étape 1 : Création d'une machine à états de longue durée

Créez une machine à états de longue durée que vous souhaitez démarrer à partir de l'`Task`état d'une machine à états différente. Pour ce didacticiel, utilisez la machine à états qui utilise une fonction Lambda.

> **Note**
>
> Assurez-vous de copier le nom et le nom de ressource Amazon de cette machine d'état dans un fichier texte pour une utilisation ultérieure.

Étape 2 : créer une machine à états pour appeler l'action de l'API Step Functions

Pour démarrer les exécutions de flux de travail à partir d'un **`Task`** état

1. Ouvrez la console Step Functions et choisissez Create state machine.
2. Dans la boîte de dialogue Choisir un modèle, sélectionnez Vide.
3. Choisissez Select (Sélectionner). Cela ouvre Workflow Studio dansMode de conception.
4. Dans l'onglet Actions, faites glisser l'action StartExecutionAPI et déposez-la sur l'état vide intitulé Drag first state here.
5. Choisissez l'StartExecutionétat et effectuez les opérations suivantes dans l'onglet Configuration de Mode de conception :

   a. Renommez l'état en. **`Start nested execution`**

   b. Pour le type d'intégration, choisissez AWS SDK - nouveau dans la liste déroulante.

   c. Dans Paramètres de l'API, procédez comme suit :

      i. Pour remplacer `StateMachineArn` l'exemple de nom de ressource Amazon par l'ARN de votre machine à états. Par exemple, entrez l'ARN de la machine à états qui utilise Lambda.

      ii. Pour `Input` node, remplacez le texte de l'espace réservé existant par la valeur suivante :

```
"Comment": "Starting workflow execution using a Step Functions API action"
```

      iii. Assurez-vous que vos entrées dans les paramètres de l'API ressemblent à ce qui suit :

```
{
  "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. (Facultatif) Choisissez Définition dans le Inspector panneau pour afficher la définition générée automatiquement Amazon States Language (ASL) de votre flux de travail.

   > **Tip**
   >
   > Vous pouvez également consulter la définition ASL dans Workflow Studio. Éditeur de code Dans l'éditeur de code, vous pouvez également modifier la définition ASL de votre flux de travail.

7. Spécifiez un nom pour votre machine à états. Pour ce faire, cliquez sur l'icône d'édition à côté du nom de la machine à états par défaut de MyStateMachine. Ensuite, dans Configuration de la machine d'état, spécifiez un nom dans le champ Nom de la machine d'état.

   Pour ce didacticiel, saisissez le nom **`ParentStateMachine`**.

8. (Facultatif) Dans Configuration de la machine à états, spécifiez d'autres paramètres de flux de travail, tels que le type de machine à états et son rôle d'exécution.

   Pour ce didacticiel, conservez toutes les sélections par défaut dans les paramètres State Machine.

   Si vous avez déjà créé un rôle IAM avec les autorisations appropriées pour votre machine d'état et que vous souhaitez l'utiliser, dans Autorisations, sélectionnez Choisir un rôle existant, puis sélectionnez un rôle dans la liste. Vous pouvez également sélectionner Entrer un ARN de rôle, puis fournir un ARN pour ce rôle IAM.

9. Dans la boîte de dialogue Confirmer la création du rôle, choisissez Confirmer pour continuer.

   Vous pouvez également choisir Afficher les paramètres des rôles pour revenir à la configuration de la machine State.

   > **Note**
   >
   > Si vous supprimez le rôle IAM créé par Step Functions, Step Functions ne pourra pas le recréer ultérieurement. De même, si vous modifiez le rôle (par exemple, en supprimant Step Functions des principes de la politique IAM), Step Functions ne pourra pas restaurer ses paramètres d'origine ultérieurement.

Étape 3 : mettre à jour la politique IAM

Pour vous assurer que votre machine à états dispose des autorisations nécessaires pour démarrer l'exécution de la machine à états qui utilise une fonction Lambda, vous devez associer une politique en ligne au rôle IAM de votre machine à états. Pour plus d'informations, consultez la section Intégration de politiques intégrées dans le guide de l'utilisateur IAM.

1. Sur la ParentStateMachinepage, choisissez l'ARN du rôle IAM pour accéder à la page Rôles IAM de votre machine à états.
2. Attribuez une autorisation appropriée au rôle IAM du ParentStateMachinepour qu'il puisse démarrer l'exécution d'une autre machine à états. Pour attribuer l'autorisation, procédez comme suit :

   a. Sur la page Rôles IAM, choisissez Ajouter des autorisations, puis choisissez Créer une politique intégrée.

   b. Sur la page Créer une politique, choisissez l'onglet JSON.

   c. Remplacez le texte existant par la politique suivante.

```
{
  "Version": "2012-10-17",
  "Statement": [
      {
          "Effect": "Allow",
          "Action": [
              "states:StartExecution"
          ],
          "Resource": [
              "arn:aws:states:us-
east-2:123456789012:stateMachine:LambdaStateMachine"
          ]
      }
  ]
}
```

   d. Choisissez Examiner une politique.

   e. Spécifiez le nom de la stratégie, puis choisissez Create policy.

Étape 4 : Exécutez la machine d'état

Les exécutions par State Machine sont des instances dans lesquelles vous exécutez votre flux de travail pour effectuer des tâches.

1. Sur la ParentStateMachinepage, choisissez Démarrer l'exécution.

   La boîte de dialogue Démarrer l'exécution s'affiche.

2. Dans la boîte de dialogue Démarrer l'exécution, procédez comme suit :

   a. (Facultatif) Pour identifier votre exécution, vous pouvez lui donner un nom dans le champ Nom. Par défaut, Step Functions génère automatiquement un nom d'exécution unique.

   > **Note**
   >
   > Step Functions vous permet de créer des noms pour les machines d'état, les exécutions et les activités, ainsi que des étiquettes contenant des caractères non ASCII. Ces noms non ASCII ne fonctionnent pas avec Amazon. CloudWatch Pour être sûr de pouvoir suivre CloudWatch les métriques, choisissez un nom qui utilise uniquement des caractères ASCII.

   b. (Facultatif) Dans la zone de saisie, entrez les valeurs d'entrée au format JSON pour exécuter votre flux de travail.

   c. Choisissez Start execution (Démarrer l'exécution).

   d. La console Step Functions vous dirige vers une page intitulée avec votre ID d'exécution. Cette page est connue sous le nom de page Détails de l'exécution. Sur cette page, vous pouvez consulter les résultats de l'exécution au fur et à mesure que l'exécution progresse ou une fois celle-ci terminée.

      Pour consulter les résultats de l'exécution, choisissez des états individuels dans la vue graphique, puis choisissez les onglets individuels du Détails de l'étape volet pour afficher les détails de chaque état, y compris les entrées, les sorties et la définition respectivement. Pour plus de détails sur les informations d'exécution que vous pouvez consulter sur la page Détails de l'exécution, voirPage de détails d'exécution — Vue d'ensemble de l'interface.

3. Ouvrez la LambdaStateMachinepage et remarquez une nouvelle exécution déclenchée par le ParentStateMachine.

Utilisation d'une fonction Lambda pour poursuivre une nouvelle exécution

Vous pouvez créer une machine à états qui utilise une fonction Lambda pour démarrer une nouvelle exécution avant la fin de l'exécution en cours. L'utilisation de cette approche pour poursuivre votre travail en cours dans le cadre d'une nouvelle exécution vous permet de disposer d'une machine à états capable de diviser les tâches volumineuses en flux de travail plus petits, ou d'avoir une machine à états qui fonctionne indéfiniment.

Ce didacticiel repose sur le concept d'utilisation d'une fonction Lambda externe pour modifier votre flux de travail, qui a été démontré dans le Itérer une boucle avec Lambda didacticiel. Vous utilisez la même fonction Lambda (`Iterator`) pour itérer une boucle un certain nombre de fois. En outre, vous créez une autre fonction Lambda pour démarrer une nouvelle exécution de votre flux de travail et pour décrémenter un nombre à chaque fois qu'il commence une nouvelle exécution. En définissant le nombre d'exécutions dans l'entrée, cette machine d'état termine et redémarre une exécution un nombre de fois spécifié.

La machine d'état que vous allez créer implémente les états suivants.

État	Objectif
`ConfigureCount`	`Pass`État qui configure les `step` valeurs `countindex`, et utilisées par la fonction `Iterator` Lambda pour effectuer des itérations de travail.
`Iterator`	`Task`État qui fait référence à la fonction `Iterator` Lambda.
`IsCountReached`	`Choice`État qui utilise une valeur booléenne issue de la `Iterator` fonction pour décider si la machine à états doit continuer le travail d'exemple ou passer à l'`ShouldRestart` état.
`ExampleWork`	`Pass`État qui représente l'`Task`état qui exécuterait le travail dans une implémentation réelle.
`ShouldRestart`	`Choice`État qui utilise la `executionCount` valeur pour décider s'il doit mettre fin à une exécution et en démarrer une autre, ou simplement se terminer.
`Restart`	`Task`État qui utilise une fonction Lambda pour démarrer une nouvelle exécution de votre machine à états. Comme la fonction

État	Objectif
	`Iterator`, cette fonction décrémente également un nombre. L'`Restart`état transmet la valeur décrémentée du décompte à l'entrée de la nouvelle exécution.

Prérequis

Avant de commencer, suivez le Création d'une machine d'état Step Functions utilisant Lambda didacticiel pour vous assurer que vous êtes familiarisé avec l'utilisation conjointe de Lambda et de Step Functions.

Rubriques

- Étape 1 : créer une fonction Lambda pour itérer un décompte
- Étape 2 : Création d'une fonction Restart Lambda pour démarrer une nouvelle exécution de Step Functions
- Étape 3 : créer une machine à états
- Étape 4 : Mettre à jour la politique IAM
- Étape 5 : Exécutez la machine d'état

Étape 1 : créer une fonction Lambda pour itérer un décompte

> **Note**
>
> Si vous avez terminé le Itérer une boucle avec Lambda didacticiel, vous pouvez ignorer cette étape et utiliser cette fonction Lambda.

Cette section et le Itérer une boucle avec Lambda didacticiel montrent comment utiliser une fonction Lambda pour suivre un décompte, par exemple le nombre d'itérations d'une boucle dans votre machine à états.

La fonction Lambda suivante reçoit les valeurs d'entrée pour `count`, `index`, et. `step` Elle renvoie ces valeurs avec un `index` mis à jour et une valeur booléenne nommée `continue`. La fonction Lambda est définie `continue` sur `true` si le `index` est inférieur à. `count`

Votre machine d'état implémente alors un état `Choice` qui exécute une certaine logique d'application si `continue` a la valeur `true`, ou de déplace à `ShouldRestart` si `continue` a la valeur `false`.

Création de la fonction Iterate Lambda

1. Ouvrez la console Lambda, puis choisissez Create function (Créer une fonction.
2. Sur la page Create function, sélectionnez Author from scratch.
3. Dans la section Informations de base, configurez votre fonction Lambda comme suit :

   a. Sous Nom de la fonction, saisissez `Iterator`.

   b. Pour Exécution, choisissez Node.js 16.x.

   c. Conservez toutes les sélections par défaut sur la page, puis choisissez Créer une fonction.

   Lorsque votre fonction Lambda est créée, notez son Amazon Resource Name (ARN) dans le coin supérieur droit de la page, par exemple :

   ```
   arn:aws:lambda:us-east-1:123456789012:function:Iterator
   ```

4. Copiez le code suivant pour la fonction Lambda dans la section Source du code de la page ***`Iterator`*** de la console Lambda.

   ```
   exports.handler = function iterator (event, context, callback) {
     let index = event.iterator.index;
     let step = event.iterator.step;
     let count = event.iterator.count;

     index = index + step;

     callback(null, {
       index,
       step,
       count,
       continue: index < count
     })
   }
   ```

   Le code accepte les valeurs d'entrée pour `count`, `index` et `step`. Il incrémente `index` de la valeur de `step` et retourne ces valeurs, et la valeur booléenne `continue`. La valeur de `continue` est `true` si `index` est inférieur à `count`.

5. Choisissez Deploy pour déployer le code.

Tester la fonction Iterate Lambda

Pour vérifier le fonctionnement de votre fonction `Iterate`, exécutez-la avec des valeurs numériques. Vous pouvez fournir des valeurs d'entrée pour votre fonction Lambda qui imitent une itération pour voir quelle sortie vous obtenez avec des valeurs d'entrée spécifiques.

Pour tester votre fonction Lambda

1. Dans la boîte de dialogue Configure test event, choisissez Create new test event, puis entrez `TestIterator` pour Event name.
2. Remplacez les données de l'exemple par les suivantes.

```
{
  "Comment": "Test my Iterator function",
  "iterator": {
    "count": 10,
    "index": 5,
    "step": 1
  }
}
```

   Ces valeurs simulent ce qui provient de votre machine d'état au cours d'une itération. La fonction Lambda incrémente l'index et renvoie sous la forme. `continue` `true` Quand l'index n'est pas inférieur à `count`, la fonction renvoie `continue` avec la valeur `false`. Pour ce test, l'index a déjà été incrémenté à 5. Les résultats doivent incrémenter `index` à 6 et définir `continue` avec la valeur `true`.
3. Choisissez Créer.
4. Sur la page ***`Iterator`*** de votre console Lambda, TestIteratorassurez-vous qu'il est répertorié, puis choisissez Test.

   Les résultats du test sont affichés en haut de la page. Choisissez Details et vérifiez le résultat.

```
{
  "index": 6,
  "step": 1,
  "count": 10,
  "continue": true
}
```

> **Note**
>
> Si vous définissez `index` avec la valeur 9 pour ce test, `index` s'incrémente à `10` et `continue` a la valeur `false`.

Étape 2 : Création d'une fonction Restart Lambda pour démarrer une nouvelle exécution de Step Functions

1. Ouvrez la console Lambda, puis choisissez Create function (Créer une fonction.
2. Sur la page Create function, sélectionnez Author from scratch.
3. Dans la section Informations de base, configurez votre fonction Lambda comme suit :
   a. Sous Nom de la fonction, saisissez `Restart`.
   b. Pour Exécution, choisissez Node.js 16.x.
4. Conservez toutes les sélections par défaut sur la page, puis choisissez Créer une fonction.

   Lorsque votre fonction Lambda est créée, notez son Amazon Resource Name (ARN) dans le coin supérieur droit de la page, par exemple :

   ```
   arn:aws:lambda:us-east-1:123456789012:function:Iterator
   ```

5. Copiez le code suivant pour la fonction Lambda dans la section Source du code de la page ***Redémarrer*** de la console Lambda.

   Le code suivant décrémente le compte du nombre d'exécutions et démarre une nouvelle exécution de votre machine d'état, valeur décrémentée incluse.

   ```
   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. Choisissez Deploy pour déployer le code.

Étape 3 : créer une machine à états

Maintenant que vous avez créé vos deux fonctions Lambda, créez une machine à états. Dans cette machine d'état, les états `ShouldRestart` et `Restart` se rapportent à la façon dont vous scindez votre tâche en plusieurs exécutions.

Example ShouldRestart État du choix

L'extrait suivant montre l'`ShouldRestart`[`Choice`]état. Cet état détermine si vous devez ou non redémarrer l'exécution.

```
"ShouldRestart": {
"Type": "Choice",
"Choices": [
  {
    "Variable": "$.restart.executionCount",
    "NumericGreaterThan": 1,
    "Next": "Restart"
  }
],
```

La valeur `$.restart.executionCount` est incluse dans les données d'entrée de l'exécution initiale. Elle est décrémenté d'une unité chaque fois que la fonction `Restart` est appelée, puis placée dans l'entrée pour chaque exécution suivante.

Example Redémarrer l'état de la tâche

L'extrait suivant montre l'`RestartTask`état. Cet état utilise la fonction Lambda que vous avez créée précédemment pour redémarrer l'exécution et décrémenter le nombre d'exécutions afin de suivre le nombre d'exécutions restantes à démarrer.

```
"Restart": {
  "Type": "Task",
  "Resource": "arn:aws:lambda:us-east-1:123456789012:function:Restart",
  "Next": "Done"
},
```

Pour créer la machine d'état

1. Ouvrez la console Step Functions et choisissez Create state machine.

   > **Important**
   >
   > Assurez-vous que votre machine à états se trouve sous le même AWS compte et dans la même région que les fonctions Lambda que vous avez créées précédemment aux étapes 1 et 2.

2. Dans la boîte de dialogue Choisir un modèle, sélectionnez Vide.
3. Choisissez Select (Sélectionner). Cela ouvre Workflow Studio dansMode de conception.
4. Pour ce didacticiel, vous allez écrire la définition Amazon States Language (ASL) de votre machine à états dans leÉditeur de code. Pour ce faire, sélectionnez Code.
5. Supprimez le code standard existant et collez le code suivant. N'oubliez pas de remplacer les ARN de ce code par les ARN des fonctions Lambda que vous avez créées.

   ```
   {
       "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. Spécifiez un nom pour votre machine à états. Pour ce faire, cliquez sur l'icône d'édition à côté du nom de la machine à états par défaut de MyStateMachine. Ensuite, dans Configuration de la machine d'état, spécifiez un nom dans le champ Nom de la machine d'état.

   Pour ce didacticiel, saisissez le nom **`ContinueAsNew`**.

7. (Facultatif) Dans Configuration de la machine à états, spécifiez d'autres paramètres de flux de travail, tels que le type de machine à états et son rôle d'exécution.

   Pour ce didacticiel, conservez toutes les sélections par défaut dans les paramètres State Machine.

   Si vous avez déjà créé un rôle IAM avec les autorisations appropriées pour votre machine d'état et que vous souhaitez l'utiliser, dans Autorisations, sélectionnez Choisir un rôle existant, puis sélectionnez un rôle dans la liste. Vous pouvez également sélectionner Entrer un ARN de rôle, puis fournir un ARN pour ce rôle IAM.

8. Dans la boîte de dialogue Confirmer la création du rôle, choisissez Confirmer pour continuer.

   Vous pouvez également choisir Afficher les paramètres des rôles pour revenir à la configuration de la machine State.

   > **Note**
   >
   > Si vous supprimez le rôle IAM créé par Step Functions, Step Functions ne pourra pas le recréer ultérieurement. De même, si vous modifiez le rôle (par exemple, en supprimant Step Functions des principes de la politique IAM), Step Functions ne pourra pas restaurer ses paramètres d'origine ultérieurement.

9. Enregistrez le nom de ressource Amazon (ARN) de cette machine d'état dans un fichier texte. Vous devez fournir l'ARN tout en autorisant la fonction Lambda à démarrer une nouvelle exécution de Step Functions.

Étape 4 : Mettre à jour la politique IAM

Pour vous assurer que votre fonction Lambda est autorisée à démarrer une nouvelle exécution de Step Functions, associez une politique en ligne au rôle IAM que vous utilisez pour votre fonction Lambda. `Restart` Pour plus d'informations, consultez la section Intégration de politiques intégrées dans le guide de l'utilisateur IAM.

> **Note**
>
> Vous pouvez mettre à jour la ligne `Resource` de l'exemple précédent pour faire référence à l'ARN de votre machine d'état `ContinueAsNew`. Cette limite la stratégie afin qu'elle ne puisse démarrer qu'une exécution de cette machine d'état spécifique.

```
{
 "Version": "2012-10-17",
 "Statement": [
     {
         "Sid": "VisualEditor0",
         "Effect": "Allow",
         "Action": [
             "states:StartExecution"
         ],
         "Resource": "arn:aws:states:us-east-2:123456789012stateMachine:ContinueAsNew"
     }
 ]
}
```

Étape 5 : Exécutez la machine d'état

Pour démarrer une exécution, fournissez une entrée qui inclut l'ARN de la machine d'état et un `executionCount` relatif au nombre de fois où une nouvelle exécution doit être démarrée.

1. Sur la ContinueAsNewpage, choisissez Démarrer l'exécution.

   La boîte de dialogue Démarrer l'exécution s'affiche.
2. Dans la boîte de dialogue Démarrer l'exécution, procédez comme suit :

   a. (Facultatif) Pour identifier votre exécution, vous pouvez lui donner un nom dans le champ Nom. Par défaut, Step Functions génère automatiquement un nom d'exécution unique.

> **Note**
>
> Step Functions vous permet de créer des noms pour les machines d'état, les exécutions et les activités, ainsi que des étiquettes contenant des caractères non ASCII. Ces noms non ASCII ne fonctionnent pas avec Amazon. CloudWatch Pour être sûr de pouvoir suivre CloudWatch les métriques, choisissez un nom qui utilise uniquement des caractères ASCII.

b. Dans le champ Entrée, entrez l'entrée JSON suivante pour exécuter votre flux de travail.

```
{
  "restart": {
    "StateMachineArn": "arn:aws:states:us-
east-1:123456789012:stateMachine:ContinueAsNew",
    "executionCount": 4
  }
}
```

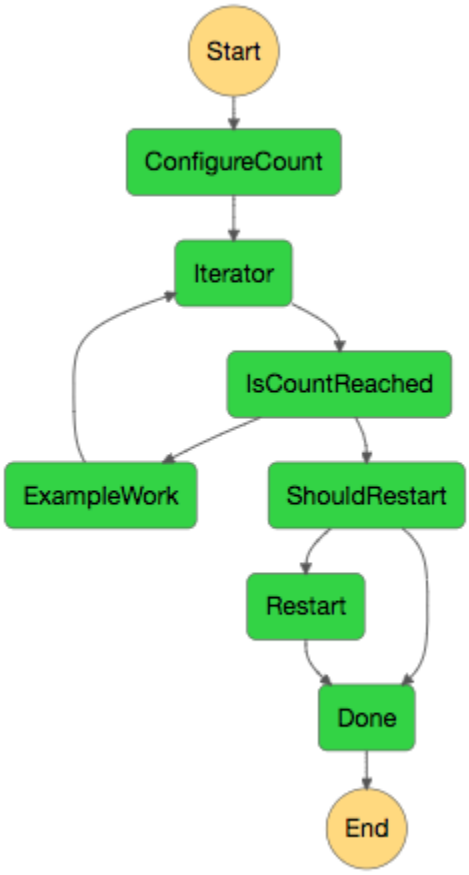
c. Mettez à jour le champ `StateMachineArn` avec l'ARN de votre machine d'état `ContinueAsNew`.

d. Choisissez Start execution (Démarrer l'exécution).

e. La console Step Functions vous dirige vers une page intitulée avec votre ID d'exécution. Cette page est connue sous le nom de page Détails de l'exécution. Sur cette page, vous pouvez consulter les résultats de l'exécution au fur et à mesure que l'exécution progresse ou une fois celle-ci terminée.

   Pour consulter les résultats de l'exécution, choisissez des états individuels dans la vue graphique, puis choisissez les onglets individuels du Détails de l'étape volet pour afficher les détails de chaque état, y compris les entrées, les sorties et la définition respectivement. Pour plus de détails sur les informations d'exécution que vous pouvez consulter sur la page Détails de l'exécution, voirPage de détails d'exécution — Vue d'ensemble de l'interface.

   La vue graphique affiche la première des quatre exécutions. Avant de se terminer, il va transmettre l'état `Restart` et démarrer une nouvelle exécution.



Au fur et à mesure que cette exécution est terminée, vous pouvez regarder la prochaine exécution en cours. Cliquez ContinueAsNewsur le lien en haut pour voir la liste des exécutions. Vous devriez voir à la fois l'exécution récemment clôturée et une exécution en cours lancée par la fonction `Restart` Lambda.

Une fois toutes les exécutions terminées, vous devez voir quatre exécutions réussies dans la liste. La première exécution démarrée affiche le nom que vous avez choisi et un nom a été généré pour les exécutions suivantes.

Déployer un exemple de projet d'approbation humaine

Ce didacticiel vous montre comment déployer un projet d'approbation humaine qui autorise une exécution AWS Step Functions à s'interrompre exécution au cours d'une tâche, et à attendre qu'un utilisateur réponde à un e-mail. Le flux de travail progresse jusqu'à l'état suivant une fois que l'utilisateur a approuvé la tâche pour poursuivre.

Le déploiement de la AWS CloudFormation pile incluse dans ce didacticiel créera toutes les ressources nécessaires, notamment :

- Ressources Amazon API Gateway
- Et AWS Lambda fonctions
- Une machine AWS Step Functions étatique
- Sujet d'e-mail d'Amazon Simple Notification Service
- AWS Identity and Access Management Rôles et autorisations associés

> **Note**
>
> Vous devrez fournir une adresse e-mail valide à laquelle vous aurez accès lors de la création de la AWS CloudFormation pile.

Pour plus d'informations, consultez la section Utilisation des CloudFormation modèles et la `AWS::StepFunctions::StateMachine` ressource du Guide de AWS CloudFormation l'utilisateur.

Rubriques

Étape 1 : Création d'un AWS CloudFormation modèle

1. Copiez l'exemple de code à partir de la section AWS CloudFormation Code source du modèle.

2. Collez la source du AWS CloudFormation modèle dans un fichier sur votre ordinateur local.

   Dans cet exemple, le fichier est nommé `human-approval.yaml`.

Étape 2 : créer une pile

1. Connectez-vous à la console AWS CloudFormation.
2. Choisissez Create Stack, puis choisissez Avec de nouvelles ressources (standard).
3. Sur la page Créer une pile, procédez de la manière suivante :

   a. Dans la section Prérequis - Préparer le modèle, assurez-vous que le modèle est prêt est sélectionné.

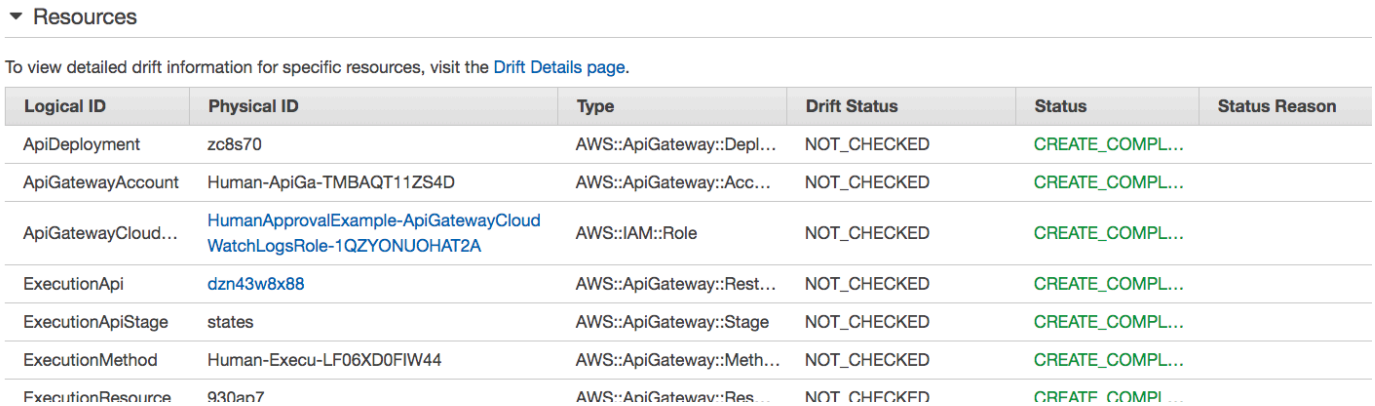
   b. Dans la section Spécifier le modèle, choisissez Télécharger un fichier modèle, puis choisissez Choisir un fichier pour télécharger le `human-approval.yaml` fichier que vous avez créé précédemment qui inclut le code source du modèle.
4. Choisissez Suivant.
5. Sur la page Spécifier les détails de la pile, procédez comme suit :

   a. Dans Nom de la pile, entrez le nom de votre pile.

b. Sous Paramètres, entrez une adresse e-mail valide. Vous utiliserez cette adresse e-mail pour vous abonner à la rubrique Amazon SNS.

6. Choisissez Next, puis de nouveau Next.
7. Sur la page de révision, choisissez Je reconnais que cela AWS CloudFormation pourrait créer des ressources IAM, puis sélectionnez Créer.

   AWS CloudFormation commence à créer votre pile et affiche le statut CREATE_IN_PROGRESS. Lorsque le processus est terminé, AWS CloudFormation affiche le statut CREATE_COMPLETE.
8. (Facultatif) Pour afficher les ressources de votre pile, sélectionnez la pile et choisissez l'onglet Ressources.

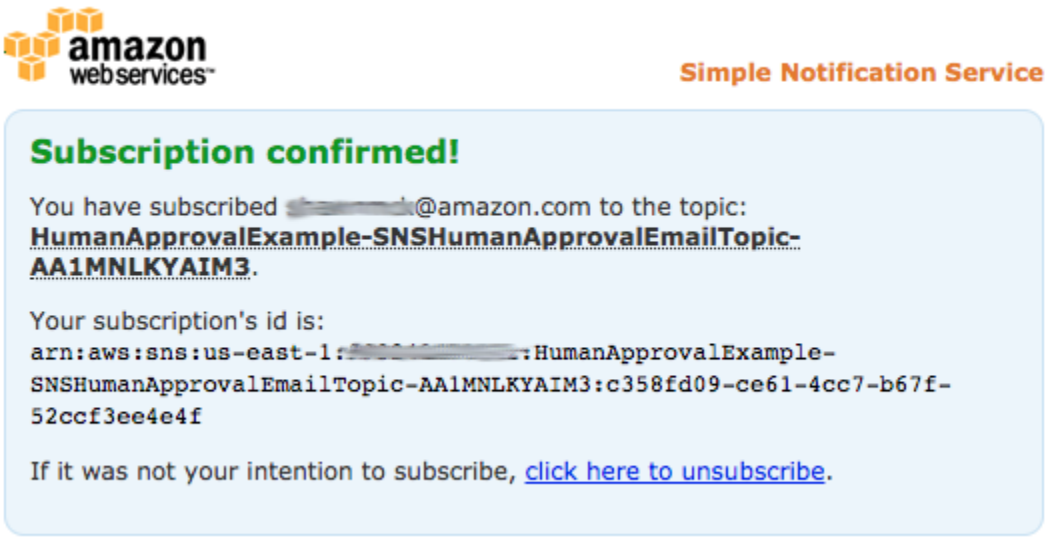


Étape 3 : Approuver l'abonnement Amazon SNS

Une fois le sujet Amazon SNS créé, vous recevrez un e-mail vous demandant de confirmer votre inscription.

1. Ouvrez le compte e-mail que vous avez fourni lors de la création de la AWS CloudFormation pile.
2. Ouvrez le message AWS Notification - Confirmation d'abonnement envoyé par no-reply@sns.amazonaws.com

   L'e-mail indiquera le nom de la ressource Amazon pour le sujet Amazon SNS, ainsi qu'un lien de confirmation.
3. Choisissez le lien de confirmation de l'abonnement.



Étape 4 : Exécutez la machine d'état

1. Sur la HumanApprovalLambdaStateMachinepage, choisissez Démarrer l'exécution.

   La boîte de dialogue Démarrer l'exécution s'affiche.

2. Dans la boîte de dialogue Démarrer l'exécution, procédez comme suit :

   a. (Facultatif) Pour identifier votre exécution, vous pouvez lui donner un nom dans le champ Nom. Par défaut, Step Functions génère automatiquement un nom d'exécution unique.

   > **Note**
   >
   > Step Functions vous permet de créer des noms pour les machines d'état, les exécutions et les activités, ainsi que des étiquettes contenant des caractères non ASCII. Ces noms non ASCII ne fonctionnent pas avec Amazon. CloudWatch Pour être sûr de pouvoir suivre CloudWatch les métriques, choisissez un nom qui utilise uniquement des caractères ASCII.

   b. Dans le champ Entrée, entrez l'entrée JSON suivante pour exécuter votre flux de travail.

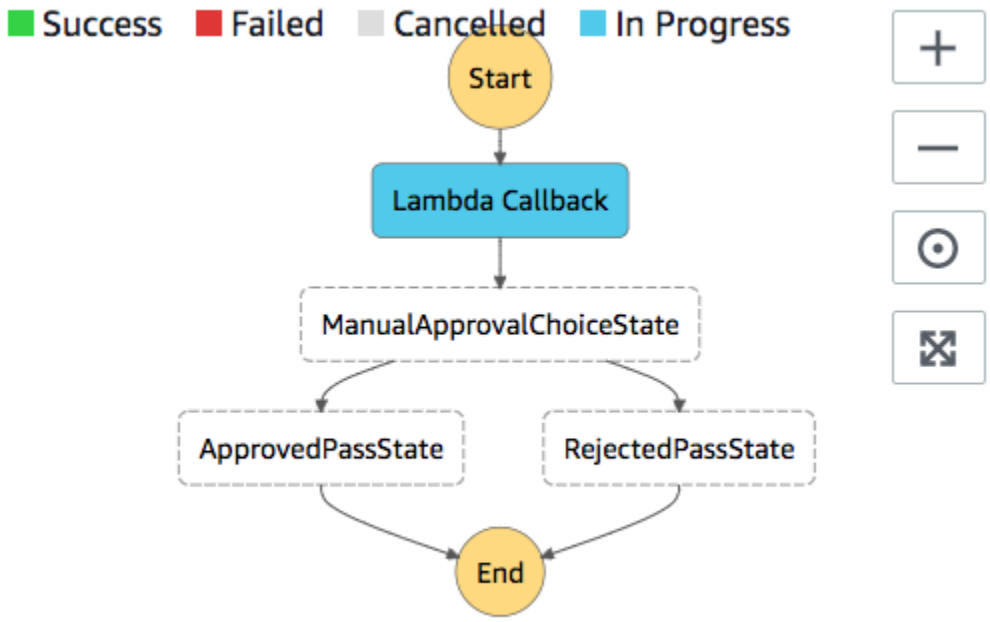
```
{
    "Comment": "Testing the human approval tutorial."
}
```

c. Choisissez Start execution (Démarrer l'exécution).

L'exécution de la machine à ApprovalTestétats commence et s'arrête lors de la tâche Lambda Callback.

d. La console Step Functions vous dirige vers une page intitulée avec votre ID d'exécution. Cette page est connue sous le nom de page Détails de l'exécution. Sur cette page, vous pouvez consulter les résultats de l'exécution au fur et à mesure que l'exécution progresse ou une fois celle-ci terminée.

Pour consulter les résultats de l'exécution, choisissez des états individuels dans la vue graphique, puis choisissez les onglets individuels du Détails de l'étape volet pour afficher les détails de chaque état, y compris les entrées, les sorties et la définition respectivement. Pour plus de détails sur les informations d'exécution que vous pouvez consulter sur la page Détails de l'exécution, voirPage de détails d'exécution — Vue d'ensemble de l'interface.

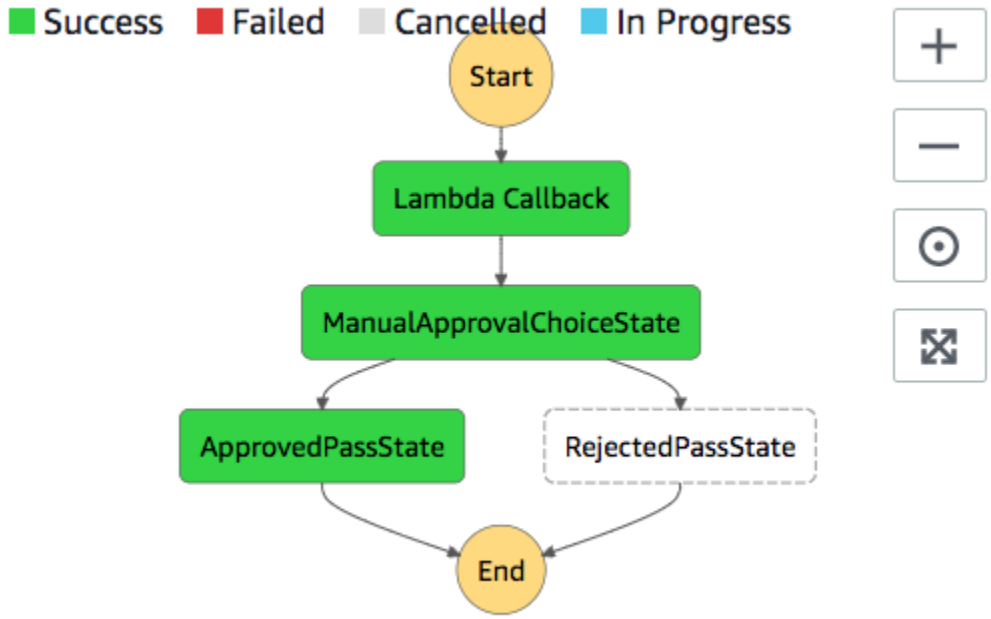


3. Dans le compte e-mail que vous avez utilisé pour le sujet Amazon SNS plus tôt, ouvrez le message dont le sujet est Required approval from. AWS Step Functions

Le message comprend les URL distinctes pour Approve (Approuver) et Reject (Rejeter).

4. Choisissez l'URL Approve (Approuver).

Le flux de travail continue en fonction de votre choix.



AWS CloudFormation Code source du modèle

Utilisez ce AWS CloudFormation modèle pour déployer un exemple de flux de travail de processus d'approbation humaine.

```
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
```

Afficher les traces X-Ray dans Step Functions

Dans ce didacticiel, vous allez apprendre à utiliser X-Ray pour suivre les erreurs qui se produisent lors de l'exécution d'une machine à états. Vous pouvez l'utiliser AWS X-Raypour visualiser les composants de votre machine d'état, identifier les goulots d'étranglement liés aux performances et résoudre les demandes qui ont entraîné une erreur. Dans ce didacticiel, vous allez créer plusieurs fonctions Lambda qui produisent des erreurs de manière aléatoire, que vous pourrez ensuite suivre et analyser à l'aide de X-Ray.

Le Création d'une machine d'état Step Functions utilisant Lambda didacticiel vous explique comment créer une machine à états qui appelle une fonction Lambda. Si vous avez terminé ce didacticiel, passez à l'étape 2 et utilisez le rôle AWS Identity and Access Management (IAM) que vous avez créé précédemment.

Rubriques

- Étape 1 : créer un rôle IAM pour Lambda
- Étape 2 : créer une fonction Lambda
- Étape 3 : créer deux autres fonctions Lambda
- Étape 4 : Création d'une machine à états
- Étape 5 : Exécutez la machine d'état

Étape 1 : créer un rôle IAM pour Lambda

Les deux AWS Lambda AWS Step Functions peuvent exécuter du code et accéder à AWS des ressources (par exemple, des données stockées dans des compartiments Amazon S3). Pour garantir la sécurité, vous devez autoriser Lambda et Step Functions à accéder à ces ressources.

Lambda vous oblige à attribuer un rôle AWS Identity and Access Management (IAM) lorsque vous créez une fonction Lambda, de la même manière que Step Functions vous oblige à attribuer un rôle IAM lorsque vous créez une machine à états.

Vous utilisez la console IAM pour créer un rôle lié à un service.

Pour créer un rôle (console)

1. Connectez-vous à la console IAM AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/iam/.
2. Dans le panneau de navigation de la console IAM, sélectionnez Roles (Rôles). Puis choisissez Create role (Créer un rôle).
3. Choisissez le type AWS de rôle de service, puis Lambda.
4. Choisissez le cas d'utilisation de Lambda. Les cas d'utilisation sont définis par le service pour inclure la politique d'approbation requise par le service. Choisissez ensuite Suivant : Autorisations.
5. Choisissez une ou plusieurs stratégies d'autorisation à attacher au rôle (par exemple, `AWSLambdaBasicExecutionRole`). Consultez Modèle d'autorisations AWS Lambda.

   Cochez la case en regard de la stratégie qui affecte les autorisations que vous voulez octroyer au rôle, puis choisissez Next: Review.
6. Entrez un nom de rôle.
7. (Facultatif) Dans le champ Description du rôle, modifiez la description du nouveau rôle lié à un service.
8. Passez en revue les informations du rôle, puis choisissez Create role (Créer un rôle).

Étape 2 : créer une fonction Lambda

Votre fonction Lambda générera des erreurs ou expirera de manière aléatoire, produisant des exemples de données à afficher dans X-Ray.

> **Important**
>
> Assurez-vous que votre fonction Lambda se trouve sous le même AWS compte et dans la même AWS région que votre machine à états.

1. Ouvrez la console Lambda et choisissez Create function.
2. Dans la section Créer une fonction, choisissez Créer à partir de zéro.
3. Dans la section Informations de base, configurez votre fonction Lambda :
   a. Sous Nom de la fonction, saisissez `TestFunction1`.
   b. Pour Exécution, choisissez Node.js 18.x.
   c. Pour Rôle, sélectionnez Choisir un rôle existant.
   d. Pour Rôle existant, sélectionnez le rôle Lambda que vous avez créé précédemment.

      > **Note**
      >
      > Si le rôle IAM que vous avez créé n'apparaît pas dans la liste, il se peut qu'il ait encore besoin de quelques minutes pour se propager vers Lambda.

   e. Choisissez Créer une fonction.

      Lorsque votre fonction Lambda est créée, notez son Amazon Resource Name (ARN) dans le coin supérieur droit de la page. Par exemple :

      ```
      arn:aws:lambda:us-east-1:123456789012:function:TestFunction1
      ```

4. Copiez le code suivant pour la fonction Lambda dans la section Code de fonction de la page ***TestFunction1***.

```
function getRandomSeconds(max) {
    return Math.floor(Math.random() * Math.floor(max)) * 1000;
}
function sleep(ms) {
    return new Promise(resolve => setTimeout(resolve, ms));
}
export const handler = async (event) => {
    if(getRandomSeconds(4) === 0) {
```

```
        throw new Error("Something went wrong!");
    }
    let wait_time = getRandomSeconds(5);
    await sleep(wait_time);
    return { 'response': true }
};
```

Ce code crée des défaillances chronométrées de manière aléatoire, qui seront utilisées pour générer des exemples d'erreurs dans votre machine à états, qui pourront être visualisés et analysés à l'aide des traces X-Ray.

5. Choisissez Enregistrer.

Étape 3 : créer deux autres fonctions Lambda

Créez deux autres fonctions Lambda.

1. Répétez l'étape 2 pour créer deux autres fonctions Lambda. Pour la fonction suivante, dans Nom de la fonction, entrez`TestFunction2`. Pour la dernière fonction, dans Nom de la fonction, entrez`TestFunction3`.
2. Dans la console Lambda, vérifiez que vous disposez désormais de trois fonctions Lambda,, et`TestFunction1`. `TestFunction2` `TestFunction3`

Étape 4 : Création d'une machine à états

Dans cette étape, vous allez utiliser la console Step Functions pour créer une machine à états à trois `Task` états. Chaque `Task` état fera référence à l'une de vos trois fonctions Lambda.

1. Ouvrez la console Step Functions et choisissez Create state machine.

> **Important**
>
> Assurez-vous que votre machine à états se trouve sous le même AWS compte et dans la même région que les fonctions Lambda que vous avez créées précédemment aux étapes 2 et 3.

2. Dans la boîte de dialogue Choisir un modèle, sélectionnez Vide.
3. Choisissez Select (Sélectionner). Cela ouvre Workflow Studio dansMode de conception.

4. Pour ce didacticiel, vous allez écrire la définition Amazon States Language (ASL) de votre machine à états dans leÉditeur de code. Pour ce faire, sélectionnez Code.

5. Supprimez le code standard existant et collez le code suivant. Dans la définition de l'état de la tâche, n'oubliez pas de remplacer les exemples d'ARN par les ARN des fonctions Lambda que vous avez créées.

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

Voici une description de votre machine à états utilisant le langage Amazon States. Il définit trois `Task` états nommés`CallTestFunction1`, `CallTestFunction2` et`CallTestFunction3`. Chacune appelle l'une de vos trois fonctions Lambda. Pour plus d'informations, consultez Structure de la machine d'état.

6. Spécifiez un nom pour votre machine à états. Pour ce faire, cliquez sur l'icône d'édition à côté du nom de la machine à états par défaut de MyStateMachine. Ensuite, dans Configuration de la machine d'état, spécifiez un nom dans le champ Nom de la machine d'état.

   Pour ce didacticiel, saisissez le nom **`TraceFunctions`**.

7. (Facultatif) Dans Configuration de la machine à états, spécifiez d'autres paramètres de flux de travail, tels que le type de machine à états et son rôle d'exécution.

   Pour ce didacticiel, sous Configuration supplémentaire, choisissez Enable X-Ray Tracing. Conservez toutes les autres sélections par défaut dans les paramètres State Machine.

Si vous avez déjà créé un rôle IAM avec les autorisations appropriées pour votre machine d'état et que vous souhaitez l'utiliser, dans Autorisations, sélectionnez Choisir un rôle existant, puis sélectionnez un rôle dans la liste. Vous pouvez également sélectionner Entrer un ARN de rôle, puis fournir un ARN pour ce rôle IAM.

8. Dans la boîte de dialogue Confirmer la création du rôle, choisissez Confirmer pour continuer.

   Vous pouvez également choisir Afficher les paramètres des rôles pour revenir à la configuration de la machine State.

> **Note**
>
> Si vous supprimez le rôle IAM créé par Step Functions, Step Functions ne pourra pas le recréer ultérieurement. De même, si vous modifiez le rôle (par exemple, en supprimant Step Functions des principes de la politique IAM), Step Functions ne pourra pas restaurer ses paramètres d'origine ultérieurement.

Étape 5 : Exécutez la machine d'état

Les exécutions par State Machine sont des instances dans lesquelles vous exécutez votre flux de travail pour effectuer des tâches.

1. Sur la ***TraceFunctions***page, choisissez Démarrer l'exécution.

   La page Nouvelle exécution s'affiche.

2. Dans la boîte de dialogue Démarrer l'exécution, procédez comme suit :

   a. (Facultatif) Pour identifier votre exécution, vous pouvez lui donner un nom dans le champ Nom. Par défaut, Step Functions génère automatiquement un nom d'exécution unique.

> **Note**
>
> Step Functions vous permet de créer des noms pour les machines d'état, les exécutions et les activités, ainsi que des étiquettes contenant des caractères non ASCII. Ces noms non ASCII ne fonctionnent pas avec Amazon. CloudWatch Pour être sûr de pouvoir suivre CloudWatch les métriques, choisissez un nom qui utilise uniquement des caractères ASCII.

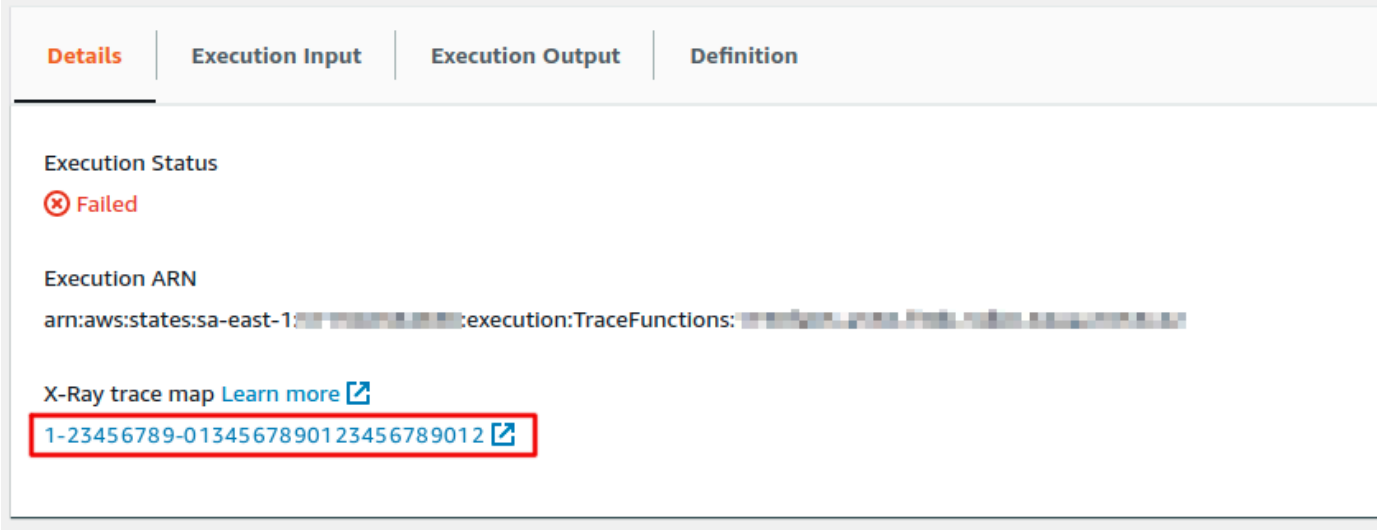
b. Choisissez Start execution (Démarrer l'exécution).

c. La console Step Functions vous dirige vers une page intitulée avec votre ID d'exécution. Cette page est connue sous le nom de page Détails de l'exécution. Sur cette page, vous pouvez consulter les résultats de l'exécution au fur et à mesure que l'exécution progresse ou une fois celle-ci terminée.

   Pour consulter les résultats de l'exécution, choisissez des états individuels dans la vue graphique, puis choisissez les onglets individuels du Détails de l'étape volet pour afficher les détails de chaque état, y compris les entrées, les sorties et la définition respectivement. Pour plus de détails sur les informations d'exécution que vous pouvez consulter sur la page Détails de l'exécution, voirPage de détails d'exécution — Vue d'ensemble de l'interface.

   Exécutez plusieurs (au moins trois) exécutions.

3. Une fois les exécutions terminées, suivez le lien de la carte de trace X-Ray. Vous pouvez consulter la trace pendant qu'une exécution est toujours en cours, mais vous souhaiterez peut-être voir les résultats de l'exécution avant de consulter la carte de trace de X-Ray.



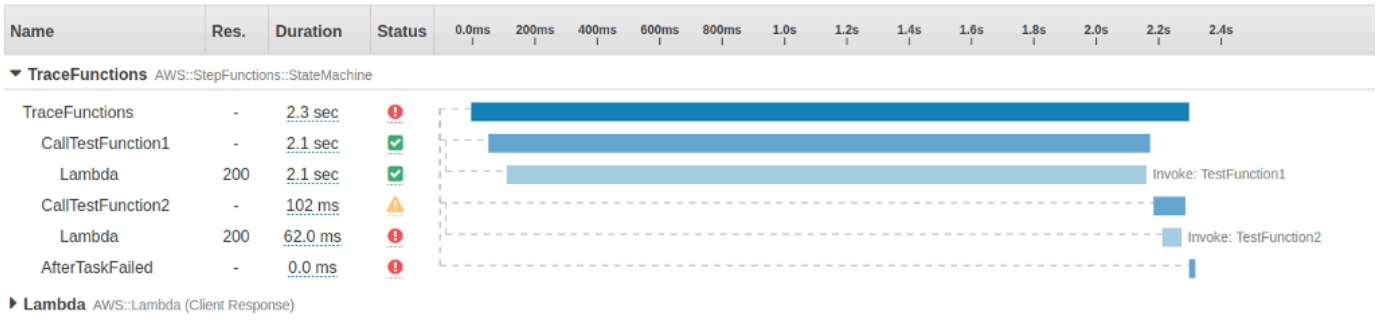
4. Consultez la carte des services pour identifier les erreurs, les connexions présentant une latence élevée ou les traces de demandes infructueuses. Dans cet exemple, vous pouvez voir le volume de trafic reçu par chaque fonction. `TestFunction2`a été appelé plus souvent que`TestFunction3`, et `TestFunction1` a été appelé plus de deux fois plus souvent que`TestFunction2`.

   La cartographie des services indique l'état de chaque nœud en lui appliquant une couleur déterminée par le rapport entre le nombre d'appels réussis et le nombre d'erreurs ou d'échecs:

- Vert pour les appels réussis
- Rouge pour les erreurs serveur (erreurs de type 500)
- Jaune pour les erreurs client (erreurs de type 400)
- Violet pour les erreurs de limitation (erreur 429, nombre de requêtes trop élevé)

Vous pouvez également choisir un nœud de service pour afficher les demandes relatives à ce nœud, ou une limite entre deux nœuds pour afficher les demandes ayant transité par cette connexion.

5. Consultez la carte de suivi X-Ray pour voir ce qui s'est passé à chaque exécution. La vue chronologique affiche une hiérarchie de segments et de sous-segments. La première entrée de la liste est le segment, qui représente toutes les données enregistrées par le service pour une seule demande. Sous le segment se trouvent les sous-segments. Cet exemple montre les sous-segments enregistrés par les fonctions Lambda.



Pour plus d'informations sur la compréhension des traces de rayons X et sur l'utilisation de X-Ray avec Step Functions, consultez le AWS X-Ray et Step Functions

Collectez des informations sur le compartiment Amazon S3 à l'aide des AWS intégrations de services du SDK

Ce didacticiel explique comment effectuer une intégration du AWS SDK avec Amazon Simple Storage Service. La machine d'état que vous créez dans ce didacticiel collecte des informations sur vos compartiments Amazon S3, puis répertorie vos compartiments ainsi que les informations de version pour chaque compartiment de la région actuelle.

Rubriques

- Étape 1 : Création de la machine à états
- Étape 2 : ajouter les autorisations de rôle IAM nécessaires
- Étape 3 : Exécuter une exécution automatique à l'état standard
- Étape 4 : Exécuter une exécution par machine à états express

Étape 1 : Création de la machine à états

À l'aide de la console Step Functions, vous allez créer une machine à `Task` états qui inclut un état répertoriant tous les buckets Amazon S3 du compte et de la région actuels. Ensuite, vous ajouterez un autre `Task` état qui invoquera l'`HeadBucket`API pour vérifier si le compartiment renvoyé est accessible dans la région actuelle. Si le bucket n'est pas accessible, l'appel `HeadBucket` d'API renvoie l'`S3.S3Exception`erreur. Vous allez inclure un `Catch` bloc pour intercepter cette exception et un `Pass` état comme état de secours.

1. Ouvrez la console Step Functions et choisissez Create state machine.

2. Dans la boîte de dialogue Choisir un modèle, sélectionnez Vide.
3. Choisissez Select (Sélectionner). Cela ouvre Workflow Studio dansMode de conception.
4. Pour ce didacticiel, vous allez écrire la définition Amazon States Language (ASL) de votre machine à états dans leÉditeur de code. Pour ce faire, sélectionnez Code.
5. Supprimez le code standard existant et collez la définition de machine à états suivante.

```
{
  "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. Spécifiez un nom pour votre machine à états. Pour ce faire, cliquez sur l'icône d'édition à côté du nom de la machine à états par défaut de MyStateMachine. Ensuite, dans Configuration de la machine d'état, spécifiez un nom dans le champ Nom de la machine d'état.

   Pour ce didacticiel, saisissez le nom **`Gather-S3-Bucket-Info-Standard`**.

7. (Facultatif) Dans Configuration de la machine à états, spécifiez d'autres paramètres de flux de travail, tels que le type de machine à états et son rôle d'exécution.

   Conservez toutes les sélections par défaut dans les paramètres State Machine.

   Si vous avez déjà créé un rôle IAM avec les autorisations appropriées pour votre machine d'état et que vous souhaitez l'utiliser, dans Autorisations, sélectionnez Choisir un rôle existant, puis sélectionnez un rôle dans la liste. Vous pouvez également sélectionner Entrer un ARN de rôle, puis fournir un ARN pour ce rôle IAM.

8. Dans la boîte de dialogue Confirmer la création du rôle, choisissez Confirmer pour continuer.

Vous pouvez également choisir Afficher les paramètres des rôles pour revenir à la configuration de la machine State.

> **Note**
>
> Si vous supprimez le rôle IAM créé par Step Functions, Step Functions ne pourra pas le recréer ultérieurement. De même, si vous modifiez le rôle (par exemple, en supprimant Step Functions des principes de la politique IAM), Step Functions ne pourra pas restaurer ses paramètres d'origine ultérieurement.

À l'étape 2, vous allez ajouter les autorisations manquantes au rôle de machine à états.

Étape 2 : ajouter les autorisations de rôle IAM nécessaires

Pour recueillir des informations sur les compartiments Amazon S3 de votre région actuelle, vous devez fournir à votre machine d'état les autorisations nécessaires pour accéder aux compartiments Amazon S3.

1. Sur la page State Machine, choisissez l'ARN du rôle IAM pour ouvrir la page Rôles du rôle State Machine.
2. Sélectionnez Ajouter des autorisations, puis Ajouter la politique.
3. Choisissez l'onglet JSON, puis collez les autorisations suivantes dans l'éditeur JSON.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "s3:ListAllMyBuckets",
                "s3:ListBucket",
                "s3:GetBucketVersioning"
            ],
            "Resource": "*"
        }
    ]
```

```
}
```

4. Choisissez Examiner une politique.
5. Sous Examiner une stratégie, pour le Nom de la stratégie, saisissez **`s3-bucket-permissions`**.
6. Choisissez Créer une stratégie.

Étape 3 : Exécuter une exécution automatique à l'état standard

1. Sur la page Gather-S3-Bucket-Info-Standard, choisissez Démarrer l'exécution.
2. Dans la boîte de dialogue Démarrer l'exécution, procédez comme suit :

   a. (Facultatif) Pour identifier votre exécution, vous pouvez lui donner un nom dans le champ Nom. Par défaut, Step Functions génère automatiquement un nom d'exécution unique.

   > **Note**
   >
   > Step Functions vous permet de créer des noms pour les machines d'état, les exécutions et les activités, ainsi que des étiquettes contenant des caractères non ASCII. Ces noms non ASCII ne fonctionnent pas avec Amazon. CloudWatch Pour être sûr de pouvoir suivre CloudWatch les métriques, choisissez un nom qui utilise uniquement des caractères ASCII.

   b. Choisissez Start execution (Démarrer l'exécution).

   c. La console Step Functions vous dirige vers une page intitulée avec votre ID d'exécution. Cette page est connue sous le nom de page Détails de l'exécution. Sur cette page, vous pouvez consulter les résultats de l'exécution au fur et à mesure que l'exécution progresse ou une fois celle-ci terminée.

   Pour consulter les résultats de l'exécution, choisissez des états individuels dans la vue graphique, puis choisissez les onglets individuels du Détails de l'étape volet pour afficher les détails de chaque état, y compris les entrées, les sorties et la définition respectivement. Pour plus de détails sur les informations d'exécution que vous pouvez consulter sur la page Détails de l'exécution, voirPage de détails d'exécution — Vue d'ensemble de l'interface.

Étape 4 : Exécuter une exécution par machine à états express

1. Créez une machine à états Express à l'aide de la définition de machine à états fournie à l'étape 1. Assurez-vous d'inclure également les autorisations de rôle IAM nécessaires, comme expliqué à l'étape 2.

   > **Tip**
   >
   > Pour la distinguer de la machine standard que vous avez créée précédemment, nommez la machine à états Express comme**`Gather-S3-Bucket-Info-Express`**.

2. Sur la page Gather-S3-Bucket-Info-Standard, choisissez Démarrer l'exécution.
3. Dans la boîte de dialogue Démarrer l'exécution, procédez comme suit :

   a. (Facultatif) Pour identifier votre exécution, vous pouvez lui donner un nom dans le champ Nom. Par défaut, Step Functions génère automatiquement un nom d'exécution unique.

      > **Note**
      >
      > Step Functions vous permet de créer des noms pour les machines d'état, les exécutions et les activités, ainsi que des étiquettes contenant des caractères non ASCII. Ces noms non ASCII ne fonctionnent pas avec Amazon. CloudWatch Pour être sûr de pouvoir suivre CloudWatch les métriques, choisissez un nom qui utilise uniquement des caractères ASCII.

   b. Choisissez Start execution (Démarrer l'exécution).

   c. La console Step Functions vous dirige vers une page intitulée avec votre ID d'exécution. Cette page est connue sous le nom de page Détails de l'exécution. Sur cette page, vous pouvez consulter les résultats de l'exécution au fur et à mesure que l'exécution progresse ou une fois celle-ci terminée.

      Pour consulter les résultats de l'exécution, choisissez des états individuels dans la vue graphique, puis choisissez les onglets individuels du Détails de l'étape volet pour afficher les détails de chaque état, y compris les entrées, les sorties et la définition respectivement. Pour plus de détails sur les informations d'exécution que vous pouvez consulter sur la page Détails de l'exécution, voirPage de détails d'exécution — Vue d'ensemble de l'interface.

Outils pour développeurs

Les ressources suivantes contiennent des informations supplémentaires sur la création de flux de travail sans serveur et l'utilisation de machines à états :

- AWS CDK

- AWS Boîte à outils pour VS Code

Les rubriques ci-dessous contiennent des informations qui vous apprennent à créer, tester et déboguer des machines à états.

Rubriques

- Options de développement
- AWS Step Functions et AWS SAM
- Utilisation de Workflow Studio dans Application Composer
- Création d'une machine à états Lambda pour Step Functions à l'aide de AWS CloudFormation
- Création d'une machine à Lambda états pour Step Functions utiliser AWS CDK
- Création d'une API REST API Gateway avec une machine synchrone Express State à l'aide du AWS CDK
- AWS Step Functions SDK de science des données pour Python
- Déploiement de machines d'état à l'aide de Terraform

Options de développement

Vous pouvez implémenter vos machines AWS Step Functions d'état de plusieurs manières, par exemple en utilisant la console, les SDK ou une version locale de Step Functions pour les tests et le développement.

Rubriques

- Console Step Functions
- AWS SDK
- Flux de travail standard et express

- API du service HTTPS
- Environnements de développement
- Points de terminaison
- AWS CLI
- Step Functions Local
- AWS Toolkit for Visual Studio Code
- AWS Serverless Application Model et Step Functions
- Terraform et Step Functions
- Support du format de définition

Console Step Functions

Vous pouvez définir une machine à états à l'aide de la console Step Functions. Vous pouvez écrire des machines à états complexes dans le cloud sans utiliser d'environnement de développement local en les utilisant AWS Lambda pour fournir du code pour vos tâches. Une fois écrit, vous pouvez utiliser la console Step Functions pour définir votre machine à états à l'aide de l'Amazon States Language.

Le didacticiel Creating a Lambda State Machine utilise cette technique pour créer une machine à états simple, l'exécuter et afficher ses résultats.

Simulateur de flux de données

Vous pouvez concevoir, implémenter et déboguer des flux de travail dans la console Step Functions. Vous pouvez également contrôler le flux de données dans vos flux de travail en utilisant le traitement des `JsonPath` entrées et des sorties. Utilisez le simulateur de flux de données de la console Step Functions pour découvrir comment les informations circulent d'un état à l'autre et pour comprendre comment filtrer et manipuler les données. Cet outil simule chacun des champs utilisés par Step Functions pour traiter les données, tels que`InputPath`,`Parameters`, `ResultSelectorOutputPath`, et`ResultPath`.

Pour plus d’informations, consultez Simulateur de flux de données.

AWS SDK

Step Functions est compatible avec AWS les SDK pour Java, .NET, Ruby, PHP, Python (Boto 3) JavaScript, Go et C++. Ces SDK constituent un moyen pratique d'utiliser les actions de l'API HTTPS Step Functions dans plusieurs langages de programmation.

Vous pouvez développer des machines d'état, des activités ou des démarreurs de machine d'état à l'aide des actions d'API proposées par les bibliothèques de kits SDK. Vous pouvez également accéder aux opérations de visibilité à l'aide de ces bibliothèques pour développer vos propres outils de surveillance et de reporting Step Functions.

Pour utiliser Step Functions avec d'autres AWS services, consultez la documentation de référence AWS des kits SDK et outils actuels pour Amazon Web Services.

> **Note**
>
> Step Functions ne prend en charge que les points de terminaison HTTPS.

Flux de travail standard et express

Lorsque vous créez une nouvelle machine d'état, vous devez sélectionner un Type Standard ou Express. Dans les deux cas, vous définissez votre machine à états à l'aide de l'Amazon States Language. Les exécutions de votre machine d'état se comportent différemment, en fonction du Type que vous sélectionnez. Le type que vous choisissez ne peut pas être modifié une fois votre machine à états créée.

Pour plus d’informations, consultez Journalisation à l'aideCloudWatchJournaux.

API du service HTTPS

Step Functions fournit des opérations de service accessibles via des requêtes HTTPS. Vous pouvez utiliser ces opérations pour communiquer directement avec Step Functions et pour développer vos propres bibliothèques dans n'importe quel langage capable de communiquer avec Step Functions via HTTPS.

Vous pouvez développer des machines d'état, des activités ou des démarreurs de machine d'état à l'aide des actions d'API de service. Vous pouvez également accéder aux opérations de visibilité via les actions d'API pour développer vos propres outils de surveillance et de reporting.

Pour obtenir des informations détaillées sur les actions d'API, consultez la référence des AWS Step Functions API.

Environnements de développement

Vous devez configurer un environnement de développement compatible avec le langage de programmation que vous comptez utiliser.

Par exemple, pour développer pour Step Functions à l'aide de Java, vous devez installer un environnement de développement Java, tel que le AWS SDK for Java, sur chacun de vos postes de travail de développement. Si vous utilisez Eclipse IDE pour le développement Java, vous devez également installer le AWS Toolkit for Eclipse. Le plug-in Eclipse ajoute des fonctions utiles pour le développement sur AWS.

Si votre langage de programmation nécessite un environnement d'exécution, vous devez configurer l'environnement sur chaque ordinateur sur lequel ces processus seront exécutés.

Points de terminaison

Pour réduire la latence et stocker les données dans un emplacement qui répond à vos besoins, Step Functions fournit des points de terminaison dans différentes AWS régions.

Chaque point de terminaison de Step Functions est totalement indépendant. Une machine d'état ou une activité n'existe que dans la région où elle a été créée. Toutes les machines d'état et activités que vous créez dans une région ne partagent pas de données ou d'attributs avec celles créées dans une autre région. Par exemple, vous pouvez enregistrer une machine d'état nommée `STATES-Flows-1` dans deux régions différentes. La machine `STATES-Flows-1` d'état d'une région ne partagera pas de données ou d'attributs avec la machine d'`STATES-Flow-1`état de l'autre région.

Pour une liste des points de terminaison Step Functions, voir AWS Step Functions Régions et points de terminaison dans le. Références générales AWS

AWS CLI

Vous pouvez accéder à de nombreuses fonctionnalités de Step Functions depuis le AWS Command Line Interface (AWS CLI). AWS CLI Il s'agit d'une alternative à l'utilisation de la console Step Functions ou, dans certains cas, à la programmation à l'aide des actions de l'API Step Functions. Par exemple, vous pouvez utiliser le AWS CLI pour créer une machine à états, puis répertorier vos machines à états existantes.

Vous pouvez utiliser les commandes Step Functions AWS CLI pour démarrer et gérer des exécutions, interroger les activités, enregistrer le rythme cardiaque des tâches, etc. Pour une liste complète des commandes Step Functions, une description des arguments disponibles et des exemples illustrant leur utilisation, consultez le AWS CLI Command Reference.

AWS CLI les commandes suivent de près le langage Amazon States. Vous pouvez donc les utiliser AWS CLI pour en savoir plus sur les actions de l'API Step Functions. Vous pouvez également utiliser vos connaissances actuelles en matière d'API pour prototyper du code ou exécuter des actions Step Functions depuis la ligne de commande.

Step Functions Local

À des fins de test et de développement, vous pouvez installer et exécuter Step Functions sur votre machine locale. Avec Step Functions Local, vous pouvez démarrer une exécution sur n'importe quelle machine.

La version locale de Step Functions peut invoquer des AWS Lambda fonctions, à la fois dans AWS et lors d'une exécution locale. Vous pouvez également coordonner d'autres AWS services pris en charge. Pour plus d’informations, consultez Tester les machines d'état localement.

> **Note**
>
> Step Functions Local utilise des comptes factices pour fonctionner.

AWS Toolkit for Visual Studio Code

Vous pouvez utiliser VS Code pour interagir avec des machines à états distants et développer des machines à états localement. Vous pouvez créer ou mettre à jour des machines à états, répertorier les machines à états existantes et exécuter ou télécharger une machine à états. VS Code vous permet également de créer de nouvelles machines d'état à partir de modèles, de voir une visualisation de votre machine d'état et d’accéder à des extraits de code, mais aussi de saisir et de valider du code.

Pour plus d'informations, consultez le guide de AWS Toolkit for Visual Studio Code l'utilisateur

AWS Serverless Application Model et Step Functions

Step Functions est intégré au AWS Serverless Application Model, qui vous permet d'intégrer des flux de travail aux fonctions Lambda, aux API et aux événements pour créer des applications sans serveur.

Vous pouvez également utiliser la AWS SAM CLI conjointement avec le dans le AWS Toolkit for Visual Studio Code cadre d'une expérience intégrée.

Pour plus d’informations, consultez AWS Step Functions et AWS SAM.

Terraform et Step Functions

Terraform by HashiCorp est un framework permettant de créer des applications utilisant l'infrastructure en tant que code (IaC). Avec Terraform, vous pouvez créer des machines d'état et utiliser des fonctionnalités, telles que la prévisualisation des déploiements d'infrastructure et la création de modèles réutilisables. Les modèles Terraform vous aident à maintenir et à réutiliser le code en le décomposant en petits morceaux.

Pour plus d’informations, consultez Déploiement de machines d'état à l'aide de Terraform.

Support du format de définition

Step Functions propose une variété d'outils qui vous permettent de fournir les définitions de vos machines à états dans différents formats. Une définition du langage Amazon States (ASL) qui spécifie les détails de votre machine à états peut être fournie sous forme de chaîne ou d'objet sérialisé à l'aide de JSON ou YAML.

> **Note**
>
> YAML autorise les commentaires sur une seule ligne. Les commentaires YAML fournis dans la partie de définition de la machine d'état d'un modèle ne seront pas reportés dans la définition de la ressource créée. Vous pouvez plutôt utiliser la `Comment` propriété dans la définition de la machine à états. Pour plus d'informations, consultez la page Structure de la machine d'État.

Le tableau suivant indique quels outils prennent en charge les définitions basées sur le protocole ASL.

Support du format de définition par outil

	JSON	YAML	Langue des États Amazon Stringifiée		
Console Step Functions	✓				
API du service HTTPS			✓		
AWS INTERFACE DE LIGNE DE COMMANDE (CLI)			✓		
Step Functions Local			✓		
AWS Toolkit for Visual Studio Code	✓	✓			
AWS SAM	✓	✓			
AWS CloudFormation	✓	✓	✓		

> **Note**
>
> AWS CloudFormation et vous permettent AWS SAM également de télécharger vos définitions de machine à états sur Amazon S3 au format JSON ou YAML, et de fournir l'emplacement Amazon S3 de la définition dans le modèle. Cela peut améliorer la lisibilité de vos modèles lorsque la définition de votre machine à états est complexe. Pour plus d'informations, consultez la page AWS::StepFunctions::StateMachine S3Location.

Les exemples de AWS CloudFormation modèles suivants montrent comment vous pouvez fournir la même définition de machine à états en utilisant différents formats d'entrée.

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 et AWS SAM

Vous pouvez utiliser la AWS SAM CLI conjointement avec le dans le AWS Toolkit for Visual Studio Code cadre d'une expérience intégrée pour créer des machines d'état localement. Vous pouvez créer une application sans serveur avec AWS SAM, puis créer votre machine d'état dans l'IDE VS Code.

Vous pouvez ensuite valider, empaqueter et déployer vos ressources. Facultativement, vous pouvez également publier sur le AWS Serverless Application Repository.

> **Tip**
>
> Pour déployer un exemple d'application sans serveur qui démarre un flux de travail Step Functions AWS SAM à l'aide de votre Compte AWS, consultez le Module 11 - Deploy with AWS SAM de The AWS Step Functions Workshop.

Rubriques

Pourquoi utiliser Step Functions avec AWS SAM ?

Lorsque vous utilisez Step Functions avec, AWS SAM vous pouvez :

- Commencez à utiliser un AWS SAM exemple de modèle.
- Construire votre machine d'état dans votre application sans serveur.
- Utilisez la substitution de variables pour remplacer les ARN dans votre machine d'état au moment du déploiement.

  AWS CloudFormationsupports `DefinitionSubstitutions`qui vous permettent d'ajouter des références dynamiques dans la définition de votre flux de travail à une valeur que vous fournissez dans votre CloudFormation modèle. Vous pouvez ajouter des références dynamiques en ajoutant des substitutions à la définition de votre flux de travail à l'aide de la `${dollar_sign_brace}` notation. Vous devez également définir ces références dynamiques dans la `DefinitionSubstitutions` propriété de la StateMachine ressource de votre CloudFormation modèle. Ces substitutions sont remplacées par des valeurs réelles lors du processus de création de la CloudFormation pile. Pour plus d’informations, consultez DefinitionSubstitutions dans les AWS SAM modèles.
- Spécifiez le rôle de votre machine à états à l'aide AWS SAM de modèles de politiques.

- Lancez des exécutions par machine à états à l'aide d'API Gateway, d' EventBridge événements ou selon un calendrier défini dans votre AWS SAM modèle.

Step Functions : intégration avec la AWS SAM spécification

Vous pouvez utiliser les modèles AWS SAM de politique pour ajouter des autorisations à votre machine d'état. Avec ces autorisations, vous pouvez orchestrer les fonctions Lambda et AWS d'autres ressources pour créer des flux de travail complexes et robustes.

Intégration de Step Functions avec l’interface de ligne de commande SAM

Step Functions est intégré à la AWS SAM CLI. Utilisez cela pour développer rapidement une machine d'état dans votre application sans serveur.

Essayez le Créez une machine à états Step Functions à l'aideAWS SAM didacticiel pour apprendre à l'utiliser AWS SAM pour créer des machines à états.

Les fonctions AWS SAM CLI prises en charge incluent :

Commande de l'interface de ligne de commande (CLI)	Description
démarrage sam	Initialise une application sans serveur à l'aide d'un AWS SAM modèle. Peut être utilisée avec un modèle SAM pour Step Functions.
valider sam	Valide un AWS SAM modèle.
package sam	Regroupe une AWS SAM application. Il crée un fichier ZIP contenant votre code et vos dépendances, puis le télécharge sur Amazon S3. Elle renvoie ensuite une copie de votre modèle AWS SAM , en remplaçant les références vers des artefacts locaux par l'emplacement Amazon S3 où la commande a chargé les artefacts.
déploiement sam	Déploie une AWS SAM application.

Commande de l'interface de ligne de commande (CLI)	Description
publication sam	Publiez une AWS SAM application dans le AWS Serverless Application Repository. Cette commande prend un AWS SAM modèle empaqueté et publie l'application dans la région spécifiée.

> **Note**
>
> Lorsque vous utilisez le AWS SAM mode local, vous pouvez émuler Lambda et API Gateway localement. Cependant, vous ne pouvez pas émuler Step Functions localement à l'aide AWS SAM de.

DefinitionSubstitutions dans les AWS SAM modèles

Vous pouvez définir des machines à états à l'aide CloudFormation de modèles avecAWS SAM. Avec AWS SAM, vous pouvez définir la machine d'état en ligne dans le modèle ou dans un fichier séparé. Le AWS SAM modèle suivant inclut une machine à états qui simule un flux de travail de négociation d'actions. Cette machine à états invoque trois Lambda fonctions pour vérifier le prix d'une action et déterminer s'il faut acheter ou vendre l'action. Cette transaction est ensuite enregistrée dans un Amazon DynamoDB tableau. Les ARN des Lambda fonctions et du DynamoDB tableau du modèle suivant sont spécifiés à l'aide `DefinitionSubstitutions`de.

```
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
```

Le code suivant est la définition de la machine à états dans le fichier `stock_trader.asl.json` utilisé dans le Créez une machine à états Step Functions à l'aideAWS SAM didacticiel. Cette définition de machine à états contient plusieurs définitions `DefinitionSubstitutions` désignées par la notation. `${dollar_sign_brace}` Par exemple, au lieu de spécifier un ARN de Lambda fonction statique pour la `Check Stock Value` tâche, la substitution `${StockCheckerFunctionArn}` est utilisée. Cette substitution est définie dans la DefinitionSubstitutions propriété du modèle. `DefinitionSubstitutions`est une carte de paires clé-valeur pour la ressource State Machine. Dans`DefinitionSubstitutions`, ${StockCheckerFunctionArn} correspond à l'ARN de la `StockCheckerFunction` ressource à l'aide de la fonction CloudFormation intrinsèque `!GetAtt`. Lorsque vous déployez le AWS SAM modèle, les valeurs `DefinitionSubstitutions` contenues dans le modèle sont remplacées par les valeurs réelles.

```
{
    "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
        }
    }
}
```

Étapes suivantes

Pour en savoir plus sur l'utilisation de Step Functions, consultez les ressources suivantes : AWS SAM

- Suivez le Créez une machine à états Step Functions à l'aideAWS SAM didacticiel pour créer une machine à états avec AWS SAM.
- Spécifiez une AWS::Serverless::StateMachineressource.
- Recherchez des modèles de stratégie AWS SAM à utiliser.
- À utiliser AWS Toolkit for Visual Studio Codeavec Step Functions.
- Consultez la référence de la AWS SAM CLI pour en savoir plus sur les fonctionnalités disponibles dans AWS SAM.

Vous pouvez également concevoir et créer vos flux de travail dans l'infrastructure sous forme de code (IaC) à l'aide de générateurs visuels, tels que Workflow Studio inApplication Composer. Pour plus d'informations, consultez Utilisation de Workflow Studio dans Application Composer.

Utilisation de Workflow Studio dans Application Composer

Composeur d'applications AWSest un constructeur visuel qui vous aide à développer AWS SAM des AWS CloudFormation modèles à l'aide d'une interface graphique simple. AvecApplication Composer, vous concevez une architecture d'application en faisant glisser, en regroupant et Services AWS en connectant dans un canevas visuel. Application Composercrée ensuite un modèle d'infrastructure sous forme de code (IaC) à partir de votre conception que vous pouvez utiliser pour déployer votre application à l'aide de l'interface de ligne de AWS SAM commande (AWS

SAMCLI) ouCloudFormation. Pour en savoir plus sur Application Composer, consultez Présentation d'Application Composer.

Workflow Studio est disponible pour vous aider Application Composer à concevoir et à créer vos flux de travail. Workflow Studio in Application Composer fournit un environnement IaC visuel qui vous permet d'intégrer facilement des flux de travail dans vos applications sans serveur créées à l'aide d'outils IaC, tels que des CloudFormation modèles. Lorsque vous utilisez Workflow Studio dansApplication Composer, il connecte les différentes étapes du flux de travail aux AWS ressources et génère les configurations des ressources dans un AWS SAM modèle. Il ajoute également les IAM autorisations requises pour l'exécution de votre flux de travail. À l'aide de Workflow StudioApplication Composer, vous pouvez créer des prototypes de vos applications et les transformer en applications prêtes pour la production.

Lorsque vous utilisez Workflow Studio dansApplication Composer, vous pouvez basculer entre le Application Composer canevas et Workflow Studio.

Rubriques

- Utilisation de Workflow Studio Application Composer pour créer un flux de travail sans serveur
- Référencez dynamiquement les ressources à l'aide de substitutions de CloudFormation définitions dans Workflow Studio
- Connect les tâches d'intégration des services à des cartes de composants améliorées
- Importez des projets existants et synchronisez-les localement
- Fonctionnalités de Workflow Studio non disponibles dans Composeur d'applications AWS

Utilisation de Workflow Studio Application Composer pour créer un flux de travail sans serveur

1. Ouvrez la console Application Composer et choisissez Create project pour créer un projet.
2. Dans le champ de recherche de la palette Ressources, entrez **`state machine`**.
3. Faites glisser la ressource Step FunctionsState machine sur le canevas.
4. Choisissez Modifier dans Workflow Studio pour modifier la ressource de votre machine d'état.

   L'animation suivante montre comment passer au Workflow Studio pour modifier la définition de votre machine à états.
   Animation illustrant la manière dont vous pouvez utiliser Workflow Studio dansApplication Composer.

L'intégration avec Workflow Studio pour modifier les ressources des machines d'état créées dans n'Application Composerest disponible que pour les `AWS::Serverless::StateMachine`ressources. Cette intégration n'est pas disponible pour les modèles qui utilisent la `AWS::StepFunctions::StateMachine`ressource.

Référencez dynamiquement les ressources à l'aide de substitutions de CloudFormation définitions dans Workflow Studio

Dans Workflow Studio, vous pouvez utiliser des substitutions de CloudFormation définitions dans votre définition de flux de travail pour référencer dynamiquement les ressources que vous avez définies dans votre modèle iAC. Vous pouvez ajouter des substitutions d'espaces réservés à la définition de votre flux de travail à l'aide de la `${dollar_sign_brace}` notation et elles sont remplacées par des valeurs réelles lors du processus de création de la CloudFormation pile. Pour plus d'informations sur les substitutions de définitions, consultezDefinitionSubstitutions dans les AWS SAM modèles.

L'animation suivante montre comment ajouter des substitutions d'espaces réservés pour les ressources dans votre définition de machine à états.
Animation qui montre comment référencer dynamiquement des ressources, telles que des AWS Lambda fonctions ou des substitutions de définitions lorsque vous utilisez Workflow Studio dansApplication Composer.

Connect les tâches d'intégration des services à des cartes de composants améliorées

Vous pouvez connecter les tâches qui font appel à des intégrations de services optimisées à des cartes de composants améliorées dans Application Composer Canvas. Cela permet de mapper automatiquement toutes les substitutions d'espaces réservés spécifiées par la `${dollar_sign_brace}` notation de la définition de votre flux de travail et par la `DefinitionSubstitution` propriété de votre `StateMachine` ressource. Il ajoute également les AWS SAM politiques appropriées pour la machine d'État.

Si vous mappez des tâches d'intégration de services optimisées avec des cartes de composants standard, la ligne de connexion n'apparaît pas sur le Application Composer canevas.

L'animation suivante montre comment connecter une tâche optimisée à une carte de composant améliorée et afficher les modifications dans Change Inspector.

Animation qui montre comment connecter des tâches faisant appel à des intégrations de services optimisées à des cartes de composants améliorées lorsque vous utilisez Workflow Studio dansApplication Composer.

Vous ne pouvez pas connecter les intégrations de AWS SDK dans votre état de tâche avec des cartes de composants améliorées ou des intégrations de services optimisées avec des cartes de composants standard. Pour ces tâches, vous pouvez mapper les substitutions dans le panneau des propriétés des ressources du Application Composer canevas et ajouter des politiques dans le AWS SAM modèle.

> **Tip**
>
> Vous pouvez également mapper les substitutions d'espaces réservés pour votre machine à états sous Substitutions de définitions dans le panneau des propriétés des ressources. Dans ce cas, vous devez ajouter les autorisations requises pour les appels d'état de Service AWS votre tâche dans le rôle d'exécution de la machine à états. Pour plus d'informations sur les autorisations dont votre rôle d'exécution peut avoir besoin, consultezRôles d'exécution dans Workflow Studio.

L'animation suivante montre comment mettre à jour manuellement le mappage de substitution des espaces réservés dans le panneau des propriétés des ressources.
Animation illustrant comment mettre à jour manuellement le mappage de substitution des espaces réservés dans le panneau des propriétés des ressources lorsque vous utilisez Workflow Studio dansApplication Composer.

Importez des projets existants et synchronisez-les localement

Vous pouvez ouvrir des fichiers existants CloudFormation et AWS SAM des projets pour les visualiser Application Composer afin de mieux comprendre et de modifier leurs conceptions. Grâce à Application Composer la fonction de synchronisation locale, vous pouvez automatiquement synchroniser et enregistrer vos modèles et fichiers de code sur votre machine de génération locale. L'utilisation du mode de synchronisation local peut compléter vos flux de développement existants. Assurez-vous que votre navigateur prend en charge l'API d'accès au système de fichiers, qui permet aux applications Web de lire, d'écrire et d'enregistrer des fichiers dans votre système de fichiers local. Nous vous recommandons d’utiliser Google Chrome ou Microsoft Edge.

Fonctionnalités de Workflow Studio non disponibles dans Composeur d'applications AWS

Lorsque vous utilisez Workflow Studio dansApplication Composer, certaines fonctionnalités de Workflow Studio ne sont pas disponibles. En outre, la section Paramètres de l'API disponible dans le Inspector panneau prend en charge les substitutions de CloudFormation définitions. Vous pouvez ajouter les substitutions à l'Mode codeaide de la `${dollar_sign_brace}` notation. Pour plus d'informations sur cette notation, consultezDefinitionSubstitutions dans les AWS SAM modèles.

La liste suivante décrit les fonctionnalités de Workflow Studio qui ne sont pas disponibles lorsque vous utilisez Workflow Studio dans Application Composer :

- Modèles de démarrage — Les modèles de démarrage sont des ready-to-run exemples de projets qui créent automatiquement les prototypes et les définitions du flux de travail. Ces modèles déploient toutes les AWS ressources connexes dont votre projet a besoin sur votre Compte AWS compte.
- Mode Config — Ce mode vous permet de gérer la configuration de vos machines d'état. Vous pouvez mettre à jour les configurations de vos machines à états dans vos modèles iAc ou utiliser le panneau des propriétés des ressources dans Application Composer Canvas. Pour plus d'informations sur la mise à jour des configurations dans le panneau des propriétés des ressources, consultezConnect les tâches d'intégration des services à des cartes de composants améliorées.
- API TestState
- Possibilité d'importer ou d'exporter des définitions de flux de travail à partir du bouton déroulant Actions de Workflow Studio. Application ComposerDans le menu, sélectionnez plutôt Ouvrir > Dossier du projet. Assurez-vous d'avoir activé le mode de synchronisation local pour enregistrer automatiquement vos modifications dans le Application Composer canevas directement sur votre machine locale.
- Bouton Exécuter. Lorsque vous utilisez Workflow Studio dansApplication Composer, Application Composer génère le code iAc pour votre flux de travail. Par conséquent, vous devez d'abord déployer le modèle. Exécutez ensuite le flux de travail dans la console ou via le AWS Command Line Interface(AWS CLI).

Création d'une machine à états Lambda pour Step Functions à l'aide de AWS CloudFormation

Ce didacticiel explique comment créer une AWS Lambda fonction de base à l'aide de AWS CloudFormation. Vous allez utiliser la AWS CloudFormation console et un modèle YAML pour créer la pile (rôles IAM, fonction Lambda et machine à états). Ensuite, vous utiliserez la AWS Step Functions console pour démarrer l'exécution de la machine à états.

Pour plus d'informations, consultez la section Utilisation des CloudFormation modèles et la `AWS::StepFunctions::StateMachine` ressource du Guide de AWS CloudFormation l'utilisateur.

Rubriques

Étape 1 : Configurez votre AWS CloudFormation modèle

Avant d'utiliser les exemples de modèle, vous devez comprendre comment déclarer les différentes parties d'un modèle AWS CloudFormation .

Rubriques

Pour créer un rôle IAM pour Lambda

Définissez la politique de confiance associée au rôle IAM pour la fonction Lambda. Les exemples suivants définissent une politique de confiance à l'aide de YAML ou de JSON.

YAML

```
LambdaExecutionRole:
  Type: "AWS::IAM::Role"
```

```
Properties:
  AssumeRolePolicyDocument:
    Version: "2012-10-17"
    Statement:
      - Effect: Allow
        Principal:
          Service: lambda.amazonaws.com
        Action: "sts:AssumeRole"
```

JSON

```
        "LambdaExecutionRole": {
          "Type": "AWS::IAM::Role",
          "Properties": {
              "AssumeRolePolicyDocument": {
                  "Version": "2012-10-17",
                  "Statement": [
                      {
                          "Effect": "Allow",
                          "Principal": {
                              "Service": "lambda.amazonaws.com"
                          },
                          "Action": "sts:AssumeRole"
                      }
                  ]
              }
          }
```

Pour créer une fonction Lambda

Définissez les propriétés suivantes pour une fonction Lambda qui imprimera le message. `Hello World`

> **Important**
>
> Assurez-vous que votre fonction Lambda se trouve sous le même AWS compte et dans la même AWS région que votre machine à états.

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"
            }
        },
```

Pour créer un rôle IAM pour l'exécution de la machine à états

Définissez la politique de confiance associée au rôle IAM pour l'exécution de la machine à états.

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": "*"
                            }
                        ]
                    }
                }
            ]
        }
    },
```

Pour créer une machine à états Lambda

Définissez la machine à états Lambda.

YAML

```
MyStateMachine:
  Type: "AWS::StepFunctions::StateMachine"
  Properties:
    DefinitionString:
      !Sub
        - |-
          {
            "Comment": "A Hello World example using an AWS Lambda function",
            "StartAt": "HelloWorld",
            "States": {
              "HelloWorld": {
                "Type": "Task",
```

```
                "Resource": "${lambdaArn}",
                "End": true
              }
            }
          }
        - {lambdaArn: !GetAtt [ MyLambdaFunction, Arn ]}
    RoleArn: !GetAtt [ StatesExecutionRole, Arn ]
```

JSON

```
        "MyStateMachine": {
            "Type": "AWS::StepFunctions::StateMachine",
            "Properties": {
                "DefinitionString": {
                    "Fn::Sub": [
                        "{\n  \"Comment\": \"A Hello World example using an
 AWS Lambda function\",\n  \"StartAt\": \"HelloWorld\",\n  \"States\": {\n
 \"HelloWorld\": {\n      \"Type\": \"Task\",\n      \"Resource\": \"${lambdaArn}\",
\n      \"End\": true\n    }\n  }\n}",
                        {
                            "lambdaArn": {
                                "Fn::GetAtt": [
                                    "MyLambdaFunction",
                                    "Arn"
                                ]
                            }
                        }
                    ]
                },
                "RoleArn": {
                    "Fn::GetAtt": [
                        "StatesExecutionRole",
                        "Arn"
                    ]
                }
            }
        }
```

Étape 2 : utiliser le AWS CloudFormation modèle pour créer une machine à états Lambda

Une fois que vous avez compris les composants du AWS CloudFormation modèle, vous pouvez les assembler et utiliser le modèle pour créer une AWS CloudFormation pile.

Pour créer la machine à états Lambda

1. Copiez l'exemple suivant dans un fichier nommé `MyStateMachine.yaml` pour l'exemple YAML ou `MyStateMachine.json` pour 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

```
{
```

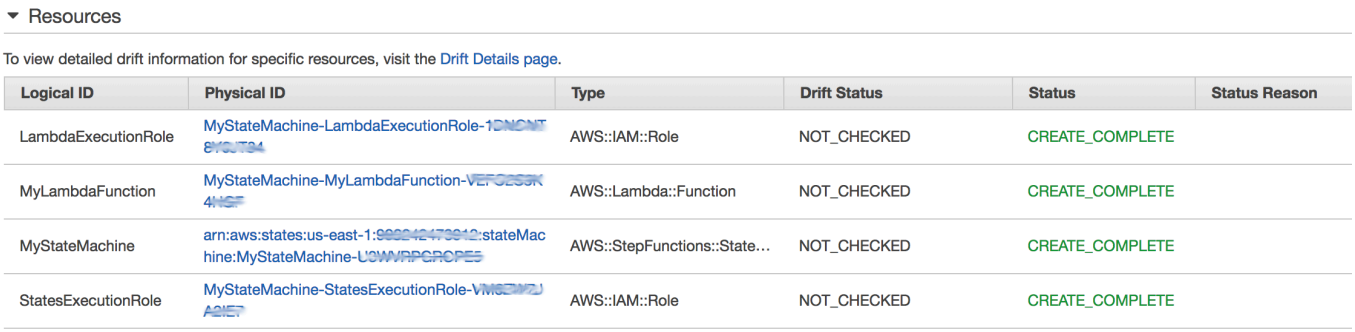
```
   "AWSTemplateFormatVersion": "2010-09-09",
   "Description": "An example template with an IAM role for a Lambda state
machine.",
   "Resources": {
       "LambdaExecutionRole": {
           "Type": "AWS::IAM::Role",
           "Properties": {
               "AssumeRolePolicyDocument": {
                   "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. Ouvrez la console AWS CloudFormation et choisissez Créer une pile.
3. Sur la page Sélectionner un modèle, choisissez Télécharger un modèle sur Amazon S3. Choisissez votre fichier `MyStateMachine`, puis choisissez Suivant.
4. Sur la page Spécifier les détails, pour Nom de la pile, tapez `MyStateMachine`, puis choisissez Suivant.
5. Dans la page Options, choisissez Suivant.
6. Sur la page de révision, choisissez Je reconnais que cela AWS CloudFormation pourrait créer des ressources IAM. puis choisissez Create.

   AWS CloudFormation commence à créer la `MyStateMachine` pile et affiche le statut CREATE_IN_PROGRESS. Lorsque le processus est terminé, AWS CloudFormation affiche l'état CREATE_COMPLETE.
7. (Facultatif) Pour afficher les ressources de votre pile, sélectionnez la pile et choisissez l'onglet Ressources.



▼ Resources

To view detailed drift information for specific resources, visit the Drift Details page.

Logical ID	Physical ID	Type	Drift Status	Status	Status Reason
LambdaExecutionRole	MyStateMachine-LambdaExecutionRole-1[redacted]	AWS::IAM::Role	NOT_CHECKED	CREATE_COMPLETE	
MyLambdaFunction	MyStateMachine-MyLambdaFunction-[redacted]	AWS::Lambda::Function	NOT_CHECKED	CREATE_COMPLETE	
MyStateMachine	arn:aws:states:us-east-1:[redacted]:stateMachine:MyStateMachine-[redacted]	AWS::StepFunctions::State…	NOT_CHECKED	CREATE_COMPLETE	
StatesExecutionRole	MyStateMachine-StatesExecutionRole-[redacted]	AWS::IAM::Role	NOT_CHECKED	CREATE_COMPLETE	

Étape 3 : démarrer une exécution de State Machine

Après avoir créé votre machine d'état Lambda, vous pouvez commencer son exécution.

Pour démarrer l'exécution de la machine d'état

1. Ouvrez la console Step Functions et choisissez le nom de la machine à états que vous avez créée avec AWS CloudFormation.
2. Sur la page ***MyStateMachine-ABCDEFGHIJ1K***, choisissez Nouvelle exécution.

   La page Nouvelle exécution s'affiche.
3. (Facultatif) Pour identifier votre exécution, vous pouvez lui donner un nom dans le champ Nom. Par défaut, Step Functions génère automatiquement un nom d'exécution unique.

   > **Note**
   >
   > Step Functions vous permet de créer des noms pour les machines d'état, les exécutions et les activités, ainsi que des étiquettes contenant des caractères non ASCII. Ces noms non ASCII ne fonctionnent pas avec Amazon. CloudWatch Pour être sûr de pouvoir suivre CloudWatch les métriques, choisissez un nom qui utilise uniquement des caractères ASCII.
4. Choisissez Démarrer une exécution.

   Une nouvelle exécution de votre machine d'état commence et une nouvelle page indiquant votre exécution en cours s'affiche.
5. (Facultatif) Dans la section Execution Details (Détails de l'exécution), choisissez Execution Status (Statut de l'exécution) et les horodatages Started (Démarré) et Closed (Fermé).

6. Pour afficher les résultats de votre exécution, choisissez Sortie.

Création d'une machine à Lambda états pour Step Functions utiliser AWS CDK

Ce didacticiel explique comment créer une machine à AWS Step Functions états contenant une AWS Lambda fonction à l'aide duAWS Cloud Development Kit (AWS CDK). AWS CDKIl s'agit d'un framework d'infrastructure en tant que code (IAC) qui vous permet de définir une AWS infrastructure à l'aide d'un langage de programmation complet. Vous pouvez écrire une application dans l'une des CDK langues prises en charge contenant une ou plusieurs piles. Vous pouvez ensuite le synthétiser dans un AWS CloudFormation modèle et le déployer sur votre AWS compte. Nous utiliserons cette méthode pour définir une machine à Step Functions états contenant une Lambda fonction, puis utiliserons la AWS Management Console pour exécuter la machine à états.

Avant de commencer ce didacticiel, vous devez configurer votre environnement de AWS CDK développement comme décrit dans la section Getting Started With the AWS CDK - Preuisites du manuel du AWS Cloud Development Kit (AWS CDK) développeur. Ensuite, installez le AWS CDK à l'aide de la commande suivante sur AWS CLI :

```
npm install -g aws-cdk
```

Ce didacticiel produit le même résultat quethe section called “Création d'une machine à états Lambda à l'aide de AWS CloudFormation”. Cependant, dans ce didacticiel, vous AWS CDK n'êtes pas obligé de créer des IAM rôles ; il le AWS CDK fait pour vous. La AWS CDK version inclut également une Succeed étape illustrant comment ajouter des étapes supplémentaires à votre machine d'état.

> **Tip**
>
> Pour déployer un exemple d'application sans serveur qui démarre un Step Functions flux de travail en utilisant AWS CDK with TypeScript pour votre Compte AWS, consultez le module 10 - Déployer avec AWS CDK de The AWS Step Functions Workshop.

Rubriques

- Étape 1 : Configurer votre projet AWS CDK
- Étape 2 : utilisation AWS CDK pour créer une machine à états

Étape 1 : Configurer votre projet AWS CDK

1. Dans votre répertoire personnel, ou dans un autre répertoire si vous préférez, exécutez la commande suivante pour créer un répertoire pour votre nouvelle AWS CDK application.

   > **Important**
   >
   > Assurez-vous de donner un nom au répertoire`step`. Le modèle AWS CDK d'application utilise le nom du répertoire pour générer des noms pour les classes et les fichiers sources. Si vous utilisez un nom différent, votre appli ne correspondra pas à ce didacticiel.

   TypeScript

   ```
   mkdir step && cd step
   ```

   JavaScript

   ```
   mkdir step && cd step
   ```

   Python

   ```
   mkdir step && cd step
   ```

   Java

   ```
   mkdir step && cd step
   ```

   C#

   Assurez-vous d'avoir installé .NET version 6.0 ou supérieure. Pour plus d'informations, consultez la section Versions prises en charge.

```
mkdir step && cd step
```

2. Initialisez l'application à l'aide de la commande cdk init. Spécifiez le modèle (« app ») et le langage de programmation souhaités, comme indiqué dans les exemples suivants.

   TypeScript

   ```
   cdk init --language typescript
   ```

   JavaScript

   ```
   cdk init --language javascript
   ```

   Python

   ```
   cdk init --language python
   ```

   Une fois le projet initialisé, activez l'environnement virtuel du projet et installez ses dépendances AWS CDK de base.

   ```
   source .venv/bin/activate
   python -m pip install -r requirements.txt
   ```

   Java

   ```
   cdk init --language java
   ```

   C#

   ```
   cdk init --language csharp
   ```

Étape 2 : utilisation AWS CDK pour créer une machine à états

Nous allons d'abord présenter les différents éléments de code qui définissent la Lambda fonction et la machine à Step Functions états. Ensuite, nous expliquerons comment les assembler dans votre AWS CDK application. Enfin, vous allez voir comment synthétiser et déployer ces ressources.

Pour créer une fonction Lambda

Le AWS CDK code suivant définit la Lambda fonction en fournissant son code source en ligne.

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)
});
```

Vous pouvez voir dans ce court exemple de code :

- Le nom logique de la fonction,`MyLambdaFunction`.
- Le code source de la fonction, intégré sous forme de chaîne dans le code source de l'AWS CDKapplication.
- Autres attributs de fonction, tels que le temps d'exécution à utiliser (Node 18.x), le point d'entrée de la fonction et un délai d'attente.

Pour créer une machine d'état

Notre machine à états possède deux états : une tâche Lambda fonctionnelle et un Succeed état. La fonction nécessite que nous créions un Step Functions the section called “Tâche” qui invoque notre fonction. Cet état de tâche est utilisé comme première étape dans la machine à états. L'état de réussite est ajouté à la machine d'état à l'aide de la `next()` méthode de l'état des tâches. Le code

suivant appelle d'abord la fonction nomméeMyLambdaTask, puis utilise la next() méthode pour définir un état de réussite nomméGreetedWorld.

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")))
});
```

Pour créer et déployer l'AWS CDKapplication

Dans le AWS CDK projet que vous venez de créer, modifiez le fichier contenant la définition de la pile pour qu'il ressemble à l'exemple de code suivant. Vous reconnaîtrez les définitions de la Lambda fonction et de la machine à Step Functions états figurant dans les sections précédentes.

1. Mettez à jour la pile comme indiqué dans les exemples suivants.

   TypeScript

   Effectuez la mise à jour `lib/step-stack.ts` avec le code suivant.

```
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

Effectuez la mise à jour `lib/step-stack.js` avec le code suivant.

```
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

Effectuez la mise à jour `step/step_stack.py` avec le code suivant.

```
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

Effectuez la mise à jour `src/main/java/com.myorg/StepStack.java` avec le code suivant.

```
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#

Effectuez la mise à jour `scr/Step/StepStack.cs` avec le code suivant.

```
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. Enregistrez le fichier source, puis exécutez la `cdk synth` commande dans le répertoire principal de l'application.

   AWS CDKexécute l'application et synthétise un AWS CloudFormation modèle à partir de celle-ci. AWS CDKaffiche ensuite le modèle.

   > **Note**
   >
   > Si vous avez TypeScript créé votre AWS CDK projet, l'exécution de la `cdk synth` commande peut renvoyer l'erreur suivante.

```
TSError: # Unable to compile TypeScript:
bin/step.ts:7:33 - error TS2554: Expected 2 arguments, but got 3.
```

Modifiez le `bin/step.ts` fichier comme indiqué dans l'exemple suivant pour résoudre cette erreur.

```
#!/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. Pour déployer la fonction Lambda et la machine d'état Step Functions sur votre AWS compte, lancez. `cdk deploy` Il vous sera demandé d'approuver les politiques IAM générées. AWS CDK

Étape 3 : démarrer l'exécution d'une machine à états

Après avoir créé votre machine d'état, vous pouvez commencer son exécution.

Pour démarrer l'exécution de la machine d'état

1. Ouvrez la console Step Functions et choisissez le nom de la machine à états que vous avez créée avecAWS CDK.
2. Sur la page State Machine, choisissez Démarrer l'exécution.

   La boîte de dialogue Démarrer l'exécution s'affiche.
3. (Facultatif) Pour identifier votre exécution, vous pouvez lui donner un nom dans le champ Nom. Par défaut, Step Functions génère automatiquement un nom d'exécution unique.

> **Note**
>
> Step Functions vous permet de créer des noms pour les machines d'état, les exécutions et les activités, ainsi que des étiquettes contenant des caractères non ASCII. Ces noms non ASCII ne fonctionnent pas avec Amazon. CloudWatch Pour être sûr de

pouvoir suivre CloudWatch les métriques, choisissez un nom qui utilise uniquement des caractères ASCII.

4. Choisissez Démarrer une exécution.

   L'exécution de votre machine d'état démarre et une nouvelle page indiquant votre exécution en cours s'affiche.

5. La console Step Functions vous dirige vers une page intitulée avec votre ID d'exécution. Cette page est connue sous le nom de page Détails de l'exécution. Sur cette page, vous pouvez consulter les résultats de l'exécution au fur et à mesure que l'exécution progresse ou une fois celle-ci terminée.

   Pour consulter les résultats de l'exécution, choisissez des états individuels dans la vue graphique, puis choisissez les onglets individuels du Détails de l'étape volet pour afficher les détails de chaque état, y compris les entrées, les sorties et la définition respectivement. Pour plus de détails sur les informations d'exécution que vous pouvez consulter sur la page Détails de l'exécution, voirPage de détails d'exécution — Vue d'ensemble de l'interface.

Étape 4 : nettoyage

Après avoir testé votre machine à états, nous vous recommandons de supprimer à la fois votre machine d'état et la fonction Lambda associée afin de libérer des ressources dans votre. Compte AWS Exécutez la `cdk destroy` commande dans le répertoire principal de votre application pour supprimer votre machine d'état.

Étapes suivantes

Pour en savoir plus sur le développement AWS d'une infrastructure à l'aideAWS CDK, consultez le guide du AWS CDK développeur.

Pour plus d'informations sur l'écriture d'applis AWS CDK dans la langue de votre choix, veuillez consulter :

TypeScript

- Travailler avec AWS CDK in TypeScript

JavaScript

- Travailler avec AWS CDK in JavaScript

Python

Travailler avec AWS CDK en Python

Java

Utilisation AWS CDK en Java

C#

Travailler avec AWS CDK en C#

Pour plus d'informations sur les modules AWS Construct Library utilisés dans ce didacticiel, consultez les aperçus des références AWS CDK d'API suivants :

- aws-lambda
- fonctions aws-step
- aws-stepfunctions-tasks

Création d'une API REST API Gateway avec une machine synchrone Express State à l'aide du AWS CDK

Ce didacticiel explique comment créer une API REST API Gateway avec Synchronous Express State Machine comme intégration principale à l'aide du. AWS Cloud Development Kit (AWS CDK) Ce didacticiel utilisera la `StepFunctionsRestApi` construction pour connecter la State Machine à l'API Gateway. La `StepFunctionsRestApi` construction configurera un mappage d'entrée/sortie par défaut et l'API REST API Gateway, avec les autorisations requises et une méthode HTTP « ANY ». AWS CDK Il s'agit d'un framework d'infrastructure en tant que code (IAC) qui vous permet de définir une AWS infrastructure à l'aide d'un langage de programmation complet. Vous écrivez une application dans l'une des langues prises en charge par le CDK, contenant une ou plusieurs piles, puis vous la synthétisez dans un AWS CloudFormation modèle et vous la déployez sur votre compte. AWS Nous l'utiliserons pour définir une API REST API Gateway, qui est intégrée à Synchronous Express State Machine en tant que backend, puis nous l'utiliserons AWS Management Console pour lancer l'exécution.

Avant de commencer ce didacticiel, configurez votre environnement de AWS CDK développement comme décrit dans Getting Started With the AWS CDK - Prerequisites, puis installez-le AWS CDK en émettant :

```
npm install -g aws-cdk
```

Rubriques

Étape 1 : Configurez votre AWS CDK projet

Créez d'abord un répertoire pour votre nouvelle AWS CDK application et initialisez le projet.

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
```

Une fois le projet initialisé, activez l'environnement virtuel du projet et installez ses dépendances AWS CDK de base.

```
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**
>
> Assurez-vous de donner un nom au répertoire`stepfunctions-rest-api`. Le modèle AWS CDK d'application utilise le nom du répertoire pour générer des noms pour les classes et les fichiers sources. Si vous utilisez un nom différent, votre appli ne correspondra pas à ce didacticiel.

Installez maintenant les modules de bibliothèque de construction pour AWS Step Functions Amazon API Gateway.

TypeScript

```
npm install @aws-cdk/aws-stepfunctions @aws-cdk/aws-apigateway
```

JavaScript

```
npm install @aws-cdk/aws-stepfunctions @aws-cdk/aws-apigateway
```

Python

```
python -m pip install aws-cdk.aws-stepfunctions
python -m pip install aws-cdk.aws-apigateway
```

Java

Modifiez le `pom.xml` du projet pour ajouter les dépendances suivantes dans le conteneur `<dependencies>` existant.

```
        <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 installe automatiquement ces dépendances lors de la prochaine création de votre appli. Pour créer, émettez `mvn compile` ou utilisez la commande Build de votre IDE Java.

C#

```
dotnet add src/StepfunctionsRestApi package Amazon.CDK.AWS.Stepfunctions
dotnet add src/StepfunctionsRestApi package Amazon.CDK.AWS.APIGateway
```

Vous pouvez également installer les packages indiqués à l'aide de l' NuGetinterface graphique de Visual Studio, disponible via Outils > Gestionnaire de NuGet packages > Gérer les NuGet packages pour la solution.

Une fois les modules installés, vous pouvez les utiliser dans votre AWS CDK application en important les packages suivants.

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

Ajoutez ce qui suit à `import` l'intérieur`stepfunctions-rest-api.go`.

```
"github.com/aws/aws-cdk-go/awscdk/awsapigateway"
"github.com/aws/aws-cdk-go/awscdk/awsstepfunctions"
```

Étape 2 : utilisez le AWS CDK pour créer une API REST API Gateway avec intégration du backend Synchrous Express State Machine

Nous allons d'abord présenter les différents éléments de code qui définissent la Synchronous Express State Machine et l'API REST API Gateway, puis nous expliquerons comment les intégrer dans votre AWS CDK application. Vous verrez ensuite comment synthétiser et déployer ces ressources.

> **Note**
>
> La machine à états que nous allons montrer ici sera une simple machine à états avec un `Pass` état.

Pour créer une machine Express State

C'est le AWS CDK code qui définit une machine à états simple avec un `Pass` état.

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,
})
```

Vous pouvez voir les éléments suivants dans ce court extrait :

- La définition de machine nomméePassState, qui est un Pass État.
- Le nom logique de State Machine,MyStateMachine.
- La définition de machine est utilisée comme définition de State Machine.
- Le type de machine à états est défini comme suit : EXPRESS il n'StepFunctionsRestApiautorisera qu'une machine à état synchrone Express.

Pour créer l'API REST API Gateway à l'aide de **StepFunctionsRestApi** construct

Nous utiliserons StepFunctionsRestApi construct pour créer l'API REST API Gateway avec les autorisations requises et le mappage des entrées/sorties par défaut.

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,
})
```

Pour créer et déployer l' AWS CDK application

Dans le AWS CDK projet que vous avez créé, modifiez le fichier contenant la définition de la pile pour qu'il ressemble au code ci-dessous. Vous reconnaîtrez les définitions de la machine à états Step Functions et de l'API Gateway ci-dessus.

TypeScript

Mettez à jour `lib/stepfunctions-rest-api-stack.ts` comme suit.

```
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

Mettez à jour `lib/stepfunctions-rest-api-stack.js` comme suit.

```
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

Mettez à jour `stepfunctions_rest_api/stepfunctions_rest_api_stack.py` comme suit.

```
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

Mettez à jour `src/main/java/com.myorg/StepfunctionsRestApiStack.java` comme suit.

```
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#

Mettez à jour `src/StepfunctionsRestApi/StepfunctionsRestApiStack.cs` comme suit.

```
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

Mettez à jour `stepfunctions-rest-api.go` comme suit.

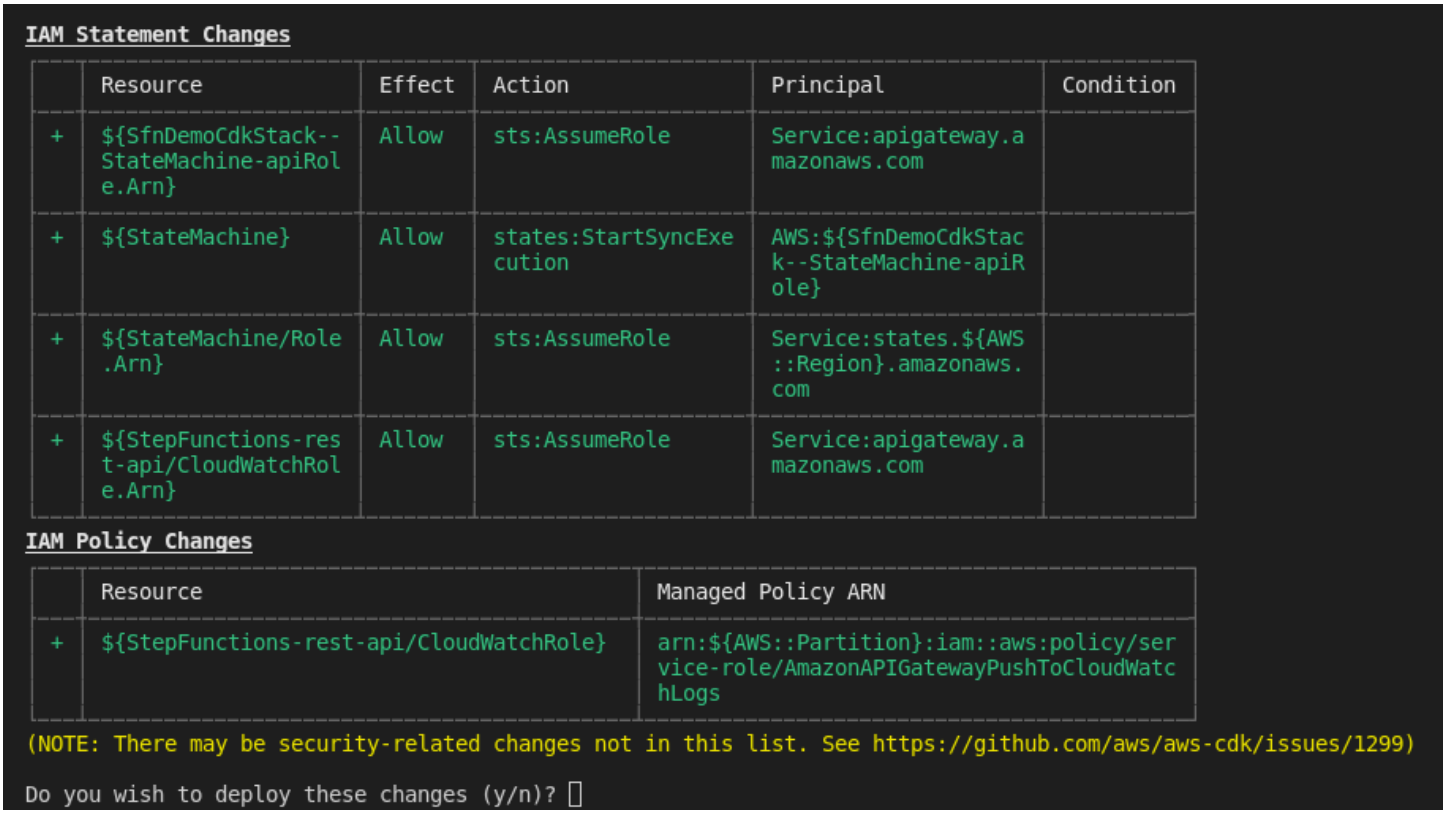
```
package main
import (
    "github.com/aws/aws-cdk-go/awscdk"
    "github.com/aws/aws-cdk-go/awscdk/awsapigateway"
    "github.com/aws/aws-cdk-go/awscdk/awsstepfunctions"
    "github.com/aws/constructs-go/constructs/v3"
    "github.com/aws/jsii-runtime-go"
)


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")),
    // }
}
```

Enregistrez le fichier source, puis émettez `cdk synth` dans le répertoire principal de l'application. AWS CDK Exécute l'application et synthétise un AWS CloudFormation modèle à partir de celle-ci, puis affiche le modèle.

Pour réellement déployer l'Amazon API Gateway et la machine AWS Step Functions d'état sur votre compte AWS, lancez`cdk deploy`. Il vous sera demandé d'approuver les politiques IAM générées. AWS CDK Les politiques créées ressembleront à ceci :

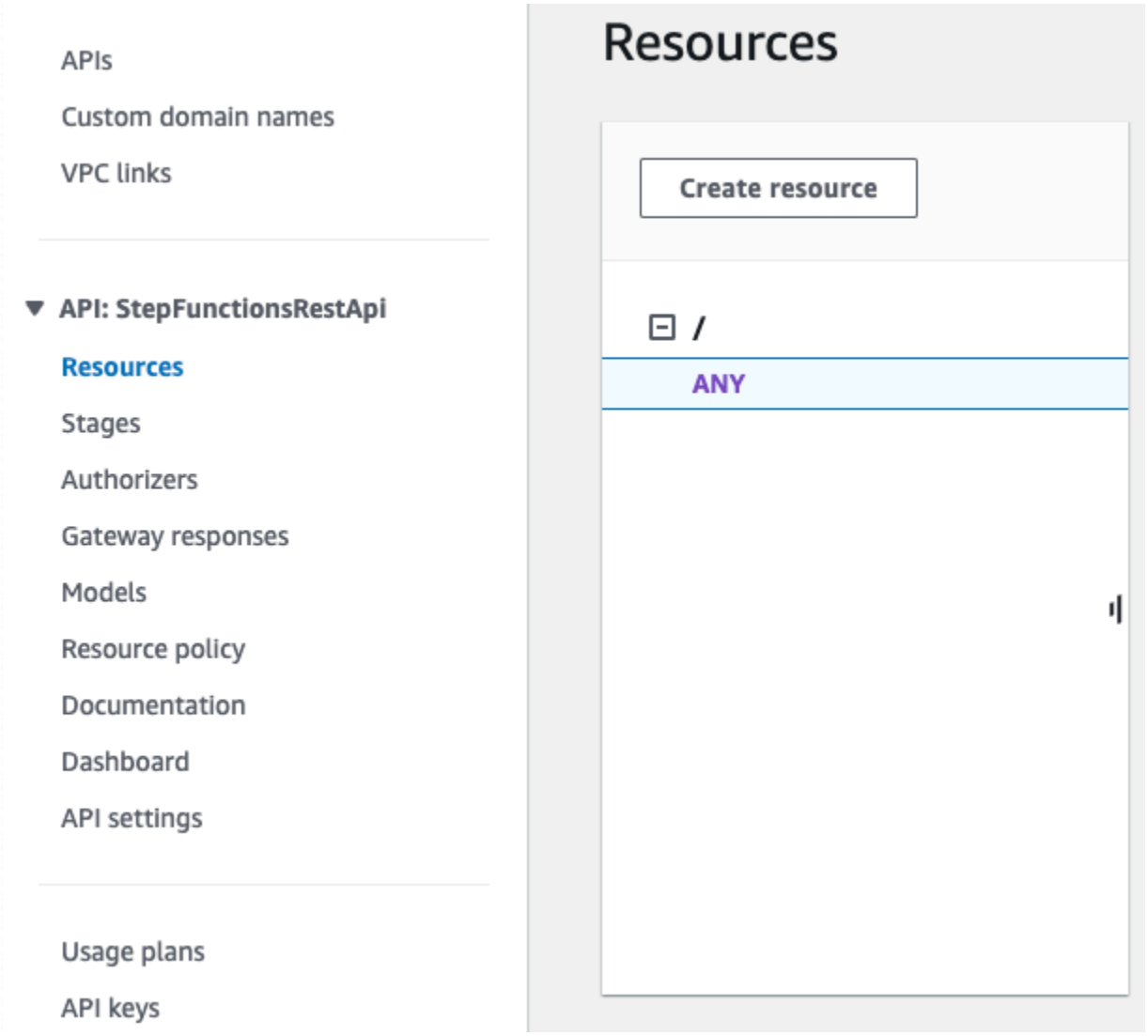


Étape 3 : tester l'API Gateway

Après avoir créé votre API REST API Gateway avec Synchronous Express State Machine comme intégration principale, vous pouvez tester l'API Gateway.

Pour tester l'API Gateway déployée à l'aide de la console API Gateway

1. Ouvrez la console Amazon API Gateway et connectez-vous.
2. Choisissez le nom de votre API REST`StepFunctionsRestApi`.
3. Dans le volet Ressources, choisissez la `ANY` méthode.



4. Choisissez l’onglet Test. Vous devrez peut-être choisir la flèche droite pour afficher l'onglet.
5. Pour Method (Méthode), choisissez POST.
6. Pour le corps de la demande, copiez les paramètres de demande suivants.

```
{
    "key": "Hello"
}
```

7. Sélectionnez Test. Les informations suivantes s'affichent alors :

   - Request est le chemin de la ressource qui a été appelée pour la méthode.
   - Status est le code d'état HTTP de la réponse.
   - Latency correspond au temps entre la réception de la demande de l'appelant et la réponse renvoyée.
   - Corps de la réponse correspond au corps de la réponse HTTP.
   - Le paramètre En-têtes de réponse correspond aux en-têtes de réponse HTTP.
   - Le journal affiche les entrées Amazon CloudWatch Logs simulées qui auraient été écrites si cette méthode avait été appelée en dehors de la console API Gateway.

   > **Note**
   >
   > Bien que les entrées CloudWatch Logs soient simulées, les résultats de l'appel de méthode sont réels.

La sortie du corps de la réponse devrait ressembler à ceci :

```
"Hello"
```

> **Tip**
>
> Essayez l'API Gateway avec différentes méthodes et une entrée non valide pour voir le résultat d'erreur. Vous souhaiterez peut-être modifier la machine d'état pour rechercher une clé particulière et, pendant les tests, fournir la mauvaise clé pour faire échouer l'exécution de la machine à états et générer un message d'erreur dans le corps de réponse en sortie.

Pour tester l'API déployée à l'aide de cURL

1. Ouvrez une fenêtre du terminal.

2. Copiez la commande cURL suivante et collez-la dans la fenêtre du terminal, en remplaçant `<api-id>` par l'ID de l'API et `<region>` par la région ou l'API est déployée.

```
curl -X POST\
 'https://<api-id>.execute-api.<region>.amazonaws.com/prod' \
 -d '{"key":"Hello"}' \
 -H 'Content-Type: application/json'
```

La sortie du Response Body devrait ressembler à ceci :

```
"Hello"
```

> **Tip**
>
> Essayez l'API Gateway avec différentes méthodes et une entrée non valide pour voir le résultat d'erreur. Vous souhaiterez peut-être modifier la machine d'état pour rechercher une clé particulière et, pendant les tests, fournir la mauvaise clé pour faire échouer l'exécution de la machine à états et générer un message d'erreur dans la sortie du corps de réponse.

Étape 4 : nettoyage

Lorsque vous avez terminé d'essayer votre API Gateway, vous pouvez démonter à la fois la machine d'état et l'API Gateway à l'aide du kit AWS CDK. Émettez `cdk destroy` dans le répertoire principal de votre appli.

AWS Step Functions SDK de science des données pour Python

Le AWS Step Functions Data Science SDK est une bibliothèque open source destinée aux data scientists. Avec ce SDK, vous pouvez créer des flux de travail qui traitent et publient des modèles d'apprentissage automatique à l'aide SageMaker de Step Functions. Vous pouvez également créer des flux de travail d'apprentissage automatique en plusieurs étapes en Python qui orchestrent AWS l'infrastructure à grande échelle, sans avoir à fournir et à intégrer les AWS services séparément.

Le SDK AWS Step Functions Data Science fournit une API Python qui permet de créer et d'invoquer des flux de travail Step Functions. Vous pouvez gérer et exécuter ces flux de travail directement dans Python, ainsi que dans les blocs-notes Jupyter.

En plus de créer des flux de travail prêts pour la production directement en Python, le SDK AWS Step Functions Data Science vous permet de copier ce flux de travail, d'expérimenter de nouvelles options, puis de mettre en production le flux de travail affiné.

Pour plus d'informations sur le SDK AWS Step Functions Data Science, consultez les rubriques suivantes :

- Projet sur Github
- Documentation des kits SDK
- Les exemples de blocs-notes suivants, disponibles dans les instances de blocs-notes Jupyter de la SageMaker console et du projet associé : GitHub
  - `hello_world_workflow.ipynb`
  - `machine_learning_workflow_abalone.ipynb`
  - `training_pipeline_pytorch_mnist.ipynb`

Déploiement de machines d'état à l'aide de Terraform

Terraform by HashiCorp est un framework permettant de créer des applications utilisant l'infrastructure en tant que code (IaC). Avec Terraform, vous pouvez créer des machines d'état et utiliser des fonctionnalités, telles que la prévisualisation des déploiements d'infrastructure et la création de modèles réutilisables. Les modèles Terraform vous aident à maintenir et à réutiliser le code en le décomposant en petits morceaux.

Si vous connaissez Terraform, vous pouvez suivre le cycle de développement décrit dans cette rubrique en tant que modèle pour créer et déployer vos machines d'état dans Terraform. Si vous n'êtes pas familier avec Terraform, nous vous recommandons de suivre d'abord l'atelier Introduction à Terraform AWS pour vous familiariser avec Terraform.

> **Tip**
>
> Pour déployer un exemple de machine à états créée à l'aide de Terraform sur votre ordinateurCompte AWS, consultez le module Gestion des machines à états avec infrastructure en tant que code de The AWS Step Functions Workshop.

Dans cette rubrique

Prérequis

Avant de commencer, assurez-vous de remplir les conditions préalables suivantes :

- Installez Terraform sur votre machine. Pour plus d'informations sur l'installation de Terraform, voir Installer Terraform.
- Installez Step Functions Local sur votre machine. Nous vous recommandons d'installer l'image Docker Step Functions Local pour utiliser Step Functions Local. Pour plus d'informations, veuillez consulter Tester les machines d'état localement.
- Installez la AWS SAM CLI. Pour plus d'informations sur l'installation, consultez la section Installation de la AWS SAM CLI dans le manuel du AWS Serverless Application Model développeur.
- Installez le AWS Toolkit for Visual Studio Code pour afficher le diagramme de flux de travail de vos machines d'état. Pour plus d'informations sur l'installation, consultez la section Installation du AWS Toolkit for Visual Studio Code dans le guide de AWS Toolkit for Visual Studio Code l'utilisateur.

Cycle de vie du développement des machines State avec Terraform

La procédure suivante explique comment utiliser un prototype de machine à états que vous créez à l'aide de Workflow Studio dans la console Step Functions comme point de départ pour le développement local avec Terraform et le. AWS Toolkit for Visual Studio Code

Pour consulter l'exemple complet qui traite du développement de machines à états avec Terraform et présente les meilleures pratiques en détail, voir Meilleures pratiques pour l'écriture de projets Step Functions Terraform.

Pour démarrer le cycle de développement d'une machine à états avec Terraform

1. Démarrez un nouveau projet Terraform avec la commande suivante.

   ```
   terraform init
   ```

2. Ouvrez la console Step Functions pour créer un prototype pour votre machine à états.

3. Dans Workflow Studio, procédez comme suit :

   a. Créez votre prototype de flux de travail.

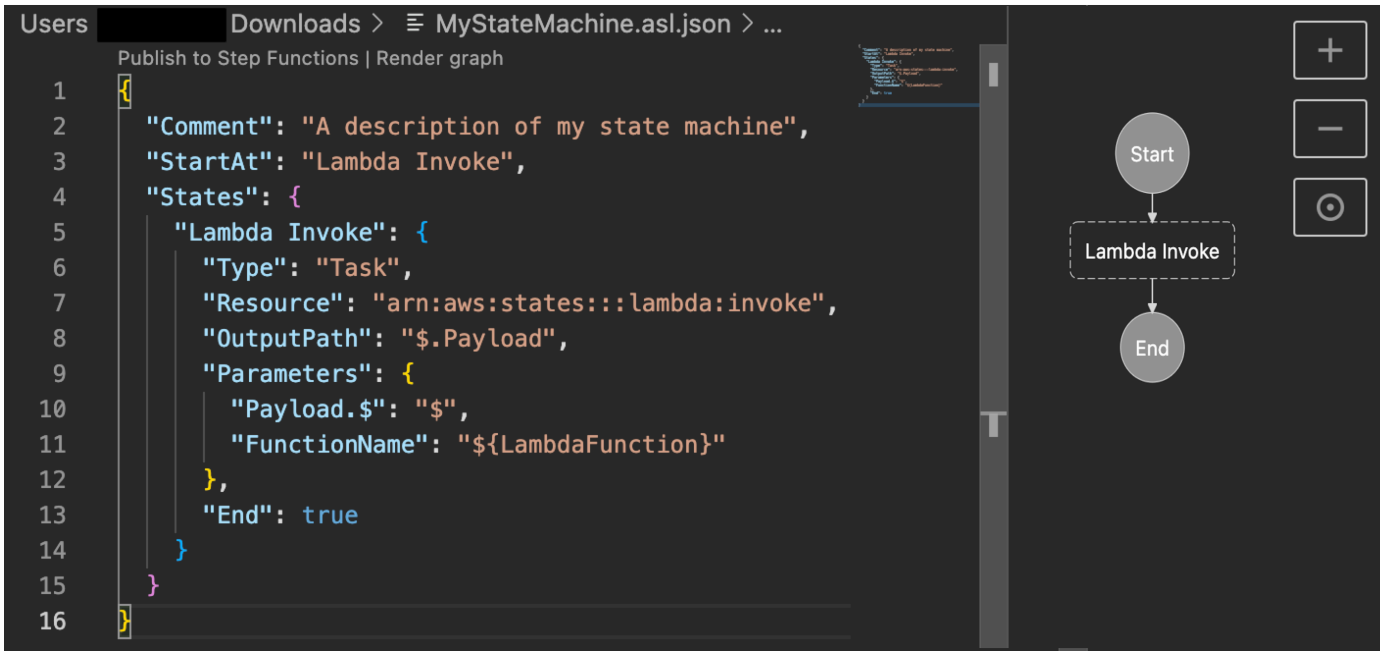
   b. Exportez la définition Amazon States Language (ASL) de votre flux de travail. Pour ce faire, choisissez la liste déroulante Import/Export, puis sélectionnez Exporter la définition JSON.

4. Enregistrez la définition ASL exportée dans le répertoire de votre projet.

   Vous transmettez la définition ASL exportée en tant que paramètre d'entrée à la ressource `aws_sfn_state_machine`Terraform qui utilise la fonction. `templatefile` Cette fonction est utilisée dans le champ de définition qui transmet la définition ASL exportée et toutes les substitutions de variables.

   > **Tip**
   >
   > Le fichier de définition ASL pouvant contenir de longs blocs de texte, nous vous recommandons d'éviter la méthode EOF intégrée. Cela facilite la substitution de paramètres dans la définition de votre machine à états.

5. (Facultatif) Mettez à jour la définition ASL dans votre IDE et visualisez vos modifications à l'aide duAWS Toolkit for Visual Studio Code.



   Pour éviter d'exporter continuellement votre définition et de la refactoriser dans votre projet, nous vous recommandons d'effectuer des mises à jour localement dans votre IDE et de suivre ces mises à jour avec Git.

6. Testez votre flux de travail à l'aide de Step Functions Local.

> **Tip**
>
> Vous pouvez également tester localement les intégrations de services avec les fonctions Lambda et les API API Gateway dans votre machine d'état à l'aide de AWS SAMCLI Local.

7. Prévisualisez votre machine d'état et d'autres AWS ressources avant de déployer la machine d'état. Pour ce faire, exécutez la commande suivante.

```
terraform plan
```

8. Déployez votre machine d'état depuis votre environnement local ou via des pipelines CI/CD à l'aide de la commande suivante.

```
terraform apply
```

9. (Facultatif) Nettoyez vos ressources et supprimez la machine d'état à l'aide de la commande suivante.

```
terraform destroy
```

Rôles et politiques IAM pour votre machine étatique

Utilisez les politiques d'intégration du service Terraform pour ajouter les autorisations IAM nécessaires à votre machine d'état, par exemple l'autorisation d'invoquer des fonctions Lambda. Vous pouvez également définir des rôles et des politiques explicites et les associer à votre machine d'état.

L'exemple de politique IAM suivant accorde à votre machine d'état l'accès pour appeler une fonction Lambda nommée. *myFunction*

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
```

```
        "lambda:InvokeFunction"
      ],
      "Resource": "arn:aws:lambda:us-east-1:123456789012:function:myFunction"
    }
  ]
}
```

Nous vous recommandons également d'utiliser la source de aws_iam_policy_documentdonnées lors de la définition des politiques IAM pour vos machines d'état dans Terraform. Cela vous permet de vérifier si votre politique est mal formée et de remplacer les ressources par des variables.

L'exemple de politique IAM suivant utilise la source de `aws_iam_policy_document` données et accorde à votre machine d'état l'accès pour appeler une fonction Lambda nommée. *`myFunction`*

```
data "aws_iam_policy_document" "state_machine_role_policy" {

  statement {
    effect = "Allow"

    actions = [
      "lambda:InvokeFunction"
    ]

    resources = ["${aws_lambda_function.[[myFunction]].arn}:*"]
  }

}
```

> **Tip**
>
> Pour voir des modèles AWS architecturaux plus avancés déployés avec Terraform, consultez les exemples de Terraform dans la collection Serverless Land Workflows.

Test et débogage

Step Functionsfournit différentes méthodes pour tester et déboguer les machines d'état. Par exemple, vous pouvez tester et déboguer vos machines d'état dans la console, utiliser l'TestStateAPI pour tester un état individuel ou utiliser Step Functions Local pour tester les machines d'état localement.

À l'aide de l'TestStateAPI, vous définissez un état unique et vous l'exécutez. Vous pouvez tester un seul état sans créer de machine à états ni mettre à jour une machine à états existante.

Step FunctionsLocal est une version téléchargeable Step Functions qui vous permet de développer et de tester des applications à l'aide d'une version Step Functions exécutée dans votre propre environnement de développement. Avec Step Functions Local, vous pouvez exécuter vos machines d'état pour tester leurs flux de données d'entrée et de sortie, leurs intégrations avec les services pris en charge, etc. dans votre environnement de développement local.

Rubriques

- Utilisation de TestState l'API pour tester un état
- Tester les machines d'état localement

Utilisation de TestState l'API pour tester un état

L'TestStateAPI accepte la définition d'un état unique et l'exécute. Vous pouvez tester un état sans créer de machine à états ni mettre à jour une machine à états existante.

À l'aide de TestState l'API, vous pouvez tester les éléments suivants :

- Flux de données de traitement d'entrée et de sortie d'un État.
- Une Service AWS intégration avec d'autres Services AWS demandes et réponses
- Une demande et une réponse à une tâche HTTP

Pour tester un état, vous pouvez également utiliser la Step Functionsconsole ou le SDK. AWS Command Line Interface (AWS CLI)

L'`TestState`API assume un rôle IAM qui doit contenir les IAM autorisations requises pour les ressources auxquelles votre État accède. Pour plus d'informations sur les autorisations dont un État peut avoir besoin, consultezIAMautorisations d'utilisation de l' TestState API.

Rubriques

Considérations relatives à l'utilisation de l' TestState API

À l'aide de l'TestStateAPI, vous ne pouvez tester qu'un seul état à la fois. Les états que vous pouvez tester sont les suivants :

- Tous les types de tâches, sauf les activités
- Pass
- Attente
- Choice
- Succeed
- Fail

Lorsque vous utilisez l'`TestState`API, gardez à l'esprit les considérations suivantes.

- L' TestState API ne prend pas en charge les éléments suivants :
  - État de la tâcheÉtats qui utilisent les types de ressources suivants :
    - Activité
    - Modèles d'intégration des services de type `.sync` ou `.waitForTaskToken`
  - Parallèleétat
  - Mapétat
- Un test peut durer jusqu'à cinq minutes. Si un test dépasse cette durée, il échoue avec l'`States.Timeout`erreur.

Utilisation des niveaux d'inspection dans TestState l'API

Pour tester un état à l'aide de l'TestStateAPI, vous devez fournir la définition de cet état. Le test renvoie ensuite une sortie. Pour chaque État, vous pouvez spécifier la quantité de détails que vous souhaitez afficher dans les résultats des tests. Ces informations fournissent des informations supplémentaires sur l'état que vous testez. Par exemple, si vous avez utilisé des filtres de traitement des données d'entrée et de sortie, tels que `InputPath`ou `ResultPath`dans un état, vous pouvez afficher les résultats intermédiaires et finaux du traitement des données.

Step Functionsfournit les niveaux suivants pour spécifier les détails que vous souhaitez afficher :

- INFOS
- DÉBOGAGE
- TRACE

Tous ces niveaux renvoient également les `nextState` champs `status` et. `status`indique le statut de l'exécution de l'état. Par exemple, `SUCCEEDED`,`FAILED`,`RETRIABLE`, et`CAUGHT_ERROR`. `nextState`indique le nom du prochain état vers lequel effectuer la transition. Si vous n'avez pas défini d'état suivant dans votre définition, ce champ renvoie une valeur vide.

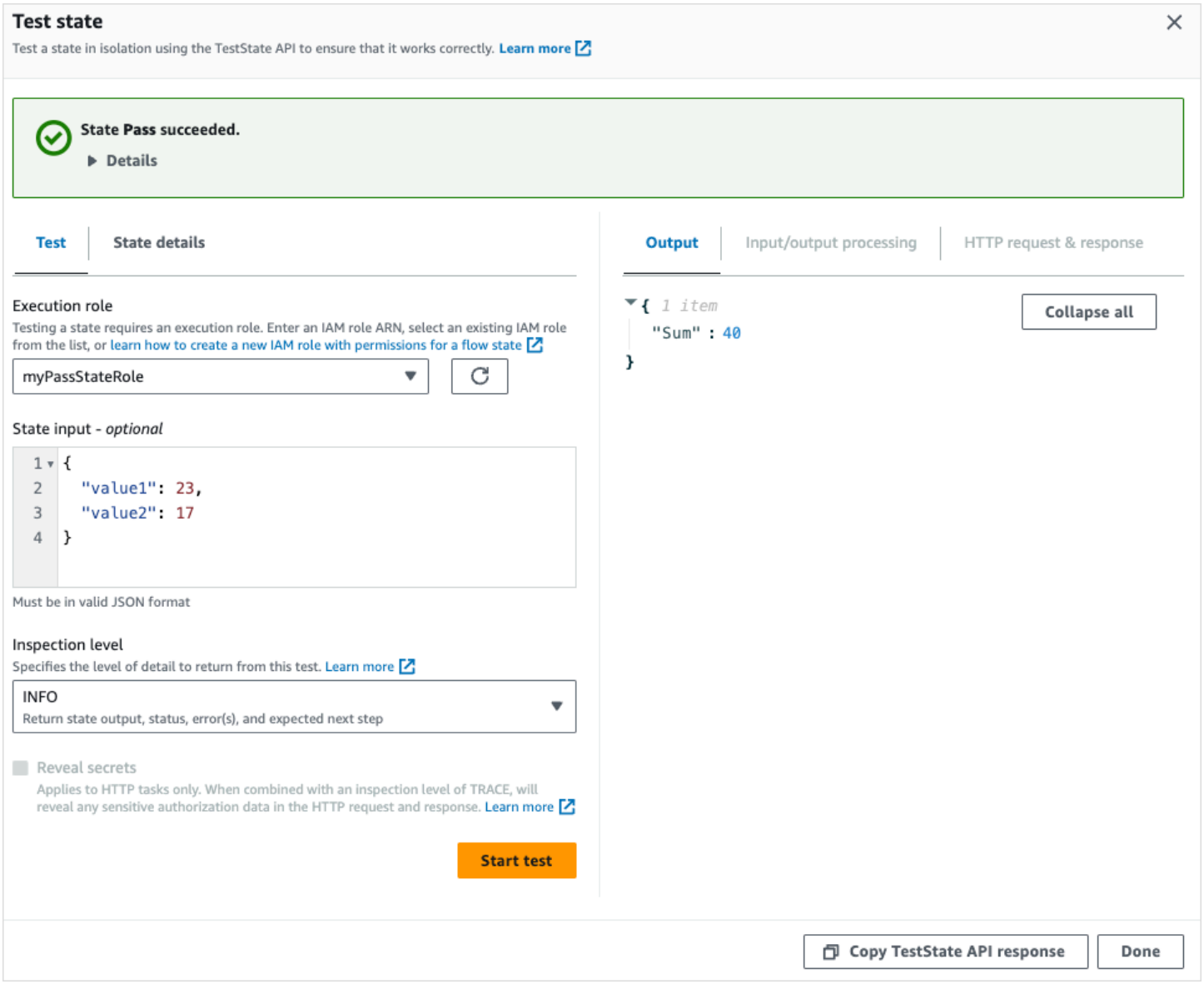
Pour plus d'informations sur le test d'un état à l'aide de ces niveaux d'inspection dans la Step Functions consoleAWS CLI, voir Tester un état (console) etTester un état en utilisant AWS CLI.

Niveau d'inspection INFO

En cas de succès du test, ce niveau indique l'état de sortie. Si le test échoue, ce niveau indique le résultat de l'erreur. Par défaut, Step Functions définit le niveau d'inspection sur INFO si vous ne spécifiez aucun niveau.

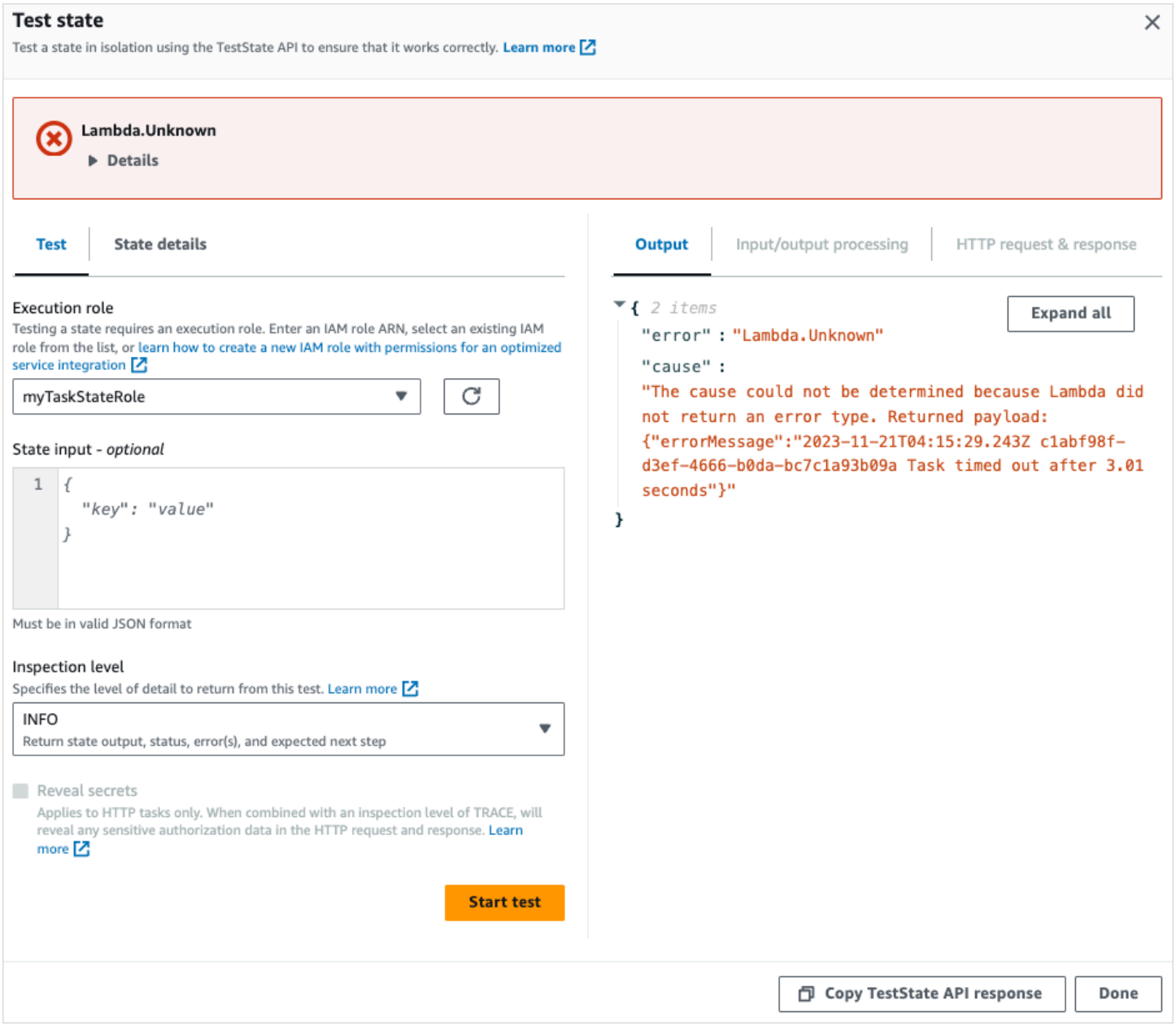
Exemple de test réussi avec le niveau INFO

L'image suivante montre un test de réussite du statut Pass. Le niveau d'inspection pour cet état est défini sur INFO et le résultat correspondant à l'état apparaît dans l'onglet Sortie.



Exemple de test avec un niveau INFO qui échoue

L'image suivante montre un test qui a échoué pour un état de tâche lorsque le niveau d'inspection est défini sur INFO. L'onglet Sortie affiche le résultat d'erreur qui inclut le nom de l'erreur et une explication détaillée de la cause de cette erreur.



Niveau d'inspection DEBUG

Si le test réussit, ce niveau indique l'état de sortie et le résultat du traitement des données d'entrée et de sortie.

Si le test échoue, ce niveau indique le résultat de l'erreur. Ce niveau indique les résultats intermédiaires du traitement des données jusqu'au point de défaillance. Supposons, par exemple, que vous ayez testé un état de tâche qui appelle une Lambda fonction. Imaginez que vous ayez appliqué les OutputPath filtres InputPathParamètres,ResultPath, et à l'état de la tâche. Supposons

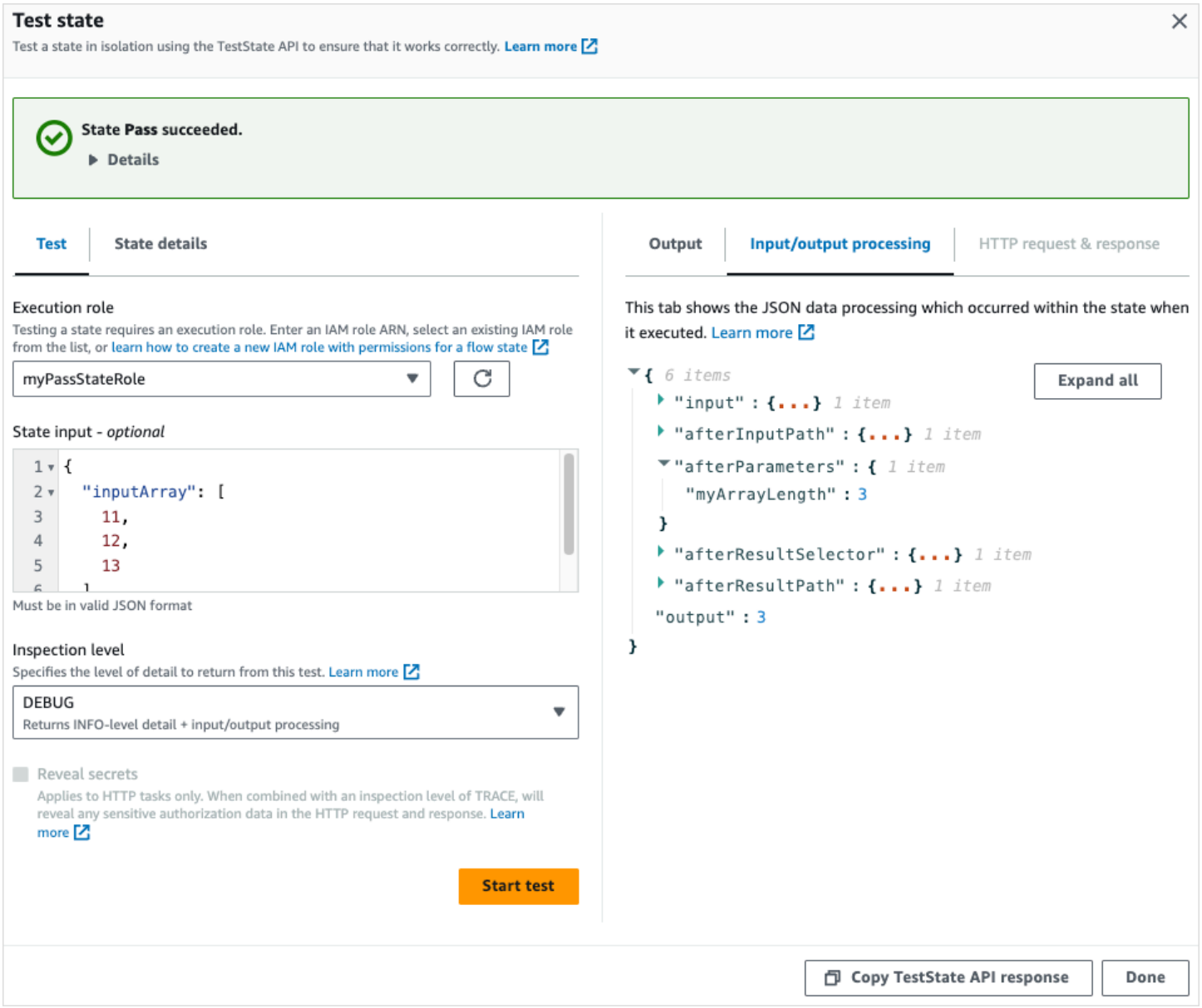
que l'invocation ait échoué. Dans ce cas, le `DEBUG` niveau affiche les résultats du traitement des données en fonction de l'application des filtres dans l'ordre suivant :

- `input`— Entrée d'état brute
- `afterInputPath`— Entrez après avoir Step Functions appliqué le `InputPath` filtre.
- `afterParameters`— L'entrée effective après Step Functions application du `Parameters` filtre.

Les informations de diagnostic disponibles à ce niveau peuvent vous aider à résoudre les problèmes liés à une intégration de services ou à un flux de traitement des données d'entrée et de sortie que vous avez peut-être défini.

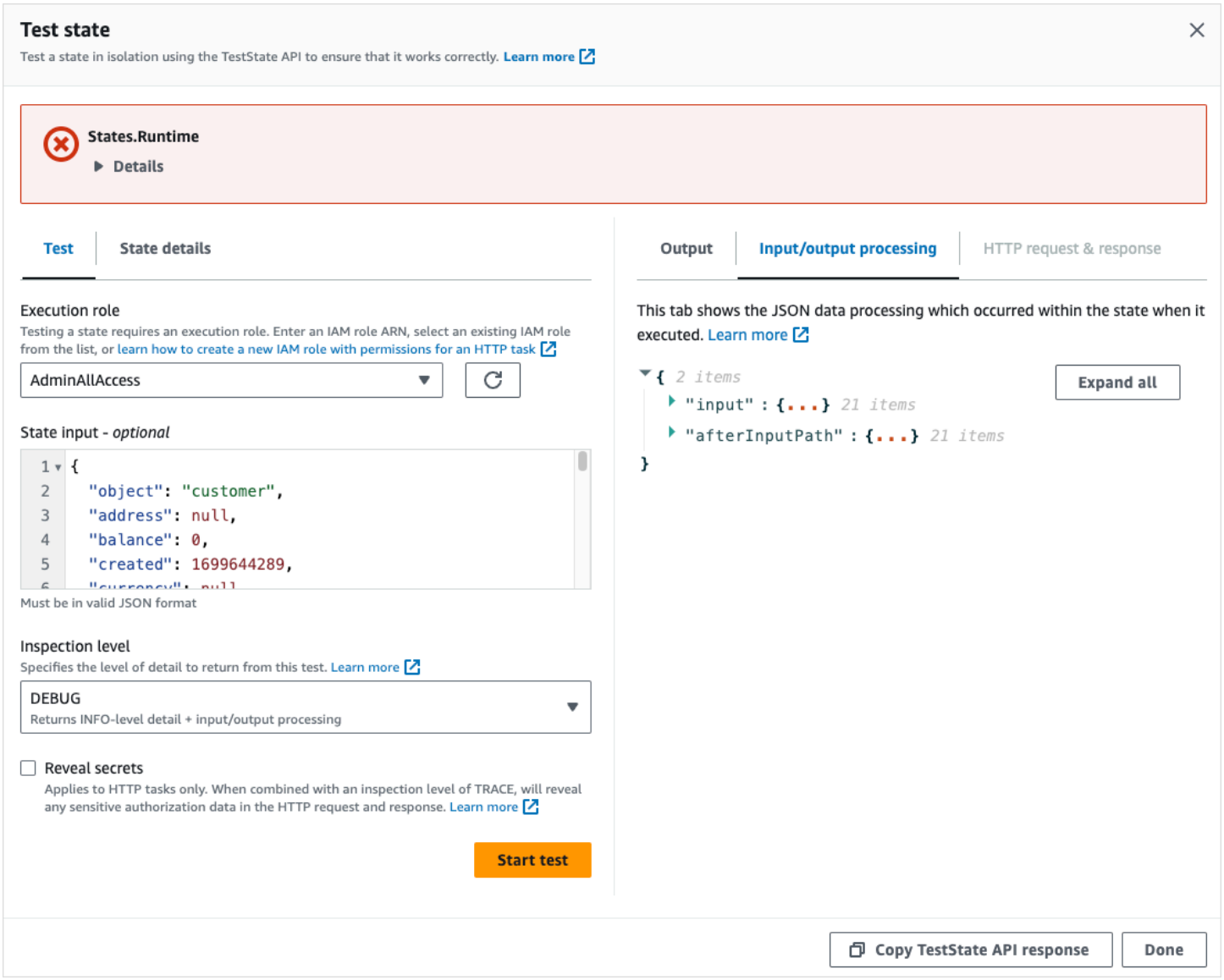
Exemple de test réussi avec un niveau DEBUG

L'image suivante montre un test de réussite du statut Pass. Le niveau d'inspection pour cet état est défini sur DEBUG. L'onglet Traitement des entrées/sorties de l'image suivante montre le résultat de l'application de `Parameters`sur l'entrée prévue pour cet état.



Exemple de test avec un niveau DEBUG qui échoue

L'image suivante montre un test qui a échoué pour un état de tâche lorsque le niveau d'inspection est défini sur DEBUG. L'onglet Traitement des entrées/sorties de l'image suivante montre le résultat du traitement des données d'entrée et de sortie pour l'état jusqu'à sa défaillance.



Niveau d'inspection TRACE

Step Functionsfournit le niveau TRACE pour tester une tâche HTTP. Ce niveau renvoie des informations sur la requête HTTP émise Step Functions et la réponse renvoyée par une API tierce. La réponse peut contenir des informations, telles que les en-têtes et le corps de la demande. De plus, vous pouvez visualiser l'état de sortie et le résultat du traitement des données d'entrée et de sortie à ce niveau.

Si le test échoue, ce niveau indique le résultat de l'erreur.

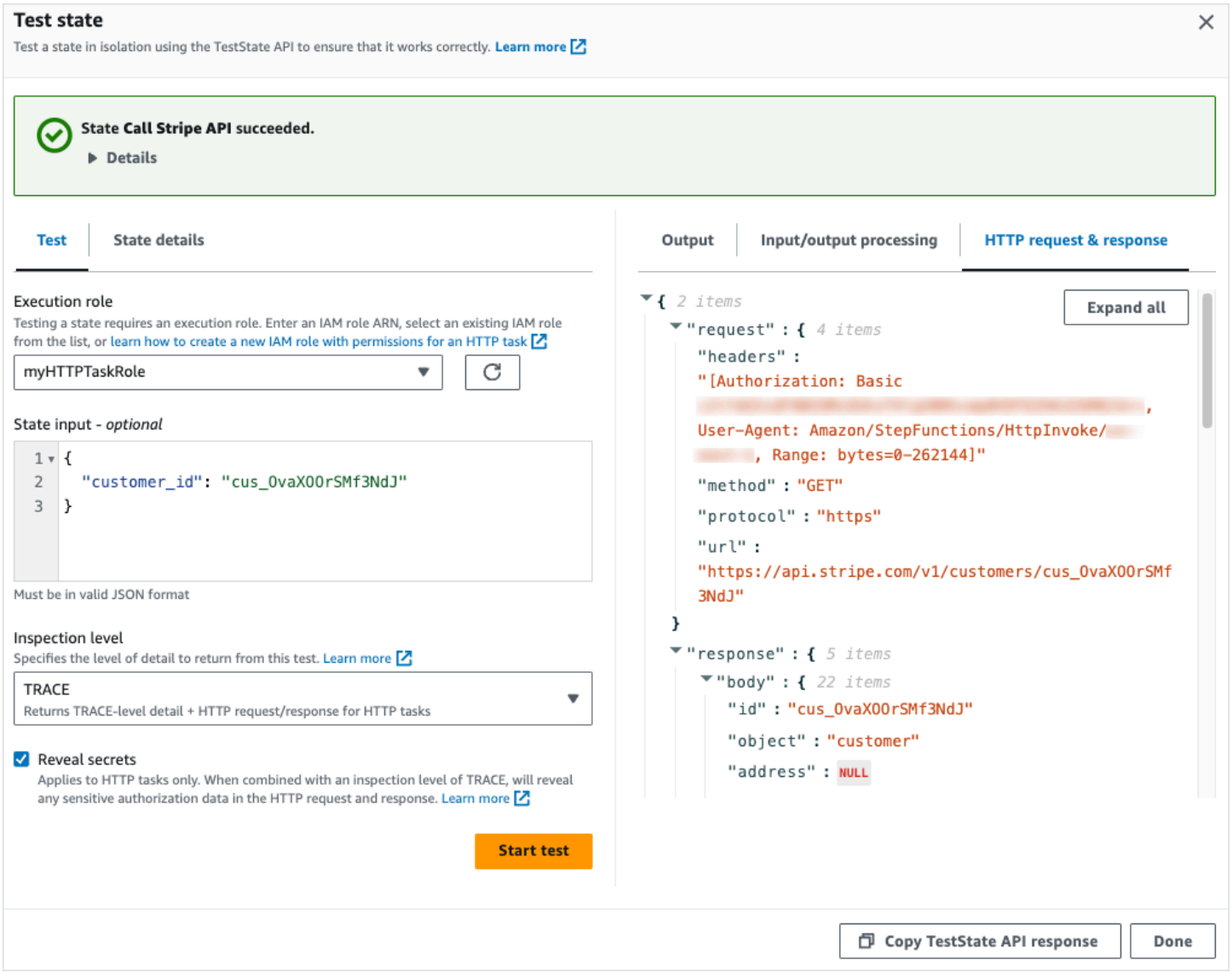
Ce niveau ne s'applique qu'à la tâche HTTP. Step Functionsgénère une erreur si vous utilisez ce niveau pour d'autres types d'états.

Lorsque vous définissez le niveau d'inspection sur TRACE, vous pouvez également consulter les secrets inclus dans la EventBridge connexion. Pour ce faire, vous devez définir le `revealSecrets` paramètre sur `true` dans l'TestStateAPI. En outre, vous devez vous assurer que l'IAMutilisateur qui appelle l' TestState API est autorisé à effectuer cette `states:RevealSecrets` action. Pour un exemple de IAM politique définissant l'`states:RevealSecrets`autorisation, voirIAMautorisations d'utilisation de l' TestState API. Sans cette autorisation, Step Functions renvoie une erreur de refus d'accès.

Si vous définissez le `revealSecrets` paramètre sur`false`, Step Functions omet tous les secrets dans les données de requête et de réponse HTTP.

Exemple de test réussi avec le niveau TRACE

L'image suivante montre un test de réussite d'une tâche HTTP. Le niveau d'inspection pour cet état est défini sur TRACE. L'onglet Requête et réponse HTTP de l'image suivante montre le résultat de l'appel d'API tiers.



IAMautorisations d'utilisation de l' TestState API

L'IAMutilisateur qui appelle l'`TestState`API doit être autorisé à effectuer les `iam:PassRole` actions `states:TestState` et. En outre, si vous définissez le paramètre RevealSecrets sur`true`, vous devez vous assurer que l'IAMutilisateur est autorisé à effectuer l'`states:RevealSecrets`action. Sans cette autorisation, Step Functions renvoie une erreur de refus d'accès.

Vous devez également vous assurer que votre rôle d'exécution contient les IAM autorisations requises pour les ressources auxquelles votre État accède. Pour plus d'informations sur les autorisations dont un État peut avoir besoin, consultez la section Gestion des rôles d'exécution.

L'exemple IAM de politique suivant définit les `states:RevealSecrets` autorisations `states:TestStateiam:PassRole`, et.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:TestState",
        "states:RevealSecrets",
        "iam:PassRole"
      ],
      "Resource": "*"
    }
  ]
}
```

Tester un état (console)

Vous pouvez tester un état dans la console et vérifier l'état de sortie ou le flux de traitement des données d'entrée et de sortie. Pour une tâche HTTP, vous pouvez tester la requête et la réponse HTTP brutes.

Pour tester un état

1. Ouvrez la console Step Functions.
2. Choisissez Créer une machine à états pour commencer à créer une machine à états ou choisissez une machine à états existante.
3. Dans Workflow Studio, choisissez l'état que vous souhaitez tester. Mode de conception
4. Choisissez Test state dans le Inspector panneau de Workflow Studio.
5. Dans la boîte de dialogue État du test, procédez comme suit :

    a. Pour Rôle d'exécution, choisissez un rôle d'exécution pour tester l'état. Assurez-vous que vous disposez des IAMautorisations requises pour l'état que vous souhaitez tester.

    b. (Facultatif) Fournissez toute entrée JSON dont l'état sélectionné a besoin pour le test.

    c. Pour le niveau d'inspection, sélectionnez l'une des options suivantes en fonction des valeurs que vous souhaitez afficher :

    - INFO — Affiche l'état de sortie dans l'onglet Sortie si le test réussit. Si le test échoue, INFO affiche le résultat de l'erreur, qui inclut le nom de l'erreur et une explication détaillée

de la cause de cette erreur. Par défaut, Step Functions définit le niveau d'inspection sur INFO si vous ne sélectionnez aucun niveau.

- DEBUG — Affiche l'état de sortie et le résultat du traitement des données d'entrée et de sortie en cas de réussite du test. Si le test échoue, DEBUG affiche le résultat de l'erreur, qui inclut le nom de l'erreur et une explication détaillée de la cause de cette erreur.
- TRACE — Affiche la requête et la réponse HTTP brutes et est utile pour vérifier les en-têtes, les paramètres de requête et d'autres détails spécifiques à l'API. Cette option n'est disponible que pour la tâche HTTP.

  Vous pouvez éventuellement choisir de sélectionner Révéler les secrets. Associé à TRACE, ce paramètre vous permet de voir les données sensibles insérées par la EventBridge connexion, telles que les clés d'API. L'identité IAM utilisateur que vous utilisez pour accéder à la console doit être autorisée à effectuer l'`states:RevealSecrets`action. Sans cette autorisation, Step Functions génère une erreur de refus d'accès lorsque vous démarrez le test. Pour un exemple de IAM politique définissant l'`states:RevealSecrets`autorisation, consultezIAMautorisations d'utilisation de l' TestState API.

d. Choisissez Démarrer le test.

Tester un état en utilisant AWS CLI

Vous pouvez tester un état pris en charge à l'aide de l'TestStateAPI duAWS CLI. Cette API accepte la définition d'un état et l'exécute.

Pour chaque État, vous pouvez spécifier la quantité de détails que vous souhaitez afficher dans les résultats des tests. Ces informations fournissent des informations supplémentaires sur l'exécution de l'État, notamment le résultat du traitement des données d'entrée et de sortie et les informations de requête et de réponse HTTP. Les exemples suivants présentent les différents niveaux d'inspection que vous pouvez spécifier pour l' TestState API. N'oubliez pas de remplacer le texte *`en italique`* par les informations spécifiques à votre ressource.

Cette section contient les exemples suivants qui décrivent comment vous pouvez utiliser les différents niveaux d'inspection Step Functions fournis dans AWS CLI :

- Utilisation d'INFO InspectionLevel
- Utilisation de DEBUG InspectionLevel
- Utilisation de TRACE InspectionLevel

- Utilisation de l'utilitaire jq AWS CLI pour filtrer et imprimer la réponse HTTP renvoyée par l' TestState API

Exemple 1 : utilisation d'INFO InspectionLevel pour tester un état Choice

Pour tester un état à l'aide du `INFO` paramètre InspectionLevel dans le AWS CLI, exécutez la `test-state` commande comme indiqué dans l'exemple suivant.

```
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}'
```

Cet exemple utilise un état Choice pour déterminer le chemin d'exécution de cet état en fonction de l'entrée numérique que vous fournissez. Par défaut, Step Functions définit la valeur `inspectionLevel` à `INFO` si vous ne définissez aucun niveau.

Step Functionsrenvoie le résultat suivant.

```
{
    "output": "{\"number\": 2}",
    "nextState": "Equals 2",
    "status": "SUCCEEDED"
}
```

Exemple 2 : utilisation de DEBUG InspectionLevel pour déboguer le traitement des données d'entrée et de sortie dans un état Passage

Pour tester un état à l'aide du `DEBUG` paramètre InspectionLevel dans le AWS CLI, exécutez la `test-state` commande comme indiqué dans l'exemple suivant.

```
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
```

Cet exemple utilise un Pass état pour montrer comment filtrer et manipuler les données JSON d'entrée à l'aide des Step Functions filtres de traitement des données d'entrée et de sortie. Cet exemple utilise les filtres suivants : `InputPathParamètres`,`ResultPath`, et`OutputPath`.

Step Functionsrenvoie le résultat suivant.

```
{
    "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"
}
```

Exemple 3 : utilisation de TRACE InspectionLevel et RevealSecrets pour inspecter la requête HTTP envoyée à une API tierce

Pour tester une tâche HTTP en utilisant `TRACE` InspectionLevel avec le paramètre RevealSecrets dans le AWS CLI, exécutez la `test-state` commande comme indiqué dans l'exemple suivant.

```
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
```

Cet exemple teste si la tâche HTTP appelle l'API tierce spécifiée,`https://httpbin.org/`. Il affiche également les données de requête et de réponse HTTP pour l'appel d'API.

```
{
```

```
    "output": "{\"Headers\":{\"date\":[\"Tue, 21 Nov 2023 00:06:17 GMT\"],
\"access-control-allow-origin\":[\"*\"],\"content-length\":[\"620\"],\"server\":
[\"gunicorn/19.9.0\"],\"access-control-allow-credentials\":[\"true\"],\"content-
type\":[\"application/json\"]},\"ResponseBody\":{\"args\":{\"QueryParam1\":
\"QueryParamValue1\",\"queryParam\":\"q1\"},\"headers\":{\"Authorization
\":\"Basic XXXXXXXX\",\"Content-Type\":\"application/json; charset=UTF-8\",
\"Customheader1\":\"CustomHeaderValue1\",\"Definitionheader\":\"h1\",\"Host\":
\"httpbin.org\",\"Range\":\"bytes=0-262144\",\"Transfer-Encoding\":\"chunked\",
\"User-Agent\":\"Amazon|StepFunctions|HttpInvoke|us-east-1\",\"X-Amzn-Trace-Id\":
\"Root=1-0000000-0000-0000-0000-000000000000\"},\"origin\":\"12.34.567.891\",\"url\":
\"https://httpbin.org/get?queryParam=q1&QueryParam1=QueryParamValue1\"},\"StatusCode
\":200,\"StatusText\":\"OK\"}",
    "inspectionData": {
        "input": "{}",
        "afterInputPath": "{}",
        "afterParameters": "{\"Method\":\"GET\",\"Authentication\":{\"ConnectionArn
\":\"arn:aws:events:us-east-1:123456789012:connection/foo/a59c10f0-a315-4c1f-
be6a-559b9a0c6250\"},\"ApiEndpoint\":\"https://httpbin.org/get\",\"Headers\":
{\"definitionHeader\":\"h1\"},\"RequestBody\":{\"message\":\"Hello from Step Functions!
\"},\"QueryParameters\":{\"queryParam\":\"q1\"}}",
        "result": "{\"Headers\":{\"date\":[\"Tue, 21 Nov 2023 00:06:17 GMT\"],
\"access-control-allow-origin\":[\"*\"],\"content-length\":[\"620\"],\"server\":
[\"gunicorn/19.9.0\"],\"access-control-allow-credentials\":[\"true\"],\"content-
type\":[\"application/json\"]},\"ResponseBody\":{\"args\":{\"QueryParam1\":
\"QueryParamValue1\",\"queryParam\":\"q1\"},\"headers\":{\"Authorization
\":\"Basic XXXXXXXX\",\"Content-Type\":\"application/json; charset=UTF-8\",
\"Customheader1\":\"CustomHeaderValue1\",\"Definitionheader\":\"h1\",\"Host\":
\"httpbin.org\",\"Range\":\"bytes=0-262144\",\"Transfer-Encoding\":\"chunked\",
\"User-Agent\":\"Amazon|StepFunctions|HttpInvoke|us-east-1\",\"X-Amzn-Trace-Id\":
\"Root=1-0000000-0000-0000-0000-000000000000\"},\"origin\":\"12.34.567.891\",\"url\":
\"https://httpbin.org/get?queryParam=q1&QueryParam1=QueryParamValue1\"},\"StatusCode
\":200,\"StatusText\":\"OK\"}",
        "afterResultSelector": "{\"Headers\":{\"date\":[\"Tue, 21 Nov 2023
 00:06:17 GMT\"],\"access-control-allow-origin\":[\"*\"],\"content-length\":
[\"620\"],\"server\":[\"gunicorn/19.9.0\"],\"access-control-allow-credentials
\":[\"true\"],\"content-type\":[\"application/json\"]},\"ResponseBody\":{\"args
\":{\"QueryParam1\":\"QueryParamValue1\",\"queryParam\":\"q1\"},\"headers\":
{\"Authorization\":\"Basic XXXXXXXX\",\"Content-Type\":\"application/json;
 charset=UTF-8\",\"Customheader1\":\"CustomHeaderValue1\",\"Definitionheader\":\"h1\",
\"Host\":\"httpbin.org\",\"Range\":\"bytes=0-262144\",\"Transfer-Encoding\":\"chunked
\",\"User-Agent\":\"Amazon|StepFunctions|HttpInvoke|us-east-1\",\"X-Amzn-Trace-Id\":
\"Root=1-0000000-0000-0000-0000-000000000000\"},\"origin\":\"12.34.567.891\",\"url\":
\"https://httpbin.org/get?queryParam=q1&QueryParam1=QueryParamValue1\"},\"StatusCode
\":200,\"StatusText\":\"OK\"}",
```

```
        "afterResultPath": "{\"Headers\":{\"date\":[\"Tue, 21 Nov 2023 00:06:17
 GMT\"],\"access-control-allow-origin\":[\"*\"],\"content-length\":[\"620\"],
\"server\":[\"gunicorn/19.9.0\"],\"access-control-allow-credentials\":[\"true\"],
\"content-type\":[\"application/json\"]},\"ResponseBody\":{\"args\":{\"QueryParam1\":
\"QueryParamValue1\",\"queryParam\":\"q1\"},\"headers\":{\"Authorization\":
\"Basic XXXXXXXX\",\"Content-Type\":\"application/json; charset=UTF-8\",
\"Customheader1\":\"CustomHeaderValue1\",\"Definitionheader\":\"h1\",\"Host\":
\"httpbin.org\",\"Range\":\"bytes=0-262144\",\"Transfer-Encoding\":\"chunked\",
\"User-Agent\":\"Amazon|StepFunctions|HttpInvoke|us-east-1\",\"X-Amzn-Trace-Id\":
\"Root=1-0000000-0000-0000-0000-000000000000\"},\"origin\":\"12.34.567.891\",\"url\":
\"https://httpbin.org/get?queryParam=q1&QueryParam1=QueryParamValue1\"},\"StatusCode
\":200,\"StatusText\":\"OK\"}",
        "request": {
            "protocol": "https",
            "method": "GET",
            "url": "https://httpbin.org/get?
queryParam=q1&QueryParam1=QueryParamValue1",
            "headers": "[definitionHeader: h1, Authorization: Basic XXXXXXXX,
 CustomHeader1: CustomHeaderValue1, User-Agent: Amazon|StepFunctions|HttpInvoke|us-
east-1, Range: bytes=0-262144]",
            "body": "{\"message\":\"Hello from Step Functions!\",\"BodyKey1\":
\"BodyValue1\"}"
        },
        "response": {
            "protocol": "https",
            "statusCode": "200",
            "statusMessage": "OK",
            "headers": "[date: Tue, 21 Nov 2023 00:06:17 GMT, content-type:
 application/json, content-length: 620, server: gunicorn/19.9.0, access-control-allow-
origin: *, access-control-allow-credentials: true]",
            "body": "{\n  \"args\": {\n    \"QueryParam1\": \"QueryParamValue1\", \n
    \"queryParam\": \"q1\"\n  }, \n  \"headers\": {\n    \"Authorization\": \"Basic
 XXXXXXXX\", \n    \"Content-Type\": \"application/json; charset=UTF-8\", \n
 \"Customheader1\": \"CustomHeaderValue1\", \n    \"Definitionheader\": \"h1\", \n
   \"Host\": \"httpbin.org\", \n    \"Range\": \"bytes=0-262144\", \n    \"Transfer-
Encoding\": \"chunked\", \n    \"User-Agent\": \"Amazon|StepFunctions|HttpInvoke|us-
east-1\", \n    \"X-Amzn-Trace-Id\": \"Root=1-0000000-0000-0000-0000-000000000000\"\n
  }, \n  \"origin\": \"12.34.567.891\", \n  \"url\": \"https://httpbin.org/get?
queryParam=q1&QueryParam1=QueryParamValue1\"\n}\n"
        }
    },
    "status": "SUCCEEDED"
}
```

Exemple 4 : Utilisation de l'utilitaire jq pour filtrer et imprimer la réponse renvoyée par l' TestState API

L' TestState API renvoie les données JSON sous forme de chaînes échappées dans sa réponse. L'AWS CLIexemple suivant étend l'exemple 3 et utilise l'`jq`utilitaire pour filtrer et imprimer la réponse HTTP renvoyée par l' TestState API dans un format lisible par l'homme. Pour plus d'informations `jq` et ses instructions d'installation, consultez jq on GitHub.

```
aws stepfunctions test-state \
    --definition '{"Type": "Task", "Resource": "arn:aws:states:::http:invoke",
 "Parameters": {"Method": "GET", "Authentication": {"ConnectionArn":
 "arn:aws:events:us-
east-1:123456789012:connection/MyConnection/0000000-0000-0000-0000-000000000000"},
 "ApiEndpoint": "https://httpbin.org/get", "Headers": {"definitionHeader": "h1"},
 "RequestBody": {"message": "Hello from Step Functions!"}, "QueryParameters":
 {"queryParam": "q1"}}, "End": true}' \
    --role-arn arn:aws:iam::123456789012:role/myRole \
    --inspection-level TRACE \
    --reveal-secrets \
    | jq '.inspectionData.response.body | fromjson'
```

L'exemple suivant montre le résultat renvoyé dans un format lisible par l'homme.

```
{
  "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"
```

```
}
```

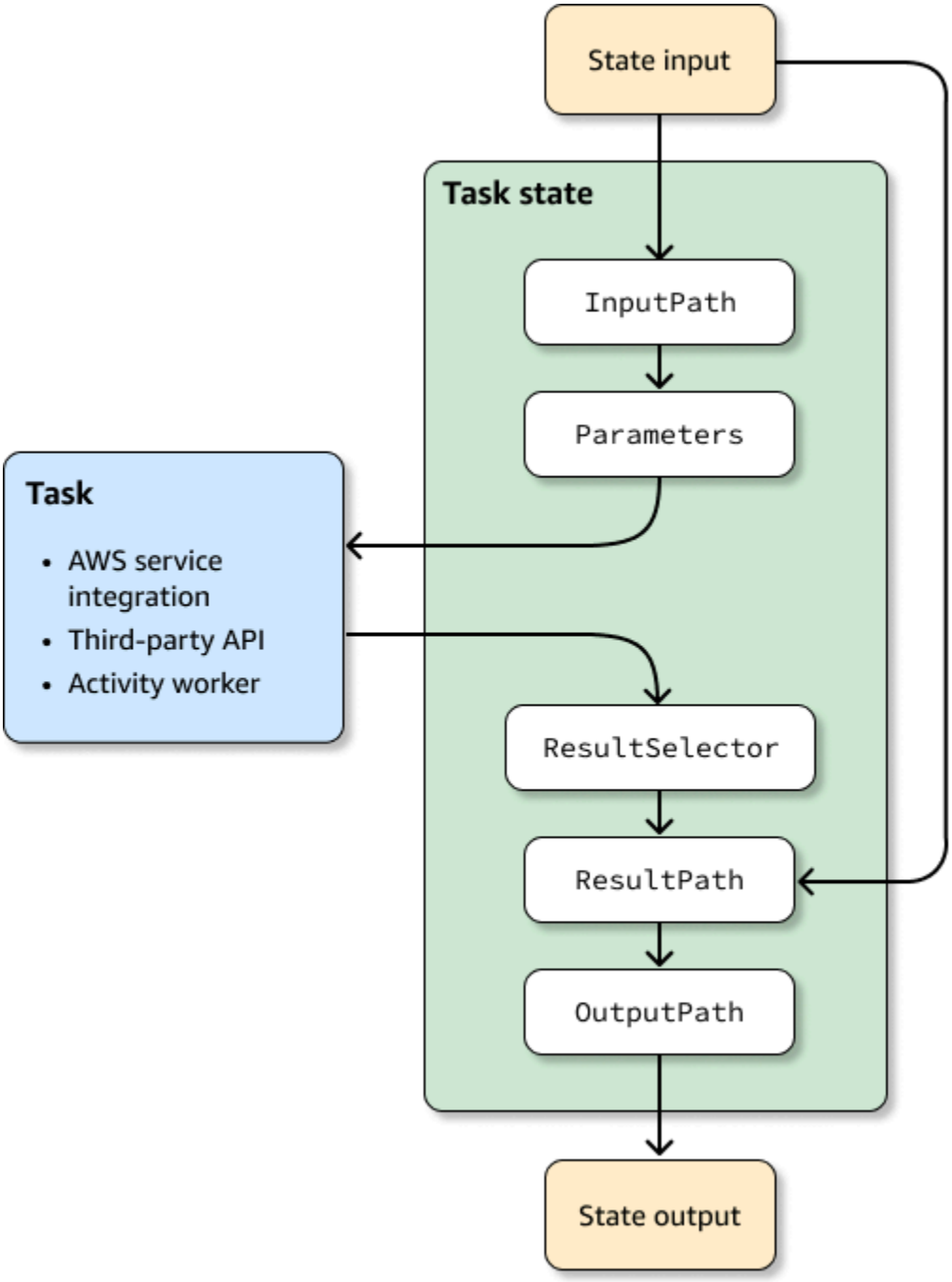
Test et débogage du flux de données d'entrée et de sortie

L'`TestState`API est utile pour tester et déboguer les données qui circulent dans votre flux de travail. Cette section fournit quelques concepts clés et explique comment les utiliser TestState à cette fin.

Concepts clés

DansStep Functions, le processus de filtrage et de manipulation des données JSON lorsqu'elles passent par les états de votre machine à états est appelé traitement des entrées et des sorties. Pour plus d'informations sur la façon dont cela fonctionne, consultez Traitement des entrées et des sorties dans Step Functions.

Tous les types d'états du Amazon States Language (ASL) (Task, Parallel, Map, Pass, Wait, Choice, Succeed et Fail) partagent un ensemble de champs communs permettant de filtrer et de manipuler les données JSON qui les traversent. Ces champs sont les suivants : InputPathParamètres,ResultSelector,ResultPath, etOutputPath. Support pour chaque domaine varie d'un État à l'autre. Au moment de l'exécution, Step Functions applique chaque champ dans un ordre spécifique. Le schéma suivant montre l'ordre dans lequel ces champs sont appliqués aux données dans un état de tâche :



La liste suivante décrit l'ordre d'application des champs de traitement d'entrée et de sortie présentés dans le diagramme.

1. L'entrée d'état correspond aux données JSON transmises à l'état actuel à partir d'un état précédent.
2. InputPathfiltre une partie de l'entrée d'état brut.
3. Paramètresconfigure l'ensemble de valeurs à transmettre à la tâche.

4. La tâche exécute un travail et renvoie un résultat.
5. ResultSelectorsélectionne un ensemble de valeurs à ne pas inclure dans le résultat de la tâche.
6. ResultPathcombine le résultat avec l'entrée d'état brute ou remplace le résultat par celle-ci.
7. OutputPathfiltre une partie de la sortie pour passer à l'état suivant.
8. La sortie d'état correspond aux données JSON passées de l'état actuel à l'état suivant.

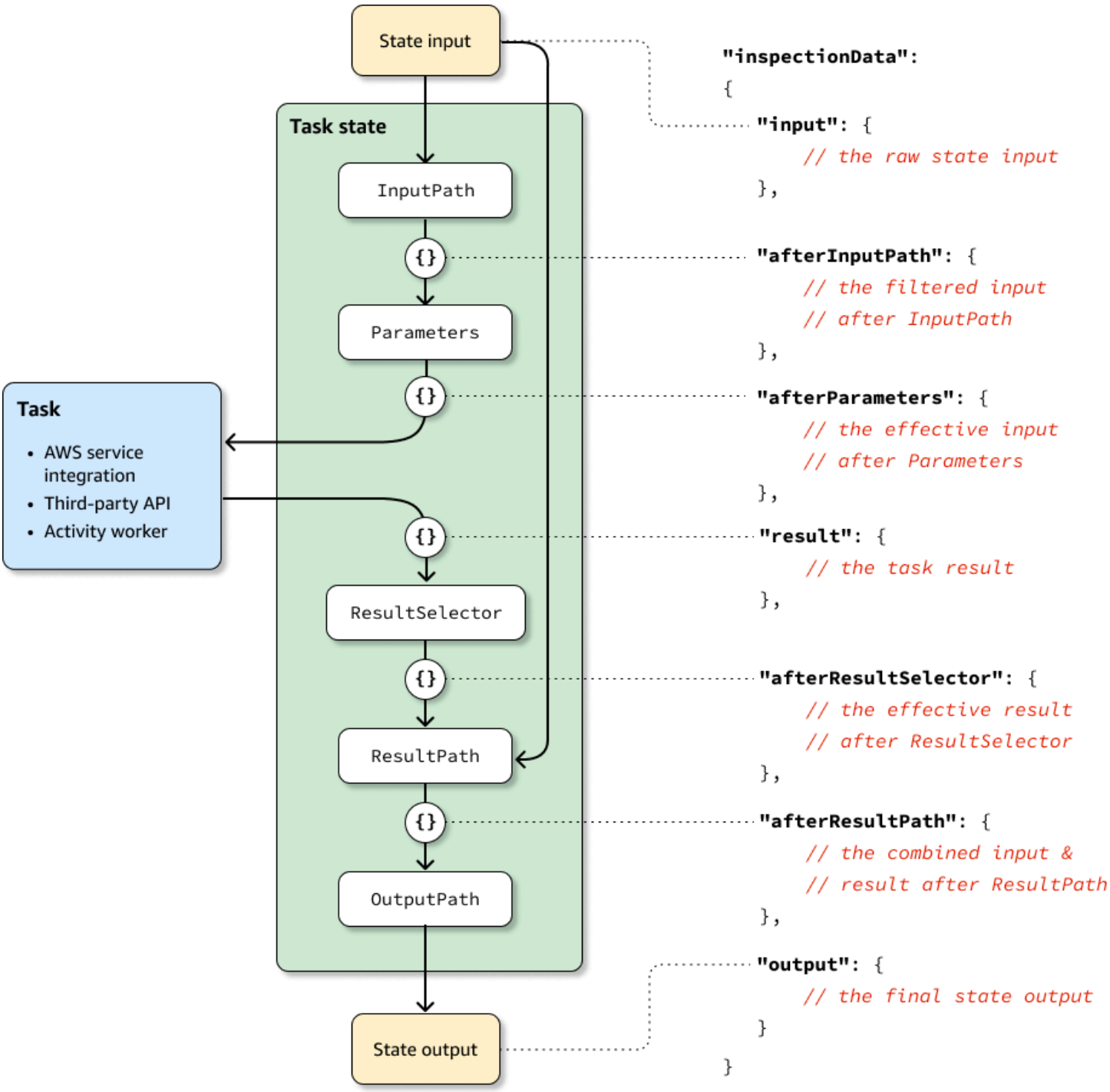
Ces champs de traitement d'entrée et de sortie sont facultatifs. Si vous n'utilisez aucun de ces champs dans votre définition d'état, la tâche consommera l'entrée d'état brute et renverra le résultat de la tâche en tant que sortie d'état.

Utilisation TestState pour inspecter le traitement des entrées et des sorties

Lorsque vous appelez l'`TestState`API et que vous définissez le `inspectionLevel` paramètre sur`DEBUG`, la réponse de l'API inclut un objet appelé`inspectionData`. Cet objet contient des champs qui vous aident à inspecter la manière dont les données ont été filtrées ou manipulées dans l'état dans lequel elles ont été exécutées. L'exemple suivant montre l'`inspectionData`objet correspondant à un état de tâche.

```
"inspectionData":   {
  "input": string,
  "afterInputPath": string,
  "afterParameters": string,
  "result": string,
  "afterResultSelector": string,
  "afterResultPath": string,
  "output": string
}
```

Dans cet exemple, chaque champ contenant le `after` préfixe affiche les données après l'application d'un champ particulier. Par exemple, `afterInputPath` montre l'effet de l'application du `InputPath` champ pour filtrer l'entrée d'état brute. Le schéma suivant met en correspondance chaque champ de définition ASL avec le champ correspondant dans l'`inspectionData`objet :



Pour des exemples d'utilisation de l' TestState API pour déboguer le traitement des entrées et des sorties, consultez les pages suivantes :

- Tester un état à l'aide du niveau d'inspection DEBUG dans la console Step Functions
- Tester un état à l'aide du niveau d'inspection DEBUG dans AWS CLI

Tester les machines d'état localement

AWSStep Functions Local est une version téléchargeable de Step Functions qui vous permet de développer et de tester des applications à l'aide d'une version de Step Functions exécutée dans votre propre environnement de développement. La version locale de Step Functions peut invoquer AWS Lambda des fonctions, à la fois dans AWS et pendant l'exécution locale. Vous pouvez également coordonner d'autres services AWS pris en charge.

> **Note**
>
> Step Functions Local utilise des comptes factices pour fonctionner.

Lorsque vous exécutez Step Functions Local, vous pouvez utiliser l'une des méthodes suivantes pour invoquer des intégrations de services :

- Configuration de points de terminaison locaux pour AWS Lambda et d'autres services. Pour plus d'informations sur les points de terminaison pris en charge, consultezConfiguration des options de configuration pour Step Functions Local.
- Téléphoner directement à un AWS service depuis Step Functions Local.
- Se moquer de la réponse des intégrations de services. Pour plus d'informations sur l'utilisation d'intégrations de services simulées, consultez. Utilisation d'intégrations de services simulées

AWSStep Functions Local est disponible sous forme de package JAR ou d'image Docker autonome qui s'exécute sur Microsoft Windows, Linux, macOS et d'autres plateformes compatibles avec Java ou Docker.

> **Warning**
>
> La version téléchargeable de AWS Step Functions est destinée à être utilisée uniquement à des fins de test et ne doit jamais être utilisée pour traiter des informations sensibles.

> **Tip**
>
> Assurez-vous d'utiliser Step Functions Local version 1.12.0 ou supérieure pour pouvoir inclure toutes les fonctions intrinsèques dans vos flux de travail.

Les rubriques suivantes décrivent comment configurer Step Functions Local à l'aide de Docker et de fichiers JAR, et comment exécuter Step Functions Local pour fonctionner avec AWS Lambda AWS Serverless Application Model (AWS SAM) CLI Local ou d'autres services pris en charge.

Rubriques

Configuration de Step Functions Local (version téléchargeable) et Docker

L'image Docker Step Functions Local vous permet de démarrer rapidement avec Step Functions Local en utilisant une image Docker avec toutes les dépendances nécessaires. L'image Docker vous permet d'inclure Step Functions Local dans vos versions conteneurisées et dans le cadre de vos tests d'intégration continus.

Pour obtenir l'image Docker de Step Functions Local, consultez https://hub.docker.com/r/amazon/aws-stepfunctions-local ou entrez la commande Docker `pull` suivante.

```
docker pull amazon/aws-stepfunctions-local
```

Pour démarrer la version téléchargeable de Step Functions sur Docker, exécutez la commande `run` Docker suivante

```
docker run -p 8083:8083 amazon/aws-stepfunctions-local
```

Pour interagir avec les services pris en charge AWS Lambda ou avec d'autres services pris en charge, vous devez d'abord configurer vos informations d'identification et les autres options de configuration. Pour plus d'informations, consultez les rubriques suivantes :

- Configuration des options de configuration pour Step Functions Local
- Informations d'identification et configuration pour Docker

Configuration de Step Functions en local (version téléchargeable) - Version Java

La version téléchargeable de AWS Step Functions est fournie sous forme de fichier JAR exécutable et d'image Docker. L'application Java s'exécute sur Windows, Linux, macOS X et autres plateformes qui prennent en charge Java. En plus de Java, vous devez installer le kit AWS Command Line Interface (AWS CLI). Pour plus d'informations sur l'installation et la configuration duAWS CLI, consultez le guide de AWS Command Line Interface l'utilisateur.

Pour configurer et exécuter Step Functions sur votre ordinateur

1. Téléchargez Step Functions à l'aide des liens suivants.

Télécharger les liens	Total de contrôle
.tar.gz	.tar.gz.md5
.zip	.zip .md5

2. Extrayez le fichier `.zip`.
3. Testez le téléchargement et affichez les informations de version.

```
$ java -jar StepFunctionsLocal.jar -v
Step Function Local
Version: 1.0.0
Build: 2019-01-21
```

4. (Facultatif) Affichez la liste des commandes disponibles.

```
$ java -jar StepFunctionsLocal.jar -h
```

5. Pour démarrer Step Functions sur votre ordinateur, ouvrez une invite de commande, accédez au répertoire dans lequel vous avez effectué l'extraction `StepFunctionsLocal.jar` et entrez la commande suivante.

```
java -jar StepFunctionsLocal.jar
```

6. Pour accéder à Step Functions s'exécutant localement, utilisez le `--endpoint-url` paramètre. Par exemple, en utilisant leAWS CLI, vous devez spécifier les commandes Step Functions comme suit :

```
aws stepfunctions --endpoint-url http://localhost:8083 command
```

> **Note**
>
> Par défaut, Step Functions Local utilise un compte de test local et des informations d'identification, et la AWS région est définie sur USA Est (Virginie du Nord). Pour utiliser Step Functions Local avec AWS Lambda ou d'autres services pris en charge, vous devez configurer vos informations d'identification et votre région.
> Si vous utilisez des flux de travail Express avec Step Functions Local, l'historique d'exécution sera enregistré dans un fichier journal. Il n'est pas enregistré dans CloudWatch Logs. Le chemin du fichier journal sera basé sur l'ARN du groupe de CloudWatch journaux des journaux fourni lors de la création de la machine d'état locale. Le fichier journal sera stocké `/aws/states/log-group-name/${execution_arn}.log` par rapport à l'emplacement où vous exécutez Step Functions Local. Par exemple, si l'ARN d'exécution est :
>
> ```
> arn:aws:states:us-east-1:123456789012:express:test:example-ExpressLogGroup-wJalrXUtnFEMI
> ```
>
> le fichier journal sera :
>
> ```
> aws/states/log-group-name/arn:aws:states:us-east-1:123456789012:express:test:example-ExpressLogGroup-wJalrXUtnFEMI.log
> ```

Configuration des options de configuration pour Step Functions Local

Lorsque vous démarrez AWS Step Functions Local à l'aide du fichier JAR, vous pouvez définir les options de configuration en utilisant le AWS Command Line Interface (AWS CLI) ou en les incluant dans l'environnement système. Pour Docker, vous devez spécifier ces options dans un fichier auquel vous faites référence lors du démarrage de Step Functions Local.

Options de configuration

Lorsque vous configurez le conteneur Step Functions Local pour utiliser un point de terminaison de remplacement tel que Lambda Endpoint et Batch Endpoint, et que vous appelez ce point de terminaison, Step Functions Local n'utilise pas les informations d'identification que vous spécifiez. La définition de ces remplacements de point de terminaison est facultative.

Option	Ligne de commande	Environnement
Compte	-compte, --aws-account	AWS_ACCOUNT_ID
Région	-région, --aws-region	AWS_DEFAULT_REGION
Échelle de délai d'attente	-waitTimeScale, --wait-time-scale	WAIT_TIME_SCALE
Point de terminaison Lambda	-Point de terminaison Lambda, --lambda-point de terminaison	LAMBDA_ENDPOINT
Point de terminaison par lot	-BatchEndpoint, --batch-endpoint	BATCH_ENDPOINT
Point de terminaison DynamoDB	-point de terminaison DynamoDB, --dynamodb-point de terminaison	DYNAMODB_ENDPOINT
Point de terminaison ECS	-ECSendpoint, --ecs-endpoint	ECS_ENDPOINT
Point de terminaison Glue	-glueEndpoint, --glue-endpoint	GLUE_ENDPOINT
SageMaker Point final	-sageMakerEndpoint, --sagemaker-endpoint	SAGE_MAKER_ENDPOINT
Point de terminaison SQS	-SQSendpoint, --sqs-endpoint	SQS_ENDPOINT
Point de terminaison SNS	-SNSendpoint, --sns-endpoint	SNS_ENDPOINT
Point de terminaison Step Functions	-stepFunctionsEndpoint, --step-functions-endpoint	STEP_FUNCTIONS_ENDPOINT

Informations d'identification et configuration pour Docker

Pour configurer Step Functions Local pour Docker, créez le fichier suivant : `aws-stepfunctions-local-credentials.txt`

Ce fichier contient vos informations d'identification et d'autres options de configuration. Ce qui suit peut être utilisé comme modèle lors de la création du `aws-stepfunctions-local-credentials.txt` fichier.

```
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
```

Une fois que vous avez configuré vos informations d'identification et vos options de configuration dans`aws-stepfunctions-local-credentials.txt`, lancez Step Functions avec la commande suivante.

```
docker run -p 8083:8083 --env-file aws-stepfunctions-local-credentials.txt amazon/aws-stepfunctions-local
```

> **Note**
>
> Il est recommandé d'utiliser le nom DNS spécial`host.docker.internal`, qui correspond à l'adresse IP interne utilisée par l'hôte, par exemple`http://host.docker.internal:8000`. Pour plus d'informations, consultez la documentation Docker pour Mac et Windows sous Fonctionnalités réseau dans Docker Desktop pour Mac et Fonctionnalités réseau dans Docker Desktop pour Windows respectivement.

Exécution de Step Functions Local sur votre ordinateur

Utilisez la version locale de Step Functions pour configurer, développer et tester des machines d'état sur votre ordinateur.

Exécuter une machine HelloWorld d'état localement

Après avoir exécuté Step Functions localement avec le AWS Command Line Interface (AWS CLI), vous pouvez démarrer une exécution State Machine.

1. Créez une machine à états à partir du AWS CLI en échappant à la définition de la machine à états.

```
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**
>
> Le n'`role-arn`est pas utilisé pour Step Functions Local, mais vous devez l'inclure avec la syntaxe appropriée. Vous pouvez utiliser l'ARN (Amazon Resource Name) de l'exemple précédent.

Si vous créez correctement la machine à états, Step Functions répond en indiquant la date de création et l'ARN de la machine à états.

```
{
    "creationDate": 1548454198.202,
    "stateMachineArn": "arn:aws:states:us-
east-1:123456789012:stateMachine:HelloWorld"
}
```

2. Démarrez une exécution à l'aide de l'ARN de la machine d'état que vous avez créée.

```
aws stepfunctions --endpoint-url http://localhost:8083 start-execution --state-
machine-arn arn:aws:states:us-east-1:123456789012:stateMachine:HelloWorld
```

Step Functions Local avec AWS SAM CLI Local

Vous pouvez utiliser la version locale de Step Functions avec une version locale deAWS Lambda. Pour configurer cela, vous devez installer et configurer AWS SAM.

Pour plus d'informations sur la configuration et l'exécution de AWS SAM, consultez les rubriques suivantes :

- Configuration d'AWS SAM
- Démarrez la version locale de l'interface de ligne de commande AWS SAM.

Lorsque Lambda est exécuté sur votre système local, vous pouvez démarrer Step Functions Local. Dans le répertoire où vous avez extrait vos fichiers JAR locaux de Step Functions, lancez Step Functions Local et utilisez le `--lambda-endpoint` paramètre pour configurer le point de terminaison Lambda local.

```
java -jar StepFunctionsLocal.jar --lambda-endpoint http://127.0.0.1:3001 command
```

Pour plus d'informations sur l'exécution de Step Functions Local avecAWS Lambda, consultezTester les fonctions de l'étape et l'AWS SAMinterface de ligne de commande locale.

Tester les fonctions de l'étape et l'AWS SAMinterface de ligne de commande locale

En utilisant les deux AWS Step Functions et AWS Lambda en les exécutant sur votre machine locale, vous pouvez tester votre machine d'état et les fonctions Lambda sans y déployer votre code. AWS

Pour plus d'informations, consultez les rubriques suivantes :

- Tester les machines d'état localement
- Configuration d'AWS SAM

Rubriques

Étape 1 : Configurer AWS SAM

AWS Serverless Application Model (AWS SAM) CLI Local nécessite que l'AWS Command Line Interface, AWS SAM et Docker soient installés.

1. Installez l'interface de ligne de commande AWS SAM.

   > **Note**
   >
   > Avant d'installer l'interface de ligne de commande AWS SAM, vous devez installer l'AWS CLI et Docker. Consultez les Prérequis pour l'installation de l'interface de ligne de commande AWS SAM.

2. Lisez la documentation AWS SAM Quick Start. Assurez-vous de suivre les étapes pour effectuer les opérations suivantes :

   1. Initialiser l'application
   2. Tester l'application localement

   Cela crée un `sam-app` répertoire et crée un environnement qui inclut une fonction Lambda Hello World basée sur Python.

Étape 2 : Test de la version locale de l'interface de ligne de commande AWS SAM

Maintenant que vous avez installé AWS SAM et créé la fonction Lambda Hello World, vous pouvez la tester. Dans le répertoire `sam-app`, entrez la commande suivante :

```
sam local start-api
```

Cela lance une instance locale de votre fonction Lambda. Vous devriez voir un résultat semblable à ce qui suit :

```
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)
```

Ouvrez un navigateur et saisissez les informations suivantes :

```
http://127.0.0.1:3000/hello
```

Cela produira une réponse similaire à la suivante :

```
{"message": "hello world", "location": "72.21.198.66"}
```

Entrez CTRL+C pour terminer l'API Lambda.

Étape 3 : Démarrage de la version locale de l'interface de ligne de commande AWS SAM

Maintenant que vous avez vérifié que la fonction est opérationnelle, démarrez la version locale de l'interface de ligne de commande AWS SAM. Dans le répertoire `sam-app`, entrez la commande suivante :

```
sam local start-lambda
```

Cela démarre AWS SAM CLI Local et fournit le point de terminaison à utiliser, comme dans la sortie suivante :

```
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)
```

Étape 4 : Démarrez Step Functions Local

Fichier JAR

Si vous utilisez la version de `.jar` fichier de Step Functions Local, démarrez Step Functions et spécifiez le point de terminaison Lambda. Dans le répertoire où vous avez extrait les `.jar` fichiers, entrez la commande suivante :

```
java -jar StepFunctionsLocal.jar --lambda-endpoint http://localhost:3001
```

Lorsque Step Functions Local démarre, il vérifie l'environnement, puis les informations d'identification configurées dans votre `~/.aws/credentials` fichier. Par défaut, il commence par utiliser un identifiant utilisateur fictif et est répertorié sous la forme. `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

Si vous utilisez la version Docker de Step Functions Local, lancez Step Functions à l'aide de la commande suivante :

```
docker run -p 8083:8083 amazon/aws-stepfunctions-local
```

Pour plus d'informations sur l'installation de la version Docker de Step Functions, consultezConfiguration de Step Functions Local (version téléchargeable) et Docker.

> **Note**
>
> Vous pouvez spécifier le point de terminaison via la ligne de commande ou en définissant des variables d'environnement si vous lancez Step Functions à partir du `.jar` fichier.

Pour la version Docker, vous devez spécifier les points de terminaison et les informations d'identification dans un fichier texte. Consultez Configuration des options de configuration pour Step Functions Local.

Étape 5 : Création d'une machine d'état référençant la fonction de votre version locale de l'interface de ligne de commande AWS SAM

Une fois que Step Functions Local est en cours d'exécution, créez une machine d'état qui fait référence à `HelloWorldFunction` celle dans Étape 1 : Configurer AWS SAM laquelle vous vous êtes initialisé.

```
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"
```

Cela créera une machine d'état et fournira un Amazon Resource Name (ARN) que vous pourrez utiliser pour démarrer une exécution.

```
{
    "creationDate": 1548805711.403,
    "stateMachineArn": "arn:aws:states:us-east-1:123456789012:stateMachine:HelloWorld"
}
```

Étape 6 : Démarrer une exécution de votre machine d'état locale

Une fois que vous avez créé une machine d'état, lancez une exécution. Vous devez référencer l'ARN du point de terminaison et de la machine d'état lorsque vous utilisez la **`aws stepfunctions`** commande suivante :

```
aws stepfunctions --endpoint http://localhost:8083 start-execution --state-machine
 arn:aws:states:us-east-1:123456789012:stateMachine:HelloWorld --name test
```

Cela lance une exécution nommée en fonction `test` de votre machine `HelloWorld` d'état.

```
{
    "startDate": 1548810641.52,
    "executionArn": "arn:aws:states:us-east-1:123456789012:execution:HelloWorld:test"
}
```

Maintenant que Step Functions s'exécute localement, vous pouvez interagir avec elle à l'aide duAWS CLI. Par exemple, pour obtenir des informations sur cette exécution, utilisez la commande suivante :

```
aws stepfunctions --endpoint http://localhost:8083 describe-execution --execution-arn
 arn:aws:states:us-east-1:123456789012:execution:HelloWorld:test
```

L'`describe-execution`appel à une exécution fournit des détails plus complets, similaires à la sortie suivante :

```
{
    "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": "{}"
}
```

Utilisation d'intégrations de services simulées

Dans Step Functions Local, vous pouvez tester les chemins d'exécution de vos machines d'état sans réellement appeler de services intégrés en utilisant des intégrations de services simulées. Pour configurer vos machines d'état afin d'utiliser des intégrations de services simulées, vous devez créer un fichier de configuration fictif. Dans ce fichier, vous définissez le résultat souhaité de vos intégrations de services sous forme de réponses simulées et les exécutions qui utilisent vos réponses simulées pour simuler un chemin d'exécution sous forme de cas de test.

En fournissant le fichier de configuration fictif à Step Functions Local, vous pouvez tester les appels d'intégration de services en exécutant des machines d'état qui utilisent les réponses simulées spécifiées dans les scénarios de test au lieu de passer de véritables appels d'intégration de services.

> **Note**
>
> Si vous ne spécifiez pas de réponses d'intégration de service simulées dans le fichier de configuration fictif, Step Functions Local invoquera l'intégration de AWS services en utilisant le point de terminaison que vous avez configuré lors de la configuration de Step Functions Local. Pour plus d'informations sur la configuration des points de terminaison pour Step Functions Local, consultezConfiguration des options de configuration pour Step Functions Local.

Rubriques

Concepts clés de cette rubrique

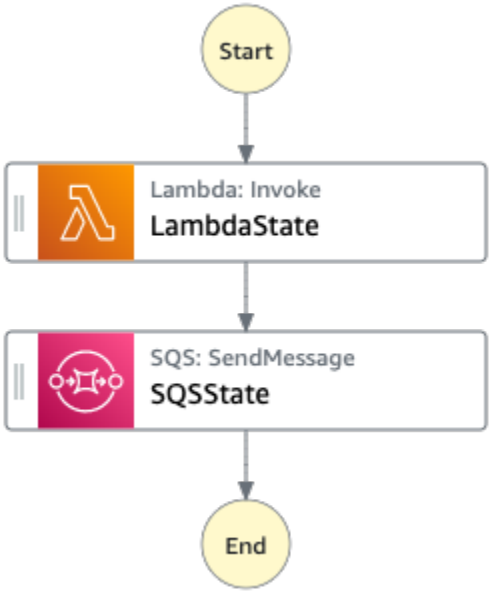
Cette rubrique utilise plusieurs concepts définis dans la liste suivante :

- Intégrations de services simulées - Fait référence aux états des tâches configurés pour utiliser des réponses simulées au lieu d'effectuer de véritables appels de service.

- Réponses simulées - Fait référence aux données fictives que les états des tâches peuvent être configurés pour utiliser.
- Cas de test : fait référence aux exécutions automatiques configurées pour utiliser des intégrations de services simulées.
- Fichier de configuration fictif - Fait référence au fichier de configuration fictif contenant du JSON, qui définit les intégrations de services simulées, les réponses simulées et les cas de test.

Étape 1 : Spécifier les intégrations de services simulées dans un fichier de configuration fictif

Vous pouvez tester le AWS SDK Step Functions et les intégrations de services optimisées à l'aide de Step Functions Local. L'image suivante montre la machine à états définie dans l'onglet Définition de la machine à états :



Pour ce faire, vous devez créer un fichier de configuration fictif contenant les sections définies dansPrésentation de la structure de la configuration fictive.

1. Créez un fichier nommé `MockConfigFile.json` pour configurer les tests avec des intégrations de services simulées.

   L'exemple suivant montre un fichier de configuration fictif faisant référence à une machine à états avec deux états définis nommés `LambdaState` et`SQSState`.

Mock configuration file example

Voici un exemple de fichier de configuration fictif qui montre comment simuler des réponses suite à l'appel d'une fonction Lambda et à l'envoi d'un message à Amazon SQS. Dans cet exemple, la machine à `LambdaSQSIntegration`états contient trois cas de test nommés `HappyPath``RetryPath`, et `HybridPath` qui simulent les `Task` états nommés `LambdaState` et`SQSState`. Ces états utilisent les réponses de `MockedLambdaSuccess` service `MockedSQSSuccess` simulées et les `MockedLambdaRetry` réponses de service simulées. Ces réponses de service simulées sont définies dans la `MockedResponses` section du fichier.

```
{
  "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

Voici un exemple de définition de machine à états appeléeLambdaSQSIntegration, qui définit deux états de tâches d'intégration de services nommés LambdaState etSQSState. LambdaStatecontient une politique de nouvelle tentative basée surStates.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
    }
  }
}
```

Vous pouvez exécuter la définition de machine à `LambdaSQSIntegration` états référencée dans le fichier de configuration fictif en utilisant l'un des cas de test suivants :

- `HappyPath`- Ce test simule la sortie de `LambdaState` et l'`SQSState`utilisation de `MockedLambdaSuccess` et `MockedSQSSuccess` respectivement.
  - Le `LambdaState` renverra la valeur suivante :

```
"0":{
  "Return":{
    "StatusCode":200,
    "Payload":{
      "StatusCode":200,
      "body":"Hello from Lambda!"
    }
  }
}
```

  - Le `SQSState` renverra la valeur suivante :

```
"0":{
  "Return":{
    "MD5OfMessageBody":"3bcb6e8e-7h85-4375-b0bc-1a59812c6e51",
    "MessageId":"3bcb6e8e-8b51-4375-b0bc-1a59812c6e51"
  }
}
```

- `RetryPath`- Ce test simule la sortie de `LambdaState` et l'`SQSState`utilisation de `MockedLambdaRetry` et `MockedSQSSuccess` respectivement. En outre, `LambdaState` est configuré pour effectuer quatre tentatives de nouvelle tentative. Les réponses simulées pour ces tentatives sont définies et indexées dans l'`MockedLambdaRetry`état.
  - La tentative initiale se termine par un échec de tâche contenant un message de cause et d'erreur, comme illustré dans l'exemple suivant :

```
"0":{
  "Throw": {
    "Error": "Lambda.ResourceNotReadyException",
    "Cause": "Lambda resource is not ready."
  }
}
```

- La première et la deuxième tentative se terminent par un échec de tâche contenant un message de cause et d'erreur, comme illustré dans l'exemple suivant :

```
"1-2":{
  "Throw": {
    "Error": "Lambda.TimeoutException",
    "Cause": "Lambda timed out."
  }
}
```

- La troisième tentative se termine par une tâche réussie contenant le résultat d'état de la section Payload dans la réponse Lambda simulée.

```
"3":{
  "Return": {
    "StatusCode": 200,
    "Payload": {
      "StatusCode": 200,
      "body": "Hello from Lambda!"
    }
  }
}
```

> **Note**
>
> - Pour les États dotés d'une politique de nouvelles tentatives, Step Functions Local épuisera le nombre de tentatives définies dans la politique jusqu'à ce qu'il reçoive une réponse positive. Cela signifie que vous devez indiquer les simulations pour les tentatives avec des numéros de tentatives consécutifs et que vous devez couvrir toutes les tentatives avant de renvoyer une réponse positive.
> - Si vous ne spécifiez pas de réponse simulée pour une nouvelle tentative spécifique, par exemple, réessayez « 3 », l'exécution de la machine à états échouera.

- `HybridPath`- Ce test simule le résultat de. `LambdaState` Une fois l'`LambdaState`exécution réussie et la réception de données simulées en réponse, `SQSState` effectue un véritable appel de service à la ressource spécifiée en production.

Pour plus d'informations sur la façon de démarrer des exécutions de tests avec des intégrations de services simulées, consultez. Étape 3 : Exécuter des tests d'intégration de services simulés

2. Assurez-vous que la structure des réponses simulées est conforme à la structure des réponses de service réelles que vous recevez lorsque vous passez des appels de service intégrés. Pour plus d'informations sur les exigences structurelles applicables aux réponses simulées, consultezConfiguration d'intégrations de services simulées.

   Dans l'exemple de fichier de configuration fictif précédent, les réponses simulées sont définies `MockedLambdaSuccess` et `MockedLambdaRetry` conformes à la structure des réponses réelles renvoyées par un appel`HelloFromLambda`.

   > **Important**
   >
   > AWS la structure des réponses aux services peut varier d'un service à l'autre. Step Functions Local ne vérifie pas si les structures de réponse simulées sont conformes aux structures de réponse de service réelles. Vous devez vous assurer que vos réponses simulées sont conformes aux réponses réelles avant le test. Pour examiner la structure des réponses du service, vous pouvez soit effectuer les appels de service proprement dits à l'aide de Step Functions, soit consulter la documentation relative à ces services.

Étape 2 : Fournir le fichier de configuration fictif à Step Functions Local

Vous pouvez fournir le fichier de configuration fictif à Step Functions Local de l'une des manières suivantes :

Docker

> **Note**
>
> Si vous utilisez la version Docker de Step Functions Local, vous pouvez fournir le fichier de configuration fictif en utilisant uniquement une variable d'environnement. En outre, vous devez monter le fichier de configuration fictif sur le conteneur Step Functions Local lors du démarrage initial du serveur.

Montez le fichier de configuration fictif dans n'importe quel répertoire du conteneur Step Functions Local. Définissez ensuite une variable d'environnement nommée `SFN_MOCK_CONFIG` qui contient le chemin d'accès au fichier de configuration fictif dans le conteneur. Cette méthode permet au fichier de configuration fictif de porter n'importe quel nom tant que la variable d'environnement contient le chemin et le nom du fichier.

La commande suivante indique le format de démarrage de l'image Docker.

```
docker run -p 8083:8083
--mount type=bind,readonly,source={absolute path to mock config file},destination=/
home/StepFunctionsLocal/MockConfigFile.json
-e SFN_MOCK_CONFIG="/home/StepFunctionsLocal/MockConfigFile.json" amazon/aws-
stepfunctions-local
```

L'exemple suivant utilise la commande pour démarrer l'image Docker.

```
docker run -p 8083:8083
--mount type=bind,readonly,source=/Users/admin/Desktop/workplace/
MockConfigFile.json,destination=/home/StepFunctionsLocal/MockConfigFile.json
-e SFN_MOCK_CONFIG="/home/StepFunctionsLocal/MockConfigFile.json" amazon/aws-
stepfunctions-local
```

JAR File

Utilisez l'une des méthodes suivantes pour fournir le fichier de configuration fictif à Step Functions Local :

- Placez le fichier de configuration fictif dans le même répertoire que`Step FunctionsLocal.jar`. Lorsque vous utilisez cette méthode, vous devez nommer le fichier de configuration fictif`MockConfigFile.json`.
- Dans la session d'exécution de Step Functions Local, définissez une variable d'environnement nommée`SFN_MOCK_CONFIG`, sur le chemin complet du fichier de configuration fictif. Cette méthode permet au fichier de configuration fictif de porter n'importe quel nom tant que la variable d'environnement contient son chemin et son nom de fichier. Dans l'exemple suivant, la `SFN_MOCK_CONFIG` variable est définie pour pointer vers un fichier de configuration fictif nommé`EnvSpecifiedMockConfig.json`, situé dans le `/home/workspace` répertoire.

  ```
  export SFN_MOCK_CONFIG="/home/workspace/EnvSpecifiedMockConfig.json"
  ```

> **Note**
>
> - Si vous ne fournissez pas la variable d'environnement `SFN_MOCK_CONFIG` à Step Functions Local, Step Functions Local essaiera par défaut de lire un fichier de configuration fictif nommé `MockConfigFile.json` dans le répertoire à partir duquel vous avez lancé Step Functions Local.
> - Si vous placez le fichier de configuration fictif dans le même répertoire que la variable d'environnement `Step FunctionsLocal.jar` et que vous la définissez`SFN_MOCK_CONFIG`, Step Functions Local lira le fichier spécifié par la variable d'environnement.

Étape 3 : Exécuter des tests d'intégration de services simulés

Après avoir créé et fourni un fichier de configuration fictif à Step Functions Local, exécutez la machine d'état configurée dans le fichier de configuration fictif à l'aide d'intégrations de services simulées. Vérifiez ensuite les résultats de l'exécution à l'aide d'une action d'API.

1. Créez une machine à états basée sur la définition mentionnée précédemment dans le fichier de configuration fictif.

```
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. Exécutez la machine d'état à l'aide d'intégrations de services simulées.

Pour utiliser le fichier de configuration fictif, effectuez un appel d'[StartExecution](#)API sur une machine d'état configurée dans le fichier de configuration fictif. Pour ce faire, ajoutez le suffixe#*test_name*, à l'ARN de la machine à états utilisé par. `StartExecution` *test_name*est un cas de test configuré pour la machine d'état dans le même fichier de configuration fictif.

La commande suivante est un exemple d'utilisation de la machine à `LambdaSQSIntegration` états et de la configuration fictive. Dans cet exemple, la machine à `LambdaSQSIntegration` états est exécutée à l'aide du `HappyPath` test défini dans[Étape 1 : Spécifier les intégrations de services simulées dans un fichier de configuration fictif](#). Le `HappyPath` test contient la configuration pour l'exécution afin de gérer les appels d'intégration de services fictifs que `SQSState` les États `LambdaState` et les États effectuent à l'aide `MockedLambdaSuccess` des réponses `MockedSQSSuccess` de service simulées.

```
aws stepfunctions start-execution \
    --endpoint http://localhost:8083 \
    --name executionWithHappyPathMockedServices \
    --state-machine arn:aws:states:us-
east-1:123456789012:stateMachine:LambdaSQSIntegration#HappyPath
```

3. Consultez la réponse d'exécution de la machine à états.

   La réponse à un appel `StartExecution` utilisant un test d'intégration de service simulé est identique à la réponse à un appel `StartExecution` normal, qui renvoie l'ARN d'exécution et la date de début.

   Voici un exemple de réponse à un appel `StartExecution` utilisant le test d'intégration de services simulé :

```
{
   "startDate":"2022-01-28T15:03:16.981000-05:00",
   "executionArn":"arn:aws:states:us-
east-1:123456789012:execution:LambdaSQSIntegration:executionWithHappyPathMockedServices"
}
```

4. Vérifiez les résultats de l'exécution en effectuant un appel [ListExecutionsDescribeExecution](#), ou [GetExecutionHistory](#) API.

```
aws stepfunctions get-execution-history \
```

```
    --endpoint http://localhost:8083 \
    --execution-arn arn:aws:states:us-
east-1:123456789012:execution:LambdaSQSIntegration:executionWithHappyPathMockedServices
```

L'exemple suivant illustre certaines parties d'une réponse à un appel `GetExecutionHistory` utilisant l'ARN d'exécution à partir de l'exemple de réponse présenté à l'étape 2. Dans cet exemple, la sortie de `LambdaState` et `SQSState` correspond aux données fictives définies dans `MockedLambdaSuccess` et `MockedSQSSuccess` dans le fichier de configuration fictif. En outre, les données simulées sont utilisées de la même manière que les données renvoyées lors de l'exécution d'appels d'intégration de services réels. De plus, dans cet exemple, la sortie de `LambdaState` est transmise `SQSState` en entrée.

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

Fichier de configuration pour les intégrations de services simulées

Pour utiliser des intégrations de services simulées, vous devez d'abord créer un fichier de configuration fictif nommé `MockConfigFile.json` contenant vos configurations fictives. Fournissez ensuite à Step Functions Local le fichier de configuration fictif. Ce fichier de configuration définit des scénarios de test, qui contiennent des états fictifs utilisant des réponses d'intégration de services

simulées. La section suivante contient des informations sur la structure de la configuration fictive, y compris les états fictifs et les réponses simulées :

Rubriques

Présentation de la structure de la configuration fictive

Une configuration fictive est un objet JSON contenant les champs de niveau supérieur suivants :

- `StateMachines`- Les champs de cet objet représentent les machines d'État configurées pour utiliser des intégrations de services simulées.
- `MockedResponse`- Les champs de cet objet représentent des réponses simulées pour des appels d'intégration de services.

Voici un exemple de fichier de configuration fictif qui inclut une `StateMachine` définition et`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!"
```

```
          }
        }
      }
    }
  }
}
```

Référence de champ de configuration fictive

Les sections suivantes décrivent les champs d'objet de niveau supérieur que vous devez définir dans votre configuration fictive.

- StateMachines
- MockedResponses

StateMachines

L'`StateMachines`objet définit les machines d'État qui utiliseront des intégrations de services simulées. La configuration de chaque machine d'état est représentée sous la forme d'un champ de niveau supérieur de`StateMachines`. Le nom du champ est le nom de la machine d'état et la valeur est un objet contenant un seul champ nommé`TestCases`, dont les champs représentent des cas de test de cette machine d'état.

La syntaxe suivante montre une machine d'état avec deux scénarios de test :

```
"MyStateMachine": {
  "TestCases": {
    "HappyPath": {
      ...
    },
    "SadPath": {
      ...
    }
  }
```

TestCases

Les champs de `TestCases` représentent des cas de test individuels pour la machine d'État. Le nom de chaque scénario de test doit être unique par machine d'état et la valeur de chaque scénario de test est un objet spécifiant une réponse simulée à utiliser pour les états des tâches dans la machine d'état.

L'exemple de A suivant `TestCase` lie deux `Task` états à deux `MockedResponses` :

```
"HappyPath": {
  "SomeTaskState": "SomeMockedResponse",
  "AnotherTaskState": "AnotherMockedResponse"
}
```

MockedResponses

`MockedResponses`est un objet contenant plusieurs objets de réponse simulés avec des noms de champs uniques. Un objet de réponse simulé définit le résultat réussi ou la sortie d'erreur pour chaque appel d'un état de tâche simulé. Vous spécifiez le numéro d'appel à l'aide de chaînes entières individuelles, telles que « 0 », « 1 », « 2 » et « 3 », ou d'une plage complète d'entiers, telle que « 0-1 », « 2-3 ».

Lorsque vous simulez une tâche, vous devez spécifier une réponse simulée pour chaque appel. Une réponse doit contenir un seul champ nommé `Return` ou `Throw` dont la valeur est le résultat ou la sortie d'erreur pour l'appel de tâche simulé. Si vous ne spécifiez pas de réponse simulée, l'exécution de la machine d'état échouera.

Voici un exemple d'un `MockedResponse` avec `Throw` et d'`Return`objets. Dans cet exemple, les trois premières fois que la machine d'état est exécutée, la réponse spécifiée dans `"0-2"` est renvoyée, et la quatrième fois que la machine d'état s'exécute, la réponse spécifiée dans `"3"` est renvoyée.

```
"SomeMockedResponse": {
  "0-2": {
    "Throw": {
      ...
    }
  },
  "3": {
    "Return": {
      ...
    }
  }
}
```

> **Note**
>
> Si vous utilisez un `Map` état et souhaitez garantir des réponses prévisibles pour `Map` cet état, définissez la valeur `maxConcurrency` sur 1. Si vous définissez une valeur supérieure à 1, Step Functions Local exécutera plusieurs itérations simultanément, ce qui rendra imprévisible l'ordre d'exécution global des états entre les itérations. Cela peut également amener Step Functions Local à utiliser différentes réponses simulées pour les états d'itération d'une exécution à l'autre.

Return

`Return`est représenté sous la forme d'un champ d'`MockedResponse`objets. Il indique le résultat positif d'un état de tâche simulé.

Voici un exemple d'`Return`objet qui contient une réponse simulée pour appeler une `Invoke`fonction Lambda :

```
"Return": {
  "StatusCode": 200,
  "Payload": {
    "StatusCode": 200,
    "body": "Hello from Lambda!"
  }
}
```

Jeter

`Throw`est représenté sous la forme d'un champ d'`MockedResponse`objets. Il indique la sortie d'erreur d'une tâche ayant échoué. La valeur de `Throw` doit être un objet contenant des `Cause` champs `Error` et avec des valeurs de chaîne. En outre, la valeur de chaîne que vous spécifiez dans le `Error` champ `MockConfigFile.json` doit correspondre aux erreurs traitées dans les `Catch` sections `Retry` et de votre machine d'état.

Voici un exemple d'`Throw`objet qui contient une réponse simulée pour appeler une `Invoke`fonction Lambda :

```
"Throw": {
  "Error": "Lambda.TimeoutException",
  "Cause": "Lambda timed out."
```

```
}
```

Configuration d'intégrations de services simulées

Vous pouvez simuler n'importe quelle intégration de service à l'aide de Step Functions Local. Cependant, Step Functions Local ne fait pas en sorte que les simulacres soient les mêmes que les vraies API. Une tâche simulée n'appellera jamais le point de terminaison du service. Si vous ne spécifiez pas de réponse simulée, une tâche tentera d'appeler les points de terminaison du service. En outre, Step Functions Local génère automatiquement un jeton de tâche lorsque vous simulez une tâche à l'aide du `.waitForTaskToken`.

Meilleures pratiques pour Step Functions

Les bonnes pratiques suivantes pour implémenter les flux de travail AWS Step Functions peuvent vous aider à optimiser les performances de vos implémentations.

Rubriques

Utilisez les délais d'attente pour éviter les exécutions bloquées

Par défaut, l'Amazon States Language ne spécifie pas de délais d'expiration pour les définitions des machines à états. En l'absence de délai d'attente explicite, Step Functions s'appuie souvent uniquement sur la réponse d'un intervenant chargé de l'activité pour savoir qu'une tâche est terminée. Si quelque chose ne va pas et que le `TimeoutSeconds` champ n'est pas spécifié pour un `Task` état `Activity` ou, une exécution est bloquée dans l'attente d'une réponse qui n'arrivera jamais.

Pour éviter cette situation, spécifiez un délai d'attente raisonnable lorsque vous créez une machine `Task` dans votre état. Par exemple :

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

Si vous utilisez un rappel avec un jeton de tâche (. waitForTaskToken), nous vous recommandons d'utiliser les pulsations cardiaques et d'ajouter le `HeartbeatSeconds` champ dans la définition de votre `Task` état. Vous pouvez le régler `HeartbeatSeconds` pour qu'il soit inférieur au délai

d'expiration de la tâche. Ainsi, si votre flux de travail échoue en raison d'une erreur cardiaque, vous savez que c'est à cause de l'échec de la tâche et non du fait que l'exécution de la tâche prend du temps.

```
{
  "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
    }
  }
}
```

Pour plus d'informations, consultez État de la tâche la documentation Amazon States Language.

> **Note**
>
> Vous pouvez définir un délai d'expiration pour votre machine à états en utilisant le `TimeoutSeconds` champ de votre définition de Amazon States Language. Pour plus d'informations, veuillez consulter Structure de la machine d'État.

Utilisez les ARN d'Amazon S3 au lieu de transmettre des charges utiles importantes

Les exécutions qui transmettent de grandes charges utiles de données entre états peuvent être résiliées. Si les données que vous transmettez d'un État à l'autre peuvent atteindre plus de 256 Ko, utilisez Amazon Simple Storage Service (Amazon S3) pour stocker les données, puis analysez le nom Amazon Resource Name (ARN) du bucket dans le paramètre pour obtenir `Payload` le nom du bucket et la valeur clé. Sinon, vous pouvez aussi ajuster votre implémentation de façon à transmettre des charges utiles plus petites dans vos exécutions.

Dans l'exemple suivant, une machine à états transmet une entrée à une AWS Lambda fonction qui traite un fichier JSON dans un compartiment Amazon S3. Après avoir exécuté cette machine à états, la fonction Lambda lit le contenu du fichier JSON et renvoie le contenu du fichier en sortie.

Créer la fonction Lambda

La fonction Lambda nommée ci-dessous *pass-large-payload* lit le contenu d'un fichier JSON stocké dans un compartiment Amazon S3 spécifique.

> **Note**
>
> Après avoir créé cette fonction Lambda, assurez-vous de fournir à son rôle IAM l'autorisation appropriée pour lire depuis un compartiment Amazon S3. Par exemple, attachez l'ReadOnlyAccessautorisation AmazonS3 au rôle de la fonction Lambda.

```
import json
import boto3
import io
import os

s3 = boto3.client('s3')

def lambda_handler(event, context):
    event = event['Input']
    final_json = str()

    s3 = boto3.resource('s3')
    bucket = event['bucket'].split(':')[-1]
    filename = event['key']
    directory = "/tmp/{}".format(filename)

    s3.Bucket(bucket).download_file(filename, directory)

    with open(directory, "r") as jsonfile:

        final_json = json.load(jsonfile)

    os.popen("rm -rf /tmp")

    return final_json
```

Création de la machine à états

La machine à états suivante appelle la fonction Lambda que vous avez créée précédemment.

```
{
   "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
      }
   }
}
```

Plutôt que de transmettre une grande quantité de données en entrée, vous pouvez enregistrer ces données dans un compartiment Amazon S3 et transmettre le nom de ressource Amazon (ARN) du compartiment dans le `Payload` paramètre pour obtenir le nom du compartiment et la valeur clé. Votre fonction Lambda peut ensuite utiliser cet ARN pour accéder directement aux données. Voici un exemple d'entrée pour l'exécution de la machine à états, où les données sont stockées `data.json` dans un compartiment Amazon S3 nommé*large-payload-json*.

```
{
  "key": "data.json",
  "bucket": "arn:aws:s3:::large-payload-json"
}
```

Évitez d'atteindre le quota d'historique

AWS Step Functionsdispose d'un quota strict de 25 000 entrées dans l'historique des événements d'exécution. Lorsqu'une exécution atteint 24 999 événements, elle attend que le prochain événement se produise.

- Si le numéro d'événement 25 000 est égal à 25 000`ExecutionSucceeded`, l'exécution se termine correctement.
- Si le numéro d'événement 25 000 ne l'est pas`ExecutionSucceeded`, l'`ExecutionFailed`événement est enregistré et l'exécution de la machine à états échoue car la limite d'historique a été atteinte

Pour éviter d'atteindre ce quota pour les exécutions de longue durée, vous pouvez essayer l'une des solutions suivantes :

- Utilisez l'état de la carte en mode distribué. Dans ce mode, l'`Map`état exécute chaque itération en tant qu'exécution d'un flux de travail enfant, ce qui permet une simultanéité élevée de jusqu'à 10 000 exécutions parallèles de flux de travail enfant. Chaque exécution de flux de travail enfant possède son propre historique d'exécution distinct de celui du flux de travail parent.
- Démarrez une nouvelle exécution par machine à états directement à partir de l'`Task`état d'une exécution en cours. Pour démarrer de telles exécutions de flux de travail imbriqués, utilisez l'action `StartExecution` API de Step Functions dans la machine d'état parent avec les paramètres nécessaires. Pour plus d'informations sur l'utilisation de flux de travail imbriqués, voir Démarrer les exécutions de flux de travail à partir d'un état de tâche ou Utiliser une action d'API Step Functions pour poursuivre un nouveau didacticiel d'exécution.

  > **Tip**
  >
  > Pour déployer un exemple de flux de travail imbriqué sur votreCompte AWS, consultez le Module 13 - Flux de travail express imbriqués.

- Implémentez un modèle qui utilise une AWS Lambda fonction capable de démarrer une nouvelle exécution de votre machine à états afin de répartir le travail en cours entre plusieurs exécutions de flux de travail. Pour plus d'informations, consultez le didacticiel Utilisation d'une fonction Lambda pour poursuivre une nouvelle exécution.

Gérer les exceptions du service Lambda

AWS Lambda peut occasionnellement rencontrer des erreurs de service transitoires. Dans ce cas, l'invocation de Lambda entraîne une erreur 500, telle `ClientExecutionTimeoutException` que, `ServiceExceptionAWSLambdaException`, ou. `SdkClientException` Il est recommandé de

gérer ces exceptions de manière proactive dans votre machine à états avant d'`Retry`appeler votre fonction Lambda ou de corriger l'erreur. `Catch`

Les erreurs Lambda sont signalées sous forme de. `Lambda.`*`ErrorName`* Pour réessayer une erreur d'exception du service Lambda, vous pouvez utiliser le `Retry` code suivant.

```
"Retry": [ {
   "ErrorEquals": [ "Lambda.ClientExecutionTimeoutException",
 "Lambda.ServiceException", "Lambda.AWSLambdaException", "Lambda.SdkClientException"],
   "IntervalSeconds": 2,
   "MaxAttempts": 6,
   "BackoffRate": 2
} ]
```

> **Note**
>
> Les erreurs non gérées dans Lambda sont signalées `Lambda.Unknown` comme dans le résultat d'erreur. Il s'agit notamment out-of-memory des erreurs et des délais d'expiration des fonctions. Vous pouvez effectuer une correspondance ou `States.TaskFailed` pour gérer ces erreurs. `Lambda.Unknown` `States.ALL` Lorsque Lambda atteint le nombre maximum d'appels, l'erreur est. `Lambda.TooManyRequestsException` Pour plus d'informations sur les erreurs liées aux fonctions Lambda, consultez la section Gestion des erreurs et tentatives automatiques dans le Guide du AWS Lambda développeur.

Pour plus d'informations, consultez les ressources suivantes :

- Réessayer après une erreur
- Gestion des conditions d'erreur à l'aide d'une machine à états Step Functions
- Erreurs d'appel Lambda

Évitez la latence lorsque vous interrogez pour des tâches d'activité

L'API `GetActivityTask` est conçue pour fournir un `taskToken` exactement une fois. Si un `taskToken` est supprimé lors de la communication avec un travail d'activité, un certain nombre de demandes `GetActivityTask` peuvent être bloquées pendant 60 secondes dans l'attente d'une réponse avant expiration de `GetActivityTask`.

Si vous avez seulement un petit nombre d'interrogations en attente de réponse, il est possible que toutes les demandes soient placées en file d'attente derrière la demande bloquée et s'arrêtent. Toutefois, si vous avez un grand nombre de sondages en suspens pour chaque activité Amazon Resource Name (ARN) et qu'un certain pourcentage de vos demandes sont en attente, il y en aura encore beaucoup d'autres qui pourront encore obtenir un résultat `taskToken` et commencer à traiter.

Pour les systèmes de production, nous recommandons au moins 100 interrogations ouvertes par ARN d'activité à un moment donné. Si une interrogation est bloquée et qu'une partie de ces interrogations sont placées en file d'attente derrière celle-ci, un grand nombre de demandes recevra encore un `taskToken` pour poursuivre le travail tandis que la demande `GetActivityTask` est bloquée.

Pour éviter ces problèmes de latence lors de la recherche de tâches :

- Implémentez les observateurs sous forme de threads distincts du travail dans l'implémentation de votre travail d'activité.
- Ayez au moins 100 interrogations ouvertes par ARN d'activité à un moment donné.

> **Note**
>
> Le passage à 100 interrogations ouvertes par ARN peut être coûteux. Par exemple, l'interrogation de 100 fonctions Lambda par ARN coûte 100 fois plus cher que d'avoir une seule fonction Lambda avec 100 threads d'interrogation. Pour réduire la latence et minimiser les coûts, utilisez un langage qui comporte des E/S asynchrones et implémentez plusieurs threads d'interrogation par unité de travail. Pour obtenir un exemple de travail d'activité où les threads de l'outil d'interrogation sont distincts des threads de travail, consultez Exemple d'activité de travail en Ruby.

Pour plus d'informations sur les activités et les travaux d'activité, consultez Activités.

Choisir des workflows Standard ou Express

AWS Step Functions propose des workflows Standard comme type de workflow par défaut, avec l'option de choisir des workflows Express.

Vous pouvez choisir des flux de travail standard lorsque vous avez besoin de flux de travail durables, durables et contrôlables, ou des flux de travail express pour des charges de travail de traitement

d'événements à volume élevé. Les exécutions de votre machine d'état se comportent différemment, en fonction du `Type` que vous sélectionnez. Le `Type` choisi ne peut pas être modifié après la création de votre machine d'état.

- Pour des informations détaillées sur les différences entre les flux de travail standard et express, consultezFlux de travail standard ou express.
- Pour plus d'informations sur l'optimisation des coûts lors de la création de flux de travail sans serveur à l'aide de Step Functions, consultezOptimisation des coûts à l'aide d'Express Workflows.

Politique relative CloudWatch aux ressources et restrictions de taille d'Amazon Logs

CloudWatch Les politiques relatives aux ressources des journaux sont limitées à 5 120 caractères. Lorsque CloudWatch Logs détecte qu'une politique approche cette limite de taille, elle active automatiquement les groupes de journaux commençant par`/aws/vendedlogs/`.

Lorsque vous créez une machine à états avec la journalisation activée, Step Functions doit mettre à jour votre politique de ressources de CloudWatch journaux avec le groupe de journaux que vous spécifiez. Pour éviter d'atteindre la limite de taille de la politique de gestion des CloudWatch journaux, préfixez les noms de vos groupes de CloudWatch journaux de journaux par`/aws/vendedlogs/`. Lorsque vous créez un groupe de journaux dans la console Step Functions, les noms des groupes de journaux sont préfixés par`/aws/vendedlogs/states`. Pour plus d'informations, consultez la section Activation de la journalisation à partir de certains AWS services.

Si vous ne parvenez pas à accéder aux CloudWatch journaux, reportez-vous à la section Unable to access the CloudWatch Logs pour plus d'informations.

Utilisation AWS Step Functions avec d'autres services

Découvrez comment coordonner d'autres API Services AWS ou appeler des API tierces avec AWS Step Functions.

Rubriques

Appelez d'autres AWS services

Grâce aux intégrations de AWS services, vous pouvez appeler des actions d'API et coordonner les exécutions directement depuis votre flux de travail. Vous pouvez utiliser les intégrations du AWS SDK de Step Functions pour appeler l'un des plus de deux cents AWS services directement depuis votre machine d'état, ce qui vous donne accès à plus de neuf mille actions d'API. Vous pouvez également utiliser les intégrations optimisées de Step Functions, dont chacune a été personnalisée pour fournir des fonctionnalités spécifiques à votre flux de travail. Certaines actions d'API sont disponibles dans les deux types d'intégration. Dans la mesure du possible, nous vous recommandons d'utiliser l'intégration optimisée.

Vous coordonnez ces services directement depuis un `Task` État dans l'Amazon States Language. Par exemple, à l'aide de Step Functions, vous pouvez appeler d'autres services pour :

- Invoquez une AWS Lambda fonction.
- Exécutez une AWS Batch tâche, puis effectuez différentes actions en fonction des résultats.
- Insérez ou récupérez un élément depuis Amazon DynamoDB.
- Exécutez une tâche Amazon Elastic Container Service (Amazon ECS) et attendez qu'elle soit terminée.
- Publiez sur un sujet dans Amazon Simple Notification Service (Amazon SNS).

- Envoyez un message dans Amazon Simple Queue Service (Amazon SQS).
- Gérez une offre d'emploi pour Amazon AWS Glue ou pour Amazon SageMaker.
- Créez des flux de travail pour exécuter des tâches Amazon EMR.
- Lancez l'exécution d'un AWS Step Functions flux de travail.

Intégrations optimisées

Les intégrations optimisées ont été personnalisées par Step Functions afin de fournir des fonctionnalités spéciales pour un contexte de flux de travail. Par exemple, `Lambda Invoke` convertit la sortie de son API d'un JSON échappé en un objet JSON. AWS Batch`SubmitJob`vous permet de suspendre l'exécution jusqu'à ce que le travail soit terminé. Le premier ensemble d'intégrations optimisées a été publié en 2018, et il existe désormais plus de cinquante API.

AWS Intégrations du SDK

AWS Les intégrations du SDK fonctionnent exactement comme un appel d'API standard utilisant le AWS SDK. Ils permettent d'appeler plus de neuf mille API dans plus de deux cents AWS services directement à partir de la définition de votre machine d'état.

Support des modèles d'intégration

Les flux de travail standard et les flux de travail express prennent en charge les mêmes intégrations, mais pas les mêmes modèles d'intégration.

- La prise en charge des modèles d'intégration optimisés est différente pour chaque intégration.
- Express Workflows ne prend pas en charge Run a Job (.sync) ou Wait for Callback (.waitForTaskJeton).
- Pour plus d’informations, consultez Flux de travail standard ou express.

Standard Workflows

Intégrations de services prises en charge

	Service	Réponse à la requête	Exécuter une tâche (`.sync`)	Attendre le rappel (`.waitForTaskToken`)
Intégrations optimisées	Amazon API Gateway	✓		✓
	Amazon Athena	✓	✓	
	AWS Batch	✓	✓	
	Amazon Bedrock	✓	✓	✓
	AWS CodeBuild	✓	✓	
	Amazon DynamoDB	✓		
	Amazon ECS/Fargate	✓	✓	✓
	Amazon EKS	✓	✓	✓
	Amazon EMR	✓	✓	
	Amazon EMR on EKS	✓	✓	
	Amazon EMR Serverless	✓	✓	
	Amazon EventBridge	✓		✓
	AWS Glue	✓	✓	
	AWS Glue DataBrew	✓	✓	
	AWS Lambda	✓		✓
	AWS Elemental MediaConvert	✓	✓	
	Amazon SageMaker	✓	✓	

	Service	Réponse à la requête	Exécuter une tâche (`.sync`)	Attendre le rappel (`.waitForTaskToken`)
	Amazon SNS	✓		✓
	Amazon SQS	✓		✓
	AWS Step Functions	✓	✓	✓
AWS Intégrations du SDK	Plus de deux cents	✓		✓

Express Workflows

Intégrations de services prises en charge

	Service	Réponse à la requête	Exécuter une tâche (`.sync`)	Attendre le rappel (`.waitForTaskToken`)
Intégrations optimisées	Amazon API Gateway	✓		
	Amazon Athena	✓		
	AWS Batch	✓		
	Amazon Bedrock	✓		
	AWS CodeBuild	✓		
	Amazon DynamoDB	✓		
	Amazon ECS/Fargate	✓		
	Amazon EKS	✓		

	Service	Réponse à la requête	Exécuter une tâche (`.sync`)	Attendre le rappel (`.waitForTaskToken`)
	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 Intégrations du SDK	Plus de deux cents	✓		

Accès intercomptes

Step Functions fournit un accès entre comptes aux ressources configurées selon différents flux Comptes AWS de travail. Grâce aux intégrations de services Step Functions, vous pouvez invoquer n'importe quelle AWS ressource entre comptes, même si celle-ci Service AWS ne prend pas en charge les politiques basées sur les ressources ou les appels entre comptes.

Pour plus d'informations, consultez Accès à des ressources Comptes AWS dans d'autres flux de travail.

AWS Intégrations de services SDK

AWS Step Functions s'intègre à Services AWS, vous permettant d'appeler les actions d'API de chaque service depuis votre flux de travail. Vous pouvez utiliser les intégrations du AWS SDK de Step Functions pour appeler presque toutes les actions Service AWS d'API depuis votre machine à états. Vous pouvez également utiliser les intégrations optimisées de Step Functions, dont chacune a été personnalisée pour fournir des fonctionnalités spéciales pour votre flux de travail.

Certains services ou SDK peuvent ne pas être disponibles sous forme d'intégrations de AWS SDK, que ce soit temporairement ou définitivement. Les interactions avec le SDK ne seront peut-être pas disponibles pour les services récemment publiés avant une mise à jour ultérieure. Certains services nécessitent une configuration personnalisée, telle que la spécification d'un point de terminaison spécifique au client, qui peut être plus adapté à une intégration optimisée. Les autres SDK ne sont pas adaptés à une utilisation dans un flux de travail, tels que ceux destinés au streaming audio ou vidéo. Enfin, certains services peuvent être suspendus jusqu'à ce qu'ils passent certaines validations internes effectuées par Step Functions.

> **Tip**
>
> Pour déployer un exemple de flux de travail utilisant des intégrations de AWS SDK dans votre application Compte AWS, consultez le module 9 - Intégrations de services AWS SDK dans The Workshop. AWS Step Functions

Rubriques

Utilisation des AWS intégrations de services du SDK

Pour utiliser les intégrations du AWS SDK, vous devez spécifier le nom du service et l'appel d'API et, éventuellement, un modèle d'intégration de service.

> **Note**
>
> - Les paramètres Step Functions sont exprimés en PascalCase, même si l'API de service native est dans CamelCase. Par exemple, vous pouvez utiliser l'action Step Functions API `startSyncExecution` et spécifier son paramètre sous la forme`StateMachineArn`.
> - Pour les actions d'API qui acceptent des paramètres énumérés, telles que l'action d'`DescribeLaunchTemplateVersions`API pour Amazon EC2, spécifiez une version au pluriel du nom du paramètre. Par exemple, spécifiez `Filters` le `Filter.N` paramètre de l'action d'`DescribeLaunchTemplateVersions`API.

Vous pouvez appeler les services du AWS SDK directement depuis le langage Amazon States dans le `Resource` champ de l'état d'une tâche. Pour ce faire, utilisez la syntaxe suivante :

`arn:aws:states:::aws-sdk:`*`serviceName`*`:`*`apiAction`*`.`*`[serviceIntegrationPattern]`*

Par exemple, pour Amazon EC2, vous pouvez utiliser. `arn:aws:states:::aws-sdk:ec2:describeInstances` Cela renverrait le résultat tel que défini pour l'appel d'API Amazon EC2 DescribeInstances.

Si une intégration du AWS SDK rencontre une erreur, le champ d'erreur résultant sera composé du nom du service et du nom de l'erreur, séparés par un point :*`ServiceName`*`.`*`ErrorName`*. Le nom du service et le nom de l'erreur sont tous deux en cas de Pascal. Vous pouvez également voir le nom du service, en minuscules, dans le champ Ressource de l'état de la tâche. Vous trouverez les noms des erreurs potentielles dans la documentation de référence de l'API du service cible.

Par exemple, vous pouvez utiliser l'intégration du `arn:aws:states:::aws-sdk:acmpca:deleteCertificateAuthority` AWS SDK. La référence AWS Private Certificate Authority d'API indique que l'action d'`DeleteCertificateAuthority`API peut entraîner, par exemple`ResourceNotFoundException`, un. Pour gérer cette erreur, vous devez donc spécifier l'erreur `AcmPca.ResourceNotFoundException` dans les récupérateurs ou les capteurs de votre état de tâche. Pour plus d'informations sur la gestion des erreurs dans Step Functions, consultezGestion des erreurs dans Step Functions.

Step Functions ne peut pas générer automatiquement de politiques IAM pour les intégrations de AWS SDK. Après avoir créé votre machine d'état, vous devez accéder à la console IAM et configurer vos politiques de rôle. Pour plus d'informations, consultez Politiques IAM pour les services intégrés.

Consultez le Collectez des informations sur le compartiment Amazon S3 à l'aide des AWS intégrations de services du SDK didacticiel pour découvrir un exemple d'utilisation des intégrations de AWS SDK.

Intégrations de services AWS SDK prises en charge

Le tableau suivant répertorie les intégrations de services du AWS SDK prises en charge par Step Functions. La colonne des ressources d'*Task*état répertorie la syntaxe permettant d'appeler une action d'API spécifique lors de l'utilisation de l'intégration pour le service spécifié dans la colonne Nom du service sur la gauche. La colonne Date prise en charge fournit des informations sur les dates auxquelles l'intégration du service a été prise en charge. En outre, la colonne Préfixe d'exception sur la droite répertorie les préfixes d'exception pour chaque intégration de services. Ces préfixes d'exception sont présents dans les exceptions générées lorsque vous effectuez par erreur une intégration de service AWS SDK avec Step Functions.

> **Note**
>
> - Les services marqués d'un*** comportent des actions d'API qui ne sont pas prises en charge par Step Functions pour le moment. Pour plus d'informations sur les actions qui ne sont pas prises en charge pour un service, consultez le Actions d'API non prises en charge pour les services pris en charge tableau.
> - Pour plus d'informations sur les mises à jour effectuées à chaque lancement afin d'étendre la prise en charge des intégrations de AWS SDK, consultez. Journal des modifications pour les intégrations de AWS SDK prises en charge

> **Important**
>
> Le support aux actions de l'API est publié tous les trimestres. Les mises à jour des actions déjà prises en charge, telles que les nouveaux paramètres, peuvent ne pas être disponibles immédiatement.

Nom du service	**Task**ressource de l'État	Date prise en charge	Préfixe d'exception
AWS AppFabric	`arn:aws:states:::aws-sdk:appfabric:` *`[apiAction]`*	18 janvier 2024	AppFabric
B2B Data Interchange	`arn:aws:states:::aws-sdk:b2bi:` *`[apiAction]`*	18 janvier 2024	B2Bi
Exportations de données AWS	`arn:aws:states:::aws-sdk:bcmdataexports:` *`[apiAction]`*	18 janvier 2024	BcmDataExports
Amazon Bedrock	`arn:aws:states:::aws-sdk:bedrock:` *`[apiAction]`*	18 janvier 2024	Bedrock
Amazon Bedrock Agents	`arn:aws:states:::aws-sdk:bedrockagent:` *`[apiAction]`*	18 janvier 2024	BedrockAgent
Amazon Bedrock Runtime Agents	`arn:aws:states:::aws-sdk:bedrockagentruntime:` *`[apiAction]`*	18 janvier 2024	BedrockAgentRuntime
Amazon Bedrock Runtime	`arn:aws:states:::aws-sdk:bedrockruntime:` *`[apiAction]`*	18 janvier 2024	BedrockRuntime
AWS Clean Rooms	`arn:aws:states:::aws-sdk:cleanroomsml:` *`[apiAction]`*	18 janvier 2024	CleanRoomsMl
Amazon CloudFront KeyValueStore	`arn:aws:states:::aws-sdk:cl`	18 janvier 2024	CloudFrontKeyValueStore

Nom du service	**Task**ressource de l'État	Date prise en charge	Préfixe d'exception
	`oudfrontkeyvaluestore: [apiAction]`		
CodeGuru Security	`arn:aws:states:::aws-sdk:codegurusecurity: [apiAction]`	18 janvier 2024	CodeGuruSecurity
Hub d'optimisation des coûts	`arn:aws:states:::aws-sdk:costoptimizationhub: [apiAction]`	18 janvier 2024	CostOptimizationHub
Amazon DataZone	`arn:aws:states:::aws-sdk:datazone: [apiAction]`	18 janvier 2024	DataZone
Amazon EKS Auth	`arn:aws:states:::aws-sdk:eksauth: [apiAction]`	18 janvier 2024	EksAuth
Résolution des entités AWS	`arn:aws:states:::aws-sdk:entityresolution: [apiAction]`	18 janvier 2024	EntityResolution
Niveau gratuit d'AWS	`arn:aws:states:::aws-sdk:freetier: [apiAction]`	18 janvier 2024	FreeTier
Amazon Inspector Scan	`arn:aws:states:::aws-sdk:inspectorscan: [apiAction]`	18 janvier 2024	InspectorScan
AWS Launch Wizard	`arn:aws:states:::aws-sdk:launchwizard: [apiAction]`	18 janvier 2024	LaunchWizard

Nom du service	**Task**ressource de l'État	Date prise en charge	Préfixe d'exception
Amazon Managed Blockchain Query	`arn:aws:states:::aws-sdk:managedblockchainquery: [apiAction]`	18 janvier 2024	ManagedBlockchainQuery
AWS Elemental MediaPackage V2	`arn:aws:states:::aws-sdk:mediapackagev2: [apiAction]`	18 janvier 2024	MediaPackageV2
AWS HealthImaging	`arn:aws:states:::aws-sdk:medicalimaging: [apiAction]`	18 janvier 2024	MedicalImaging
Network Manager	`arn:aws:states:::aws-sdk:networkmanager: [apiAction]`	18 janvier 2024	NetworkManager
AWS Payment Cryptography	`arn:aws:states:::aws-sdk:paymentcryptography: [apiAction]`	18 janvier 2024	PaymentCryptography
AWS Payment Cryptography Data	`arn:aws:states:::aws-sdk:paymentcryptographydata: [apiAction]`	18 janvier 2024	PaymentCryptographyData
AWS Private CA Connector for Active Directory	`arn:aws:states:::aws-sdk:pcaconnectorad: [apiAction]`	18 janvier 2024	PcaConnectorAd
Amazon Q Business	`arn:aws:states:::aws-sdk:qbusiness: [apiAction]`	18 janvier 2024	QBusiness

Nom du service	**Task**ressource de l'État	Date prise en charge	Préfixe d'exception
Amazon Q Connect	`arn:aws:states:::aws-sdk:qconnect: [apiAction]`	18 janvier 2024	QConnect
AWS re:Post	`arn:aws:states:::aws-sdk:repostspace: [apiAction]`	18 janvier 2024	Repostspace
Amazon Timestream Query	`arn:aws:states:::aws-sdk:timestreamquery: [apiAction]`	18 janvier 2024	TimestreamQuery
Amazon Timestream Write	`arn:aws:states:::aws-sdk:timestreamwrite: [apiAction]`	18 janvier 2024	TimestreamWrite
Trusted Advisor	`arn:aws:states:::aws-sdk:trustedadvisor: [apiAction]`	18 janvier 2024	TrustedAdvisor
Verified Permissions	`arn:aws:states:::aws-sdk:verifiedpermissions: [apiAction]`	18 janvier 2024	VerifiedPermissions
Amazon WorkSpaces Thin Client	`arn:aws:states:::aws-sdk:workspacesthinclient: [apiAction]`	18 janvier 2024	WorkSpacesThinClient
AWS CloudTrail Data	`arn:aws:states:::aws-sdk:cloudtraildata: [apiAction]`	16 juin 2023	CloudTrailData

Nom du service	**Task**ressource de l'État	Date prise en charge	Préfixe d'exception
Amazon CloudWatch Internet Monitor	`arn:aws:states:::aws-sdk:internetmonitor:` *[apiAction]*	16 juin 2023	InternetMonitor
Amazon Interactive Video Service RealTime	`arn:aws:states:::aws-sdk:ivsrealtime:` *[apiAction]*	16 juin 2023	IvsRealTime
AWS IoT TwinMaker	`arn:aws:states:::aws-sdk:iottwinmaker:` *[apiAction]*	16 juin 2023	IoTTwinMaker
Amazon OpenSearch Ingestion	`arn:aws:states:::aws-sdk:osis:` *[apiAction]*	16 juin 2023	Osis
AWS Telco Network Builder	`arn:aws:states:::aws-sdk:` **`tnb`**`:`*[apiAction]*	16 juin 2023	Tnb
Amazon VPC Lattice	`arn:aws:states:::aws-sdk:vpclattice:` *[apiAction]*	16 juin 2023	VpcLattice
Amazon Chime Media Pipelines	`arn:aws:states:::aws-sdk:chimesdkmediapipelines:` *[apiAction]*	17 février 2023	ChimeSdkMediaPipelines
Amazon Chime Voice	`arn:aws:states:::aws-sdk:chimesdkvoice:` *[apiAction]*	17 février 2023	ChimeSdkVoice

Nom du service	**Task**ressource de l'État	Date prise en charge	Préfixe d'exception
Amazon CodeCatalyst	`arn:aws:states:::aws-sdk:codecatalyst:` *[apiAction]*	17 février 2023	CodeCatalyst
Amazon Connect Cases	`arn:aws:states:::aws-sdk:connectcases:` *[apiAction]*	17 février 2023	ConnectCases
Amazon DocumentDB Elastic Clusters	`arn:aws:states:::aws-sdk:docdbelastic:` *[apiAction]*	17 février 2023	DocDbElastic
Amazon EMR Serverless	`arn:aws:states:::aws-sdk:emrserverless:` *[apiAction]*	17 février 2023	EmrServerless
Amazon IVS Chat	`arn:aws:states:::aws-sdk:ivs:` *[apiAction]*	17 février 2023	Ivs
Amazon Kendra Intelligent Ranking	`arn:aws:states:::aws-sdk:kendrarankingng:` *[apiAction]*	17 février 2023	KendraRanking
AWS HealthOmics	`arn:aws:states:::aws-sdk:omics:` *[apiAction]*	17 février 2023	Omics
Amazon Redshift Serverless	`arn:aws:states:::aws-sdk:redshiftserverless:` *[apiAction]*	17 février 2023	RedshiftServerless
Amazon Security Lake	`arn:aws:states:::aws-sdk:securitylake:` *[apiAction]*	17 février 2023	SecurityLake

Nom du service	**Task**ressource de l'État	Date prise en charge	Préfixe d'exception
AWS Backup Storage	`arn:aws:states:::aws-sdk:backupstorage: [apiAction]`	17 février 2023	BackupStorage
AWS Clean Rooms	`arn:aws:states:::aws-sdk:cleanrooms: [apiAction]`	17 février 2023	CleanRooms
AWS Control Tower	`arn:aws:states:::aws-sdk:controltower: [apiAction]`	17 février 2023	ControlTower
AWS Health	`arn:aws:states:::aws-sdk:health: [apiAction]`	17 février 2023	Health
AWS IoT FleetWise	`arn:aws:states:::aws-sdk:iotfleetwise: [apiAction]`	17 février 2023	IotFleetWise
AWS Mainframe Modernization	`arn:aws:states:::aws-sdk:m2: [apiAction]`	17 février 2023	M2
Orchestrateur de l'AWS Migration Hub	`arn:aws:states:::aws-sdk:migrationhuborchestrator: [apiAction]`	17 février 2023	Migration HubOrchestrator
AWS Private 5G	`arn:aws:states:::aws-sdk:privatenetworks: [apiAction]`	17 février 2023	PrivateNetworks

Nom du service	**Task**ressource de l'État	Date prise en charge	Préfixe d'exception
Explorateur de ressources AWS	`arn:aws:states:::aws-sdk:resourceexplorer2: [apiAction]`	17 février 2023	ResourceExplorer2
AWS SimSpace Weaver	`arn:aws:states:::aws-sdk:simspaceweaver: [apiAction]`	17 février 2023	SimSpaceWeaver
AWS Support App	`arn:aws:states:::aws-sdk:supportapp: [apiAction]`	17 février 2023	SupportApp
CloudWatch Observability Access Manager	`arn:aws:states:::aws-sdk:oam: [apiAction]`	17 février 2023	Oam
EventBridge Pipes	`arn:aws:states:::aws-sdk:pipes: [apiAction]`	17 février 2023	Pipes
EventBridge Scheduler	`arn:aws:states:::aws-sdk:scheduler: [apiAction]`	17 février 2023	Scheduler
IAM Roles Anywhere	`arn:aws:states:::aws-sdk:rolesanywhere: [apiAction]`	17 février 2023	RolesAnywhere
Kinesis Video WebRTC Storage	`arn:aws:states:::aws-sdk:kinesisvideowebrtcstorage: [apiAction]`	17 février 2023	KinesisVideoWebRtcStorage

Nom du service	**Task**ressource de l'État	Date prise en charge	Préfixe d'exception
License Manager Linux Subscriptions	`arn:aws:states:::aws-sdk:licensemanagerlinuxsubscriptions: [apiAction]`	17 février 2023	LicenseManagerLinuxSubscriptions
License Manager User Subscriptions	`arn:aws:states:::aws-sdk:licensemanagerusersubscriptions: [apiAction]`	17 février 2023	LicenseManagerUserSubscriptions
OpenSearch Serverless	`arn:aws:states:::aws-sdk:opensearchserverless: [apiAction]`	17 février 2023	OpenSearchServerless
Route 53 ARC Zonal Shift	`arn:aws:states:::aws-sdk:arczonalshift: [apiAction]`	17 février 2023	ArcZonalShift
SageMaker Geospatial	`arn:aws:states:::aws-sdk:sagemakergeospatial: [apiAction]`	17 février 2023	SageMaker Geospatial
SageMaker Metrics	`arn:aws:states:::aws-sdk:sagemakermetrics: [apiAction]`	17 février 2023	SageMakerMetrics
Systems Manager for SAP	`arn:aws:states:::aws-sdk:ssmsap: [apiAction]`	17 février 2023	SsmSap
AWS Account Management	`arn:aws:states:::aws-sdk:account: [apiAction]`	19 avril 2022	Account

Nom du service	**Task**ressource de l'État	Date prise en charge	Préfixe d'exception
AWS Amplify	`arn:aws:states:::aws-sdk:amplify: [apiAction]`	30 septembre 2021	Amplify
AWS App Mesh	`arn:aws:states:::aws-sdk:appmesh: [apiAction]`	30 septembre 2021	AppMesh
AWS App Runner	`arn:aws:states:::aws-sdk:apprunner: [apiAction]`	30 septembre 2021	AppRunner
AWS AppConfig	`arn:aws:states:::aws-sdk:appconfig: [apiAction]`	30 septembre 2021	AppConfig
AWS AppConfig Data	`arn:aws:states:::aws-sdk:appconfigdata: [apiAction]`	19 avril 2022	AppConfigData
AWS AppSync	`arn:aws:states:::aws-sdk:appsync: [apiAction]`	30 septembre 2021	AppSync
AWS Application Discovery Service	`arn:aws:states:::aws-sdk:applicationdiscovery: [apiAction]` ***	30 septembre 2021	ApplicationDiscovery
AWS Application Migration Service	`arn:aws:states:::aws-sdk:mgn: [apiAction]`	30 septembre 2021	Mgn

Nom du service	**Task**ressource de l'État	Date prise en charge	Préfixe d'exception
AWS Audit Manager	`arn:aws:states:::aws-sdk:auditmanager: [apiAction]`	30 septembre 2021	AuditManager
AWS Auto Scaling Plans	`arn:aws:states:::aws-sdk:autoscalingplans: [apiAction]`	30 septembre 2021	AutoScalingPlans
AWS Backup	`arn:aws:states:::aws-sdk:backup: [apiAction]`	30 septembre 2021	Backup
AWS Backup gateway	`arn:aws:states:::aws-sdk:backupgateway: [apiAction]`	19 avril 2022	BackupGateway
AWS Batch	`arn:aws:states:::aws-sdk:batch: [apiAction]`	30 septembre 2021	Batch
AWS Billing Conductor	`arn:aws:states:::aws-sdk:billingconductor: [apiAction]`	26 juillet 2022	Billingconductor
AWS Budgets	`arn:aws:states:::aws-sdk:budgets: [apiAction]`	30 septembre 2021	Budgets
AWS Certificate Manager	`arn:aws:states:::aws-sdk:acm: [apiAction]`	30 septembre 2021	Acm
AWS Private Certificate Authority	`arn:aws:states:::aws-sdk:acmpca: [apiAction]`	30 septembre 2021	AcmPca

Nom du service	**Task**ressource de l'État	Date prise en charge	Préfixe d'exception
AWS Cloud Map	`arn:aws:states:::aws-sdk:servicediscovery: [apiAction]`	30 septembre 2021	ServiceDiscovery
AWS Cloud9	`arn:aws:states:::aws-sdk:cloud9: [apiAction]`	30 septembre 2021	Cloud9
AWS CloudFormation	`arn:aws:states:::aws-sdk:cloudformation: [apiAction]`	30 septembre 2021	CloudFormation
AWS CloudHSM	`arn:aws:states:::aws-sdk:cloudhsm: [apiAction]`	30 septembre 2021	CloudHsm
AWS CloudHSM	`arn:aws:states:::aws-sdk:cloudhsmv2: [apiAction]`	30 septembre 2021	CloudHsmV2
AWS CloudTrail	`arn:aws:states:::aws-sdk:cloudtrail: [apiAction]`	30 septembre 2021	CloudTrail
AWS Cloud Control	`arn:aws:states:::aws-sdk:cloudcontrol: [apiAction]`	19 avril 2022	CloudControl
AWS CodeBuild	`arn:aws:states:::aws-sdk:codebuild: [apiAction]`	30 septembre 2021	CodeBuild
AWS CodeCommit	`arn:aws:states:::aws-sdk:codecommit: [apiAction]`	30 septembre 2021	CodeCommit

Nom du service	**Task**ressource de l'État	Date prise en charge	Préfixe d'exception
AWS CodeDeploy	`arn:aws:states:::aws-sdk:codedeploy:` *`[apiAction]`* ***	30 septembre 2021	CodeDeploy
AWS CodePipeline	`arn:aws:states:::aws-sdk:codepipeline:` *`[apiAction]`*	30 septembre 2021	CodePipeline
AWS CodeStar	`arn:aws:states:::aws-sdk:codestar:` *`[apiAction]`*	30 septembre 2021	CodeStar
AWS CodeStar	`arn:aws:states:::aws-sdk:codestarnotifications:` *`[apiAction]`*	30 septembre 2021	CodestarNotifications
AWS CodeStar	`arn:aws:states:::aws-sdk:codestarconnections:` *`[apiAction]`*	30 septembre 2021	CodeStarConnections
AWS Compute Optimizer	`arn:aws:states:::aws-sdk:computeoptimizer:` *`[apiAction]`*	30 septembre 2021	ComputeOptimizer
AWS Config	`arn:aws:states:::aws-sdk:config:` *`[apiAction]`*	30 septembre 2021	Config
AWS Cost Explorer Service	`arn:aws:states:::aws-sdk:costexplorer:` *`[apiAction]`*	30 septembre 2021	CostExplorer

Nom du service	**Task**ressource de l'État	Date prise en charge	Préfixe d'exception
AWS Cost and Usage Report	`arn:aws:states:::aws-sdk:costandusagereport: [apiAction]`	30 septembre 2021	CostAndUsageReport
AWS Data Exchange	`arn:aws:states:::aws-sdk:dataexchange: [apiAction]`	30 septembre 2021	DataExchange
AWS Data Pipeline	`arn:aws:states:::aws-sdk:datapipeline: [apiAction]`	30 septembre 2021	DataPipeline
AWS DataSync	`arn:aws:states:::aws-sdk:datasync: [apiAction]`	30 septembre 2021	DataSync
AWS Database Migration Service	`arn:aws:states:::aws-sdk:databasemigration: [apiAction]`	30 septembre 2021	DatabaseMigration
AWS Device Farm	`arn:aws:states:::aws-sdk:devicefarm: [apiAction]`	30 septembre 2021	DeviceFarm
AWS Direct Connect	`arn:aws:states:::aws-sdk:directconnect: [apiAction]` ***	30 septembre 2021	DirectConnect
AWS Directory Service	`arn:aws:states:::aws-sdk:directory: [apiAction]`	30 septembre 2021	Directory
AWS EC2 Instance Connect	`arn:aws:states:::aws-sdk:ec2instanceconnect: [apiAction]`	30 septembre 2021	Ec2InstanceConnect

Nom du service	**Task**ressource de l'État	Date prise en charge	Préfixe d'exception
AWS Elastic Beanstalk	`arn:aws:states:::aws-sdk:elasticbeanstalk:` *[apiAction]*	30 septembre 2021	ElasticBeanstalk
AWS Elemental MediaLive	`arn:aws:states:::aws-sdk:medialive:` *[apiAction]*	30 septembre 2021	MediaLive
AWS Elemental MediaPackage	`arn:aws:states:::aws-sdk:mediapackage:` *[apiAction]* *** —	30 septembre 2021	MediaPackage
AWS Elemental MediaPackage VOD	`arn:aws:states:::aws-sdk:mediapackagevod:` *[apiAction]*	30 septembre 2021	MediaPackageVod
AWS Elemental MediaStore	`arn:aws:states:::aws-sdk:mediastore:` *[apiAction]*	30 septembre 2021	MediaStore
AWS Fault Injection Service	`arn:aws:states:::aws-sdk:fis:` *[apiAction]*	30 septembre 2021	Fis
AWS Firewall Manager	`arn:aws:states:::aws-sdk:fms:` *[apiAction]*	30 septembre 2021	Fms
AWS Glue	`arn:aws:states:::aws-sdk:glue:` *[apiAction]*	30 septembre 2021	Glue
AWS Glue DataBrew	`arn:aws:states:::aws-sdk:databrew:` *[apiAction]*	30 septembre 2021	DataBrew

Nom du service	**Task**ressource de l'État	Date prise en charge	Préfixe d'exception
AWS IoT Greengrass	`arn:aws:states:::aws-sdk:greengrass:` *`[apiAction]`*	30 septembre 2021	Greengrass
AWS Ground Station	`arn:aws:states:::aws-sdk:groundstation:` *`[apiAction]`*	30 septembre 2021	GroundStation
AWS Identity and Access Management	`arn:aws:states:::aws-sdk:iam:` *`[apiAction]`*	30 septembre 2021	Iam
AWS IoT	`arn:aws:states:::aws-sdk:iot:` *`[apiAction]`* ***	30 septembre 2021	Iot
AWS IoT 1-Click	`arn:aws:states:::aws-sdk:iot1clickprojects:` *`[apiAction]`*	30 septembre 2021	Iot1ClickProjects
AWS IoT Analytics	`arn:aws:states:::aws-sdk:iotanalytics:` *`[apiAction]`*	30 septembre 2021	IoTAnalytics
AWS IoT Core Device Advisor	`arn:aws:states:::aws-sdk:iotdeviceadvisor:` *`[apiAction]`* ***	30 septembre 2021	IotDeviceAdvisor
AWS IoT Events	`arn:aws:states:::aws-sdk:ioteventsdata:` *`[apiAction]`*	30 septembre 2021	IotEvents
AWS IoT Events Data	`arn:aws:states:::aws-sdk:ioteventsdata:` *`[apiAction]`*	30 septembre 2021	IotEventsData

Nom du service	**Task**ressource de l'État	Date prise en charge	Préfixe d'exception
AWS IoT Fleet Hub	`arn:aws:states:::aws-sdk:iotfleethub: [apiAction]`	30 septembre 2021	IoTFleetHub
AWS IoT Greengrass Version 2	`arn:aws:states:::aws-sdk:greengrassv2: [apiAction]`	30 septembre 2021	GreengrassV2
Données de tâches AWS IoT Plane	`arn:aws:states:::aws-sdk:iotjobsdataplane: [apiAction]`	30 septembre 2021	IotJobsDataPlane
AWS IoT Secure Tunneling	`arn:aws:states:::aws-sdk:iotsecuretunneling: [apiAction]`	30 septembre 2021	IoTSecure Tunneling
AWS IoT SiteWise	`arn:aws:states:::aws-sdk:iotsitewise: [apiAction]`	30 septembre 2021	IoTSiteWise
AWS IoT Wireless	`arn:aws:states:::aws-sdk:iotwireless: [apiAction]`	30 septembre 2021	IotWireless
AWS Key Management Service	`arn:aws:states:::aws-sdk:kms: [apiAction]`	30 septembre 2021	Kms
AWS Lake Formation	`arn:aws:states:::aws-sdk:lakeformation: [apiAction]`	30 septembre 2021	LakeFormation
AWS Lambda	`arn:aws:states:::aws-sdk:lambda: [apiAction]` *** —	30 septembre 2021	Lambda

Nom du service	**Task**ressource de l'État	Date prise en charge	Préfixe d'exception
AWS License Manager	`arn:aws:states:::aws-sdk:licensemanager: [apiAction]`	30 septembre 2021	LicenseManager
AWS Marketplace	`arn:aws:states:::aws-sdk:marketplacecatalog: [apiAction]`	30 septembre 2021	MarketplaceCatalog
AWS Marketplace Commerce Analytics	`arn:aws:states:::aws-sdk:marketplacecommerceanalytics: [apiAction]`	30 septembre 2021	MarketplaceCommerceAnalytics
AWS Marketplace Entitlement Service	`arn:aws:states:::aws-sdk:marketplaceentitlement: [apiAction]`	30 septembre 2021	MarketplaceEntitlement
AWS Elemental MediaTailor	`arn:aws:states:::aws-sdk:mediatailor: [apiAction]`	30 septembre 2021	MediaTailor
AWS Migration Hub	`arn:aws:states:::aws-sdk:migrationhub: [apiAction]`	30 septembre 2021	MigrationHub
AWS Migration Hub Config	`arn:aws:states:::aws-sdk:migrationhubconfig: [apiAction]`	30 septembre 2021	MigrationHubConfig
Recommandations stratégiques d'AWS Migration Hub	`arn:aws:states:::aws-sdk:migrationhubstrategy: [apiAction]`	19 avril 2022	MigrationHubStrategy

Nom du service	**Task**ressource de l'État	Date prise en charge	Préfixe d'exception
AWS Mobile	`arn:aws:states:::aws-sdk:mobile: [apiAction]`	30 septembre 2021	
AWS Network Firewall	`arn:aws:states:::aws-sdk:networkfirewall: [apiAction]`	30 septembre 2021	NetworkFirewall
AWS OpsWorks	`arn:aws:states:::aws-sdk:opsworks: [apiAction]`	30 septembre 2021	OpsWorks
AWS OpsWorks CM	`arn:aws:states:::aws-sdk:opsworkscm: [apiAction]`	30 septembre 2021	OpsWorksCm
AWS Organizations	`arn:aws:states:::aws-sdk:organizations: [apiAction]`	30 septembre 2021	Organizations
AWS Outposts	`arn:aws:states:::aws-sdk:outposts: [apiAction]`	30 septembre 2021	Outposts
AWS Panorama	`arn:aws:states:::aws-sdk:panorama: [apiAction]`	19 avril 2022	Panorama
Amazon Relational Database Service Performance Insights	`arn:aws:states:::aws-sdk:pi: [apiAction]`	30 septembre 2021	Pi

Nom du service	**Task**ressource de l'État	Date prise en charge	Préfixe d'exception
AWS Price List	`arn:aws:states:::aws-sdk:pricing: [apiAction]`	30 septembre 2021	Pricing
Amazon Relational Database Service	`arn:aws:states:::aws-sdk:rdsdata: [apiAction]` *** —	30 septembre 2021	RdsData
AWS Resilience Hub	`arn:aws:states:::aws-sdk:resilienceh ub: [apiAction]`	19 avril 2022	Resiliencehub
AWS Resource Access Manager	`arn:aws:states:::aws-sdk:ram: [apiAction]`	30 septembre 2021	Ram
AWS Resource Groups	`arn:aws:states:::aws-sdk:resourcegroups: [apiAction]`	30 septembre 2021	ResourceGroups
AWS Resource Groups Tagging API	`arn:aws:states:::aws-sdk:resourcegroupstaggingapi: [apiAction]`	30 septembre 2021	ResourceGroupsTaggingApi
AWS RoboMaker	`arn:aws:states:::aws-sdk:robomaker: [apiAction]`	30 septembre 2021	RoboMaker
AWS IAM Identity Center	`arn:aws:states:::aws-sdk:identitystore: [apiAction]`	30 septembre 2021	Identitystore

Nom du service	**Task**ressource de l'État	Date prise en charge	Préfixe d'exception
IAM Identity Center OIDC	`arn:aws:states:::aws-sdk:ssooidc:` *`[apiAction]`*	30 septembre 2021	SsoOidc
AWS Secrets Manager	`arn:aws:states:::aws-sdk:secretsmanager:` *`[apiAction]`*	30 septembre 2021	SecretsManager
AWS Security Token Service	`arn:aws:states:::aws-sdk:sts:` *`[apiAction]`* ***	30 septembre 2021	Sts
AWS Security Hub	`arn:aws:states:::aws-sdk:securityhub:` *`[apiAction]`*	30 septembre 2021	SecurityHub
AWS Server Migration Service	`arn:aws:states:::aws-sdk:sms:` *`[apiAction]`*	30 septembre 2021	Sms
AWS Service Catalog	`arn:aws:states:::aws-sdk:servicecatalog:` *`[apiAction]`*	30 septembre 2021	ServiceCatalog
AWS Service Catalog AppRegistry	`arn:aws:states:::aws-sdk:servicecatalogappregistry:` *`[apiAction]`*	30 septembre 2021	ServiceCatalogAppRegistry
AWS Shield	`arn:aws:states:::aws-sdk:shield:` *`[apiAction]`* ***	30 septembre 2021	Shield

Nom du service	**Task**ressource de l'État	Date prise en charge	Préfixe d'exception
AWS Signer	`arn:aws:states:::aws-sdk:signer: [apiAction]`	30 septembre 2021	Signer
IAM Identity Center	`arn:aws:states:::aws-sdk:sso: [apiAction]`	30 septembre 2021	Sso
IAM Identity Center Admin	`arn:aws:states:::aws-sdk:ssoadmin: [apiAction]`	30 septembre 2021	SsoAdmin
AWS Step Functions	`arn:aws:states:::aws-sdk:sfn: [apiAction]`	30 septembre 2021	Sfn
AWS Storage Gateway	`arn:aws:states:::aws-sdk:storagegateway: [apiAction]`	30 septembre 2021	StorageGateway
AWS Support	`arn:aws:states:::aws-sdk:support: [apiAction]`	30 septembre 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 septembre 2021	Transfer
AWS WAF	`arn:aws:states:::aws-sdk:waf: [apiAction]`	30 septembre 2021	Waf

Nom du service	**Task**ressource de l'État	Date prise en charge	Préfixe d'exception
AWS WAF Regional	`arn:aws:states:::aws-sdk:wafregional: [apiAction]`	30 septembre 2021	WafRegional
AWS WAFV2	`arn:aws:states:::aws-sdk:wafv2: [apiAction]`	30 septembre 2021	Wafv2
AWS Well-Architected Tool	`arn:aws:states:::aws-sdk:wellarchitected: [apiAction]`	30 septembre 2021	WellArchitected
AWS X-Ray	`arn:aws:states:::aws-sdk:xray: [apiAction]`	30 septembre 2021	XRay
AWS Marketplace Metering Service	`arn:aws:states:::aws-sdk:marketplacemetering: [apiAction]`	30 septembre 2021	MarketplaceMetering
AWS Serverless Application Repository	`arn:aws:states:::aws-sdk:serverlessapplicationrepository: [apiAction]`	30 septembre 2021	ServerlessApplicationRepository
AWS Identity and Access Management Access Analyzer	`arn:aws:states:::aws-sdk:accessanalyzer: [apiAction]`	30 septembre 2021	AccessAnalyzer
Alexa for Business	`arn:aws:states:::aws-sdk:alexaforbusiness: [apiAction]`	30 septembre 2021	AlexaForBusiness

Nom du service	**Task**ressource de l'État	Date prise en charge	Préfixe d'exception
Amazon API Gateway	`arn:aws:states:::aws-sdk:apigateway:` *`[apiAction]`*	30 septembre 2021	ApiGateway
Amazon API Gateway	`arn:aws:states:::aws-sdk:apigatewayv2:` *`[apiAction]`*	30 septembre 2021	ApiGatewayV2
Amazon AppIntegrations	`arn:aws:states:::aws-sdk:appintegrations:` *`[apiAction]`*	30 septembre 2021	AppIntegrations
Amazon AppStream 2.0	`arn:aws:states:::aws-sdk:appstream:` *`[apiAction]`*	30 septembre 2021	AppStream
Amazon AppFlow	`arn:aws:states:::aws-sdk:appflow:` *`[apiAction]`*	30 septembre 2021	Appflow
Amazon Athena	`arn:aws:states:::aws-sdk:athena:` *`[apiAction]`*	30 septembre 2021	Athena
Amazon Augmented AI	`arn:aws:states:::aws-sdk:sagemakera2iruntime:` *`[apiAction]`*	30 septembre 2021	SageMaker A2IRuntime
Amazon Braket	`arn:aws:states:::aws-sdk:braket:` *`[apiAction]`*	30 septembre 2021	Braket

Nom du service	**Task**ressource de l'État	Date prise en charge	Préfixe d'exception
Amazon Chime	`arn:aws:states:::aws-sdk:chime: [apiAction]`	30 septembre 2021	Chime
Amazon Chime Meetings	`arn:aws:states:::aws-sdk:chimesdkmeetings: [apiAction]`	19 avril 2022	ChimeSdkMeetings
Amazon Cloud Directory	`arn:aws:states:::aws-sdk:clouddirectory: [apiAction]`	30 septembre 2021	CloudDirectory
Amazon CloudFront	`arn:aws:states:::aws-sdk:cloudfront: [apiAction]`	30 septembre 2021	CloudFront
Amazon CloudSearch	`arn:aws:states:::aws-sdk:cloudsearch: [apiAction]`	30 septembre 2021	CloudSearch
Amazon CloudWatch	`arn:aws:states:::aws-sdk:cloudwatch: [apiAction]`	30 septembre 2021	CloudWatch
Amazon CloudWatch Application Insights	`arn:aws:states:::aws-sdk:applicationinsights: [apiAction]`	30 septembre 2021	ApplicationInsights
CloudWatch Evidently	`arn:aws:states:::aws-sdk:evidently: [apiAction]`	19 avril 2022	Evidently

Nom du service	**Task**ressource de l'État	Date prise en charge	Préfixe d'exception
Amazon CloudWatch Logs	`arn:aws:states:::aws-sdk:cloudwatchlogs: [apiAction]`	30 septembre 2021	CloudWatchLogs
Amazon CloudWatch RUM	`arn:aws:states:::aws-sdk:rum: [apiAction]`	19 avril 2022	Rum
Amazon CloudWatch Synthetics	`arn:aws:states:::aws-sdk:synthetics: [apiAction]`	30 septembre 2021	Synthetics
Amazon CodeGuru Profiler	`arn:aws:states:::aws-sdk:codeguruprofiler: [apiAction]`	30 septembre 2021	CodeGuruProfiler
Amazon CodeGuru Reviewer	`arn:aws:states:::aws-sdk:codegurureviewer: [apiAction]`	30 septembre 2021	CodeGuruReviewer
Amazon Cognito	`arn:aws:states:::aws-sdk:cognitoidentity: [apiAction]`	30 septembre 2021	CognitoIdentity
Amazon Cognito Identity Provider	`arn:aws:states:::aws-sdk:cognitoidentityprovider: [apiAction]`	30 septembre 2021	CognitoIdentityProvider
Amazon Cognito Sync	`arn:aws:states:::aws-sdk:cognitosync: [apiAction]`	30 septembre 2021	CognitoSync

Nom du service	**Task**ressource de l'État	Date prise en charge	Préfixe d'exception
Amazon Comprehend	`arn:aws:states:::aws-sdk:comprehend:` *`[apiAction]`*	30 septembre 2021	Comprehend
Amazon Comprehend Medical	`arn:aws:states:::aws-sdk:comprehendmedical:` *`[apiAction]`* ***	30 septembre 2021	ComprehendMedical
Amazon Connect Contact Lens	`arn:aws:states:::aws-sdk:connectcontactlens:` *`[apiAction]`*	30 septembre 2021	ConnectContactLens
Amazon Connect Participant Service	`arn:aws:states:::aws-sdk:connectparticipant:` *`[apiAction]`*	30 septembre 2021	ConnectParticipant
Amazon Connect	`arn:aws:states:::aws-sdk:connect:` *`[apiAction]`*	30 septembre 2021	Connect
Amazon Connect Voice ID	`arn:aws:states:::aws-sdk:voiceid:` *`[apiAction]`*	19 avril 2022	VoiceId
Amazon Connect Wisdom	`arn:aws:states:::aws-sdk:wisdom:` *`[apiAction]`*	19 avril 2022	Wisdom
Amazon Data Lifecycle Manager	`arn:aws:states:::aws-sdk:dlm:` *`[apiAction]`*	30 septembre 2021	Dlm
Amazon Detective	`arn:aws:states:::aws-sdk:detective:` *`[apiAction]`*	30 septembre 2021	Detective

Nom du service	**Task**ressource de l'État	Date prise en charge	Préfixe d'exception
Amazon DevOps Guru	`arn:aws:states:::aws-sdk:devopsguru:` *`[apiAction]`*	30 septembre 2021	DevOpsGuru
Amazon DocumentDB (with MongoDB compatibility)	`arn:aws:states:::aws-sdk:docdb:` *`[apiAction]`*	30 septembre 2021	DocDb
Amazon DynamoDB	`arn:aws:states:::aws-sdk:dynamodb:` *`[apiAction]`*	30 septembre 2021	DynamoDb
Amazon DynamoDB Streams	`arn:aws:states:::aws-sdk:dynamodbstreams:` *`[apiAction]`*	30 septembre 2021	DynamoDbStreams
Amazon EC2 Container Registry	`arn:aws:states:::aws-sdk:ecr:` *`[apiAction]`*	30 septembre 2021	Ecr
Amazon EC2 Container Service	`arn:aws:states:::aws-sdk:ecs:` *`[apiAction]`*	30 septembre 2021	Ecs
Amazon EC2 Systems Manager	`arn:aws:states:::aws-sdk:ssm:` *`[apiAction]`*	30 septembre 2021	Ssm
Amazon EMR	`arn:aws:states:::aws-sdk:emrcontainers:` *`[apiAction]`* ***	30 septembre 2021	EmrContainers

Nom du service	**Task**ressource de l'État	Date prise en charge	Préfixe d'exception
Amazon ElastiCache	`arn:aws:states:::aws-sdk:elasticache:` *`[apiAction]`*	30 septembre 2021	ElastiCache
Amazon Elastic Inference	`arn:aws:states:::aws-sdk:elasticinference:` *`[apiAction]`*	30 septembre 2021	ElasticInference
Amazon Elastic Block Store	`arn:aws:states:::aws-sdk:ebs:` *`[apiAction]`*	30 septembre 2021	Ebs
Amazon Elastic Compute Cloud	`arn:aws:states:::aws-sdk:ec2:` *`[apiAction]`*	30 septembre 2021	Ec2
Amazon Elastic Container Registry Public	`arn:aws:states:::aws-sdk:ecrpublic:` *`[apiAction]`*	30 septembre 2021	EcrPublic
Amazon Elastic File System	`arn:aws:states:::aws-sdk:efs:` *`[apiAction]`* ***	30 septembre 2021	Efs
Amazon Elastic Kubernetes Service	`arn:aws:states:::aws-sdk:eks:` *`[apiAction]`*	30 septembre 2021	Eks
Amazon EMR	`arn:aws:states:::aws-sdk:emr:` *`[apiAction]`*	30 septembre 2021	Emr
Amazon Elastic Transcoder	`arn:aws:states:::aws-sdk:elastictranscoder:` *`[apiAction]`* ***	30 septembre 2021	ElasticTranscoder

Nom du service	**Task**ressource de l'État	Date prise en charge	Préfixe d'exception
Amazon OpenSearch Service	`arn:aws:states:::aws-sdk:elasticsearch: [apiAction]`	30 septembre 2021	Elasticsearch
Amazon OpenSearch Service	`arn:aws:states:::aws-sdk:opensearch: [apiAction]`	19 avril 2022	OpenSearch
Amazon EventBridge	`arn:aws:states:::aws-sdk:eventbridge: [apiAction]`	30 septembre 2021	EventBridge
Amazon FSx	`arn:aws:states:::aws-sdk:fsx: [apiAction]`	30 septembre 2021	FSx
Amazon Forecast Query	`arn:aws:states:::aws-sdk:forecastquery: [apiAction]`	30 septembre 2021	Forecastquery
Amazon Forecast Service	`arn:aws:states:::aws-sdk:forecast: [apiAction]`	30 septembre 2021	Forecast
Amazon Fraud Detector	`arn:aws:states:::aws-sdk:frauddetector: [apiAction]`	30 septembre 2021	FraudDetector
Amazon GameLift	`arn:aws:states:::aws-sdk:gamelift: [apiAction]`	30 septembre 2021	Amazon GameLift
Amazon GameSparks	`arn:aws:states:::aws-sdk:gamesparks: [apiAction]`	27 juillet 2022	GameSparks

Nom du service	**Task**ressource de l'État	Date prise en charge	Préfixe d'exception
Amazon S3 Glacier	`arn:aws:states:::aws-sdk:glacier: [apiAction]`	30 septembre 2021	Glacier
Amazon GuardDuty	`arn:aws:states:::aws-sdk:guardduty: [apiAction]`	30 septembre 2021	GuardDuty
AWS HealthLake	`arn:aws:states:::aws-sdk:healthlake: [apiAction]`	30 septembre 2021	HealthLake
Amazon Honeycode	`arn:aws:states:::aws-sdk:honeycode: [apiAction]`	30 septembre 2021	Honeycode
Amazon Inspector	`arn:aws:states:::aws-sdk:inspector: [apiAction]`	30 septembre 2021	Inspector
Amazon Inspector V2	`arn:aws:states:::aws-sdk:inspector2: [apiAction]`	19 avril 2022	Inspector2
Amazon Interactive Video Service	`arn:aws:states:::aws-sdk:ivs: [apiAction]`	30 septembre 2021	Ivs
Amazon Kendra	`arn:aws:states:::aws-sdk:kendra: [apiAction]`	30 septembre 2021	Kendra
Amazon Kinesis	`arn:aws:states:::aws-sdk:kinesis: [apiAction]` ***	30 septembre 2021	Kinesis

Nom du service	**Task**ressource de l'État	Date prise en charge	Préfixe d'exception
Amazon Kinesis Analytics	`arn:aws:states:::aws-sdk:kinesisanalytics: [apiAction]`	30 septembre 2021	KinesisAnalytics
Amazon Kinesis Analytics V2	`arn:aws:states:::aws-sdk:kinesisanalyticsv2: [apiAction]`	30 septembre 2021	KinesisAnalyticsV2
Amazon Kinesis Firehose	`arn:aws:states:::aws-sdk:firehose: [apiAction]`	30 septembre 2021	Firehose
Amazon Kinesis Video Signaling Channels	`arn:aws:states:::aws-sdk:kinesisvideosignaling: [apiAction]`	30 septembre 2021	KinesisVideoSignaling
Amazon Kinesis Video Streams	`arn:aws:states:::aws-sdk:kinesisvideo: [apiAction]`	30 septembre 2021	KinesisVideo
Amazon Kinesis Video Streams Archived Media	`arn:aws:states:::aws-sdk:kinesisvideoarchivedmedia: [apiAction]`	30 septembre 2021	KinesisVideoArchivedMedia
Amazon Kinesis video stream	`arn:aws:states:::aws-sdk:kinesisvideomedia: [apiAction]`	30 septembre 2021	KinesisVideoMedia
Amazon Lex Model Building Service	`arn:aws:states:::aws-sdk:lexmodelbuilding: [apiAction]`	30 septembre 2021	LexModelBuilding

Nom du service	**Task**ressource de l'État	Date prise en charge	Préfixe d'exception
Amazon Lex Model Building Service V2	`arn:aws:states:::aws-sdk:lexmodelsv2:` *`[apiAction]`*	30 septembre 2021	LexModelsV2
Amazon Lex	`arn:aws:states:::aws-sdk:lexruntime:` *`[apiAction]`*	30 septembre 2021	LexRuntime
Amazon Lex Runtime V2	`arn:aws:states:::aws-sdk:lexruntimev2:` *`[apiAction]`* ***	30 septembre 2021	LexRuntimeV2
Amazon Lightsail	`arn:aws:states:::aws-sdk:lightsail:` *`[apiAction]`*	30 septembre 2021	Lightsail
Amazon Location Service	`arn:aws:states:::aws-sdk:location:` *`[apiAction]`*	30 septembre 2021	Location
Amazon Lookout for Equipment	`arn:aws::states:::aws-sdk:lookoutequipment:` *`[apiAction]`*	30 septembre 2021	LookoutEquipment
Amazon Lookout for Metrics	`arn:aws:states:::aws-sdk:lookoutmetrics:` *`[apiAction]`*	30 septembre 2021	LookoutMetrics
Amazon Lookout for Vision	`arn:aws:states:::aws-sdk:lookoutvision:` *`[apiAction]`*	30 septembre 2021	LookoutVision
Amazon MQ	`arn:aws:states:::aws-sdk:mq:` *`[apiAction]`*	30 septembre 2021	Mq

Nom du service	**Task**ressource de l'État	Date prise en charge	Préfixe d'exception
Amazon Macie	`arn:aws:states:::aws-sdk:macie: [apiAction]`	30 septembre 2021	
Amazon Macie 2	`arn:aws:states:::aws-sdk:macie2: [apiAction]`	30 septembre 2021	Macie2
Amazon Managed Blockchain	`arn:aws:states:::aws-sdk:managedblockchain: [apiAction]`	30 septembre 2021	ManagedBlockchain
Amazon Managed Grafana	`arn:aws:states:::aws-sdk:grafana: [apiAction]`	19 avril 2022	Grafana
Amazon Managed Service for Prometheus	`arn:aws:states:::aws-sdk:amp: [apiAction]`	30 septembre 2021	Amp
Amazon Managed Streaming for Apache Kafka	`arn:aws:states:::aws-sdk:kafka: [apiAction]`	30 septembre 2021	Kafka
Amazon MSK Connect	`arn:aws:states:::aws-sdk:kafkaconnect: [apiAction]`	19 avril 2022	KafkaConnect
Amazon Managed Workflows for Apache Airflow	`arn:aws:states:::aws-sdk:mwaa: [apiAction]`	30 septembre 2021	Mwaa
Amazon Mechanical Turk	`arn:aws:states:::aws-sdk:mturk: [apiAction]`	30 septembre 2021	MTurk

Nom du service	**Task**ressource de l'État	Date prise en charge	Préfixe d'exception
Amazon MemoryDB for Redis	`arn:aws:states:::aws-sdk:memorydb:` *`[apiAction]`*	19 avril 2022	MemoryDB
Amazon Nimble Studio	`arn:aws:states:::aws-sdk:nimble:` *`[apiAction]`*	30 septembre 2021	Nimble
Amazon Personalize	`arn:aws:states:::aws-sdk:personalize:` *`[apiAction]`*	30 septembre 2021	Personalize
Amazon Personalize Events	`arn:aws:states:::aws-sdk:personalizeevents:` *`[apiAction]`*	30 septembre 2021	PersonalizeEvents
Amazon Personalize Runtime	`arn:aws:states:::aws-sdk:personalizeruntime:` *`[apiAction]`*	30 septembre 2021	PersonalizeRuntime
Amazon Pinpoint	`arn:aws:states:::aws-sdk:pinpoint:` *`[apiAction]`*	30 septembre 2021	Pinpoint
Amazon Pinpoint Email Service	`arn:aws:states:::aws-sdk:pinpointemail:` *`[apiAction]`*	30 septembre 2021	PinpointEmail
Amazon Pinpoint SMS and Voice Service	`arn:aws:states:::aws-sdk:pinpointsmsvoice:` *`[apiAction]`*	30 septembre 2021	PinpointSmsVoice
Amazon Pinpoint SMS and Voice V2 Service	`arn:aws:states:::aws-sdk:pinpointsmsvoicev2:` *`[apiAction]`*	27 juillet 2022	PinpointSmsVoiceV2

Nom du service	**Task**ressource de l'État	Date prise en charge	Préfixe d'exception
Amazon Polly	`arn:aws:states:::aws-sdk:polly: [apiAction]`	30 septembre 2021	Polly
Amazon QLDB	`arn:aws:states:::aws-sdk:qldb: [apiAction]`	30 septembre 2021	Qldb
Amazon QLDB Session	`arn:aws:states:::aws-sdk:qldbsession: [apiAction]`	30 septembre 2021	QldbSession
Amazon QuickSight	`arn:aws:states:::aws-sdk:quicksight: [apiAction]`	30 septembre 2021	QuickSight
Amazon Redshift	`arn:aws:states:::aws-sdk:redshift: [apiAction]`	30 septembre 2021	Redshift
Amazon Redshift Data API	`arn:aws:states:::aws-sdk:redshiftdata: [apiAction]`	30 septembre 2021	RedshiftData
Amazon Rekognition	`arn:aws:states:::aws-sdk:rekognition: [apiAction]`	30 septembre 2021	Rekognition
Amazon Relational Database Service	`arn:aws:states:::aws-sdk:rds: [apiAction]`	30 septembre 2021	Rds
Amazon Route 53	`arn:aws:states:::aws-sdk:route53: [apiAction]`	30 septembre 2021	Route53

Nom du service	**Task**ressource de l'État	Date prise en charge	Préfixe d'exception
Amazon Route 53 Recovery Control Config	`arn:aws:states:::aws-sdk:route53recoverycontrolconfig:` *[apiAction]*	30 septembre 2021	Route53RecoveryControlConfig
Amazon Route 53 Domains	`arn:aws:states:::aws-sdk:route53domains:` *[apiAction]*	30 septembre 2021	Route53Domains
Amazon Route 53 Resolver	`arn:aws:states:::aws-sdk:route53resolver:` *[apiAction]*	30 septembre 2021	Route53Resolver
Amazon S3 on Outposts	`arn:aws:states:::aws-sdk:s3outposts:` *[apiAction]*	30 septembre 2021	S3Outposts
Amazon SageMaker Runtime Feature Store Runtime	`arn:aws:states:::aws-sdk:sagemakerfeaturestoreruntime:` *[apiAction]*	30 septembre 2021	SageMakerRuntimeFeatureStoreRuntime
Amazon SageMaker Runtime Runtime	`arn:aws:states:::aws-sdk:sagemakerruntime:` *[apiAction]*	30 septembre 2021	SageMakerRuntimeRuntime
Amazon SageMaker	`arn:aws:states:::aws-sdk:sagemaker:` *[apiAction]*	30 septembre 2021	SageMakerRuntime
Amazon SageMaker Edge Manager	`arn:aws:states:::aws-sdk:sagemakeredge:` *[apiAction]*	30 septembre 2021	SagemakerEdge

Nom du service	**Task**ressource de l'État	Date prise en charge	Préfixe d'exception
Amazon Simple Email Service	`arn:aws:states:::aws-sdk:ses:` *`[apiAction]`*	30 septembre 2021	Ses
Amazon Simple Email Service V2	`arn:aws:states:::aws-sdk:sesv2:` *`[apiAction]`*	30 septembre 2021	SesV2
Amazon Simple Notification Service	`arn:aws:states:::aws-sdk:sns:` *`[apiAction]`*	30 septembre 2021	Sns
Amazon Simple Queue Service	`arn:aws:states:::aws-sdk:sqs:` *`[apiAction]`*	30 septembre 2021	Sqs
Amazon Simple Storage Service	`arn:aws:states:::aws-sdk:s3:` *`[apiAction]`* ***	30 septembre 2021	S3
Amazon Simple Workflow Service	`arn:aws:states:::aws-sdk:swf:` *`[apiAction]`*	30 septembre 2021	Swf
Amazon Textract	`arn:aws:states:::aws-sdk:textract:` *`[apiAction]`*	30 septembre 2021	Textract
Amazon Transcribe	`arn:aws:states:::aws-sdk:transcribe:` *`[apiAction]`*	30 septembre 2021	Transcribe
Amazon Translate	`arn:aws:states:::aws-sdk:translate:` *`[apiAction]`*	30 septembre 2021	Translate

Nom du service	**Task**ressource de l'État	Date prise en charge	Préfixe d'exception
Amazon WorkDocs	`arn:aws:states:::aws-sdk:workdocs:` *`[apiAction]`*	30 septembre 2021	WorkDocs
Amazon WorkMail	`arn:aws:states:::aws-sdk:workmail:` *`[apiAction]`*	30 septembre 2021	WorkMail
Amazon WorkMail Message Flow	`arn:aws:states:::aws-sdk:workmailmessageflow:` *`[apiAction]`*	30 septembre 2021	WorkMailMessageFlow
Amazon WorkSpaces	`arn:aws:states:::aws-sdk:workspaces:` *`[apiAction]`*	30 septembre 2021	WorkSpaces
Amazon WorkSpaces Web	`arn:aws:states:::aws-sdk:workspacesweb:` *`[apiAction]`*	19 avril 2022	WorkSpacesWeb
Amplify	`arn:aws:states:::aws-sdk:amplifybackend:` *`[apiAction]`*	30 septembre 2021	AmplifyBackend
Amplify UI Builder	`arn:aws:states:::aws-sdk:amplifyuibuilder:` *`[apiAction]`*	19 avril 2022	AmplifyUiBuilder
Application Auto Scaling	`arn:aws:states:::aws-sdk:applicationautoscaling:` *`[apiAction]`*	30 septembre 2021	ApplicationAutoScaling

Nom du service	**Task**ressource de l'État	Date prise en charge	Préfixe d'exception
Amazon EC2 Auto Scaling	`arn:aws:states:::aws-sdk:autoscaling:` *[apiAction]*	30 septembre 2021	Auto Scaling
CodeArtifact	`arn:aws:states:::aws-sdk:codeartifact:` *[apiAction]*	30 septembre 2021	Codeartifact
DynamoDB Accelerator	`arn:aws:states:::aws-sdk:dax:` *[apiAction]*	30 septembre 2021	Dax
EC2 Image Builder	`arn:aws:states:::aws-sdk:imagebuilder:` *[apiAction]*	30 septembre 2021	Imagebuilder
AWS Elastic Disaster Recovery	`arn:aws:states:::aws-sdk:drs:` *[apiAction]*	19 avril 2022	Drs
Elastic Load Balancing	`arn:aws:states:::aws-sdk:elasticloadbalancing:` *[apiAction]*	30 septembre 2021	ElasticLoadBalancing
Elastic Load Balancing V2	`arn:aws:states:::aws-sdk:elasticloadbalancingv2:` *[apiAction]*	30 septembre 2021	ElasticLoadBalancingV2
MediaConnect	`arn:aws:states:::aws-sdk:mediaconnect:` *[apiAction]*	30 septembre 2021	MediaConnect

Nom du service	**Task**ressource de l'État	Date prise en charge	Préfixe d'exception
Amazon S3 Control	`arn:aws:states:::aws-sdk:s3control:` *[apiAction]* ***	30 septembre 2021	S3Control
Recycle Bin for Amazon EBS	`arn:aws:states:::aws-sdk:rbin:` *[apiAction]*	19 avril 2022	Rbin
Savings Plans	`arn:aws:states:::aws-sdk:savingsplans:` *[apiAction]*	30 septembre 2021	Savingsplans
Amazon EventBridge Schema Registry	`arn:aws:states:::aws-sdk:schemas:` *[apiAction]*	30 septembre 2021	Schemas
Service Quotas	`arn:aws:states:::aws-sdk:servicequotas:` *[apiAction]*	30 septembre 2021	ServiceQuotas
AWS Snowball	`arn:aws:states:::aws-sdk:snowball:` *[apiAction]*	30 septembre 2021	Snowball

Actions d'API non prises en charge pour les services pris en charge

Le tableau suivant répertorie les actions d'API non prises en charge pour les intégrations de services AWS SDK. La colonne de droite contient les actions d'API qui ne sont actuellement pas prises en charge pour le service répertorié dans la colonne de gauche.

Nom du service	Action d'API non prise en charge
AWS Application Discovery Service	• `DescribeExportConfigurations`

Nom du service	Action d'API non prise en charge
	• `ExportConfigurations`
Amazon Bedrock	• `InvokeModelWithResponseStream`
Agents pour 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 Principal Device Advisor	• `ListTestCases`
Amazon Kinesis	• `SubscribeToShard`

Nom du service	Action d'API non prise en charge
AWS Lambda	• `InvokeAsync` • `InvokeWithResponseStream`
Amazon Lex Runtime V2	• `StartConversation`
AWS Elemental MediaPackage	• `RotateChannelCredentials`
Amazon Relational Database Service	• `ExecuteSql`
Amazon Simple Storage Service	• `SelectObjectContent`
Contrôle Amazon S3	• `SelectObjectContent`
AWS Shield	• `DeleteSubscription`
AWS Security Token Service	• `AssumeRole` • `AssumeRoleWithSAML` • `AssumeRoleWithWebIdentity`

Intégrations de services AWS SDK obsolètes

Les intégrations de services AWS SDK suivantes sont désormais obsolètes :

- AWS Portable
- Amazon Macie
- AWS IoT RoboRunner

Intégrations optimisées pour Step Functions

Les rubriques suivantes incluent les API, les paramètres et la syntaxe de demande/réponse pris en charge dans l'Amazon States Language pour la coordination d'autres services. AWS Les rubriques fournissent également un exemple de code. Vous pouvez appeler les services d'intégration optimisés directement depuis l'Amazon States Language dans le `Resource` champ d'un `Task` État.

Vous pouvez utiliser trois modèles d'intégration de services :

- Demander une réponse (par défaut) : attendez la réponse HTTP, puis passez à l'état suivant
- Run a Job (`.sync`) : attendez que le travail soit terminé
- Wait for Callback (`.waitForTaskToken`) : suspend un flux de travail jusqu'à ce qu'un jeton de tâche soit renvoyé

Les flux de travail standard et les flux de travail express prennent en charge les mêmes intégrations, mais pas les mêmes modèles d'intégration.

- La prise en charge des modèles d'intégration optimisés est différente pour chaque intégration.
- Express Workflows ne prend pas en charge Run a Job (.sync) ou Wait for Callback (. waitForTaskJeton).
- Pour plus d'informations, consultez Flux de travail standard ou express.

Standard Workflows

Intégrations de services prises en charge

	Service	Réponse à la requête	Exécuter une tâche (`.sync`)	Attendre le rappel (`.waitForTaskToken`)
Intégrations optimisées	Amazon API Gateway	✓		✓
	Amazon Athena	✓	✓	
	AWS Batch	✓	✓	
	Amazon Bedrock	✓	✓	✓
	AWS CodeBuild	✓	✓	
	Amazon DynamoDB	✓		
	Amazon ECS/Fargate	✓	✓	✓
	Amazon EKS	✓	✓	✓
	Amazon EMR	✓	✓	

	Service	Réponse à la requête	Exécuter une tâche (`.sync`)	Attendre le rappel (`.waitForTaskToken`)
	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 Intégrations du SDK	Plus de deux cents	✓		✓

Express Workflows

Intégrations de services prises en charge

	Service	Réponse à la requête	Exécuter une tâche (`.sync`)	Attendre le rappel (`.waitForTaskToken`)
Intégrations optimisées	Amazon API Gateway	✓		
	Amazon Athena	✓		
	AWS Batch	✓		
	Amazon Bedrock	✓		
	AWS CodeBuild	✓		
	Amazon DynamoDB	✓		
	Amazon ECS/Fargate	✓		
	Amazon EKS	✓		
	Amazon EMR	✓		
	Amazon EMR on EKS	✓		
	Amazon EMR Serverless	✓		
	Amazon EventBridge	✓		
	AWS Glue	✓		
	AWS Glue DataBrew	✓		
	AWS Lambda	✓		
	AWS Elemental MediaConvert	✓		
	Amazon SageMaker	✓		

	Service	Réponse à la requête	Exécuter une tâche (`.sync`)	Attendre le rappel (`.waitForTaskToken`)
	Amazon SNS	✓		
	Amazon SQS	✓		
	AWS Step Functions	✓		
AWS Intégrations du SDK	Plus de deux cents	✓		

Call API Gateway avec Step Functions

Step Functions peut contrôler certains AWS services directement depuis Amazon States Language (ASL). Pour en savoir plus, consultez Utilisation avec d'autres services et Transmettre des paramètres à une API de service.

> En quoi l'intégration d'Optimized API Gateway est différente de l'intégration du AWS SDK API Gateway
>
> - `apigateway:invoke:`n'a pas d'équivalent dans l'intégration des services du AWS SDK. Au lieu de cela, le service Optimized API Gateway appelle directement votre point de terminaison API Gateway.

Vous utilisez Amazon API Gateway pour créer, publier, gérer et surveiller les API HTTP et REST. Pour intégrer API Gateway, vous définissez un `Task` état dans Step Functions qui appelle directement un point de terminaison HTTP ou REST API Gateway, sans écrire de code ni recourir à une autre infrastructure. Une définition d'`Task`état inclut toutes les informations nécessaires à l'appel d'API. Vous pouvez également sélectionner différentes méthodes d'autorisation.

> **Note**
>
> Step Functions permet d'appeler des points de terminaison HTTP via API Gateway, mais ne permet pas actuellement d'appeler des points de terminaison HTTP génériques.

Support des fonctionnalités d'API Gateway

L'intégration de Step Functions API Gateway prend en charge certaines fonctionnalités d'API Gateway, mais pas toutes. Pour une liste plus détaillée des fonctionnalités prises en charge, consultez ce qui suit.

- Pris en charge à la fois par l'API REST Step Functions API Gateway et par les intégrations de l'API HTTP API Gateway :
  - Autorisateurs : IAM (utilisant Signature Version 4), No Auth, Autorisateurs Lambda (basés sur des paramètres de demande et basés sur des jetons avec en-tête personnalisé)
  - Types d'API : régionaux
  - Gestion des API : noms de domaine de l'API API Gateway, stage de l'API, chemin, paramètres de requête, corps de la requête
- Pris en charge par l'intégration de l'API HTTP Step Functions API Gateway. L'intégration de l'API REST Step Functions API Gateway qui fournit l'option d'API optimisées pour Edge n'est pas prise en charge.
- Non pris en charge par l'intégration de l'API Gateway Step Functions :
  - Autorisateurs : Amazon Cognito, Native Open ID Connect/OAuth 2.0, en-tête d'autorisation pour les autorisateurs Lambda basés sur des jetons
  - Types d'API : privés
  - Gestion des API : noms de domaine personnalisés

Pour plus d'informations sur API Gateway et ses API HTTP et REST, consultez ce qui suit.

- La page des concepts d'Amazon API Gateway.
- Choisir entre les API HTTP et les API REST dans le guide du développeur d'API Gateway.

Format des demandes

Lorsque vous créez votre définition `Task` d'état, Step Functions valide les paramètres, crée l'URL nécessaire pour effectuer l'appel, puis appelle l'API. La réponse inclut le code d'état HTTP, les en-têtes et le corps de la réponse. Le format de demande comporte des paramètres obligatoires et facultatifs.

Paramètres de demande requis

- `ApiEndpoint`
  - Type : `String`
  - Le nom d'hôte d'une URL d'API Gateway. Le format est *`<API ID>`*`.execute-api.`*`<region>`*`.amazonaws.com`.

    L'ID d'API ne peut contenir qu'une combinaison des caractères alphanumériques suivants : `0123456789abcdefghijklmnopqrstuvwxyz`
- `Method`
  - Type : `Enum`
  - La méthode HTTP, qui doit être l'une des suivantes :
    - `GET`
    - `POST`
    - `PUT`
    - `DELETE`
    - `PATCH`
    - `HEAD`
    - `OPTIONS`

Paramètres de demande facultatifs

- `Headers`
  - Type : `JSON`
  - Les en-têtes HTTP permettent d'obtenir une liste de valeurs associées à la même clé.
- `Stage`
  - Type : `String`

  - Nom de l'étape sur laquelle l'API est déployée dans API Gateway. C'est facultatif pour toute API HTTP qui utilise le `$default` stage.
- `Path`
  - Type : `String`
  - Paramètres de chemin ajoutés après le point de terminaison de l'API.
- `QueryParameters`
  - Type : `JSON`
  - Les chaînes de requête n'autorisent qu'une liste de valeurs associées à la même clé.
- `RequestBody`
  - Type : `JSON` ou `String`.
  - Le corps de la requête HTTP. Son type peut être un `JSON` objet ou`String`. `RequestBody`n'est pris en charge que pour `PATCH` `POST` les méthodes `PUT` HTTP, et.
- `AllowNullValues`
  - Type : `BOOLEAN` — valeur par défaut : `false`
  - Avec le paramètre par défaut, aucune valeur nulle dans l'état d'entrée de la demande ne sera envoyée à votre API. Dans l'exemple suivant, le `category` champ ne sera pas inclus dans la demande, sauf s'`AllowNullValues`il est défini comme tel `true` dans la définition de votre machine à états.

```
{
    "NewPet": {
        "type": "turtle",
        "price": 123,
        "category": null
    }
}
```

> **Note**
>
> Par défaut, les champs contenant des valeurs nulles dans l'état d'entrée de la demande ne seront pas envoyés à votre API. Vous pouvez forcer l'envoi de valeurs nulles à votre API en le définissant `true` dans `AllowNullValues` la définition de votre machine d'état.

- `AuthType`

- Type : `JSON`
- La méthode d'authentification. La méthode par défaut est`NO_AUTH`. Les valeurs autorisées sont :
  - `NO_AUTH`
  - `IAM_ROLE`
  - `RESOURCE_POLICY`

  Voir Authentification et autorisation pour plus d'informations.

> **Note**
>
> Pour des raisons de sécurité, les clés d'en-tête HTTP suivantes ne sont actuellement pas autorisées :
>
> - Tout ce qui est préfixé par`X-Forwarded`, `X-Amz` ou`X-Amzn`.
> - `Authorization`
> - `Connection`
> - `Content-md5`
> - `Expect`
> - `Host`
> - `Max-Forwards`
> - `Proxy-Authenticate`
> - `Server`
> - `TE`
> - `Transfer-Encoding`
> - `Trailer`
> - `Upgrade`
> - `Via`
> - `Www-Authenticate`

L'exemple de code suivant montre comment invoquer API Gateway à l'aide de Step Functions.


```
{
```

```
    "Type": "Task",
    "Resource":"arn:aws:states:::apigateway:invoke",
    "Parameters": {
        "ApiEndpoint": "example.execute-api.us-east-1.amazonaws.com",
        "Method": "GET",
        "Headers": {
            "key": ["value1", "value2"]
        },
        "Stage": "prod",
        "Path": "bills",
        "QueryParameters": {
            "billId": ["123456"]
        },
        "RequestBody": {},
        "AuthType": "NO_AUTH"
    }
}
```

Authentification et autorisation

Vous pouvez utiliser les méthodes d'authentification suivantes :

- Aucune autorisation : appelez l'API directement sans méthode d'autorisation.
- Rôle IAM : avec cette méthode, Step Functions joue le rôle de machine à états, signe la demande avec Signature Version 4 (SigV4), puis appelle l'API.
- Politique en matière de ressources : Step Functions authentifie la demande, puis appelle l'API. Vous devez associer une politique de ressources à l'API qui spécifie les éléments suivants :
  1. La machine d'état qui invoquera API Gateway.

     > **Important**
     >
     > Vous devez spécifier votre machine d'état pour en limiter l'accès. Si ce n'est pas le cas, toute machine d'état qui authentifie sa demande API Gateway avec une authentification par politique de ressources auprès de votre API sera autorisée à y accéder.

  2. That Step Functions est le service qui appelle API Gateway :`"Service": "states.amazonaws.com"`.
  3. La ressource à laquelle vous souhaitez accéder, notamment :
     - La *région*.

- L'*identifiant du compte* dans la région spécifiée.
- L'identifiant de *l'API*.
- Le nom de *scène.*
- Le *HTTP-VERB* (méthode).
- Le *resource-path-specifier*.

Pour un exemple de politique de ressources, consultez les politiques IAM pour Step Functions et API Gateway.

Pour plus d'informations sur le format des ressources, consultez la section Format des ressources des autorisations d'exécution de l'API dans API Gateway dans le Guide du développeur d'API Gateway.

> **Note**
>
> Les politiques de ressources ne sont prises en charge que pour l'API REST.

Modèles d'intégration des services

L'intégration d'API Gateway prend en charge deux modèles d'intégration de services :

- Réponse à la requête, qui est le modèle d'intégration par défaut. Il permet à Step Functions de passer à l'étape suivante immédiatement après avoir reçu une réponse HTTP.
- Attendre un rappel avec le jeton de tâche(`.waitForTaskToken`), qui attend qu'un jeton de tâche soit renvoyé avec une charge utile. Pour utiliser le `.waitForTaskToken` modèle, ajoutez-le. waitForTaskJeton à la fin du champ Ressource de votre définition de tâche, comme illustré dans l'exemple suivant :

```
{
    "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"
    }
}
```

Format de sortie

Les paramètres de sortie suivants sont fournis :

Nom	Type	Description
`ResponseBody`	`JSON` ou `String`	Le corps de réponse de l'appel d'API.
`Headers`	`JSON`	Les en-têtes de réponse.
`StatusCode`	`Integer`	Code de statut HTTP de la réponse.
`StatusText`	`String`	Le texte d'état de la réponse.

Exemple de réponse :

```
{
    "ResponseBody": {
        "myBills": []
    },
    "Headers": {
        "key": ["value1", "value2"]
    },
    "StatusCode": 200,
    "StatusText": "OK"
}
```

Gestion des erreurs

Lorsqu'une erreur se produit, un `error` et `cause` est renvoyé comme suit :

- Si le code d'état HTTP est disponible, l'erreur sera renvoyée au format`ApiGateway.<HTTP Status Code>`.
- Si le code d'état HTTP n'est pas disponible, l'erreur sera renvoyée au format`ApiGateway.<Exception>`.

Dans les deux cas, le `cause` est renvoyé sous forme de chaîne.

L'exemple suivant montre une réponse dans laquelle une erreur s'est produite :

```
{
    "error": "ApiGateway.403",
    "cause": "{\"message\":\"Missing Authentication Token\"}"
}
```

> **Note**
>
> Le code d'état 2XX indique le succès et aucune erreur ne sera renvoyée. Tous les autres codes d'état ou exceptions déclenchées provoqueront une erreur.

Pour plus d'informations, consultez :

- Les concepts d'Amazon API Gateway présentés dans le guide du développeur d'API Gateway.
- Politiques IAM pour Amazon API Gateway
- Un exemple de projet qui montre comment Passez un appel à API Gateway

Les concepts d'Amazon API Gateway présentés dans le guide du développeur d'API Gateway.

Appelez Athéna avec Step Functions

Step Functions peut contrôler certains AWS services directement depuis Amazon States Language (ASL). Pour en savoir plus, consultez Utilisation avec d'autres services et Transmettre des paramètres à une API de service.

En quoi l'intégration optimisée d'Athena est différente de l'intégration du SDK Athena AWS

- Le modèle Exécuter une tâche (.sync) d'intégration est pris en charge.
- Il n'y a aucune optimisation pour le modèle Réponse à la requête d'intégration.
- Le modèle Attendre un rappel avec le jeton de tâche d'intégration n'est pas pris en charge.

L'intégration du AWS Step Functions service avec Amazon Athena vous permet d'utiliser Step Functions pour démarrer et arrêter l'exécution des requêtes et obtenir les résultats des requêtes. À l'aide de Step Functions, vous pouvez exécuter des requêtes de données ad hoc ou planifiées et récupérer des résultats ciblant vos lacs de données S3. Athena fonctionne sans serveur ; vous n'avez donc pas d'infrastructure à configurer ni à gérer, et vous ne payez que pour les requêtes que vous exécutez.

Pour intégrer AWS Step Functions Amazon Athena, vous utilisez les API d'intégration des services Athena fournies.

Les API d'intégration de services sont les mêmes que les API Athena correspondantes. Toutes les API ne prennent pas en charge tous les modèles d'intégration, comme le montre le tableau suivant :

API	Réponse à la requête	Exécuter une tâche (.sync)
`StartQueryExecution`	✓	✓
`StopQueryExecution`	✓	
`GetQueryExecution`	✓	
`GetQueryResults`	✓	

API Amazon Athena prises en charge :

Note

Il existe un quota pour la taille maximale des données d'entrée ou de résultat pour une tâche dans Step Functions. Cela vous limite à 256 Ko de données sous forme de chaîne codée en

UTF-8 lorsque vous envoyez ou recevez des données d'un autre service. veuillez consulter Quotas liés aux exécutions par les machines de l'État.

- StartQueryExecution
  - Syntaxe de demande
  - Paramètres pris en charge :
    - ClientRequestToken
    - ExecutionParameters
    - QueryExecutionContext
    - QueryString
    - ResultConfiguration
    - WorkGroup
  - Response syntax
- StopQueryExecution
  - Syntaxe de demande
  - Paramètres pris en charge :
    - QueryExecutionId
- GetQueryExecution
  - Syntaxe de demande
  - Paramètres pris en charge :
    - QueryExecutionId
  - Response syntax
- GetQueryResults
  - Syntaxe de demande
  - Paramètres pris en charge :
    - MaxResults
    - NextToken
    - QueryExecutionId
  - Response syntax

Ce qui suit inclut un état de tâche qui lance une requête Athena.

```
"Start an Athena query": {
  "Type": "Task",
  "Resource": "arn:aws:states:::athena:startQueryExecution.sync",
  "Parameters": {
    "QueryString": "SELECT * FROM \"myDatabase\".\"myTable\" limit 1",
    "WorkGroup": "primary",
    "ResultConfiguration": {
       "OutputLocation": "s3://athenaQueryResult"
    }
  },
  "Next": "Get results of the query"
}
```

Pour plus d'informations sur la configuration IAM des autorisations lors de l'utilisation Step Functions avec d'autres AWS services, consultezPolitiques IAM pour les services intégrés.

Gérez AWS Batch avec Step Functions

Step Functions peut contrôler certains AWS services directement depuis Amazon States Language (ASL). Pour en savoir plus, consultez Utilisation avec d'autres services et Transmettre des paramètres à une API de service.

> En quoi l' AWS Batch intégration optimisée est différente de l'intégration du AWS BatchAWS SDK
>
> - Le modèle Exécuter une tâche (.sync) d'intégration est disponible.
>
> Notez qu'il n'y a aucune optimisation pour les modèles d'Attendre un rappel avec le jeton de tâcheintégration Réponse à la requête ou.

AWS Batch API prises en charge :

- `SubmitJob`
  - Syntaxe de demande
  - Paramètres pris en charge :

- ArrayProperties
- ContainerOverrides
- DependsOn
- JobDefinition
- JobName
- JobQueue
- Parameters
- RetryStrategy
- Timeout
- Tags

- Syntaxe de réponse

> **Les paramètres in Step Functions sont exprimés en PascalCase**
>
> Même si l'API de service native se trouve dans CamelCase, par exemple l'`startSyncExecution`action d'API, vous spécifiez des paramètres PascalCase dans, tels que :. `StateMachineArn`

Ce qui suit inclut un `Task` État qui soumet une AWS Batch tâche et attend qu'elle soit terminée.

```
{
  "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
    }
  }
}
```

Pour plus d'informations sur la configuration IAM des autorisations lors de l'utilisation Step Functions avec d'autres AWS services, consultezPolitiques IAM pour les services intégrés.

Appelez Amazon Bedrock avec Step Functions

Step Functions peut contrôler certains AWS services directement depuis Amazon States Language (ASL). Pour en savoir plus, consultez Utilisation avec d'autres services et Transmettre des paramètres à une API de service.

Rubriques

- Amazon BedrockAPI d'intégration de services
- Définition de l'état des tâches pour Amazon Bedrock l'intégration

Amazon BedrockAPI d'intégration de services

Pour AWS Step Functions l'intégrerAmazon Bedrock, vous pouvez utiliser les API suivantes. Ces API sont similaires aux Amazon Bedrock API correspondantes, avec quelques différences dans les champs de demande transmis.

Les différences entre chaque API d'intégration de service et son API Amazon Bedrock correspondante sont illustrées dans le tableau suivant :

Amazon BedrockAPI d'intégration de services et Amazon Bedrock API correspondantes

Amazon BedrockAPI d'intégration de services	Amazon BedrockAPI correspondante	Différences
InvokeModel Invoque le Amazon Bedrock modèle spécifié pour exécuter	InvokeModel	Le corps de la demande d'API d'intégration de Amazon Bedrock services inclut les

Amazon BedrockAPI d'intégration de services	Amazon BedrockAPI correspondante	Différences
l'inférence à l'aide de l'entrée que vous fournissez dans le corps de la demande. Vous l'utilisez `InvokeModel` pour exécuter l'inférence pour les modèles de texte, les modèles d'image et les modèles d'intégration.		paramètres supplémentaires suivants. • `Body`— Spécifie les données d'entrée au format spécifié dans l'en-tête de demande de type de contenu. `Body`contient des paramètres spécifiques au modèle cible. Si vous utilisez l'`InvokeModel` API, vous devez spécifier le `Body` paramètre. Step Functionsne valide pas la saisie que vous fournissez`Body`. Lorsque vous spécifiez `Body` à l'aide de l'intégration Amazon Bedrock optimisée, vous pouvez spécifier une charge utile allant jusqu'à 256 Ko. Si votre charge utile dépasse 256 Ko, nous vous recommandons d'utiliser `Input`. • `Input`— Spécifie la source à partir de laquelle récupérer les données d'entrée. Ce champ facultatif est spécifique à l'intégration Amazon Bedrock optimisée avecStep Functions. Dans

Amazon BedrockAPI d'intégration de services	Amazon BedrockAPI correspondante	Différences
		ce champ, vous pouvez spécifier un`S3Uri`. Vous pouvez le spécifier soit `Body` dans les paramètre `sInput`, soit dans les deux. Lorsque vous spécifiez `Input` sans spécifier `ContentType` , le type de contenu de la source de données d'entrée devient la valeur de`ContentType` . • `Output`— Spécifie la destination où la réponse de l'API est écrite. Ce champ facultatif est spécifique à l'intégration Amazon Bedrock optimisée avecStep Functions. Dans ce champ, vous pouvez spécifier un`S3Uri`. Si vous spécifiez ce champ, le corps de la réponse de l'API est remplacé par une référence à l'Amazon S3emplacement de la sortie d'origine. L'exemple suivant montre la syntaxe de l' InvokeModel API pour Amazon Bedrock l'intégration.

Amazon BedrockAPI d'intégration de services	Amazon BedrockAPI correspondante	Différences

		<pre>{
		    "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
		    }
		}</pre>

CreateModelCustomizationJob Crée une tâche de réglage fin pour personnaliser un modèle de base.	CreateModelCustomizationJob	Aucun
CreateModelCustomizationJob.sync Crée une tâche de réglage fin pour personnaliser un modèle de base.	CreateModelCustomizationJob	Aucun

Pour plus d'informations sur la configuration IAM des autorisations lors de l'utilisation Step Functions avec d'autres AWS services, consultezPolitiques IAM pour les services intégrés.

Définition de l'état des tâches pour Amazon Bedrock l'intégration

La définition d'état de tâche suivante montre comment vous pouvez intégrer vos machines Amazon Bedrock à états. Cet exemple montre un état de tâche qui extrait le résultat complet de l'invocation du modèle spécifié par le chemin,`result_one`. Ceci est basé sur les paramètres d'inférence pour les modèles de base. Cet exemple utilise le modèle de langage large (LLM) de la commande Cohere.

```
{
  "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**
>
> Pour déployer un exemple de machine à états qui s'intègre Amazon Bedrock à votre Compte AWS, consultezRéalisez un enchaînement d'instructions basé sur l'IA avec Amazon Bedrock.

Appelez AWS CodeBuild avec Step Functions

Step Functions peut contrôler certains AWS services directement depuis Amazon States Language (ASL). Pour en savoir plus, consultez Utilisation avec d'autres services et Transmettre des paramètres à une API de service.

En quoi l' CodeBuild intégration optimisée est différente de l'intégration du CodeBuild AWS SDK

- Le modèle Exécuter une tâche (.sync) d'intégration est pris en charge.
- Une fois que vous avez appelé `StopBuild` ou`StopBuildBatch`, le build ou le batch de build n'est pas immédiatement supprimé tant que certains travaux internes ne sont pas terminés CodeBuild pour finaliser l'état de la ou des versions. Si vous essayez d'utiliser `BatchDeleteBuilds` ou `DeleteBuildBatch` pendant cette période, il est possible que le build ou le batch de build ne soit pas supprimé. Les intégrations de services optimisées pour `BatchDeleteBuilds` et `DeleteBuildBatch` incluent une nouvelle tentative interne afin de simplifier le cas d'utilisation de la suppression immédiatement après l'arrêt.

L'intégration du AWS Step Functions service vous AWS CodeBuild permet d'utiliser Step Functions pour déclencher, arrêter et gérer des builds, ainsi que pour partager des rapports de build. À l'aide de Step Functions, vous pouvez concevoir et exécuter des pipelines d'intégration continue pour valider les modifications apportées aux logiciels pour les applications.

Toutes les API ne prennent pas en charge tous les modèles d'intégration, comme le montre le tableau suivant :

API	Réponse à la requête	Exécuter une tâche (.sync)
StartBuild	✓	✓
StopBuild	✓	
BatchDeleteBuilds	✓	
BatchGetReports	✓	
StartBuildBatch	✓	✓
StopBuildBatch	✓	
RetryBuildBatch	✓	✓
DeleteBuildBatch	✓	

> **Les paramètres in Step Functions sont exprimés en PascalCase**
>
> Même si l'API de service native se trouve dans CamelCase, par exemple l'`startSyncExecution`action d'API, vous spécifiez des paramètres PascalCase dans, tels que :. `StateMachineArn`

CodeBuild API et syntaxe prises en charge :

- `StartBuild`
  - Syntaxe de demande
  - Paramètres pris en charge :
    - `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
    - Syntaxe de réponse
- StopBuild
    - Syntaxe de demande
    - Paramètres pris en charge :
        - Id
    - Syntaxe de réponse
- BatchDeleteBuilds
    - Syntaxe de demande
    - Paramètres pris en charge :
        - Ids
    - Syntaxe de réponse
- BatchGetReports
    - Syntaxe de demande
    - Paramètres pris en charge :
        - ReportArns
    - Syntaxe de réponse
- StartBuildBatch
    - Syntaxe de demande
    - Paramètres pris en charge :
        - 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

- Syntaxe de réponse

- StopBuildBatch
  - Syntaxe de demande
  - Paramètres pris en charge :
    - Id
  - Syntaxe de réponse
- RetryBuildBatch
  - Syntaxe de demande
  - Paramètres pris en charge :
    - Id
    - IdempotencyToken
    - RetryType
  - Syntaxe de réponse
- DeleteBuildBatch
  - Syntaxe de demande
  - Paramètres pris en charge :
    - Id
  - Syntaxe de réponse

> **Note**
>
> Vous pouvez utiliser l'opérateur de descente récursive JSONPath (`..`) pour `BatchDeleteBuilds` Un tableau est renvoyé, ce qui vous permet de transformer le champ `Arn` de `StartBuild` en un paramètre `Ids` pluriel, comme illustré dans l'exemple suivant.

```
"BatchDeleteBuilds": {
    "Type": "Task",
    "Resource": "arn:aws:states:::codebuild:batchDeleteBuilds",
    "Parameters": {
        "Ids.$": "$.Build..Arn"
    },
    "Next": "MyNextState"
},
```

Pour plus d'informations sur la configuration IAM des autorisations lors de l'utilisation Step Functions avec d'autres AWS services, consultezPolitiques IAM pour les services intégrés.

Appelez les API DynamoDB avec Step Functions

Step Functions peut contrôler certains AWS services directement depuis Amazon States Language (ASL). Pour en savoir plus, consultez Utilisation avec d'autres services et Transmettre des paramètres à une API de service.

> **Note**
>
> Il existe un quota pour la taille maximale des données d'entrée ou de résultat pour une tâche dans Step Functions. Cela vous limite à 256 Ko de données sous forme de chaîne codée en UTF-8 lorsque vous envoyez ou recevez des données d'un autre service. veuillez consulter Quotas liés aux exécutions par les machines de l'État.

> En quoi l'intégration DynamoDB optimisée est différente de l'intégration DynamoDB SDK AWS
>
> - Il n'y a aucune optimisation pour le modèle Réponse à la requête d'intégration.
> - Le modèle Attendre un rappel avec le jeton de tâche d'intégration n'est pas pris en charge.
> - Seules `GetItem`les actions `PutItem`, `UpdateItem`, et `DeleteItem`API sont disponibles grâce à une intégration optimisée. D'autres actions d'API, telles que celles `CreateTable`disponibles à l'aide de l'intégration du SDK AWS DynamoDB.

API et syntaxe Amazon DynamoDB prises en charge :

- `GetItem`
  - Syntaxe de demande
  - Paramètres pris en charge :
    - `Key`
    - `TableName`
    - `AttributesToGet`
    - `ConsistentRead`

    - ExpressionAttributeNames
    - ProjectionExpression
    - ReturnConsumedCapacity
  - Syntaxe de réponse
- PutItem
  - Syntaxe de demande
  - Paramètres pris en charge :
    - Item
    - TableName
    - ConditionalOperator
    - ConditionExpression
    - Expected
    - ExpressionAttributeNames
    - ExpressionAttributeValues
    - ReturnConsumedCapacity
    - ReturnItemCollectionMetrics
    - ReturnValues
  - Syntaxe de réponse
- DeleteItem
  - Syntaxe de demande
  - Paramètres pris en charge :
    - Key
    - TableName
    - ConditionalOperator
    - ConditionExpression
    - Expected
    - ExpressionAttributeNames
    - ExpressionAttributeValues
    - ReturnConsumedCapacity
    - ReturnItemCollectionMetrics

    - ReturnValues
  - Syntaxe de réponse
- UpdateItem
  - Syntaxe de demande
  - Paramètres pris en charge :
    - Key
    - TableName
    - AttributeUpdates
    - ConditionalOperator
    - ConditionExpression
    - Expected
    - ExpressionAttributeNames
    - ExpressionAttributeValues
    - ReturnConsumedCapacity
    - ReturnItemCollectionMetrics
    - ReturnValues
    - UpdateExpression
  - Syntaxe de réponse

> **Les paramètres in Step Functions sont exprimés en PascalCase**
>
> Même si l'API de service native se trouve dans CamelCase, par exemple l'`startSyncExecution`action d'API, vous spécifiez des paramètres PascalCase dans, tels que :. `StateMachineArn`

L'`Task`état suivant permet de récupérer un message depuis DynamoDB.

```
    "Read Next Message from DynamoDB": {
      "Type": "Task",
      "Resource": "arn:aws:states:::dynamodb:getItem",
      "Parameters": {
        "TableName": "TransferDataRecords-DDBTable-3I41R5L5EAGT",
        "Key": {
```

```
          "MessageId": {"S.$": "$.List[0]"}
        }
      },
      "ResultPath": "$.DynamoDB",
      "Next": "Send Message to SQS"
    },
```

Pour voir cet état dans un exemple pratique, consultez l'exemple de projet Transférer des enregistrements de données (Lambda,DynamoDB,Amazon SQS).

Pour plus d'informations sur la configuration IAM des autorisations lors de l'utilisation Step Functions avec d'autres AWS services, consultezPolitiques IAM pour les services intégrés.

Gérez les tâches Amazon ECS ou Fargate avec Step Functions

Step Functions peut contrôler certains AWS services directement depuis Amazon States Language (ASL). Pour en savoir plus, consultez Utilisation avec d'autres services et Transmettre des paramètres à une API de service.

> En quoi l'intégration optimisée Amazon ECS/Fargate est différente de l'intégration Amazon ECS ou Fargate SDK AWS
>
> - Le modèle Exécuter une tâche (.sync) d'intégration est pris en charge.
> - `ecs:runTask`peut renvoyer une réponse HTTP 200, mais avoir un `Failures` champ non vide comme suit :
>   - Demande de réponse : renvoie la réponse et n'échoue pas à la tâche. Cela revient à ne pas optimiser.
>   - Exécuter un Job ou un jeton de tâche : si un `Failures` champ non vide est détecté, la tâche échoue avec une `AmazonECS.Unknown` erreur.

API et syntaxe Amazon ECS/Fargate prises en charge :

> **Les paramètres in Step Functions sont exprimés en PascalCase**
>
> Même si l'API de service native se trouve dans CamelCase, par exemple l'`startSyncExecution`action d'API, vous spécifiez des paramètres PascalCase dans, tels que :. `StateMachineArn`

- `RunTask` démarre une nouvelle tâche à l'aide de la définition de tâche spécifiée.
  - Syntaxe de demande
  - Paramètres pris en charge :
    - `Cluster`
    - `Group`
    - `LaunchType`
    - `NetworkConfiguration`
    - `Overrides`
    - `PlacementConstraints`
    - `PlacementStrategy`
    - `PlatformVersion`
    - `PropagateTags`
    - `TaskDefinition`
    - `EnableExecuteCommand`
  - Syntaxe de réponse

Transmission de données à une tâche Amazon ECS

Step Functions peut contrôler certains AWS services directement depuis Amazon States Language (ASL). Pour en savoir plus, consultez Utilisation avec d'autres services et Transmettre des paramètres à une API de service.

Vous pouvez l'utiliser `overrides` pour remplacer la commande par défaut d'un conteneur et transmettre des données à vos tâches Amazon ECS. veuillez consulter `ContainerOverride`. Dans l'exemple, nous avons utilisé JsonPath pour transmettre des valeurs `Task` de l'entrée à l'`Task`état.

Ce qui suit inclut un `Task` état qui exécute une tâche Amazon ECS et attend qu'elle soit terminée.

```
{
 "StartAt": "Run an ECS Task and wait for it to complete",
 "States": {
   "Run an ECS Task and wait for it to complete": {
     "Type": "Task",
     "Resource": "arn:aws:states:::ecs:runTask.sync",
     "Parameters": {
                "Cluster": "cluster-arn",
                "TaskDefinition": "job-id",
                "Overrides": {
                    "ContainerOverrides": [
                        {
                            "Name": "container-name",
                            "Command.$": "$.commands"
                        }
                    ]
                }
            },
     "End": true
    }
  }
}
```

La ligne `"Command.$": "$.commands"` dans `ContainerOverrides` passe les commandes de l'entrée d'état au conteneur.

Pour l'exemple précédent, chacune des commandes sera passée comme un remplacement de conteneur si l'entrée de l'exécution est la suivante :

```
{
  "commands": [
    "test command 1",
    "test command 2",
    "test command 3"
  ]
}
```

Ce qui suit inclut un `Task` état qui exécute une tâche Amazon ECS, puis attend que le jeton de tâche soit renvoyé. veuillez consulter Attendre un rappel avec le jeton de tâche.

```
{
   "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
      }
   }
}
```

Pour plus d'informations sur la configuration IAM des autorisations lors de l'utilisation Step Functions avec d'autres AWS services, consultezPolitiques IAM pour les services intégrés.

Appelez Amazon EKS avec Step Functions

Step Functions peut contrôler certains AWS services directement depuis Amazon States Language (ASL). Pour en savoir plus, consultez Utilisation avec d'autres services et Transmettre des paramètres à une API de service.

> En quoi l'intégration optimisée d'Amazon EKS est différente de l'intégration du AWS SDK Amazon EKS
>
> - Le modèle Exécuter une tâche (.sync) d'intégration est pris en charge.
> - Il n'y a aucune optimisation pour le modèle Réponse à la requête d'intégration.

- Le modèle Attendre un rappel avec le jeton de tâche d'intégration n'est pas pris en charge.

Pour plus d'informations sur la configuration IAM des autorisations lors de l'utilisation Step Functions avec d'autres AWS services, consultezPolitiques IAM pour les services intégrés.

Step Functions fournit deux types d'API d'intégration de services pour l'intégration à Amazon Elastic Kubernetes Service. L'une vous permet d'utiliser les API Amazon EKS pour créer et gérer un cluster Amazon EKS. L'autre vous permet d'interagir avec votre cluster à l'aide de l'API Kubernetes et d'exécuter des tâches dans le cadre du flux de travail de votre application. Vous pouvez utiliser les intégrations d'API Kubernetes avec les clusters Amazon EKS créés à l'aide de Step Functions, avec les clusters Amazon EKS créés par l'outil eksctl ou la console Amazon EKS, ou avec des méthodes similaires. Pour plus d'informations, consultez la section Création d'un cluster Amazon EKS dans le guide de l'utilisateur Amazon EKS.

> **Note**
>
> L'intégration EKS de Step Functions ne prend en charge que les API Kubernetes avec accès public aux terminaux. Par défaut, les points de terminaison du serveur API des clusters EKS ont un accès public. Pour plus d'informations, consultez la section Contrôle d'accès aux points de terminaison du cluster Amazon EKS dans le guide de l'utilisateur Amazon EKS.

Step Functions ne met pas fin automatiquement à un cluster Amazon EKS si l'exécution est arrêtée. Si votre machine d'état s'arrête avant la fin de votre cluster Amazon EKS, celui-ci peut continuer à fonctionner indéfiniment et entraîner des frais supplémentaires. Pour éviter cela, assurez-vous que tout cluster Amazon EKS que vous créez est correctement résilié. Pour plus d'informations, consultez :

- Suppression d'un cluster dans le guide de l'utilisateur Amazon EKS.
- Exécuter une tâche (.sync)dans les modèles d'intégration des services.

> **Note**
>
> Il existe un quota pour la taille maximale des données d'entrée ou de résultat pour une tâche dans Step Functions. Cela vous limite à 256 Ko de données sous forme de chaîne codée en

UTF-8 lorsque vous envoyez ou recevez des données d'un autre service. veuillez consulter Quotas liés aux exécutions par les machines de l'État.

Intégrations d'API Kubernetes

Step Functions prend en charge les API Kubernetes suivantes :

RunJob

L'intégration des `eks:runJob` services vous permet d'exécuter une tâche sur votre cluster Amazon EKS. La `eks:runJob.sync` variante vous permet d'attendre que le travail soit terminé et, éventuellement, de récupérer les journaux.

Votre serveur d'API Kubernetes doit accorder des autorisations au rôle IAM utilisé par votre machine d'état. Pour plus d'informations, consultez Autorisations.

Pour le modèle Run a Job (`.sync`), le statut de la tâche est déterminé par un sondage. Step Functions interroge initialement à un rythme d'environ 1 sondage par minute. Ce taux finit par ralentir à environ 1 sondage toutes les 5 minutes. Si vous avez besoin d'un sondage plus fréquent ou d'un meilleur contrôle de la stratégie de sondage, vous pouvez utiliser l'`eks:call`intégration pour demander le statut de la tâche.

L'`eks:runJob`intégration est spécifique à `batch/v1` Kubernetes Jobs. Pour plus d'informations, consultez la section Jobs dans la documentation de Kubernetes. Si vous souhaitez gérer d'autres ressources Kubernetes, y compris des ressources personnalisées, utilisez l'`eks:call`intégration de services. Vous pouvez utiliser Step Functions pour créer des boucles d'interrogation, comme le montre l'the section called “Sondage pour connaître le statut du poste (Lambda,) AWS Batch”exemple de projet.

Les paramètres pris en charge incluent :

- `ClusterName`: nom du cluster Amazon EKS que vous souhaitez appeler.
  - `Type`: `String`
  - Obligatoire : oui
- `CertificateAuthority`: les données de certificat codées en Base64 nécessaires pour communiquer avec votre cluster. Vous pouvez obtenir cette valeur depuis la console Amazon EKS ou en utilisant l'DescribeClusterAPI Amazon EKS.
  - `Type`: `String`

  - Obligatoire : oui
- `Endpoint`: URL du point de terminaison de votre serveur d'API Kubernetes. Vous pouvez obtenir cette valeur depuis la console Amazon EKS ou en utilisant l'DescribeClusterAPI Amazon EKS.
  - `Type`: `String`
  - Obligatoire : oui
- `Namespace`: espace de noms dans lequel exécuter le job. S'il n'est pas fourni, l'espace de noms `default` est utilisé.
  - `Type`: `String`
  - Obligatoire : non
- `Job`: définition du Kubernetes Job. Consultez la section Jobs dans la documentation de Kubernetes.
  - `Type` : `JSON` ou `String`
  - Obligatoire : oui
- `LogOptions`: ensemble d'options permettant de contrôler la récupération facultative des journaux. Applicable uniquement si le modèle d'intégration du service Run a Job (.sync) est utilisé pour attendre la fin de la tâche.
  - `Type`: `JSON`
  - Obligatoire : non
  - Les journaux sont inclus dans la réponse sous la clé`logs`. La tâche peut comporter plusieurs modules, chacun contenant plusieurs conteneurs.

```
{
  ...
  "logs": {
    "pods": {
      "pod1": {
        "containers": {
          "container1": {
            "log": <log>
          },
          ...
        }
      },
      ...
    }
  }
```

  - La récupération des journaux est effectuée dans la mesure du possible. En cas d'erreur lors de la récupération d'un journal, les champs `error` et`cause`. `log`
- `LogOptions.RetrieveLogs`: active la récupération du journal une fois le travail terminé. Par défaut, les journaux ne sont pas récupérés.
  - `Type`: `Boolean`
  - Obligatoire : non
- `LogOptions.RawLogs`: S'il `RawLogs` est défini sur true, les journaux seront renvoyés sous forme de chaînes brutes sans qu'il soit nécessaire de les analyser en JSON. Par défaut, les journaux sont désérialisés au format JSON si possible. Dans certains cas, une telle analyse peut introduire des modifications indésirables, telles que la limitation de la précision des nombres contenant de nombreux chiffres.
  - `Type`: `Boolean`
  - Obligatoire : non
- `LogOptions.LogParameters`: L'API Read Log de l'API Kubernetes prend en charge les paramètres de requête pour contrôler la récupération des journaux. Par exemple, vous pouvez utiliser `tailLines` ou `limitBytes` limiter la taille des journaux récupérés tout en respectant le quota de taille des données de Step Functions. Pour plus d'informations, consultez la section Read Log de la référence des API Kubernetes.
  - `Type: Carte de String à List of Strings`
  - Obligatoire : non
  - Exemple :

```
"LogParameters": {
  "tailLines": [ "6" ]
}
```

L'exemple suivant inclut un `Task` état qui exécute une tâche, attend qu'elle soit terminée, puis extrait les journaux de la tâche :

```
{
  "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`

L'intégration des `eks:call` services vous permet d'utiliser l'API Kubernetes pour lire et écrire des objets de ressources Kubernetes via un point de terminaison d'API Kubernetes.

Votre serveur d'API Kubernetes doit accorder des autorisations au rôle IAM utilisé par votre machine d'état. Pour plus d'informations, consultez Autorisations.

Pour plus d'informations sur les opérations disponibles, consultez la référence des API Kubernetes.

Les paramètres pris en charge `Call` incluent :

- `ClusterName`: nom du cluster Amazon EKS que vous souhaitez appeler.
  - `Type` : chaîne
  - Obligatoire : oui
- `CertificateAuthority`: les données de certificat codées en Base64 nécessaires pour communiquer avec votre cluster. Vous pouvez obtenir cette valeur depuis la console Amazon EKS ou en utilisant l'DescribeClusterAPI Amazon EKS.
  - `Type`: `String`
  - Obligatoire : oui
- `Endpoint`: URL du point de terminaison de votre serveur d'API Kubernetes. Vous pouvez trouver cette valeur sur la console Amazon EKS ou en utilisant l' DescribeCluster API Amazon EKS.
  - `Type`: `String`
  - Obligatoire : oui
- `Method`: méthode HTTP de votre requête. Soit `GET`, `POST`, `PUT`, `DELETE`, `HEAD` ou `PATCH`.
  - `Type`: `String`
  - Obligatoire : oui
- `Path`: chemin HTTP de l'opération de l'API REST de Kubernetes.
  - `Type`: `String`
  - Obligatoire : oui
- `QueryParameters`: paramètres de requête HTTP de l'opération de l'API REST de Kubernetes.
  - `Type:` Carte de `String` à `List of Strings`
  - Obligatoire : non
  - Exemple :

```
"QueryParameters": {
  "labelSelector": [ "job-name=example-job" ]
}
```

- `RequestBody`: corps du message HTTP de l'opération de l'API REST de Kubernetes.
  - `Type` : `JSON` ou `String`
  - Obligatoire : non

Ce qui suit inclut un `Task` état qui permet `eks:call` de répertorier les pods appartenant à la tâche`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
    }
  }
}
```

Ce qui suit inclut un `Task` état utilisé `eks:call` pour supprimer la tâche `example-job` et définit le `propagationPolicy` pour garantir que les modules de la tâche sont également supprimés.

```
{
  "StartAt": "Call EKS",
  "States": {
```

```
    "Call EKS": {
      "Type": "Task",
      "Resource": "arn:aws:states:::eks:call",
      "Parameters": {
        "ClusterName": "MyCluster",
        "CertificateAuthority": "ANPAJ2UCCR6DPCEXAMPLE",
        "Endpoint": "https://444455556666.yl4.us-east-1.eks.amazonaws.com",
        "Method": "DELETE",
        "Path": "/apis/batch/v1/namespaces/default/jobs/example-job",
        "QueryParameters": {
          "propagationPolicy": [
            "Foreground"
          ]
        }
      },
      "End": true
    }
  }
}
```

API Amazon EKS prises en charge

Les API et syntaxes Amazon EKS prises en charge incluent :

- `CreateCluster`
  - Syntaxe de demande
  - Syntaxe de réponse

    Lorsqu'un cluster Amazon EKS est créé à l'aide de l'intégration de `eks:createCluster` services, le rôle IAM est ajouté à la table d'autorisation Kubernetes RBAC en tant qu'administrateur (avec les autorisations system:masters). Au départ, seule cette entité IAM peut appeler le serveur d'API Kubernetes. Pour plus d’informations, consultez :

    - Gestion des utilisateurs ou des rôles IAM pour votre cluster dans le guide de l'utilisateur Amazon EKS
    - La Autorisations section

    Amazon EKS utilise des rôles liés à un service qui contiennent les autorisations dont Amazon EKS a besoin pour appeler d'autres services en votre nom. Si ces rôles liés à un service n'existent pas déjà dans votre compte, vous devez ajouter l'`iam:CreateServiceLinkedRole`autorisation au rôle IAM utilisé par Step Functions. Pour

plus d'informations, consultez la section Utilisation des rôles liés à un service dans le guide de l'utilisateur Amazon EKS.

Le rôle IAM utilisé par Step Functions doit disposer des `iam:PassRole` autorisations nécessaires pour transmettre le rôle IAM du cluster à Amazon EKS. Pour plus d'informations, consultez le rôle IAM du cluster Amazon EKS dans le guide de l'utilisateur Amazon EKS.

- `DeleteCluster`
  - Syntaxe de demande
  - Syntaxe de réponse

  Vous devez supprimer tous les profils Fargate ou groupes de nœuds avant de supprimer un cluster.
- `CreateFargateProfile`
  - Syntaxe de demande
  - Syntaxe de réponse

  Amazon EKS utilise des rôles liés à un service qui contiennent les autorisations dont Amazon EKS a besoin pour appeler d'autres services en votre nom. Si ces rôles liés à un service n'existent pas déjà dans votre compte, vous devez ajouter l'`iam:CreateServiceLinkedRole`autorisation au rôle IAM utilisé par Step Functions. Pour plus d'informations, consultez la section Utilisation des rôles liés à un service dans le guide de l'utilisateur Amazon EKS.

  Amazon EKS on Fargate n'est peut-être pas disponible dans toutes les régions. Pour plus d'informations sur la disponibilité des régions, consultez la section sur Fargate dans le guide de l'utilisateur Amazon EKS.

  Le rôle IAM utilisé par Step Functions doit disposer des `iam:PassRole` autorisations nécessaires pour transmettre le rôle IAM d'exécution du pod à Amazon EKS. Pour plus d'informations, consultez la section Rôle d'exécution du Pod dans le guide de l'utilisateur Amazon EKS.
- `DeleteFargateProfile`
  - Syntaxe de demande
  - Syntaxe de réponse
- `CreateNodegroup`
  - Syntaxe de demande

- Syntaxe de réponse

  Amazon EKS utilise un rôle lié à un service qui contient les autorisations dont Amazon EKS a besoin pour appeler d'autres services en votre nom. Si ces rôles liés à un service n'existent pas déjà dans votre compte, vous devez ajouter l'`iam:CreateServiceLinkedRole`autorisation au rôle IAM utilisé par Step Functions. Pour plus d'informations, consultez la section Utilisation des rôles liés à un service dans le guide de l'utilisateur Amazon EKS.

  Le rôle IAM utilisé par Step Functions doit disposer des `iam:PassRole` autorisations nécessaires pour transmettre le rôle IAM du nœud à Amazon EKS. Pour plus d'informations, consultez la section Utilisation des rôles liés à un service dans le guide de l'utilisateur Amazon EKS.

- `DeleteNodegroup`
  - Syntaxe de demande
  - Syntaxe de réponse

Ce qui suit inclut un `Task` qui crée un cluster Amazon EKS.

```
{
  "StartAt": "CreateCluster.sync",
  "States": {
    "CreateCluster.sync": {
      "Type": "Task",
      "Resource": "arn:aws:states:::eks:createCluster.sync",
      "Parameters": {
        "Name": "MyCluster",
        "ResourcesVpcConfig": {
          "SubnetIds": [
            "subnet-053e7c47012341234",
            "subnet-027cfea4b12341234"
          ]
        },
        "RoleArn": "arn:aws:iam::123456789012:role/MyEKSClusterRole"
      },
      "End": true
    }
  }
}
```

Ce qui suit inclut un `Task` état qui supprime un cluster Amazon EKS.

```
{
  "StartAt": "DeleteCluster.sync",
  "States": {
    "DeleteCluster.sync": {
      "Type": "Task",
      "Resource": "arn:aws:states:::eks:deleteCluster.sync",
      "Parameters": {
        "Name": "MyCluster"
      },
      "End": true
    }
  }
}
```

Ce qui suit inclut un `Task` état qui crée un profil Fargate.

```
{
  "StartAt": "CreateFargateProfile.sync",
  "States": {
    "CreateFargateProfile.sync": {
      "Type": "Task",
      "Resource": "arn:aws:states:::eks:createFargateProfile.sync",
      "Parameters": {
        "ClusterName": "MyCluster",
        "FargateProfileName": "MyFargateProfile",
        "PodExecutionRoleArn": "arn:aws:iam::123456789012:role/
MyFargatePodExecutionRole",
        "Selectors": [{
            "Namespace": "my-namespace",
            "Labels": { "my-label": "my-value" }
          }]
      },
      "End": true
    }
  }
}
```

Ce qui suit inclut un `Task` état qui supprime un profil Fargate.

```
{
```

```
  "StartAt": "DeleteFargateProfile.sync",
  "States": {
    "DeleteFargateProfile.sync": {
      "Type": "Task",
      "Resource": "arn:aws:states:::eks:deleteFargateProfile.sync",
      "Parameters": {
        "ClusterName": "MyCluster",
        "FargateProfileName": "MyFargateProfile"
      },
      "End": true
    }
  }
}
```

Ce qui suit inclut un `Task` état qui crée un groupe de nœuds.

```
{
  "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
    }
  }
}
```

Ce qui suit inclut un `Task` état qui supprime un groupe de nœuds.

```
{
  "StartAt": "DeleteNodegroup.sync",
  "States": {
    "DeleteNodegroup.sync": {
      "Type": "Task",
      "Resource": "arn:aws:states:::eks:deleteNodegroup.sync",
      "Parameters": {
        "ClusterName": "MyCluster",
```

```
        "NodegroupName": "MyNodegroup"
      },
      "End": true
    }
  }
}
```

Autorisations

Lorsqu'un cluster Amazon EKS est créé à l'aide de l'intégration de `eks:createCluster` services, le rôle IAM est ajouté à la table d'autorisation Kubernetes RBAC en tant qu'administrateur, avec des autorisations. `system:masters` Au départ, seule cette entité IAM peut appeler le serveur d'API Kubernetes. Par exemple, vous ne pourrez pas utiliser kubectl pour interagir avec votre serveur d'API Kubernetes, sauf si vous assumez le même rôle que votre machine d'état Step Functions ou si vous configurez Kubernetes pour accorder des autorisations à des entités IAM supplémentaires. Pour plus d'informations, consultez la section Gestion des utilisateurs ou des rôles IAM pour votre cluster dans le guide de l'utilisateur Amazon EKS.

Vous pouvez ajouter des autorisations pour des entités IAM supplémentaires, telles que des utilisateurs ou des rôles, en les ajoutant à l'espace de noms `aws-auth` `ConfigMap` in the kube-system. Si vous créez votre cluster à partir de Step Functions, utilisez l'intégration `eks:call` de services.

Ce qui suit inclut un `Task` état qui crée un `aws-auth` `ConfigMap` et accorde `system:masters` l'autorisation à l'utilisateur `arn:aws:iam::123456789012:user/my-user` et au rôle `arn:aws:iam::123456789012:role/my-role` IAM.

```
{
  "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**
>
> L'ARN d'un rôle IAM peut s'afficher dans un format incluant le chemin /service-role/, tel que. `arn:aws:iam::123456789012:role/service-role/my-role` Ce jeton de chemin de rôle de service ne doit pas être inclus lors de la liste du rôle dans. `aws-auth`

Lorsque votre cluster est créé pour la `aws-auth` `ConfigMap` première fois, il n'existera pas, mais sera ajouté automatiquement si vous créez un profil Fargate. Vous pouvez récupérer la valeur actuelle de`aws-auth`, ajouter les autorisations supplémentaires et créer PUT une nouvelle version. Il est généralement plus facile de le créer `aws-auth` avant le profil Fargate.

Si votre cluster a été créé en dehors de Step Functions, vous pouvez configurer kubectl pour communiquer avec votre serveur d'API Kubernetes. Créez ensuite une nouvelle `aws-auth` `ConfigMap` utilisation `kubectl apply -f aws-auth.yaml` ou modifiez-en une qui existe déjà à l'aide de`kubectl edit -n kube-system configmap/aws-auth`. Pour plus d'informations, consultez :

- Créez un kubeconfig pour Amazon EKS dans le guide de l'utilisateur Amazon EKS.
- Gestion des utilisateurs ou des rôles IAM pour votre cluster dans le guide de l'utilisateur Amazon EKS.

Si votre rôle IAM ne dispose pas d'autorisations suffisantes dans Kubernetes, les intégrations de `eks:runJob` services `eks:call` ou de services échoueront avec l'erreur suivante :

```
Error:
EKS.401

Cause:
{
  "ResponseBody": {
    "kind": "Status",
    "apiVersion": "v1",
    "metadata": {},
    "status": "Failure",
    "message": "Unauthorized",
    "reason": "Unauthorized",
    "code": 401
  },
  "StatusCode": 401,
  "StatusText": "Unauthorized"
}
```

Appelez Amazon EMR avec Step Functions

Step Functions peut contrôler certains AWS services directement depuis Amazon States Language (ASL). Pour en savoir plus, consultez Utilisation avec d'autres services et Transmettre des paramètres à une API de service.

> **En quoi l'intégration optimisée d'Amazon EMR est différente de l'intégration du SDK Amazon EMR AWS**
>
> L'intégration optimisée du service Amazon EMR comprend un ensemble personnalisé d'API qui englobe les API Amazon EMR sous-jacentes, décrites ci-dessous. De ce fait, elle est très différente de l'intégration du service Amazon EMR AWS SDK. De plus, le modèle Exécuter une tâche (.sync) d'intégration est pris en charge.

Pour intégrer AWS Step Functions Amazon EMR, vous utilisez les API d'intégration du service Amazon EMR fournies. Les API d'intégration de services sont similaires aux API Amazon EMR

correspondantes, avec quelques différences dans les champs transmis et dans les réponses renvoyées.

Step Functions ne met pas automatiquement fin à un cluster Amazon EMR si l'exécution est arrêtée. Si votre machine d'état s'arrête avant la fin de votre cluster Amazon EMR, celui-ci peut continuer à fonctionner indéfiniment et entraîner des frais supplémentaires. Pour éviter cela, assurez-vous que tout cluster Amazon EMR que vous créez est correctement résilié. Pour plus d'informations, consultez :

- Contrôlez la résiliation du cluster dans le guide de l'utilisateur Amazon EMR.
- La Exécuter une tâche (.sync) section Modèles d'intégration des services.

> **Note**
>
> À partir de `emr-5.28.0`, vous pouvez spécifier le paramètre `StepConcurrencyLevel` lors de la création d'un cluster pour permettre à plusieurs étapes de s'exécuter en parallèle sur un seul cluster. Vous pouvez utiliser les Step Functions `Map` et `Parallel` les états pour soumettre des travaux en parallèle au cluster.

La disponibilité de l'intégration du service Amazon EMR dépend de la disponibilité des API Amazon EMR. Consultez la documentation Amazon EMR pour connaître les restrictions applicables dans certaines régions.

> **Note**
>
> Pour l'intégration à Amazon EMR, Step Functions dispose d'une fréquence d'interrogation des offres d'emploi codée en dur de 60 secondes pendant les 10 premières minutes, puis de 300 secondes par la suite.

Le tableau suivant décrit les différences entre chaque API d'intégration de services et l'API Amazon EMR correspondante.

API d'intégration du service Amazon EMR et API Amazon EMR correspondantes

API d'intégration du service Amazon EMR	API EMR correspondante	Différences
createCluster Crée et démarre l'exécution d'un cluster (flux de travail). Amazon EMR est directement lié à un type unique de rôle IAM connu sous le nom de rôle lié à un service. Pour que `createCluster` et `createCluster.sync` fonctionnent, vous devez avoir configuré les autorisations nécessaires pour créer le rôle lié au service `AWSServiceRoleForEMRCleanup`. Pour plus d'informations à ce sujet, y compris une déclaration que vous pouvez ajouter à votre politique d'autorisations IAM, consultez Utilisation du rôle lié à un service pour Amazon EMR.	runJobFlow	`createCluster` utilise la même syntaxe de demande que runJobFlow, à l'exception de ce qui suit : • Le champ `Instances.KeepJobFlowAliveWhenNoSteps` est obligatoire et doit avoir la valeur booléenne `TRUE`. • Le champ `Steps` n'est pas autorisé. • Le champ `Instances.InstanceFleets[index].Name` doit être fourni et doit être unique si l'API de connecteur `modifyInstanceFleetByName` facultative est utilisée. • Le champ `Instances.InstanceGroups[index].Name` doit être fourni et doit être unique si l'API facultative `modifyInstanceGroupByName` est utilisée. La réponse est la suivante : `{ "ClusterId": "string"`

API d'intégration du service Amazon EMR	API EMR correspondante	Différences

		`}`
		
		Amazon EMR utilise ceci :
		
		`{ "JobFlowId": "string" }`

createCluster.sync Crée et démarre l'exécution d'un cluster (flux de travail).	runJobFlow	Le même que `createCluster` , mais attend que le cluster atteigne l'état `WAITING`.

setClusterTerminationProtection
 	
Verrouille un cluster (flux de travail) afin que les instances EC2 du cluster ne puissent pas être arrêtées par l'intervention de l'utilisateur, un appel d'API ou une erreur de flux de travail.
	setTerminationProtection	La demande utilise ceci :
		
		`{ "ClusterId": "string" }`
		
		Amazon EMR utilise ceci :
		
		`{ "JobFlowIds": ["string"] }`

API d'intégration du service Amazon EMR	API EMR correspondante	Différences

terminateCluster
 	
Arrête un cluster (flux de travail).
	terminateJobFlows	La demande utilise ceci :
		
		`{ "ClusterId": "string" }`
		
		Amazon EMR utilise ceci :
		
		`{ "JobFlowIds": ["string"] }`

terminateCluster.sync Arrête un cluster (flux de travail).	terminateJobFlows	Identique à `terminateCluster` , mais attend que le cluster ait terminé.

API d'intégration du service Amazon EMR	API EMR correspondante	Différences

addStep
 	
Ajoute une nouvelle étape à un cluster en cours d'exécution.
 	
Facultativement, vous pouvez également spécifier le `ExecutionRoleArn` paramètre lors de l'utilisation de cette API.
	addJobFlowÉtapes	La demande utilise la clé`"ClusterId"` . Amazon EMR utilise. `"JobFlowId"` La demande utilise une seule étape.
		
		`{ "Step": <"StepConfig object"> }`
		
		Amazon EMR utilise ceci :
		
		`{ "Steps": [<StepConfig objects>] }`
		
		La réponse est la suivante :
		
		`{ "StepId": "string" }`
		
		Amazon EMR renvoie ce qui suit :
		
		`{ "StepIds": [<strings>] }`

API d'intégration du service Amazon EMR	API EMR correspondante	Différences
addStep.sync Ajoute une nouvelle étape à un cluster en cours d'exécution. Facultativement, vous pouvez également spécifier le `ExecutionRoleArn` paramètre lors de l'utilisation de cette API.	addJobFlowÉtapes	Identique à `addStep`, mais attend que l'étape se termine.

API d'intégration du service Amazon EMR	API EMR correspondante	Différences

cancelStep
 	
Annule une étape en attente dans un cluster en cours d'exécution
	cancelSteps	La demande utilise ceci :
		
		`{ "StepId": "string" }`
		
		Amazon EMR utilise ceci :
		
		`{ "StepIds": [<strings >] }`
		
		La réponse est la suivante :
		
		`{ "CancelStepsInfo": <CancelStepsInfo object> }`
		
		Amazon EMR utilise ceci :
		
		`{ "CancelStepsInfoLi st": [<CancelStepsInfo objects>] }`

API d'intégration du service Amazon EMR	API EMR correspondante	Différences
modifyInstanceFleetByName Modifie les capacités d'instances à la demande et d'instances ponctuelles cibles pour le parc d'instances avec l'`InstanceFleetName` spécifié.	modifyInstanceFleet	La demande est la même que pour `modifyInstanceFleet` , sauf pour ce qui suit : • Le champ `Instance.InstanceFleetId` n'est pas autorisé. • Lors de l'exécution, le `InstanceFleetId` est déterminé automatiquement par l'intégration du service en appelant `ListInstanceFleets` et en analysant le résultat.

API d'intégration du service Amazon EMR	API EMR correspondante	Différences

modifyInstanceGroupByName
 	
Modifie le nombre de nœuds et de paramètres de configuration d'un groupe d'instances.
	modifyInstanceGroups	La demande est la suivante :
		
		`{ "ClusterId": "string", "InstanceGroup": <InstanceGroupModifyConfig object> }`
		
		Amazon EMR utilise une liste :
		
		`{ "ClusterId": ["string"], "InstanceGroups": [<InstanceGroupModifyConfig objects>] }`
		
		Dans l'objet `InstanceGroupModifyConfig`, le champ `InstanceGroupId` n'est pas autorisé.
		
		Un nouveau champ, `InstanceGroupName`, a été ajouté. Lors de l'exécution, le `InstanceGroupId` est déterminé automatiquement par l'intégration du service en appelant `ListInstanceGroups` et en analysant le résultat.

L'exemple suivant inclut un état `Task` qui crée un cluster.

```
"Create_Cluster": {
    "Type": "Task",
    "Resource": "arn:aws:states:::elasticmapreduce:createCluster.sync",
    "Parameters": {
        "Name": "MyWorkflowCluster",
        "VisibleToAllUsers": true,
        "ReleaseLabel": "emr-5.28.0",
        "Applications": [
            {
                "Name": "Hive"
            }
        ],
        "ServiceRole": "EMR_DefaultRole",
        "JobFlowRole": "EMR_EC2_DefaultRole",
        "LogUri": "s3n://aws-logs-123456789012-us-east-1/elasticmapreduce/",
        "Instances": {
            "KeepJobFlowAliveWhenNoSteps": true,
            "InstanceFleets": [
                {
                    "InstanceFleetType": "MASTER",
                    "Name": "MASTER",
                    "TargetOnDemandCapacity": 1,
                    "InstanceTypeConfigs": [
                        {
                            "InstanceType": "m4.xlarge"
                        }
                    ]
                },
                {
                    "InstanceFleetType": "CORE",
                    "Name": "CORE",
                    "TargetOnDemandCapacity": 1,
                    "InstanceTypeConfigs": [
                        {
                            "InstanceType": "m4.xlarge"
                        }
                    ]
                }
            ]
        }
    },
    "End": true
```

```
}
```

Ce qui suit inclut un état `Task` qui permet la protection contre la résiliation.

```
"Enable_Termination_Protection": {
    "Type": "Task",
    "Resource": "arn:aws:states:::elasticmapreduce:setClusterTerminationProtection",
    "Parameters": {
        "ClusterId.$": "$.ClusterId",
        "TerminationProtected": true
    },
    "End": true
}
```

Ce qui suit inclut un état `Task` qui soumet une étape à un cluster.

```
"Step_One": {
    "Type": "Task",
    "Resource": "arn:aws:states:::elasticmapreduce:addStep.sync",
    "Parameters": {
        "ClusterId.$": "$.ClusterId",
        "ExecutionRoleArn": "arn:aws:iam::123456789012:role/myEMR-execution-role",
        "Step": {
            "Name": "The first step",
            "ActionOnFailure": "CONTINUE",
            "HadoopJarStep": {
                "Jar": "command-runner.jar",
                "Args": [
                    "hive-script",
                    "--run-hive-script",
                    "--args",
                    "-f",
                    "s3://<region>.elasticmapreduce.samples/cloudfront/code/
Hive_CloudFront.q",
                    "-d",
                    "INPUT=s3://<region>.elasticmapreduce.samples",
                    "-d",
                    "OUTPUT=s3://<mybucket>/MyHiveQueryResults/"
                ]
            }
        }
    },
    "End": true
```

```
}
```

Ce qui suit inclut un état `Task` qui annule une étape.

```
"Cancel_Step_One": {
    "Type": "Task",
    "Resource": "arn:aws:states:::elasticmapreduce:cancelStep",
    "Parameters": {
        "ClusterId.$": "$.ClusterId",
        "StepId.$": "$.AddStepsResult.StepId"
    },
    "End": true
}
```

Ce qui suit inclut un état `Task` qui met fin à un cluster.

```
"Terminate_Cluster": {
    "Type": "Task",
    "Resource": "arn:aws:states:::elasticmapreduce:terminateCluster.sync",
    "Parameters": {
        "ClusterId.$": "$.ClusterId"
    },
    "End": true
}
```

Ce qui suit inclut un état `Task` qui met à l'échelle un cluster vers le haut ou vers le bas pour un groupe d'instances.

```
"ModifyInstanceGroupByName": {
    "Type": "Task",
    "Resource": "arn:aws:states:::elasticmapreduce:modifyInstanceGroupByName",
    "Parameters": {
        "ClusterId": "j-1234567890123",
        "InstanceGroupName": "MyCoreGroup",
        "InstanceGroup": {
            "InstanceCount": 8
        }
    },
    "End": true
}
```

Ce qui suit inclut un état `Task` qui met à l'échelle un cluster vers le haut ou vers le bas pour une flotte d'instances.

```
"ModifyInstanceFleetByName": {
    "Type": "Task",
    "Resource": "arn:aws:states:::elasticmapreduce:modifyInstanceFleetByName",
    "Parameters": {
        "ClusterId": "j-1234567890123",
        "InstanceFleetName": "MyCoreFleet",
        "InstanceFleet": {
            "TargetOnDemandCapacity": 8,
            "TargetSpotCapacity": 0
        }
    },
    "End": true
}
```

Pour plus d'informations sur la configuration IAM des autorisations lors de l'utilisation Step Functions avec d'autres AWS services, consultezPolitiques IAM pour les services intégrés.

Appelez Amazon EMR sur EKS avec AWS Step Functions

Step Functions peut contrôler certains AWS services directement depuis Amazon States Language (ASL). Pour en savoir plus, consultez Utilisation avec d'autres services et Transmettre des paramètres à une API de service.

> En quoi l'intégration optimisée d'Amazon EMR sur EKS est différente de l'intégration Amazon EMR sur EKS SDK AWS
>
> - Le modèle Exécuter une tâche (.sync) d'intégration est pris en charge.
> - Il n'y a aucune optimisation pour le modèle Réponse à la requête d'intégration.
> - Le modèle Attendre un rappel avec le jeton de tâche d'intégration n'est pas pris en charge.

> **Note**
>
> Pour l'intégration à Amazon EMR, Step Functions dispose d'une fréquence d'interrogation des offres d'emploi codée en dur de 60 secondes pendant les 10 premières minutes, puis de 300 secondes par la suite.

Pour intégrer AWS Step Functions Amazon EMR on EKS, utilisez les API d'intégration de services Amazon EMR on EKS. Les API d'intégration de services sont les mêmes que les API Amazon EMR on EKS correspondantes, mais toutes les API ne prennent pas en charge tous les modèles d'intégration, comme indiqué dans le tableau suivant.

API	Réponse à la demande	Exécuter une tâche (.sync)
CreateVirtualCluster	✓	
DeleteVirtualCluster	✓	✓
StartJobRun	✓	✓

API Amazon EMR sur EKS prises en charge :

> **Note**
>
> Il existe un quota pour la taille maximale des données d'entrée ou de résultat pour une tâche dans Step Functions. Cela vous limite à 256 Ko de données sous forme de chaîne codée en UTF-8 lorsque vous envoyez ou recevez des données d'un autre service. veuillez consulter Quotas liés aux exécutions par les machines de l'État.

- `CreateVirtualCluster`
  - Syntaxe de demande
  - Paramètres pris en charge
  - Syntaxe de réponse
- `DeleteVirtualCluster`
  - Syntaxe de demande

- Paramètres pris en charge
- Syntaxe de réponse
- `StartJobRun`
  - Syntaxe de demande
  - Paramètres pris en charge
  - Syntaxe de réponse

Ce qui suit inclut un `Task` état qui crée un cluster virtuel.

```
"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
}
```

Ce qui suit inclut un `Task` état qui soumet une tâche à un cluster virtuel et attend qu'elle soit terminée.

```
"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
}
```

Ce qui suit inclut un `Task` état qui supprime un cluster virtuel et attend que la suppression soit terminée.

```
"Delete_Virtual_Cluster": {
  "Type": "Task",
```

```
  "Resource": "arn:aws:states:::emr-containers:deleteVirtualCluster.sync",
  "Parameters": {
    "Id.$": "$.VirtualClusterId"
  },
  "End": true
}
```

Pour plus d'informations sur la configuration IAM des autorisations lors de l'utilisation Step Functions avec d'autres AWS services, consultezPolitiques IAM pour les services intégrés.

Appelez Amazon EMR Serverless avec Step Functions

Step Functions peut contrôler certains AWS services directement depuis Amazon States Language (ASL). Pour en savoir plus, consultez Utilisation avec d'autres services et Transmettre des paramètres à une API de service.

> En quoi l'EMR Serverlessintégration optimisée est différente de l'intégration du EMR Serverless AWS SDK
>
> - L'intégration EMR Serverless de services optimisée comprend un ensemble personnalisé d'API qui encapsulent les EMR Serverless API sous-jacentes. En raison de cette personnalisation, l'EMR Serverlessintégration optimisée diffère considérablement de l'intégration des services du EMR Serverless AWS SDK. De plus, l'EMR Serverlessintégration optimisée prend en charge le modèle Exécuter une tâche (.sync) d'intégration.
> - Le modèle Attendre un rappel avec le jeton de tâche d'intégration n'est pas pris en charge.

Dans cette rubrique

EMR ServerlessAPI d'intégration de services

Pour effectuer AWS Step Functions l'intégrationEMR Serverless, vous pouvez utiliser les six API d'intégration EMR Serverless de services suivantes. Ces API d'intégration de services sont similaires aux EMR Serverless API correspondantes, avec quelques différences dans les champs transmis et dans les réponses renvoyées.

Les différences entre chaque API d'intégration de service et son API EMR Serverless correspondante sont illustrées dans le tableau suivant :

EMR ServerlessAPI d'intégration de services et EMR Serverless API correspondantes

EMR ServerlessAPI d'intégration de services	EMR ServerlessAPI correspondante	Différences
Création d'une application Crée une application. EMR Serverlessest lié à un type de IAM rôle unique appelé rôle lié à un service. Pour que `createApplication` et `createApplication.sync` fonctionnent, vous devez avoir configuré les autorisations nécessaires pour créer le rôle lié au service `AWSServiceRoleForAmazonEMRServerless`. Pour plus d'informations à ce sujet, y compris une déclaration que vous pouvez ajouter à votre politique d'IAMautorisations, consultez la section Utilisation de rôles liés à un service pour. EMR Serverless	CreateApplication	Aucun
CreateApplication.sync Crée une application.	CreateApplication	Aucune différence entre les demandes et les réponses de l'EMR ServerlessAPI et de l'API d'intégration des EMR Serverless services. Cependant, CreateApplication.

EMR ServerlessAPI d'intégration de services	EMR ServerlessAPI correspondante	Différences
		sync attend que l'application atteigne cet état. `CREATED`
Démarrer l'application Démarre une application spécifiée et initialise la capacité initiale de l'application si elle est configurée.	StartApplication	La réponse de l'EMR ServerlessAPI ne contient aucune donnée, mais la réponse de l'API d'intégration des EMR Serverless services inclut les données suivantes. `{ "ApplicationId": "string" }`
Démarrez Application.sync Démarre une application spécifiée et initialise la capacité initiale si elle est configurée.	StartApplication	La réponse de l'EMR ServerlessAPI ne contient aucune donnée, mais la réponse de l'API d'intégration des EMR Serverless services inclut les données suivantes. `{ "ApplicationId": "string" }` StartApplication.sync attend également que l'application atteigne son état. `STARTED`

EMR ServerlessAPI d'intégration de services	EMR ServerlessAPI correspondante	Différences
Arrêter l'application Arrête une application spécifiée et libère sa capacité initiale si elle est configurée. Toutes les tâches planifiées et en cours d'exécution doivent être terminées ou annulées avant d'arrêter une application.	StopApplication	La réponse de l'EMR ServerlessAPI ne contient aucune donnée, mais la réponse de l'API d'intégration des EMR Serverless services inclut les données suivantes. `{ "ApplicationId": "string" }`
Arrêter Application.sync Arrête une application spécifiée et libère sa capacité initiale si elle est configurée. Toutes les tâches planifiées et en cours d'exécution doivent être terminées ou annulées avant d'arrêter une application.	StopApplication	La réponse de l'EMR ServerlessAPI ne contient aucune donnée, mais la réponse de l'API d'intégration des EMR Serverless services inclut les données suivantes. `{ "ApplicationId": "string" }` StopApplication.sync attend également que l'application atteigne l'état. `STOPPED`

EMR ServerlessAPI d'intégration de services	EMR ServerlessAPI correspondante	Différences
Supprimer l'application Supprime une application. Une application doit être à l'`CREATED`état `STOPPED` ou pour être supprimée.	DeleteApplication	La réponse de l'EMR ServerlessAPI ne contient aucune donnée, mais la réponse de l'API d'intégration des EMR Serverless services inclut les données suivantes. `{ "ApplicationId": "string" }`
Supprimer Application.sync Supprime une application. Une application doit être à l'`CREATED`état `STOPPED` ou pour être supprimée.	DeleteApplication	La réponse de l'EMR ServerlessAPI ne contient aucune donnée, mais la réponse de l'API d'intégration des EMR Serverless services inclut les données suivantes. `{ "ApplicationId": "string" }` StopApplication.sync attend également que l'application atteigne l'état. `TERMINATED`
startJobRun Démarre l'exécution d'une tâche.	StartJobRun	Aucun

EMR ServerlessAPI d'intégration de services	EMR ServerlessAPI correspondante	Différences
startJobRun.sync Démarre l'exécution d'une tâche.	StartJobRun	Aucune différence entre les demandes et les réponses de l'EMR ServerlessAPI et de l'API d'intégration des EMR Serverless services. Cependant, startJobRun.sync attend que l'application atteigne cet état. `SUCCESS`
cancelJobRun Annule l'exécution d'une tâche.	CancelJobRun	Aucun
cancelJobRun.sync Annule l'exécution d'une tâche.	CancelJobRun	Aucune différence entre les demandes et les réponses de l'EMR ServerlessAPI et de l'API d'intégration des EMR Serverless services. Cependant, cancelJob Run.sync attend que l'application atteigne cet état. `CANCELLED`

Cas d'utilisation de l'intégration EMR sans serveur

Pour l'intégration optimisée des EMR Serverless services, nous vous recommandons de créer une seule application, puis de l'utiliser pour exécuter plusieurs tâches. Par exemple, dans une machine à états unique, vous pouvez inclure plusieurs startJobRundemandes, qui utilisent toutes la même application. Les exemples État de la tâche d'états suivants présentent des cas d'utilisation dans lesquels intégrer des EMR Serverless APIStep Functions. Pour plus d'informations sur les autres cas d'utilisation deEMR Serverless, voir Qu'est-ce que Amazon EMR Serverless.

> **Tip**
>
> Pour déployer un exemple de machine à états qui s'intègre EMR Serverless pour exécuter plusieurs tâches sur votre Compte AWS, consultezExécuter une EMR Serverless tâche.

- Création d’une application
- Lancer une application
- Arrêter une application
- Supprimer une application
- Commencer un emploi dans une application
- Annuler une offre d'emploi dans une candidature

Pour plus d'informations sur la configuration IAM des autorisations lors de l'utilisation Step Functions avec d'autres AWS services, consultezPolitiques IAM pour les services intégrés.

Dans les exemples illustrés dans les cas d'utilisation suivants, remplacez le texte en *`italique par des informations`* spécifiques à votre ressource. Remplacez par exemple *`yourApplicationId`*par l'ID de votre EMR Serverless application, tel que`00yv7iv71inak893`.

Création d’une application

L'exemple d'état de tâche suivant crée une application à l'aide de l'API d'intégration du service CreateApplication.sync.

```
"Create_Application": {
    "Type": "Task",
    "Resource": "arn:aws:states:::emr-serverless:createApplication.sync",
    "Parameters": {
        "Name": "MyApplication",
        "ReleaseLabel": "emr-6.9.0",
        "Type": "SPARK"
    },
    "End": true
}
```

Lancer une application

L'exemple d'état de tâche suivant démarre une application à l'aide de l'API d'intégration du service StartApplication.sync.

```
"Start_Application": {
    "Type": "Task",
    "Resource": "arn:aws:states:::emr-serverless:startApplication.sync",
    "Parameters": {
        "ApplicationId": "yourApplicationId"
    },
    "End": true
}
```

Arrêter une application

L'exemple d'état de tâche suivant arrête une application à l'aide de l'API d'intégration du service StopApplication.sync.

```
"Stop_Application": {
    "Type": "Task",
    "Resource": "arn:aws:states:::emr-serverless:stopApplication.sync",
    "Parameters": {
        "ApplicationId": "yourApplicationId"
    },
    "End": true
}
```

Supprimer une application

L'exemple d'état de tâche suivant supprime une application à l'aide de l'API d'intégration du service DeleteApplication.sync.

```
"Delete_Application": {
    "Type": "Task",
    "Resource": "arn:aws:states:::emr-serverless:deleteApplication.sync",
    "Parameters": {
        "ApplicationId": "yourApplicationId"
    },
    "End": true
}
```

Commencer un emploi dans une application

L'exemple d'état de tâche suivant démarre une tâche dans une application à l'aide de l'API d'intégration du service startJobRun.sync.

```
"Start_Job": {
    "Type": "Task",
    "Resource": "arn:aws:states:::emr-serverless:startJobRun.sync",
    "Parameters": {
        "ApplicationId": "yourApplicationId",
        "ExecutionRoleArn": "arn:aws:iam::123456789012:role/myEMRServerless-execution-
role",
        "JobDriver": {
            "SparkSubmit": {
                "EntryPoint": "s3://<mybucket>/sample.py",
                "EntryPointArguments": ["1"],
                "SparkSubmitParameters": "--conf spark.executor.cores=4 --conf
 spark.executor.memory=4g --conf spark.driver.cores=2 --conf spark.driver.memory=4g --
conf spark.executor.instances=1"
            }
        }
    },
    "End": true
}
```

Annuler une offre d'emploi dans une candidature

L'exemple d'état de tâche suivant annule une tâche dans une application à l'aide de l'API d'intégration du service cancelJobRun.sync.

```
"Cancel_Job": {
    "Type": "Task",
    "Resource": "arn:aws:states:::emr-serverless:cancelJobRun.sync",
    "Parameters": {
        "ApplicationId.$": "$.ApplicationId",
        "JobRunId.$": "$.JobRunId"
    },
    "End": true
}
```

Appelez EventBridge avec Step Functions

Step Functions peut contrôler certains AWS services directement depuis Amazon States Language (ASL). Pour en savoir plus, consultez Utilisation avec d'autres services et Transmettre des paramètres à une API de service.

> En quoi l' EventBridge intégration optimisée est différente de l'intégration du EventBridge AWS SDK
>
> - L'ARN d'exécution et l'ARN de la machine à états sont automatiquement ajoutés au `Resources` champ de chacun`PutEventsRequestEntry`.
> - Si la réponse de `PutEvents` contient une valeur différente de zéro`FailedEntryCount`, l'`Task`état échoue avec l'erreur`EventBridge.FailedEntry`.

Pour plus d'informations sur la configuration IAM des autorisations lors de l'utilisation Step Functions avec d'autres AWS services, consultezPolitiques IAM pour les services intégrés.

Step Functions fournit une API d'intégration de services pour l'intégration à Amazon EventBridge. Cela vous permet de créer des applications pilotées par des événements en envoyant des événements personnalisés directement à partir des flux de travail de Step Functions.

Pour utiliser l'`PutEvents`API, vous devez créer une EventBridge règle dans votre compte qui correspond au modèle spécifique des événements que vous allez envoyer. Par exemple, vous pouvez accorder les accès suivants :

- Créez une fonction Lambda dans votre compte qui reçoit et imprime un événement correspondant à une EventBridge règle.
- Créez une EventBridge règle dans votre compte sur le bus d'événements par défaut qui correspond à un modèle d'événement spécifique et cible la fonction Lambda.

Pour plus d’informations, consultez :

- Ajouter EventBridge des événements Amazon PutEvents dans le guide de EventBridge l'utilisateur.
- Attendre un rappel avec le jeton de tâchedans les modèles d'intégration des services.

> **Note**
>
> Il existe un quota pour la taille maximale des données d'entrée ou de résultat pour une tâche dans Step Functions. Cela vous limite à 256 Ko de données sous forme de chaîne codée en UTF-8 lorsque vous envoyez ou recevez des données d'un autre service. veuillez consulter Quotas liés aux exécutions par les machines de l'État.

EventBridge API prise en charge

EventBridge L'API et la syntaxe prises en charge incluent :

- `PutEvents`
  - Syntaxe de demande
  - Paramètre pris en charge :
    - `Entries`
  - Syntaxe de réponse

Ce qui suit inclut un `Task` qui envoie un événement personnalisé :

```
{
  "Type": "Task",
  "Resource": "arn:aws:states:::events:putEvents",
  "Parameters": {
    "Entries": [
      {
        "Detail": {
          "Message": "MyMessage"
        },
        "DetailType": "MyDetailType",
        "EventBusName": "MyEventBus",
        "Source": "my.source"
      }
    ]
  },
  "End": true
}
```

Gestion des erreurs

L'`PutEvents`API accepte un tableau d'entrées en entrée, puis renvoie un tableau d'entrées de résultats. Tant que l'`PutEvents`action est réussie, `PutEvents` renvoie une réponse HTTP 200, même si une ou plusieurs entrées ont échoué. `PutEvents`renvoie le nombre de saisies échouées dans le `FailedEntryCount` champ.

Step Functions vérifie si le `FailedEntryCount` est supérieur à zéro. S'il est supérieur à zéro, Step Functions ne correspond pas à l'état avec l'erreur`EventBridge.FailedEntry`. Cela vous permet d'utiliser la gestion des erreurs intégrée de Step Functions sur les états des tâches pour détecter ou réessayer en cas d'échec des entrées, plutôt que d'avoir à utiliser un état supplémentaire pour analyser la `FailedEntryCount` réponse.

> **Note**
>
> Si vous avez implémenté l'idempotencie et que vous pouvez réessayer toutes les entrées en toute sécurité, vous pouvez utiliser la logique de nouvelle tentative de Step Functions. Step Functions ne supprime pas les entrées réussies du tableau `PutEvents` d'entrées avant de réessayer. Au lieu de cela, il réessaie avec le tableau d'entrées d'origine.

Gérez les AWS Glue tâches avec Step Functions

Step Functions peut contrôler certains AWS services directement depuis Amazon States Language (ASL). Pour en savoir plus, consultez Utilisation avec d'autres services et Transmettre des paramètres à une API de service.

> **En quoi l' AWS Glue intégration optimisée est différente de l'intégration du AWS GlueAWS SDK**
>
> - Le modèle Exécuter une tâche (.sync) d'intégration est disponible.
> - Le `JobName` champ est extrait de la demande et inséré dans la réponse, qui contient normalement uniquement`JobRunID`.

AWS Glue API prise en charge :

- `StartJobRun`

> **Les paramètres in Step Functions sont exprimés en PascalCase**
>
> Même si l'API de service native se trouve dans CamelCase, par exemple l'`startSyncExecution`action d'API, vous spécifiez des paramètres PascalCase dans, tels que :. `StateMachineArn`

Ce qui suit inclut un `Task` état qui démarre une AWS Glue tâche.

```
"Glue StartJobRun": {
      "Type": "Task",
      "Resource": "arn:aws:states:::glue:startJobRun.sync",
      "Parameters": {
        "JobName": "GlueJob-JTrRO5l98qMG"
      },
      "Next": "ValidateOutput"
    },
```

Pour plus d'informations sur la configuration IAM des autorisations lors de l'utilisation Step Functions avec d'autres AWS services, consultezPolitiques IAM pour les services intégrés.

Gérez les AWS Glue DataBrew tâches avec Step Functions

Step Functions peut contrôler certains AWS services directement depuis Amazon States Language (ASL). Pour en savoir plus, consultez Utilisation avec d'autres services et Transmettre des paramètres à une API de service.

Vous pouvez utiliser cette DataBrew intégration pour ajouter des étapes de nettoyage et de normalisation des données à vos flux de travail d'analyse et d'apprentissage automatique.

DataBrew API prise en charge :

- `StartJobRun`

Ce qui suit inclut un `Task` état qui démarre une tâche de demande-réponse DataBrew.

```
"DataBrew StartJobRun": {
            "Type": "Task",
            "Resource": "arn:aws:states:::databrew:startJobRun",
            "Parameters": {
```

```
                "Name": "sample-proj-job-1"
             },
             "Next": "NEXT_STATE"
          },
```

Ce qui suit inclut un `Task` état qui démarre une DataBrew tâche de synchronisation.

```
"DataBrew StartJobRun": {
           "Type": "Task",
           "Resource": "arn:aws:states:::databrew:startJobRun.sync",
           "Parameters": {
              "Name": "sample-proj-job-1"
           },
           "Next": "NEXT_STATE"
          },
```

Pour plus d'informations sur la configuration IAM des autorisations lors de l'utilisation Step Functions avec d'autres AWS services, consultezPolitiques IAM pour les services intégrés.

Invoquez Lambda avec Step Functions

Step Functions peut contrôler certains AWS services directement depuis Amazon States Language (ASL). Pour en savoir plus, consultez Utilisation avec d'autres services et Transmettre des paramètres à une API de service.

> En quoi l'intégration Lambda optimisée est différente de l'intégration du SDK Lambda AWS
>
> - Le `Payload` champ de la réponse est analysé de Json échappé à Json.
> - Si la réponse contient le champ `FunctionError` ou si une exception est déclenchée dans la fonction Lambda, la tâche échoue.

Pour de plus amples informations sur la gestion des états d'entrée, des états de sortie et des résultats, veuillez consulter Traitement des entrées et des sorties dans Step Functions.

AWS Lambda API prises en charge :

- `Invoke`

- Syntaxe de la demande
- Paramètres pris en charge
  - `ClientContext`
  - `FunctionName`
  - `InvocationType`
  - `Qualifier`
  - `Payload`
- Syntaxe de réponse

> **Les paramètres in Step Functions sont exprimés en PascalCase**
>
> Même si l'API de service native se trouve dans CamelCase, par exemple l'`startSyncExecution`action d'API, vous spécifiez des paramètres PascalCase dans, tels que :. `StateMachineArn`

Ce qui suit inclut un `Task` état qui invoque une fonction Lambda.

```
{
   "StartAt":"CallLambda",
   "States":{
      "CallLambda":{
         "Type":"Task",
         "Resource":"arn:aws:states:::lambda:invoke",
         "Parameters":{
            "FunctionName":"arn:aws:lambda:us-east-1:123456789012:function:MyFunction"
         },
         "End":true
      }
   }
}
```

L'exemple suivant inclut un état `Task` qui implémente le modèle d'intégration de service de rappel.

```
{
   "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
      }
   }
}
```

Lorsque vous appelez une fonction Lambda, l'exécution attend que la fonction soit terminée. Si vous appelez la fonction Lambda avec une tâche de rappel, le délai d'expiration du rythme cardiaque ne commence à être compté qu'une fois que la fonction Lambda a terminé son exécution et renvoyé un résultat. Tant que la fonction Lambda est exécutée, le délai d'expiration du rythme cardiaque n'est pas appliqué.

Il est également possible d'appeler Lambda de manière asynchrone à l'aide du `InvocationType` paramètre, comme le montre l'exemple suivant :

> **Note**
>
> Pour les appels asynchrones de fonctions Lambda, le délai d'expiration du rythme cardiaque commence immédiatement.

```
{

  "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
    }
  }
}
```

Lorsque le `Task` résultat est renvoyé, la sortie de la fonction est imbriquée dans un dictionnaire de métadonnées. Par exemple :

```
{

   "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
}
```

Vous pouvez également invoquer une fonction Lambda en spécifiant un ARN de fonction directement dans le champ « Ressource ». Lorsque vous appelez une fonction Lambda de cette manière, vous ne pouvez pas la spécifier `.waitForTaskToken` et le résultat de la tâche contient uniquement le résultat de la fonction.

```
{
   "StartAt":"CallFunction",
   "States":{
```

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

Vous pouvez appeler une version ou un alias de fonction Lambda spécifique en spécifiant ces options dans l'ARN du `Resource` champ. Consultez les informations suivantes dans la documentation Lambda :

- Gestion des versions de fonction AWS Lambda
- AWS Lambda alias

Pour plus d'informations sur la façon de configurer IAM les autorisations lors de l'utilisation Step Functions avec d'autres AWS services, consultezPolitiques IAM pour les services intégrés.

Gérez AWS Elemental MediaConvert avec Step Functions

> **Expérimentez avec Step Functions et MediaConvert**
>
> Découvrez comment utiliser l'intégration MediaConvert optimisée dans un flux de travail qui détecte et supprime les barres de couleur SMTPE de longueur inconnue au début d'un clip vidéo. Lisez le billet de blog du 12 avril 2024 : Workflows low code avec AWS Elemental MediaConvert

Step Functions peut contrôler certains AWS services directement depuis Amazon States Language (ASL). Pour en savoir plus, consultez Utilisation avec d'autres services et Transmettre des paramètres à une API de service.

> **En quoi l'intégration optimisée est différente de l'intégration standard du AWS SDK**
>
> - Le modèle Exécuter une tâche (.sync) d'intégration est disponible.
> - Aucune optimisation ni aucun modèle d'Réponse à la requêteAttendre un rappel avec le jeton de tâcheintégration.

MediaConvert API prises en charge :

- `CreateJob`
  - Syntaxe de demande
  - Paramètres pris en charge :
    - `Role` (Obligatoire)
    - `Settings` (Obligatoire)
    - `CreateJobRequest` (facultatif)
  - Syntaxe de réponse — voir CreateJobResponse schéma

Ce qui suit inclut un `Task` État qui soumet une MediaConvert tâche et attend qu'elle soit terminée.

```
{
    "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
        }
    }
}
```

Pour plus d'informations sur la configuration IAM des autorisations lors de l'utilisation Step Functions avec MediaConvert, consultezPolitiques IAM pour AWS Elemental MediaConvert.

> **Les paramètres in Step Functions sont exprimés en PascalCase**
>
> Même si l'API de service native se trouve dans CamelCase, par exemple l'`startSyncExecution`action d'API, vous spécifiez des paramètres PascalCase dans, tels que :. `StateMachineArn`

Gérez SageMaker avec Step Functions

Step Functions peut contrôler certains AWS services directement depuis Amazon States Language (ASL). Pour en savoir plus, consultez Utilisation avec d'autres services et Transmettre des paramètres à une API de service.

> **En quoi l' SageMaker intégration optimisée est différente de l'intégration du SageMaker AWS SDK**
>
> - Le modèle Exécuter une tâche (.sync) d'intégration est pris en charge.
> - Il n'y a aucune optimisation pour le modèle Réponse à la requête d'intégration.
> - Le modèle Attendre un rappel avec le jeton de tâche d'intégration n'est pas pris en charge.

SageMaker API et syntaxe prises en charge :

- `CreateEndpoint`
  - Syntaxe de demande
  - Paramètres pris en charge :
    - `EndpointConfigName`
    - `EndpointName`
    - `Tags`
  - Syntaxe de réponse
- `CreateEndpointConfig`
  - Syntaxe de demande
  - Paramètres pris en charge :
    - `EndpointConfigName`
    - `KmsKeyId`

    - ProductionVariants
    - Tags
  - Syntaxe de réponse
- CreateHyperParameterTuningJob

> **Note**
>
> Cette action d'API prend en charge le modèle .sync d'intégration.

  - Syntaxe de demande
  - Paramètres pris en charge :
    - HyperParameterTuningJobConfig
    - HyperParameterTuningJobName
    - Tags
    - TrainingJobDefinition
    - WarmStartConfig
  - Syntaxe de réponse
- CreateLabelingJob

> **Note**
>
> Cette action d'API prend en charge le modèle .sync d'intégration.

  - Syntaxe de demande
  - Paramètres pris en charge :
    - HumanTaskConfig
    - InputConfig
    - LabelAttributeName
    - LabelCategoryConfigS3Uri
    - LabelingJobAlgorithmsConfig
    - LabelingJobName

    - OutputConfig
    - RoleArn
    - StoppingConditions
    - Tags
  - Syntaxe de réponse
- CreateModel
  - Syntaxe de demande
  - Paramètres pris en charge :
    - Containers
    - EnableNetworkIsolation
    - ExecutionRoleArn
    - ModelName
    - PrimaryContainer
    - Tags
    - VpcConfig
- CreateProcessingJob

> **Note**
>
> Cette action d'API prend en charge le modèle .sync d'intégration.

  - Syntaxe de demande
  - Paramètres pris en charge :
    - AppSpecification
    - Environment
    - ExperimentConfig
    - NetworkConfig
    - ProcessingInputs
    - ProcessingJobName
    - ProcessingOutputConfig
    - ProcessingResources

    - RoleArn
    - StoppingCondition
    - Tags
  - Syntaxe de réponse
- CreateTrainingJob

> **Note**
>
> Cette action d'API prend en charge le modèle .sync d'intégration.

  - Syntaxe de demande
  - Paramètres pris en charge :
    - AlgorithmSpecification
    - HyperParameters
    - InputDataConfig
    - OutputDataConfig
    - ResourceConfig
    - RoleArn
    - StoppingCondition
    - Tags
    - TrainingJobName
    - VpcConfig
  - Syntaxe de réponse
- CreateTransformJob

> **Note**
>
> Cette action d'API prend en charge le modèle .sync d'intégration.

> **Note**
>
> AWS Step Functions ne créera pas automatiquement de politique pour`CreateTransformJob`. Vous devez associer une stratégie en ligne au rôle créé. Pour plus d'informations, consultez cet exemple de politique IAM :`CreateTrainingJob`.

- Syntaxe de demande
- Paramètres pris en charge :
  - `BatchStrategy`
  - `Environment`
  - `MaxConcurrentTransforms`
  - `MaxPayloadInMB`
  - `ModelName`
  - `Tags`
  - `TransformInput`
  - `TransformJobName`
  - `TransformOutput`
  - `TransformResources`
- Syntaxe de réponse

- `UpdateEndpoint`
  - Syntaxe de demande
  - Paramètres pris en charge :
    - `EndpointConfigName`
    - `EndpointName`
  - Syntaxe de réponse

SageMaker Exemple de Transform Job

Ce qui suit inclut un `Task` état qui crée une tâche de SageMaker transformation Amazon, spécifiant l'emplacement Amazon S3 pour `DataSource` et`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 Exemple de job de formation

Ce qui suit inclut un `Task` état qui crée une tâche de SageMaker formation Amazon.

```
{
   "SageMaker CreateTrainingJob":{
      "Type":"Task",
      "Resource":"arn:aws:states:::sagemaker:createTrainingJob.sync",
      "Parameters":{
         "TrainingJobName":"search-model",
         "ResourceConfig":{
            "InstanceCount":4,
            "InstanceType":"ml.c4.8xlarge",
            "VolumeSizeInGB":20
         },
```

```
         "HyperParameters":{
            "mode":"batch_skipgram",
            "epochs":"5",
            "min_count":"5",
            "sampling_threshold":"0.0001",
            "learning_rate":"0.025",
            "window_size":"5",
            "vector_dim":"300",
            "negative_samples":"5",
            "batch_size":"11"
         },
         "AlgorithmSpecification":{
            "TrainingImage":"...",
            "TrainingInputMode":"File"
         },
         "OutputDataConfig":{
            "S3OutputPath":"s3://bucket-name/doc-search/model"
         },
         "StoppingCondition":{
            "MaxRuntimeInSeconds":100000
         },
         "RoleArn":"arn:aws:iam::123456789012:role/docsearch-stepfunction-iam-role",
         "InputDataConfig":[
            {
               "ChannelName":"train",
               "DataSource":{
                  "S3DataSource":{
                     "S3DataType":"S3Prefix",
                     "S3Uri":"s3://bucket-name/doc-search/interim-data/training-data/",
                     "S3DataDistributionType":"FullyReplicated"
                  }
               }
            }
         ]
      },
      "Retry":[
         {
            "ErrorEquals":[
               "SageMaker.AmazonSageMakerException"
            ],
            "IntervalSeconds":1,
            "MaxAttempts":100,
            "BackoffRate":1.1
         },
```

```
         {
            "ErrorEquals":[
               "SageMaker.ResourceLimitExceededException"
            ],
            "IntervalSeconds":60,
            "MaxAttempts":5000,
            "BackoffRate":1
         },
         {
            "ErrorEquals":[
               "States.Timeout"
            ],
            "IntervalSeconds":1,
            "MaxAttempts":5,
            "BackoffRate":1
         }
      ],
      "Catch":[
         {
            "ErrorEquals":[
               "States.ALL"
            ],
            "ResultPath":"$.cause",
            "Next":"Sagemaker Training Job Error"
         }
      ],
      "Next":"Delete Interim Data Job"
   }
}
```

SageMaker Exemple de job d'étiquetage

Ce qui suit inclut un `Task` état qui crée une tâche SageMaker d'étiquetage Amazon.

```
{
  "StartAt": "SageMaker CreateLabelingJob",
  "TimeoutSeconds": 3600,
  "States": {
    "SageMaker CreateLabelingJob": {
      "Type": "Task",
      "Resource": "arn:aws:states:::sagemaker:createLabelingJob.sync",
```

```
      "Parameters": {
        "HumanTaskConfig": {
          "AnnotationConsolidationConfig": {
            "AnnotationConsolidationLambdaArn": "arn:aws:lambda:us-
west-2:123456789012:function:ACS-TextMultiClass"
          },
          "NumberOfHumanWorkersPerDataObject": 1,
          "PreHumanTaskLambdaArn": "arn:aws:lambda:us-west-2:123456789012:function:PRE-
TextMultiClass",
          "TaskDescription": "Classify the following text",
          "TaskKeywords": [
            "tc",
            "Labeling"
          ],
          "TaskTimeLimitInSeconds": 300,
          "TaskTitle": "Classify short bits of text",
          "UiConfig": {
            "UiTemplateS3Uri": "s3://s3bucket-example/TextClassification.template"
          },
          "WorkteamArn": "arn:aws:sagemaker:us-west-2:123456789012:workteam/private-
crowd/ExampleTesting"
        },
        "InputConfig": {
          "DataAttributes": {
            "ContentClassifiers": [
              "FreeOfPersonallyIdentifiableInformation",
              "FreeOfAdultContent"
            ]
          },
          "DataSource": {
            "S3DataSource": {
              "ManifestS3Uri": "s3://s3bucket-example/manifest.json"
            }
          }
        },
        "LabelAttributeName": "Categories",
        "LabelCategoryConfigS3Uri": "s3://s3bucket-example/labelcategories.json",
        "LabelingJobName": "example-job-name",
        "OutputConfig": {
          "S3OutputPath": "s3://s3bucket-example/output"
        },
        "RoleArn": "arn:aws:iam::123456789012:role/service-role/AmazonSageMaker-
ExecutionRole",
        "StoppingConditions": {
```

```
          "MaxHumanLabeledObjectCount": 10000,
          "MaxPercentageOfInputDatasetLabeled": 100
        }
      },
      "Next": "ValidateOutput"
    },
    "ValidateOutput": {
      "Type": "Choice",
      "Choices": [
        {
          "Not": {
            "Variable": "$.LabelingJobArn",
            "StringEquals": ""
          },
          "Next": "Succeed"
        }
      ],
      "Default": "Fail"
    },
    "Succeed": {
      "Type": "Succeed"
    },
    "Fail": {
      "Type": "Fail",
      "Error": "InvalidOutput",
      "Cause": "Output is not what was expected. This could be due to a service outage
 or a misconfigured service integration."
    }
  }
}
```

SageMaker Exemple de job de traitement

Ce qui suit inclut un `Task` état qui crée une tâche SageMaker de traitement Amazon.

```
{
  "StartAt": "SageMaker CreateProcessingJob Sync",
  "TimeoutSeconds": 3600,
  "States": {
    "SageMaker CreateProcessingJob Sync": {
      "Type": "Task",
      "Resource": "arn:aws:states:::sagemaker:createProcessingJob.sync",
      "Parameters": {
```

```
        "AppSpecification": {
          "ImageUri": "737474898029.dkr.ecr.sa-east-1.amazonaws.com/sagemaker-scikit-
learn:0.20.0-cpu-py3"
        },
        "ProcessingResources": {
          "ClusterConfig": {
            "InstanceCount": 1,
            "InstanceType": "ml.t3.medium",
            "VolumeSizeInGB": 10
          }
        },
        "RoleArn": "arn:aws:iam::123456789012:role/SM-003-
CreateProcessingJobAPIExecutionRole",
        "ProcessingJobName.$": "$.id"
      },
      "Next": "ValidateOutput"
    },
    "ValidateOutput": {
      "Type": "Choice",
      "Choices": [
        {
          "Not": {
            "Variable": "$.ProcessingJobArn",
            "StringEquals": ""
          },
          "Next": "Succeed"
        }
      ],
      "Default": "Fail"
    },
    "Succeed": {
      "Type": "Succeed"
    },
    "Fail": {
      "Type": "Fail",
      "Error": "InvalidConnectorOutput",
      "Cause": "Connector output is not what was expected. This could be due to a
 service outage or a misconfigured connector."
    }
  }
}
```

Pour plus d'informations sur la configuration IAM des autorisations lors de l'utilisation Step Functions avec d'autres AWS services, consultezPolitiques IAM pour les services intégrés.

Appelez Amazon SNS avec Step Functions

Step Functions peut contrôler certains AWS services directement depuis Amazon States Language (ASL). Pour en savoir plus, consultez Utilisation avec d'autres services et Transmettre des paramètres à une API de service.

> En quoi l'intégration optimisée d'Amazon SNS est différente de l'intégration du SDK Amazon AWS SNS
>
> Il n'y a aucune optimisation pour les modèles Réponse à la requête d'Attendre un rappel avec le jeton de tâcheintégration.

API Amazon SNS prises en charge :

> Note
>
> Il existe un quota pour la taille maximale des données d'entrée ou de résultat pour une tâche dans Step Functions. Cela vous limite à 256 Ko de données sous forme de chaîne codée en UTF-8 lorsque vous envoyez ou recevez des données d'un autre service. veuillez consulter Quotas liés aux exécutions par les machines de l'État.

- `Publish`
  - Syntaxe de demande
  - Paramètres pris en charge
    - `Message`
    - `MessageAttributes`
    - `MessageStructure`
    - `PhoneNumber`
    - `Subject`
    - `TargetArn`
    - `TopicArn`

- Syntaxe de réponse

> **Les paramètres in Step Functions sont exprimés en PascalCase**
>
> Même si l'API de service native se trouve dans CamelCase, par exemple l'`startSyncExecution`action d'API, vous spécifiez des paramètres PascalCase dans, tels que :. `StateMachineArn`

Ce qui suit inclut un `Task` état qui publie sur une rubrique Amazon Simple Notification Service (Amazon SNS).

```
{
 "StartAt": "Publish to SNS",
 "States": {
   "Publish to SNS": {
     "Type": "Task",
     "Resource": "arn:aws:states:::sns:publish",
     "Parameters": {
       "TopicArn": "arn:aws:sns:us-east-1:123456789012:myTopic",
       "Message.$": "$.input.message",
       "MessageAttributes": {
         "my_attribute_no_1": {
           "DataType": "String",
           "StringValue": "value of my_attribute_no_1"
         },
         "my_attribute_no_2": {
           "DataType": "String",
           "StringValue": "value of my_attribute_no_2"
         }
       }
     },
     "End": true
    }
  }
}
```

Transmission de valeurs dynamiques. Vous pouvez modifier l'exemple ci-dessus pour transmettre dynamiquement un attribut à partir de cette charge utile JSON :

```
{
"input": {
  "message": "Hello world"
},
"SNSDetails": {
  "attribute1": "some value",
  "attribute2": "some other value",
}
}
```

Ajoutez le `.$` au `StringValue` champ :

```
"MessageAttributes": {
  "my_attribute_no_1": {
      "DataType": "String",
      "StringValue.$": "$.SNSDetails.attribute1"
  },
  "my_attribute_no_2": {
      "DataType": "String",
      "StringValue.$": "$.SNSDetails.attribute2"
  }
```

Ce qui suit inclut un `Task` état qui publie sur une rubrique Amazon SNS, puis attend que le jeton de tâche soit renvoyé. veuillez consulter Attendre un rappel avec le jeton de tâche.

```
{
   "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
      }
   }
}
```

Pour plus d'informations sur la configuration IAM des autorisations lors de l'utilisation Step Functions avec d'autres AWS services, consultezPolitiques IAM pour les services intégrés.

Appelez Amazon SQS avec Step Functions

Step Functions peut contrôler certains AWS services directement depuis Amazon States Language (ASL). Pour en savoir plus, consultez Utilisation avec d'autres services et Transmettre des paramètres à une API de service.

> En quoi l'intégration optimisée d'Amazon SQS est différente de l'intégration du SDK Amazon AWS SQS
>
> Il n'y a aucune optimisation pour les modèles Réponse à la requête d'Attendre un rappel avec le jeton de tâcheintégration.

API Amazon SQS prises en charge :

> Note
>
> Il existe un quota pour la taille maximale des données d'entrée ou de résultat pour une tâche dans Step Functions. Cela vous limite à 256 Ko de données sous forme de chaîne codée en UTF-8 lorsque vous envoyez ou recevez des données d'un autre service. veuillez consulter Quotas liés aux exécutions par les machines de l'État.

- `SendMessage`

  Paramètres pris en charge :

  - `DelaySeconds`
  - `MessageAttribute`
  - `MessageBody`
  - `MessageDeduplicationId`
  - `MessageGroupId`
  - `QueueUrl`
- `Response syntax`

> **Les paramètres in Step Functions sont exprimés en PascalCase**
>
> Même si l'API de service native se trouve dans CamelCase, par exemple l'`startSyncExecution`action d'API, vous spécifiez des paramètres PascalCase dans, tels que :. `StateMachineArn`

Ce qui suit inclut un `Task` état qui envoie un message Amazon Simple Queue Service (Amazon SQS).

```
{
 "StartAt": "Send to SQS",
 "States": {
   "Send to SQS": {
     "Type": "Task",
     "Resource": "arn:aws:states:::sqs:sendMessage",
     "Parameters": {
       "QueueUrl": "https://sqs.us-east-1.amazonaws.com/123456789012/myQueue",
       "MessageBody.$": "$.input.message",
       "MessageAttributes": {
         "my_attribute_no_1": {
           "DataType": "String",
           "StringValue": "attribute1"
         },
         "my_attribute_no_2": {
           "DataType": "String",
           "StringValue": "attribute2"
         }
       }
     },
     "End": true
    }
  }
}
```

Ce qui suit inclut un `Task` état qui publie dans une file d'attente Amazon SQS, puis attend que le jeton de tâche soit renvoyé. veuillez consulter Attendre un rappel avec le jeton de tâche.

```
{
   "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
      }
   }
}
```

Pour en savoir plus sur la réception de messages dans Amazon SQS, consultez la section Recevoir et supprimer votre message dans le manuel Amazon Simple Queue Service Developer Guide.

Pour plus d'informations sur la façon de configurer IAM les autorisations lors de l'utilisation Step Functions avec d'autres AWS services, consultezPolitiques IAM pour les services intégrés.

Gérez AWS Step Functions les exécutions en tant que service intégré

Step Functions intègre sa propre API en tant que service d'intégration. Cela permet à Step Functions de démarrer une nouvelle exécution d'une machine d'état directement à partir de l'état de la tâche d'une exécution en cours. Utilisez des exécutions de workflows imbriqués pour réduire la complexité de vos principaux workflows et réutiliser les processus courants lors de la création de nouveaux workflows.

En quoi l'intégration d'Optimized Step Functions est différente de l'intégration du AWS SDK Step Functions

- Le modèle Exécuter une tâche (.sync) d'intégration est disponible.

  Notez qu'il n'y a aucune optimisation pour les modèles d'Attendre un rappel avec le jeton de tâcheintégration Réponse à la requête ou.

Pour plus d’informations, consultez les ressources suivantes :

- Démarrer les exécutions à partir d'une tâche
- Utilisation avec d'autres services
- Transmettre des paramètres à une API de service

API et syntaxe Step Functions prises en charge :

- `StartExecution`
  - Syntaxe de la demande
  - Paramètres pris en charge
    - `Input`
    - `Name`
    - `StateMachineArn`
  - Syntaxe de réponse

L'exemple suivant inclut un état `Task` qui démarre l'exécution d'une autre machine d'état et attend qu'elle se termine.

```
{
   "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
}
```

L'exemple suivant inclut un état `Task` qui démarre l'exécution d'une autre machine d'état.

```
{
   "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
}
```

L'exemple suivant inclut un état `Task` qui implémente le modèle d'intégration de service de rappel.

```
{
   "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
}
```

Pour associer une exécution de workflow imbriqué à l'exécution parent qui l'a démarrée, transmettez un paramètre spécialement nommé qui inclut l'ID d'exécution extrait de l'objet de contexte. Lorsque vous démarrez une exécution imbriquée, utilisez un paramètre nommé `AWS_STEP_FUNCTIONS_STARTED_BY_EXECUTION_ID`. Transmettez l'ID d'exécution en ajoutant `.$` au nom du paramètre et en référençant l'ID dans l'objet de contexte avec `$$.Execution.Id`. Pour plus d'informations, consultez Accès à l'objet de contexte.

```
{
   "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
}
```

Les machines d'état imbriquées renvoient les éléments suivants :

Ressource	Sortie
startExecution.sync	Chaîne
startExecution.sync:2	JSON

Ces deux éléments attendent que la machine d'état imbriquée se termine, mais renvoient des formats `Output` différents. Par exemple, si vous créez une fonction Lambda qui renvoie l'objet`{ "MyKey": "MyValue" }`, vous obtiendrez les réponses suivantes :

Pour startExecution.sync :

```
{
   <other fields>
   "Output": "{ \"MyKey\": \"MyValue\" }"
}
```

Pour startExecution.sync:2 :

```
{
   <other fields>
   "Output": {
      "MyKey": "MyValue"
   }
}
```

Configuration des autorisations IAM pour les machines à états imbriqués

Une machine d'état parent détermine si une machine d'état enfant a terminé son exécution à l'aide d'interrogations et d'événements. Le sondage nécessite une autorisation,

`states:DescribeExecution` tandis que les événements envoyés EventBridge à Step Functions nécessitent des autorisations pour `events:PutTargetsevents:PutRule`, et`events:DescribeRule`. Si ces autorisations sont absentes de votre rôle IAM, un délai peut s'écouler avant qu'une machine d'état parent ne soit informée de la fin de l'exécution de la machine d'état enfant.

Pour une machine à états nécessitant l'exécution d'un seul flux de travail imbriqué, utilisez une politique IAM qui limite les autorisations à cette machine à états. `StartExecution`

Pour plus d'informations, consultez la section Autorisations IAM pour Step Functions.

Appelez des API tierces

Une tâche HTTP est un type d'État de la tâcheétat qui vous permet d'appeler n'importe quelle API publique tierce, telle que Salesforce et Stripe, dans vos flux de travail. Pour appeler une API tierce, utilisez l'état de la tâche avec la `arn:aws:states:::http:invoke` ressource. Fournissez ensuite les détails de configuration du point de terminaison de l'API, tels que l'URL de l'API, la méthode que vous souhaitez utiliser et les détails d'authentification.

Si vous utilisez Workflow Studio pour créer votre machine d'état contenant une tâche HTTP, Workflow Studio génère automatiquement un rôle d'exécution avec des IAM politiques pour la tâche HTTP. Pour plus d’informations, consultez Rôle pour tester les tâches HTTP dans Workflow Studio.

Rubriques

- Définition de tâche HTTP
- Champs de tâches HTTP
- Authentification pour une tâche HTTP
- Fusion des données de définition de EventBridge connexion et de tâche HTTP
- Appliquer le codage d'URL sur le corps de la demande
- Autorisations IAM pour exécuter une tâche HTTP
- Exemple de tâche HTTP
- Test d'une tâche HTTP
- Réponses aux tâches HTTP non prises en charge

Définition de tâche HTTP

La définition ASL représente une tâche HTTP avec `http:invoke` ressource. La définition de tâche HTTP suivante invoque une API Stripe qui renvoie une liste de tous les clients.

```
"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
}
```

Champs de tâches HTTP

Une tâche HTTP inclut les champs suivants dans sa définition.

`Resource` (Obligatoire)

Pour spécifier un type de tâche, indiquez son ARN dans le `Resource` champ. Pour une tâche HTTP, vous devez spécifier le `Resource` champ comme suit.

```
"Resource": "arn:aws:states:::http:invoke"
```

`Parameters` (Obligatoire)

Contient les `ConnectionArn` champs `ApiEndpoint``Method`, et qui fournissent des informations sur l'API tierce que vous souhaitez appeler. `Parameters`contient également des champs facultatifs, tels que `Headers` et`QueryParameters`.

Vous pouvez spécifier une combinaison de JSON statique et de JsonPathsyntaxe comme `Parameters` dans le `Parameters` champ. Pour plus d'informations, consultez Transmettre des paramètres à une API de service.

`ApiEndpoint`(Obligatoire)

Spécifie l'URL de l'API tierce que vous souhaitez appeler. Pour ajouter des paramètres de requête à l'URL, utilisez le champ. `QueryParameters` L'exemple suivant montre comment vous pouvez appeler une API Stripe pour récupérer la liste de tous les clients.

```
"ApiEndpoint":"https://api.stripe.com/v1/customers"
```

Vous pouvez également spécifier un chemin de référence à l'aide de la JsonPathsyntaxe permettant de sélectionner le nœud JSON contenant l'URL de l'API tierce. Supposons, par exemple, que vous souhaitiez appeler l'une des API de Stripe à l'aide d'un identifiant client spécifique. Imaginez que vous avez fourni l'entrée d'état suivante.

```
{
    "customer_id": "1234567890",
    "name": "John Doe"
}
```

Pour récupérer les détails de cet identifiant client à l'aide d'une API Stripe, spécifiez le `ApiEndpoint` comme indiqué dans l'exemple suivant. Cet exemple utilise une fonction intrinsèque et un chemin de référence.

```
"ApiEndpoint.$":"States.Format('https://api.stripe.com/v1/customers/{}',
 $.customer_id)"
```

Au moment de l'exécution, Step Functions résout la valeur de la manière `ApiEndpoint` suivante.

```
https://api.stripe.com/v1/customers/1234567890
```

`Method`(Obligatoire)

Spécifie la méthode HTTP que vous souhaitez utiliser pour appeler une API tierce. Vous pouvez spécifier l'une des méthodes suivantes dans votre tâche HTTP : GET, POST, PUT, DELETE, PATCH, OPTIONS ou HEAD.

Par exemple, pour utiliser la méthode GET, spécifiez le `Method` champ comme suit.

```
"Method": "GET"
```

Vous pouvez également utiliser un chemin de référence pour spécifier la méthode lors de l'exécution. Par exemple, **`"Method.$": "$.myHTTPMethod"`**.

`Authentication`(Obligatoire)

Contient le `ConnectionArn` champ qui indique comment authentifier un appel d'API tiers. Step Functionsprend en charge l'authentification pour un élément spécifié `ApiEndpoint` à l'aide de la ressource de connexion deAmazon EventBridge.

`ConnectionArn`(Obligatoire)

Spécifie l'ARN de EventBridge connexion.

Une tâche HTTP nécessite une EventBridge connexion qui gère de manière sécurisée les informations d'authentification d'un fournisseur d'API. Une connexion indique le type d'autorisation et les informations d'identification à utiliser pour autoriser une API tierce. L'utilisation d'une connexion vous permet d'éviter de coder en dur des secrets, tels que des clés d'API, dans la définition de votre machine à états. Dans une connexion, vous pouvez également spécifier `HeadersQueryParameters`, et des `RequestBody` paramètres.

Lorsque vous créez une EventBridge connexion, vous fournissez vos informations d'authentification. Pour plus d'informations sur le fonctionnement de l'authentification pour une tâche HTTP, consultezAuthentification pour une tâche HTTP.

L'exemple suivant montre comment vous pouvez spécifier le `Authentication` champ dans votre définition de tâche HTTP.

```
"Authentication": {
  "ConnectionArn": "arn:aws:events:us-east-2:123456789012:connection/
Stripe/81210c42-8af1-456b-9c4a-6ff02fc664ac"
}
```

`Headers` (facultatif)

Fournit un contexte et des métadonnées supplémentaires au point de terminaison de l'API. Vous pouvez spécifier les en-têtes sous forme de chaîne ou de tableau JSON.

Vous pouvez spécifier des en-têtes dans la EventBridge connexion et dans le `Headers` champ d'une tâche HTTP. Nous vous recommandons de ne pas inclure les informations d'authentification de vos fournisseurs d'API `Headers` sur le terrain. Nous vous recommandons d'inclure ces informations dans votre EventBridge connexion.

Step Functionsajoute les en-têtes que vous spécifiez dans la EventBridge connexion aux en-têtes que vous spécifiez dans la définition de la tâche HTTP. Si les mêmes clés d'en-tête sont présentes dans la définition et la connexion, Step Functions utilise les valeurs correspondantes spécifiées dans la EventBridge connexion pour ces en-têtes. Pour plus d'informations sur le Step Functions mode de fusion des données, consultezFusion des données de définition de EventBridge connexion et de tâche HTTP.

L'exemple suivant spécifie un en-tête qui sera inclus dans un appel d'API tiers :`content-type`.

```
"Headers": {
  "content-type": "application/json"
}
```

Vous pouvez également utiliser un chemin de référence pour spécifier les en-têtes lors de l'exécution. Par exemple, **`"Headers.$": "$.myHTTPHeaders"`**.

Step Functionsdéfinit les `Host` en-têtes `User-Agent``Range`, et. Step Functionsdéfinit la valeur de l'`Host`en-tête en fonction de l'API que vous appelez. Voici un exemple de ces en-têtes.

```
User-Agent: Amazon|StepFunctions|HttpInvoke|us-east-1,
Range: bytes=0-262144,
Host: api.stripe.com
```

Vous ne pouvez pas utiliser les en-têtes suivants dans votre définition de tâche HTTP. Si vous utilisez ces en-têtes, la tâche HTTP échoue avec l'`States.Runtime`erreur.

- A-IM
- Accept-Charset
- Accept-Datetime
- Accept-Encoding
- Cache-Control
- Connexion
- Encodage-Contenu
- Content-MD5
- Date

- Expect
- Forwarded
- De
- Host (Hôte)
- HTTP2-Settings
- If-Match
- If-Modified-Since
- If-None-Match
- If-Range
- If-Unmodified-Since
- Max-Forwards
- Origin
- Pragma
- Proxy-Authorization
- Référent
- Serveur
- TE
- Trailer
- Transfer-Encoding
- Upgrade
- Via
- Avertissement
- X-Forwarded-*
- x-amz-*
- x-amzn-*

`QueryParameters` (facultatif)

Insère des paires clé-valeur à la fin d'une URL d'API. Vous pouvez spécifier les paramètres de requête sous forme de chaîne, de tableau JSON ou d'objet JSON. Step Functionsencode automatiquement les paramètres de requête en URL lorsqu'il appelle une API tierce.

Supposons, par exemple, que vous souhaitiez appeler l'API Stripe pour rechercher des clients qui effectuent leurs transactions en dollars américains (USD). Imaginez que vous avez fourni ce qui suit `QueryParameters` comme entrée d'état.

```
"QueryParameters": {
  "currency": "usd"
}
```

Lors de l'exécution, Step Functions ajoute l'URL `QueryParameters` à l'API comme suit.

```
https://api.stripe.com/v1/customers/search?currency=usd
```

Vous pouvez également utiliser un chemin de référence pour spécifier les paramètres de requête lors de l'exécution. Par exemple, **`"QueryParameters.$": "$.myQueryParameters"`**.

Si vous avez spécifié des paramètres de requête dans votre EventBridge connexion, Step Functions ajoutez ces paramètres de requête aux paramètres de requête que vous spécifiez dans la définition de la tâche HTTP. Si les mêmes clés de paramètres de requête sont présentes dans la définition et la connexion, Step Functions utilise les valeurs correspondantes spécifiées dans la EventBridge connexion pour ces en-têtes. Pour plus d'informations sur le Step Functions mode de fusion des données, consultezFusion des données de définition de EventBridge connexion et de tâche HTTP.

`Transform` (facultatif)

Contient les `RequestEncodingOptions` champs `RequestBodyEncoding` et. Par défaut, Step Functions envoie le corps de la requête sous forme de données JSON à un point de terminaison d'API.

Si votre fournisseur d'API accepte les corps de `form-urlencoded` requête, utilisez le `Transform` champ pour spécifier le codage URL des corps de requête. Vous devez également spécifier l'`content-type`en-tête sous la forme`application/x-www-form-urlencoded`. Step Functionspuis encode automatiquement l'URL du corps de votre demande.

`RequestBodyEncoding`

Spécifie le codage URL du corps de votre demande. Vous pouvez spécifier l'une des valeurs suivantes : `NONE` ou`URL_ENCODED`.

- `NONE`— Le corps de la requête HTTP sera le JSON sérialisé du `RequestBody` champ. C'est la valeur par défaut.
- `URL_ENCODED`— Le corps de la requête HTTP sera constitué des données de formulaire codées en URL du `RequestBody` champ.

`RequestEncodingOptions`

Détermine l'option de codage à utiliser pour les tableaux dans le corps de votre demande si vous définissez sur`RequestBodyEncoding`. `URL_ENCODED`

Step Functionsprend en charge les options de codage de tableau suivantes. Pour plus d'informations sur ces options et leurs exemples, consultezAppliquer le codage d'URL sur le corps de la demande.

- `INDICES`— Encode les tableaux en utilisant la valeur d'index des éléments du tableau. Step FunctionsUtilise cette option de codage par défaut.
- `REPEAT`— Répète une clé pour chaque élément d'un tableau.
- `COMMAS`— Encode toutes les valeurs d'une clé sous forme de liste de valeurs séparées par des virgules.
- `BRACKETS`— Répète une clé pour chaque élément d'un tableau et ajoute un crochet, [], à la clé pour indiquer qu'il s'agit d'un tableau.

L'exemple suivant définit le codage URL pour les données du corps de la demande. Il indique également d'utiliser l'option d'`COMMAS`encodage pour les tableaux dans le corps de la requête.

```
"Transform": {
  "RequestBodyEncoding": "URL_ENCODED",
  "RequestEncodingOptions": {
    "ArrayFormat": "COMMAS"
  }
}
```

`RequestBody` (facultatif)

Accepte les données JSON que vous fournissez dans l'entrée d'état. Dans`RequestBody`, vous pouvez spécifier une combinaison de JSON statique et de JsonPathsyntaxe. Supposons, par exemple, que vous fournissiez l'entrée d'état suivante :

```
{
```

```
    "CardNumber": "1234567890",
    "ExpiryDate": "09/25"
}
```

Pour utiliser ces valeurs de `CardNumber` et `ExpiryDate` dans le corps de votre demande lors de l'exécution, vous pouvez spécifier les données JSON suivantes dans le corps de votre demande.

```
"RequestBody": {
  "Card": {
    "Number.$": "$.CardNumber",
    "Expiry.$": "$.ExpiryDate",
    "Name": "John Doe",
    "Address": "123 Any Street, Any Town, USA"
  }
}
```

Si l'API tierce que vous souhaitez appeler nécessite des corps de `form-urlencoded` requête, vous devez spécifier le codage URL pour les données du corps de votre demande. Pour plus d'informations, consultez Appliquer le codage d'URL sur le corps de la demande.

Authentification pour une tâche HTTP

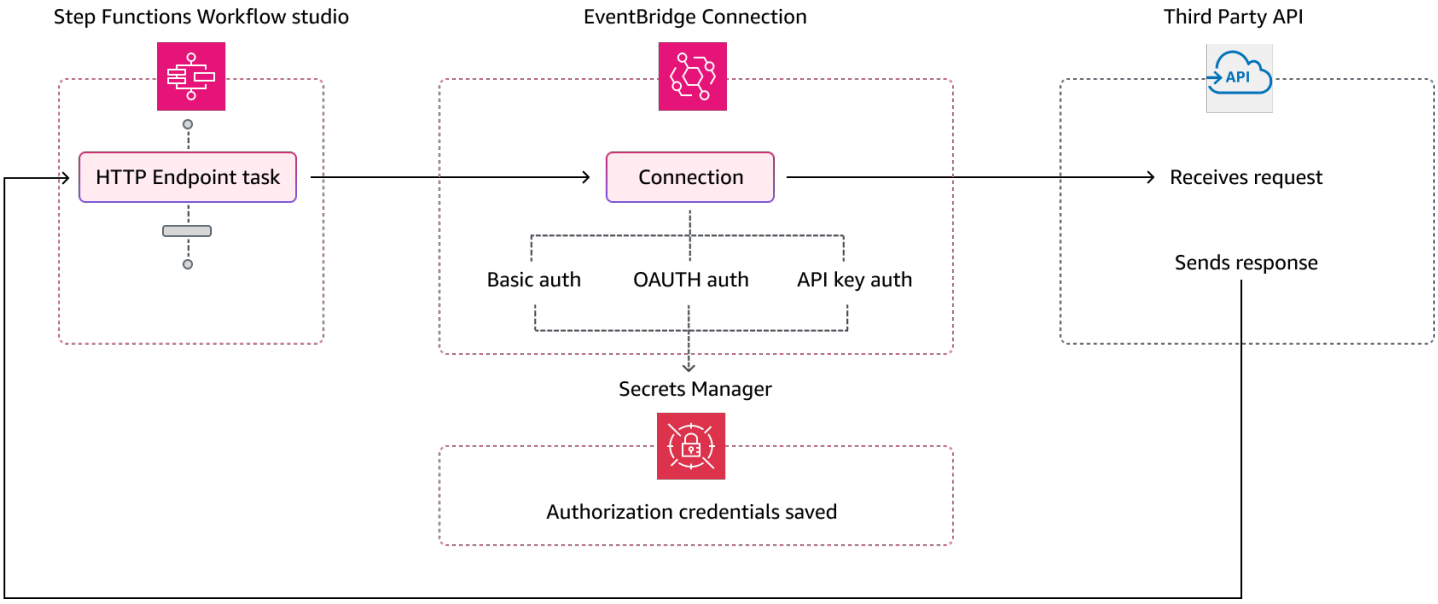
Une tâche HTTP nécessite une EventBridge connexion qui gère de manière sécurisée les informations d'authentification d'un fournisseur d'API. Une connexion indique le type d'autorisation et les informations d'identification à utiliser pour autoriser une API tierce. L'utilisation d'une connexion vous permet d'éviter de coder en dur des secrets, tels que des clés d'API, dans la définition de votre machine à états. Une EventBridge connexion prend en charge les schémas d'autorisation Basic, OAuth et API Key.

Lorsque vous créez une EventBridge connexion, vous fournissez vos informations d'authentification. Vous pouvez également inclure les paramètres d'en-tête, de corps et de requête requis pour l'autorisation auprès d'une API. Vous devez inclure l'ARN de connexion dans toute tâche HTTP qui appelle une API tierce.

Lorsque vous créez une connexion et que vous ajoutez des paramètres d'autorisation, EventBridge crée un secret dans AWS Secrets Manager. Dans ce secret, EventBridge stocke les paramètres de connexion et d'autorisation sous une forme cryptée. Pour créer ou mettre à jour correctement une connexion, vous devez utiliser une Compte AWS personne autorisée à utiliser Secrets Manager. Pour

plus d'informations sur les IAM autorisations dont votre machine d'état a besoin pour accéder à une EventBridge connexion, consultezAutorisations IAM pour exécuter une tâche HTTP.

L'image suivante montre comment Step Functions gère l'authentification pour les appels d'API tiers à l'aide d'une EventBridge connexion.



Fusion des données de définition de EventBridge connexion et de tâche HTTP

Lorsque vous appelez une tâche HTTP, vous pouvez spécifier des données dans votre EventBridge connexion et dans votre définition de tâche HTTP. Ces données incluent `HeadersQueryParameters`, et les `RequestBody` paramètres. Avant d'appeler une API tierce, Step Functions fusionne le corps de la requête avec les paramètres du corps de connexion dans tous les cas, sauf si le corps de votre demande est une chaîne et que les paramètres du corps de connexion ne sont pas vides. Dans ce cas, la tâche HTTP échoue avec l'`States.Runtime`erreur.

Si des clés dupliquées sont spécifiées dans la définition de la tâche HTTP et dans la EventBridge connexionStep Functions, les valeurs de la tâche HTTP sont remplacées par celles de la connexion.

La liste suivante décrit comment Step Functions fusionner les données avant d'appeler une API tierce :

- En-têtes : Step Functions ajoute tous les en-têtes que vous avez spécifiés dans la connexion aux en-têtes du `Headers` champ de la tâche HTTP. En cas de conflit entre les clés d'en-tête, Step Functions utilise les valeurs spécifiées dans la connexion pour ces en-têtes. Par exemple, si vous

avez spécifié l'`content-type`en-tête dans la définition de la tâche HTTP et dans la EventBridge connexion, Step Functions utilise la valeur `content-type` d'en-tête spécifiée dans la connexion.

- Paramètres de requête : Step Functions ajoute tous les paramètres de requête que vous avez spécifiés dans la connexion aux paramètres de requête dans le `QueryParameters` champ de la tâche HTTP. En cas de conflit entre les clés des paramètres de requête, Step Functions utilise les valeurs spécifiées dans la connexion pour ces paramètres de requête. Par exemple, si vous avez spécifié le paramètre de `maxItems` requête à la fois dans la définition de la tâche HTTP et dans la EventBridge connexion, Step Functions utilise la valeur du paramètre de `maxItems` requête spécifiée dans la connexion.
- Paramètres de corps
  - Step Functionsajoute toutes les valeurs du corps de requête spécifiées dans la connexion au corps de la demande dans le `RequestBody` champ de la tâche HTTP. En cas de conflit entre les clés du corps de la demande, Step Functions utilise les valeurs spécifiées dans la connexion pour le corps de la demande. Supposons, par exemple, que vous ayez spécifié un `Mode` champ dans la définition `RequestBody` de la tâche HTTP et dans la EventBridge connexion. Step Functionsutilise la valeur du `Mode` champ que vous avez spécifiée dans la connexion.
  - Si vous spécifiez le corps de la demande sous forme de chaîne au lieu d'un objet JSON et que la EventBridge connexion contient également le corps de la demande, Step Functions vous ne pouvez pas fusionner le corps de demande spécifié à ces deux endroits. La tâche HTTP échoue avec l'`States.Runtime`erreur.

  Step Functionsapplique toutes les transformations et sérialise le corps de la demande une fois la fusion terminée.

L'exemple suivant définit les `RequestBody` champs `HeadersQueryParameters`, et dans la tâche HTTP et dans la EventBridge connexion.

Définition de tâche HTTP

```
{
  "Comment": "Data merging example for HTTP Task and EventBridge connection",
  "StartAt": "ListCustomers",
  "States": {
    "ListCustomers": {
      "Type": "Task",
      "Resource": "arn:aws:states:::http:invoke",
      "Parameters": {
        "Authentication": {
```

```
          "ConnectionArn": "arn:aws:events:us-
east-1:123456789012:connection/Example/81210c42-8af1-456b-9c4a-6ff02fc664ac"
        },
        "ApiEndpoint": "https:/example.com/path",
        "Method": "GET",
        "Headers": {
          "Request-Id": "my_request_id",
          "Header-Param": "state_machine_header_param"
        },
        "RequestBody": {
          "Job": "Software Engineer",
          "Company": "AnyCompany",
          "BodyParam": "state_machine_body_param"
        },
        "QueryParameters": {
          "QueryParam": "state_machine_query_param"
        }
      }
    }
  }
}
```

Connexion EventBridge

```
{
  "AuthorizationType": "API_KEY",
  "AuthParameters": {
    "ApiKeyAuthParameters": {
      "ApiKeyName": "ApiKey",
      "ApiKeyValue": "key_value"
    },
    "InvocationHttpParameters": {
      "BodyParameters": [
        {
          "Key": "BodyParam",
          "Value": "connection_body_param"
        }
      ],
      "HeaderParameters": [
        {
          "Key": "Header-Param",
          "Value": "connection_header_param"
        }
```

```
      ],
      "QueryStringParameters": [
        {
          "Key": "QueryParam",
          "Value": "connection_query_param"
        }
      ]
    }
  }
}
```

Dans cet exemple, les clés dupliquées sont spécifiées dans la tâche et la EventBridge connexion HTTP. Par conséquent, Step Functions remplace les valeurs de la tâche HTTP par celles de la connexion. L'extrait de code suivant montre la requête HTTP envoyée à Step Functions l'API tierce.

```
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"}
```

Appliquer le codage d'URL sur le corps de la demande

Par défaut, Step Functions envoie le corps de la requête sous forme de données JSON à un point de terminaison d'API. Si votre fournisseur d'API tiers a besoin de corps de `form-urlencoded` requête, vous devez spécifier le codage URL pour les corps de requête. Step Functionsencode ensuite automatiquement le corps de la demande en fonction de l'option de codage d'URL que vous avez sélectionnée.

Vous spécifiez le codage des URL à l'aide du `Transform` champ. Ce champ contient le `RequestBodyEncoding` champ qui indique si vous souhaitez ou non appliquer un codage URL aux corps de vos demandes. Lorsque vous spécifiez le `RequestBodyEncoding` champ, Step Functions convertit le corps de votre demande JSON en corps de `form-urlencoded` demande avant d'appeler l'API tierce. Vous devez également spécifier l'`content-type`en-tête, `application/`

`x-www-form-urlencoded` car les API qui acceptent les données codées en URL attendent l'`content-type`en-tête.

Pour coder des tableaux dans le corps de votre requête, Step Functions fournit les options de codage des tableaux suivantes.

- `INDICES`— Répète une clé pour chaque élément d'un tableau et ajoute un crochet, [], à la clé pour indiquer qu'il s'agit d'un tableau. Ce crochet contient l'index de l'élément du tableau. L'ajout de l'index permet de définir l'ordre des éléments du tableau. Step FunctionsUtilise cette option de codage par défaut.

  Par exemple, si le corps de votre requête contient le tableau suivant.

  ```
  {"array": ["a","b","c","d"]}
  ```

  Step Functionscode ce tableau dans la chaîne suivante.

  ```
  array[0]=a&array[1]=b&array[2]=c&array[3]=d
  ```

- `REPEAT`— Répète une clé pour chaque élément d'un tableau.

  Par exemple, si le corps de votre requête contient le tableau suivant.

  ```
  {"array": ["a","b","c","d"]}
  ```

  Step Functionscode ce tableau dans la chaîne suivante.

  ```
  array=a&array=b&array=c&array=d
  ```

- `COMMAS`— Encode toutes les valeurs d'une clé sous forme de liste de valeurs séparées par des virgules.

  Par exemple, si le corps de votre requête contient le tableau suivant.

  ```
  {"array": ["a","b","c","d"]}
  ```

  Step Functionscode ce tableau dans la chaîne suivante.

  ```
  array=a,b,c,d
  ```

- `BRACKETS`— Répète une clé pour chaque élément d'un tableau et ajoute un crochet, [], à la clé pour indiquer qu'il s'agit d'un tableau.

  Par exemple, si le corps de votre requête contient le tableau suivant.

  ```
  {"array": ["a","b","c","d"]}
  ```

  Step Functionscode ce tableau dans la chaîne suivante.

  ```
  array[]=a&array[]=b&array[]=c&array[]=d
  ```

Autorisations IAM pour exécuter une tâche HTTP

Votre rôle d'exécution de machine à états doit disposer des `secretsmanager:DescribeSecret` autorisations `states:InvokeHTTPEndpoint` `events:RetrieveConnectionCredentialssecretsmanager:GetSecretValue`,, et pour qu'une tâche HTTP appelle une API tierce. L'exemple de politique IAM suivant accorde le minimum de privilèges requis à votre rôle de machine d'état pour appeler les API Stripe. Cette politique IAM autorise également le rôle de machine d'état à accéder à une EventBridge connexion spécifique, y compris le secret de cette connexion qui est stocké dans Secrets Manager.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Statement1",
            "Effect": "Allow",
            "Action": "states:InvokeHTTPEndpoint",
            "Resource": "arn:aws:states:us-
east-2:123456789012:stateMachine:myStateMachine",
            "Condition": {
                "StringEquals": {
                    "states:HTTPMethod": "GET"
                },
                "StringLike": {
                    "states:HTTPEndpoint": "https://api.stripe.com/*"
                }
            }
        },
        {
```

```
            "Sid": "Statement2",
            "Effect": "Allow",
            "Action": [
                "events:RetrieveConnectionCredentials",
            ],
            "Resource": "arn:aws:events:us-
east-2:123456789012:connection/oauth_connection/aeabd89e-d39c-4181-9486-9fe03e6f286a"
        },
        {
            "Sid": "Statement3",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue",
                "secretsmanager:DescribeSecret"
            ],
            "Resource": "arn:aws:secretsmanager:*:*:secret:events!connection/*"
        }
    ]
}
```

Exemple de tâche HTTP

La définition de machine à états suivante montre une tâche HTTP qui inclut les `RequestBody` paramètres `Headers` `QueryParametersTransform`,, et. La tâche HTTP appelle une API Stripe, https://api.stripe.com/v1/invoices, pour générer une facture. La tâche HTTP spécifie également le codage URL pour le corps de la requête à l'aide de l'option d'`INDICES`encodage.

Assurez-vous d'avoir créé une EventBridge connexion. L'exemple suivant montre une connexion créée à l'aide du type d'authentification BASIC.

```
{
    "Type": "BASIC",
    "AuthParameters": {
        "BasicAuthParameters": {
            "Password": "myPassword",
            "Username": "myUsername"
         },
    }
}
```

N'oubliez pas de remplacer le texte *en italique* par les informations spécifiques à votre ressource.

```
{
  "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
    }
  }
}
```

Pour exécuter cette machine d'état, entrez l'ID client comme entrée, comme indiqué dans l'exemple suivant :

```
{
    "customer_id": "1234567890"
}
```

L'exemple suivant montre la requête HTTP envoyée Step Functions à l'API Stripe.

```
POST /v1/invoices HTTP/1.1
Authorization: Basic <base64 of username and password>
Content-Type: application/x-www-form-urlencoded
Host: api.stripe.com
Range: bytes=0-262144
Transfer-Encoding: chunked
User-Agent: Amazon|StepFunctions|HttpInvoke|us-east-1

description=Monthly%20subscription&metadata%5Border_details%5D=monthly%20report
%20data&customer=1234567890
```

Test d'une tâche HTTP

Vous pouvez utiliser l'TestStateAPI via la console, le SDK ou le AWS CLI pour tester une tâche HTTP. La procédure suivante décrit comment utiliser l' TestState API dans la Step Functions console. Vous pouvez tester de manière itérative les détails de la demande, de la réponse et de l'authentification de l'API jusqu'à ce que votre tâche HTTP fonctionne comme prévu.

Tester l'état d'une tâche HTTP dans Step Functions la console

1. Ouvrez la console Step Functions.
2. Choisissez Créer une machine à états pour commencer à créer une machine à états ou choisissez une machine à états existante contenant une tâche HTTP.

   Reportez-vous à l'étape 4 si vous testez la tâche sur une machine à états existante.
3. Dans Workflow Studio, configurez visuellement une tâche HTTP. Mode de conception Vous pouvez également choisir le mode Code pour copier-coller la définition de la machine à états depuis votre environnement de développement local.
4. En mode Design, choisissez Test state dans le Inspector panneau de Workflow Studio.
5. Dans la boîte de dialogue État du test, procédez comme suit :

   a. Pour Rôle d'exécution, choisissez un rôle d'exécution pour tester l'état. Si vous ne disposez pas d'un rôle doté d'autorisations suffisantes pour une tâche HTTP, consultez la section Rôle pour tester les tâches HTTP dans Workflow Studio pour créer un rôle.

   b. (Facultatif) Fournissez toute entrée JSON dont l'état sélectionné a besoin pour le test.

   c. Pour le niveau d'inspection, conservez la sélection par défaut INFO. Ce niveau indique le statut de l'appel d'API et l'état de sortie. Cela est utile pour vérifier rapidement la réponse de l'API.

   d. Choisissez Démarrer le test.

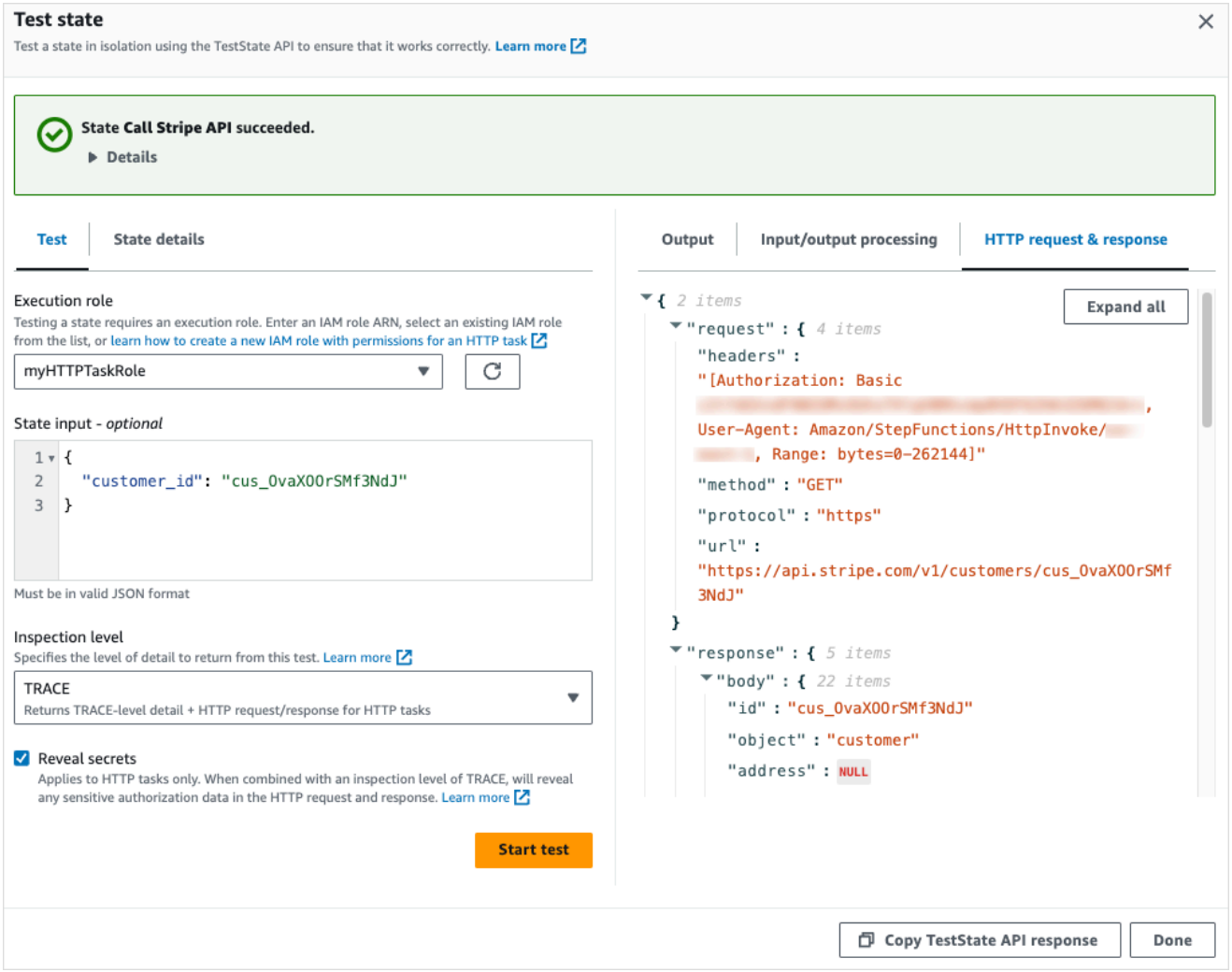
   e. Si le test réussit, la sortie d'état apparaît sur le côté droit de la boîte de dialogue État du test. Si le test échoue, une erreur apparaît.

      Dans l'onglet Détails de l'état de la boîte de dialogue, vous pouvez voir la définition de l'état et un lien vers votre EventBridgeconnexion.

   f. Changez le niveau d'inspection en TRACE. Ce niveau affiche la requête et la réponse HTTP brutes et est utile pour vérifier les en-têtes, les paramètres de requête et d'autres détails spécifiques à l'API.

g. Cochez la case Révéler les secrets. Associé à TRACE, ce paramètre vous permet de voir les données sensibles insérées par la EventBridge connexion, telles que les clés d'API. L'identité IAM utilisateur que vous utilisez pour accéder à la console doit être autorisée à effectuer l'`states:RevealSecrets`action. Sans cette autorisation, Step Functions génère une erreur de refus d'accès lorsque vous démarrez le test. Pour un exemple de IAM politique définissant l'`states:RevealSecrets`autorisation, consultezIAMautorisations d'utilisation de l' TestState API.

   L'image suivante montre un test de réussite d'une tâche HTTP. Le niveau d'inspection pour cet état est défini sur TRACE. L'onglet Requête et réponse HTTP de l'image suivante montre le résultat de l'appel d'API tiers.



h. Choisissez Démarrer le test.

i. Si le test réussit, vous pouvez voir les détails de votre protocole HTTP sous l'onglet Requête et réponse HTTP.

Réponses aux tâches HTTP non prises en charge

Une tâche HTTP échoue avec l'`States.Runtime`erreur si l'une des conditions suivantes est vraie pour la réponse renvoyée :

- La réponse contient un en-tête de type contenu`application/octet-stream`, `image/*``video/*`, ou. `audio/*`
- La réponse ne peut pas être lue comme une chaîne valide. Par exemple, des données binaires ou d'image.

Modèles d'intégration des services

AWS Step Functions s'intègre aux services directement dans l'Amazon States Language. Vous pouvez contrôler ces services AWS à l'aide de trois modèles d'intégration de service.

- Appelez un service et laissez Step Functions passer à l'état suivant immédiatement après avoir reçu une réponse HTTP.
- Appelez un service et demandez à Step Functions d'attendre qu'une tâche soit terminée.
- Appelez un service avec un jeton de tâche et demandez à Step Functions d'attendre que ce jeton soit renvoyé avec une charge utile.

Chacun de ces modèles d'intégration de service est contrôlé par la façon dont vous créez un URI dans le champ `"Resource"` de votre définition de tâche.

Méthodes pour appeler un service intégré

- Réponse à la requête
- Exécuter une tâche (.sync)
- Attendre un rappel avec le jeton de tâche

Pour plus d'informations sur la configuration AWS Identity and Access Management (IAM) des services intégrés, consultezPolitiques IAM pour les services intégrés.

Réponse à la requête

Lorsque vous spécifiez un service dans la `"Resource"` chaîne de l'état de votre tâche et que vous ne fournissez que la ressource, Step Functions attend une réponse HTTP, puis passe à l'état suivant. Step Functions n'attendra pas qu'une tâche soit terminée.

L'exemple suivant montre comment publier une rubrique Amazon SNS.

```
"Send message to SNS":{
   "Type":"Task",
   "Resource":"arn:aws:states:::sns:publish",
   "Parameters":{
      "TopicArn":"arn:aws:sns:us-east-1:123456789012:myTopic",
      "Message":"Hello from Step Functions!"
   },
   "Next":"NEXT_STATE"
}
```

Cet exemple fait référence à l'API de publication d'Amazon SNS. Le flux de travail progresse à l'état suivant après l'appel de l'API `Publish`.

> **Tip**
>
> Pour déployer un exemple de flux de travail utilisant le modèle d'intégration du service Request Response sur votre Compte AWS ordinateur, consultez le Module 2 - Demande de réponse de The AWS Step Functions Workshop.

Exécuter une tâche (.sync)

Pour les services intégrés tels qu' AWS Batch Amazon ECS, Step Functions peut attendre qu'une demande soit terminée avant de passer à l'état suivant. Pour que Step Functions attende, spécifiez le `"Resource"` champ dans la définition de l'état de votre tâche avec le `.sync` suffixe ajouté après l'URI de la ressource.

Par exemple, lorsque vous soumettez une AWS Batch tâche, utilisez le `"Resource"` champ dans la définition de la machine à états, comme indiqué dans cet exemple.

```
"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"
}
```

L'ajout de la `.sync` partie à la ressource Amazon Resource Name (ARN) signifie que Step Functions attend que le travail soit terminé. Après l'appel de AWS Batch `submitJob`, le flux de travail s'interrompt. Lorsque le travail est terminé, Step Functions passe à l'état suivant. Pour plus d'informations, consultez l' AWS Batch exemple de projet :Gérer un traitement par lots (AWS Batch,Amazon SNS).

Si une tâche utilisant ce modèle d'intégration de service (`.sync`) est abandonnée et que Step Functions ne parvient pas à annuler la tâche, le service intégré peut vous facturer des frais supplémentaires. Une tâche peut être abandonnée si :

- L'exécution de la machine à états est arrêtée.
- Une branche différente d'un état parallèle échoue avec une erreur non détectée.
- Une itération de l'état d'une carte échoue avec une erreur non détectée.

Step Functions fera de son mieux pour annuler la tâche. Par exemple, si une `states:startExecution.sync` tâche Step Functions est abandonnée, elle appellera l'action `StopExecution` API Step Functions. Cependant, il est possible que Step Functions ne soit pas en mesure d'annuler la tâche. Les raisons en sont notamment les suivantes :

- Votre rôle d'exécution IAM n'est pas autorisé à effectuer l'appel d'API correspondant.
- Une interruption de service temporaire s'est produite.

Lorsque vous utilisez le modèle d'intégration des `.sync` services, Step Functions utilise des sondages qui utilisent le quota et les événements qui vous ont été assignés pour surveiller le statut d'une tâche. Pour les `.sync` invocations au sein d'un même compte, Step Functions utilise EventBridge des événements et interroge les API que vous spécifiez dans l'`Task`état. Pour les `.sync`invocations entre comptes, Step Functions utilise uniquement le sondage. Par exemple,

pour`states:StartExecution.sync`, Step Functions interroge l'DescribeExecutionAPI et utilise le quota qui vous a été attribué.

> **Tip**
>
> Pour déployer un exemple de flux de travail utilisant le modèle d'intégration du service Run a Job (.sync) sur votre ordinateur Compte AWS, consultez le module 3 - Run a Job (.sync) de The AWS Step Functions Workshop.

Pour afficher la liste des services intégrés qui prennent en charge l'attente de la fin d'exécution d'une tâche (`.sync`), consultez Intégrations optimisées pour Step Functions.

> **Note**
>
> Les intégrations de services qui utilisent le `.sync` modèle nécessitent des autorisations IAM supplémentaires. Pour plus d'informations, consultez Politiques IAM pour les services intégrés.

Dans certains cas, vous souhaiterez peut-être que Step Functions continue votre flux de travail avant que le travail ne soit complètement terminé. Vous pouvez y parvenir de la même manière que lorsque vous utilisez le modèle d'intégration des Attendre un rappel avec le jeton de tâche services. Pour ce faire, transmettez un jeton de tâche à votre tâche, puis renvoyez-le à l'aide d'un appel d'`SendTaskFailure`API `SendTaskSuccess`ou d'API. Step Functions utilisera les données que vous fournissez lors de cet appel pour terminer la tâche, arrêter de surveiller la tâche et poursuivre le flux de travail.

Attendre un rappel avec le jeton de tâche

Les tâches de rappel fournissent un moyen de suspendre un flux de travail jusqu'à ce qu'un jeton de tâche soit renvoyé. Une tâche peut avoir besoin d'attendre une approbation humaine, de s'intégrer à un tiers ou d'appeler un système existant. Pour de telles tâches, vous pouvez suspendre Step Functions jusqu'à ce que l'exécution du flux de travail atteigne le quota de service d'un an (voir,Quotas liés à l'étranglement de l'État), et attendre qu'un processus ou un flux de travail externe soit terminé. Dans ces situations, Step Functions vous permet de transmettre un jeton de tâche aux intégrations de services du AWS SDK, ainsi qu'à certaines intégrations de services

optimisées. La tâche s'arrête jusqu'à ce qu'elle reçoive à nouveau le jeton de tâche avec un appel `SendTaskSuccess` ou `SendTaskFailure`.

Si un `Task` état utilisant le jeton de tâche de rappel expire, un nouveau jeton aléatoire est généré. Vous pouvez accéder aux jetons de tâche depuis l'objet de contexte.

> **Note**
>
> Un jeton de tâche doit contenir au moins un caractère et ne peut pas dépasser 1 024 caractères.

Pour être utilisée dans le cadre `.waitForTaskToken` d'une intégration au AWS SDK, l'API que vous utilisez doit comporter un champ de paramètres dans lequel placer le jeton de tâche.

> **Note**
>
> Vous devez transmettre les jetons de tâches des principaux d'un même AWS compte. Les jetons ne fonctionneront pas si vous les envoyez depuis un compte principal sur un autre AWS compte.

> **Tip**
>
> Pour déployer un exemple de flux de travail utilisant un modèle d'intégration du service de jeton de tâche de rappel sur votre Compte AWS ordinateur, consultez le module 4 - Attendre un rappel avec le jeton de tâche de The AWS Step Functions Workshop.

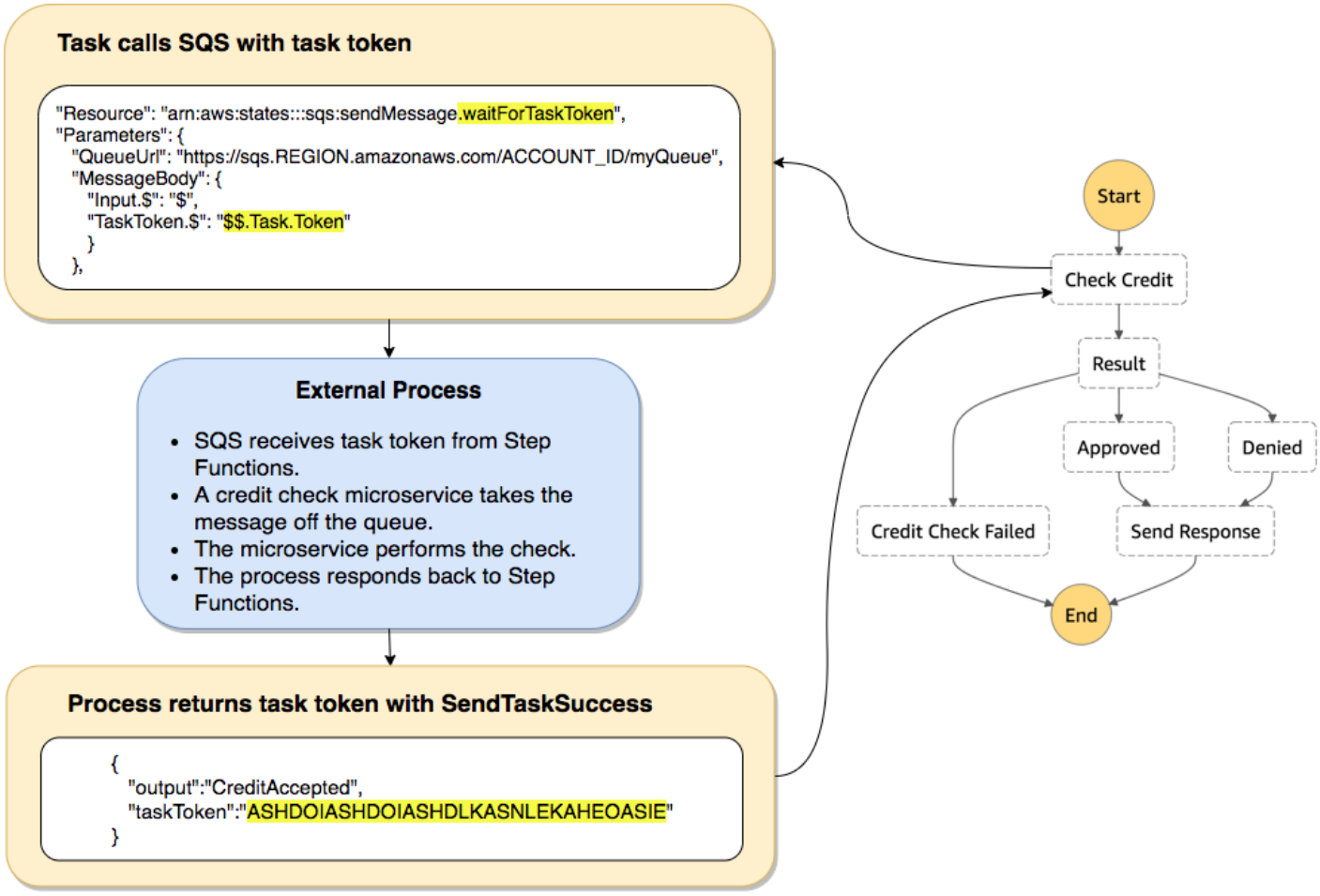
Pour afficher la liste des services intégrés qui prennent en charge l'attente d'un jeton de tâche (`.waitForTaskToken`), consultez Intégrations optimisées pour Step Functions.

Rubriques

- Exemple de jeton de tâche
- Obtenir un jeton à partir de l'objet de contexte
- Configurer un délai de pulsation pour une tâche en attente

Exemple de jeton de tâche

Dans cet exemple, un flux de travail Step Functions doit être intégré à un microservice externe pour effectuer une vérification de solvabilité dans le cadre d'un flux de travail d'approbation. Step Functions publie un message Amazon SQS qui inclut un jeton de tâche dans le message. Un système externe s'intègre à Amazon SQS et extrait le message de la file d'attente. Lorsque cela est terminé, il renvoie le résultat et le jeton de tâche d'origine. Step Functions poursuit ensuite son flux de travail.



Le `"Resource"` champ de la définition de tâche qui fait référence à Amazon SQS est `.waitForTaskToken` ajouté à la fin.

```
"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"
}
```

Cela indique à Step Functions de faire une pause et d'attendre le jeton de tâche. Lorsque vous spécifiez une ressource à l'aide de `.waitForTaskToken`, le jeton de tâche est accessible dans le champ `"Parameters"` de votre définition d'état avec une désignation de chemin spécifique (`$$.Task.Token`). La première `$$.` désigne que le chemin accède à l'objet de contexte, et obtient le jeton de la tâche pour la tâche courante d'une exécution en cours.

Une fois qu'il a terminé, le service externe appelle `SendTaskSuccess` ou `SendTaskFailure` avec `taskToken` inclus. Seul alors le flux de travail continue à l'état suivant.

> **Note**
>
> Pour éviter d'attendre indéfiniment si un processus échoue à envoyer le jeton de tâche en même temps que `SendTaskSuccess` ou `SendTaskFailure`, veuillez consulter Configurer un délai de pulsation pour une tâche en attente.

Obtenir un jeton à partir de l'objet de contexte

L'objet de contexte est un objet JSON interne qui contient des informations sur votre exécution. Comme entrée d'état, il peut être consulté avec un chemin d'accès à partir du champ `"Parameters"` pendant une exécution. En cas d'accès à partir d'une définition de tâche, l'objet comprend des informations sur l'exécution spécifique, y compris le jeton de tâche.

```
{
    "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"
    }
}
```

Vous pouvez accéder au jeton de tâche à l'aide d'un chemin d'accès spécial depuis l'intérieur du champ `"Parameters"` de votre définition de tâche. Pour accéder à l'entrée ou l'objet de contexte, vous devez d'abord spécifier que le paramètre sera un chemin d'accès en ajoutant un `.$` au nom de paramètre. La commande suivante spécifie les nœuds des données d'entrée et de l'objet de contexte dans une spécification `"Parameters"`.

```
"Parameters": {
   "Input.$": "$",
   "TaskToken.$": "$$.Task.Token"
 },
```

Dans les deux cas, l'ajout au nom du paramètre indique `.$` à Step Functions de s'attendre à un chemin. Dans le premier cas, `"$"` est un chemin d'accès qui inclut l'ensemble de l'entrée. Dans le deuxième cas, `$$.` spécifie que le chemin doit accéder à l'objet de contexte et `$$.Task.Token` définit le paramètre avec la valeur du jeton de tâche dans l'objet de contexte d'une exécution en cours.

Dans l'exemple d'Amazon SQS, le `"Resource"` champ indique `.waitForTaskToken` à Step Functions d'attendre que le jeton de tâche soit renvoyé. Le `"TaskToken.$": "$$.Task.Token"` paramètre transmet ce jeton dans le message Amazon SQS.

```
"Send message to SQS": {
  "Type": "Task",
  "Resource": "arn:aws:states:::sqs:sendMessage.waitForTaskToken",
  "Parameters": {
    "QueueUrl": "https://sqs.us-east-2.amazonaws.com/123456789012/myQueue",
```

```
    "MessageBody": {
        "Message": "Hello from Step Functions!",
        "TaskToken.$": "$$.Task.Token"
     }
  },
  "Next": "NEXT_STATE"
}
```

Pour plus d'informations sur l'objet de contexte, consultez Objet Contexte dans la section Traitement des entrées et des sorties de ce guide.

Configurer un délai de pulsation pour une tâche en attente

Une tâche qui attend un jeton de tâche attendra jusqu'à ce que l'exécution ait atteint le quota d'une année de service (consultez, Quotas liés à l'étranglement de l'État). Pour éviter les exécutions bloquées, vous pouvez configurer un intervalle de pulsation dans la définition de votre machine d'état. Utilisez le champ `HeartbeatSeconds` pour spécifier l'intervalle de délai d'expiration.

```
{
  "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
    }
  }
}
```

Dans cette définition de machine à états, une tâche envoie un message à Amazon SQS et attend qu'un processus externe le rappelle avec le jeton de tâche fourni. Le champ `"HeartbeatSeconds": 600` définit l'intervalle de pulsation sur 10 minutes. La tâche attend que le jeton de tâche soit renvoyé avec l'une de ces actions d'API :

- `SendTaskSuccess`

- SendTaskFailure
- SendTaskHeartbeat

Si la tâche d'attente ne reçoit pas un jeton de tâche valide au sein de cette période de 10 minutes, la tâche échoue avec un nom d'erreur `States.Timeout`.

Pour de plus amples informations, veuillez consulter l'exemple de projet de tâche de rappel Exemple de modèle de rappel (Amazon SQS, Amazon SNS, Lambda) .

Transmettre des paramètres à une API de service

Utilisez le champ `Parameters` dans un état `Task` pour contrôler quels paramètres sont transmis à une API de service.

À l'intérieur du `Parameters` champ, vous devez utiliser la forme plurielle des paramètres du tableau dans une action d'API. Par exemple, si vous utilisez le champ Filtre de l'action d'`DescribeSnapshots`API pour l'intégration à Amazon EC2, vous devez définir le champ comme`Filters`. Si vous n'utilisez pas la forme plurielle, Step Functions renvoie l'erreur suivante :

```
The field Filter is not supported by Step Functions.
```

Transmettre du JSON statique en tant que paramètres

Vous pouvez inclure un objet JSON directement dans la définition de votre machine d'état pour transmettre en tant que paramètre vers une ressource.

Par exemple, pour définir le paramètre `RetryStrategy` de l'API `SubmitJob` pour AWS Batch, vous pouvez inclure ce qui suit dans vos paramètres.

```
"RetryStrategy": {
  "attempts": 5
}
```

Vous pouvez également transmettre plusieurs paramètres avec JSON statique. À titre d'exemple plus complet, voici les `Parameters` champs `Resource` et de la spécification d'une tâche qui publie sur une rubrique Amazon SNS nommée*`myTopic`*.

```
"Resource": "arn:aws:states:::sns:publish",
```

```
  "Parameters": {
     "TopicArn": "arn:aws:sns:us-east-2:123456789012:myTopic",
     "Message": "test message",
     "MessageAttributes": {
       "my attribute no 1": {
         "DataType": "String",
         "StringValue": "value of my attribute no 1"
       },
       "my attribute no 2": {
         "DataType": "String",
         "StringValue": "value of my attribute no 2"
       }
     }
  },
```

Transmettre l'entrée d'état en tant que paramètres à l'aide de chemins

Vous pouvez transmettre des parties de l'entrée d'état sous forme de paramètres à l'aide de chemins. Un chemin est une chaîne commençant par$, utilisée pour identifier les composants du texte JSON. Les chemins Step Functions utilisent JsonPathla syntaxe.

Pour spécifier qu'un paramètre utilise un chemin, terminez le nom du paramètre par `.$`. Par exemple, si votre entrée d'état contient du texte dans un nœud nommé`message`, vous pouvez transmettre ce texte en tant que paramètre à l'aide d'un chemin.

Prenez en compte l'entrée d'état suivante :

```
{
  "comment": "A message in the state input",
  "input": {
    "message": "foo",
    "otherInfo": "bar"
  },
  "data": "example"
}
```

Pour transmettre la valeur du nœud nommé `message` en tant que paramètre, spécifiez la syntaxe suivante :

```
"Parameters": {"myMessage.$": "$.input.message"},
```

Step Functions transmet ensuite la valeur `foo` en tant que paramètre.

Pour plus d'informations sur l'utilisation des paramètres dans Step Functions, consultez les rubriques suivantes :

- Traitement des entrées et des sorties
- InputPath, Paramètres et ResultSelector

Transmettre les nœuds d’objet de contexte comme paramètres

En plus du contenu statique, et des nœuds de l’état d'entrée, vous pouvez transmettre les nœuds du contexte d'objet comme paramètres. L'objet de contexte correspond à des données JSON dynamiques qui existent pendant l'exécution d'une machine d'état. Il inclut des informations sur votre machine d'état et l'exécution actuelle. Vous pouvez accéder à l'objet de contexte à l'aide d'un chemin d'accès dans le champ `"Parameters"` d'une définition d’état.

Pour de plus amples informations sur l'objet de contexte et la façon d'accéder à ces données à partir d'un champ `"Parameters"`, veuillez consulter :

- Objet Contexte
- Accès à l'objet de contexte
- Obtenir un jeton à partir de l'objet de contexte

Journal des modifications pour les intégrations de AWS SDK prises en charge

Le tableau suivant indique à quel moment les services sont initialement intégrés à Step Functions et à quel moment leur API d'intégration a été mise à jour pour la dernière fois. Pour plus de détails sur l'utilisation des intégrations, consultezAWS Intégrations de services SDK.

> **Important**
>
> Le support aux actions de l'API est publié tous les trimestres. Les mises à jour des actions déjà prises en charge, telles que les nouveaux paramètres, peuvent ne pas être disponibles immédiatement.

Service	Support initial	Mis à jour	
AWS AppFabric	18 janvier 2024		
B2B Data Interchange	18 janvier 2024		
Exportations de données AWS	18 janvier 2024		
Amazon Bedrock	18 janvier 2024		
Amazon Bedrock Agents	18 janvier 2024		
Amazon Bedrock Runtime Agents	18 janvier 2024		
Amazon Bedrock Runtime	18 janvier 2024		
Amazon CloudFront KeyValueStore	18 janvier 2024		
Amazon CodeGuru Security	18 janvier 2024		
Hub d'optimisation des coûts	18 janvier 2024		
Amazon DataZone	18 janvier 2024		
Amazon EKS Auth	18 janvier 2024		
Résolution des entités AWS	18 janvier 2024		
Niveau gratuit d'AWS	18 janvier 2024		
Amazon Inspector Scan	18 janvier 2024		

Service	Support initial	Mis à jour	
AWS Launch Wizard	18 janvier 2024		
Amazon Managed Blockchain Query	18 janvier 2024		
AWS Elemental MediaPackage V2	18 janvier 2024		
AWS HealthImaging	18 janvier 2024		
Network Manager	18 janvier 2024		
AWS Payment Cryptography	18 janvier 2024		
AWS Payment Cryptography Data	18 janvier 2024		
AWS Private CA Connector for Active Directory	18 janvier 2024		
Amazon Q Business	18 janvier 2024		
Amazon Q Connect	18 janvier 2024		
AWS re:Post	18 janvier 2024		
Amazon Timestream Query	18 janvier 2024		
Amazon Timestream Write	18 janvier 2024		
Trusted Advisor	18 janvier 2024		
Verified Permissions	18 janvier 2024		

Service	Support initial	Mis à jour	
Amazon WorkSpaces Thin Client	18 janvier 2024		
AWS CloudTrail Data	16 juin 2023		
Amazon CloudWatch Internet Monitor	16 juin 2023		
Amazon Interacti ve Video Service RealTime	16 juin 2023		
AWS IoT TwinMaker	16 juin 2023		
Amazon OpenSearch Ingestion	16 juin 2023		
AWS Telco Network Builder	16 juin 2023		
Amazon VPC Lattice	16 juin 2023		
AWS Backup Storage	17 février 2023		
Amazon Chime Media Pipelines	17 février 2023		
Amazon Chime Voice	17 février 2023		
AWS Clean Rooms	17 février 2023	18 janvier 2024	
Amazon CodeCatalyst	17 février 2023		
Amazon Connect Cases	17 février 2023		
AWS Control Tower	17 février 2023		

Service	Support initial	Mis à jour	
Amazon DocumentDB Elastic Clusters	17 février 2023		
Amazon EMR Serverless	17 février 2023		
Amazon IVS Chat	17 février 2023		
Amazon Kendra Intelligent Ranking	17 février 2023		
AWS HealthOmics	17 février 2023		
Amazon Redshift Serverless	17 février 2023		
Amazon Security Lake	17 février 2023		
AWS Health	17 février 2023		
AWS IoT FleetWise	17 février 2023		
AWS IoT RoboRunner	17 février 2023		
AWS Mainframe Modernization	17 février 2023		
Orchestrateur de l'AWS Migration Hub	17 février 2023		
AWS Private 5G	17 février 2023		
Explorateur de ressources AWS	17 février 2023		
AWS SimSpace Weaver	17 février 2023		

Service	Support initial	Mis à jour	
AWS Support App	17 février 2023		
CloudWatch Observability Access Manager	17 février 2023		
EventBridge Pipes	17 février 2023		
EventBridge Scheduler	17 février 2023		
IAM Roles Anywhere	17 février 2023		
Kinesis Video WebRTC Storage	17 février 2023		
License Manager Linux Subscriptions	17 février 2023		
License Manager User Subscriptions	17 février 2023		
OpenSearch Serverless	17 février 2023		
Route 53 ARC Zonal Shift	17 février 2023		
SageMaker Geospatial	17 février 2023		
SageMaker Metrics	17 février 2023		
Systems Manager for SAP	17 février 2023		
AWS Account Management	19 avril 2022		

Service	Support initial	Mis à jour	
AWS Amplify	30 septembre 2021		
AWS App Mesh	30 septembre 2021		
AWS App Runner	30 septembre 2021	17 février 2023	
AWS AppConfig	30 septembre 2021		
AWS AppConfig Data	19 avril 2022		
AWS AppSync	30 septembre 2021	17 février 2023	
AWS Application Discovery Service	30 septembre 2021		
AWS Application Migration Service	30 septembre 2021		
AWS Audit Manager	30 septembre 2021		
AWS Auto Scaling Plans	30 septembre 2021		
AWS Backup	30 septembre 2021	17 février 2023	
AWS Backup gateway	19 avril 2022	17 février 2023	
AWS Batch	30 septembre 2021	17 février 2023	
AWS Billing Conductor	26 juillet 2022	17 février 2023	
AWS Budgets	30 septembre 2021		
AWS Certificate Manager	30 septembre 2021		
AWS Private Certificate Authority	30 septembre 2021		

Service	Support initial	Mis à jour	
AWS Cloud Map	30 septembre 2021		
AWS Cloud9	30 septembre 2021		
AWS CloudFormation	30 septembre 2021	17 février 2023	
AWS CloudHSM	30 septembre 2021		
AWS CloudHSM	30 septembre 2021		
AWS CloudTrail	30 septembre 2021	17 février 2023	
AWS Cloud Control	19 avril 2022		
AWS CodeBuild	30 septembre 2021		
AWS CodeCommit	30 septembre 2021	17 février 2023	
AWS CodeDeploy	30 septembre 2021		
AWS CodePipeline	30 septembre 2021		
AWS CodeStar	30 septembre 2021		
AWS CodeStar	30 septembre 2021		
AWS CodeStar	30 septembre 2021		
AWS Compute Optimizer	30 septembre 2021	17 février 2023	
AWS Config	30 septembre 2021	26 juillet 2022	
AWS Cost Explorer Service	30 septembre 2021	17 février 2023	
AWS Cost and Usage Report	30 septembre 2021		
AWS Data Exchange	30 septembre 2021	26 juillet 2022	

Service	Support initial	Mis à jour	
AWS Data Pipeline	30 septembre 2021		
AWS DataSync	30 septembre 2021	26 juillet 2022	
AWS Database Migration Service	30 septembre 2021		
AWS Device Farm	30 septembre 2021		
AWS Direct Connect	30 septembre 2021		
AWS Directory Service	30 septembre 2021	17 février 2023	
AWS EC2 Instance Connect	30 septembre 2021		
AWS Elastic Beanstalk	30 septembre 2021		
AWS Elemental MediaLive	30 septembre 2021		
AWS Elemental MediaPackage	30 septembre 2021		
AWS Elemental MediaPackage VOD	30 septembre 2021		
AWS Elemental MediaStore	30 septembre 2021		
AWS Fault Injection Service	30 septembre 2021		
AWS Firewall Manager	30 septembre 2021	17 février 2023	

Service	Support initial	Mis à jour	
AWS Glue	30 septembre 2021	17 février 2023	
AWS Glue DataBrew	30 septembre 2021		
AWS IoT Greengrass	30 septembre 2021		
AWS Ground Station	30 septembre 2021	17 février 2023	
AWS Identity and Access Management	30 septembre 2021		
AWS IoT	30 septembre 2021	17 février 2023	
AWS IoT 1-Click	30 septembre 2021		
AWS IoT Analytics	30 septembre 2021		
AWS IoT Core Device Advisor	30 septembre 2021		
AWS IoT Events	30 septembre 2021		
AWS IoT Events Data	30 septembre 2021		
AWS IoT Fleet Hub	30 septembre 2021		
AWS IoT Greengrass Version 2	30 septembre 2021		
AWS IoT Jobs Data Plane	30 septembre 2021		
AWS IoT Secure Tunneling	30 septembre 2021		
AWS IoT SiteWise	30 septembre 2021	17 février 2023	
AWS IoT Things Graph	30 septembre 2021		

Service	Support initial	Mis à jour	
AWS IoT Wireless	30 septembre 2021	17 février 2023	
AWS Key Management Service	30 septembre 2021	26 juillet 2022	
AWS Lake Formation	30 septembre 2021	17 février 2023	
AWS Lambda	30 septembre 2021	17 février 2023	
AWS License Manager	30 septembre 2021	17 février 2023	
AWS Marketplace	30 septembre 2021	17 février 2023	
AWS Marketplace Commerce Analytics	30 septembre 2021		
AWS Marketplace Entitlement Service	30 septembre 2021		
AWS Elemental MediaTailor	30 septembre 2021	26 juillet 2022	
AWS Migration Hub	30 septembre 2021		
AWS Migration Hub Config	30 septembre 2021		
Recommandations stratégiques d'AWS Migration Hub	19 avril 2022	17 février 2023	
AWS Mobile	30 septembre 2021		
AWS Network Firewall	30 septembre 2021		
AWS OpsWorks	30 septembre 2021		
AWS OpsWorks CM	30 septembre 2021		

Service	Support initial	Mis à jour	
AWS Organizations	30 septembre 2021	17 février 2023	
AWS Outposts	30 septembre 2021		
AWS Panorama	19 avril 2022	17 février 2023	
Amazon Relationa l Database Service Performance Insights	30 septembre 2021		
AWS Price List	30 septembre 2021		
Amazon Relational Database Service	30 septembre 2021		
AWS Resilience Hub	19 avril 2022		
AWS Resource Access Manager	30 septembre 2021		
AWS Resource Groups	30 septembre 2021	17 février 2023	
AWS Resource Groups Tagging API	30 septembre 2021		
AWS RoboMaker	30 septembre 2021		
AWS IAM Identity Center	30 septembre 2021	17 février 2023	
AWS SSO OIDC	30 septembre 2021		
AWS Secrets Manager	30 septembre 2021		
AWS Security Token Service	30 septembre 2021		

Service	Support initial	Mis à jour	
AWS Security Hub	30 septembre 2021		
AWS Server Migration Service	30 septembre 2021		
AWS Service Catalog	30 septembre 2021		
AWS Service Catalog AppRegistry	30 septembre 2021	17 février 2023	
AWS Shield	30 septembre 2021		
AWS Signer	30 septembre 2021		
AWS IAM Identity Center	30 septembre 2021		
AWS IAM Identity Center Admin	30 septembre 2021		
AWS Step Functions	30 septembre 2021	17 février 2023	
AWS Storage Gateway	30 septembre 2021		
AWS Support	30 septembre 2021		
AWS Transfer Family	30 septembre 2021	17 février 2023	
AWS WAF	30 septembre 2021		
AWS WAF Regional	30 septembre 2021		
AWS WAFV2	30 septembre 2021		
AWS Well-Architected Tool	30 septembre 2021	17 février 2023	
AWS X-Ray	30 septembre 2021	17 février 2023	

Service	Support initial	Mis à jour	
AWS Marketplace Metering Service	30 septembre 2021		
AWS Serverless Application Repository	30 septembre 2021		
AWS Identity and Access Management Access Analyzer	30 septembre 2021		
Alexa for Business	30 septembre 2021		
Amazon API Gateway	30 septembre 2021	17 février 2023	
Amazon API Gateway	30 septembre 2021		
Amazon AppIntegrations	30 septembre 2021		
Amazon AppStream 2.0	30 septembre 2021		
Amazon AppFlow	30 septembre 2021	17 février 2023	
Amazon Athena	30 septembre 2021	17 février 2023	
Amazon Augmented AI	30 septembre 2021		
Amazon Braket	30 septembre 2021		
Amazon Chime	30 septembre 2021		
Amazon Chime Meetings	19 avril 2022	17 février 2023	
Amazon Cloud Directory	30 septembre 2021		

Service	Support initial	Mis à jour	
Amazon CloudFront	30 septembre 2021	17 février 2023	
Amazon CloudSearch	30 septembre 2021		
Amazon CloudWatch	30 septembre 2021	17 février 2023	
Amazon CloudWatch Application Insights	30 septembre 2021		
CloudWatch Evidently	19 avril 2022		
Amazon CloudWatch Logs	30 septembre 2021		
Amazon CloudWatch RUM	19 avril 2022	17 février 2023	
Amazon CloudWatch Synthetics	30 septembre 2021		
Amazon CodeGuru Profiler	30 septembre 2021		
Amazon CodeGuru Reviewer	30 septembre 2021		
Amazon Cognito	30 septembre 2021		
Amazon Cognito Identity Provider	30 septembre 2021		
Amazon Cognito Sync	30 septembre 2021		
Amazon Comprehend	30 septembre 2021	17 février 2023	
Amazon Comprehend Medical	30 septembre 2021		

Service	Support initial	Mis à jour	
Amazon Connect Contact Lens	30 septembre 2021		
Amazon Connect Participant Service	30 septembre 2021		
Amazon Connect	30 septembre 2021	17 février 2023	
Amazon Connect Voice ID	19 avril 2022		
Amazon Connect Wisdom	19 avril 2022		
Amazon Data Lifecycle Manager	30 septembre 2021		
Amazon Detective	30 septembre 2021		
Amazon DevOps Guru	30 septembre 2021	26 juillet 2022	
Amazon DocumentDB (with MongoDB compatibility)	30 septembre 2021		
Amazon DynamoDB	30 septembre 2021	17 février 2023	
Amazon DynamoDB Streams	30 septembre 2021		
Amazon EC2 Container Registry	30 septembre 2021		
Amazon EC2 Container Service	30 septembre 2021	17 février 2023	

Service	Support initial	Mis à jour	
Amazon EC2 Systems Manager	30 septembre 2021	17 février 2023	
Amazon EMR	30 septembre 2021	17 février 2023	
Amazon ElastiCache	30 septembre 2021		
Amazon Elastic Inference	30 septembre 2021		
Amazon Elastic Block Store	30 septembre 2021		
Amazon Elastic Compute Cloud	30 septembre 2021	17 février 2023	
Amazon Elastic Container Registry Public	30 septembre 2021		
Amazon Elastic File System	30 septembre 2021		
Amazon Elastic Kubernetes Service	30 septembre 2021	17 février 2023	
Amazon EMR	30 septembre 2021		
Amazon Elastic Transcoder	30 septembre 2021		
Amazon OpenSearch Service	30 septembre 2021	17 février 2023	
Amazon OpenSearch Service	19 avril 2022	17 février 2023	
Amazon EventBridge	30 septembre 2021	17 février 2023	

Service	Support initial	Mis à jour	
Amazon FSx	30 septembre 2021	17 février 2023	
Amazon Forecast Query	30 septembre 2021	17 février 2023	
Amazon Forecast Service	30 septembre 2021	17 février 2023	
Amazon Fraud Detector	30 septembre 2021		
Amazon GameLift	30 septembre 2021	17 février 2023	
Amazon GameSparks	26 juillet 2022		
Amazon S3 Glacier	30 septembre 2021		
Amazon GuardDuty	30 septembre 2021		
AWS HealthLake	30 septembre 2021		
Amazon Honeycode	30 septembre 2021		
Amazon Inspector	30 septembre 2021		
Amazon Inspector V2	19 avril 2022		
Amazon Interactive Video Service	30 septembre 2021		
Amazon Kendra	30 septembre 2021		
Amazon Kinesis	30 septembre 2021		
Amazon Kinesis Analytics	30 septembre 2021		

Service	Support initial	Mis à jour	
Amazon Kinesis Analytics V2	30 septembre 2021		
Amazon Kinesis Firehose	30 septembre 2021		
Amazon Kinesis Video Signaling Channels	30 septembre 2021		
Amazon Kinesis Video Streams	30 septembre 2021	17 février 2023	
Amazon Kinesis Video Streams Archived Media	30 septembre 2021		
Amazon Kinesis video stream	30 septembre 2021		
Amazon Lex Model Building Service	30 septembre 2021		
Amazon Lex Model Building Service V2	30 septembre 2021	17 février 2023	
Amazon Lex	30 septembre 2021		
Amazon Lex Runtime V2	30 septembre 2021		
Amazon Lightsail	30 septembre 2021	17 février 2023	
Amazon Location Service	30 septembre 2021	17 février 2023	
Amazon Lookout for Equipment	30 septembre 2021		

Service	Support initial	Mis à jour	
Amazon Lookout for Metrics	30 septembre 2021	17 février 2023	
Amazon Lookout for Vision	30 septembre 2021		
Amazon MQ	30 septembre 2021		
Amazon Macie	30 septembre 2021		
Amazon Macie 2	30 septembre 2021	17 février 2023	
Amazon Managed Blockchain	30 septembre 2021	17 février 2023	
Amazon Managed Grafana	19 avril 2022	17 février 2023	
Amazon Managed Service for Prometheus	30 septembre 2021	17 février 2023	
Amazon Managed Streaming for Apache Kafka	30 septembre 2021	17 février 2023	
Amazon Managed Streaming for Apache Kafkax	19 avril 2022		
Amazon Managed Workflows for Apache Airflow	30 septembre 2021		
Amazon Mechanical Turk	30 septembre 2021		

Service	Support initial	Mis à jour	
Amazon MemoryDB for Redis	19 avril 2022	17 février 2023	
Amazon Nimble Studio	30 septembre 2021		
Amazon Personalize	30 septembre 2021	17 février 2023	
Amazon Personalize Events	30 septembre 2021		
Amazon Personalize Runtime	30 septembre 2021		
Amazon Pinpoint	30 septembre 2021		
Amazon Pinpoint Email Service	30 septembre 2021		
Amazon Pinpoint SMS and Voice Service	30 septembre 2021		
Amazon Pinpoint SMS and Voice V2 Service	26 juillet 2022		
Amazon Polly	30 septembre 2021		
Amazon QLDB	30 septembre 2021		
Amazon QLDB Session	30 septembre 2021		
Amazon QuickSight	30 septembre 2021	17 février 2023	
Amazon Redshift	30 septembre 2021		

Service	Support initial	Mis à jour	
Amazon Redshift Data API	30 septembre 2021		
Amazon Rekognition	30 septembre 2021	17 février 2023	
Amazon Relational Database Service	30 septembre 2021	17 février 2023	
Amazon Route 53	30 septembre 2021		
Amazon Route 53 Recovery Control Config	30 septembre 2021	26 juillet 2022	
Amazon Route 53 Domains	30 septembre 2021	17 février 2023	
Amazon Route 53 Resolver	30 septembre 2021		
Amazon S3 on Outposts	30 septembre 2021	26 juillet 2022	
Amazon SageMaker Runtime Feature Store Runtime	30 septembre 2021		
Amazon SageMaker Runtime Runtime	30 septembre 2021		
Amazon SageMaker	30 septembre 2021	17 février 2023	
Amazon SageMaker Edge Manager	30 septembre 2021		
Amazon Simple Email Service	30 septembre 2021		

Service	Support initial	Mis à jour	
Amazon Simple Email Service V2	30 septembre 2021	17 février 2023	
Amazon Simple Notification Service	30 septembre 2021	17 février 2023	
Amazon Simple Queue Service	30 septembre 2021	17 février 2023	
Amazon Simple Storage Service	30 septembre 2021	17 février 2023	
Amazon Simple Workflow Service	30 septembre 2021		
Amazon Textract	30 septembre 2021	17 février 2023	
Amazon Transcribe	30 septembre 2021		
Amazon Translate	30 septembre 2021	17 février 2023	
Amazon WorkDocs	30 septembre 2021	17 février 2023	
Amazon WorkMail	30 septembre 2021	17 février 2023	
Amazon WorkMail Message Flow	30 septembre 2021		
Amazon WorkSpaces	30 septembre 2021	17 février 2023	
Amazon WorkSpaces Web	19 avril 2022	17 février 2023	
Amplify	30 septembre 2021		
Amplify UI Builder	19 avril 2022	17 février 2023	
Application Auto Scaling	30 septembre 2021		

Service	Support initial	Mis à jour	
Amazon EC2 Auto Scaling	30 septembre 2021	17 février 2023	
CodeArtifact	30 septembre 2021		
DynamoDB Accelerator	30 septembre 2021		
EC2 Image Builder	30 septembre 2021		
AWS Elastic Disaster Recovery	19 avril 2022	17 février 2023	
Elastic Load Balancing	30 septembre 2021		
Elastic Load Balancing V2	30 septembre 2021		
MediaConnect	30 septembre 2021		
Amazon S3 Control	30 septembre 2021	17 février 2023	
Recycle Bin for Amazon EBS	19 avril 2022	17 février 2023	
Savings Plans	30 septembre 2021		
Amazon EventBridge Schema Registry	30 septembre 2021		
Service Quotas	30 septembre 2021		
AWS Snowball	30 septembre 2021		

Exemples de projets pour Step Functions

Dans la AWS Step Functionsconsole, vous pouvez choisir l'un des modèles de démarrage suivants pour déployer des machines d'état sur votreComptes AWS. Ces modèles de démarrage sont des ready-to-run exemples de projets qui créent automatiquement le prototype et la définition du flux de travail, ainsi que toutes les AWS ressources associées au projet.

Vous pouvez utiliser ces exemples de projets pour les déployer et les exécuter tels quels, ou utiliser les prototypes de flux de travail pour les exploiter. Si vous vous basez sur ces projets, Step Functions crée le prototype du flux de travail, mais ne déploie pas les ressources répertoriées dans la définition du flux de travail.

Lorsque vous déployez les exemples de projets, ils fournissent une machine d'état entièrement fonctionnelle et créent les ressources associées pour que la machine d'état puisse s'exécuter. Lorsque vous créez un exemple de projet, Step Functions l'utilise AWS CloudFormation pour créer les ressources associées référencées par la machine à états.

Rubriques

- Gérer un traitement par lots (AWS Batch,Amazon SNS)
- Gérer une tâche de conteneur (Amazon ECS,Amazon SNS)
- Transférer des enregistrements de données (Lambda,DynamoDB,Amazon SQS)
- Sondage pour connaître le statut du poste (Lambda,) AWS Batch
- Minuteur de tâches (Lambda, Amazon SNS)
- Exemple de modèle de rappel (Amazon SQS, Amazon SNS, Lambda)
- Gérer une offre d'emploi Amazon EMR
- Exécuter une EMR Serverless tâche
- Démarrer un flux de travail au sein d'un flux de travail (Step Functions, Lambda)
- Traitement dynamique des données avec un état cartographique
- Traiter un fichier CSV avec une carte distribuée
- Traitez les données dans un compartiment Amazon S3 avec Distributed Map
- Former un modèle de Machine Learning
- Régler un modèle de Machine Learning
- Traitement de gros volumes de messages depuis Amazon SQS (Express Workflows)

- Exemple de point de contrôle sélectif (workflows express)
- Création d'un AWS CodeBuild projet (CodeBuild, Amazon SNS)
- Prétraitez les données et entraînez un modèle d'apprentissage automatique
- Exemple d'orchestration Lambda
- Lancer une requête Athena
- Exécuter plusieurs requêtes (Amazon Athena, Amazon SNS)
- Interrogez de grands ensembles de données (Amazon Athena, Amazon S3 AWS Glue, Amazon SNS)
- Maintenir les données à jour (Amazon Athena, Amazon S3,) AWS Glue
- Gérer un cluster Amazon EKS
- Passez un appel à API Gateway
- Appelez un microservice s'exécutant sur Fargate à l'aide de l'intégration d'API Gateway
- Envoyer un événement personnalisé à EventBridge
- Invoquer des flux de travail express synchrones
- Exécutez des flux de travail ETL/ELT à l'aide d'Amazon Redshift (Lambda, API de données Amazon Redshift)
- Utilisation Step Functions et gestion AWS Batch des erreurs
- Répartissez un AWS Batch travail
- AWS Batch avec Lambda
- Réalisez un enchaînement d'instructions basé sur l'IA avec Amazon Bedrock

Gérer un traitement par lots (AWS Batch,Amazon SNS)

Cet exemple montre comment soumettre une tâche AWS Batch, puis comment envoyer une notification Amazon SNS selon que cette tâche a réussi ou échoué. Le déploiement de cet exemple de projet crée une machine d'état AWS Step Functions, une tâche AWS Batch et une rubrique Amazon SNS.

Dans ce projet, Step Functions utilise une machine d'état pour appeler la tâche AWS Batch de manière synchrone. Il attend ensuite que la tâche réussisse ou échoue, et il envoie une rubrique Amazon SNS avec un message pour savoir si la tâche a réussi ou échoué.

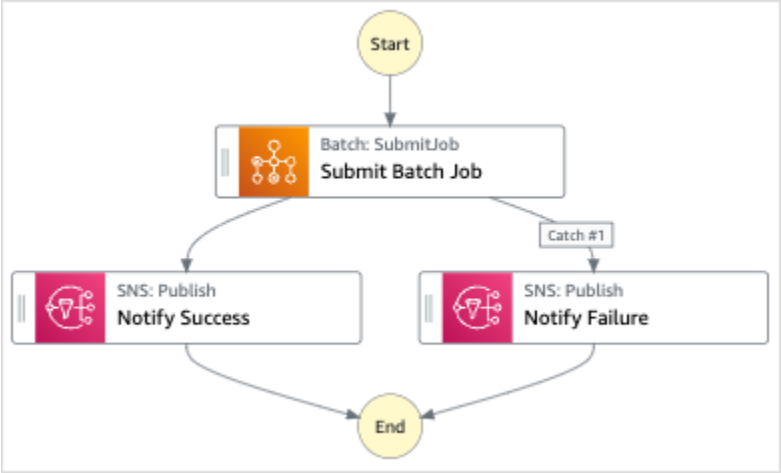
Étape 1 : créer la machine à états et provisionner les ressources

1. Ouvrez la console Step Functions et choisissez Create state machine.
2. Tapez **`Manage a batch job`** dans le champ de recherche, puis choisissez Gérer un traitement par lots dans les résultats de recherche renvoyés.
3. Choisissez Next (Suivant) pour continuer.
4. Step Functions répertorie les Services AWS éléments utilisés dans l'exemple de projet que vous avez sélectionné. Il montre également un graphique de flux de travail pour l'exemple de projet. Déployez ce projet sur votre site Compte AWS ou utilisez-le comme point de départ pour créer vos propres projets. Selon la façon dont vous souhaitez procéder, choisissez Exécuter une démo ou Construire à partir de celle-ci.

   Cet exemple de projet déploie les ressources suivantes :

   - Un AWS Batch travail
   - Une rubrique Amazon SNS
   - Une machine AWS Step Functions étatique
   - Rôles associés AWS Identity and Access Management (IAM)

   L'image suivante montre le graphique du flux de travail pour l'exemple de projet Gérer un travail par lots :

5. Choisissez Utiliser le modèle pour poursuivre votre sélection.
6. Effectuez l’une des actions suivantes :

- Si vous avez sélectionné Build on it, Step Functions crée le prototype de flux de travail pour l'exemple de projet que vous avez sélectionné. Step Functions ne déploie pas les ressources répertoriées dans la définition du flux de travail.

  Dans Workflow StudioMode de conception, glissez-déposez les états depuis le Navigateur d'états pour continuer à créer votre prototype de flux de travail. Vous pouvez également passer à un éditeur de code intégré similaire à VS Code pour mettre à jour la définition Amazon States Language (ASL) de votre machine à états dans la console Step Functions. Mode code Pour plus d'informations sur l'utilisation de Workflow Studio pour créer vos machines d'état, consultezUtilisation de Workflow Studio.

  > **Important**
  >
  > N'oubliez pas de mettre à jour l'espace réservé Amazon Resource Name (ARN) pour les ressources utilisées dans l'exemple de projet avant d'exécuter votre flux de travail.

- Si vous avez sélectionné Run a demo, Step Functions crée un exemple de projet en lecture seule qui utilise un AWS CloudFormation modèle pour déployer les AWS ressources répertoriées dans ce modèle sur votre. Compte AWS

  > **Tip**
  >
  > Pour afficher la définition de la machine à états de l'exemple de projet, choisissez Code.

  Lorsque vous êtes prêt, choisissez Déployer et exécuter pour déployer l'exemple de projet et créer les ressources.

  La création de ces ressources et des autorisations IAM associées peut prendre jusqu'à 10 minutes. Pendant le déploiement de vos ressources, vous pouvez ouvrir le lien CloudFormation Stack ID pour voir quelles ressources sont mises en service.

  Une fois que toutes les ressources de l'exemple de projet ont été créées, vous pouvez voir le nouvel exemple de projet répertorié sur la page State machines.

> **Important**
>
> Des frais standard peuvent s'appliquer pour chaque service utilisé dans le CloudFormation modèle.

Étape 2 : Exécuter la machine à états

1. Sur la page State machines, choisissez votre exemple de projet.
2. Sur la page d'exemple de projet, choisissez Démarrer l'exécution.
3. Dans la boîte de dialogue Démarrer l'exécution, procédez comme suit :

    1. (Facultatif) Pour identifier votre exécution, vous pouvez lui donner un nom dans le champ Nom. Par défaut, Step Functions génère automatiquement un nom d'exécution unique.

        > **Note**
        >
        > Step Functions vous permet de créer des noms pour les machines d'état, les exécutions et les activités, ainsi que des étiquettes contenant des caractères non ASCII. Ces noms non ASCII ne fonctionnent pas avec Amazon. CloudWatch Pour être sûr de pouvoir suivre CloudWatch les métriques, choisissez un nom qui utilise uniquement des caractères ASCII.

    2. (Facultatif) Dans la zone de saisie, entrez les valeurs d'entrée au format JSON pour exécuter votre flux de travail.

        Si vous avez choisi d'exécuter une démo, vous n'avez pas besoin de fournir d'entrée d'exécution.

        > **Note**
        >
        > Si le projet de démonstration que vous avez déployé contient des données d'entrée d'exécution préremplies, utilisez ces entrées pour exécuter la machine à états.

    3. Choisissez Start execution (Démarrer l'exécution).
    4. La console Step Functions vous dirige vers une page intitulée avec votre ID d'exécution. Cette page est connue sous le nom de page Détails de l'exécution. Sur cette page, vous pouvez

consulter les résultats de l'exécution au fur et à mesure que l'exécution progresse ou une fois celle-ci terminée.

Pour consulter les résultats de l'exécution, choisissez des états individuels dans la vue graphique, puis choisissez les onglets individuels du Détails de l'étape volet pour afficher les détails de chaque état, y compris les entrées, les sorties et la définition respectivement. Pour plus de détails sur les informations d'exécution que vous pouvez consulter sur la page Détails de l'exécution, voirPage de détails d'exécution — Vue d'ensemble de l'interface.

Exemple de code de machine d'état

Dans cet exemple de projet, la machine à états s'intègre AWS Batch à Amazon SNS en transmettant des paramètres directement à ces ressources.

Parcourez cet exemple de machine à états pour découvrir comment Step Functions contrôle AWS Batch Amazon SNS en vous connectant à l'Amazon Resource Name (ARN) `Resource` sur le terrain et en accédant `Parameters` à l'API du service.

Pour plus d'informations sur la manière de AWS Step Functions contrôler d'autres AWS services, consultezUtilisation AWS Step Functions avec d'autres services.

```
{
  "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
    }
  }
}
```

Exemple IAM

Cet exemple de politique AWS Identity and Access Management (IAM) généré par l'exemple de projet inclut le minimum de privilèges nécessaires pour exécuter la machine à états et les ressources associées. Nous vous recommandons de n'inclure que les autorisations nécessaires dans vos politiques IAM.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "sns:Publish"
            ],
            "Resource": [
```

```
                "arn:aws:sns:ap-northeast-1:123456789012:ManageBatchJob-SNSTopic-
JHLYYG7AZPZI"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "batch:SubmitJob",
                "batch:DescribeJobs",
                "batch:TerminateJob"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "events:PutTargets",
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": [
                "arn:aws:events:ap-northeast-1:123456789012:rule/
StepFunctionsGetEventsForBatchJobsRule"
            ],
            "Effect": "Allow"
        }
    ]
}
```

Pour plus d'informations sur la configuration de l'IAM lors de l'utilisation de Step Functions avec d'autres AWS services, consultezPolitiques IAM pour les services intégrés.

Gérer une tâche de conteneur (Amazon ECS,Amazon SNS)

Cet exemple de projet montre comment exécuter une AWS Fargate tâche, puis envoyer une Amazon SNS notification en fonction de la réussite ou de l'échec de cette tâche. Le déploiement de cet exemple de projet créera une machine à AWS Step Functions états, un Fargate cluster et une Amazon SNS rubrique.

Dans ce projet, Step Functions utilise une machine à états pour appeler la Fargate tâche de manière synchrone. Il attend ensuite que la tâche réussisse ou échoue, et il envoie une rubrique Amazon SNS avec un message pour savoir si la tâche a réussi ou échoué.

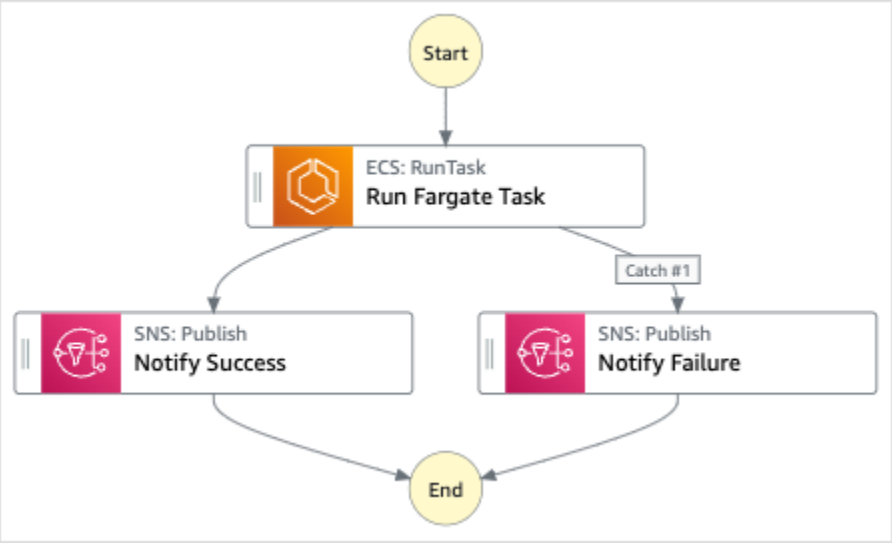
Étape 1 : créer la machine à états et provisionner les ressources

1. Ouvrez la console Step Functions et choisissez Create state machine.
2. Tapez **`Manage a container task`** dans la zone de recherche, puis choisissez Gérer une tâche de conteneur dans les résultats de recherche renvoyés.
3. Choisissez Next (Suivant) pour continuer.
4. Step Functions répertorie les Services AWS éléments utilisés dans l'exemple de projet que vous avez sélectionné. Il montre également un graphique de flux de travail pour l'exemple de projet. Déployez ce projet sur votre site Compte AWS ou utilisez-le comme point de départ pour créer vos propres projets. Selon la façon dont vous souhaitez procéder, choisissez Exécuter une démo ou Construire à partir de celle-ci.

   Cet exemple de projet déploie les ressources suivantes :

   - Un AWS Fargate cluster
   - Une rubrique Amazon SNS
   - Une machine AWS Step Functions étatique
   - Rôles associés AWS Identity and Access Management (IAM)

   L'image suivante montre le graphique du flux de travail pour le projet d'exemple de tâche Gérer un conteneur :



5. Choisissez Utiliser le modèle pour poursuivre votre sélection.
6. Effectuez l’une des actions suivantes :

- Si vous avez sélectionné Build on it, Step Functions crée le prototype de flux de travail pour l'exemple de projet que vous avez sélectionné. Step Functions ne déploie pas les ressources répertoriées dans la définition du flux de travail.

  Dans Workflow StudioMode de conception, glissez-déposez les états depuis le Navigateur d'états pour continuer à créer votre prototype de flux de travail. Vous pouvez également passer à un éditeur de code intégré similaire à VS Code pour mettre à jour la définition Amazon States Language (ASL) de votre machine à états dans la console Step Functions. Mode code Pour plus d'informations sur l'utilisation de Workflow Studio pour créer vos machines d'état, consultezUtilisation de Workflow Studio.

  > **Important**
  >
  > N'oubliez pas de mettre à jour l'espace réservé Amazon Resource Name (ARN) pour les ressources utilisées dans l'exemple de projet avant d'exécuter votre flux de travail.

- Si vous avez sélectionné Run a demo, Step Functions crée un exemple de projet en lecture seule qui utilise un AWS CloudFormation modèle pour déployer les AWS ressources répertoriées dans ce modèle sur votre. Compte AWS

  > **Tip**
  >
  > Pour afficher la définition de la machine à états de l'exemple de projet, choisissez Code.

  Lorsque vous êtes prêt, choisissez Déployer et exécuter pour déployer l'exemple de projet et créer les ressources.

  La création de ces ressources et des autorisations IAM associées peut prendre jusqu'à 10 minutes. Pendant le déploiement de vos ressources, vous pouvez ouvrir le lien CloudFormation Stack ID pour voir quelles ressources sont mises en service.

  Une fois que toutes les ressources de l'exemple de projet ont été créées, vous pouvez voir le nouvel exemple de projet répertorié sur la page State machines.

> **Important**
>
> Des frais standard peuvent s'appliquer pour chaque service utilisé dans le CloudFormation modèle.

Étape 2 : Exécuter la machine d'état

1. Sur la page State machines, choisissez votre exemple de projet.
2. Sur la page d'exemple de projet, choisissez Démarrer l'exécution.
3. Dans la boîte de dialogue Démarrer l'exécution, procédez comme suit :

    1. (Facultatif) Pour identifier votre exécution, vous pouvez lui donner un nom dans le champ Nom. Par défaut, Step Functions génère automatiquement un nom d'exécution unique.

        > **Note**
        >
        > Step Functions vous permet de créer des noms pour les machines d'état, les exécutions et les activités, ainsi que des étiquettes contenant des caractères non ASCII. Ces noms non ASCII ne fonctionnent pas avec Amazon. CloudWatch Pour être sûr de pouvoir suivre CloudWatch les métriques, choisissez un nom qui utilise uniquement des caractères ASCII.

    2. (Facultatif) Dans la zone de saisie, entrez les valeurs d'entrée au format JSON pour exécuter votre flux de travail.

        Si vous avez choisi d'exécuter une démo, vous n'avez pas besoin de fournir d'entrée d'exécution.

        > **Note**
        >
        > Si le projet de démonstration que vous avez déployé contient des données d'entrée d'exécution préremplies, utilisez ces entrées pour exécuter la machine à états.

    3. Choisissez Start execution (Démarrer l'exécution).
    4. La console Step Functions vous dirige vers une page intitulée avec votre ID d'exécution. Cette page est connue sous le nom de page Détails de l'exécution. Sur cette page, vous pouvez

consulter les résultats de l'exécution au fur et à mesure que l'exécution progresse ou une fois celle-ci terminée.

Pour consulter les résultats de l'exécution, choisissez des états individuels dans la vue graphique, puis choisissez les onglets individuels du Détails de l'étape volet pour afficher les détails de chaque état, y compris les entrées, les sorties et la définition respectivement. Pour plus de détails sur les informations d'exécution que vous pouvez consulter sur la page Détails de l'exécution, voirPage de détails d'exécution — Vue d'ensemble de l'interface.

Exemple de code de machine d'état

Dans cet exemple de projet, la machine à états s'intègre AWS Fargate à Amazon SNS en transmettant des paramètres directement à ces ressources. Parcourez cet exemple de machine à états pour découvrir comment Step Functions utilise une machine à états pour appeler la tâche Fargate de manière synchrone, attend que la tâche réussisse ou échoue et envoie une rubrique Amazon SNS avec un message indiquant si la tâche a réussi ou échoué.

Pour plus d'informations sur la manière de AWS Step Functions contrôler d'autres AWS services, consultezUtilisation AWS Step Functions avec d'autres services.

```
{
  "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
    }
  }
}
```

Exemple IAM

Cet exemple de politique AWS Identity and Access Management (IAM) généré par l'exemple de projet inclut le minimum de privilèges nécessaires pour exécuter la machine à états et les ressources associées. Il est recommandé d'inclure uniquement les autorisations nécessaires dans vos politiques IAM.

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "sns:Publish"
            ],
            "Resource": [
                "arn:aws:sns:ap-northeast-1:123456789012:FargateTaskNotification-
SNSTopic-1XYW5YD5V0M7C"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "ecs:RunTask"
            ],
            "Resource": [
                "arn:aws:ecs:ap-northeast-1:123456789012:task-definition/
FargateTaskNotification-ECSTaskDefinition-13YOJT8Z2LY5Q:1"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "ecs:StopTask",
                "ecs:DescribeTasks"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "events:PutTargets",
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": [
                "arn:aws:events:ap-northeast-1:123456789012:rule/
StepFunctionsGetEventsForECSTaskRule"
            ],
            "Effect": "Allow"
        }
    ]
```

```
}
```

Pour plus d'informations sur la configuration de l'IAM lors de l'utilisation de Step Functions avec d'autres AWS services, consultezPolitiques IAM pour les services intégrés.

Transférer des enregistrements de données (Lambda,DynamoDB,Amazon SQS)

Cet exemple de projet montre comment lire de manière itérative des éléments d'une Amazon DynamoDB table et les envoyer vers une Amazon SQS file d'attente à l'aide d'une machine à Step Functions états. Le déploiement de cet exemple de projet créera une machine à Step Functions états, une DynamoDB table, une AWS Lambda fonction et une Amazon SQS file d'attente.

Dans ce projet, Step Functions utilise la Lambda fonction pour remplir le DynamoDB tableau. La machine d'état utilise également une `for` boucle pour lire chacune des entrées, puis envoie chaque entrée dans une Amazon SQS file d'attente.

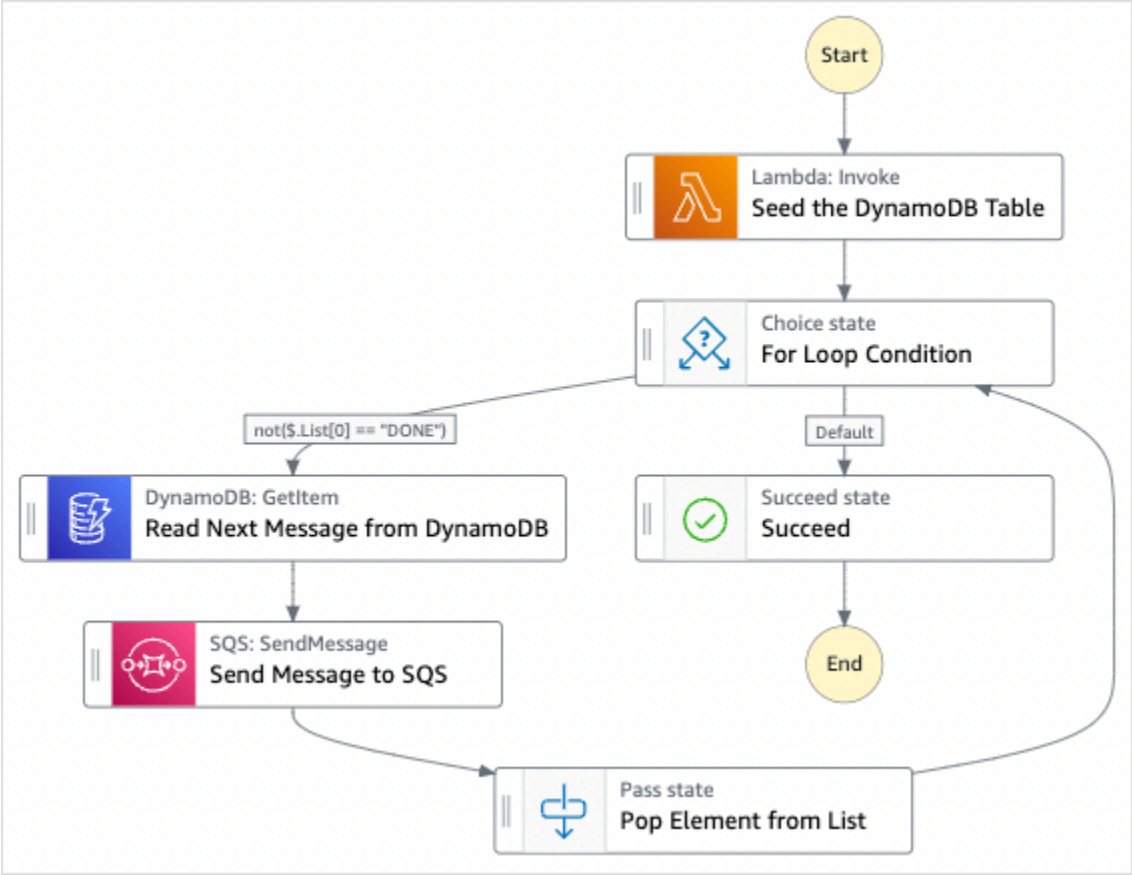
Étape 1 : créer la machine à états et provisionner les ressources

1. Ouvrez la console Step Functions et choisissez Create state machine.
2. Tapez **`Transfer data records`** dans la zone de recherche, puis choisissez Transférer les enregistrements de données à partir des résultats de recherche renvoyés.
3. Choisissez Next (Suivant) pour continuer.
4. Step Functions répertorie les Services AWS éléments utilisés dans l'exemple de projet que vous avez sélectionné. Il montre également un graphique de flux de travail pour l'exemple de projet. Déployez ce projet sur votre site Compte AWS ou utilisez-le comme point de départ pour créer vos propres projets. Selon la façon dont vous souhaitez procéder, choisissez Exécuter une démo ou Construire à partir de celle-ci.

   Cet exemple de projet déploie les ressources suivantes :

   - Une fonction Lambda pour l'amorçage de la table DynamoDB
   - Une file d'attente Amazon SQS
   - Table DynamoDB
   - Une machine AWS Step Functions étatique
   - Rôles associés AWS Identity and Access Management (IAM)

L'image suivante montre le graphique du flux de travail pour le projet d'exemple de transfert d'enregistrements de données :



5. Choisissez Utiliser le modèle pour poursuivre votre sélection.
6. Effectuez l'une des actions suivantes :

   - Si vous avez sélectionné Build on it, Step Functions crée le prototype de flux de travail pour l'exemple de projet que vous avez sélectionné. Step Functions ne déploie pas les ressources répertoriées dans la définition du flux de travail.

     Dans Workflow StudioMode de conception, glissez-déposez les états depuis le Navigateur d'états pour continuer à créer votre prototype de flux de travail. Vous pouvez également passer à un éditeur de code intégré similaire à VS Code pour mettre à jour la définition Amazon States Language (ASL) de votre machine à états dans la console Step Functions. Mode code Pour plus d'informations sur l'utilisation de Workflow Studio pour créer vos machines d'état, consultezUtilisation de Workflow Studio.

> **Important**
>
> N'oubliez pas de mettre à jour l'espace réservé Amazon Resource Name (ARN) pour les ressources utilisées dans l'exemple de projet avant d'exécuter votre flux de travail.

- Si vous avez sélectionné Run a demo, Step Functions crée un exemple de projet en lecture seule qui utilise un AWS CloudFormation modèle pour déployer les AWS ressources répertoriées dans ce modèle sur votre. Compte AWS

  > **Tip**
  >
  > Pour afficher la définition de la machine à états de l'exemple de projet, choisissez Code.

  Lorsque vous êtes prêt, choisissez Déployer et exécuter pour déployer l'exemple de projet et créer les ressources.

  La création de ces ressources et des autorisations IAM associées peut prendre jusqu'à 10 minutes. Pendant le déploiement de vos ressources, vous pouvez ouvrir le lien CloudFormation Stack ID pour voir quelles ressources sont mises en service.

  Une fois que toutes les ressources de l'exemple de projet ont été créées, vous pouvez voir le nouvel exemple de projet répertorié sur la page State machines.

  > **Important**
  >
  > Des frais standard peuvent s'appliquer pour chaque service utilisé dans le CloudFormation modèle.

Étape 2 : Exécuter la machine à états

1. Sur la page State machines, choisissez votre exemple de projet.
2. Sur la page d'exemple de projet, choisissez Démarrer l'exécution.
3. Dans la boîte de dialogue Démarrer l'exécution, procédez comme suit :

1. (Facultatif) Pour identifier votre exécution, vous pouvez lui donner un nom dans le champ Nom. Par défaut, Step Functions génère automatiquement un nom d'exécution unique.

   > **Note**
   >
   > Step Functions vous permet de créer des noms pour les machines d'état, les exécutions et les activités, ainsi que des étiquettes contenant des caractères non ASCII. Ces noms non ASCII ne fonctionnent pas avec Amazon. CloudWatch Pour être sûr de pouvoir suivre CloudWatch les métriques, choisissez un nom qui utilise uniquement des caractères ASCII.

2. (Facultatif) Dans la zone de saisie, entrez les valeurs d'entrée au format JSON pour exécuter votre flux de travail.

   Si vous avez choisi d'exécuter une démo, vous n'avez pas besoin de fournir d'entrée d'exécution.

   > **Note**
   >
   > Si le projet de démonstration que vous avez déployé contient des données d'entrée d'exécution préremplies, utilisez ces entrées pour exécuter la machine à états.

3. Choisissez Start execution (Démarrer l'exécution).

4. La console Step Functions vous dirige vers une page intitulée avec votre ID d'exécution. Cette page est connue sous le nom de page Détails de l'exécution. Sur cette page, vous pouvez consulter les résultats de l'exécution au fur et à mesure que l'exécution progresse ou une fois celle-ci terminée.

   Pour consulter les résultats de l'exécution, choisissez des états individuels dans la vue graphique, puis choisissez les onglets individuels du Détails de l'étape volet pour afficher les détails de chaque état, y compris les entrées, les sorties et la définition respectivement. Pour plus de détails sur les informations d'exécution que vous pouvez consulter sur la page Détails de l'exécution, voirPage de détails d'exécution — Vue d'ensemble de l'interface.

Exemple de code de machine d'état

La machine à états de cet exemple de projet s'intègre à DynamoDB et Amazon SQS en transmettant des paramètres directement à ces ressources.

Parcourez cet exemple de machine à états pour découvrir comment Step Functions contrôle DynamoDB et Amazon SQS en se connectant à l'Amazon Resource Name (ARN) sur le terrain et en accédant à `Resource` l'API du service. `Parameters`

Pour plus d'informations sur la manière de AWS Step Functions contrôler d'autres AWS services, consultezUtilisation AWS Step Functions avec d'autres services.

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

Pour plus d'informations sur le passage de paramètres et la gestion des résultats, consultez ce qui suit :

- Transmettre des paramètres à une API de service
- ResultPath

Exemple IAM

Cet exemple de politique AWS Identity and Access Management (IAM) généré par l'exemple de projet inclut le minimum de privilèges nécessaires pour exécuter la machine à états et les ressources

associées. Il est recommandé d'inclure uniquement les autorisations nécessaires dans vos politiques IAM.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "dynamodb:GetItem"
            ],
            "Resource": [
                "arn:aws:dynamodb:ap-northeast-1:123456789012:table/
TransferDataRecords-DDBTable-3I41R5L5EAGT"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "sqs:SendMessage"
            ],
            "Resource": [
                "arn:aws:sqs:ap-northeast-1:123456789012:TransferDataRecords-SQSQueue-
BKWXTS09LIW1"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "lambda:invokeFunction"
            ],
            "Resource": [
                "arn:aws:lambda:ap-
northeast-1:123456789012:function:TransferDataRecords-SeedingFunction-VN4KY2TPAZSR"
            ],
            "Effect": "Allow"
        }
    ]
}
```

Pour plus d'informations sur la configuration de l'IAM lors de l'utilisation de Step Functions avec d'autres AWS services, consultezPolitiques IAM pour les services intégrés.

Sondage pour connaître le statut du poste (Lambda,) AWS Batch

Cet exemple de projet crée un sondage d' AWS Batch offres d'emploi. Il implémente une machine AWS Lambda à AWS Step Functions états qui permet de créer une boucle d'`Wait`état qui vérifie une AWS Batch tâche.

Cet exemple de projet crée et configure toutes les ressources afin que votre flux de travail Step Functions soumette une AWS Batch tâche et attende que cette tâche soit terminée avant de se terminer correctement.

> **Note**
>
> Vous pouvez également implémenter ce modèle sans utiliser de fonction Lambda. Pour plus d'informations sur le contrôle AWS Batch direct, consultezUtilisation AWS Step Functions avec d'autres services.

Cet exemple de projet crée la machine à états, deux fonctions Lambda et une AWS Batch file d'attente, et configure les autorisations IAM associées.

Pour plus d'informations sur la manière de AWS Step Functions contrôler d'autres AWS services, consultezUtilisation AWS Step Functions avec d'autres services.

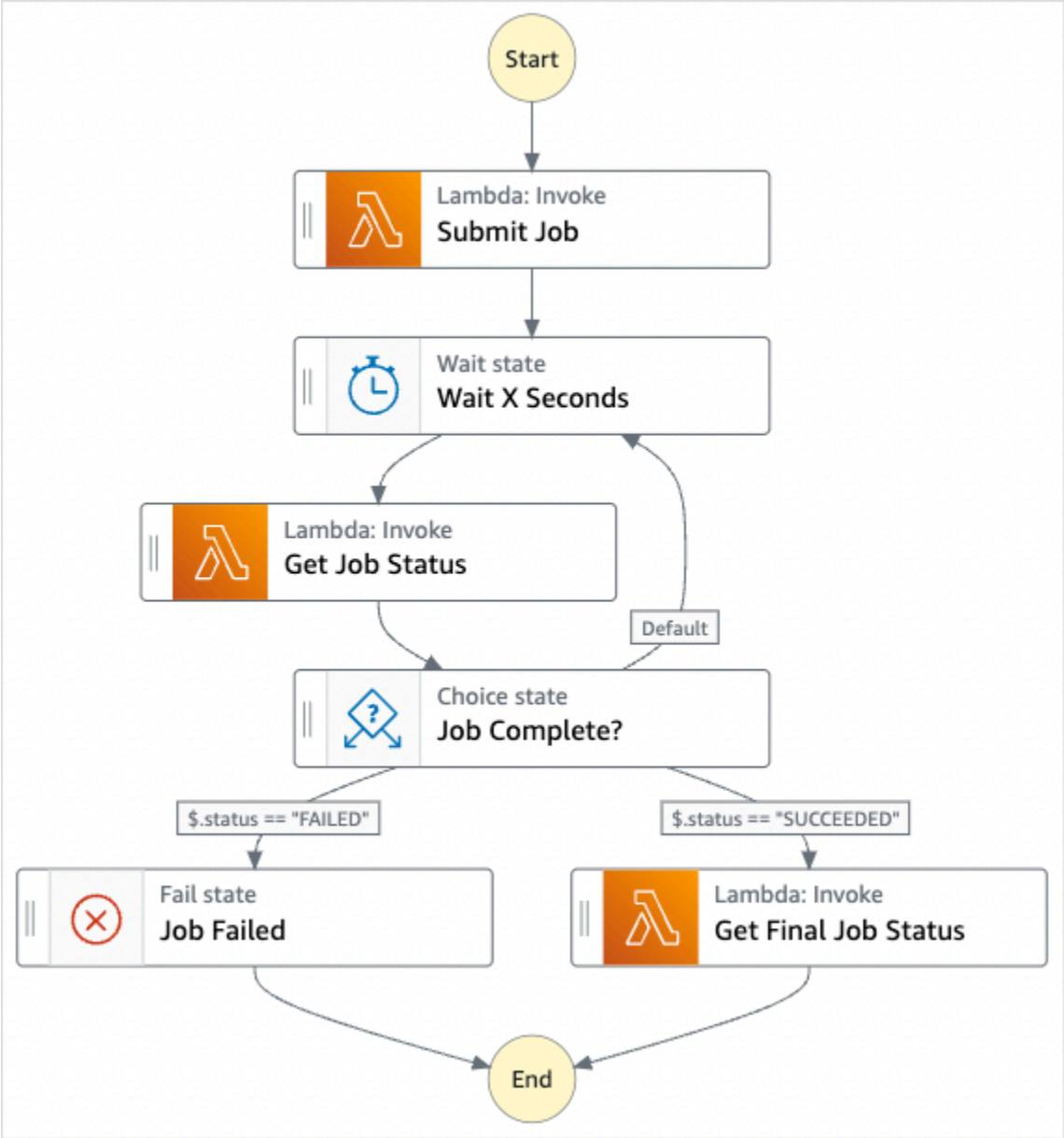
Étape 1 : créer la machine à états et provisionner les ressources

1. Ouvrez la console Step Functions et choisissez Create state machine.
2. Tapez **`Job Poller`** dans le champ de recherche, puis choisissez Job Poller dans les résultats de recherche renvoyés.
3. Choisissez Next (Suivant) pour continuer.
4. Step Functions répertorie les Services AWS éléments utilisés dans l'exemple de projet que vous avez sélectionné. Il montre également un graphique de flux de travail pour l'exemple de projet. Déployez ce projet sur votre site Compte AWS ou utilisez-le comme point de départ pour créer vos propres projets. Selon la façon dont vous souhaitez procéder, choisissez Exécuter une démo ou Construire à partir de celle-ci.

   Cet exemple de projet déploie les ressources suivantes :

- Trois fonctions Lambda permettant de soumettre une AWS Batch tâche, d'obtenir le statut actuel de la AWS Batch tâche soumise et l'état d'achèvement de la tâche finale.
- Un AWS Batch travail
- Une machine AWS Step Functions étatique
- Rôles associés AWS Identity and Access Management (IAM)

L'image suivante montre le graphique du flux de travail de l'exemple de projet Job Poller :



5. Choisissez Utiliser le modèle pour poursuivre votre sélection.
6. Effectuez l’une des actions suivantes :

- Si vous avez sélectionné Build on it, Step Functions crée le prototype de flux de travail pour l'exemple de projet que vous avez sélectionné. Step Functions ne déploie pas les ressources répertoriées dans la définition du flux de travail.

  Dans Workflow StudioMode de conception, glissez-déposez les états depuis le Navigateur d'états pour continuer à créer votre prototype de flux de travail. Vous pouvez également passer à un éditeur de code intégré similaire à VS Code pour mettre à jour la définition Amazon States Language (ASL) de votre machine à états dans la console Step Functions. Mode code Pour plus d'informations sur l'utilisation de Workflow Studio pour créer vos machines d'état, consultezUtilisation de Workflow Studio.

  > **Important**
  >
  > N'oubliez pas de mettre à jour l'espace réservé Amazon Resource Name (ARN) pour les ressources utilisées dans l'exemple de projet avant d'exécuter votre flux de travail.

- Si vous avez sélectionné Run a demo, Step Functions crée un exemple de projet en lecture seule qui utilise un AWS CloudFormation modèle pour déployer les AWS ressources répertoriées dans ce modèle sur votre. Compte AWS

  > **Tip**
  >
  > Pour afficher la définition de la machine à états de l'exemple de projet, choisissez Code.

  Lorsque vous êtes prêt, choisissez Déployer et exécuter pour déployer l'exemple de projet et créer les ressources.

  La création de ces ressources et des autorisations IAM associées peut prendre jusqu'à 10 minutes. Pendant le déploiement de vos ressources, vous pouvez ouvrir le lien CloudFormation Stack ID pour voir quelles ressources sont mises en service.

  Une fois que toutes les ressources de l'exemple de projet ont été créées, vous pouvez voir le nouvel exemple de projet répertorié sur la page State machines.

> **Important**
>
> Des frais standard peuvent s'appliquer pour chaque service utilisé dans le CloudFormation modèle.

Étape 2 : Exécuter la machine à états

Une fois que toutes les ressources ont été provisionnées et déployées, la boîte de dialogue Démarrer l'exécution s'affiche avec un exemple de saisie similaire à celui ci-dessous.

```
{
    "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` ordonne à l'état `Wait` de faire une boucle toutes les soixante secondes.

- Dans la boîte de dialogue Démarrer l'exécution, procédez comme suit :

  1. (Facultatif) Pour identifier votre exécution, vous pouvez lui donner un nom dans le champ Nom. Par défaut, Step Functions génère automatiquement un nom d'exécution unique.

     > **Note**
     >
     > Step Functions vous permet de créer des noms pour les machines d'état, les exécutions et les activités, ainsi que des étiquettes contenant des caractères non ASCII. Ces noms non ASCII ne fonctionnent pas avec Amazon. CloudWatch Pour être sûr de pouvoir suivre CloudWatch les métriques, choisissez un nom qui utilise uniquement des caractères ASCII.

2. (Facultatif) Dans la zone de saisie, entrez les valeurs d'entrée au format JSON pour exécuter votre flux de travail.

   Si vous avez choisi d'exécuter une démo, vous n'avez pas besoin de fournir d'entrée d'exécution.

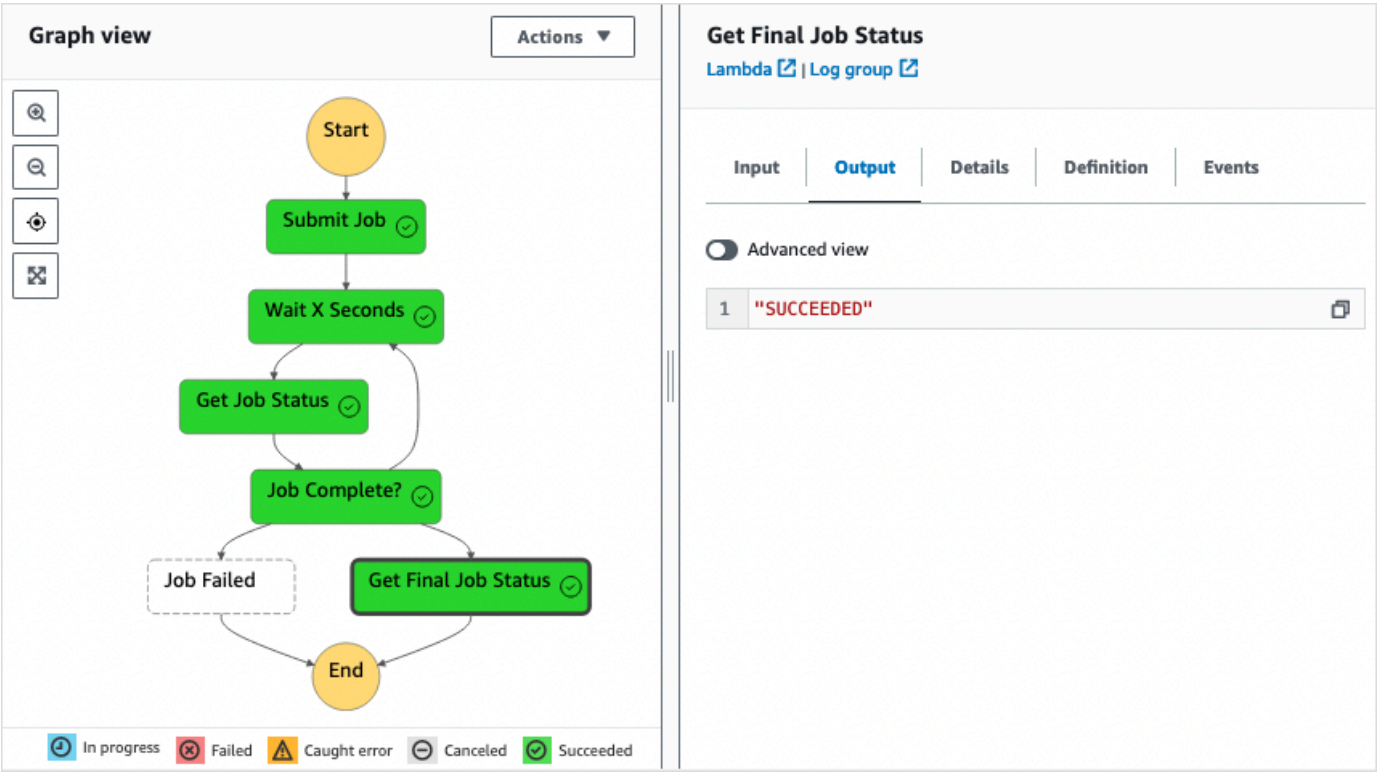
   > **Note**
   >
   > Si le projet de démonstration que vous avez déployé contient des données d'entrée d'exécution préremplies, utilisez ces entrées pour exécuter la machine à états.

3. Choisissez Start execution (Démarrer l'exécution).
4. La console Step Functions vous dirige vers une page intitulée avec votre ID d'exécution. Cette page est connue sous le nom de page Détails de l'exécution. Sur cette page, vous pouvez consulter les résultats de l'exécution au fur et à mesure que l'exécution progresse ou une fois celle-ci terminée.

   Pour consulter les résultats de l'exécution, choisissez des états individuels dans la vue graphique, puis choisissez les onglets individuels du Détails de l'étape volet pour afficher les détails de chaque état, y compris les entrées, les sorties et la définition respectivement. Pour plus de détails sur les informations d'exécution que vous pouvez consulter sur la page Détails de l'exécution, voirPage de détails d'exécution — Vue d'ensemble de l'interface.

Par exemple, pour voir l'évolution du statut de votre AWS Batch tâche et les résultats en boucle de votre exécution, choisissez l'onglet Sortie.

L'image suivante montre le graphique de l'état d'exécution dans la vue graphique. Il affiche également le résultat d'exécution de l'étape sélectionnée dans l'onglet Sortie.



Exemple de code de machine d'état

Dans cet exemple de projet, la machine à états s'intègre AWS Lambda à pour soumettre une AWS Batch tâche. Parcourez cet exemple de machine à états pour découvrir comment Step Functions contrôle Lambda et. AWS Batch

Pour plus d'informations sur la manière de AWS Step Functions contrôler d'autres AWS services, consultezUtilisation AWS Step Functions avec d'autres services.

```
{
    "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
      }
```

```
    }
  }
```

Minuteur de tâches (Lambda, Amazon SNS)

Cet exemple de projet crée un temporisateur de tâche. Il implémente une AWS Step Functions machine à états qui implémente un `Wait` état et utilise une AWS Lambda fonction qui envoie une notification Amazon Simple Notification Service (Amazon SNS). Un état Attente est un type d'état qui attend un déclencheur pour effectuer une seule unité de travail.

> **Note**
>
> Cet exemple de projet implémente une AWS Lambda fonction permettant d'envoyer une notification Amazon Simple Notification Service (Amazon SNS). Vous pouvez également envoyer une notification Amazon SNS directement depuis l'Amazon States Language. veuillez consulter Utilisation AWS Step Functions avec d'autres services.

Cet exemple de projet crée la machine à états, une fonction Lambda et une rubrique Amazon SNS, et configure les autorisations (IAM) AWS Identity and Access Management associées. Pour plus d'informations sur les ressources créées avec l'exemple de projet Temporisateur de tâche, consultez :

Pour plus d'informations sur la manière de AWS Step Functions contrôler d'autres AWS services, consultezUtilisation AWS Step Functions avec d'autres services.

- AWS CloudFormation Guide de l'utilisateur
- Guide du développeur Amazon Simple Notification Service
- AWS Lambda Manuel du développeur
- Guide de démarrage IAM

Étape 1 : créer la machine à états et provisionner les ressources

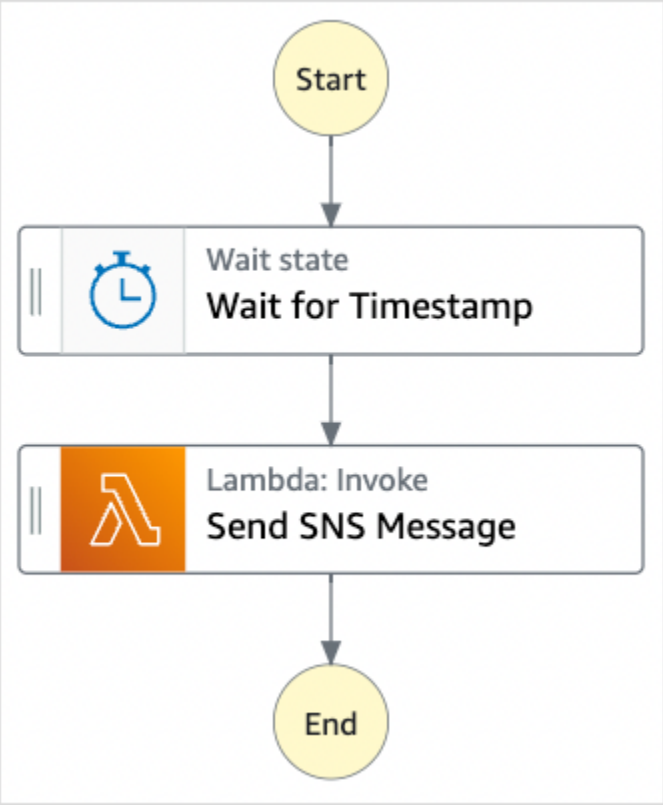
1. Ouvrez la console Step Functions et choisissez Create state machine.
2. Tapez **`Task Timer`** dans la zone de recherche, puis choisissez Task Timer dans les résultats de recherche renvoyés.
3. Choisissez Next (Suivant) pour continuer.

4. Step Functions répertorie les Services AWS éléments utilisés dans l'exemple de projet que vous avez sélectionné. Il montre également un graphique de flux de travail pour l'exemple de projet. Déployez ce projet sur votre site Compte AWS ou utilisez-le comme point de départ pour créer vos propres projets. Selon la façon dont vous souhaitez procéder, choisissez Exécuter une démo ou Construire à partir de celle-ci.

   Cet exemple de projet déploie les ressources suivantes :

   - une fonction Lambda qui envoie une notification Amazon SNS.
   - Une machine AWS Step Functions étatique
   - Rôles associés AWS Identity and Access Management (IAM)

   L'image suivante montre le graphique du flux de travail de l'exemple de projet Task Timer :



5. Choisissez Utiliser le modèle pour poursuivre votre sélection.
6. Effectuez l’une des actions suivantes :

   - Si vous avez sélectionné Build on it, Step Functions crée le prototype de flux de travail pour l'exemple de projet que vous avez sélectionné. Step Functions ne déploie pas les ressources répertoriées dans la définition du flux de travail.

Dans Workflow StudioMode de conception, glissez-déposez les états depuis le Navigateur d'états pour continuer à créer votre prototype de flux de travail. Vous pouvez également passer à un éditeur de code intégré similaire à VS Code pour mettre à jour la définition Amazon States Language (ASL) de votre machine à états dans la console Step Functions. Mode code Pour plus d'informations sur l'utilisation de Workflow Studio pour créer vos machines d'état, consultezUtilisation de Workflow Studio.

> **Important**
>
> N'oubliez pas de mettre à jour l'espace réservé Amazon Resource Name (ARN) pour les ressources utilisées dans l'exemple de projet avant d'exécuter votre flux de travail.

- Si vous avez sélectionné Run a demo, Step Functions crée un exemple de projet en lecture seule qui utilise un AWS CloudFormation modèle pour déployer les AWS ressources répertoriées dans ce modèle sur votre. Compte AWS

  > **Tip**
  >
  > Pour afficher la définition de la machine à états de l'exemple de projet, choisissez Code.

  Lorsque vous êtes prêt, choisissez Déployer et exécuter pour déployer l'exemple de projet et créer les ressources.

  La création de ces ressources et des autorisations IAM associées peut prendre jusqu'à 10 minutes. Pendant le déploiement de vos ressources, vous pouvez ouvrir le lien CloudFormation Stack ID pour voir quelles ressources sont mises en service.

  Une fois que toutes les ressources de l'exemple de projet ont été créées, vous pouvez voir le nouvel exemple de projet répertorié sur la page State machines.

  > **Important**
  >
  > Des frais standard peuvent s'appliquer pour chaque service utilisé dans le CloudFormation modèle.

Étape 2 : Exécuter la machine à états

Une fois que toutes les ressources ont été provisionnées et déployées, la boîte de dialogue Démarrer l'exécution s'affiche avec un exemple de saisie similaire à celui ci-dessous.

```
{
    "jobName": "my-job",{
  "topic": "arn:aws:sns:us-east-2:123456789012:StepFunctionsSample-TaskTimercc68840e-c3d3-42a8-911e-821b7ce248e5-SNSTopic-44UjcFxzhACT",
  "message": "HelloWorld",
  "timer_seconds": 10
}
```

- Dans la boîte de dialogue Démarrer l'exécution, procédez comme suit :

  1. (Facultatif) Pour identifier votre exécution, vous pouvez lui donner un nom dans le champ Nom. Par défaut, Step Functions génère automatiquement un nom d'exécution unique.

     > **Note**
     >
     > Step Functions vous permet de créer des noms pour les machines d'état, les exécutions et les activités, ainsi que des étiquettes contenant des caractères non ASCII. Ces noms non ASCII ne fonctionnent pas avec Amazon. CloudWatch Pour être sûr de pouvoir suivre CloudWatch les métriques, choisissez un nom qui utilise uniquement des caractères ASCII.

  2. (Facultatif) Dans la zone de saisie, entrez les valeurs d'entrée au format JSON pour exécuter votre flux de travail.

     Si vous avez choisi d'exécuter une démo, vous n'avez pas besoin de fournir d'entrée d'exécution.

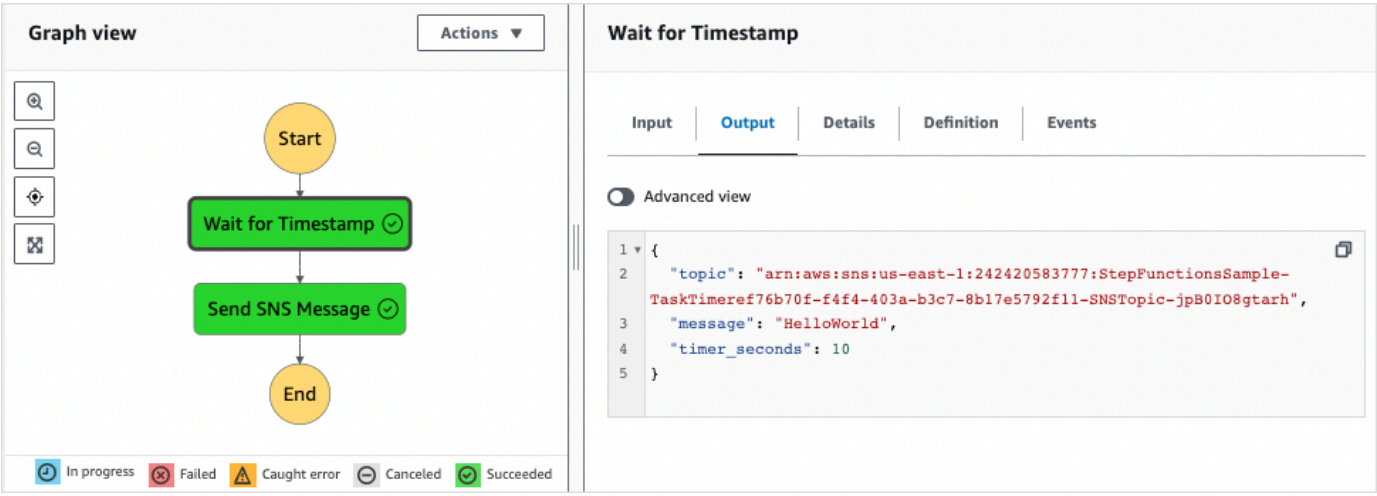
     > **Note**
     >
     > Si le projet de démonstration que vous avez déployé contient des données d'entrée d'exécution préremplies, utilisez ces entrées pour exécuter la machine à états.

  3. Choisissez Start execution (Démarrer l'exécution).

4. La console Step Functions vous dirige vers une page intitulée avec votre ID d'exécution. Cette page est connue sous le nom de page Détails de l'exécution. Sur cette page, vous pouvez consulter les résultats de l'exécution au fur et à mesure que l'exécution progresse ou une fois celle-ci terminée.

   Pour consulter les résultats de l'exécution, choisissez des états individuels dans la vue graphique, puis choisissez les onglets individuels du Détails de l'étape volet pour afficher les détails de chaque état, y compris les entrées, les sorties et la définition respectivement. Pour plus de détails sur les informations d'exécution que vous pouvez consulter sur la page Détails de l'exécution, voirPage de détails d'exécution — Vue d'ensemble de l'interface.

Par exemple, l'image suivante montre le résultat de l'étape sélectionnée Attendre l'horodatage. Le résultat de cette étape est transmis en entrée à l'étape Envoyer un message SNS.



Exemple de modèle de rappel (Amazon SQS, Amazon SNS, Lambda)

Cet exemple de projet montre comment faire une AWS Step Functions pause pendant une tâche et attendre qu'un processus externe renvoie un jeton de tâche généré au démarrage de la tâche.

Lorsque cet exemple de projet est déployé et qu'une exécution est démarrée, les étapes suivantes se produisent.

1. Step Functions transmet un message contenant un jeton de tâche à une file d'attente Amazon Simple Queue Service (Amazon SQS).

2. Step Functions fait ensuite une pause en attendant que ce jeton soit renvoyé.
3. La file d'attente Amazon SQS déclenche une AWS Lambda fonction qui appelle `SendTaskSuccess`avec le même jeton de tâche.
4. Lorsque le jeton de la tâche est reçu, le flux de travail se poursuit.
5. La `"Notify Success"` tâche publie un message Amazon Simple Notification Service (Amazon SNS) indiquant que le rappel a été reçu.

Pour savoir comment implémenter le modèle de rappel dans Step Functions, consultezAttendre un rappel avec le jeton de tâche.

Pour plus d'informations sur la manière de AWS Step Functions contrôler d'autres AWS services, consultezUtilisation AWS Step Functions avec d'autres services.

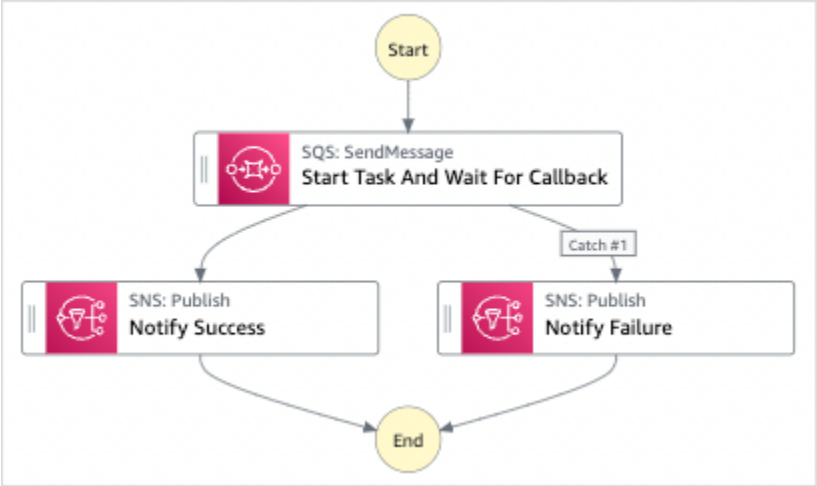
Étape 1 : créer la machine à états et provisionner les ressources

1. Ouvrez la console Step Functions et choisissez Create state machine.
2. Tapez **`Callback pattern example`** dans la zone de recherche, puis choisissez Exemple de modèle de rappel dans les résultats de recherche renvoyés.
3. Choisissez Next (Suivant) pour continuer.
4. Step Functions répertorie les Services AWS éléments utilisés dans l'exemple de projet que vous avez sélectionné. Il montre également un graphique de flux de travail pour l'exemple de projet. Déployez ce projet sur votre site Compte AWS ou utilisez-le comme point de départ pour créer vos propres projets. Selon la façon dont vous souhaitez procéder, choisissez Exécuter une démo ou Construire à partir de celle-ci.

   Cet exemple de projet déploie les ressources suivantes :

   - Une file d'attente de messages Amazon SQS.
   - Fonction Lambda qui appelle l'action d'API Step Functions. SendTaskSuccess
   - Rubrique Amazon SNS signalant le succès ou l'échec d'une tâche et indiquant si le flux de travail peut se poursuivre ou non.
   - Une machine AWS Step Functions étatique
   - Rôles associés AWS Identity and Access Management (IAM)

L'image suivante montre le graphique du flux de travail pour l'exemple de projet d'exemple de modèle de rappel :



5. Choisissez Utiliser le modèle pour poursuivre votre sélection.
6. Effectuez l'une des actions suivantes :

   - Si vous avez sélectionné Build on it, Step Functions crée le prototype de flux de travail pour l'exemple de projet que vous avez sélectionné. Step Functions ne déploie pas les ressources répertoriées dans la définition du flux de travail.

     Dans Workflow StudioMode de conception, glissez-déposez les états depuis le Navigateur d'états pour continuer à créer votre prototype de flux de travail. Vous pouvez également passer à un éditeur de code intégré similaire à VS Code pour mettre à jour la définition Amazon States Language (ASL) de votre machine à états dans la console Step Functions. Mode code Pour plus d'informations sur l'utilisation de Workflow Studio pour créer vos machines d'état, consultezUtilisation de Workflow Studio.

     > **Important**
     >
     > N'oubliez pas de mettre à jour l'espace réservé Amazon Resource Name (ARN) pour les ressources utilisées dans l'exemple de projet avant d'exécuter votre flux de travail.

   - Si vous avez sélectionné Run a demo, Step Functions crée un exemple de projet en lecture seule qui utilise un AWS CloudFormation modèle pour déployer les AWS ressources répertoriées dans ce modèle sur votre. Compte AWS

> **Tip**
>
> Pour afficher la définition de la machine à états de l'exemple de projet, choisissez Code.

Lorsque vous êtes prêt, choisissez Déployer et exécuter pour déployer l'exemple de projet et créer les ressources.

La création de ces ressources et des autorisations IAM associées peut prendre jusqu'à 10 minutes. Pendant le déploiement de vos ressources, vous pouvez ouvrir le lien CloudFormation Stack ID pour voir quelles ressources sont mises en service.

Une fois que toutes les ressources de l'exemple de projet ont été créées, vous pouvez voir le nouvel exemple de projet répertorié sur la page State machines.

> **Important**
>
> Des frais standard peuvent s'appliquer pour chaque service utilisé dans le CloudFormation modèle.

Étape 2 : Exécuter la machine à états

1. Sur la page State machines, choisissez votre exemple de projet.
2. Sur la page d'exemple de projet, choisissez Démarrer l'exécution.
3. Dans la boîte de dialogue Démarrer l'exécution, procédez comme suit :

    1. (Facultatif) Pour identifier votre exécution, vous pouvez lui donner un nom dans le champ Nom. Par défaut, Step Functions génère automatiquement un nom d'exécution unique.

        > **Note**
        >
        > Step Functions vous permet de créer des noms pour les machines d'état, les exécutions et les activités, ainsi que des étiquettes contenant des caractères non ASCII. Ces noms non ASCII ne fonctionnent pas avec Amazon. CloudWatch Pour

être sûr de pouvoir suivre CloudWatch les métriques, choisissez un nom qui utilise uniquement des caractères ASCII.

2. (Facultatif) Dans la zone de saisie, entrez les valeurs d'entrée au format JSON pour exécuter votre flux de travail.

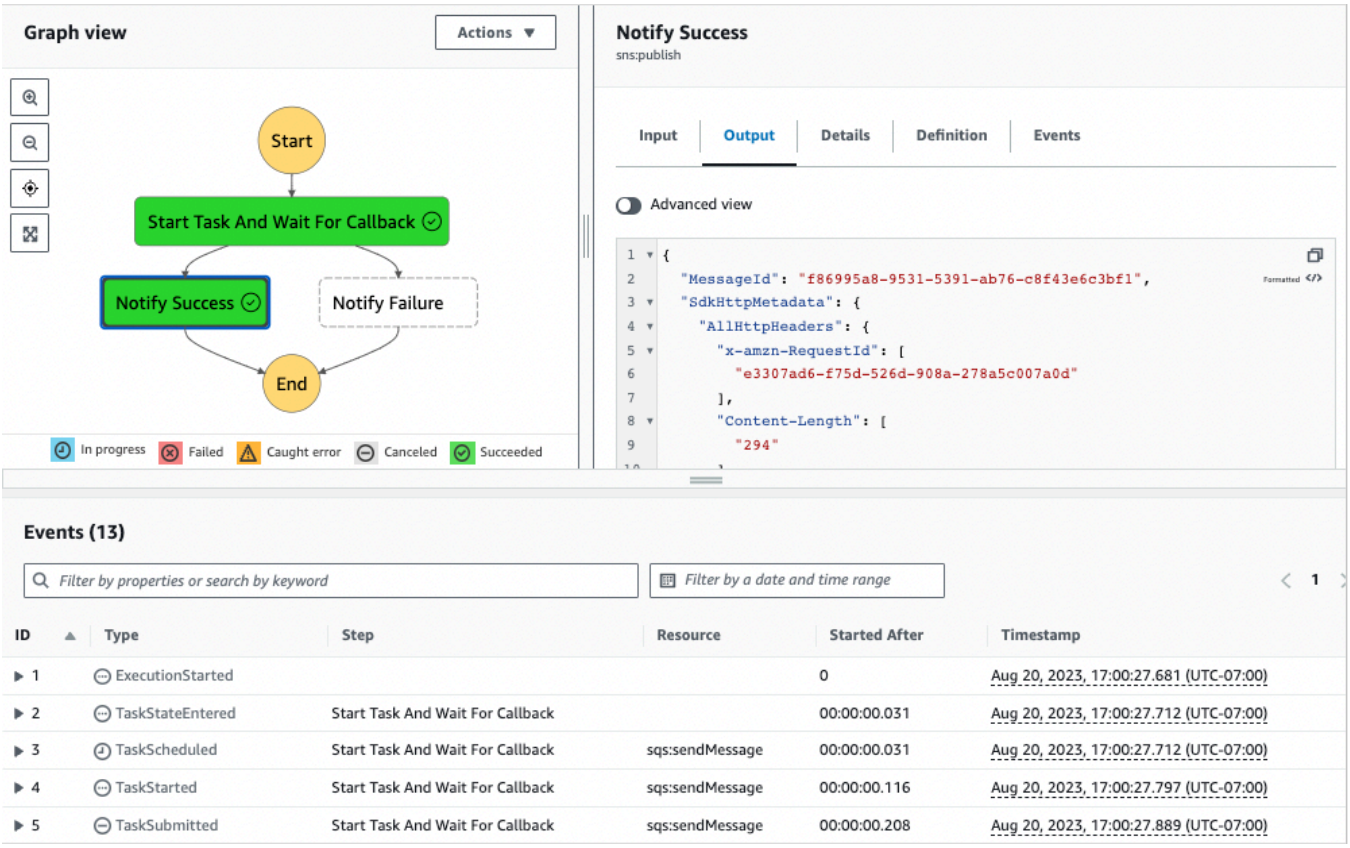
   Si vous avez choisi d'exécuter une démo, vous n'avez pas besoin de fournir d'entrée d'exécution.

   > **Note**
   >
   > Si le projet de démonstration que vous avez déployé contient des données d'entrée d'exécution préremplies, utilisez ces entrées pour exécuter la machine à états.

3. Choisissez Start execution (Démarrer l'exécution).
4. La console Step Functions vous dirige vers une page intitulée avec votre ID d'exécution. Cette page est connue sous le nom de page Détails de l'exécution. Sur cette page, vous pouvez consulter les résultats de l'exécution au fur et à mesure que l'exécution progresse ou une fois celle-ci terminée.

   Pour consulter les résultats de l'exécution, choisissez des états individuels dans la vue graphique, puis choisissez les onglets individuels du Détails de l'étape volet pour afficher les détails de chaque état, y compris les entrées, les sorties et la définition respectivement. Pour plus de détails sur les informations d'exécution que vous pouvez consulter sur la page Détails de l'exécution, voirPage de détails d'exécution — Vue d'ensemble de l'interface.

Par exemple, pour voir comment Step Functions a progressé dans le flux de travail et a reçu un rappel d'Amazon SQS, consultez les entrées du tableau Events. L'image suivante montre le résultat d'exécution de l'étape Notify Success. Il affiche également les cinq premiers événements de l'historique des événements d'exécution. Développez chaque événement pour afficher plus de détails sur cet événement.



Exemple de rappel Lambda

Pour voir comment les composants de cet exemple de projet fonctionnent ensemble, consultez les ressources qui ont été déployées dans votre AWS compte. Par exemple, voici la fonction Lambda qui appelle Step Functions avec le jeton de tâche.

```
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);
        });
    }
};
```

Gérer une offre d'emploi Amazon EMR

Cet exemple de projet illustre Amazon EMR et AWS Step Functions son intégration.

Il explique comment créer un cluster Amazon EMR, ajouter plusieurs étapes et les exécuter, puis mettre fin au cluster.

> **Important**
>
> Amazon EMR ne propose pas de niveau de tarification gratuit. L'exécution de l'exemple de projet entraînera des coûts. Vous trouverez des informations sur les prix sur la page de tarification d'Amazon EMR. La disponibilité de l'intégration du service Amazon EMR dépend de la disponibilité des API Amazon EMR. De ce fait, cet exemple de projet risque de ne pas fonctionner correctement dans certaines AWS régions. Consultez la documentation Amazon EMR pour connaître les limites applicables à certaines régions.

Étape 1 : créer la machine à états et provisionner les ressources

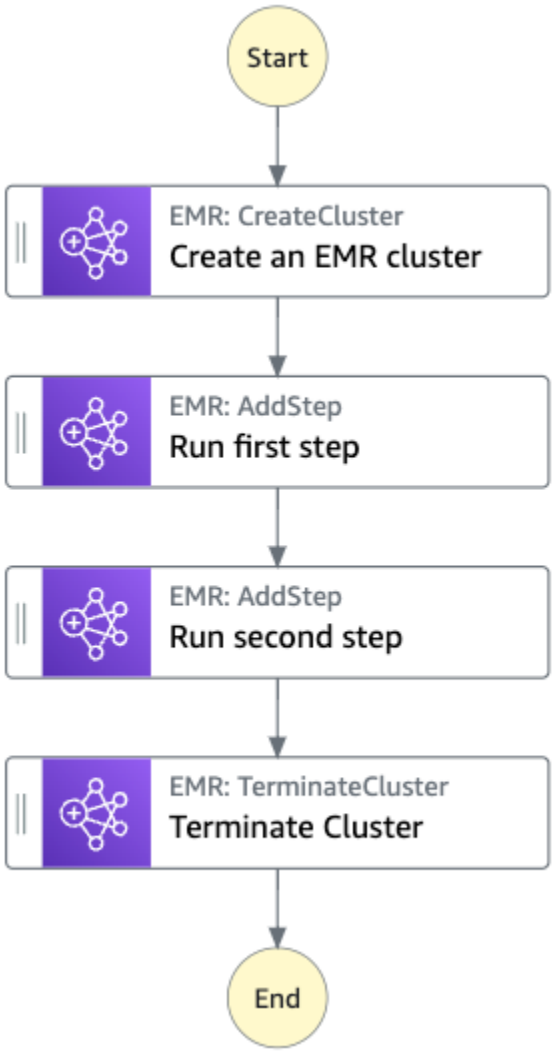
1. Ouvrez la console Step Functions et choisissez Create state machine.
2. Tapez **`Manage an EMR job`** dans le champ de recherche, puis choisissez Gérer une tâche EMR dans les résultats de recherche renvoyés.
3. Choisissez Next (Suivant) pour continuer.

4. Step Functions répertorie les Services AWS éléments utilisés dans l'exemple de projet que vous avez sélectionné. Il montre également un graphique de flux de travail pour l'exemple de projet. Déployez ce projet sur votre site Compte AWS ou utilisez-le comme point de départ pour créer vos propres projets. Selon la façon dont vous souhaitez procéder, choisissez Exécuter une démo ou Construire à partir de celle-ci.

   Cet exemple de projet déploie les ressources suivantes :

   - Un compartiment Amazon S3
   - Un Amazon EMR cluster
   - Une machine AWS Step Functions étatique
   - Rôles associés AWS Identity and Access Management (IAM)

   L'image suivante montre le graphique du flux de travail pour le projet d'exemple de tâche Gérer un EMR :



5. Choisissez Utiliser le modèle pour poursuivre votre sélection.
6. Effectuez l'une des actions suivantes :

   - Si vous avez sélectionné Build on it, Step Functions crée le prototype de flux de travail pour l'exemple de projet que vous avez sélectionné. Step Functions ne déploie pas les ressources répertoriées dans la définition du flux de travail.

     Dans Workflow StudioMode de conception, glissez-déposez les états depuis le Navigateur d'états pour continuer à créer votre prototype de flux de travail. Vous pouvez également passer à un éditeur de code intégré similaire à VS Code pour mettre à jour la définition Amazon States Language (ASL) de votre machine à états dans la console Step Functions. Mode code Pour plus d'informations sur l'utilisation de Workflow Studio pour créer vos machines d'état, consultezUtilisation de Workflow Studio.

> **Important**
>
> N'oubliez pas de mettre à jour l'espace réservé Amazon Resource Name (ARN) pour les ressources utilisées dans l'exemple de projet avant d'exécuter votre flux de travail.

- Si vous avez sélectionné Run a demo, Step Functions crée un exemple de projet en lecture seule qui utilise un AWS CloudFormation modèle pour déployer les AWS ressources répertoriées dans ce modèle sur votre. Compte AWS

  > **Tip**
  >
  > Pour afficher la définition de la machine à états de l'exemple de projet, choisissez Code.

  Lorsque vous êtes prêt, choisissez Déployer et exécuter pour déployer l'exemple de projet et créer les ressources.

  La création de ces ressources et des autorisations IAM associées peut prendre jusqu'à 10 minutes. Pendant le déploiement de vos ressources, vous pouvez ouvrir le lien CloudFormation Stack ID pour voir quelles ressources sont mises en service.

  Une fois que toutes les ressources de l'exemple de projet ont été créées, vous pouvez voir le nouvel exemple de projet répertorié sur la page State machines.

  > **Important**
  >
  > Des frais standard peuvent s'appliquer pour chaque service utilisé dans le CloudFormation modèle.

Étape 2 : Exécuter la machine à états

1. Sur la page State machines, choisissez votre exemple de projet.
2. Sur la page d'exemple de projet, choisissez Démarrer l'exécution.
3. Dans la boîte de dialogue Démarrer l'exécution, procédez comme suit :

1. (Facultatif) Pour identifier votre exécution, vous pouvez lui donner un nom dans le champ Nom. Par défaut, Step Functions génère automatiquement un nom d'exécution unique.

   > **Note**
   >
   > Step Functions vous permet de créer des noms pour les machines d'état, les exécutions et les activités, ainsi que des étiquettes contenant des caractères non ASCII. Ces noms non ASCII ne fonctionnent pas avec Amazon. CloudWatch Pour être sûr de pouvoir suivre CloudWatch les métriques, choisissez un nom qui utilise uniquement des caractères ASCII.

2. (Facultatif) Dans la zone de saisie, entrez les valeurs d'entrée au format JSON pour exécuter votre flux de travail.

   Si vous avez choisi d'exécuter une démo, vous n'avez pas besoin de fournir d'entrée d'exécution.

   > **Note**
   >
   > Si le projet de démonstration que vous avez déployé contient des données d'entrée d'exécution préremplies, utilisez ces entrées pour exécuter la machine à états.

3. Choisissez Start execution (Démarrer l'exécution).

4. La console Step Functions vous dirige vers une page intitulée avec votre ID d'exécution. Cette page est connue sous le nom de page Détails de l'exécution. Sur cette page, vous pouvez consulter les résultats de l'exécution au fur et à mesure que l'exécution progresse ou une fois celle-ci terminée.

   Pour consulter les résultats de l'exécution, choisissez des états individuels dans la vue graphique, puis choisissez les onglets individuels du Détails de l'étape volet pour afficher les détails de chaque état, y compris les entrées, les sorties et la définition respectivement. Pour plus de détails sur les informations d'exécution que vous pouvez consulter sur la page Détails de l'exécution, voirPage de détails d'exécution — Vue d'ensemble de l'interface.

Exemple de code de machine d'état

Dans cet exemple de projet, la machine à états s'intègre à Amazon EMR en transmettant des paramètres directement à ces ressources. Parcourez cet exemple de machine à états pour découvrir comment Step Functions utilise une machine à états pour appeler la tâche Amazon EMR de manière synchrone, attend que la tâche réussisse ou échoue, puis arrête le cluster.

Pour plus d'informations sur la manière de AWS Step Functions contrôler d'autres AWS services, consultezUtilisation AWS Step Functions avec d'autres services.

```
{
  "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
    }
  }
}
```

Exemple IAM

Cet exemple de politique AWS Identity and Access Management (IAM) généré par l'exemple de projet inclut le minimum de privilèges nécessaires pour exécuter la machine à états et les ressources associées. Il est recommandé d'inclure uniquement les autorisations nécessaires dans vos politiques IAM.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
```

```
                "elasticmapreduce:RunJobFlow",
                "elasticmapreduce:DescribeCluster",
                "elasticmapreduce:TerminateJobFlows"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": [
                "arn:aws:iam::123456789012:role/StepFunctionsSample-EMRJobManagement-
EMRServiceRole-ANPAJ2UCCR6DPCEXAMPLE",
                "arn:aws:iam::123456789012:role/StepFunctionsSample-
EMRJobManagementWJALRXUTNFEMI-ANPAJ2UCCR6DPCEXAMPLE-
EMREc2InstanceProfile-1ANPAJ2UCCR6DPCEXAMPLE"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "events:PutTargets",
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": [
                "arn:aws:events:sa-east-1:123456789012:rule/
StepFunctionsGetEventForEMRRunJobFlowRule"
            ]
        }
    ]
}
```

La stratégie suivante garantit qu'`addStep` dispose d'autorisations suffisantes.

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

Pour plus d'informations sur la configuration de l'IAM lors de l'utilisation de Step Functions avec d'autres AWS services, consultezPolitiques IAM pour les services intégrés.

Exécuter une EMR Serverless tâche

Cet exemple de projet montre comment créer et démarrer une EMR Serverless application. Ce projet montre également comment exécuter plusieurs tâches dans cette application.

Cet exemple de projet crée la machine d'état, les AWS ressources de support et configure les autorisations IAM associées. Explorez cet exemple de projet pour en savoir plus sur l'exécution de EMR Serverless tâches à Step Functions l'aide de machines à états ou utilisez-le comme point de départ pour vos propres projets.

> **Important**
>
> EMR Serverless n'a pas de niveau de tarification gratuit. L'exécution de l'exemple de projet entraînera des coûts. Vous trouverez des informations sur les prix sur la page Tarification Amazon EMR Serverless
> En outre, la disponibilité de l'intégration des EMR Serverless services dépend de la disponibilité des EMR Serverless API. De ce fait, cet exemple de projet peut ne pas fonctionner correctement ou ne pas être disponible dans certains cas Régions AWS.

Consultez la rubrique Autres considérations pour plus d'informations sur la disponibilité de EMR Serverless l'entrée Régions AWS.

AWS CloudFormationModèle et ressources supplémentaires

Vous utilisez un CloudFormation modèle pour déployer cet exemple de projet. Ce modèle crée les ressources suivantes dans votre Compte AWS :

- Une machine Step Functions étatique.
- Rôle d'exécution pour la machine à états. Ce rôle accorde les autorisations dont votre machine d'état a besoin pour accéder à Services AWS d'autres ressources telles que l'EMR ServerlessCreateApplicationaction.
- Rôle d'exécution de tâches pourEMR Serverless. Ce rôle accorde les autorisations que l'exécution EMR Serverless d'une tâche peut assumer lorsqu'elle appelle d'autres services en votre nom.

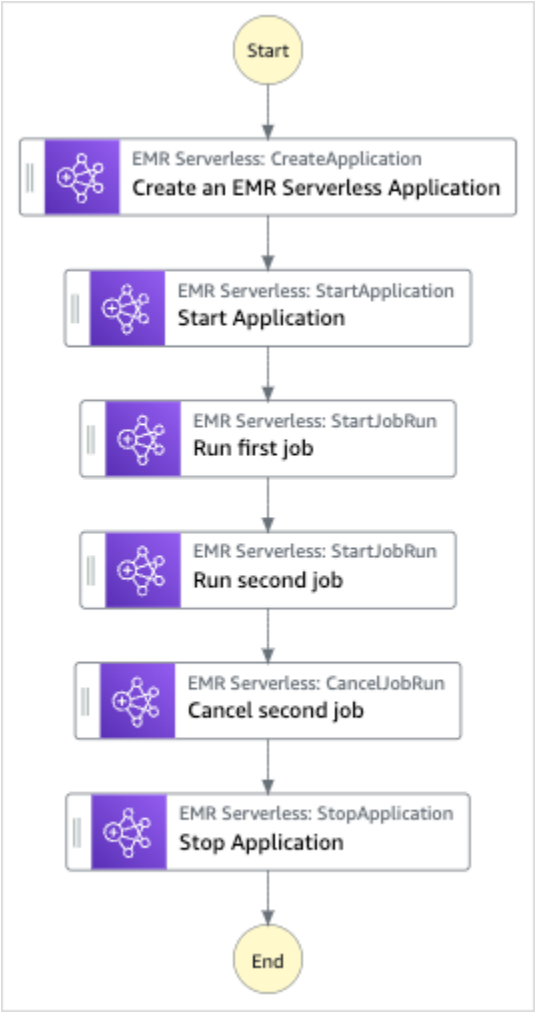
Étape 1 : créer la machine à états et provisionner les ressources

1. Ouvrez la console Step Functions et choisissez Create state machine.
2. Tapez **`EMR Serverless`** dans la zone de recherche, puis choisissez Exécuter une EMR Serverless tâche dans les résultats de recherche renvoyés.
3. Choisissez Next (Suivant) pour continuer.
4. Step Functions répertorie les Services AWS éléments utilisés dans l'exemple de projet que vous avez sélectionné. Il montre également un graphique de flux de travail pour l'exemple de projet. Déployez ce projet sur votre site Compte AWS ou utilisez-le comme point de départ pour créer vos propres projets. Selon la façon dont vous souhaitez procéder, choisissez Exécuter une démo ou Construire à partir de celle-ci.

   Cet exemple de projet déploie les ressources suivantes :

   - Une machine d'état Step Functions
   - Rôles AWS Identity and Access Management (IAM) connexes

   L'image suivante montre le graphique du flux de travail pour le projet d'exemple Run an EMR Serverless job :



5. Choisissez Utiliser le modèle pour poursuivre votre sélection.
6. Effectuez l'une des actions suivantes :

   - Si vous avez sélectionné Build on it, Step Functions crée le prototype de flux de travail pour l'exemple de projet que vous avez sélectionné. Step Functions ne déploie pas les ressources répertoriées dans la définition du flux de travail.

     Dans Workflow StudioMode de conception, glissez-déposez les états depuis le Navigateur d'états pour continuer à créer votre prototype de flux de travail. Vous pouvez également passer à un éditeur de code intégré similaire à VS Code pour mettre à jour la définition Amazon States Language (ASL) de votre machine à états dans la console Step Functions. Mode code Pour plus d'informations sur l'utilisation de Workflow Studio pour créer vos machines d'état, consultezUtilisation de Workflow Studio.

> **Important**
>
> N'oubliez pas de mettre à jour l'espace réservé Amazon Resource Name (ARN) pour les ressources utilisées dans l'exemple de projet avant d'exécuter votre flux de travail.

- Si vous avez sélectionné Run a demo, Step Functions crée un exemple de projet en lecture seule qui utilise un AWS CloudFormation modèle pour déployer les AWS ressources répertoriées dans ce modèle sur votre. Compte AWS

  > **Tip**
  >
  > Pour afficher la définition de la machine à états de l'exemple de projet, choisissez Code.

  Lorsque vous êtes prêt, choisissez Déployer et exécuter pour déployer l'exemple de projet et créer les ressources.

  La création de ces ressources et des autorisations IAM associées peut prendre jusqu'à 10 minutes. Pendant le déploiement de vos ressources, vous pouvez ouvrir le lien CloudFormation Stack ID pour voir quelles ressources sont mises en service.

  Une fois que toutes les ressources de l'exemple de projet ont été créées, vous pouvez voir le nouvel exemple de projet répertorié sur la page State machines.

  > **Important**
  >
  > Des frais standard peuvent s'appliquer pour chaque service utilisé dans le CloudFormation modèle.

Étape 2 : Exécuter la machine à états

1. Sur la page State machines, choisissez votre exemple de projet.
2. Sur la page d'exemple de projet, choisissez Démarrer l'exécution.
3. Dans la boîte de dialogue Démarrer l'exécution, procédez comme suit :

1. (Facultatif) Pour identifier votre exécution, vous pouvez lui donner un nom dans le champ Nom. Par défaut, Step Functions génère automatiquement un nom d'exécution unique.

   > **Note**
   >
   > Step Functions vous permet de créer des noms pour les machines d'état, les exécutions, les activités et les étiquettes contenant des caractères non ASCII. Ces noms non ASCII ne fonctionnent pas avec Amazon. CloudWatch Pour être sûr de pouvoir suivre CloudWatch les métriques, choisissez un nom qui utilise uniquement des caractères ASCII.

2. (Facultatif) Dans la zone de saisie, entrez les valeurs d'entrée au format JSON pour exécuter votre flux de travail.

   Si vous avez choisi d'exécuter une démo, vous n'avez pas besoin de fournir d'entrée d'exécution.

3. Choisissez Start execution (Démarrer l'exécution).

4. La console Step Functions vous dirige vers une page intitulée avec votre ID d'exécution. Cette page est connue sous le nom de page Détails de l'exécution. Sur cette page, vous pouvez consulter les résultats de l'exécution au fur et à mesure que l'exécution progresse ou une fois celle-ci terminée.

   Pour consulter les résultats de l'exécution, choisissez des états individuels dans la vue graphique, puis choisissez les onglets individuels du Détails de l'étape volet pour afficher les détails de chaque état, y compris les entrées, les sorties et la définition respectivement. Pour plus de détails sur les informations d'exécution que vous pouvez consulter sur la page Détails de l'exécution, voirPage de détails d'exécution — Vue d'ensemble de l'interface.

Démarrer un flux de travail au sein d'un flux de travail (Step Functions, Lambda)

Cet exemple de projet montre comment utiliser une machine à AWS Step Functions états pour démarrer d'autres exécutions de machines à états. Pour plus d'informations sur le lancement d'exécutions par machine à états à partir d'une autre machine à états, consultezDémarrer les exécutions de flux de travail à partir d'un état de tâche.

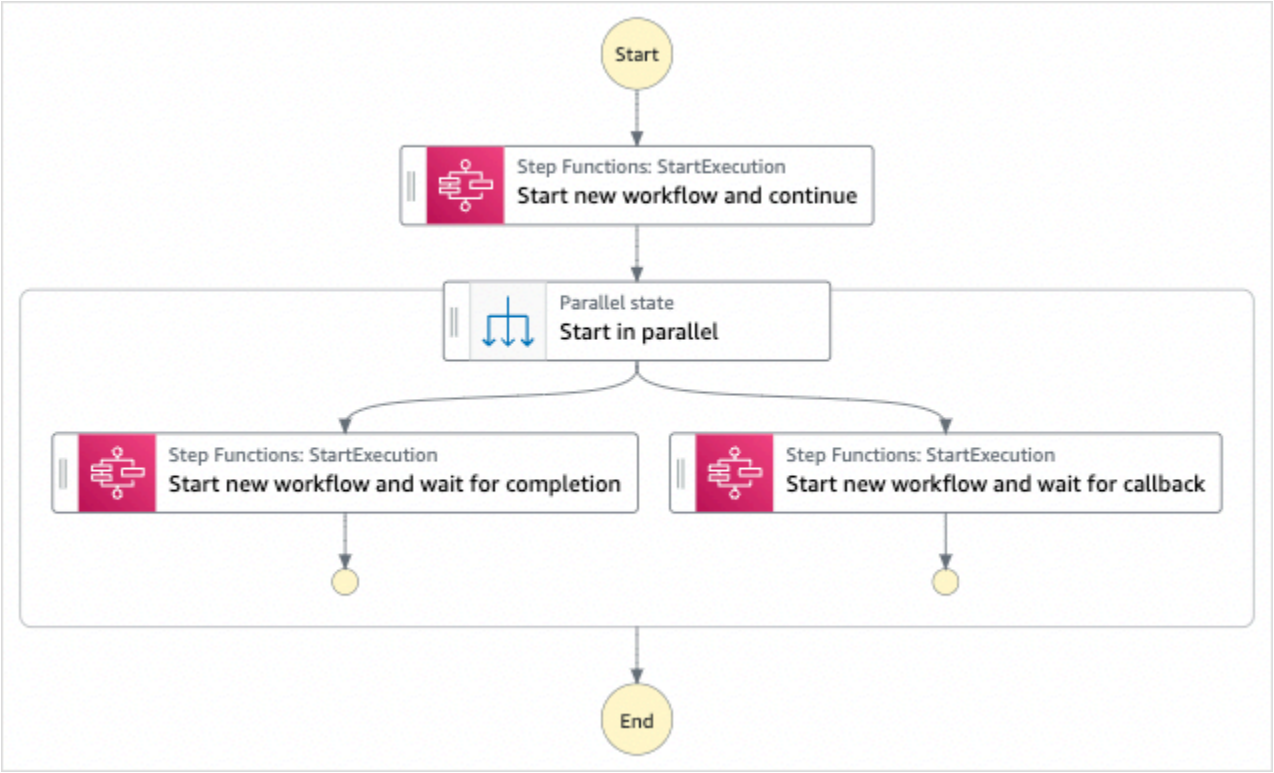
Étape 1 : créer la machine à états et provisionner les ressources

1. Ouvrez la console Step Functions et choisissez Create state machine.
2. Tapez **`Start a workflow within a workflow`** dans la zone de recherche, puis choisissez Démarrer un flux de travail dans un flux de travail à partir des résultats de recherche renvoyés.
3. Choisissez Next (Suivant) pour continuer.
4. Step Functions répertorie les Services AWS éléments utilisés dans l'exemple de projet que vous avez sélectionné. Il montre également un graphique de flux de travail pour l'exemple de projet. Déployez ce projet sur votre site Compte AWS ou utilisez-le comme point de départ pour créer vos propres projets. Selon la façon dont vous souhaitez procéder, choisissez Exécuter une démo ou Construire à partir de celle-ci.

   Cet exemple de projet déploie les ressources suivantes :

   - Une machine d'état supplémentaire. L'exécution de cette machine d'état est lancée par la machine d'état que vous exécutez.
   - Une fonction Lambda de rappel. Cette fonction est utilisée dans la machine à états supplémentaire pour implémenter le mécanisme de rappel.
   - Une machine AWS Step Functions étatique
   - Rôles associés AWS Identity and Access Management (IAM)

   L'image suivante montre le graphique du flux de travail pour le projet Démarrer un flux de travail dans un exemple de flux de travail :



5. Choisissez Utiliser le modèle pour poursuivre votre sélection.
6. Effectuez l'une des actions suivantes :

   - Si vous avez sélectionné Build on it, Step Functions crée le prototype de flux de travail pour l'exemple de projet que vous avez sélectionné. Step Functions ne déploie pas les ressources répertoriées dans la définition du flux de travail.

     Dans Workflow StudioMode de conception, glissez-déposez les états depuis le Navigateur d'états pour continuer à créer votre prototype de flux de travail. Vous pouvez également passer à un éditeur de code intégré similaire à VS Code pour mettre à jour la définition Amazon States Language (ASL) de votre machine à états dans la console Step Functions. Mode code Pour plus d'informations sur l'utilisation de Workflow Studio pour créer vos machines d'état, consultezUtilisation de Workflow Studio.

     > **Important**
     >
     > N'oubliez pas de mettre à jour l'espace réservé Amazon Resource Name (ARN) pour les ressources utilisées dans l'exemple de projet avant d'exécuter votre flux de travail.

- Si vous avez sélectionné Run a demo, Step Functions crée un exemple de projet en lecture seule qui utilise un AWS CloudFormation modèle pour déployer les AWS ressources répertoriées dans ce modèle sur votre. Compte AWS

  > **Tip**
  >
  > Pour afficher la définition de la machine à états de l'exemple de projet, choisissez Code.

  Lorsque vous êtes prêt, choisissez Déployer et exécuter pour déployer l'exemple de projet et créer les ressources.

  La création de ces ressources et des autorisations IAM associées peut prendre jusqu'à 10 minutes. Pendant le déploiement de vos ressources, vous pouvez ouvrir le lien CloudFormation Stack ID pour voir quelles ressources sont mises en service.

  Une fois que toutes les ressources de l'exemple de projet ont été créées, vous pouvez voir le nouvel exemple de projet répertorié sur la page State machines.

  > **Important**
  >
  > Des frais standard peuvent s'appliquer pour chaque service utilisé dans le CloudFormation modèle.

Étape 2 : Exécuter la machine à états

1. Sur la page State machines, choisissez votre exemple de projet.
2. Sur la page d'exemple de projet, choisissez Démarrer l'exécution.
3. Dans la boîte de dialogue Démarrer l'exécution, procédez comme suit :

   1. (Facultatif) Pour identifier votre exécution, vous pouvez lui donner un nom dans le champ Nom. Par défaut, Step Functions génère automatiquement un nom d'exécution unique.

> **Note**
>
> Step Functions vous permet de créer des noms pour les machines d'état, les exécutions et les activités, ainsi que des étiquettes contenant des caractères non ASCII. Ces noms non ASCII ne fonctionnent pas avec Amazon. CloudWatch Pour être sûr de pouvoir suivre CloudWatch les métriques, choisissez un nom qui utilise uniquement des caractères ASCII.

2. (Facultatif) Dans la zone de saisie, entrez les valeurs d'entrée au format JSON pour exécuter votre flux de travail.

   Si vous avez choisi d'exécuter une démo, vous n'avez pas besoin de fournir d'entrée d'exécution.

   > **Note**
   >
   > Si le projet de démonstration que vous avez déployé contient des données d'entrée d'exécution préremplies, utilisez ces entrées pour exécuter la machine à états.

3. Choisissez Start execution (Démarrer l'exécution).
4. La console Step Functions vous dirige vers une page intitulée avec votre ID d'exécution. Cette page est connue sous le nom de page Détails de l'exécution. Sur cette page, vous pouvez consulter les résultats de l'exécution au fur et à mesure que l'exécution progresse ou une fois celle-ci terminée.

   Pour consulter les résultats de l'exécution, choisissez des états individuels dans la vue graphique, puis choisissez les onglets individuels du Détails de l'étape volet pour afficher les détails de chaque état, y compris les entrées, les sorties et la définition respectivement. Pour plus de détails sur les informations d'exécution que vous pouvez consulter sur la page Détails de l'exécution, voirPage de détails d'exécution — Vue d'ensemble de l'interface.

Exemple de code de machine d'état

Dans cet exemple de projet, la machine à états intègre une autre machine à états AWS Lambda en transmettant des paramètres directement à ces ressources.

Parcourez cet exemple de machine à états pour voir comment Step Functions appelle l'action `StartExecution` API pour l'autre machine à états. Il lance deux instances de l'autre machine d'état en parallèle : l'une utilisant le modèle Exécuter une tâche (.sync) et l'autre utilisant le modèle Attendre un rappel avec le jeton de tâche.

Pour plus d'informations sur la manière de AWS Step Functions contrôler d'autres AWS services, consultezUtilisation AWS Step Functions avec d'autres services.

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

Pour plus d'informations sur la configuration de l'IAM lors de l'utilisation de Step Functions avec d'autres AWS services, consultezPolitiques IAM pour les services intégrés.

Traitement dynamique des données avec un état cartographique

Cet exemple de projet illustre le parallélisme dynamique à l'aide d'un état Map.

Dans ce projet, Step Functions utilise une AWS Lambda fonction pour extraire des messages d'une file d'attente Amazon SQS et transmettre un tableau JSON de ces messages à un `Map` état. Pour chaque message de la file d'attente, la machine d'état écrit le message dans DynamoDB, invoque l'autre fonction Lambda pour supprimer le message d'Amazon SQS, puis le publie dans la rubrique Amazon SNS.

Pour plus d'informations sur `Map` les états et les intégrations des services Step Functions, consultez les pages suivantes :

- Map
- Utilisation AWS Step Functions avec d'autres services

Étape 1 : créer la machine à états et provisionner les ressources

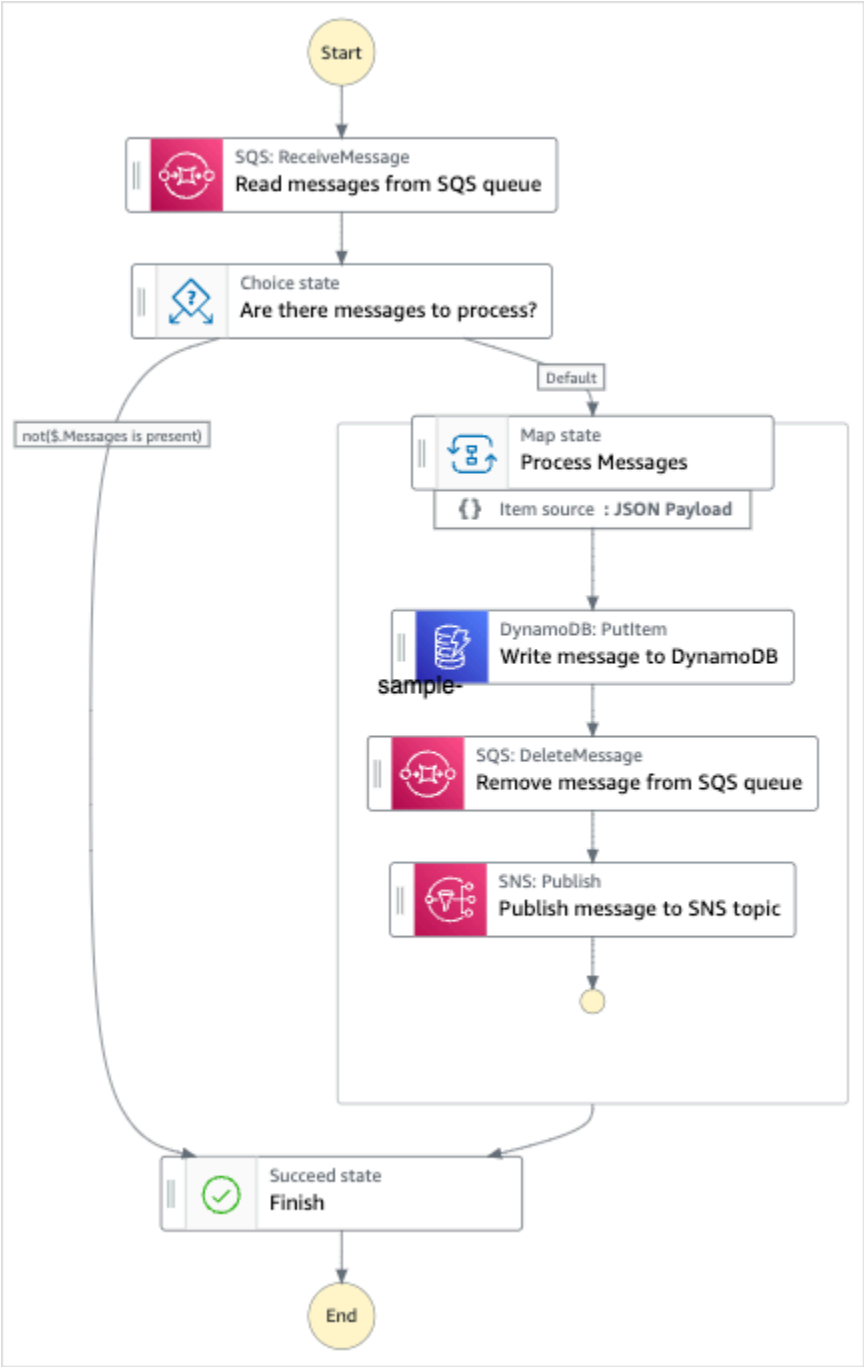
1. Ouvrez la console Step Functions et choisissez Create state machine.
2. Tapez **`Dynamically process data with a Map state`** dans la zone de recherche, puis choisissez Traiter dynamiquement les données avec un état cartographique à partir des résultats de recherche renvoyés.
3. Choisissez Next (Suivant) pour continuer.
4. Step Functions répertorie les Services AWS éléments utilisés dans l'exemple de projet que vous avez sélectionné. Il montre également un graphique de flux de travail pour l'exemple de projet. Déployez ce projet sur votre site Compte AWS ou utilisez-le comme point de départ pour créer vos propres projets. Selon la façon dont vous souhaitez procéder, choisissez Exécuter une démo ou Construire à partir de celle-ci.

   Cet exemple de projet déploie les ressources suivantes :

   - Une file d'attente Amazon SQS à partir de laquelle l'état de la carte lit et supprime les messages de manière itérative.
   - Table DynamoDB dans laquelle l'état de la carte écrit des messages de manière itérative.
   - Rubrique Amazon SNS dans laquelle Step Functions publie les messages qu'elle lit depuis la file d'attente Amazon SQS.
   - Deux AWS Lambda fonctions

- Une machine AWS Step Functions étatique
- Rôles associés AWS Identity and Access Management (IAM)

L'image suivante montre le graphique du flux de travail pour le projet d'exemple de traitement dynamique des données avec un état de carte :



5. Choisissez Utiliser le modèle pour poursuivre votre sélection.

6. Effectuez l'une des actions suivantes :

   - Si vous avez sélectionné Build on it, Step Functions crée le prototype de flux de travail pour l'exemple de projet que vous avez sélectionné. Step Functions ne déploie pas les ressources répertoriées dans la définition du flux de travail.

     Dans Workflow StudioMode de conception, glissez-déposez les états depuis le Navigateur d'états pour continuer à créer votre prototype de flux de travail. Vous pouvez également passer à un éditeur de code intégré similaire à VS Code pour mettre à jour la définition Amazon States Language (ASL) de votre machine à états dans la console Step Functions. Mode code Pour plus d'informations sur l'utilisation de Workflow Studio pour créer vos machines d'état, consultezUtilisation de Workflow Studio.

     > **Important**
     >
     > N'oubliez pas de mettre à jour l'espace réservé Amazon Resource Name (ARN) pour les ressources utilisées dans l'exemple de projet avant d'exécuter votre flux de travail.

   - Si vous avez sélectionné Run a demo, Step Functions crée un exemple de projet en lecture seule qui utilise un AWS CloudFormation modèle pour déployer les AWS ressources répertoriées dans ce modèle sur votre. Compte AWS

     > **Tip**
     >
     > Pour afficher la définition de la machine à états de l'exemple de projet, choisissez Code.

     Lorsque vous êtes prêt, choisissez Déployer et exécuter pour déployer l'exemple de projet et créer les ressources.

     La création de ces ressources et des autorisations IAM associées peut prendre jusqu'à 10 minutes. Pendant le déploiement de vos ressources, vous pouvez ouvrir le lien CloudFormation Stack ID pour voir quelles ressources sont mises en service.

     Une fois que toutes les ressources de l'exemple de projet ont été créées, vous pouvez voir le nouvel exemple de projet répertorié sur la page State machines.

> **Important**
>
> Des frais standard peuvent s'appliquer pour chaque service utilisé dans le CloudFormation modèle.

Une fois les ressources de l'exemple de projet déployées, vous devez ajouter des éléments à la file d'attente Amazon SQS et vous abonner à la rubrique Amazon SNS avant d'exécuter la machine d'état.

Étape 2 : Abonnez-vous à la rubrique Amazon SNS

1. Ouvrez la console Amazon SNS.
2. Sélectionnez Topics (Rubriques) et choisissez la rubrique qui a été créée par l'exemple de projet d'état `Map`.

   Le nom sera similaire à MapSampleProj-SNSTopic-1CQO4HQ3IR1Kn.
3. Choisissez Créer un abonnement.

   La page Create subscription (Créer un abonnement) s'affiche, affichant l'ARN de la rubrique.
4. Pour Protocol (Protocole), choisissez Email (E-mail).
5. Dans Endpoint (Point de terminaison) entrez une adresse e-mail pour vous abonner à la rubrique.
6. Choisissez Créer un abonnement.

   > **Note**
   >
   > Vous devez confirmer l'abonnement dans votre e-mail avant qu'il ne soit actif.
7. Ouvrez l'e-mail Subscription Confirmation (Confirmation d'abonnement) dans le compte associé et accédez à l'URL de confirmation d'abonnement.

   La page Subscription confirmed! (Abonnement confirmé !) s'affiche.

Étape 3 : ajouter des messages à la file d'attente Amazon SQS

1. Ouvrez la console Amazon SQS.
2. Sélectionnez la file d'attente qui a été créée par l'exemple de projet d'état `Map`.

   Le nom sera similaire à MapSampleProj-sqsqueue-1udic9vzdOrn7.
3. Choisissez Envoyer et recevoir des messages.
4. Sur la page Envoyer et recevoir des messages, entrez un message et choisissez Envoyer un message.
5. Entrez un autre message et choisissez Envoyer un message. Continuez à saisir d'autres messages jusqu'à ce que vous en ayez plusieurs dans la file d'attente Amazon SQS.

Étape 4 : Exécutez la machine d'état

> **Note**
>
> Les files d'attente sur Amazon SNS sont finalement cohérentes. Pour obtenir de meilleurs résultats, patientez quelques minutes entre le remplissage de votre file d'attente et l'exécution d'une exécution de votre machine d'état.

1. Sur la page State machines, choisissez votre exemple de projet.
2. Sur la page d'exemple de projet, choisissez Démarrer l'exécution.
3. Dans la boîte de dialogue Démarrer l'exécution, procédez comme suit :

   1. (Facultatif) Pour identifier votre exécution, vous pouvez lui donner un nom dans le champ Nom. Par défaut, Step Functions génère automatiquement un nom d'exécution unique.

      > **Note**
      >
      > Step Functions vous permet de créer des noms pour les machines d'état, les exécutions et les activités, ainsi que des étiquettes contenant des caractères non ASCII. Ces noms non ASCII ne fonctionnent pas avec Amazon. CloudWatch Pour être sûr de pouvoir suivre CloudWatch les métriques, choisissez un nom qui utilise uniquement des caractères ASCII.

2. (Facultatif) Dans la zone de saisie, entrez les valeurs d'entrée au format JSON pour exécuter votre flux de travail.

   Si vous avez choisi d'exécuter une démo, vous n'avez pas besoin de fournir d'entrée d'exécution.

   > **Note**
   >
   > Si le projet de démonstration que vous avez déployé contient des données d'entrée d'exécution préremplies, utilisez ces entrées pour exécuter la machine à états.

3. Choisissez Start execution (Démarrer l'exécution).
4. La console Step Functions vous dirige vers une page intitulée avec votre ID d'exécution. Cette page est connue sous le nom de page Détails de l'exécution. Sur cette page, vous pouvez consulter les résultats de l'exécution au fur et à mesure que l'exécution progresse ou une fois celle-ci terminée.

   Pour consulter les résultats de l'exécution, choisissez des états individuels dans la vue graphique, puis choisissez les onglets individuels du Détails de l'étape volet pour afficher les détails de chaque état, y compris les entrées, les sorties et la définition respectivement. Pour plus de détails sur les informations d'exécution que vous pouvez consulter sur la page Détails de l'exécution, voirPage de détails d'exécution — Vue d'ensemble de l'interface.

Exemple de code machine d'état

Dans cet exemple de projet, la machine à états s'intègre à Amazon SQS, Amazon SNS et Lambda en transmettant des paramètres directement à ces ressources.

Parcourez cet exemple de machine à états pour découvrir comment Step Functions contrôle Lambda, DynamoDB et Amazon SNS en se connectant à l'Amazon Resource Name (ARN) sur le terrain et `Resource` en passant à l'API du service. `Parameters`

Pour plus d'informations sur la manière de AWS Step Functions contrôler d'autres AWS services, consultezUtilisation AWS Step Functions avec d'autres services.

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

Exemple IAM

Cet exemple de politique AWS Identity and Access Management (IAM) généré par l'exemple de projet inclut le minimum de privilèges nécessaires pour exécuter la machine à états et les ressources associées. Nous vous recommandons de n'inclure que les autorisations nécessaires dans vos politiques IAM.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": [
                "arn:aws:lambda:us-east-1:012345678901:function:MapSampleProj-
ReadFromSQSQueueLambda-1MY3M63RMJVA9",
                "arn:aws:lambda:us-east-1:012345678901:function:MapSampleProj-
DeleteFromSQSQueueLambda-198J2839ZO5K2"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "dynamodb:PutItem"
            ],
            "Resource": [
                "arn:aws:dynamodb:us-east-1:012345678901:table/MapSampleProj-DDBTable-
YJDJ1MKIN6C5"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "sns:Publish"
            ],
            "Resource": [
                "arn:aws:sns:us-east-1:012345678901:MapSampleProj-
SNSTopic-1CQO4HQ3IR1KN"
            ],
            "Effect": "Allow"
        }
    ]
```

```
}
```

Pour plus d'informations sur la configuration de l'IAM lors de l'utilisation de Step Functions avec d'autres AWS services, consultezPolitiques IAM pour les services intégrés.

Traiter un fichier CSV avec une carte distribuée

Cet exemple de projet montre comment utiliser l'état de la carte distribuée pour itérer plus de 10 000 lignes d'un fichier CSV généré à l'aide d'une Lambda fonction. Le fichier CSV contient les informations d'expédition des commandes des clients et est stocké dans un compartiment Amazon S3. La carte distribuée itère un lot de 10 lignes dans le fichier CSV à des fins d'analyse des données.

La carte distribuée contient une Lambda fonction permettant de détecter les commandes retardées. La carte distribuée contient également une carte en ligne pour traiter les commandes retardées par lots et renvoie ces commandes différées dans un tableau. Pour chaque commande différée, la carte intégrée envoie un message à une Amazon SQS file d'attente. Enfin, cet exemple de projet stocke les résultats de Map Run dans un autre compartiment Amazon S3 de votre Compte AWS.

Avec Distributed Map, vous pouvez exécuter jusqu'à 10 000 exécutions parallèles de flux de travail enfants à la fois. Dans cet exemple de projet, la simultanéité maximale de Distributed Map est fixée à 1 000, ce qui la limite à 1 000 exécutions parallèles de flux de travail enfants.

Cet exemple de projet crée la machine d'état, les AWS ressources de support et configure les autorisations IAM associées. Explorez cet exemple de projet pour en savoir plus sur l'utilisation de la carte distribuée pour orchestrer des charges de travail parallèles à grande échelle, ou utilisez-la comme point de départ pour vos propres projets.

AWS CloudFormation modèle et ressources supplémentaires

Vous utilisez un CloudFormation modèle pour déployer cet exemple de projet. Ce modèle crée les ressources suivantes dans votre Compte AWS :

- Une machine à états Step Functions.
- Rôle d'exécution pour la machine à états. Ce rôle accorde les autorisations dont votre machine d'état a besoin pour accéder à d'autres Services AWS ressources telles que l'action Invoke de la fonction Lambda.
- Une fonction Lambda nommée `CSVGeneratorFunction` qui génère un fichier CSV contenant les détails de la commande du client.

- Rôle d'exécution de la fonction Lambda du générateur CSV. Ce rôle accorde à la fonction l'autorisation d'accéder à d'autres Services AWS.
- Un compartiment d'entrée Amazon S3 pour stocker le fichier CSV généré.
- Une fonction Lambda de détection des commandes différées qui analyse les données du fichier CSV et détecte les commandes retardées.
- Rôle d'exécution de la fonction Lambda à ordre différé. Ce rôle accorde à la fonction l'autorisation d'accéder à d'autres Services AWS.
- Un compartiment de sortie Amazon S3 pour stocker les résultats d'analyse des commandes des clients.
- Une file d'attente Amazon SQS à laquelle Step Functions envoie des messages pour chaque commande différée. Ces messages contiennent les identifiants des clients et leurs commandes.
- Un groupe de CloudWatch journaux qui stocke les informations relatives à l'historique d'exécution de la machine à états.

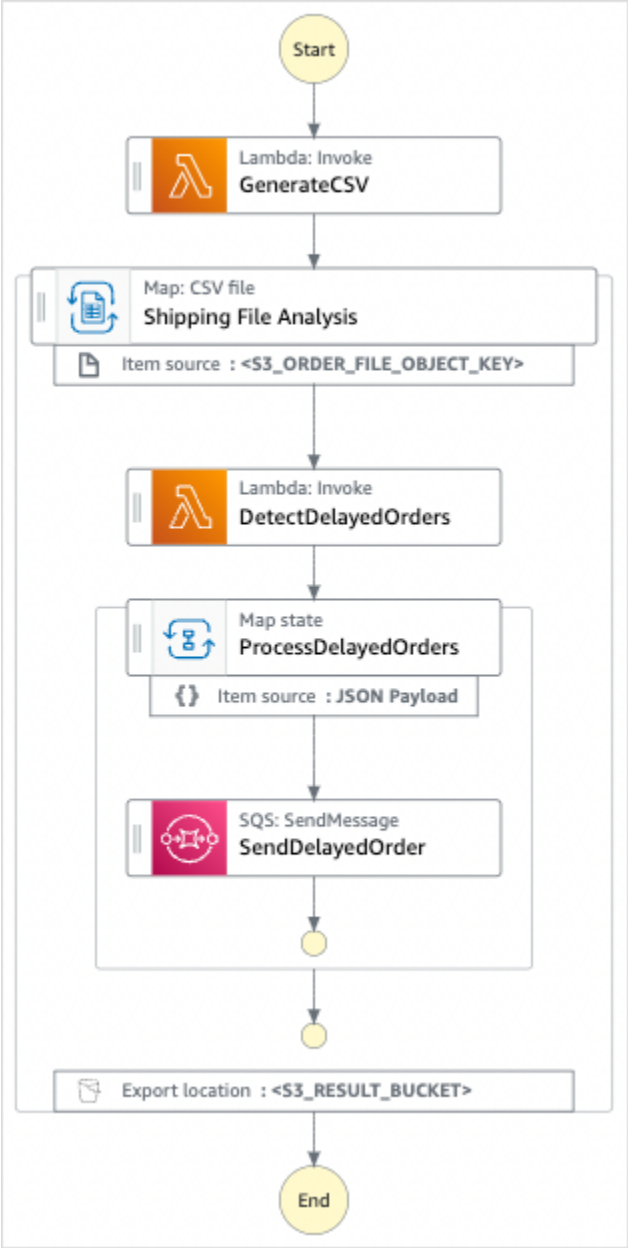
> **Important**
>
> Des frais standard s'appliquent pour chaque service.

Étape 1 : créer la machine à états et provisionner les ressources

1. Ouvrez la console Step Functions et choisissez Create state machine.
2. Tapez **`Distributed Map to process a CSV file in S3`** dans la zone de recherche, puis choisissez Distributed Map pour traiter un fichier CSV dans S3 à partir des résultats de recherche renvoyés.
3. Choisissez Next (Suivant) pour continuer.
4. Step Functions répertorie les Services AWS éléments utilisés dans l'exemple de projet que vous avez sélectionné. Il montre également un graphique de flux de travail pour l'exemple de projet. Déployez ce projet sur votre site Compte AWS ou utilisez-le comme point de départ pour créer vos propres projets. Selon la façon dont vous souhaitez procéder, choisissez Exécuter une démo ou Construire à partir de celle-ci.

   Pour plus d'informations sur les ressources qui seront créées pour cet exemple de projet, consultezAWS CloudFormation modèle et ressources supplémentaires.

L'image suivante montre le graphique du flux de travail de la carte distribuée pour traiter un fichier CSV dans un exemple de projet S3 :



5. Choisissez Utiliser le modèle pour poursuivre votre sélection.
6. Effectuez l’une des actions suivantes :

   - Si vous avez sélectionné Build on it, Step Functions crée le prototype de flux de travail pour l'exemple de projet que vous avez sélectionné. Step Functions ne déploie pas les ressources répertoriées dans la définition du flux de travail.

Dans Workflow StudioMode de conception, glissez-déposez les états depuis le Navigateur d'états pour continuer à créer votre prototype de flux de travail. Vous pouvez également passer à un éditeur de code intégré similaire à VS Code pour mettre à jour la définition Amazon States Language (ASL) de votre machine à états dans la console Step Functions. Mode code Pour plus d'informations sur l'utilisation de Workflow Studio pour créer vos machines d'état, consultezUtilisation de Workflow Studio.

> **Important**
>
> N'oubliez pas de mettre à jour l'espace réservé Amazon Resource Name (ARN) pour les ressources utilisées dans l'exemple de projet avant d'exécuter votre flux de travail.

- Si vous avez sélectionné Run a demo, Step Functions crée un exemple de projet en lecture seule qui utilise un AWS CloudFormation modèle pour déployer les AWS ressources répertoriées dans ce modèle sur votre. Compte AWS

  > **Tip**
  >
  > Pour afficher la définition de la machine à états de l'exemple de projet, choisissez Code.

  Lorsque vous êtes prêt, choisissez Déployer et exécuter pour déployer l'exemple de projet et créer les ressources.

  La création de ces ressources et des autorisations IAM associées peut prendre jusqu'à 10 minutes. Pendant le déploiement de vos ressources, vous pouvez ouvrir le lien CloudFormation Stack ID pour voir quelles ressources sont mises en service.

  Une fois que toutes les ressources de l'exemple de projet ont été créées, vous pouvez voir le nouvel exemple de projet répertorié sur la page State machines.

  > **Important**
  >
  > Des frais standard peuvent s'appliquer pour chaque service utilisé dans le CloudFormation modèle.

Étape 2 : Exécuter la machine à états

Une fois que toutes les ressources ont été provisionnées et déployées, vous pouvez exécuter la machine d'état.

1. Sur la page State machines, choisissez votre exemple de projet.
2. Sur la page d'exemple de projet, choisissez Démarrer l'exécution.
3. Dans la boîte de dialogue Démarrer l'exécution, procédez comme suit :

   a. (Facultatif) Entrez les valeurs d'entrée au format JSON pour exécuter votre exemple de projet.

      Si vous avez choisi d'exécuter une démo, vous n'avez pas besoin de fournir d'entrée d'exécution.

      > **Note**
      >
      > Si le projet de démonstration que vous avez déployé contient des données d'entrée d'exécution préremplies, utilisez ces entrées pour exécuter la machine à états.

   b. Choisissez Start execution (Démarrer l'exécution).

   c. (Facultatif) La console Step Functions vous dirige vers une page intitulée avec votre ID d'exécution. Cette page est connue sous le nom de page Détails de l'exécution. Sur cette page, vous pouvez consulter les résultats de l'exécution au fur et à mesure que l'exécution progresse ou une fois celle-ci terminée.

      Une fois l'exécution terminée, choisissez des états individuels dans la vue graphique, puis choisissez les onglets individuels du Détails de l'étape volet pour afficher les détails de chaque état, y compris les entrées, les sorties et la définition respectivement.

      - Pour plus de détails sur les informations d'exécution que vous pouvez consulter sur la page Détails de l'exécution, voirPage de détails d'exécution — Vue d'ensemble de l'interface.
      - Pour plus d'informations sur l'affichage de l'exécution d'un état de carte distribuée dans la console, consultezExamen de Map Run.

   d. (Facultatif) Vérifiez les résultats d'exécution exportés vers le compartiment Amazon S3. Ces résultats incluent des données, telles que les entrées et sorties d'exécution, l'ARN et le statut d'exécution. Pour de plus amples informations, veuillez consulter ResultWriter.

Traitez les données dans un compartiment Amazon S3 avec Distributed Map

Cet exemple de projet montre comment utiliser l'état de la carte distribuée pour traiter des données à grande échelle, par exemple, analyser les données météorologiques historiques et identifier la station météo qui affiche la température moyenne la plus élevée de la planète chaque mois. Les données météorologiques sont enregistrées dans plus de 12 000 fichiers CSV, qui sont à leur tour stockés dans un compartiment Amazon S3.

Cet exemple de projet inclut deux états de carte distribuée nommés Distributed S3 copy NOA Data et ProcessNoaData. Copie S3 distribuée NOA Data itère sur les fichiers CSV d'un compartiment public Amazon S3 nommé noaa-gsod-pdset les copie dans un compartiment Amazon S3 de votre. Compte AWS ProcessNoaaData effectue une itération sur les fichiers copiés et inclut une fonction Lambda qui effectue l'analyse de température.

L'exemple de projet vérifie d'abord le contenu du compartiment Amazon S3 en appelant l'action d'API ListObjectsV2. Sur la base du nombre de clés renvoyées en réponse à cet appel, l'exemple de projet prend l'une des décisions suivantes :

- Si le nombre de clés est supérieur ou égal à 1, le projet passe à l'état ProcessNoaaData. Cet état de carte distribuée inclut une Lambda fonction nommée TemperatureFunctionqui recherche la station météo ayant enregistré la température moyenne la plus élevée chaque mois. Cette fonction renvoie un dictionnaire avec `year-month` comme clé et un dictionnaire contenant des informations sur la station météo comme valeur.
- Si le nombre de clés renvoyées ne dépasse pas 1, l'état des données NOA de copie S3 distribuée répertorie tous les objets du compartiment public noaa-gsod-pdset copie de manière itérative les objets individuels dans un autre compartiment de votre compte par lots de 100. Une carte intégrée effectue la copie itérative des objets.

  Une fois tous les objets copiés, le projet passe à l'état ProcessNoaaData pour le traitement des données météorologiques.

L'exemple de projet passe enfin à une Lambda fonction réductrice qui effectue une agrégation finale des résultats renvoyés par la TemperatureFunctionfonction et écrit les résultats dans un Amazon DynamoDB tableau.

Avec Distributed Map, vous pouvez exécuter jusqu'à 10 000 exécutions parallèles de flux de travail enfants à la fois. Dans cet exemple de projet, la simultanéité maximale de ProcessNoaaData Distributed Map est fixée à 3 000, ce qui la limite à 3 000 exécutions parallèles de flux de travail enfants.

Cet exemple de projet crée la machine d'état, les AWS ressources de support et configure les autorisations IAM associées. Explorez cet exemple de projet pour en savoir plus sur l'utilisation de la carte distribuée pour orchestrer des charges de travail parallèles à grande échelle, ou utilisez-la comme point de départ pour vos propres projets.

> **Important**
>
> Cet exemple de projet n'est disponible que dans la région de l'est des États-Unis (Virginie du Nord).

AWS CloudFormation modèle et ressources supplémentaires

Vous utilisez un CloudFormation modèle pour déployer cet exemple de projet. Ce modèle crée les ressources suivantes dans votre Compte AWS :

- Une machine à états Step Functions.
- Rôle d'exécution pour la machine à états. Ce rôle accorde les autorisations dont votre machine d'état a besoin pour accéder à d'autres Services AWS ressources telles que l'action Invoke de la fonction Lambda.
- Un compartiment Amazon S3 nomméNOAADataBucket. Ce bucket contient les fichiers CSV contenant les données météorologiques.
- Fonction Lambda nommée `ReducerFunction` qui effectue une agrégation finale des données météorologiques et écrit les résultats dans une table Amazon DynamoDB.
- Rôle d'exécution de la fonction Lambda du réducteur. Ce rôle accorde à la fonction l'autorisation d'accéder à d'autres Services AWS.
- Un compartiment de sortie Amazon S3 nommé `ResultsBucket` pour stocker les résultats de l'analyse météo.
- Table DynamoDB `ResultsDynamoDBTable` nommée qui contient les résultats renvoyés par le. `ReducerFunction`
- Fonction Lambda nommée `TemperatureFunction` qui trouve la température moyenne mensuelle la plus élevée.

- Rôle d'exécution de la fonction Lambda. Ce rôle accorde à la fonction l'autorisation d'accéder à d'autres Services AWS.
- Un groupe de CloudWatch journaux qui stocke les informations relatives à l'historique d'exécution de la machine à états.

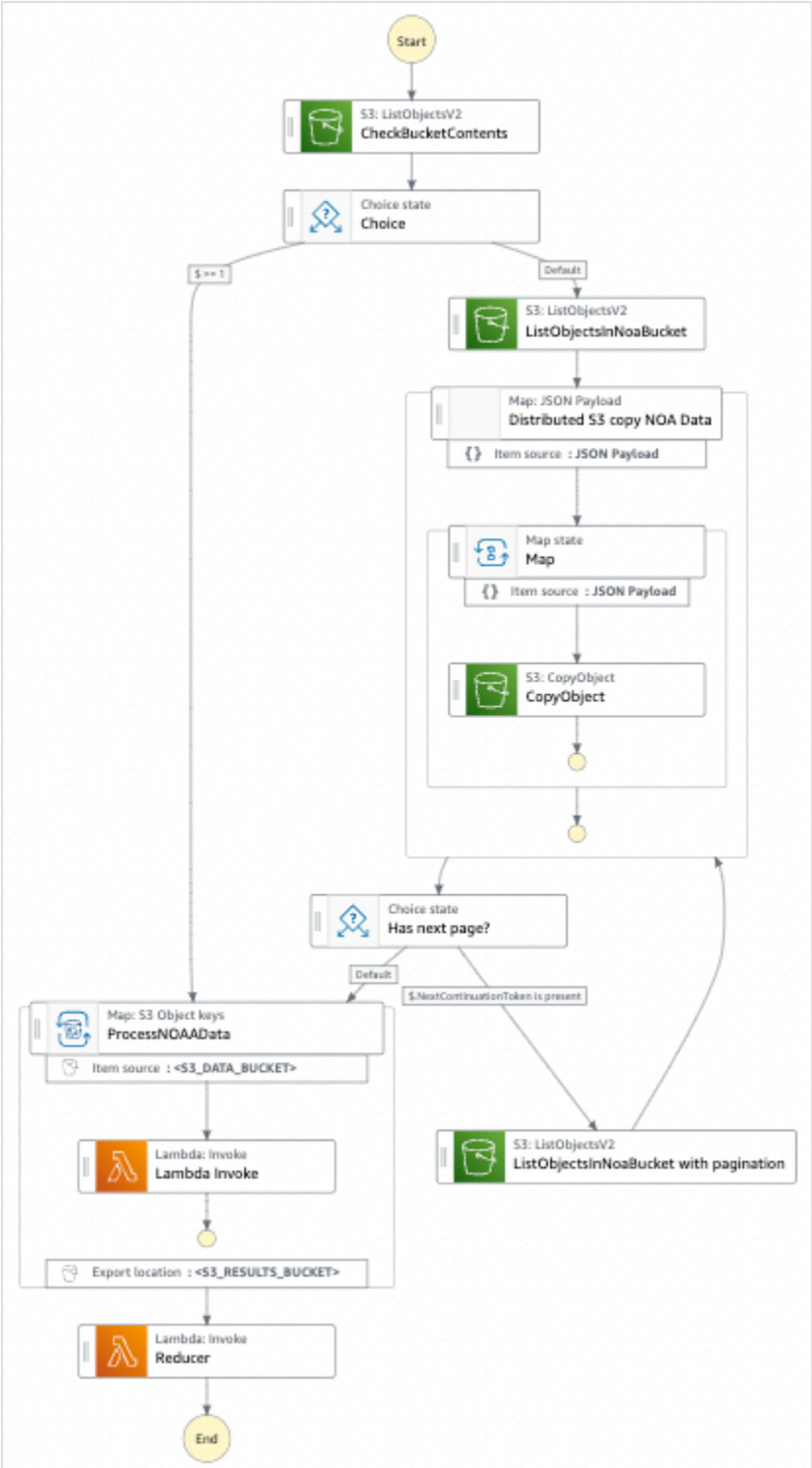
> **Important**
>
> Des frais standard s'appliquent pour chaque service.

Étape 1 : créer la machine à états et provisionner les ressources

1. Ouvrez la console Step Functions et choisissez Create state machine.
2. Tapez **`Distributed Map to process files in S3`** dans la zone de recherche, puis choisissez Distributed Map pour traiter les fichiers dans S3 à partir des résultats de recherche renvoyés.
3. Choisissez Next (Suivant) pour continuer.
4. Step Functions répertorie les Services AWS éléments utilisés dans l'exemple de projet que vous avez sélectionné. Il montre également un graphique de flux de travail pour l'exemple de projet. Déployez ce projet sur votre site Compte AWS ou utilisez-le comme point de départ pour créer vos propres projets. Selon la façon dont vous souhaitez procéder, choisissez Exécuter une démo ou Construire à partir de celle-ci.

   Pour plus d'informations sur les ressources qui seront créées pour cet exemple de projet, consultezAWS CloudFormation modèle et ressources supplémentaires.

   L'image suivante montre le graphique du flux de travail de la carte distribuée pour traiter les fichiers dans un exemple de projet S3 :



5. Choisissez Utiliser le modèle pour poursuivre votre sélection.
6. Effectuez l’une des actions suivantes :

- Si vous avez sélectionné Build on it, Step Functions crée le prototype de flux de travail pour l'exemple de projet que vous avez sélectionné. Step Functions ne déploie pas les ressources répertoriées dans la définition du flux de travail.

  Dans Workflow StudioMode de conception, glissez-déposez les états depuis le Navigateur d'états pour continuer à créer votre prototype de flux de travail. Vous pouvez également passer à un éditeur de code intégré similaire à VS Code pour mettre à jour la définition Amazon States Language (ASL) de votre machine à états dans la console Step Functions. Mode code Pour plus d'informations sur l'utilisation de Workflow Studio pour créer vos machines d'état, consultezUtilisation de Workflow Studio.

  > **Important**
  >
  > N'oubliez pas de mettre à jour l'espace réservé Amazon Resource Name (ARN) pour les ressources utilisées dans l'exemple de projet avant d'exécuter votre flux de travail.

- Si vous avez sélectionné Run a demo, Step Functions crée un exemple de projet en lecture seule qui utilise un AWS CloudFormation modèle pour déployer les AWS ressources répertoriées dans ce modèle sur votre. Compte AWS

  > **Tip**
  >
  > Pour afficher la définition de la machine à états de l'exemple de projet, choisissez Code.

  Lorsque vous êtes prêt, choisissez Déployer et exécuter pour déployer l'exemple de projet et créer les ressources.

  La création de ces ressources et des autorisations IAM associées peut prendre jusqu'à 10 minutes. Pendant le déploiement de vos ressources, vous pouvez ouvrir le lien CloudFormation Stack ID pour voir quelles ressources sont mises en service.

  Une fois que toutes les ressources de l'exemple de projet ont été créées, vous pouvez voir le nouvel exemple de projet répertorié sur la page State machines.

> **Important**
>
> Des frais standard peuvent s'appliquer pour chaque service utilisé dans le CloudFormation modèle.

Étape 2 : Exécuter la machine à états

Une fois que toutes les ressources ont été provisionnées et déployées, vous pouvez exécuter la machine d'état.

1. Sur la page State machines, choisissez votre exemple de projet.
2. Sur la page d'exemple de projet, choisissez Démarrer l'exécution.
3. Dans la boîte de dialogue Démarrer l'exécution, procédez comme suit :

   a. (Facultatif) Entrez les valeurs d'entrée au format JSON pour exécuter votre exemple de projet.

      Si vous avez choisi d'exécuter une démo, vous n'avez pas besoin de fournir d'entrée d'exécution.

      > **Note**
      >
      > Si le projet de démonstration que vous avez déployé contient des données d'entrée d'exécution préremplies, utilisez ces entrées pour exécuter la machine à états.

   b. Choisissez Start execution (Démarrer l'exécution).

   c. (Facultatif) La console Step Functions vous dirige vers une page intitulée avec votre ID d'exécution. Cette page est connue sous le nom de page Détails de l'exécution. Sur cette page, vous pouvez consulter les résultats de l'exécution au fur et à mesure que l'exécution progresse ou une fois celle-ci terminée.

      Une fois l'exécution terminée, choisissez des états individuels dans la vue graphique, puis choisissez les onglets individuels du Détails de l'étape volet pour afficher les détails de chaque état, y compris les entrées, les sorties et la définition respectivement.

- Pour plus de détails sur les informations d'exécution que vous pouvez consulter sur la page Détails de l'exécution, voirPage de détails d'exécution — Vue d'ensemble de l'interface.
- Pour plus d'informations sur l'affichage de l'exécution d'un état de carte distribuée dans la console, consultezExamen de Map Run.

d. (Facultatif) Vérifiez les résultats d'exécution exportés vers le compartiment Amazon S3. Ces résultats incluent des données, telles que les entrées et sorties d'exécution, l'ARN et le statut d'exécution. Pour de plus amples informations, veuillez consulter ResultWriter.

Former un modèle de Machine Learning

Cet exemple de projet montre comment utiliser SageMaker et AWS Step Functions entraîner un modèle d'apprentissage automatique et comment transformer par lots un ensemble de données de test.

Dans ce projet, Step Functions utilise une fonction Lambda pour ajouter un ensemble de données de test à un bucket Amazon S3. Il entraîne ensuite un modèle d'apprentissage automatique et effectue une transformation par lots en utilisant l'intégration SageMaker de services.

Pour plus d'informations sur les intégrations de services Step Functions SageMaker et sur celles-ci, consultez les rubriques suivantes :

- Utilisation AWS Step Functions avec d'autres services
- Gérez SageMaker avec Step Functions

> **Note**
>
> Cet exemple de projet peut entraîner des frais.
> Pour AWS les nouveaux utilisateurs, un niveau d'utilisation gratuit est disponible. Dans cette offre, les services sont gratuits en-dessous d'un certain niveau d'utilisation. Pour plus d'informations sur AWS les coûts et le niveau gratuit, consultez la section SageMaker Tarification.

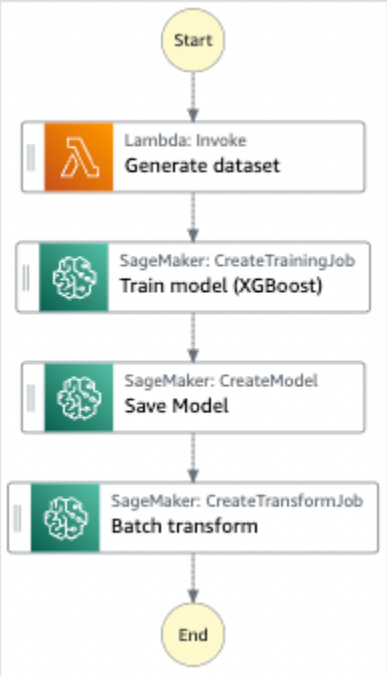
Étape 1 : créer la machine à états et provisionner les ressources

1. Ouvrez la console Step Functions et choisissez Create state machine.
2. Tapez **`Train a machine learning model`** dans le champ de recherche, puis choisissez Entraîner un modèle d'apprentissage automatique à partir des résultats de recherche renvoyés.
3. Choisissez Next (Suivant) pour continuer.
4. Step Functions répertorie les Services AWS éléments utilisés dans l'exemple de projet que vous avez sélectionné. Il montre également un graphique de flux de travail pour l'exemple de projet. Déployez ce projet sur votre site Compte AWS ou utilisez-le comme point de départ pour créer vos propres projets. Selon la façon dont vous souhaitez procéder, choisissez Exécuter une démo ou Construire à partir de celle-ci.

   Cet exemple de projet déploie les ressources suivantes :

   - Une AWS Lambda fonction
   - Un compartiment Amazon Simple Storage Service (Amazon S3)
   - Une machine AWS Step Functions étatique
   - Rôles associés AWS Identity and Access Management (IAM)

   L'image suivante montre le graphique du flux de travail de l'exemple de projet Train a machine learning model :



5. Choisissez Utiliser le modèle pour poursuivre votre sélection.
6. Effectuez l'une des actions suivantes :

   - Si vous avez sélectionné Build on it, Step Functions crée le prototype de flux de travail pour l'exemple de projet que vous avez sélectionné. Step Functions ne déploie pas les ressources répertoriées dans la définition du flux de travail.

     Dans Workflow StudioMode de conception, glissez-déposez les états depuis le Navigateur d'états pour continuer à créer votre prototype de flux de travail. Vous pouvez également passer à un éditeur de code intégré similaire à VS Code pour mettre à jour la définition Amazon States Language (ASL) de votre machine à états dans la console Step Functions. Mode code Pour plus d'informations sur l'utilisation de Workflow Studio pour créer vos machines d'état, consultezUtilisation de Workflow Studio.

     > **Important**
     >
     > N'oubliez pas de mettre à jour l'espace réservé Amazon Resource Name (ARN) pour les ressources utilisées dans l'exemple de projet avant d'exécuter votre flux de travail.

   - Si vous avez sélectionné Run a demo, Step Functions crée un exemple de projet en lecture seule qui utilise un AWS CloudFormation modèle pour déployer les AWS ressources répertoriées dans ce modèle sur votre. Compte AWS

     > **Tip**
     >
     > Pour afficher la définition de la machine à états de l'exemple de projet, choisissez Code.

     Lorsque vous êtes prêt, choisissez Déployer et exécuter pour déployer l'exemple de projet et créer les ressources.

     La création de ces ressources et des autorisations IAM associées peut prendre jusqu'à 10 minutes. Pendant le déploiement de vos ressources, vous pouvez ouvrir le lien CloudFormation Stack ID pour voir quelles ressources sont mises en service.

     Une fois que toutes les ressources de l'exemple de projet ont été créées, vous pouvez voir le nouvel exemple de projet répertorié sur la page State machines.

> **Important**
>
> Des frais standard peuvent s'appliquer pour chaque service utilisé dans le CloudFormation modèle.

Étape 2 : Exécuter la machine à états

1. Sur la page State machines, choisissez votre exemple de projet.
2. Sur la page d'exemple de projet, choisissez Démarrer l'exécution.
3. Dans la boîte de dialogue Démarrer l'exécution, procédez comme suit :

    1. (Facultatif) Pour identifier votre exécution, vous pouvez lui donner un nom dans le champ Nom. Par défaut, Step Functions génère automatiquement un nom d'exécution unique.

        > **Note**
        >
        > Step Functions vous permet de créer des noms pour les machines d'état, les exécutions et les activités, ainsi que des étiquettes contenant des caractères non ASCII. Ces noms non ASCII ne fonctionnent pas avec Amazon. CloudWatch Pour être sûr de pouvoir suivre CloudWatch les métriques, choisissez un nom qui utilise uniquement des caractères ASCII.

    2. (Facultatif) Dans la zone de saisie, entrez les valeurs d'entrée au format JSON pour exécuter votre flux de travail.

        Si vous avez choisi d'exécuter une démo, vous n'avez pas besoin de fournir d'entrée d'exécution.

        > **Note**
        >
        > Si le projet de démonstration que vous avez déployé contient des données d'entrée d'exécution préremplies, utilisez ces entrées pour exécuter la machine à états.

    3. Choisissez Start execution (Démarrer l'exécution).
    4. La console Step Functions vous dirige vers une page intitulée avec votre ID d'exécution. Cette page est connue sous le nom de page Détails de l'exécution. Sur cette page, vous pouvez

consulter les résultats de l'exécution au fur et à mesure que l'exécution progresse ou une fois celle-ci terminée.

Pour consulter les résultats de l'exécution, choisissez des états individuels dans la vue graphique, puis choisissez les onglets individuels du Détails de l'étape volet pour afficher les détails de chaque état, y compris les entrées, les sorties et la définition respectivement. Pour plus de détails sur les informations d'exécution que vous pouvez consulter sur la page Détails de l'exécution, voirPage de détails d'exécution — Vue d'ensemble de l'interface.

Exemple de code de machine d'état

Dans cet exemple de projet, la machine à états s'intègre à ces ressources SageMaker et AWS Lambda leur transmet des paramètres directement, et utilise un compartiment Amazon S3 pour la source et la sortie des données d'entraînement.

Parcourez cet exemple de machine à états pour découvrir comment Step Functions contrôle Lambda et. SageMaker

Pour plus d'informations sur la manière de AWS Step Functions contrôler d'autres AWS services, consultezUtilisation AWS Step Functions avec d'autres services.

```
{
  "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
    }
  }
}
```

Pour plus d'informations sur la configuration de l'IAM lors de l'utilisation de Step Functions avec d'autres AWS services, consultezPolitiques IAM pour les services intégrés.

Exemple IAM

Ces exemples de politiques AWS Identity and Access Management (IAM) générés par l'exemple de projet incluent le moindre privilège nécessaire pour exécuter la machine à états et les ressources

associées. Nous vous recommandons de n'inclure que les autorisations nécessaires dans vos politiques IAM.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "cloudwatch:PutMetricData",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:CreateLogGroup",
                "logs:DescribeLogStreams",
                "s3:GetObject",
                "s3:PutObject",
                "s3:ListBucket",
                "ecr:GetAuthorizationToken",
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage"
            ],
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

La politique suivante permet à la fonction Lambda d'ensemencer le compartiment Amazon S3 avec des exemples de données.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::trainandbatchtransform-s3bucket-1jn1le6gadwfz/*",
            "Effect": "Allow"
        }
    ]
}
```

Pour plus d'informations sur la configuration de l'IAM lors de l'utilisation de Step Functions avec d'autres AWS services, consultezPolitiques IAM pour les services intégrés.

Régler un modèle de Machine Learning

Cet exemple de projet montre comment SageMaker ajuster les hyperparamètres d'un modèle d'apprentissage automatique et transformer par lots un ensemble de données de test.

Dans ce projet, Step Functions utilise une fonction Lambda pour ajouter un ensemble de données de test à un bucket Amazon S3. Il crée ensuite une tâche de réglage des hyperparamètres à l'aide de l'intégration du SageMaker service. Il utilise ensuite une fonction Lambda pour extraire le chemin des données, enregistre le modèle de réglage, extrait le nom du modèle, puis exécute une tâche de transformation par lots pour effectuer une inférence. SageMaker

Pour plus d'informations sur les intégrations de services Step Functions SageMaker et sur celles-ci, consultez les rubriques suivantes :

- Utilisation AWS Step Functions avec d'autres services
- Gérez SageMaker avec Step Functions

> **Note**
>
> Cet exemple de projet peut entraîner des frais.
> Pour AWS les nouveaux utilisateurs, un niveau d'utilisation gratuit est disponible.
> Dans cette offre, les services sont gratuits en-dessous d'un certain niveau d'utilisation.
> Pour plus d'informations sur AWS les coûts et le niveau gratuit, consultez la section SageMakerTarification.

Étape 1 : créer la machine à états et provisionner les ressources

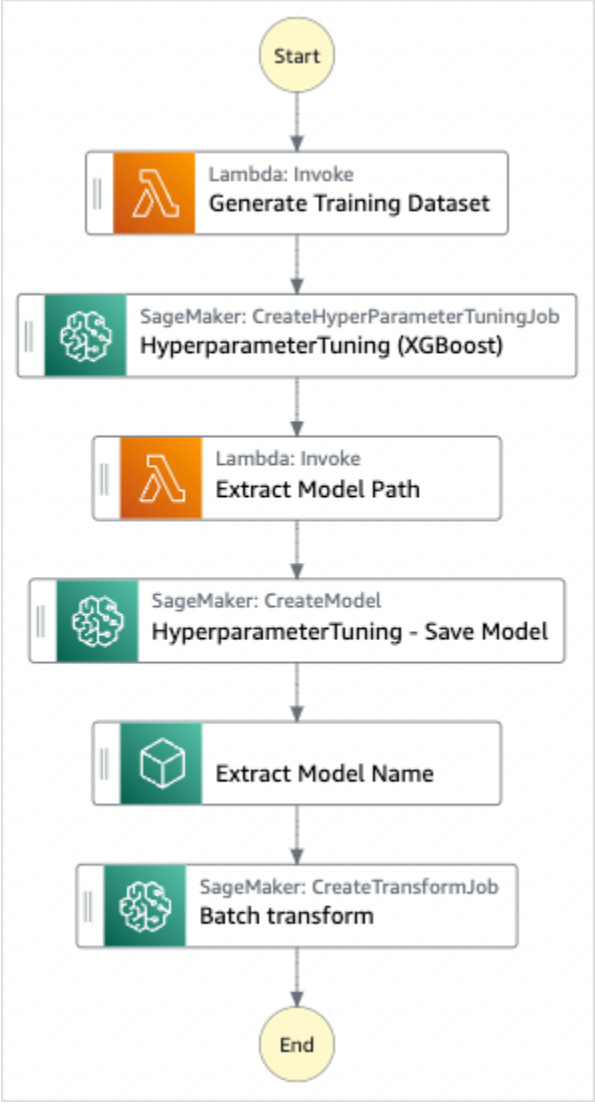
1. Ouvrez la console Step Functions et choisissez Create state machine.
2. Tapez **`Tune a machine learning model`** dans le champ de recherche, puis choisissez Tune a machine learning model à partir des résultats de recherche renvoyés.
3. Choisissez Next (Suivant) pour continuer.
4. Step Functions répertorie les Services AWS éléments utilisés dans l'exemple de projet que vous avez sélectionné. Il montre également un graphique de flux de travail pour l'exemple de projet.

Déployez ce projet sur votre site Compte AWS ou utilisez-le comme point de départ pour créer vos propres projets. Selon la façon dont vous souhaitez procéder, choisissez Exécuter une démo ou Construire à partir de celle-ci.

Cet exemple de projet déploie les ressources suivantes :

- Trois AWS Lambda fonctions
- Un compartiment Amazon Simple Storage Service (Amazon S3)
- Une machine AWS Step Functions étatique
- Rôles associés AWS Identity and Access Management (IAM)

L'image suivante montre le graphique du flux de travail de l'exemple de projet Tune a machine learning model :



5. Choisissez Utiliser le modèle pour poursuivre votre sélection.
6. Effectuez l'une des actions suivantes :

   - Si vous avez sélectionné Build on it, Step Functions crée le prototype de flux de travail pour l'exemple de projet que vous avez sélectionné. Step Functions ne déploie pas les ressources répertoriées dans la définition du flux de travail.

     Dans Workflow StudioMode de conception, glissez-déposez les états depuis le Navigateur d'états pour continuer à créer votre prototype de flux de travail. Vous pouvez également passer à un éditeur de code intégré similaire à VS Code pour mettre à jour la définition Amazon States Language (ASL) de votre machine à états dans la console Step Functions. Mode code Pour plus d'informations sur l'utilisation de Workflow Studio pour créer vos machines d'état, consultezUtilisation de Workflow Studio.

     > **Important**
     >
     > N'oubliez pas de mettre à jour l'espace réservé Amazon Resource Name (ARN) pour les ressources utilisées dans l'exemple de projet avant d'exécuter votre flux de travail.

   - Si vous avez sélectionné Run a demo, Step Functions crée un exemple de projet en lecture seule qui utilise un AWS CloudFormation modèle pour déployer les AWS ressources répertoriées dans ce modèle sur votre. Compte AWS

     > **Tip**
     >
     > Pour afficher la définition de la machine à états de l'exemple de projet, choisissez Code.

     Lorsque vous êtes prêt, choisissez Déployer et exécuter pour déployer l'exemple de projet et créer les ressources.

     La création de ces ressources et des autorisations IAM associées peut prendre jusqu'à 10 minutes. Pendant le déploiement de vos ressources, vous pouvez ouvrir le lien CloudFormation Stack ID pour voir quelles ressources sont mises en service.

     Une fois que toutes les ressources de l'exemple de projet ont été créées, vous pouvez voir le nouvel exemple de projet répertorié sur la page State machines.

> **Important**
>
> Des frais standard peuvent s'appliquer pour chaque service utilisé dans le CloudFormation modèle.

Étape 2 : Exécuter la machine à états

1. Sur la page State machines, choisissez votre exemple de projet.
2. Sur la page d'exemple de projet, choisissez Démarrer l'exécution.
3. Dans la boîte de dialogue Démarrer l'exécution, procédez comme suit :

    1. (Facultatif) Pour identifier votre exécution, vous pouvez lui donner un nom dans le champ Nom. Par défaut, Step Functions génère automatiquement un nom d'exécution unique.

        > **Note**
        >
        > Step Functions vous permet de créer des noms pour les machines d'état, les exécutions et les activités, ainsi que des étiquettes contenant des caractères non ASCII. Ces noms non ASCII ne fonctionnent pas avec Amazon. CloudWatch Pour être sûr de pouvoir suivre CloudWatch les métriques, choisissez un nom qui utilise uniquement des caractères ASCII.

    2. (Facultatif) Dans la zone de saisie, entrez les valeurs d'entrée au format JSON pour exécuter votre flux de travail.

        Si vous avez choisi d'exécuter une démo, vous n'avez pas besoin de fournir d'entrée d'exécution.

        > **Note**
        >
        > Si le projet de démonstration que vous avez déployé contient des données d'entrée d'exécution préremplies, utilisez ces entrées pour exécuter la machine à états.

    3. Choisissez Start execution (Démarrer l'exécution).
    4. La console Step Functions vous dirige vers une page intitulée avec votre ID d'exécution. Cette page est connue sous le nom de page Détails de l'exécution. Sur cette page, vous pouvez

consulter les résultats de l'exécution au fur et à mesure que l'exécution progresse ou une fois celle-ci terminée.

Pour consulter les résultats de l'exécution, choisissez des états individuels dans la vue graphique, puis choisissez les onglets individuels du Détails de l'étape volet pour afficher les détails de chaque état, y compris les entrées, les sorties et la définition respectivement. Pour plus de détails sur les informations d'exécution que vous pouvez consulter sur la page Détails de l'exécution, voirPage de détails d'exécution — Vue d'ensemble de l'interface.

Exemple de code de machine d'état

Dans cet exemple de projet, la machine à états s'intègre à ces ressources SageMaker et AWS Lambda leur transmet des paramètres directement, et utilise un compartiment Amazon S3 pour la source et la sortie des données d'entraînement.

Parcourez cet exemple de machine à états pour découvrir comment Step Functions contrôle Lambda et. SageMaker

Pour plus d'informations sur la manière de AWS Step Functions contrôler d'autres AWS services, consultezUtilisation AWS Step Functions avec d'autres services.

```
{
    "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
        }
    }
}
```

Pour plus d'informations sur la configuration de l'IAM lors de l'utilisation de Step Functions avec d'autres AWS services, consultezPolitiques IAM pour les services intégrés.

Exemples IAM

Ces exemples de politiques AWS Identity and Access Management (IAM) générés par l'exemple de projet incluent le moindre privilège nécessaire pour exécuter la machine à états et les ressources associées. Nous vous recommandons de n'inclure que les autorisations nécessaires dans vos politiques IAM.

La politique IAM suivante est attachée à la machine d'état et permet à l'exécution de la machine d'état d'accéder aux ressources nécessaires SageMaker, Lambda et Amazon S3.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "sagemaker:CreateHyperParameterTuningJob",
                "sagemaker:DescribeHyperParameterTuningJob",
                "sagemaker:StopHyperParameterTuningJob",
                "sagemaker:ListTags",
                "sagemaker:CreateModel",
                "sagemaker:CreateTransformJob",
                "iam:PassRole"
            ],
```

```
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": [
                "arn:aws:lambda:us-west-2:012345678912:function:StepFunctionsSample-
SageMa-LambdaForDataGeneration-1TF67BUE5A12U",
                "arn:aws:lambda:us-west-2:012345678912:function:StepFunctionsSample-
SageM-LambdaToExtractModelPath-V0R37CVARUS9",
                "arn:aws:lambda:us-west-2:012345678912:function:StepFunctionsSample-
SageM-LambdaToExtractModelName-8FUOB30SM5EM"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "events:PutTargets",
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": [
                "arn:aws:events:*:*:rule/
StepFunctionsGetEventsForSageMakerTrainingJobsRule",
                "arn:aws:events:*:*:rule/
StepFunctionsGetEventsForSageMakerTransformJobsRule",
                "arn:aws:events:*:*:rule/
StepFunctionsGetEventsForSageMakerTuningJobsRule"
            ],
            "Effect": "Allow"
        }
    ]
}
```

La politique IAM suivante est référencée dans les `HyperparameterTuning` champs `TrainingJobDefinition` et de l'`HyperparameterTuning`État.

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

```
            "Action": [
                "cloudwatch:PutMetricData",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:CreateLogGroup",
                "logs:DescribeLogStreams",
                "ecr:GetAuthorizationToken",
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage",
                "sagemaker:DescribeHyperParameterTuningJob",
                "sagemaker:StopHyperParameterTuningJob",
                "sagemaker:ListTags"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "s3:GetObject",
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::stepfunctionssample-sagemak-
bucketformodelanddata-80fblmdlcs9f/*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": "arn:aws:s3:::stepfunctionssample-sagemak-
bucketformodelanddata-80fblmdlcs9f",
            "Effect": "Allow"
        }
    ]
}
```

La politique IAM suivante permet à la fonction Lambda d'amorcer le compartiment Amazon S3 avec des exemples de données.

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

```
        {
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::stepfunctionssample-sagemak-
bucketformodelanddata-80fblmdlcs9f/*",
            "Effect": "Allow"
        }
    ]
}
```

Pour plus d'informations sur la configuration de l'IAM lors de l'utilisation de Step Functions avec d'autres AWS services, consultezPolitiques IAM pour les services intégrés.

Traitement de gros volumes de messages depuis Amazon SQS (Express Workflows)

Cet exemple de projet montre comment utiliser un flux de travail AWS Step Functions express pour traiter des messages ou des données provenant d'une source d'événements volumineuse, telle qu'Amazon Simple Queue Service (Amazon SQS). Comme Express Workflows peut être démarré à un rythme très élevé, il convient parfaitement pour le traitement d'événements à volume élevé ou les charges de travail de données en continu.

Voici deux méthodes couramment utilisées pour exécuter votre machine d'état à partir d'une source d'événement :

- Configurez une règle Amazon CloudWatch Events pour démarrer l'exécution d'une machine à états chaque fois que la source d'événements émet un événement. Pour plus d'informations, consultez la section Création d'une règle d' CloudWatch événements déclenchant un événement.
- Mappez la source d'événement à une fonction Lambda et écrivez le code de fonction pour exécuter votre machine d'état. La AWS Lambda fonction est invoquée chaque fois que votre source d'événement émet un événement, ce qui déclenche à son tour une exécution par machine à états. Pour plus d'informations, consultez la section Utilisation AWS Lambda avec Amazon SQS.

Cet exemple de projet utilise la deuxième méthode pour démarrer une exécution chaque fois que la file d'attente Amazon SQS envoie un message. Vous pouvez utiliser une configuration similaire pour déclencher l'exécution d'Express Workflows à partir d'autres sources d'événements, telles qu'Amazon Simple Storage Service (Amazon S3), Amazon DynamoDB et Amazon Kinesis.

Pour plus d'informations sur les intégrations des services Express Workflows et Step Functions, consultez les rubriques suivantes :

- Flux de travail standard ou express
- Utilisation AWS Step Functions avec d'autres services
- Quotas

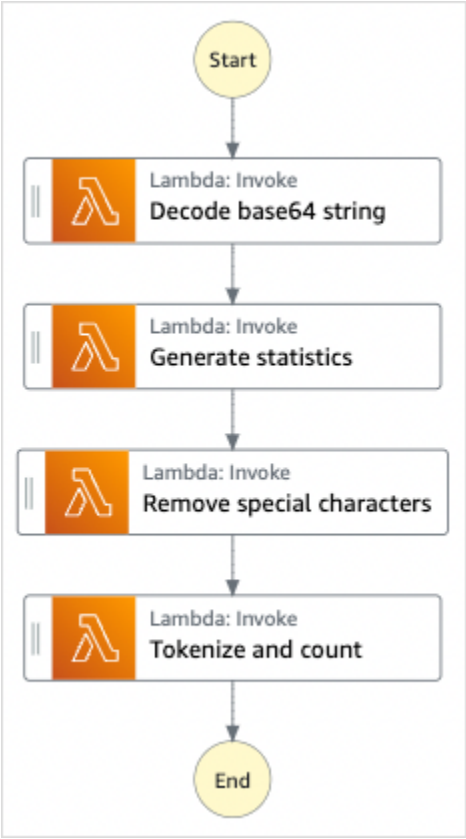
Étape 1 : créer la machine à états et provisionner les ressources

1. Ouvrez la console Step Functions et choisissez Create state machine.
2. Tapez **`Process high-volume messages from SQS`** dans la zone de recherche, puis choisissez Traiter les messages volumineux provenant de SQS dans les résultats de recherche renvoyés.
3. Choisissez Next (Suivant) pour continuer.
4. Step Functions répertorie les Services AWS éléments utilisés dans l'exemple de projet que vous avez sélectionné. Il montre également un graphique de flux de travail pour l'exemple de projet. Déployez ce projet sur votre site Compte AWS ou utilisez-le comme point de départ pour créer vos propres projets. Selon la façon dont vous souhaitez procéder, choisissez Exécuter une démo ou Construire à partir de celle-ci.

   Cet exemple de projet déploie les ressources suivantes :

   - Fonction Four Lambda
   - Une file d'attente Amazon SQS
   - Une machine AWS Step Functions étatique
   - Rôles associés AWS Identity and Access Management (IAM)

   L'image suivante montre le graphique du flux de travail pour l'exemple de projet Traiter les messages volumineux à partir de SQS :



5. Choisissez Utiliser le modèle pour poursuivre votre sélection.
6. Effectuez l'une des actions suivantes :

   - Si vous avez sélectionné Build on it, Step Functions crée le prototype de flux de travail pour l'exemple de projet que vous avez sélectionné. Step Functions ne déploie pas les ressources répertoriées dans la définition du flux de travail.

     Dans Workflow StudioMode de conception, glissez-déposez les états depuis le Navigateur d'états pour continuer à créer votre prototype de flux de travail. Vous pouvez également passer à un éditeur de code intégré similaire à VS Code pour mettre à jour la définition Amazon States Language (ASL) de votre machine à états dans la console Step Functions. Mode code Pour plus d'informations sur l'utilisation de Workflow Studio pour créer vos machines d'état, consultezUtilisation de Workflow Studio.

     > **Important**
     >
     > N'oubliez pas de mettre à jour l'espace réservé Amazon Resource Name (ARN) pour les ressources utilisées dans l'exemple de projet avant d'exécuter votre flux de travail.

- Si vous avez sélectionné Run a demo, Step Functions crée un exemple de projet en lecture seule qui utilise un AWS CloudFormation modèle pour déployer les AWS ressources répertoriées dans ce modèle sur votre. Compte AWS

  > **Tip**
  >
  > Pour afficher la définition de la machine à états de l'exemple de projet, choisissez Code.

  Lorsque vous êtes prêt, choisissez Déployer et exécuter pour déployer l'exemple de projet et créer les ressources.

  La création de ces ressources et des autorisations IAM associées peut prendre jusqu'à 10 minutes. Pendant le déploiement de vos ressources, vous pouvez ouvrir le lien CloudFormation Stack ID pour voir quelles ressources sont mises en service.

  Une fois que toutes les ressources de l'exemple de projet ont été créées, vous pouvez voir le nouvel exemple de projet répertorié sur la page State machines.

  > **Important**
  >
  > Des frais standard peuvent s'appliquer pour chaque service utilisé dans le CloudFormation modèle.

Étape 2 : Déclencher l'exécution de la machine à états

1. Ouvrez la console Amazon SQS.
2. Sélectionnez la file d'attente qui a été créée par l'exemple de projet.

   Le nom sera similaire à Example-SQSQueue-wJalrXUtnFEMI.
3. Dans la liste Actions en file d'attente, sélectionnez Envoyer un message.
4. Utilisez le bouton Copier pour copier le message suivant et, dans la fenêtre Envoyer un message saisissez-le et sélectionnez le bouton Envoyer un message .

> **Note**
>
> Dans cet exemple de message, la ligne `input:` a été formatée avec des sauts de ligne pour s'adapter à la page. Utilisez le bouton Copier ou assurez-vous qu'il est entré sous la forme d'une seule ligne sans interruption.

```
{
      "input":
 "QW5kIGxpa2UgdGhlIGJhc2VsZXNzIGZhYnJpYyBvZiB0aGlzIHZpc2lvbiwgVGhlIGNsb3VkLWNhcHBlZCB0b3dlc

  91cyBwYWxhY2VzLCBUaGUgc29sZW1uIHRlbXBsZXMsIHRoZSBncmVhdCBnbG9iZSBpdHNlbGbigJQgWWVhLCBhbGw

  ZXJpdOKAlHNoYWxsIGRpc3NvbHZlLCBBbmQgbGlrZSB0aGlzIGluc3Vic3RhbnRpYWwgcGFnZWFudCBmYWRlZCwgT

  FjayBiZWhpbmQuIFdlIGFyZSBzdWNoIHN0dWZmIEFzIGRyZWFtcyBhcmUgbWFkZSBvbiwgYW5kIG91ciBsaXR0bGU

  ZGVkIHdpdGggYSBzbGVlcC4gU2lyLCBJIGFtIHZleGVkLiBCZWFyIHdpdGggbXkgd2Vha25lc3MuIE15IG9sZCBic

  xlZC4gQmUgbm90IGRpc3R1cmJlZCB3aXRoIG15IGluZmlybWl0eS4gSWYgeW91IGJlIHBsZWFzZWQsIHJldGlyZSB

  QW5kIHRoZXJlIHJlcG9zZS4gQSB0dXJuIG9yIHR3byBJ4oCZbGwgd2FsayBUbyBzdGlsbCBteSBzdWF0aW5nIG1pb
}
```

5. Choisissez Fermer.
6. Ouvrez la console Step Functions.
7. Accédez à votre groupe de CloudWatch journaux Amazon Logs et inspectez les journaux. Le nom du groupe de journaux ressemblera à example- ExpressLogGroup -WjalRxUtnFemi.

Exemple de code de fonction Lambda

Voici un code de fonction Lambda qui montre comment la fonction Lambda initiatrice démarre une exécution de machine à états à l'aide du SDK. AWS

```
import boto3

def lambda_handler(event, context):
    message_body = event['Records'][0]['body']
    client = boto3.client('stepfunctions')
```

```
    response = client.start_execution(
        stateMachineArn='${ExpressStateMachineArn}',
        input=message_body
    )
```

Exemple de code de machine d'état

Le workflow express de cet exemple de projet consiste en un ensemble de fonctions Lambda pour le traitement de texte.

Pour plus d'informations sur la manière de AWS Step Functions contrôler d'autres AWS services, consultezUtilisation AWS Step Functions avec d'autres services.

```
{
  "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
    }
  }
}
```

Exemple IAM

Cet exemple de politique AWS Identity and Access Management (IAM) généré par l'exemple de projet inclut le minimum de privilèges nécessaires pour exécuter la machine à états et les ressources associées. Nous vous recommandons de n'inclure que les autorisations nécessaires dans vos politiques IAM.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": [
                "arn:aws:lambda:us-east-1:123456789012:function:example-
Base64DecodeLambda-wJalrXUtnFEMI",
                "arn:aws:lambda:us-east-1:123456789012:function:example-
StringCleanerLambda-je7MtGbClwBF",
                "arn:aws:lambda:us-east-1:123456789012:function:example-
TokenizerCounterLambda-wJalrXUtnFEMI",
                "arn:aws:lambda:us-east-1:123456789012:function:example-
GenerateStatsLambda-je7MtGbClwBF"
            ],
            "Effect": "Allow"
```

```
        }
    ]
}
```

La politique suivante garantit que les autorisations pour CloudWatch les journaux sont suffisantes.

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

Pour plus d'informations sur la configuration de l'IAM lors de l'utilisation de Step Functions avec d'autres AWS services, consultezPolitiques IAM pour les services intégrés.

Exemple de point de contrôle sélectif (workflows express)

Cet exemple de projet montre comment combiner les workflows Standard et Express en exécutant un workflow de commerce électronique simulé qui effectue un point de contrôle sélectif. Le déploiement de cet exemple de projet crée une machine à états Standard Workflows, une machine à états Express Workflows imbriquée, une AWS Lambda fonction, une file d'attente Amazon Simple Queue Service (Amazon SQS) et une rubrique Amazon Simple Notification Service (Amazon SNS).

Pour plus d'informations sur les flux de travail Express, les flux de travail imbriqués et les intégrations des services Step Functions, consultez les pages suivantes :

- Flux de travail standard ou express
- Démarrer les exécutions de flux de travail à partir d'un état de tâche
- Utilisation AWS Step Functions avec d'autres services

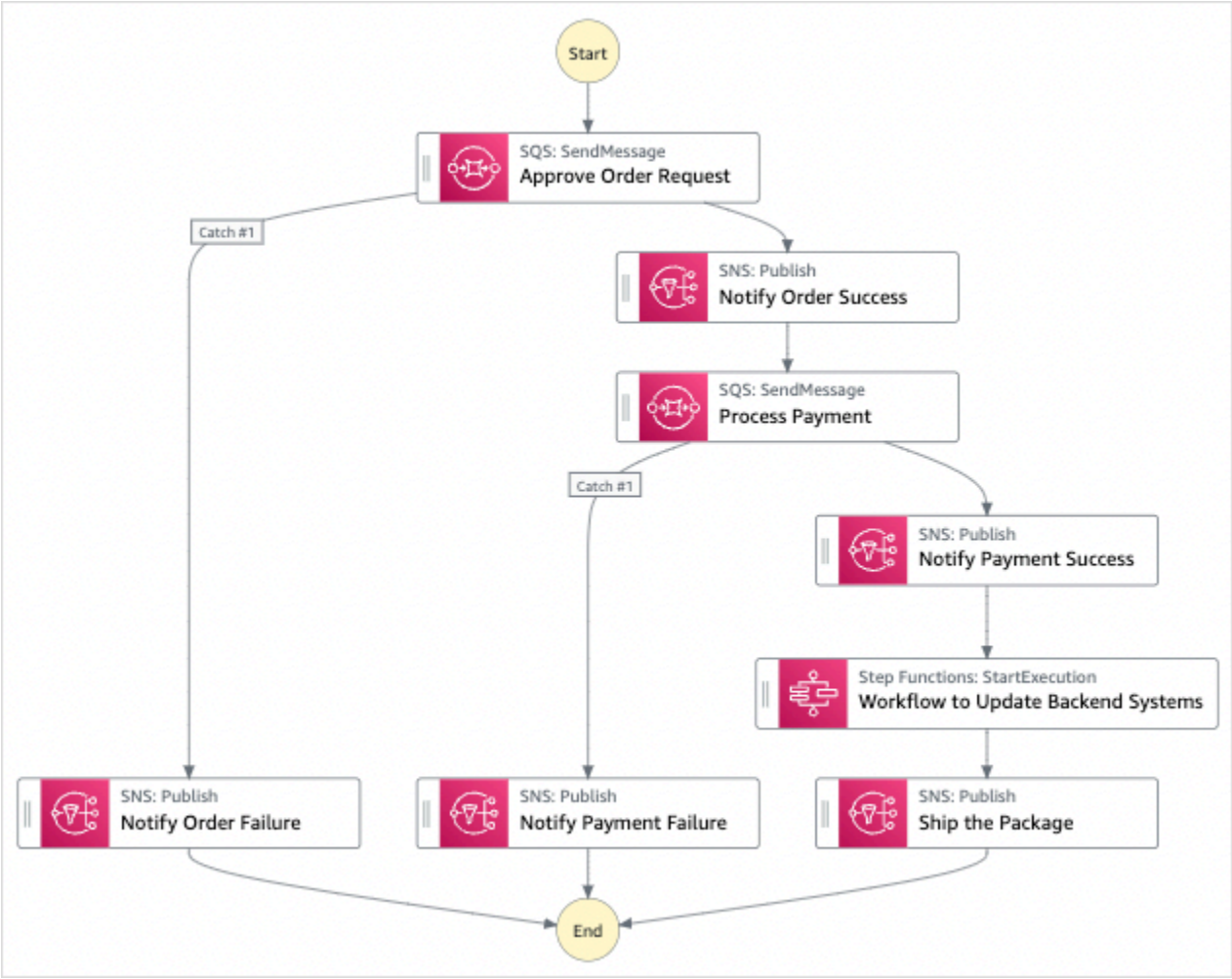
Étape 1 : créer la machine à états et provisionner les ressources

1. Ouvrez la console Step Functions et choisissez Create state machine.
2. Tapez **`Selective checkpointing example`** dans la zone de recherche, puis choisissez Exemple de point de contrôle sélectif dans les résultats de recherche renvoyés.
3. Choisissez Next (Suivant) pour continuer.
4. Step Functions répertorie les Services AWS éléments utilisés dans l'exemple de projet que vous avez sélectionné. Il montre également un graphique de flux de travail pour l'exemple de projet. Déployez ce projet sur votre site Compte AWS ou utilisez-le comme point de départ pour créer vos propres projets. Selon la façon dont vous souhaitez procéder, choisissez Exécuter une démo ou Construire à partir de celle-ci.

   Cet exemple de projet déploie les ressources suivantes :

   - Une AWS Lambda fonction
   - Une file d'attente Amazon SQS
   - Une rubrique Amazon SNS
   - Une machine à AWS Step Functions états de type Standard
   - Une machine à états Step Functions de type Express
   - Rôles associés AWS Identity and Access Management (IAM)

   L'image suivante montre le graphique du flux de travail pour l'exemple de projet de point de contrôle sélectif :



5. Choisissez Utiliser le modèle pour poursuivre votre sélection.
6. Effectuez l'une des actions suivantes :

   - Si vous avez sélectionné Build on it, Step Functions crée le prototype de flux de travail pour l'exemple de projet que vous avez sélectionné. Step Functions ne déploie pas les ressources répertoriées dans la définition du flux de travail.

     Dans Workflow StudioMode de conception, glissez-déposez les états depuis le Navigateur d'états pour continuer à créer votre prototype de flux de travail. Vous pouvez également passer à un éditeur de code intégré similaire à VS Code pour mettre à jour la définition Amazon States Language (ASL) de votre machine à états dans la console Step Functions. Mode code Pour plus d'informations sur l'utilisation de Workflow Studio pour créer vos machines d'état, consultezUtilisation de Workflow Studio.

> **Important**
>
> N'oubliez pas de mettre à jour l'espace réservé Amazon Resource Name (ARN) pour les ressources utilisées dans l'exemple de projet avant d'exécuter votre flux de travail.

- Si vous avez sélectionné Run a demo, Step Functions crée un exemple de projet en lecture seule qui utilise un AWS CloudFormation modèle pour déployer les AWS ressources répertoriées dans ce modèle sur votre. Compte AWS

  > **Tip**
  >
  > Pour afficher la définition de la machine à états de l'exemple de projet, choisissez Code.

  Lorsque vous êtes prêt, choisissez Déployer et exécuter pour déployer l'exemple de projet et créer les ressources.

  La création de ces ressources et des autorisations IAM associées peut prendre jusqu'à 10 minutes. Pendant le déploiement de vos ressources, vous pouvez ouvrir le lien CloudFormation Stack ID pour voir quelles ressources sont mises en service.

  Une fois que toutes les ressources de l'exemple de projet ont été créées, vous pouvez voir le nouvel exemple de projet répertorié sur la page State machines.

  > **Important**
  >
  > Des frais standard peuvent s'appliquer pour chaque service utilisé dans le CloudFormation modèle.

Une fois les ressources de l'exemple de projet déployées, procédez comme suit.

Étape 2 : Exécuter la machine à états

1. Sur la page State machines, choisissez votre exemple de projet.
2. Sur la page d'exemple de projet, choisissez Démarrer l'exécution.

3. Dans la boîte de dialogue Démarrer l'exécution, procédez comme suit :

   1. (Facultatif) Pour identifier votre exécution, vous pouvez lui donner un nom dans le champ Nom. Par défaut, Step Functions génère automatiquement un nom d'exécution unique.

      > **Note**
      >
      > Step Functions vous permet de créer des noms pour les machines d'état, les exécutions et les activités, ainsi que des étiquettes contenant des caractères non ASCII. Ces noms non ASCII ne fonctionnent pas avec Amazon. CloudWatch Pour être sûr de pouvoir suivre CloudWatch les métriques, choisissez un nom qui utilise uniquement des caractères ASCII.

   2. (Facultatif) Dans la zone de saisie, entrez les valeurs d'entrée au format JSON pour exécuter votre flux de travail.

      Si vous avez choisi d'exécuter une démo, vous n'avez pas besoin de fournir d'entrée d'exécution.

      > **Note**
      >
      > Si le projet de démonstration que vous avez déployé contient des données d'entrée d'exécution préremplies, utilisez ces entrées pour exécuter la machine à états.

   3. Choisissez Start execution (Démarrer l'exécution).
   4. La console Step Functions vous dirige vers une page intitulée avec votre ID d'exécution. Cette page est connue sous le nom de page Détails de l'exécution. Sur cette page, vous pouvez consulter les résultats de l'exécution au fur et à mesure que l'exécution progresse ou une fois celle-ci terminée.

      Pour consulter les résultats de l'exécution, choisissez des états individuels dans la vue graphique, puis choisissez les onglets individuels du Détails de l'étape volet pour afficher les détails de chaque état, y compris les entrées, les sorties et la définition respectivement. Pour plus de détails sur les informations d'exécution que vous pouvez consulter sur la page Détails de l'exécution, voirPage de détails d'exécution — Vue d'ensemble de l'interface.

4. Accédez à votre groupe de CloudWatch journaux Logs et inspectez les journaux. Le nom du groupe de journaux ressemblera à example- ExpressLogGroup -WjalRxUtnFemi.

Exemple de code de machine d'état pour le parent (Standard Workflows)

Dans cet exemple de projet, la machine à états s'intègre aux flux de travail Amazon SQS, Amazon SNS et Step Functions Express.

Parcourez cet exemple de machine à états pour découvrir comment Step Functions traite les entrées provenant d'Amazon SQS et Amazon SNS, puis utilise une machine d'état Express Workflows imbriquée pour mettre à jour les systèmes principaux.

Pour plus d'informations sur la manière de AWS Step Functions contrôler d'autres AWS services, consultezUtilisation AWS Step Functions avec d'autres services.

```
{
    "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
        }
    }
}
```

Exemple de rôle IAM pour la machine à états parent

Ces exemples de politiques AWS Identity and Access Management (IAM) générés par l'exemple de projet incluent le moindre privilège nécessaire pour exécuter la machine à états et les ressources associées. Nous vous recommandons de n'inclure que les autorisations nécessaires dans vos politiques IAM.

Politique Amazon SNS :

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "sns:Publish"
            ],
            "Resource": "arn:aws:sns:us-east-1:123456789012:Checkpoint-SNSTopic-
wJalrXUtnFEMI",
            "Effect": "Allow"
        }
    ]
}
```

Politique Amazon SQS :

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "sqs:SendMessage"
            ],
            "Resource": "arn:aws:sqs:us-east-1:123456789012:Checkpoint-SQSQueue-
je7MtGbClwBF",
            "Effect": "Allow"
        }
    ]
}
```

Stratégie d'exécution des états :

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

Exemple de code machine d'état pour la machine d'état imbriquée (workflows express)

La machine d'état de cet exemple de projet met à jour les informations d'arrière-plan lorsqu'elle est appelée par la machine d'état parent.

Parcourez cet exemple de machine à états pour découvrir comment Step Functions met à jour les différents composants des systèmes dorsaux de commerce électronique fictifs.

Pour plus d'informations sur la manière de AWS Step Functions contrôler d'autres AWS services, consultezUtilisation AWS Step Functions avec d'autres services.


```
{
  "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
    }
  }
}
```

Exemple de rôle IAM pour Child State Machine

Cet exemple de politique AWS Identity and Access Management (IAM) généré par l'exemple de projet inclut le minimum de privilèges nécessaires pour exécuter la machine à états et les ressources associées. Nous vous recommandons de n'inclure que les autorisations nécessaires dans vos politiques IAM.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": [
```

```
                "arn:aws:lambda:us-east-1:123456789012:function:Example-
UpdateDatabaseLambdaFunction-wJalrXUtnFEMI"
            ],
            "Effect": "Allow"
        }
    ]
}
```

La politique suivante garantit que les autorisations sont suffisantes pour les CloudWatch journaux.

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

Pour plus d'informations sur la configuration de l'IAM lors de l'utilisation de Step Functions avec d'autres AWS services, consultezPolitiques IAM pour les services intégrés.

Création d'un AWS CodeBuild projet (CodeBuild, Amazon SNS)

Cet exemple de projet montre comment AWS Step Functions créer un AWS CodeBuild projet, exécuter des tests, puis envoyer une notification Amazon SNS.

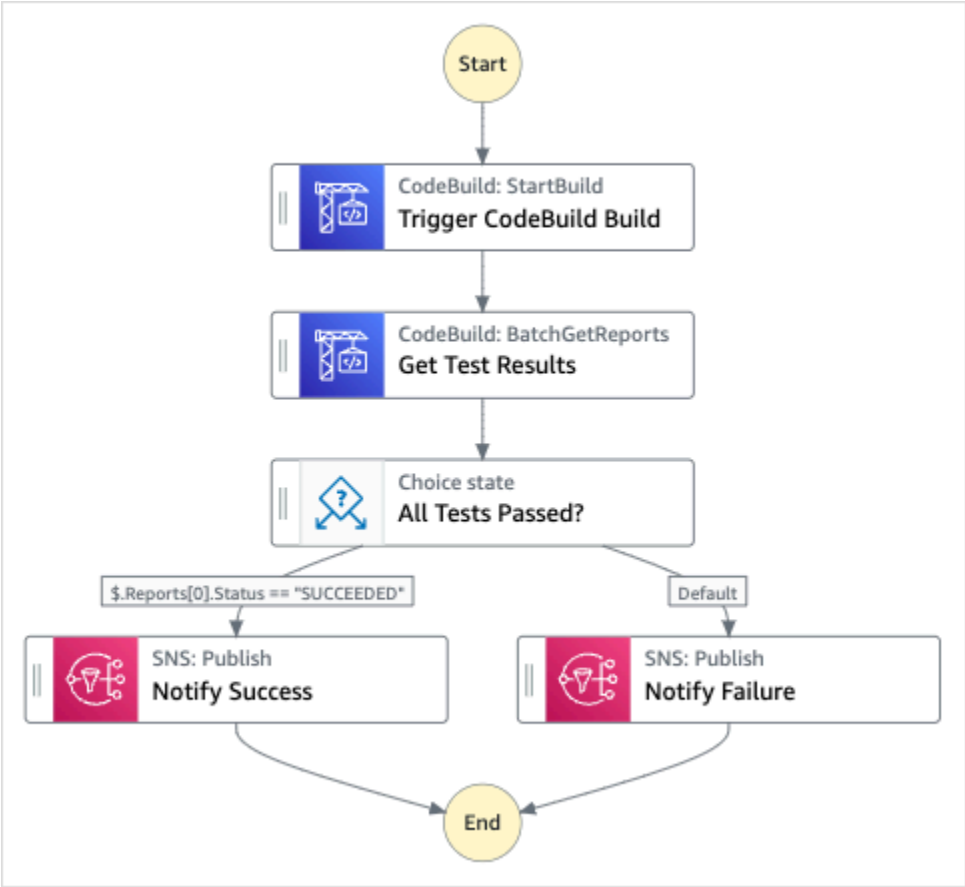
Étape 1 : créer la machine à états et provisionner les ressources

1. Ouvrez la console Step Functions et choisissez Create state machine.
2. Tapez **`Start a CodeBuild build`** dans la zone de recherche, puis choisissez Démarrer une CodeBuild construction à partir des résultats de recherche renvoyés.
3. Choisissez Next (Suivant) pour continuer.
4. Step Functions répertorie les Services AWS éléments utilisés dans l'exemple de projet que vous avez sélectionné. Il montre également un graphique de flux de travail pour l'exemple de projet. Déployez ce projet sur votre site Compte AWS ou utilisez-le comme point de départ pour créer vos propres projets. Selon la façon dont vous souhaitez procéder, choisissez Exécuter une démo ou Construire à partir de celle-ci.

   Cet exemple de projet déploie les ressources suivantes :

   - Et AWS CodeBuild construisez
   - Une rubrique Amazon SNS
   - Une machine AWS Step Functions étatique
   - Rôles associés AWS Identity and Access Management (IAM)

   L'image suivante montre le graphique du flux de travail pour le projet d'exemple Start a CodeBuild build :



5. Choisissez Utiliser le modèle pour poursuivre votre sélection.
6. Effectuez l'une des actions suivantes :

   - Si vous avez sélectionné Build on it, Step Functions crée le prototype de flux de travail pour l'exemple de projet que vous avez sélectionné. Step Functions ne déploie pas les ressources répertoriées dans la définition du flux de travail.

     Dans Workflow StudioMode de conception, glissez-déposez les états depuis le Navigateur d'états pour continuer à créer votre prototype de flux de travail. Vous pouvez également passer à un éditeur de code intégré similaire à VS Code pour mettre à jour la définition Amazon States Language (ASL) de votre machine à états dans la console Step Functions. Mode code Pour plus d'informations sur l'utilisation de Workflow Studio pour créer vos machines d'état, consultezUtilisation de Workflow Studio.

> **Important**
>
> N'oubliez pas de mettre à jour l'espace réservé Amazon Resource Name (ARN) pour les ressources utilisées dans l'exemple de projet avant d'exécuter votre flux de travail.

- Si vous avez sélectionné Run a demo, Step Functions crée un exemple de projet en lecture seule qui utilise un AWS CloudFormation modèle pour déployer les AWS ressources répertoriées dans ce modèle sur votre. Compte AWS

  > **Tip**
  >
  > Pour afficher la définition de la machine à états de l'exemple de projet, choisissez Code.

  Lorsque vous êtes prêt, choisissez Déployer et exécuter pour déployer l'exemple de projet et créer les ressources.

  La création de ces ressources et des autorisations IAM associées peut prendre jusqu'à 10 minutes. Pendant le déploiement de vos ressources, vous pouvez ouvrir le lien CloudFormation Stack ID pour voir quelles ressources sont mises en service.

  Une fois que toutes les ressources de l'exemple de projet ont été créées, vous pouvez voir le nouvel exemple de projet répertorié sur la page State machines.

  > **Important**
  >
  > Des frais standard peuvent s'appliquer pour chaque service utilisé dans le CloudFormation modèle.

Étape 2 : Exécuter la machine à états

1. Sur la page State machines, choisissez votre exemple de projet.
2. Sur la page d'exemple de projet, choisissez Démarrer l'exécution.
3. Dans la boîte de dialogue Démarrer l'exécution, procédez comme suit :

1. (Facultatif) Pour identifier votre exécution, vous pouvez lui donner un nom dans le champ Nom. Par défaut, Step Functions génère automatiquement un nom d'exécution unique.

   > **Note**
   >
   > Step Functions vous permet de créer des noms pour les machines d'état, les exécutions et les activités, ainsi que des étiquettes contenant des caractères non ASCII. Ces noms non ASCII ne fonctionnent pas avec Amazon. CloudWatch Pour être sûr de pouvoir suivre CloudWatch les métriques, choisissez un nom qui utilise uniquement des caractères ASCII.

2. (Facultatif) Dans la zone de saisie, entrez les valeurs d'entrée au format JSON pour exécuter votre flux de travail.

   Si vous avez choisi d'exécuter une démo, vous n'avez pas besoin de fournir d'entrée d'exécution.

   > **Note**
   >
   > Si le projet de démonstration que vous avez déployé contient des données d'entrée d'exécution préremplies, utilisez ces entrées pour exécuter la machine à états.

3. Choisissez Start execution (Démarrer l'exécution).
4. La console Step Functions vous dirige vers une page intitulée avec votre ID d'exécution. Cette page est connue sous le nom de page Détails de l'exécution. Sur cette page, vous pouvez consulter les résultats de l'exécution au fur et à mesure que l'exécution progresse ou une fois celle-ci terminée.

   Pour consulter les résultats de l'exécution, choisissez des états individuels dans la vue graphique, puis choisissez les onglets individuels du Détails de l'étape volet pour afficher les détails de chaque état, y compris les entrées, les sorties et la définition respectivement. Pour plus de détails sur les informations d'exécution que vous pouvez consulter sur la page Détails de l'exécution, voirPage de détails d'exécution — Vue d'ensemble de l'interface.

Exemple de code de machine d'état

Dans cet exemple de projet, la machine à états s'intègre CodeBuild à Amazon SNS.

Parcourez cet exemple de machine à états pour voir comment Step Functions utilise une machine à états pour créer un CodeBuild projet, puis envoie une rubrique Amazon SNS avec un message indiquant si la tâche a réussi ou échoué.

Pour plus d'informations sur la manière dont Step Functions peut contrôler d'autres AWS services, consultezUtilisation AWS Step Functions avec d'autres services.

```
{
  "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
    }
  }
}
```

Pour plus d'informations sur la configuration de l'IAM lors de l'utilisation de Step Functions avec d'autres AWS services, consultezPolitiques IAM pour les services intégrés.

Prétraitez les données et entraînez un modèle d'apprentissage automatique

Cet exemple de projet montre comment utiliser SageMaker et AWS Step Functions prétraiter des données et comment entraîner un modèle d'apprentissage automatique.

Dans ce projet, Step Functions utilise une fonction Lambda pour créer un bucket Amazon S3 avec un ensemble de données de test et un script Python pour le traitement des données. Il entraîne ensuite un modèle d'apprentissage automatique et effectue une transformation par lots en utilisant l'intégration SageMaker de services.

Pour plus d'informations sur les intégrations de services Step Functions SageMaker et sur celles-ci, consultez les rubriques suivantes :

- Utilisation AWS Step Functions avec d'autres services
- Gérez SageMaker avec Step Functions

> **Note**
>
> Cet exemple de projet peut entraîner des frais.
> Pour AWS les nouveaux utilisateurs, un niveau d'utilisation gratuit est disponible. Dans cette offre, les services sont gratuits en-dessous d'un certain niveau d'utilisation. Pour plus d'informations sur AWS les coûts et le niveau gratuit, consultez la section SageMaker Tarification.

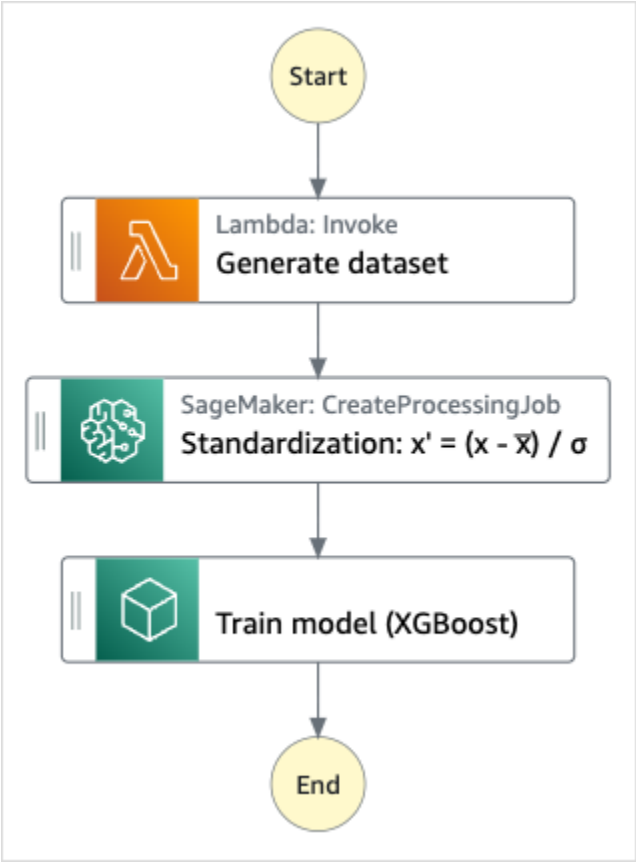
Étape 1 : créer la machine à états et provisionner les ressources

1. Ouvrez la console Step Functions et choisissez Create state machine.
2. Tapez **`Preprocess data and train a machine learning model`** dans le champ de recherche, puis choisissez Prétraiter les données et entraîner un modèle d'apprentissage automatique à partir des résultats de recherche renvoyés.
3. Choisissez Next (Suivant) pour continuer.
4. Step Functions répertorie les Services AWS éléments utilisés dans l'exemple de projet que vous avez sélectionné. Il montre également un graphique de flux de travail pour l'exemple de projet. Déployez ce projet sur votre site Compte AWS ou utilisez-le comme point de départ pour créer vos propres projets. Selon la façon dont vous souhaitez procéder, choisissez Exécuter une démo ou Construire à partir de celle-ci.

   Cet exemple de projet déploie les ressources suivantes :

   - Une AWS Lambda fonction
   - Un compartiment Amazon S3
   - Une machine AWS Step Functions étatique
   - Rôles associés AWS Identity and Access Management (IAM)

   L'image suivante montre le graphique du flux de travail pour les données de prétraitement et d'entraînement d'un exemple de modèle d'apprentissage automatique :



5. Choisissez Utiliser le modèle pour poursuivre votre sélection.
6. Effectuez l'une des actions suivantes :

   - Si vous avez sélectionné Build on it, Step Functions crée le prototype de flux de travail pour l'exemple de projet que vous avez sélectionné. Step Functions ne déploie pas les ressources répertoriées dans la définition du flux de travail.

     Dans Workflow StudioMode de conception, glissez-déposez les états depuis le Navigateur d'états pour continuer à créer votre prototype de flux de travail. Vous pouvez également passer à un éditeur de code intégré similaire à VS Code pour mettre à jour la définition Amazon States Language (ASL) de votre machine à états dans la console Step Functions. Mode code Pour plus d'informations sur l'utilisation de Workflow Studio pour créer vos machines d'état, consultezUtilisation de Workflow Studio.

     > **Important**
     >
     > N'oubliez pas de mettre à jour l'espace réservé Amazon Resource Name (ARN) pour les ressources utilisées dans l'exemple de projet avant d'exécuter votre flux de travail.

- Si vous avez sélectionné Run a demo, Step Functions crée un exemple de projet en lecture seule qui utilise un AWS CloudFormation modèle pour déployer les AWS ressources répertoriées dans ce modèle sur votre. Compte AWS

  > **Tip**
  >
  > Pour afficher la définition de la machine à états de l'exemple de projet, choisissez Code.

  Lorsque vous êtes prêt, choisissez Déployer et exécuter pour déployer l'exemple de projet et créer les ressources.

  La création de ces ressources et des autorisations IAM associées peut prendre jusqu'à 10 minutes. Pendant le déploiement de vos ressources, vous pouvez ouvrir le lien CloudFormation Stack ID pour voir quelles ressources sont mises en service.

  Une fois que toutes les ressources de l'exemple de projet ont été créées, vous pouvez voir le nouvel exemple de projet répertorié sur la page State machines.

  > **Important**
  >
  > Des frais standard peuvent s'appliquer pour chaque service utilisé dans le CloudFormation modèle.

Étape 2 : Exécuter la machine à états

1. Sur la page State machines, choisissez votre exemple de projet.
2. Sur la page d'exemple de projet, choisissez Démarrer l'exécution.
3. Dans la boîte de dialogue Démarrer l'exécution, procédez comme suit :

   1. (Facultatif) Pour identifier votre exécution, vous pouvez lui donner un nom dans le champ Nom. Par défaut, Step Functions génère automatiquement un nom d'exécution unique.

> **Note**
>
> Step Functions vous permet de créer des noms pour les machines d'état, les exécutions et les activités, ainsi que des étiquettes contenant des caractères non ASCII. Ces noms non ASCII ne fonctionnent pas avec Amazon. CloudWatch Pour être sûr de pouvoir suivre CloudWatch les métriques, choisissez un nom qui utilise uniquement des caractères ASCII.

2. (Facultatif) Dans la zone de saisie, entrez les valeurs d'entrée au format JSON pour exécuter votre flux de travail.

   Si vous avez choisi d'exécuter une démo, vous n'avez pas besoin de fournir d'entrée d'exécution.

   > **Note**
   >
   > Si le projet de démonstration que vous avez déployé contient des données d'entrée d'exécution préremplies, utilisez ces entrées pour exécuter la machine à états.

3. Choisissez Start execution (Démarrer l'exécution).
4. La console Step Functions vous dirige vers une page intitulée avec votre ID d'exécution. Cette page est connue sous le nom de page Détails de l'exécution. Sur cette page, vous pouvez consulter les résultats de l'exécution au fur et à mesure que l'exécution progresse ou une fois celle-ci terminée.

   Pour consulter les résultats de l'exécution, choisissez des états individuels dans la vue graphique, puis choisissez les onglets individuels du Détails de l'étape volet pour afficher les détails de chaque état, y compris les entrées, les sorties et la définition respectivement. Pour plus de détails sur les informations d'exécution que vous pouvez consulter sur la page Détails de l'exécution, voirPage de détails d'exécution — Vue d'ensemble de l'interface.

Exemple de code de machine d'état

Dans cet exemple de projet, la machine à états s'intègre à ces ressources SageMaker et AWS Lambda leur transmet des paramètres directement, et utilise un compartiment Amazon S3 pour la source et la sortie des données d'entraînement.

Parcourez cet exemple de machine à états pour découvrir comment Step Functions contrôle Lambda et. SageMaker

Pour plus d'informations sur la manière de AWS Step Functions contrôler d'autres AWS services, consultezUtilisation AWS Step Functions avec d'autres services.

```
{
  "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
    }
  }
}
```

Pour plus d'informations sur la configuration de l'IAM lors de l'utilisation de Step Functions avec d'autres AWS services, consultezPolitiques IAM pour les services intégrés.

Exemple IAM

Ces exemples de politiques AWS Identity and Access Management (IAM) générés par l'exemple de projet incluent le moindre privilège nécessaire pour exécuter la machine à états et les ressources associées. Nous vous recommandons de n'inclure que les autorisations nécessaires dans vos politiques IAM.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "cloudwatch:PutMetricData",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:CreateLogGroup",
                "logs:DescribeLogStreams",
                "s3:GetObject",
                "s3:PutObject",
                "s3:ListBucket",
                "ecr:GetAuthorizationToken",
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage"
            ],
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

La politique suivante permet à la fonction Lambda d'ensemencer le compartiment Amazon S3 avec des exemples de données.

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

```
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::featuretransform-
bucketforcodeanddata-1jn1le6gadwfz/*",
            "Effect": "Allow"
        }
    ]
}
```

Pour plus d'informations sur la configuration de l'IAM lors de l'utilisation de Step Functions avec d'autres AWS services, consultezPolitiques IAM pour les services intégrés.

Exemple d'orchestration Lambda

Cet exemple de projet montre comment intégrer des AWS Lambda fonctions dans les machines d'état Step Functions.

Dans ce projet, Step Functions utilise les fonctions Lambda pour vérifier le cours d'une action et déterminer une recommandation d'achat ou de vente. L'utilisateur reçoit ensuite cette recommandation et peut choisir d'acheter ou de vendre l'action. Le résultat de la transaction est renvoyé via une rubrique SNS.

Pour plus d'informations sur les intégrations de services Step Functions, consultez les rubriques suivantes :

- Utilisation AWS Step Functions avec d'autres services
- Politiques IAM pour :
  - Politiques IAM pour AWS Lambda
  - Politiques IAM pour Amazon SQS
  - Politiques IAM pour Amazon SNS

> **Note**
>
> Cet exemple de projet peut entraîner des frais.
> Pour AWS les nouveaux utilisateurs, un niveau d'utilisation gratuit est disponible. Dans cette offre, les services sont gratuits en-dessous d'un certain niveau d'utilisation. Pour plus d'informations sur AWS les coûts et le niveau gratuit, consultez la section Tarification.

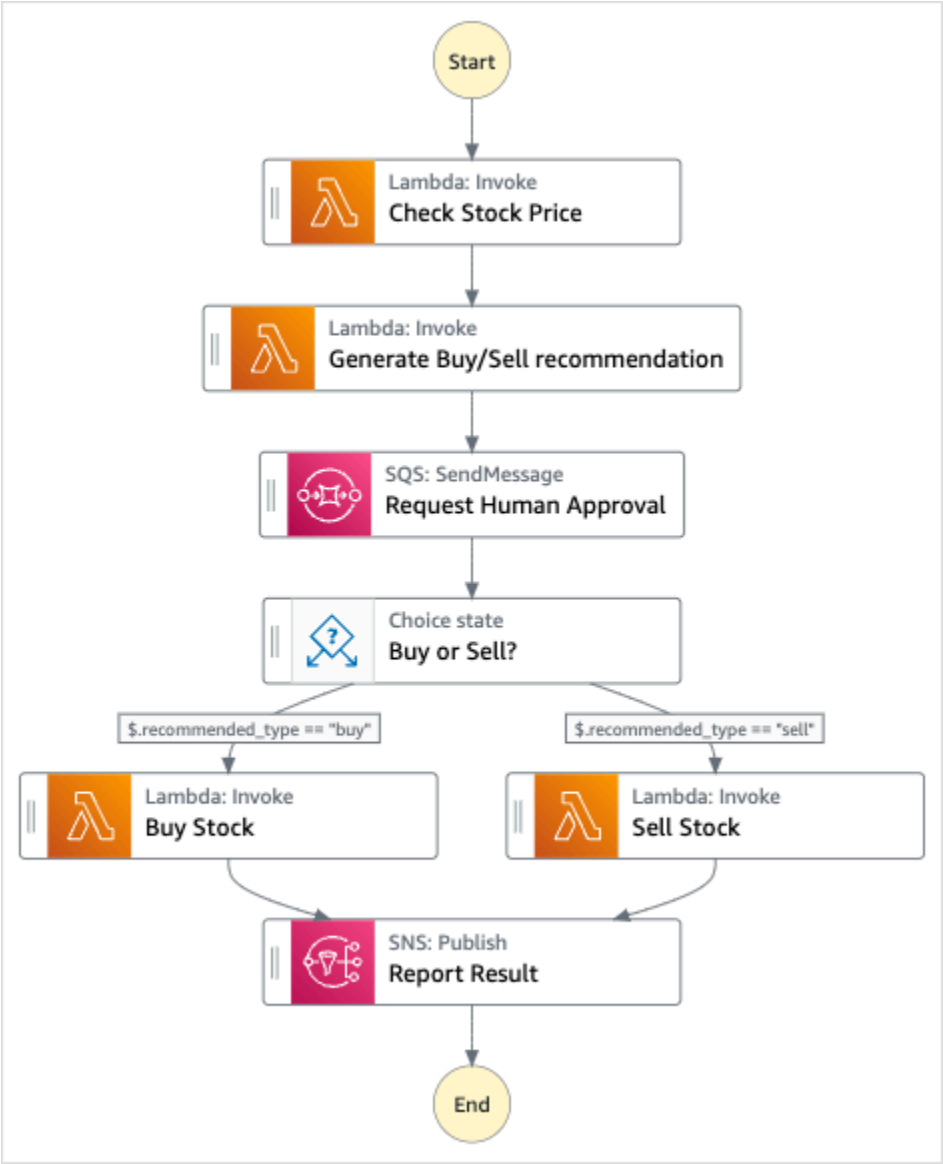
Étape 1 : créer la machine à états et provisionner les ressources

1. Ouvrez la console Step Functions et choisissez Create state machine.
2. Tapez **`Orchestrate Lambda functions`** dans la zone de recherche, puis choisissez Orchestrate Lambda functions dans les résultats de recherche renvoyés.
3. Choisissez Next (Suivant) pour continuer.
4. Step Functions répertorie les Services AWS éléments utilisés dans l'exemple de projet que vous avez sélectionné. Il montre également un graphique de flux de travail pour l'exemple de projet. Déployez ce projet sur votre site Compte AWS ou utilisez-le comme point de départ pour créer vos propres projets. Selon la façon dont vous souhaitez procéder, choisissez Exécuter une démo ou Construire à partir de celle-ci.

   Cet exemple de projet déploie les ressources suivantes :

   - Cinq Lambda fonctions
   - Une file d'attente Amazon Simple Queue Service
   - Une rubrique Amazon Simple Notification Service
   - Machine d'état AWS Step Functions
   - Rôles AWS Identity and Access Management (IAM) connexes

   L'image suivante montre le graphique du flux de travail pour l'exemple de projet Orchestrate Lambda Functions :



5. Choisissez Utiliser le modèle pour poursuivre votre sélection.
6. Effectuez l'une des actions suivantes :

   - Si vous avez sélectionné Build on it, Step Functions crée le prototype de flux de travail pour l'exemple de projet que vous avez sélectionné. Step Functions ne déploie pas les ressources répertoriées dans la définition du flux de travail.

     Dans Workflow StudioMode de conception, glissez-déposez les états depuis le Navigateur d'états pour continuer à créer votre prototype de flux de travail. Vous pouvez également passer à un éditeur de code intégré similaire à VS Code pour mettre à jour la définition Amazon States Language (ASL) de votre machine à états dans la console Step Functions.

Mode code Pour plus d'informations sur l'utilisation de Workflow Studio pour créer vos machines d'état, consultezUtilisation de Workflow Studio.

> **Important**
>
> N'oubliez pas de mettre à jour l'espace réservé Amazon Resource Name (ARN) pour les ressources utilisées dans l'exemple de projet avant d'exécuter votre flux de travail.

- Si vous avez sélectionné Run a demo, Step Functions crée un exemple de projet en lecture seule qui utilise un AWS CloudFormation modèle pour déployer les AWS ressources répertoriées dans ce modèle sur votre. Compte AWS

  > **Tip**
  >
  > Pour afficher la définition de la machine à états de l'exemple de projet, choisissez Code.

  Lorsque vous êtes prêt, choisissez Déployer et exécuter pour déployer l'exemple de projet et créer les ressources.

  La création de ces ressources et des autorisations IAM associées peut prendre jusqu'à 10 minutes. Pendant le déploiement de vos ressources, vous pouvez ouvrir le lien CloudFormation Stack ID pour voir quelles ressources sont mises en service.

  Une fois que toutes les ressources de l'exemple de projet ont été créées, vous pouvez voir le nouvel exemple de projet répertorié sur la page State machines.

  > **Important**
  >
  > Des frais standard peuvent s'appliquer pour chaque service utilisé dans le CloudFormation modèle.

Étape 2 : Exécuter la machine à états

Une fois que toutes les ressources ont été mises en service et déployées, la boîte de dialogue Démarrer l'exécution s'affiche.

1. Sur la page State machines, choisissez votre exemple de projet.
2. Sur la page d'exemple de projet, choisissez Démarrer l'exécution.
3. Dans la boîte de dialogue Démarrer l'exécution, procédez comme suit :

    1. (Facultatif) Pour identifier votre exécution, vous pouvez lui donner un nom dans le champ Nom. Par défaut, Step Functions génère automatiquement un nom d'exécution unique.

        > **Note**
        >
        > Step Functions vous permet de créer des noms pour les machines d'état, les exécutions et les activités, ainsi que des étiquettes contenant des caractères non ASCII. Ces noms non ASCII ne fonctionnent pas avec Amazon. CloudWatch Pour être sûr de pouvoir suivre CloudWatch les métriques, choisissez un nom qui utilise uniquement des caractères ASCII.

    2. (Facultatif) Dans la zone de saisie, entrez les valeurs d'entrée au format JSON pour exécuter votre flux de travail.

        Si vous avez choisi d'exécuter une démo, vous n'avez pas besoin de fournir d'entrée d'exécution.

        > **Note**
        >
        > Si le projet de démonstration que vous avez déployé contient des données d'entrée d'exécution préremplies, utilisez ces entrées pour exécuter la machine à états.

    3. Choisissez Start execution (Démarrer l'exécution).
    4. La console Step Functions vous dirige vers une page intitulée avec votre ID d'exécution. Cette page est connue sous le nom de page Détails de l'exécution. Sur cette page, vous pouvez consulter les résultats de l'exécution au fur et à mesure que l'exécution progresse ou une fois celle-ci terminée.

        Pour consulter les résultats de l'exécution, choisissez des états individuels dans la vue graphique, puis choisissez les onglets individuels du Détails de l'étape volet pour afficher les détails de chaque état, y compris les entrées, les sorties et la définition respectivement. Pour plus de détails sur les informations d'exécution que vous pouvez consulter sur la page Détails de l'exécution, voirPage de détails d'exécution — Vue d'ensemble de l'interface.

À propos de la machine à états et de son exécution

Dans cet exemple de projet, la machine à états s'intègre AWS Lambda en transmettant des paramètres directement à ces ressources, utilise une file d'attente Amazon SQS pour gérer la demande d'approbation humaine et utilise une rubrique Amazon SNS pour renvoyer les résultats de la requête.

Une exécution de Step Functions reçoit un texte JSON en entrée et transmet cette entrée au premier état du flux de travail. Les états individuels reçoivent des données JSON en entrée et transmettent généralement les données JSON en sortie à l'état suivant. Dans cet exemple de projet, le résultat de chaque étape est transmis en entrée à l'étape suivante du flux de travail. Par exemple, l'étape de recommandation Generate Buy/Sell reçoit le résultat de l'étape Vérifier le cours des actions en entrée. En outre, le résultat de l'étape de recommandation d'achat/vente est transmis en entrée à l'étape suivante, Request Human Approval, qui imite une étape d'approbation humaine.

> **Note**
>
> Pour afficher le résultat renvoyé par une étape et l'entrée transmise à une étape, ouvrez la page Détails de l'exécution pour l'exécution de votre flux de travail. Dans la section Détails de l'étape, visualisez l'entrée et la sortie de chaque étape sélectionnée en mode Affichage.

Pour implémenter une étape d'approbation humaine, vous interrompez généralement l'exécution du flux de travail jusqu'à ce qu'un jeton de tâche soit renvoyé. Dans cet exemple de projet, un message est transmis à une file d'attente Amazon SQS, qui est utilisée comme déclencheur de la fonction Lambda définie pour gérer la fonctionnalité de rappel. Le message contient un jeton de tâche et le résultat renvoyé par l'étape précédente. La fonction Lambda est invoquée avec la charge utile du message. L'exécution du flux de travail est suspendue jusqu'à ce qu'il reçoive le jeton de tâche en retour avec un appel d'`SendTaskSuccess`API. Pour plus d'informations sur les jetons de tâches, consultezAttendre un rappel avec le jeton de tâche.

Le code suivant pour la `StepFunctionsSample-HelloLambda-ApproveSqsLambda` fonction montre comment elle est définie pour approuver automatiquement toutes les tâches soumises par la file d'attente Amazon SQS via la machine d'état Step Functions.

Exemple de code de fonction Lambda pour gérer la fonctionnalité de rappel et renvoyer le jeton de tâche

```
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);
      });
  }
};
```

Parcourez cet exemple de machine à états pour découvrir comment Step Functions contrôle Lambda et Amazon SQS.

Pour plus d'informations sur la manière de AWS Step Functions contrôler d'autres AWS services, consultezUtilisation AWS Step Functions avec d'autres services.

```
{
    "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
        }
    }
}
```

Pour plus d'informations sur la configuration de l'IAM lors de l'utilisation de Step Functions avec d'autres AWS services, consultezPolitiques IAM pour les services intégrés.

Exemples IAM

Ces exemples de politiques AWS Identity and Access Management (IAM) générés par l'exemple de projet incluent le moindre privilège nécessaire pour exécuter la machine à états et les ressources associées. Nous vous recommandons de n'inclure que les autorisations nécessaires dans vos politiques IAM.

```
{
    "Statement": [
        {
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": "arn:aws:lambda:us-
west-1:111122223333:function:StepFunctionsSample-HelloLam-
CheckStockPriceLambda-444455556666",
            "Effect": "Allow"
```

```
        }
    ]
}
```

```
{
    "Statement": [
        {
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": "arn:aws:lambda:us-
west-1:111122223333:function:StepFunctionsSample-Hello-
GenerateBuySellRecommend-123456789012",
            "Effect": "Allow"
        }
    ]
}
```

```
{
    "Statement": [
        {
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": "arn:aws:lambda:us-
west-1:111122223333:function:StepFunctionsSample-HelloLambda-
BuyStockLambda-777788889999",
            "Effect": "Allow"
        }
    ]
}
```

```
{
    "Statement": [
        {
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": "arn:aws:lambda:us-
west-1:111122223333:function:StepFunctionsSample-HelloLambda-
SellStockLambda-000000000000",
            "Effect": "Allow"
```

```
        }
    ]
}
```

```
{
    "Statement": [
        {
            "Action": [
                "sqs:SendMessage*"
            ],
            "Resource": "arn:aws:sqs:us-west-1:111122223333:StepFunctionsSample-
HelloLambda4444-5555-6666-RequestHumanApprovalSqs-111111111111",
            "Effect": "Allow"
        }
    ]
}
```

```
{
    "Statement": [
        {
            "Action": [
                "sns:Publish"
            ],
            "Resource": "arn:aws:sns:us-west-1:111122223333:StepFunctionsSample-
HelloLambda1111-2222-3333-ReportResultSnsTopic-222222222222",
            "Effect": "Allow"
        }
    ]
}
```

Pour plus d'informations sur la configuration de l'IAM lors de l'utilisation de Step Functions avec d'autres AWS services, consultezPolitiques IAM pour les services intégrés.

Lancer une requête Athena

Cet exemple de projet, basé sur des flux de travail standard, montre comment utiliser Step Functions et Amazon Athena pour démarrer une requête Athena et envoyer une notification avec les résultats de la requête.

Dans ce projet, Step Functions utilise des fonctions Lambda et un AWS Glue robot d'exploration pour générer un ensemble d'exemples de données. Il exécute ensuite une requête à l'aide de l'intégration du service Athena et renvoie les résultats à l'aide d'une rubrique SNS.

Pour plus d'informations sur les intégrations des services Athena et Step Functions, consultez les rubriques suivantes :

- Utilisation AWS Step Functions avec d'autres services
- Appelez Athéna avec Step Functions

> **Note**
>
> Cet exemple de projet peut entraîner des frais.
> Pour AWS les nouveaux utilisateurs, un niveau d'utilisation gratuit est disponible. Dans cette offre, les services sont gratuits en-dessous d'un certain niveau d'utilisation. Pour plus d'informations sur AWS les coûts et le niveau gratuit, consultez la section Tarification d'Athena.

Étape 1 : créer la machine à états et provisionner les ressources

1. Ouvrez la console Step Functions et choisissez Create state machine.
2. Tapez **`Start an Athena query`** dans la zone de recherche, puis choisissez Lancer une Athena requête à partir des résultats de recherche renvoyés.
3. Choisissez Next (Suivant) pour continuer.
4. Step Functions répertorie les Services AWS éléments utilisés dans l'exemple de projet que vous avez sélectionné. Il montre également un graphique de flux de travail pour l'exemple de projet. Déployez ce projet sur votre site Compte AWS ou utilisez-le comme point de départ pour créer vos propres projets. Selon la façon dont vous souhaitez procéder, choisissez Exécuter une démo ou Construire à partir de celle-ci.

   Cet exemple de projet déploie les ressources suivantes :

   - Une Amazon Athena requête
   - Un AWS Glue crawler
   - Une rubrique Amazon SNS

- Machine d'état AWS Step Functions
- Rôles AWS Identity and Access Management (IAM) connexes

L'image suivante montre le graphique du flux de travail de l'exemple de projet Démarrer une Athena requête :

5. Choisissez Utiliser le modèle pour poursuivre votre sélection.
6. Effectuez l'une des actions suivantes :

- Si vous avez sélectionné Build on it, Step Functions crée le prototype de flux de travail pour l'exemple de projet que vous avez sélectionné. Step Functions ne déploie pas les ressources répertoriées dans la définition du flux de travail.

  Dans Workflow StudioMode de conception, glissez-déposez les états depuis le Navigateur d'états pour continuer à créer votre prototype de flux de travail. Vous pouvez également passer à un éditeur de code intégré similaire à VS Code pour mettre à jour la définition

Amazon States Language (ASL) de votre machine à états dans la console Step Functions. Mode code Pour plus d'informations sur l'utilisation de Workflow Studio pour créer vos machines d'état, consultezUtilisation de Workflow Studio.

> **Important**
>
> N'oubliez pas de mettre à jour l'espace réservé Amazon Resource Name (ARN) pour les ressources utilisées dans l'exemple de projet avant d'exécuter votre flux de travail.

- Si vous avez sélectionné Run a demo, Step Functions crée un exemple de projet en lecture seule qui utilise un AWS CloudFormation modèle pour déployer les AWS ressources répertoriées dans ce modèle sur votre. Compte AWS

  > **Tip**
  >
  > Pour afficher la définition de la machine à états de l'exemple de projet, choisissez Code.

  Lorsque vous êtes prêt, choisissez Déployer et exécuter pour déployer l'exemple de projet et créer les ressources.

  La création de ces ressources et des autorisations IAM associées peut prendre jusqu'à 10 minutes. Pendant le déploiement de vos ressources, vous pouvez ouvrir le lien CloudFormation Stack ID pour voir quelles ressources sont mises en service.

  Une fois que toutes les ressources de l'exemple de projet ont été créées, vous pouvez voir le nouvel exemple de projet répertorié sur la page State machines.

  > **Important**
  >
  > Des frais standard peuvent s'appliquer pour chaque service utilisé dans le CloudFormation modèle.

Étape 2 : Exécuter la machine à états

1. Sur la page State machines, choisissez votre exemple de projet.

2. Sur la page d'exemple de projet, choisissez Démarrer l'exécution.
3. Dans la boîte de dialogue Démarrer l'exécution, procédez comme suit :

    1. (Facultatif) Pour identifier votre exécution, vous pouvez lui donner un nom dans le champ Nom. Par défaut, Step Functions génère automatiquement un nom d'exécution unique.

        > **Note**
        >
        > Step Functions vous permet de créer des noms pour les machines d'état, les exécutions et les activités, ainsi que des étiquettes contenant des caractères non ASCII. Ces noms non ASCII ne fonctionnent pas avec Amazon. CloudWatch Pour être sûr de pouvoir suivre CloudWatch les métriques, choisissez un nom qui utilise uniquement des caractères ASCII.

    2. (Facultatif) Dans la zone de saisie, entrez les valeurs d'entrée au format JSON pour exécuter votre flux de travail.

        Si vous avez choisi d'exécuter une démo, vous n'avez pas besoin de fournir d'entrée d'exécution.

        > **Note**
        >
        > Si le projet de démonstration que vous avez déployé contient des données d'entrée d'exécution préremplies, utilisez ces entrées pour exécuter la machine à états.

    3. Choisissez Start execution (Démarrer l'exécution).
    4. La console Step Functions vous dirige vers une page intitulée avec votre ID d'exécution. Cette page est connue sous le nom de page Détails de l'exécution. Sur cette page, vous pouvez consulter les résultats de l'exécution au fur et à mesure que l'exécution progresse ou une fois celle-ci terminée.

        Pour consulter les résultats de l'exécution, choisissez des états individuels dans la vue graphique, puis choisissez les onglets individuels du Détails de l'étape volet pour afficher les détails de chaque état, y compris les entrées, les sorties et la définition respectivement. Pour plus de détails sur les informations d'exécution que vous pouvez consulter sur la page Détails de l'exécution, voirPage de détails d'exécution — Vue d'ensemble de l'interface.

Exemple de code de machine d'état

Dans cet exemple de projet, la machine à états s'intègre à Athena AWS Lambda en transmettant des paramètres directement à ces ressources, et utilise une rubrique SNS pour renvoyer les résultats de la requête.

Parcourez cet exemple de machine à états pour découvrir comment Step Functions contrôle Lambda et Athena.

Pour plus d'informations sur la manière de AWS Step Functions contrôler d'autres AWS services, consultezUtilisation AWS Step Functions avec d'autres services.

```
{
  "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
    }
  }
}
```

Pour plus d'informations sur la configuration de l'IAM lors de l'utilisation de Step Functions avec d'autres AWS services, consultezPolitiques IAM pour les services intégrés.

Exemple IAM

Ces exemples de politiques AWS Identity and Access Management (IAM) générés par l'exemple de projet incluent le moindre privilège nécessaire pour exécuter la machine à états et les ressources associées. Nous vous recommandons de n'inclure que les autorisations nécessaires dans vos politiques IAM.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": [
                "arn:aws:lambda:us-east-1:111122223333:function:StepFunctionsSample-
Athena-LambdaForDataGeneration-AKIAIOSFODNN7EXAMPLE",
                "arn:aws:lambda:us-east-1:111122223333:function:StepFunctionsSample-
Athen-LambdaForInvokingCrawler-AKIAI44QH8DHBEXAMPLE"
            ],
            "Effect": "Allow"
        },
        {
```

```
            "Action": [
                "sns:Publish"
            ],
            "Resource": [
                "arn:aws:sns:us-east-1:111122223333:StepFunctionsSample-
AthenaDataQueryd1111-2222-3333-777788889999-SNSTopic-ANPAJ2UCCR6DPCEXAMPLE"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "athena:getQueryResults",
                "athena:startQueryExecution",
                "athena:stopQueryExecution",
                "athena:getQueryExecution",
                "athena:getDataCatalog"
            ],
            "Resource": [
                "arn:aws:athena:us-east-1:111122223333:workgroup/stepfunctions-athena-
sample-project-workgroup-wJalrXUtnFEMI",
                "arn:aws:athena:us-east-1:111122223333:datacatalog/*"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:ListMultipartUploadParts",
                "s3:AbortMultipartUpload",
                "s3:CreateBucket",
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "glue:CreateDatabase",
                "glue:GetDatabase",
                "glue:GetDatabases",
                "glue:UpdateDatabase",
```

```
                "glue:DeleteDatabase",
                "glue:CreateTable",
                "glue:UpdateTable",
                "glue:GetTable",
                "glue:GetTables",
                "glue:DeleteTable",
                "glue:BatchDeleteTable",
                "glue:BatchCreatePartition",
                "glue:CreatePartition",
                "glue:UpdatePartition",
                "glue:GetPartition",
                "glue:GetPartitions",
                "glue:BatchGetPartition",
                "glue:DeletePartition",
                "glue:BatchDeletePartition"
            ],
            "Resource": [
                "arn:aws:glue:us-east-1:111122223333:database/*",
                "arn:aws:glue:us-east-1:111122223333:table/*",
                "arn:aws:glue:us-east-1:111122223333:catalog"
            ],
            "Effect": "Allow"
        }
    ]
}
```

Pour plus d'informations sur la configuration de l'IAM lors de l'utilisation de Step Functions avec d'autres AWS services, consultezPolitiques IAM pour les services intégrés.

Exécuter plusieurs requêtes (Amazon Athena, Amazon SNS)

Cet exemple de projet montre comment exécuter des requêtes Athena successivement puis en parallèle, gérer les erreurs, puis envoyer une notification Amazon SNS en fonction du succès ou de l'échec des requêtes.

Dans ce projet, Step Functions utilise une machine à états pour exécuter les requêtes Athena de manière synchrone. Une fois les résultats de la requête renvoyés, entrez dans l'état parallèle avec deux requêtes Athena exécutées en parallèle. Il attend ensuite que la tâche réussisse ou échoue, et il envoie un sujet Amazon SNS avec un message indiquant si la tâche a réussi ou échoué.

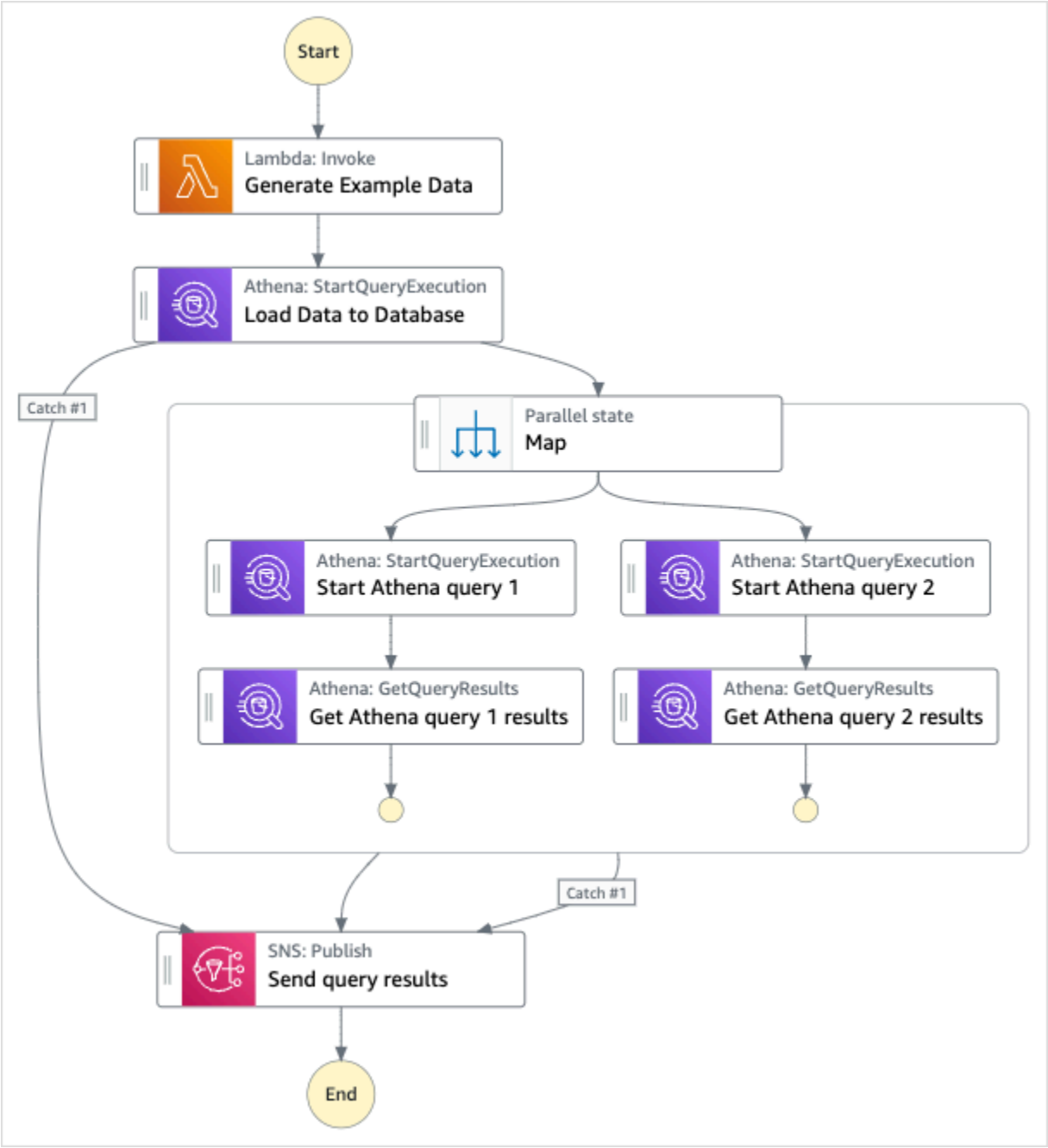
Étape 1 : créer la machine à états et provisionner les ressources

1. Ouvrez la console Step Functions et choisissez Create state machine.
2. Tapez **`Execute multiple queries`** dans la zone de recherche, puis choisissez Exécuter plusieurs requêtes à partir des résultats de recherche renvoyés.
3. Choisissez Next (Suivant) pour continuer.
4. Step Functions répertorie les Services AWS éléments utilisés dans l'exemple de projet que vous avez sélectionné. Il montre également un graphique de flux de travail pour l'exemple de projet. Déployez ce projet sur votre site Compte AWS ou utilisez-le comme point de départ pour créer vos propres projets. Selon la façon dont vous souhaitez procéder, choisissez Exécuter une démo ou Construire à partir de celle-ci.

   Cet exemple de projet déploie les ressources suivantes :

   - Amazon Athenarequêtes
   - Une rubrique Amazon SNS
   - Machine d'état AWS Step Functions
   - Rôles AWS Identity and Access Management (IAM) connexes

   L'image suivante montre le graphique du flux de travail pour l'exemple de projet Execute multiple queries :



5. Choisissez Utiliser le modèle pour poursuivre votre sélection.
6. Effectuez l'une des actions suivantes :

   - Si vous avez sélectionné Build on it, Step Functions crée le prototype de flux de travail pour l'exemple de projet que vous avez sélectionné. Step Functions ne déploie pas les ressources répertoriées dans la définition du flux de travail.

     Dans Workflow StudioMode de conception, glissez-déposez les états depuis le Navigateur d'états pour continuer à créer votre prototype de flux de travail. Vous pouvez également

passer à un éditeur de code intégré similaire à VS Code pour mettre à jour la définition Amazon States Language (ASL) de votre machine à états dans la console Step Functions. Mode code Pour plus d'informations sur l'utilisation de Workflow Studio pour créer vos machines d'état, consultezUtilisation de Workflow Studio.

> **Important**
>
> N'oubliez pas de mettre à jour l'espace réservé Amazon Resource Name (ARN) pour les ressources utilisées dans l'exemple de projet avant d'exécuter votre flux de travail.

- Si vous avez sélectionné Run a demo, Step Functions crée un exemple de projet en lecture seule qui utilise un AWS CloudFormation modèle pour déployer les AWS ressources répertoriées dans ce modèle sur votre. Compte AWS

  > **Tip**
  >
  > Pour afficher la définition de la machine à états de l'exemple de projet, choisissez Code.

  Lorsque vous êtes prêt, choisissez Déployer et exécuter pour déployer l'exemple de projet et créer les ressources.

  La création de ces ressources et des autorisations IAM associées peut prendre jusqu'à 10 minutes. Pendant le déploiement de vos ressources, vous pouvez ouvrir le lien CloudFormation Stack ID pour voir quelles ressources sont mises en service.

  Une fois que toutes les ressources de l'exemple de projet ont été créées, vous pouvez voir le nouvel exemple de projet répertorié sur la page State machines.

  > **Important**
  >
  > Des frais standard peuvent s'appliquer pour chaque service utilisé dans le CloudFormation modèle.

Étape 2 : Exécuter la machine d'état

1. Sur la page State machines, choisissez votre exemple de projet.
2. Sur la page d'exemple de projet, choisissez Démarrer l'exécution.
3. Dans la boîte de dialogue Démarrer l'exécution, procédez comme suit :

    1. (Facultatif) Pour identifier votre exécution, vous pouvez lui donner un nom dans le champ Nom. Par défaut, Step Functions génère automatiquement un nom d'exécution unique.

        > **Note**
        >
        > Step Functions vous permet de créer des noms pour les machines d'état, les exécutions, les activités et les étiquettes contenant des caractères non ASCII. Ces noms non ASCII ne fonctionnent pas avec Amazon. CloudWatch Pour être sûr de pouvoir suivre CloudWatch les métriques, choisissez un nom qui utilise uniquement des caractères ASCII.

    2. (Facultatif) Dans la zone de saisie, entrez les valeurs d'entrée au format JSON pour exécuter votre flux de travail.

        Si vous avez choisi d'exécuter une démo, vous n'avez pas besoin de fournir d'entrée d'exécution.

    3. Choisissez Start execution (Démarrer l'exécution).
    4. La console Step Functions vous dirige vers une page intitulée avec votre ID d'exécution. Cette page est connue sous le nom de page Détails de l'exécution. Sur cette page, vous pouvez consulter les résultats de l'exécution au fur et à mesure que l'exécution progresse ou une fois celle-ci terminée.

        Pour consulter les résultats de l'exécution, choisissez des états individuels dans la vue graphique, puis choisissez les onglets individuels du Détails de l'étape volet pour afficher les détails de chaque état, y compris les entrées, les sorties et la définition respectivement. Pour plus de détails sur les informations d'exécution que vous pouvez consulter sur la page Détails de l'exécution, voirPage de détails d'exécution — Vue d'ensemble de l'interface.

Exemple de code de machine d'état

Dans cet exemple de projet, la machine à états s'intègre à Amazon Athena et Amazon SNS en transmettant des paramètres directement à ces ressources.

Parcourez cet exemple de machine à états pour découvrir comment Step Functions contrôle Amazon Athena et Amazon SNS en se connectant à l'Amazon Resource Name (ARN) sur `Resource` le terrain et en accédant à l'API du `Parameters` service.

Pour plus d'informations sur la manière de AWS Step Functions contrôler d'autres AWS services, consultezUtilisation AWS Step Functions avec d'autres services.

```
{
    "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
      }
    }
  }
```

Exemples IAM

Cet exemple de politique AWS Identity and Access Management (IAM) généré par l'exemple de projet inclut le minimum de privilèges nécessaires pour exécuter la machine à états et les ressources associées. Nous vous recommandons de n'inclure que les autorisations nécessaires dans vos politiques IAM.

AthenaStartQueryExecution

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

```
        {
            "Effect": "Allow",
            "Action": [
                "athena:startQueryExecution",
                "athena:stopQueryExecution",
                "athena:getQueryExecution",
                "athena:getDataCatalog"
            ],
            "Resource": [
                "arn:aws:athena:us-east-2:123456789012:workgroup/stepfunctions-athena-
sample-project-workgroup-ztuvu9yuix",
                "arn:aws:athena:us-east-2:123456789012:datacatalog/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:ListMultipartUploadParts",
                "s3:AbortMultipartUpload",
                "s3:CreateBucket",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "glue:CreateDatabase",
                "glue:GetDatabase",
                "glue:GetDatabases",
                "glue:UpdateDatabase",
                "glue:DeleteDatabase",
                "glue:CreateTable",
                "glue:UpdateTable",
                "glue:GetTable",
                "glue:GetTables",
                "glue:DeleteTable",
                "glue:BatchDeleteTable",
```

```
                "glue:BatchCreatePartition",
                "glue:CreatePartition",
                "glue:UpdatePartition",
                "glue:GetPartition",
                "glue:GetPartitions",
                "glue:BatchGetPartition",
                "glue:DeletePartition",
                "glue:BatchDeletePartition"
            ],
            "Resource": [
                "arn:aws:glue:us-east-2:123456789012:catalog",
                "arn:aws:glue:us-east-2:123456789012:database/*",
                "arn:aws:glue:us-east-2:123456789012:table/*",
                "arn:aws:glue:us-east-2:123456789012:userDefinedFunction/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "lakeformation:GetDataAccess"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

AthenaGetQueryResults

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "athena:getQueryResults"
            ],
            "Resource": [
                "arn:aws:us-east-2:123456789012:workgroup/*"
            ]
        },
```

```
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::*"
            ]
        }
    ]
}
```

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"
            ]
        }
    ]
}
```

Pour plus d'informations sur la configuration de l'IAM lors de l'utilisation de Step Functions avec d'autres AWS services, consultezPolitiques IAM pour les services intégrés.

Interrogez de grands ensembles de données (Amazon Athena, Amazon S3 AWS Glue, Amazon SNS)

Cet exemple de projet montre comment ingérer un ensemble de données volumineux dans Amazon S3 et le partitionner via AWS Glue Crawlers, puis exécuter des requêtes Amazon Athena sur cette partition.

Dans ce projet, la machine d'état Step Functions invoque un AWS Glue robot d'exploration qui partitionne un ensemble de données volumineux dans Amazon S3. Une fois que le AWS Glue robot d'exploration renvoie un message de réussite, le flux de travail exécute les requêtes Athena sur cette partition. Une fois que l'exécution de la requête est terminée avec succès, une notification Amazon SNS est envoyée à une rubrique Amazon SNS.

Étape 1 : créer la machine à états et provisionner les ressources

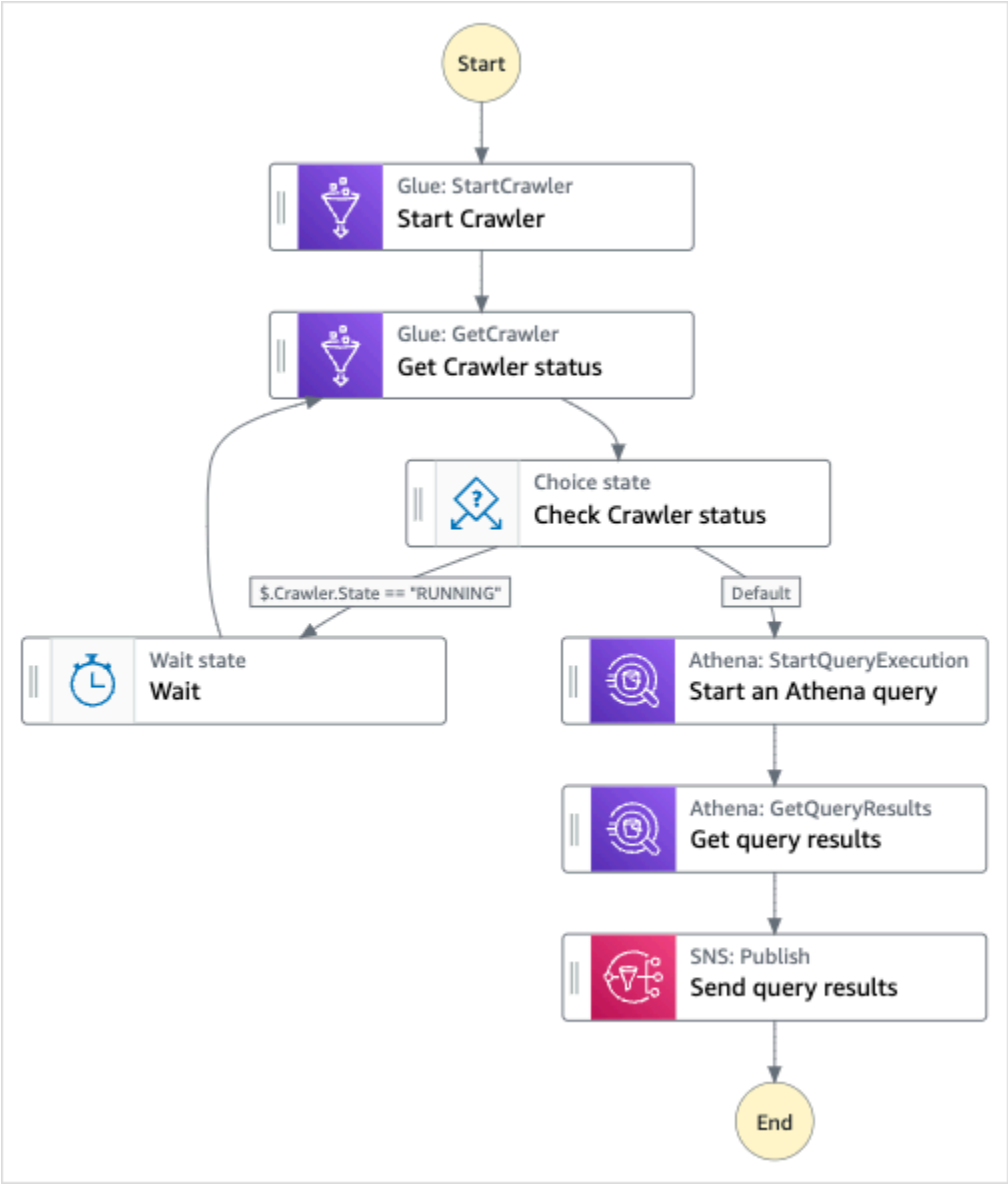
1. Ouvrez la console Step Functions et choisissez Create state machine.
2. Tapez **`Query large datasets`** dans la zone de recherche, puis choisissez Interroger de grands ensembles de données dans les résultats de recherche renvoyés.
3. Choisissez Next (Suivant) pour continuer.

4. Step Functions répertorie les Services AWS éléments utilisés dans l'exemple de projet que vous avez sélectionné. Il montre également un graphique de flux de travail pour l'exemple de projet. Déployez ce projet sur votre site Compte AWS ou utilisez-le comme point de départ pour créer vos propres projets. Selon la façon dont vous souhaitez procéder, choisissez Exécuter une démo ou Construire à partir de celle-ci.

   Cet exemple de projet déploie les ressources suivantes :

   - Un compartiment Amazon S3
   - Un AWS Glue crawler
   - Une rubrique Amazon SNS
   - Machine d'état AWS Step Functions
   - Rôles AWS Identity and Access Management (IAM) connexes

   L'image suivante montre le graphique du flux de travail pour l'exemple de projet Query large datasets :



5. Choisissez Utiliser le modèle pour poursuivre votre sélection.
6. Effectuez l'une des actions suivantes :

   - Si vous avez sélectionné Build on it, Step Functions crée le prototype de flux de travail pour l'exemple de projet que vous avez sélectionné. Step Functions ne déploie pas les ressources répertoriées dans la définition du flux de travail.

     Dans Workflow StudioMode de conception, glissez-déposez les états depuis le Navigateur d'états pour continuer à créer votre prototype de flux de travail. Vous pouvez également passer à un éditeur de code intégré similaire à VS Code pour mettre à jour la définition Amazon States Language (ASL) de votre machine à états dans la console Step Functions.

Mode code Pour plus d'informations sur l'utilisation de Workflow Studio pour créer vos machines d'état, consultezUtilisation de Workflow Studio.

> **Important**
>
> N'oubliez pas de mettre à jour l'espace réservé Amazon Resource Name (ARN) pour les ressources utilisées dans l'exemple de projet avant d'exécuter votre flux de travail.

- Si vous avez sélectionné Run a demo, Step Functions crée un exemple de projet en lecture seule qui utilise un AWS CloudFormation modèle pour déployer les AWS ressources répertoriées dans ce modèle sur votre. Compte AWS

> **Tip**
>
> Pour afficher la définition de la machine à états de l'exemple de projet, choisissez Code.

Lorsque vous êtes prêt, choisissez Déployer et exécuter pour déployer l'exemple de projet et créer les ressources.

La création de ces ressources et des autorisations IAM associées peut prendre jusqu'à 10 minutes. Pendant le déploiement de vos ressources, vous pouvez ouvrir le lien CloudFormation Stack ID pour voir quelles ressources sont mises en service.

Une fois que toutes les ressources de l'exemple de projet ont été créées, vous pouvez voir le nouvel exemple de projet répertorié sur la page State machines.

> **Important**
>
> Des frais standard peuvent s'appliquer pour chaque service utilisé dans le CloudFormation modèle.

Étape 2 : Exécuter la machine à états

1. Sur la page State machines, choisissez votre exemple de projet.
2. Sur la page d'exemple de projet, choisissez Démarrer l'exécution.

3. Dans la boîte de dialogue Démarrer l'exécution, procédez comme suit :

   1. (Facultatif) Pour identifier votre exécution, vous pouvez lui donner un nom dans le champ Nom. Par défaut, Step Functions génère automatiquement un nom d'exécution unique.

      > **Note**
      >
      > Step Functions vous permet de créer des noms pour les machines d'état, les exécutions, les activités et les étiquettes contenant des caractères non ASCII. Ces noms non ASCII ne fonctionnent pas avec Amazon. CloudWatch Pour être sûr de pouvoir suivre CloudWatch les métriques, choisissez un nom qui utilise uniquement des caractères ASCII.

   2. (Facultatif) Dans la zone de saisie, entrez les valeurs d'entrée au format JSON pour exécuter votre flux de travail.

      Si vous avez choisi d'exécuter une démo, vous n'avez pas besoin de fournir d'entrée d'exécution.

   3. Choisissez Start execution (Démarrer l'exécution).
   4. La console Step Functions vous dirige vers une page intitulée avec votre ID d'exécution. Cette page est connue sous le nom de page Détails de l'exécution. Sur cette page, vous pouvez consulter les résultats de l'exécution au fur et à mesure que l'exécution progresse ou une fois celle-ci terminée.

      Pour consulter les résultats de l'exécution, choisissez des états individuels dans la vue graphique, puis choisissez les onglets individuels du Détails de l'étape volet pour afficher les détails de chaque état, y compris les entrées, les sorties et la définition respectivement. Pour plus de détails sur les informations d'exécution que vous pouvez consulter sur la page Détails de l'exécution, voirPage de détails d'exécution — Vue d'ensemble de l'interface.

Exemple de code de machine d'état

Dans cet exemple de projet, la machine à états s'intègre à Amazon S3 AWS Glue, Amazon Athena et Amazon SNS en transmettant des paramètres directement à ces ressources.

Parcourez cet exemple de machine à états pour découvrir comment Step Functions contrôle Amazon S3 AWS Glue, Amazon Athena et Amazon SNS en se connectant à l'Amazon Resource Name (ARN) sur `Resource` le terrain et en accédant à l'API du `Parameters` service.

Pour plus d'informations sur la manière de AWS Step Functions contrôler d'autres AWS services, consultezUtilisation AWS Step Functions avec d'autres services.

```
{
    "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
      }
    }
  }
```

Exemples IAM

Ces exemples de politiques AWS Identity and Access Management (IAM) générés par l'exemple de projet incluent le moindre privilège nécessaire pour exécuter la machine à états et les ressources associées. Nous vous recommandons de n'inclure que les autorisations nécessaires dans vos politiques IAM.

AthenaGetQueryResults

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

```
            "Effect": "Allow",
            "Action": [
                "athena:getQueryResults"
            ],
            "Resource": [
                "arn:aws:athena:us-east-2:123456789012:workgroup/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::*"
            ]
        }
    ]
}
```

AthenaStartQueryExecution

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "athena:startQueryExecution",
                "athena:stopQueryExecution",
                "athena:getQueryExecution",
                "athena:getDataCatalog"
            ],
            "Resource": [
                "arn:aws:athena:us-east-2:123456789012:workgroup/stepfunctions-athena-
sample-project-workgroup-8v7bshiv70",
                "arn:aws:athena:us-east-2:123456789012:datacatalog/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
```

```
                "s3:GetObject",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:ListMultipartUploadParts",
                "s3:AbortMultipartUpload",
                "s3:CreateBucket",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "glue:CreateDatabase",
                "glue:GetDatabase",
                "glue:GetDatabases",
                "glue:UpdateDatabase",
                "glue:DeleteDatabase",
                "glue:CreateTable",
                "glue:UpdateTable",
                "glue:GetTable",
                "glue:GetTables",
                "glue:DeleteTable",
                "glue:BatchDeleteTable",
                "glue:BatchCreatePartition",
                "glue:CreatePartition",
                "glue:UpdatePartition",
                "glue:GetPartition",
                "glue:GetPartitions",
                "glue:BatchGetPartition",
                "glue:DeletePartition",
                "glue:BatchDeletePartition"
            ],
            "Resource": [
                "arn:aws:glue:us-east-2:123456789012:catalog",
                "arn:aws:glue:us-east-2:123456789012:database/*",
                "arn:aws:glue:us-east-2:123456789012:table/*",
                "arn:aws:glue:us-east-2:123456789012:userDefinedFunction/*"
            ]
        },
        {
            "Effect": "Allow",
```

```
            "Action": [
                "lakeformation:GetDataAccess"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

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"
            ]
        }
    ]
}
```

Pour plus d'informations sur la configuration de l'IAM lors de l'utilisation de Step Functions avec d'autres AWS services, consultezPolitiques IAM pour les services intégrés.

Maintenir les données à jour (Amazon Athena, Amazon S3,) AWS Glue

Cet exemple de projet montre comment interroger une table cible pour obtenir des données actuelles avec AWS Glue Catalog, puis comment la mettre à jour avec de nouvelles données provenant d'autres sources à l'aide d'Amazon Athena.

Dans ce projet, la machine d'état Step Functions appelle AWS Glue Catalog pour vérifier si une table cible existe dans un compartiment Amazon S3. Si aucune table n'est trouvée, une nouvelle table sera créée. Step FunctionsExécute ensuite une requête Athena pour ajouter des lignes à la table cible à partir d'une autre source de données : en interrogeant d'abord la table cible pour obtenir la date la plus récente, puis en interrogeant la table source pour obtenir des données plus récentes et en les insérant dans la table cible.

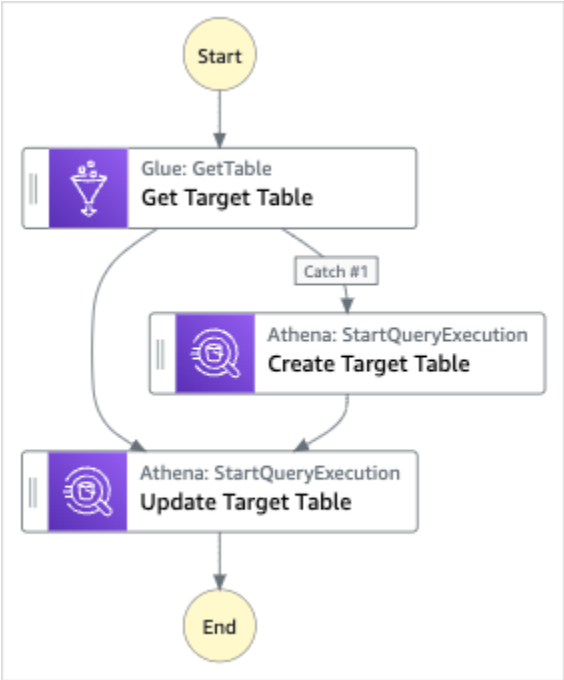
Étape 1 : créer la machine à états et provisionner les ressources

1. Ouvrez la console Step Functions et choisissez Create state machine.
2. Tapez **`Keep data up to date`** dans la zone de recherche, puis choisissez Garder les données à jour à partir des résultats de recherche renvoyés.
3. Choisissez Next (Suivant) pour continuer.
4. Step Functions répertorie les Services AWS éléments utilisés dans l'exemple de projet que vous avez sélectionné. Il montre également un graphique de flux de travail pour l'exemple de projet. Déployez ce projet sur votre site Compte AWS ou utilisez-le comme point de départ pour créer vos propres projets. Selon la façon dont vous souhaitez procéder, choisissez Exécuter une démo ou Construire à partir de celle-ci.

   Cet exemple de projet déploie les ressources suivantes :

   - Un compartiment Amazon S3
   - Amazon Athenarequêtes
   - Un AWS Glue Data Catalog appel
   - Machine d'état AWS Step Functions
   - Rôles AWS Identity and Access Management (IAM) connexes

   L'image suivante montre le graphique du flux de travail pour l'exemple de projet Maintenir les données à jour :



5. Choisissez Utiliser le modèle pour poursuivre votre sélection.
6. Effectuez l'une des actions suivantes :

   - Si vous avez sélectionné Build on it, Step Functions crée le prototype de flux de travail pour l'exemple de projet que vous avez sélectionné. Step Functions ne déploie pas les ressources répertoriées dans la définition du flux de travail.

     Dans Workflow StudioMode de conception, glissez-déposez les états depuis le Navigateur d'états pour continuer à créer votre prototype de flux de travail. Vous pouvez également passer à un éditeur de code intégré similaire à VS Code pour mettre à jour la définition Amazon States Language (ASL) de votre machine à états dans la console Step Functions. Mode code Pour plus d'informations sur l'utilisation de Workflow Studio pour créer vos machines d'état, consultezUtilisation de Workflow Studio.

     > **Important**
     >
     > N'oubliez pas de mettre à jour l'espace réservé Amazon Resource Name (ARN) pour les ressources utilisées dans l'exemple de projet avant d'exécuter votre flux de travail.

   - Si vous avez sélectionné Run a demo, Step Functions crée un exemple de projet en lecture seule qui utilise un AWS CloudFormation modèle pour déployer les AWS ressources répertoriées dans ce modèle sur votre. Compte AWS

> **Tip**
>
> Pour afficher la définition de la machine à états de l'exemple de projet, choisissez Code.

Lorsque vous êtes prêt, choisissez Déployer et exécuter pour déployer l'exemple de projet et créer les ressources.

La création de ces ressources et des autorisations IAM associées peut prendre jusqu'à 10 minutes. Pendant le déploiement de vos ressources, vous pouvez ouvrir le lien CloudFormation Stack ID pour voir quelles ressources sont mises en service.

Une fois que toutes les ressources de l'exemple de projet ont été créées, vous pouvez voir le nouvel exemple de projet répertorié sur la page State machines.

> **Important**
>
> Des frais standard peuvent s'appliquer pour chaque service utilisé dans le CloudFormation modèle.

Étape 2 : Exécuter la machine d'état

1. Sur la page State machines, choisissez votre exemple de projet.
2. Sur la page d'exemple de projet, choisissez Démarrer l'exécution.
3. Dans la boîte de dialogue Démarrer l'exécution, procédez comme suit :

    1. (Facultatif) Pour identifier votre exécution, vous pouvez lui donner un nom dans le champ Nom. Par défaut, Step Functions génère automatiquement un nom d'exécution unique.

        > **Note**
        >
        > Step Functions vous permet de créer des noms pour les machines d'état, les exécutions, les activités et les étiquettes contenant des caractères non ASCII. Ces noms non ASCII ne fonctionnent pas avec Amazon. CloudWatch Pour être sûr de

pouvoir suivre CloudWatch les métriques, choisissez un nom qui utilise uniquement des caractères ASCII.

2. (Facultatif) Dans la zone de saisie, entrez les valeurs d'entrée au format JSON pour exécuter votre flux de travail.

   Si vous avez choisi d'exécuter une démo, vous n'avez pas besoin de fournir d'entrée d'exécution.

3. Choisissez Start execution (Démarrer l'exécution).
4. La console Step Functions vous dirige vers une page intitulée avec votre ID d'exécution. Cette page est connue sous le nom de page Détails de l'exécution. Sur cette page, vous pouvez consulter les résultats de l'exécution au fur et à mesure que l'exécution progresse ou une fois celle-ci terminée.

   Pour consulter les résultats de l'exécution, choisissez des états individuels dans la vue graphique, puis choisissez les onglets individuels du Détails de l'étape volet pour afficher les détails de chaque état, y compris les entrées, les sorties et la définition respectivement. Pour plus de détails sur les informations d'exécution que vous pouvez consulter sur la page Détails de l'exécution, voirPage de détails d'exécution — Vue d'ensemble de l'interface.

Exemple de code de machine d'état

Dans cet exemple de projet, AWS Glue la machine à états s'intègre à Amazon S3 et Amazon Athena en transmettant des paramètres directement à ces ressources.

Parcourez cet exemple de machine à états pour découvrir comment Step Functions contrôle Amazon S3 et Amazon Athena en se connectant à l'Amazon Resource Name (ARN) `Resource` sur le terrain et en accédant `Parameters` à l'API du service. AWS Glue

Pour plus d'informations sur la manière de AWS Step Functions contrôler d'autres AWS services, consultezUtilisation AWS Step Functions avec d'autres services.

```
{
    "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
      }
    }
  }
```

Exemple IAM

Cet exemple de politique AWS Identity and Access Management (IAM) généré par l'exemple de projet inclut le minimum de privilèges nécessaires pour exécuter la machine à états et les ressources associées. Nous vous recommandons de n'inclure que les autorisations nécessaires dans vos politiques IAM.

AthenaStartQueryExecution

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "athena:startQueryExecution",
                "athena:stopQueryExecution",
                "athena:getQueryExecution",
                "athena:getDataCatalog"
            ],
            "Resource": [
                "arn:aws:athena:us-east-2:123456789012:workgroup/stepfunctions-athena-
sample-project-workgroup-26ujlyawxg",
                "arn:aws:athena:us-east-2:123456789012:datacatalog/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:ListMultipartUploadParts",
                "s3:AbortMultipartUpload",
                "s3:CreateBucket",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "glue:CreateDatabase",
                "glue:GetDatabase",
                "glue:GetDatabases",
                "glue:UpdateDatabase",
                "glue:DeleteDatabase",
```

```
                "glue:CreateTable",
                "glue:UpdateTable",
                "glue:GetTable",
                "glue:GetTables",
                "glue:DeleteTable",
                "glue:BatchDeleteTable",
                "glue:BatchCreatePartition",
                "glue:CreatePartition",
                "glue:UpdatePartition",
                "glue:GetPartition",
                "glue:GetPartitions",
                "glue:BatchGetPartition",
                "glue:DeletePartition",
                "glue:BatchDeletePartition"
            ],
            "Resource": [
                "arn:aws::glue:us-east-2:123456789012:catalog",
                "arn:aws::glue:us-east-2:123456789012:database/*",
                "arn:aws::glue:us-east-2:123456789012:table/*",
                "arn:aws::glue:us-east-2:123456789012:userDefinedFunction/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "lakeformation:GetDataAccess"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Pour plus d'informations sur la configuration de l'IAM lors de l'utilisation de Step Functions avec d'autres AWS services, consultezPolitiques IAM pour les services intégrés.

Gérer un cluster Amazon EKS

Cet exemple de projet montre comment utiliser Step Functions et Amazon Elastic Kubernetes Service pour créer un cluster Amazon EKS avec un groupe de nœuds, exécuter une tâche sur Amazon EKS,

puis examiner le résultat. Une fois terminé, il supprime les groupes de nœuds et le cluster Amazon EKS.

Pour plus d'informations sur les intégrations de services Step Functions et Step Functions, consultez les rubriques suivantes :

- Utilisation AWS Step Functions avec d'autres services
- Appelez Amazon EKS avec Step Functions

> **Note**
>
> Cet exemple de projet peut entraîner des frais.
> Pour AWS les nouveaux utilisateurs, un niveau d'utilisation gratuit est disponible. Dans cette offre, les services sont gratuits en-dessous d'un certain niveau d'utilisation. Pour plus d'informations sur AWS les coûts et le niveau gratuit, consultez la section Tarification d'Amazon EKS.

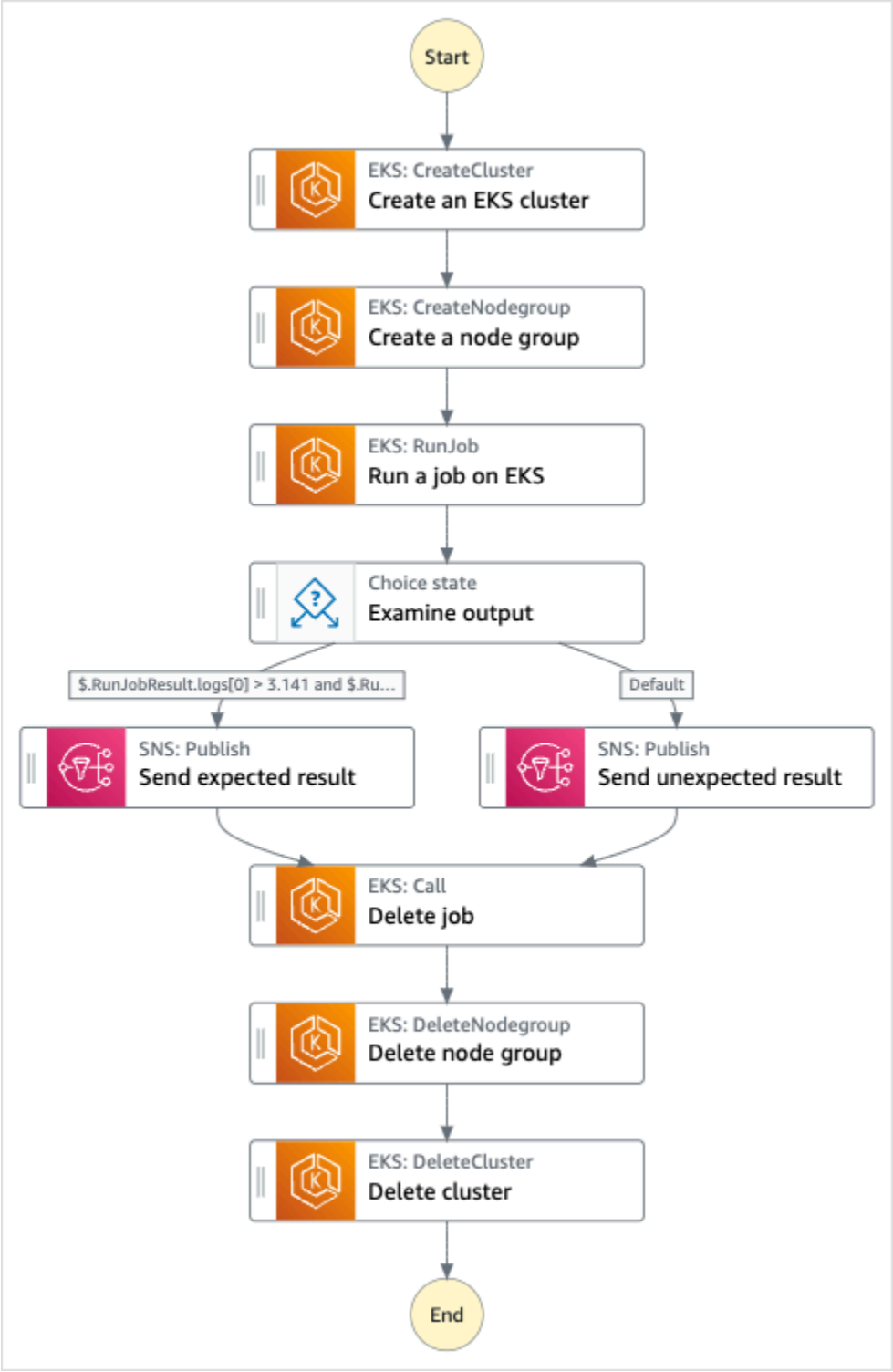
Étape 1 : créer la machine à états et provisionner les ressources

1. Ouvrez la console Step Functions et choisissez Create state machine.
2. Tapez **`Manage an EKS cluster`** dans la zone de recherche, puis choisissez Gérer un cluster EKS dans les résultats de recherche renvoyés.
3. Choisissez Next (Suivant) pour continuer.
4. Step Functions répertorie les Services AWS éléments utilisés dans l'exemple de projet que vous avez sélectionné. Il montre également un graphique de flux de travail pour l'exemple de projet. Déployez ce projet sur votre site Compte AWS ou utilisez-le comme point de départ pour créer vos propres projets. Selon la façon dont vous souhaitez procéder, choisissez Exécuter une démo ou Construire à partir de celle-ci.

   Cet exemple de projet déploie les ressources suivantes :

   - Un Amazon Elastic Kubernetes Service cluster
   - Une rubrique Amazon SNS
   - Machine d'état AWS Step Functions
   - Rôles AWS Identity and Access Management (IAM) connexes

L'image suivante montre le graphique du flux de travail pour le projet d'exemple Gérer un cluster EKS :



5. Choisissez Utiliser le modèle pour poursuivre votre sélection.

6. Effectuez l'une des actions suivantes :

   - Si vous avez sélectionné Build on it, Step Functions crée le prototype de flux de travail pour l'exemple de projet que vous avez sélectionné. Step Functions ne déploie pas les ressources répertoriées dans la définition du flux de travail.

     Dans Workflow StudioMode de conception, glissez-déposez les états depuis le Navigateur d'états pour continuer à créer votre prototype de flux de travail. Vous pouvez également passer à un éditeur de code intégré similaire à VS Code pour mettre à jour la définition Amazon States Language (ASL) de votre machine à états dans la console Step Functions. Mode code Pour plus d'informations sur l'utilisation de Workflow Studio pour créer vos machines d'état, consultezUtilisation de Workflow Studio.

     > **Important**
     >
     > N'oubliez pas de mettre à jour l'espace réservé Amazon Resource Name (ARN) pour les ressources utilisées dans l'exemple de projet avant d'exécuter votre flux de travail.

   - Si vous avez sélectionné Run a demo, Step Functions crée un exemple de projet en lecture seule qui utilise un AWS CloudFormation modèle pour déployer les AWS ressources répertoriées dans ce modèle sur votre. Compte AWS

     > **Tip**
     >
     > Pour afficher la définition de la machine à états de l'exemple de projet, choisissez Code.

     Lorsque vous êtes prêt, choisissez Déployer et exécuter pour déployer l'exemple de projet et créer les ressources.

     La création de ces ressources et des autorisations IAM associées peut prendre jusqu'à 10 minutes. Pendant le déploiement de vos ressources, vous pouvez ouvrir le lien CloudFormation Stack ID pour voir quelles ressources sont mises en service.

     Une fois que toutes les ressources de l'exemple de projet ont été créées, vous pouvez voir le nouvel exemple de projet répertorié sur la page State machines.

> **Important**
>
> Des frais standard peuvent s'appliquer pour chaque service utilisé dans le CloudFormation modèle.

Étape 2 : Exécuter la machine à états

1. Sur la page State machines, choisissez votre exemple de projet.
2. Sur la page d'exemple de projet, choisissez Démarrer l'exécution.
3. Dans la boîte de dialogue Démarrer l'exécution, procédez comme suit :

    1. (Facultatif) Pour identifier votre exécution, vous pouvez lui donner un nom dans le champ Nom. Par défaut, Step Functions génère automatiquement un nom d'exécution unique.

        > **Note**
        >
        > Step Functions vous permet de créer des noms pour les machines d'état, les exécutions et les activités, ainsi que des étiquettes contenant des caractères non ASCII. Ces noms non ASCII ne fonctionnent pas avec Amazon. CloudWatch Pour être sûr de pouvoir suivre CloudWatch les métriques, choisissez un nom qui utilise uniquement des caractères ASCII.

    2. (Facultatif) Dans la zone de saisie, entrez les valeurs d'entrée au format JSON pour exécuter votre flux de travail.

        Si vous avez choisi d'exécuter une démo, vous n'avez pas besoin de fournir d'entrée d'exécution.

        > **Note**
        >
        > Si le projet de démonstration que vous avez déployé contient des données d'entrée d'exécution préremplies, utilisez ces entrées pour exécuter la machine à états.

    3. Choisissez Start execution (Démarrer l'exécution).
    4. La console Step Functions vous dirige vers une page intitulée avec votre ID d'exécution. Cette page est connue sous le nom de page Détails de l'exécution. Sur cette page, vous pouvez

consulter les résultats de l'exécution au fur et à mesure que l'exécution progresse ou une fois celle-ci terminée.

Pour consulter les résultats de l'exécution, choisissez des états individuels dans la vue graphique, puis choisissez les onglets individuels du Détails de l'étape volet pour afficher les détails de chaque état, y compris les entrées, les sorties et la définition respectivement. Pour plus de détails sur les informations d'exécution que vous pouvez consulter sur la page Détails de l'exécution, voirPage de détails d'exécution — Vue d'ensemble de l'interface.

Exemple de code de machine d'état

Dans cet exemple de projet, la machine à états s'intègre à Amazon EKS en créant un cluster et un groupe de nœuds Amazon EKS, et utilise une rubrique SNS pour renvoyer les résultats.

Parcourez cet exemple de machine à états pour découvrir comment Step Functions gère les clusters et les groupes de nœuds Amazon EKS.

Pour plus d'informations sur la manière de AWS Step Functions contrôler d'autres AWS services, consultezUtilisation AWS Step Functions avec d'autres services.

```
{
  "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
    }
  }
}
```

Pour plus d'informations sur la configuration de l'IAM lors de l'utilisation de Step Functions avec d'autres AWS services, consultezPolitiques IAM pour les services intégrés.

Exemple IAM

Ces exemples de politiques AWS Identity and Access Management (IAM) générés par l'exemple de projet incluent le moindre privilège nécessaire pour exécuter la machine à états et les ressources associées. Nous vous recommandons de n'inclure que les autorisations nécessaires dans vos politiques IAM.

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

```
            "Effect": "Allow",
            "Action": [
                "eks:CreateCluster"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "eks:DescribeCluster",
                "eks:DeleteCluster"
            ],
            "Resource": "arn:aws:eks:sa-east-1:111122223333:cluster/*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": [
                "arn:aws:iam::111122223333:role/StepFunctionsSample-EKSClusterManag-
EKSServiceRole-ANPAJ2UCCR6DPCEXAMPLE"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "eks.amazonaws.com"
                }
            }
        }
    ]
}
```

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sns:Publish"
            ],
            "Resource": [
                "arn:aws:sns:sa-east-1:111122223333:StepFunctionsSample-
EKSClusterManagement123456789012-SNSTopic-ANPAJ2UCCR6DPCEXAMPLE"
            ]
        }
```

```
    ]
}
```

Pour plus d'informations sur la configuration de l'IAM lors de l'utilisation de Step Functions avec d'autres AWS services, consultezPolitiques IAM pour les services intégrés.

Passez un appel à API Gateway

Cet exemple de projet montre comment utiliser Step Functions pour appeler API Gateway et vérifie si l'appel a réussi.

Pour plus d'informations sur les intégrations des services API Gateway et Step Functions, consultez les pages suivantes :

- Utilisation AWS Step Functions avec d'autres services
- Call API Gateway avec Step Functions

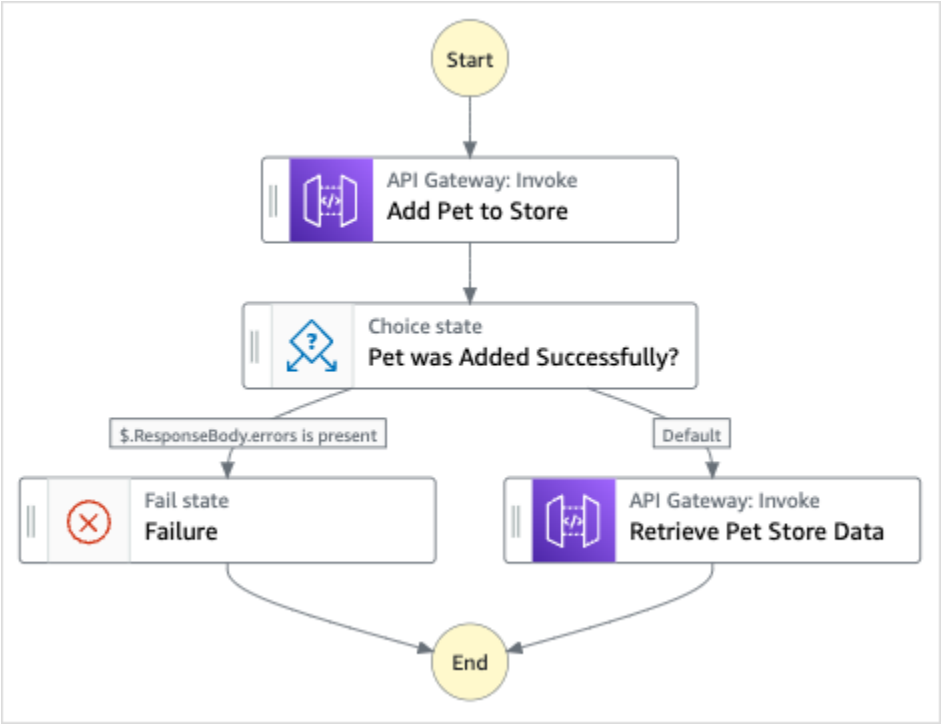
Étape 1 : créer la machine à états et provisionner les ressources

1. Ouvrez la console Step Functions et choisissez Create state machine.
2. Tapez **`Make a call to API Gateway`** dans la zone de recherche, puis choisissez Passer un appel API Gateway à dans les résultats de recherche renvoyés.
3. Choisissez Next (Suivant) pour continuer.
4. Step Functions répertorie les Services AWS éléments utilisés dans l'exemple de projet que vous avez sélectionné. Il montre également un graphique de flux de travail pour l'exemple de projet. Déployez ce projet sur votre site Compte AWS ou utilisez-le comme point de départ pour créer vos propres projets. Selon la façon dont vous souhaitez procéder, choisissez Exécuter une démo ou Construire à partir de celle-ci.

   Cet exemple de projet déploie les ressources suivantes :

   - Une Amazon API Gateway API REST appelée par la machine d'état.
   - Machine d'état AWS Step Functions
   - Rôles AWS Identity and Access Management (IAM) connexes

   L'image suivante montre le graphique du flux de travail de l'API Gatewayexemple de projet Make a call to :



5. Choisissez Utiliser le modèle pour poursuivre votre sélection.
6. Effectuez l'une des actions suivantes :

   - Si vous avez sélectionné Build on it, Step Functions crée le prototype de flux de travail pour l'exemple de projet que vous avez sélectionné. Step Functions ne déploie pas les ressources répertoriées dans la définition du flux de travail.

     Dans Workflow StudioMode de conception, glissez-déposez les états depuis le Navigateur d'états pour continuer à créer votre prototype de flux de travail. Vous pouvez également passer à un éditeur de code intégré similaire à VS Code pour mettre à jour la définition Amazon States Language (ASL) de votre machine à états dans la console Step Functions. Mode code Pour plus d'informations sur l'utilisation de Workflow Studio pour créer vos machines d'état, consultezUtilisation de Workflow Studio.

     > **Important**
     >
     > N'oubliez pas de mettre à jour l'espace réservé Amazon Resource Name (ARN) pour les ressources utilisées dans l'exemple de projet avant d'exécuter votre flux de travail.

   - Si vous avez sélectionné Run a demo, Step Functions crée un exemple de projet en lecture seule qui utilise un AWS CloudFormation modèle pour déployer les AWS ressources répertoriées dans ce modèle sur votre. Compte AWS

> **Tip**
>
> Pour afficher la définition de la machine à états de l'exemple de projet, choisissez Code.

Lorsque vous êtes prêt, choisissez Déployer et exécuter pour déployer l'exemple de projet et créer les ressources.

La création de ces ressources et des autorisations IAM associées peut prendre jusqu'à 10 minutes. Pendant le déploiement de vos ressources, vous pouvez ouvrir le lien CloudFormation Stack ID pour voir quelles ressources sont mises en service.

Une fois que toutes les ressources de l'exemple de projet ont été créées, vous pouvez voir le nouvel exemple de projet répertorié sur la page State machines.

> **Important**
>
> Des frais standard peuvent s'appliquer pour chaque service utilisé dans le CloudFormation modèle.

Étape 2 : Exécuter la machine à états

1. Sur la page State machines, choisissez votre exemple de projet.
2. Sur la page d'exemple de projet, choisissez Démarrer l'exécution.
3. Dans la boîte de dialogue Démarrer l'exécution, procédez comme suit :

    1. (Facultatif) Pour identifier votre exécution, vous pouvez lui donner un nom dans le champ Nom. Par défaut, Step Functions génère automatiquement un nom d'exécution unique.

        > **Note**
        >
        > Step Functions vous permet de créer des noms pour les machines d'état, les exécutions et les activités, ainsi que des étiquettes contenant des caractères non ASCII. Ces noms non ASCII ne fonctionnent pas avec Amazon. CloudWatch Pour

être sûr de pouvoir suivre CloudWatch les métriques, choisissez un nom qui utilise uniquement des caractères ASCII.

2. (Facultatif) Dans la zone de saisie, entrez les valeurs d'entrée au format JSON pour exécuter votre flux de travail.

   Si vous avez choisi d'exécuter une démo, vous n'avez pas besoin de fournir d'entrée d'exécution.

   > **Note**
   >
   > Si le projet de démonstration que vous avez déployé contient des données d'entrée d'exécution préremplies, utilisez ces entrées pour exécuter la machine à états.

3. Choisissez Start execution (Démarrer l'exécution).
4. La console Step Functions vous dirige vers une page intitulée avec votre ID d'exécution. Cette page est connue sous le nom de page Détails de l'exécution. Sur cette page, vous pouvez consulter les résultats de l'exécution au fur et à mesure que l'exécution progresse ou une fois celle-ci terminée.

   Pour consulter les résultats de l'exécution, choisissez des états individuels dans la vue graphique, puis choisissez les onglets individuels du Détails de l'étape volet pour afficher les détails de chaque état, y compris les entrées, les sorties et la définition respectivement. Pour plus de détails sur les informations d'exécution que vous pouvez consulter sur la page Détails de l'exécution, voirPage de détails d'exécution — Vue d'ensemble de l'interface.

Exemple de code de machine d'état

Dans cet exemple de projet, la machine à états s'intègre à API Gateway en appelant l'API REST API Gateway et en transmettant les paramètres nécessaires.

Parcourez cet exemple de machine à états pour voir comment Step Functions interagit avec API Gateway.

Pour plus d'informations sur la manière de AWS Step Functions contrôler d'autres AWS services, consultezUtilisation AWS Step Functions avec d'autres services.

```
{
  "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
    }
  }
}
```

Pour plus d'informations sur la configuration de l'IAM lors de l'utilisation de Step Functions avec d'autres AWS services, consultezPolitiques IAM pour les services intégrés.

Exemple IAM

Ces exemples de politiques AWS Identity and Access Management (IAM) générés par l'exemple de projet incluent le moindre privilège nécessaire pour exécuter la machine à états et les ressources associées. Nous vous recommandons de n'inclure que les autorisations nécessaires dans vos politiques IAM.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "execute-api:Invoke"
            ],
            "Resource": [
                "arn:aws:execute-api:us-west-1:111122223333:c8hqe4kdg5/default/GET/
pets",
                "arn:aws:execute-api:us-west-1:111122223333:c8hqe4kdg5/default/POST/
pets"
            ],
            "Effect": "Allow"
        }
    ]
}
```

Pour plus d'informations sur la configuration de l'IAM lors de l'utilisation de Step Functions avec d'autres AWS services, consultezPolitiques IAM pour les services intégrés.

Appelez un microservice s'exécutant sur Fargate à l'aide de l'intégration d'API Gateway

Cet exemple de projet montre comment utiliser Step Functions pour appeler API Gateway afin d'interagir avec un service activé AWS Fargate, et également pour vérifier si l'appel a réussi.

Pour plus d'informations sur les intégrations des services API Gateway et Step Functions, consultez les pages suivantes :

- Utilisation AWS Step Functions avec d'autres services
- Call API Gateway avec Step Functions

> **Note**
>
> Cet exemple de projet peut entraîner des frais.
> Pour AWS les nouveaux utilisateurs, un niveau d'utilisation gratuit est disponible. Dans cette offre, les services sont gratuits en-dessous d'un certain niveau d'utilisation. Pour plus d'informations sur AWS les coûts et le niveau gratuit, consultez la section Tarification.

Étape 1 : créer la machine à états et provisionner les ressources

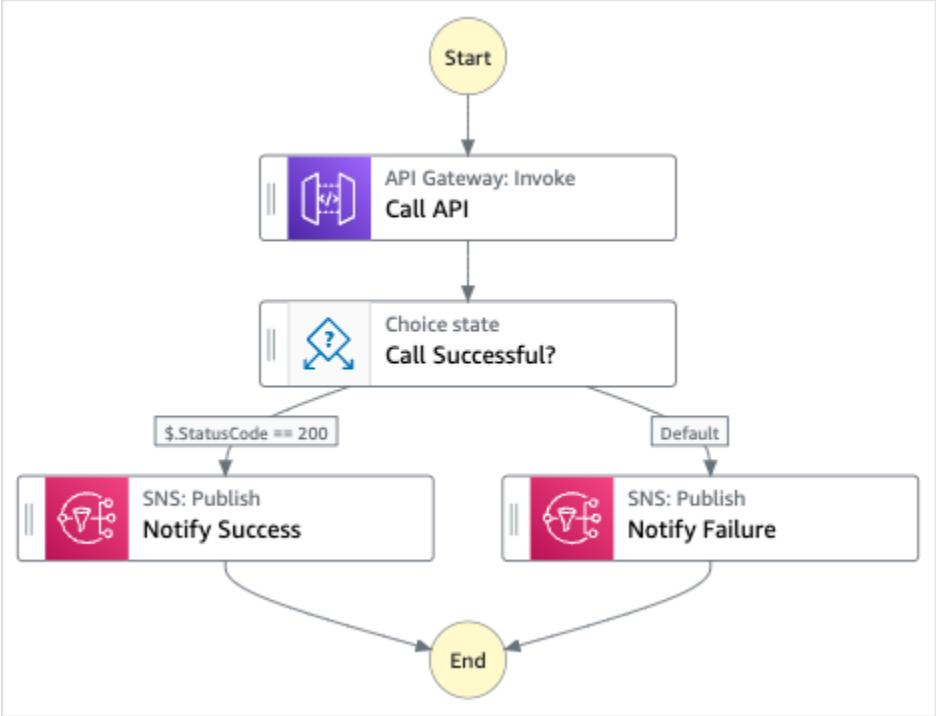
1. Ouvrez la console Step Functions et choisissez Create state machine.
2. Tapez **`Call a microservice with API Gateway`** dans la zone de recherche, puis choisissez Appeler un microservice avec dans les résultats API Gateway de recherche renvoyés.
3. Choisissez Next (Suivant) pour continuer.
4. Step Functions répertorie les Services AWS éléments utilisés dans l'exemple de projet que vous avez sélectionné. Il montre également un graphique de flux de travail pour l'exemple de projet. Déployez ce projet sur votre site Compte AWS ou utilisez-le comme point de départ pour créer vos propres projets. Selon la façon dont vous souhaitez procéder, choisissez Exécuter une démo ou Construire à partir de celle-ci.

   Cet exemple de projet déploie les ressources suivantes :

   - Une API Amazon API Gateway HTTP appelée par la machine d'état.
   - Un Amazon API Gateway Amazon VPC lien.

- Un Amazon Virtual Private Cloud.
- Un Application Load Balancer.
- Un Fargate cluster.
- Une rubrique Amazon SNS
- Une machine AWS Step Functions étatique
- Rôles associés AWS Identity and Access Management (IAM)
- Plusieurs services supplémentaires sont nécessaires pour permettre à ces ressources de fonctionner ensemble.

L'image suivante montre le graphique du flux de travail du projet Call a microservice with API Gateway sample :



5. Choisissez Utiliser le modèle pour poursuivre votre sélection.
6. Effectuez l'une des actions suivantes :

   - Si vous avez sélectionné Build on it, Step Functions crée le prototype de flux de travail pour l'exemple de projet que vous avez sélectionné. Step Functions ne déploie pas les ressources répertoriées dans la définition du flux de travail.

     Dans Workflow StudioMode de conception, glissez-déposez les états depuis le Navigateur d'états pour continuer à créer votre prototype de flux de travail. Vous pouvez également

passer à un éditeur de code intégré similaire à VS Code pour mettre à jour la définition Amazon States Language (ASL) de votre machine à états dans la console Step Functions. Mode code Pour plus d'informations sur l'utilisation de Workflow Studio pour créer vos machines d'état, consultezUtilisation de Workflow Studio.

> **Important**
>
> N'oubliez pas de mettre à jour l'espace réservé Amazon Resource Name (ARN) pour les ressources utilisées dans l'exemple de projet avant d'exécuter votre flux de travail.

- Si vous avez sélectionné Run a demo, Step Functions crée un exemple de projet en lecture seule qui utilise un AWS CloudFormation modèle pour déployer les AWS ressources répertoriées dans ce modèle sur votre. Compte AWS

  > **Tip**
  >
  > Pour afficher la définition de la machine à états de l'exemple de projet, choisissez Code.

  Lorsque vous êtes prêt, choisissez Déployer et exécuter pour déployer l'exemple de projet et créer les ressources.

  La création de ces ressources et des autorisations IAM associées peut prendre jusqu'à 10 minutes. Pendant le déploiement de vos ressources, vous pouvez ouvrir le lien CloudFormation Stack ID pour voir quelles ressources sont mises en service.

  Une fois que toutes les ressources de l'exemple de projet ont été créées, vous pouvez voir le nouvel exemple de projet répertorié sur la page State machines.

  > **Important**
  >
  > Des frais standard peuvent s'appliquer pour chaque service utilisé dans le CloudFormation modèle.

Étape 2 : Exécuter la machine à états

1. Sur la page State machines, choisissez votre exemple de projet.
2. Sur la page d'exemple de projet, choisissez Démarrer l'exécution.
3. Dans la boîte de dialogue Démarrer l'exécution, procédez comme suit :

    1. (Facultatif) Pour identifier votre exécution, vous pouvez lui donner un nom dans le champ Nom. Par défaut, Step Functions génère automatiquement un nom d'exécution unique.

        > **Note**
        >
        > Step Functions vous permet de créer des noms pour les machines d'état, les exécutions et les activités, ainsi que des étiquettes contenant des caractères non ASCII. Ces noms non ASCII ne fonctionnent pas avec Amazon. CloudWatch Pour être sûr de pouvoir suivre CloudWatch les métriques, choisissez un nom qui utilise uniquement des caractères ASCII.

    2. (Facultatif) Dans la zone de saisie, entrez les valeurs d'entrée au format JSON pour exécuter votre flux de travail.

        Si vous avez choisi d'exécuter une démo, vous n'avez pas besoin de fournir d'entrée d'exécution.

        > **Note**
        >
        > Si le projet de démonstration que vous avez déployé contient des données d'entrée d'exécution préremplies, utilisez ces entrées pour exécuter la machine à états.

    3. Choisissez Start execution (Démarrer l'exécution).
    4. La console Step Functions vous dirige vers une page intitulée avec votre ID d'exécution. Cette page est connue sous le nom de page Détails de l'exécution. Sur cette page, vous pouvez consulter les résultats de l'exécution au fur et à mesure que l'exécution progresse ou une fois celle-ci terminée.

        Pour consulter les résultats de l'exécution, choisissez des états individuels dans la vue graphique, puis choisissez les onglets individuels du Détails de l'étape volet pour afficher les détails de chaque état, y compris les entrées, les sorties et la définition respectivement. Pour

plus de détails sur les informations d'exécution que vous pouvez consulter sur la page Détails de l'exécution, voirPage de détails d'exécution — Vue d'ensemble de l'interface.

Exemple de code de machine d'état

Dans cet exemple de projet, la machine à états s'intègre à API Gateway en appelant une API HTTP API Gateway connectée à un service sur Fargate. Il est hébergé sur un sous-réseau privé et accessible via un équilibreur de charge d'application privé.

Parcourez cet exemple de machine à états pour voir comment Step Functions interagit avec API Gateway et renvoie des résultats.

Pour plus d'informations sur la manière de AWS Step Functions contrôler d'autres AWS services, consultezUtilisation AWS Step Functions avec d'autres services.

```
{
  "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
    }
  }
}
```

Pour plus d'informations sur la configuration de l'IAM lors de l'utilisation de Step Functions avec d'autres AWS services, consultezPolitiques IAM pour les services intégrés.

Exemple IAM

Ces exemples de politiques AWS Identity and Access Management (IAM) générés par l'exemple de projet incluent le moindre privilège nécessaire pour exécuter la machine à états et les ressources associées. Nous vous recommandons de n'inclure que les autorisations nécessaires dans vos politiques IAM.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "sns:Publish"
            ],
            "Resource": [
                "arn:aws:sns:us-east-1:111122223333:apigw-ecs-sample-2000-
SNSTopic-444455556666"
            ],
            "Effect": "Allow"
        },
        {
```

```
            "Action": [
                "execute-api:Invoke"
            ],
            "Resource": [
                "arn:aws:execute-api:us-east-1:111122223333:444444444444/*/GET/*"
            ],
            "Effect": "Allow"
        }
    ]
}
```

```
{
    "Statement": [
        {
            "Action": [
                "ec2:AttachNetworkInterface",
                "ec2:CreateNetworkInterface",
                "ec2:CreateNetworkInterfacePermission",
                "ec2:DeleteNetworkInterface",
                "ec2:DeleteNetworkInterfacePermission",
                "ec2:Describe*",
                "ec2:DetachNetworkInterface",
                "elasticloadbalancing:DeregisterInstancesFromLoadBalancer",
                "elasticloadbalancing:DeregisterTargets",
                "elasticloadbalancing:Describe*",
                "elasticloadbalancing:RegisterInstancesWithLoadBalancer",
                "elasticloadbalancing:RegisterTargets"
            ],
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

```
{
    "Statement": [
        {
            "Action": [
                "ecr:GetAuthorizationToken",
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage",
                "logs:CreateLogStream",
```

```
                "logs:PutLogEvents"
            ],
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

Pour plus d'informations sur la configuration de l'IAM lors de l'utilisation de Step Functions avec d'autres AWS services, consultezPolitiques IAM pour les services intégrés.

Envoyer un événement personnalisé à EventBridge

Cet exemple de projet montre comment utiliser Step Functions pour envoyer un événement personnalisé à un bus d'événements qui correspond à une règle comportant plusieurs cibles (Amazon EventBridge AWS Lambda, Amazon Simple Notification Service, Amazon Simple Queue Service).

Pour plus d'informations sur les intégrations de services Step Functions et Step Functions, consultez les rubriques suivantes :

- Utilisation AWS Step Functions avec d'autres services
- Appelez EventBridge avec Step Functions

> **Note**
>
> Cet exemple de projet peut entraîner des frais.
> Pour AWS les nouveaux utilisateurs, un niveau d'utilisation gratuit est disponible. Dans cette offre, les services sont gratuits en-dessous d'un certain niveau d'utilisation. Pour plus d'informations sur AWS les coûts et le niveau gratuit, consultez la section EventBridge Tarification.

Étape 1 : créer la machine à états et provisionner les ressources

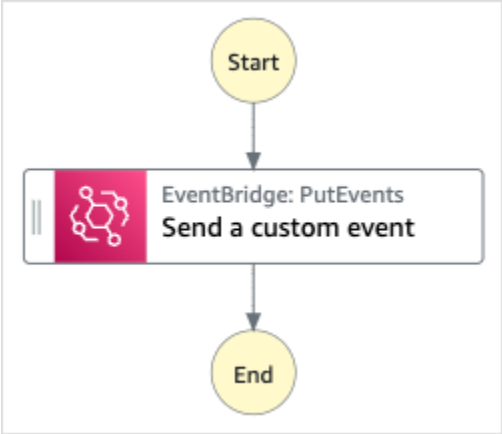
1. Ouvrez la console Step Functions et choisissez Create state machine.
2. Tapez **`Send a custom event to EventBridge`** dans la zone de recherche, puis choisissez Envoyer un événement personnalisé EventBridge à dans les résultats de recherche renvoyés.

3. Choisissez Next (Suivant) pour continuer.
4. Step Functions répertorie les Services AWS éléments utilisés dans l'exemple de projet que vous avez sélectionné. Il montre également un graphique de flux de travail pour l'exemple de projet. Déployez ce projet sur votre site Compte AWS ou utilisez-le comme point de départ pour créer vos propres projets. Selon la façon dont vous souhaitez procéder, choisissez Exécuter une démo ou Construire à partir de celle-ci.

   Cet exemple de projet déploie les ressources suivantes :

   - Un Amazon EventBridge événement
   - Une rubrique Amazon SNS
   - Une file d'attente Amazon SQS
   - Une fonction Lambda
   - Une machine AWS Step Functions étatique
   - Rôles associés AWS Identity and Access Management (IAM)

   L'image suivante montre le graphique du flux de travail du projet Envoyer un événement personnalisé vers un EventBridge exemple de projet :



5. Choisissez Utiliser le modèle pour poursuivre votre sélection.
6. Effectuez l’une des actions suivantes :

   - Si vous avez sélectionné Build on it, Step Functions crée le prototype de flux de travail pour l'exemple de projet que vous avez sélectionné. Step Functions ne déploie pas les ressources répertoriées dans la définition du flux de travail.

Dans Workflow StudioMode de conception, glissez-déposez les états depuis le Navigateur d'états pour continuer à créer votre prototype de flux de travail. Vous pouvez également passer à un éditeur de code intégré similaire à VS Code pour mettre à jour la définition Amazon States Language (ASL) de votre machine à états dans la console Step Functions. Mode code Pour plus d'informations sur l'utilisation de Workflow Studio pour créer vos machines d'état, consultezUtilisation de Workflow Studio.

> **Important**
>
> N'oubliez pas de mettre à jour l'espace réservé Amazon Resource Name (ARN) pour les ressources utilisées dans l'exemple de projet avant d'exécuter votre flux de travail.

- Si vous avez sélectionné Run a demo, Step Functions crée un exemple de projet en lecture seule qui utilise un AWS CloudFormation modèle pour déployer les AWS ressources répertoriées dans ce modèle sur votre. Compte AWS

  > **Tip**
  >
  > Pour afficher la définition de la machine à états de l'exemple de projet, choisissez Code.

  Lorsque vous êtes prêt, choisissez Déployer et exécuter pour déployer l'exemple de projet et créer les ressources.

  La création de ces ressources et des autorisations IAM associées peut prendre jusqu'à 10 minutes. Pendant le déploiement de vos ressources, vous pouvez ouvrir le lien CloudFormation Stack ID pour voir quelles ressources sont mises en service.

  Une fois que toutes les ressources de l'exemple de projet ont été créées, vous pouvez voir le nouvel exemple de projet répertorié sur la page State machines.

  > **Important**
  >
  > Des frais standard peuvent s'appliquer pour chaque service utilisé dans le CloudFormation modèle.

Étape 2 : Exécuter la machine à états

1. Sur la page State machines, choisissez votre exemple de projet.
2. Sur la page d'exemple de projet, choisissez Démarrer l'exécution.
3. Dans la boîte de dialogue Démarrer l'exécution, procédez comme suit :

    1. (Facultatif) Pour identifier votre exécution, vous pouvez lui donner un nom dans le champ Nom. Par défaut, Step Functions génère automatiquement un nom d'exécution unique.

        > **Note**
        >
        > Step Functions vous permet de créer des noms pour les machines d'état, les exécutions et les activités, ainsi que des étiquettes contenant des caractères non ASCII. Ces noms non ASCII ne fonctionnent pas avec Amazon. CloudWatch Pour être sûr de pouvoir suivre CloudWatch les métriques, choisissez un nom qui utilise uniquement des caractères ASCII.

    2. (Facultatif) Dans la zone de saisie, entrez les valeurs d'entrée au format JSON pour exécuter votre flux de travail.

        Si vous avez choisi d'exécuter une démo, vous n'avez pas besoin de fournir d'entrée d'exécution.

        > **Note**
        >
        > Si le projet de démonstration que vous avez déployé contient des données d'entrée d'exécution préremplies, utilisez ces entrées pour exécuter la machine à états.

    3. Choisissez Start execution (Démarrer l'exécution).
    4. La console Step Functions vous dirige vers une page intitulée avec votre ID d'exécution. Cette page est connue sous le nom de page Détails de l'exécution. Sur cette page, vous pouvez consulter les résultats de l'exécution au fur et à mesure que l'exécution progresse ou une fois celle-ci terminée.

        Pour consulter les résultats de l'exécution, choisissez des états individuels dans la vue graphique, puis choisissez les onglets individuels du Détails de l'étape volet pour afficher les détails de chaque état, y compris les entrées, les sorties et la définition respectivement. Pour

plus de détails sur les informations d'exécution que vous pouvez consulter sur la page Détails de l'exécution, voirPage de détails d'exécution — Vue d'ensemble de l'interface.

Exemple de code de machine d'état

Dans cet exemple de projet, la machine à états s'intègre EventBridge en envoyant un événement personnalisé à un bus d' EventBridge événements. L'événement envoyé au bus d'événements correspond à une EventBridge règle qui déclenche une fonction Lambda qui envoie des messages à une rubrique Amazon SNS et à une file d'attente Amazon SQS.

Parcourez cet exemple de machine à états pour voir comment Step Functions gère ses opérations EventBridge.

Pour plus d'informations sur la manière de AWS Step Functions contrôler d'autres AWS services, consultezUtilisation AWS Step Functions avec d'autres services.

```
{
  "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
    }
  }
}
```

Pour plus d'informations sur la configuration de l'IAM lors de l'utilisation de Step Functions avec d'autres AWS services, consultezPolitiques IAM pour les services intégrés.

Exemple IAM

Ces exemples de politiques AWS Identity and Access Management (IAM) générés par l'exemple de projet incluent le moindre privilège nécessaire pour exécuter la machine à états et les ressources associées. Nous vous recommandons de n'inclure que les autorisations nécessaires dans vos politiques IAM.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "events:PutEvents"
            ],
            "Resource": [
                "arn:aws:events:us-east-1:1234567890:event-bus/stepfunctions-
sampleproject-eventbus"
            ],
            "Effect": "Allow"
        }
    ]
}
```

Pour plus d'informations sur la configuration de l'IAM lors de l'utilisation de Step Functions avec d'autres AWS services, consultezPolitiques IAM pour les services intégrés.

Invoquer des flux de travail express synchrones

Cet exemple de projet montre comment invoquer des flux de travail synchrones Express via Amazon API Gateway pour gérer une base de données d'employés.

Dans ce projet, Step Functions utilise les points de terminaison API Gateway pour démarrer les flux de travail Step Functions Synchronous Express. Ils utilisent ensuite DynamoDB pour rechercher, ajouter et supprimer des employés dans une base de données d'employés.

Pour plus d'informations sur les flux de travail synchrones Express de Step Functions, consultezFlux de travail express synchrones et asynchrones.

> **Note**
>
> Cet exemple de projet peut entraîner des frais.

Pour AWS les nouveaux utilisateurs, un niveau d'utilisation gratuit est disponible. Dans cette offre, les services sont gratuits en-dessous d'un certain niveau d'utilisation. Pour plus d'informations sur AWS les coûts et le niveau gratuit, consultez la section Tarification de Step Functions.

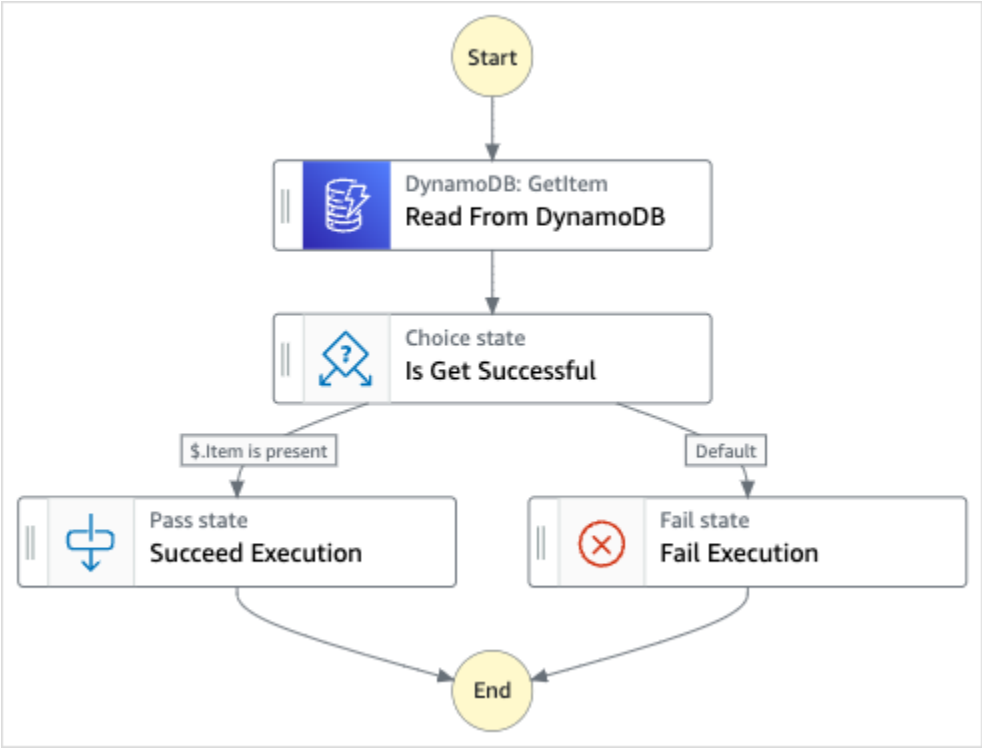
Étape 1 : créer la machine à états et provisionner les ressources

1. Ouvrez la console Step Functions et choisissez Create state machine.
2. Tapez **`Invoke Synchronous Express Workflows through API Gateway`** dans la zone de recherche, puis choisissez Invoke Synchronous Express Workflows through dans les résultats API Gateway de recherche renvoyés.
3. Choisissez Next (Suivant) pour continuer.
4. Step Functions répertorie les Services AWS éléments utilisés dans l'exemple de projet que vous avez sélectionné. Il montre également un graphique de flux de travail pour l'exemple de projet. Déployez ce projet sur votre site Compte AWS ou utilisez-le comme point de départ pour créer vos propres projets. Selon la façon dont vous souhaitez procéder, choisissez Exécuter une démo ou Construire à partir de celle-ci.

   Cet exemple de projet déploie les ressources suivantes :

   - Une API Amazon API Gateway HTTPS appelée par une machine d'état.
   - Une table Amazon DynamoDB.
   - Trois machines AWS Step Functions d'État.
   - Rôles associés AWS Identity and Access Management (IAM).

   L'image suivante montre le graphique du flux de travail pour les flux de travail Invoke Synchronous Express via API Gateway un exemple de projet :



5. Choisissez Utiliser le modèle pour poursuivre votre sélection.
6. Effectuez l'une des actions suivantes :

   - Si vous avez sélectionné Build on it, Step Functions crée le prototype de flux de travail pour l'exemple de projet que vous avez sélectionné. Step Functions ne déploie pas les ressources répertoriées dans la définition du flux de travail.

     Dans Workflow StudioMode de conception, glissez-déposez les états depuis le Navigateur d'états pour continuer à créer votre prototype de flux de travail. Vous pouvez également passer à un éditeur de code intégré similaire à VS Code pour mettre à jour la définition Amazon States Language (ASL) de votre machine à états dans la console Step Functions. Mode code Pour plus d'informations sur l'utilisation de Workflow Studio pour créer vos machines d'état, consultezUtilisation de Workflow Studio.

     > **Important**
     >
     > N'oubliez pas de mettre à jour l'espace réservé Amazon Resource Name (ARN) pour les ressources utilisées dans l'exemple de projet avant d'exécuter votre flux de travail.

- Si vous avez sélectionné Run a demo, Step Functions crée un exemple de projet en lecture seule qui utilise un AWS CloudFormation modèle pour déployer les AWS ressources répertoriées dans ce modèle sur votre. Compte AWS

  > **Tip**
  >
  > Pour afficher la définition de la machine à états de l'exemple de projet, choisissez Code.

  Lorsque vous êtes prêt, choisissez Déployer et exécuter pour déployer l'exemple de projet et créer les ressources.

  La création de ces ressources et des autorisations IAM associées peut prendre jusqu'à 10 minutes. Pendant le déploiement de vos ressources, vous pouvez ouvrir le lien CloudFormation Stack ID pour voir quelles ressources sont mises en service.

  Une fois que toutes les ressources de l'exemple de projet ont été créées, vous pouvez voir le nouvel exemple de projet répertorié sur la page State machines.

  > **Important**
  >
  > Des frais standard peuvent s'appliquer pour chaque service utilisé dans le CloudFormation modèle.

Étape 2 : Exécuter la machine à états

1. Sur la page State machines, choisissez votre exemple de projet.
2. Sur la page d'exemple de projet, choisissez Démarrer l'exécution.
3. Dans la boîte de dialogue Démarrer l'exécution, procédez comme suit :

   1. (Facultatif) Pour identifier votre exécution, vous pouvez lui donner un nom dans le champ Nom. Par défaut, Step Functions génère automatiquement un nom d'exécution unique.

> **Note**
>
> Step Functions vous permet de créer des noms pour les machines d'état, les exécutions et les activités, ainsi que des étiquettes contenant des caractères non ASCII. Ces noms non ASCII ne fonctionnent pas avec Amazon. CloudWatch Pour être sûr de pouvoir suivre CloudWatch les métriques, choisissez un nom qui utilise uniquement des caractères ASCII.

2. (Facultatif) Dans la zone de saisie, entrez les valeurs d'entrée au format JSON pour exécuter votre flux de travail.

   Si vous avez choisi d'exécuter une démo, vous n'avez pas besoin de fournir d'entrée d'exécution.

   > **Note**
   >
   > Si le projet de démonstration que vous avez déployé contient des données d'entrée d'exécution préremplies, utilisez ces entrées pour exécuter la machine à états.

3. Choisissez Start execution (Démarrer l'exécution).
4. La console Step Functions vous dirige vers une page intitulée avec votre ID d'exécution. Cette page est connue sous le nom de page Détails de l'exécution. Sur cette page, vous pouvez consulter les résultats de l'exécution au fur et à mesure que l'exécution progresse ou une fois celle-ci terminée.

   Pour consulter les résultats de l'exécution, choisissez des états individuels dans la vue graphique, puis choisissez les onglets individuels du Détails de l'étape volet pour afficher les détails de chaque état, y compris les entrées, les sorties et la définition respectivement. Pour plus de détails sur les informations d'exécution que vous pouvez consulter sur la page Détails de l'exécution, voirPage de détails d'exécution — Vue d'ensemble de l'interface.

Exemple de code de machine d'état

Dans cet exemple de projet, la machine à états s'intègre à API Gateway et DynamoDB en utilisant API Gateway pour appeler un flux de travail synchrone Express, qui met ensuite à jour ou lit la base de données des employés à l'aide de DynamoDB.

Parcourez cet exemple de machine à états pour découvrir comment Step Functions lit les données de DynamoDB pour récupérer les informations relatives aux employés.

Pour en savoir plus sur la façon d'invoquer Step Functions à l'aide d'API Gateway, consultez ce qui suit.

- Call API Gateway avec Step Functions
- Comment invoquer une passerelle privée dans le guide du développeur d'API Gateway.

Pour plus d'informations sur la manière de AWS Step Functions contrôler d'autres AWS services, consultezUtilisation AWS Step Functions avec d'autres services.

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

Pour plus d'informations sur la configuration de l'IAM lors de l'utilisation de Step Functions avec d'autres AWS services, consultezPolitiques IAM pour les services intégrés.

Exemples IAM

Ces exemples de politiques AWS Identity and Access Management (IAM) générés par l'exemple de projet incluent le moindre privilège nécessaire pour exécuter la machine à états et les ressources associées. Nous vous recommandons de n'inclure que les autorisations nécessaires dans vos politiques IAM.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogDelivery",
                "logs:GetLogDelivery",
                "logs:UpdateLogDelivery",
                "logs:DeleteLogDelivery",
```

```
                "logs:ListLogDeliveries",
                "logs:PutResourcePolicy",
                "logs:DescribeResourcePolicies",
                "logs:DescribeLogGroups"
            ],
            "Resource": "*"
        }
    ]
}
```

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "dynamodb:GetItem",
                "dynamodb:PutItem",
                "dynamodb:UpdateItem",
                "dynamodb:DeleteItem"
            ],
            "Resource": [
                "arn:aws:dynamodb:us-east-1:111122223333:table/Write"
            ]
        }
    ]
}
```

Pour plus d'informations sur la configuration de l'IAM lors de l'utilisation de Step Functions avec d'autres AWS services, consultezPolitiques IAM pour les services intégrés.

Exécutez des flux de travail ETL/ELT à l'aide d'Amazon Redshift (Lambda, API de données Amazon Redshift)

Cet exemple de projet montre comment utiliser Step Functions et l'API Amazon Redshift Data pour exécuter un flux de travail ETL/ELT qui charge des données dans l'entrepôt de données Amazon Redshift.

Dans ce projet, Step Functions utilise une AWS Lambda fonction et l'API Amazon Redshift Data pour créer les objets de base de données requis et générer un ensemble de données d'exemple, puis exécute deux tâches en parallèle qui consistent à charger des tables de dimensions, suivies d'une

table de faits. Une fois les deux tâches de chargement des dimensions terminées avec succès, Step Functions exécute la tâche de chargement pour la table d'information, exécute la tâche de validation, puis met en pause le cluster Amazon Redshift.

> **Note**
>
> Vous pouvez modifier la logique ETL pour recevoir des données provenant d'autres sources telles qu'Amazon S3, qui peut utiliser la commande COPY pour copier des données d'Amazon S3 vers une table Amazon Redshift.

Pour plus d'informations sur les intégrations des services Amazon Redshift et Step Functions, consultez les pages suivantes :

- Utilisation AWS Step Functions avec d'autres services
- Utilisation de l'API de données Amazon Redshift
- Service d'API de données Amazon Redshift
- Création d'une machine d'état Step Functions utilisant Lambda
- Politiques IAM pour :
  - Politiques IAM pour AWS Lambda
  - Autoriser l'accès à l'API Amazon Redshift Data

> **Note**
>
> Cet exemple de projet peut entraîner des frais.
> Pour AWS les nouveaux utilisateurs, un niveau d'utilisation gratuit est disponible. Dans cette offre, les services sont gratuits en-dessous d'un certain niveau d'utilisation. Pour plus d'informations sur AWS les coûts et le niveau gratuit, consultez AWS Step Functions les tarifs.

Étape 1 : créer la machine à états et provisionner les ressources

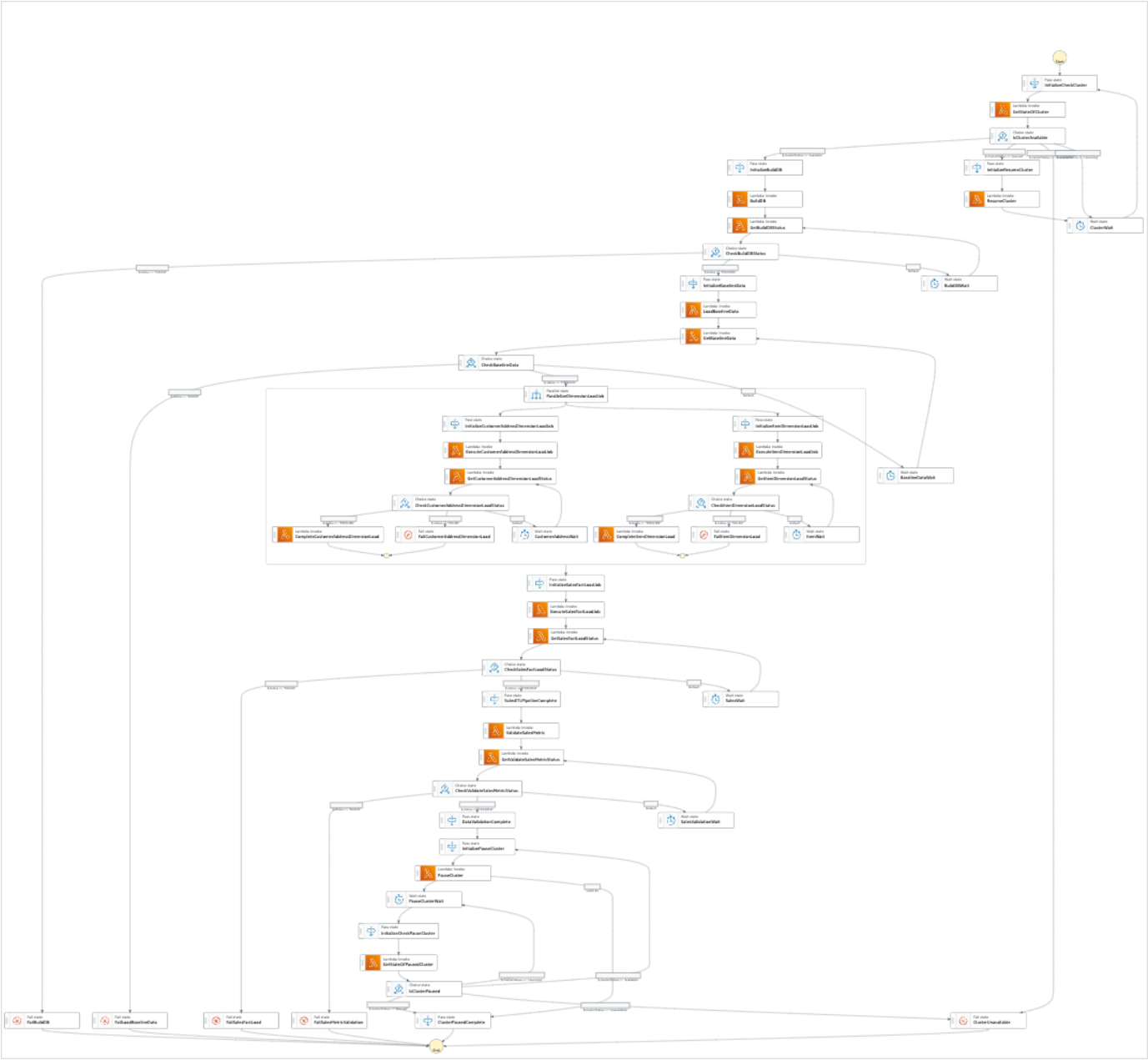
1. Ouvrez la console Step Functions et choisissez Create state machine.
2. Tapez **`ETL job in Amazon Redshift`** dans le champ de recherche, puis choisissez ETL job in dans les résultats Amazon Redshift de recherche renvoyés.

3. Choisissez Next (Suivant) pour continuer.
4. Step Functions répertorie les Services AWS éléments utilisés dans l'exemple de projet que vous avez sélectionné. Il montre également un graphique de flux de travail pour l'exemple de projet. Déployez ce projet sur votre site Compte AWS ou utilisez-le comme point de départ pour créer vos propres projets. Selon la façon dont vous souhaitez procéder, choisissez Exécuter une démo ou Construire à partir de celle-ci.

   Cet exemple de projet déploie les ressources suivantes :

   - Un Amazon Redshift cluster
   - Deux fonctions Lambda
   - Un Amazon Redshift schéma
   - Cinq Amazon Redshift tables
   - Une machine AWS Step Functions étatique
   - Rôles associés AWS Identity and Access Management (IAM).

   L'image suivante montre le graphique du flux de travail pour la tâche ETL dans Amazon Redshift un exemple de projet :



5. Choisissez Utiliser le modèle pour poursuivre votre sélection.
6. Effectuez l'une des actions suivantes :

   - Si vous avez sélectionné Build on it, Step Functions crée le prototype de flux de travail pour l'exemple de projet que vous avez sélectionné. Step Functions ne déploie pas les ressources répertoriées dans la définition du flux de travail.

     Dans Workflow StudioMode de conception, glissez-déposez les états depuis le Navigateur d'états pour continuer à créer votre prototype de flux de travail. Vous pouvez également passer à un éditeur de code intégré similaire à VS Code pour mettre à jour la définition

Amazon States Language (ASL) de votre machine à états dans la console Step Functions. Mode code Pour plus d'informations sur l'utilisation de Workflow Studio pour créer vos machines d'état, consultezUtilisation de Workflow Studio.

> **Important**
>
> N'oubliez pas de mettre à jour l'espace réservé Amazon Resource Name (ARN) pour les ressources utilisées dans l'exemple de projet avant d'exécuter votre flux de travail.

- Si vous avez sélectionné Run a demo, Step Functions crée un exemple de projet en lecture seule qui utilise un AWS CloudFormation modèle pour déployer les AWS ressources répertoriées dans ce modèle sur votre. Compte AWS

  > **Tip**
  >
  > Pour afficher la définition de la machine à états de l'exemple de projet, choisissez Code.

  Lorsque vous êtes prêt, choisissez Déployer et exécuter pour déployer l'exemple de projet et créer les ressources.

  La création de ces ressources et des autorisations IAM associées peut prendre jusqu'à 10 minutes. Pendant le déploiement de vos ressources, vous pouvez ouvrir le lien CloudFormation Stack ID pour voir quelles ressources sont mises en service.

  Une fois que toutes les ressources de l'exemple de projet ont été créées, vous pouvez voir le nouvel exemple de projet répertorié sur la page State machines.

  > **Important**
  >
  > Des frais standard peuvent s'appliquer pour chaque service utilisé dans le CloudFormation modèle.

Étape 2 : Exécuter la machine à états

1. Sur la page State machines, choisissez votre exemple de projet.

2. Sur la page d'exemple de projet, choisissez Démarrer l'exécution.
3. Dans la boîte de dialogue Démarrer l'exécution, procédez comme suit :

    1. (Facultatif) Pour identifier votre exécution, vous pouvez lui donner un nom dans le champ Nom. Par défaut, Step Functions génère automatiquement un nom d'exécution unique.

        > **Note**
        >
        > Step Functions vous permet de créer des noms pour les machines d'état, les exécutions et les activités, ainsi que des étiquettes contenant des caractères non ASCII. Ces noms non ASCII ne fonctionnent pas avec Amazon. CloudWatch Pour être sûr de pouvoir suivre CloudWatch les métriques, choisissez un nom qui utilise uniquement des caractères ASCII.

    2. (Facultatif) Dans la zone de saisie, entrez les valeurs d'entrée au format JSON pour exécuter votre flux de travail.

        Si vous avez choisi d'exécuter une démo, vous n'avez pas besoin de fournir d'entrée d'exécution.

        > **Note**
        >
        > Si le projet de démonstration que vous avez déployé contient des données d'entrée d'exécution préremplies, utilisez ces entrées pour exécuter la machine à états.

    3. Choisissez Start execution (Démarrer l'exécution).
    4. La console Step Functions vous dirige vers une page intitulée avec votre ID d'exécution. Cette page est connue sous le nom de page Détails de l'exécution. Sur cette page, vous pouvez consulter les résultats de l'exécution au fur et à mesure que l'exécution progresse ou une fois celle-ci terminée.

        Pour consulter les résultats de l'exécution, choisissez des états individuels dans la vue graphique, puis choisissez les onglets individuels du Détails de l'étape volet pour afficher les détails de chaque état, y compris les entrées, les sorties et la définition respectivement. Pour plus de détails sur les informations d'exécution que vous pouvez consulter sur la page Détails de l'exécution, voirPage de détails d'exécution — Vue d'ensemble de l'interface.

Exemple de code de machine d'état

Dans cet exemple de projet, la machine à états s'intègre AWS Lambda en transmettant la logique ETL InputPath directement à ces ressources et en l'exécutant de manière asynchrone à l'aide de l'API Amazon Redshift Data.

Parcourez cet exemple de machine à états pour découvrir comment Step Functions contrôle AWS Lambda l'API Amazon Redshift Data.

Pour plus d'informations sur la manière de AWS Step Functions contrôler d'autres AWS services, consultezUtilisation AWS Step Functions avec d'autres services.

```
{
"Comment": "A simple ETL workflow for loading dimension and fact tables",
  "StartAt": "InitializeCheckCluster",
  "States": {
    "InitializeCheckCluster": {
      "Type": "Pass",
      "Next": "GetStateOfCluster",
      "Result": {
        "input": {
          "redshift_cluster_id": "cfn36-redshiftcluster-AKIAI44QH8DHBEXAMPLE",
          "operation": "status"
        }
      }
    },
    "GetStateOfCluster": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftOperations-AKIAIOSFODNN7EXAMPLE",
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
      "Next": "IsClusterAvailable",
      "InputPath": "$",
      "ResultPath": "$.clusterStatus"
    },
    "IsClusterAvailable": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.clusterStatus",
          "StringEquals": "available",
          "Next": "InitializeBuildDB"
```

```
        },
        {
          "Variable": "$.clusterStatus",
          "StringEquals": "paused",
          "Next": "InitializeResumeCluster"
        },
        {
          "Variable": "$.clusterStatus",
          "StringEquals": "unavailable",
          "Next": "ClusterUnavailable"
        },
        {
          "Variable": "$.clusterStatus",
          "StringEquals": "resuming",
          "Next": "ClusterWait"
        }
      ]
    },
    "ClusterWait": {
      "Type": "Wait",
      "Seconds": 720,
      "Next": "InitializeCheckCluster"
    },
    "InitializeResumeCluster": {
      "Type": "Pass",
      "Next": "ResumeCluster",
      "Result": {
        "input": {
          "redshift_cluster_id": "cfn36-redshiftcluster-AKIAI44QH8DHBEXAMPLE",
          "operation": "resume"
        }
      }
    },
    "ResumeCluster": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftOperations-AKIAIOSFODNN7EXAMPLE",
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
      "Next": "ClusterWait",
      "InputPath": "$",
      "ResultPath": "$"
    },
    "InitializeBuildDB": {
```

```
      "Type": "Pass",
      "Next": "BuildDB",
      "Result": {
        "input": {
          "redshift_cluster_id": "cfn36-redshiftcluster-AKIAI44QH8DHBEXAMPLE",
          "redshift_database": "dev",
          "redshift_user": "awsuser",
          "redshift_schema": "tpcds",
          "action": "build_database",
          "sql_statement": [
            "create schema if not exists {0} authorization {1};",
            "create table if not exists {0}.customer",
            "(c_customer_sk          int4     not null encode az64",
            ",c_customer_id          char(16) not null encode zstd",
            ",c_current_addr_sk      int4              encode az64",
            ",c_first_name           char(20)          encode zstd",
            ",c_last_name            char(30)          encode zstd",
            ",primary key (c_customer_sk)",
            ") distkey(c_customer_sk);",
            "--",
            "create table if not exists {0}.customer_address",
            "(ca_address_sk    int4     not null encode az64",
            ",ca_address_id    char(16) not null encode zstd",
            ",ca_state         char(2)           encode zstd",
            ",ca_zip           char(10)          encode zstd",
            ",ca_country       varchar(20)       encode zstd",
            ",primary key (ca_address_sk)",
            ") distkey(ca_address_sk);",
            "--",
            "create table if not exists {0}.date_dim",
            "(d_date_sk           integer  not null encode az64",
            ",d_date_id           char(16) not null encode zstd",
            ",d_date              date              encode az64",
            ",d_day_name          char(9)           encode zstd",
            ",primary key (d_date_sk)",
            ") diststyle all;",
            "--",
            "create table if not exists {0}.item",
            "(i_item_sk        int4     not null encode az64",
            ",i_item_id        char(16) not null encode zstd",
            ",i_rec_start_date date              encode az64",
            ",i_rec_end_date   date              encode az64",
            ",i_current_price  numeric(7,2)      encode az64",
            ",i_category       char(50)          encode zstd",
```

```
            ",i_product_name   char(50)          encode zstd",
            ",primary key (i_item_sk)",
            ") distkey(i_item_sk) sortkey(i_category);",
            "--",
            "create table if not exists {0}.store_sales",
            "(ss_sold_date_sk       int4",
            ",ss_item_sk            int4 not null encode az64",
            ",ss_customer_sk        int4          encode az64",
            ",ss_addr_sk            int4          encode az64",
            ",ss_store_sk           int4          encode az64",
            ",ss_ticket_number      int8 not null encode az64",
            ",ss_quantity           int4          encode az64",
            ",ss_net_paid           numeric(7,2)  encode az64",
            ",ss_net_profit         numeric(7,2)  encode az64",
            ",primary key (ss_item_sk, ss_ticket_number)",
            ") distkey(ss_item_sk) sortkey(ss_sold_date_sk);"
          ]
        }
      }
    },
    "BuildDB": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
      "Next": "GetBuildDBStatus",
      "InputPath": "$",
      "ResultPath": "$"
    },
    "GetBuildDBStatus": {
      "Type": "Task",
      "Next": "CheckBuildDBStatus",
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
      "InputPath": "$",
      "ResultPath": "$.status"
    },
    "CheckBuildDBStatus": {
      "Type": "Choice",
      "Choices": [
        {
```

```
          "Variable": "$.status",
          "StringEquals": "FAILED",
          "Next": "FailBuildDB"
        },
        {
          "Variable": "$.status",
          "StringEquals": "FINISHED",
          "Next": "InitializeBaselineData"
        }
      ],
      "Default": "BuildDBWait"
    },
    "BuildDBWait": {
      "Type": "Wait",
      "Seconds": 15,
      "Next": "GetBuildDBStatus"
    },
    "FailBuildDB": {
      "Type": "Fail",
      "Cause": "Database Build Failed",
      "Error": "Error"
    },
    "InitializeBaselineData": {
      "Type": "Pass",
      "Next": "LoadBaselineData",
      "Result": {
        "input": {
          "redshift_cluster_id": "cfn36-redshiftcluster-AKIAI44QH8DHBEXAMPLE",
          "redshift_database": "dev",
          "redshift_user": "awsuser",
          "redshift_schema": "tpcds",
          "action": "load_baseline_data",
          "sql_statement": [
            "begin transaction;",
            "truncate table {0}.customer;",
            "insert into {0}.customer
  (c_customer_sk,c_customer_id,c_current_addr_sk,c_first_name,c_last_name)",
            "values",
            "(7550,'AAAAAAAAOHNBAAAA',9264662,'Michelle','Deaton'),",
            "(37079,'AAAAAAAAHNAJAAAA',13971208,'Michael','Simms'),",
            "(40626,'AAAAAAAACLOJAAAA',1959255,'Susan','Ryder'),",
            "(2142876,'AAAAAAAAMJCLACAA',7644556,'Justin','Brown');",
            "analyze {0}.customer;",
            "--",
```

```
            "truncate table {0}.customer_address;",
            "insert into {0}.customer_address
(ca_address_sk,ca_address_id,ca_state,ca_zip,ca_country)",
            "values",
            "(13971208,'AAAAAAAAIAPCFNAA','NE','63451','United States'),",
            "(7644556,'AAAAAAAAMIFKEHAA','SD','58883','United States'),",
            "(9264662,'AAAAAAAAGBOFNIAA','CA','99310','United States');",
            "analyze {0}.customer_address;",
            "--",
            "truncate table {0}.item;",
            "insert into {0}.item
(i_item_sk,i_item_id,i_rec_start_date,i_rec_end_date,i_current_price,i_category,i_product_name
            "values",

"(3417,'AAAAAAAAIFNAAAAA','1997-10-27',NULL,14.29,'Electronics','ationoughtesepri
'),",
            "(9615,'AAAAAAAAOIFCAAAA','1997-10-27',NULL,9.68,'Home','antioughtcallyn
st'),",
            "(3693,'AAAAAAAAMGOAAAAA','2001-03-12',NULL,2.10,'Men','prin
stcallypri'),",

"(3630,'AAAAAAAAMCOAAAAA','2001-10-27',NULL,2.95,'Electronics','barpricallypri'),",

"(16506,'AAAAAAAAIHAEAAAA','2001-10-27',NULL,3.85,'Home','callybaranticallyought'),",

"(7866,'AAAAAAAAILOBAAAA','2001-10-27',NULL,12.60,'Jewelry','callycallyeingation');",
            "--",
            "analyze {0}.item;",
            "truncate table {0}.date_dim;",
            "insert into {0}.date_dim (d_date_sk,d_date_id,d_date,d_day_name)",
            "values",
            "(2450521,'AAAAAAAAJFEGFCAA','1997-03-13','Thursday'),",
            "(2450749,'AAAAAAAANDFGFCAA','1997-10-27','Monday'),",
            "(2451251,'AAAAAAAADDHGFCAA','1999-03-13','Saturday'),",
            "(2451252,'AAAAAAAAEDHGFCAA','1999-03-14','Sunday'),",
            "(2451981,'AAAAAAAANAKGFCAA','2001-03-12','Monday'),",
            "(2451982,'AAAAAAAAOAKGFCAA','2001-03-13','Tuesday'),",
            "(2452210,'AAAAAAAACPKGFCAA','2001-10-27','Saturday'),",
            "(2452641,'AAAAAAAABKMGFCAA','2003-01-01','Wednesday'),",
            "(2452642,'AAAAAAAACKMGFCAA','2003-01-02','Thursday');",
            "--",
            "analyze {0}.date_dim;",
            "--  commit and End transaction",
            "commit;",
```

```
            "end transaction;"
          ]
        }
      }
    },
    "LoadBaselineData": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
      "Next": "GetBaselineData",
      "InputPath": "$",
      "ResultPath": "$"
    },
    "GetBaselineData": {
      "Type": "Task",
      "Next": "CheckBaselineData",
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
      "InputPath": "$",
      "ResultPath": "$.status"
    },
    "CheckBaselineData": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.status",
          "StringEquals": "FAILED",
          "Next": "FailLoadBaselineData"
        },
        {
          "Variable": "$.status",
          "StringEquals": "FINISHED",
          "Next": "ParallelizeDimensionLoadJob"
        }
      ],
      "Default": "BaselineDataWait"
    },
    "BaselineDataWait": {
      "Type": "Wait",
      "Seconds": 20,
```

```
      "Next": "GetBaselineData"
    },
    "FailLoadBaselineData": {
      "Type": "Fail",
      "Cause": "Load Baseline Data Failed",
      "Error": "Error"
    },
    "ParallelizeDimensionLoadJob": {
      "Type": "Parallel",
      "Next": "InitializeSalesFactLoadJob",
      "ResultPath": "$.status",
      "Branches": [
        {
          "StartAt": "InitializeCustomerAddressDimensionLoadJob",
          "States": {
            "InitializeCustomerAddressDimensionLoadJob": {
              "Type": "Pass",
              "Next": "ExecuteCustomerAddressDimensionLoadJob",
              "Result": {
                "input": {
                  "redshift_cluster_id": "cfn36-redshiftcluster-AKIAI44QH8DHBEXAMPLE",
                  "redshift_database": "dev",
                  "redshift_user": "awsuser",
                  "redshift_schema": "tpcds",
                  "action": "load_customer_address",
                  "sql_statement": [
                    "begin transaction;",
                    "/* Create a staging table to hold the input data. Staging table is
created with BACKUP NO option for faster inserts and also data temporary */",
                    "drop table if exists {0}.stg_customer_address;",
                    "create table if not exists {0}.stg_customer_address",
                    "(ca_address_id    varchar(16)  encode zstd",
                    ",ca_state         varchar(2)   encode zstd",
                    ",ca_zip           varchar(10)  encode zstd",
                    ",ca_country       varchar(20)  encode zstd",
                    ")",
                    "backup no",
                    "diststyle even;",
                    "/* Ingest data from source */",
                    "insert into {0}.stg_customer_address
(ca_address_id,ca_state,ca_zip,ca_country)",
                    "values",
                    "('AAAAAAAACFBBAAAA','NE','','United States'),",
                    "('AAAAAAAAGAEFAAAA','NE','61749','United States'),",
```

```
                    "('AAAAAAAAPJKKAAAA','OK','','United States'),",
                    "('AAAAAAAAMIHGAAAA','AL','','United States');",
                    "/* Perform UPDATE for existing data with refreshed attribute
 values */",
                    "update {0}.customer_address",
                    "   set ca_state = stg_customer_address.ca_state,",
                    "       ca_zip = stg_customer_address.ca_zip,",
                    "       ca_country = stg_customer_address.ca_country",
                    "  from {0}.stg_customer_address",
                    " where customer_address.ca_address_id =
 stg_customer_address.ca_address_id;",
                    "/* Perform insert for new rows  */",
                    "insert into {0}.customer_address",
                    "(ca_address_sk",
                    ",ca_address_id",
                    ",ca_state",
                    ",ca_zip",
                    ",ca_country",
                    ")",
                    "with max_customer_address_sk as",
                    "(select max(ca_address_sk) max_ca_address_sk",
                    "from {0}.customer_address)",
                    "select row_number() over (order by
 stg_customer_address.ca_address_id) + max_customer_address_sk.max_ca_address_sk as
 ca_address_sk",
                    ",stg_customer_address.ca_address_id",
                    ",stg_customer_address.ca_state",
                    ",stg_customer_address.ca_zip",
                    ",stg_customer_address.ca_country",
                    "from {0}.stg_customer_address,",
                    "max_customer_address_sk",
                    "where stg_customer_address.ca_address_id not in (select
 customer_address.ca_address_id from {0}.customer_address);",
                    "/* Commit and End transaction */",
                    "commit;",
                    "end transaction;"
                ]
              }
            }
          },
          "ExecuteCustomerAddressDimensionLoadJob": {
            "Type": "Task",
            "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
```

```
              "TimeoutSeconds": 180,
              "HeartbeatSeconds": 60,
              "Next": "GetCustomerAddressDimensionLoadStatus",
              "InputPath": "$",
              "ResultPath": "$"
            },
            "GetCustomerAddressDimensionLoadStatus": {
              "Type": "Task",
              "Next": "CheckCustomerAddressDimensionLoadStatus",
              "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
              "TimeoutSeconds": 180,
              "HeartbeatSeconds": 60,
              "InputPath": "$",
              "ResultPath": "$.status"
            },
            "CheckCustomerAddressDimensionLoadStatus": {
              "Type": "Choice",
              "Choices": [
                {
                  "Variable": "$.status",
                  "StringEquals": "FAILED",
                  "Next": "FailCustomerAddressDimensionLoad"
                },
                {
                  "Variable": "$.status",
                  "StringEquals": "FINISHED",
                  "Next": "CompleteCustomerAddressDimensionLoad"
                }
              ],
              "Default": "CustomerAddressWait"
            },
            "CustomerAddressWait": {
              "Type": "Wait",
              "Seconds": 5,
              "Next": "GetCustomerAddressDimensionLoadStatus"
            },
            "CompleteCustomerAddressDimensionLoad": {
              "Type": "Task",
              "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
              "TimeoutSeconds": 180,
              "HeartbeatSeconds": 60,
              "End": true
```

```
            },
            "FailCustomerAddressDimensionLoad": {
              "Type": "Fail",
              "Cause": "ETL Workflow Failed",
              "Error": "Error"
            }
          }
        },
        {
          "StartAt": "InitializeItemDimensionLoadJob",
          "States": {
            "InitializeItemDimensionLoadJob": {
              "Type": "Pass",
              "Next": "ExecuteItemDimensionLoadJob",
              "Result": {
                "input": {
                  "redshift_cluster_id": "cfn36-redshiftcluster-AKIAI44QH8DHBEXAMPLE",
                  "redshift_database": "dev",
                  "redshift_user": "awsuser",
                  "redshift_schema": "tpcds",
                  "action": "load_item",
                  "sql_statement": [
                    "begin transaction;",
                    "/* Create a staging table to hold the input data. Staging table is
created with BACKUP NO option for faster inserts and also data temporary */",
                    "drop table if exists {0}.stg_item;",
                    "create table if not exists {0}.stg_item",
                    "(i_item_id        varchar(16) encode zstd",
                    ",i_rec_start_date date encode zstd",
                    ",i_rec_end_date   date encode zstd",
                    ",i_current_price  numeric(7,2) encode zstd",
                    ",i_category       varchar(50) encode zstd",
                    ",i_product_name   varchar(50) encode zstd",
                    ")",
                    "backup no",
                    "diststyle even;",
                    "/* Ingest data from source */",
                    "insert into {0}.stg_item",

"(i_item_id,i_rec_start_date,i_rec_end_date,i_current_price,i_category,i_product_name)",
                    "values",

"('AAAAAAAAABJBAAAA','2000-10-27',NULL,4.10,'Books','ationoughtesecally'),",
```

```
                    "('AAAAAAAAOPKBAAAA','2001-10-27',NULL,4.22,'Books','ableoughtn
 stcally'),",
                    "('AAAAAAAAHGPAAAAA','1997-10-27',NULL,29.30,'Books','priesen
 stpri'),",

 "('AAAAAAAAICMAAAAA','2001-10-27',NULL,1.93,'Books','eseoughtoughtpri'),",

 "('AAAAAAAAGPGBAAAA','2001-10-27',NULL,9.96,'Books','bareingeinganti'),",
                    "('AAAAAAAAANBEBAAAA','1997-10-27',NULL,2.25,'Music','n
 steseoughtanti'),",

 "('AAAAAAAACLAAAAAA','2001-10-27',NULL,1.71,'Home','bareingought'),",

 "('AAAAAAAAOBBDAAAA','2001-10-27',NULL,5.55,'Books','callyationantiableought');",
                    "/
*************************************************************************************
                    "** Type 2 is maintained for i_current_price column.",
                    "** Update all attributes for the item when the price is not
 changed",
                    "** Sunset existing active item record with current i_rec_end_date
 and insert a new record when the price does not match",

 "*************************************************************************************
                    "update {0}.item",
                    "   set i_category = stg_item.i_category,",
                    "       i_product_name = stg_item.i_product_name",
                    "  from {0}.stg_item",
                    " where item.i_item_id = stg_item.i_item_id",
                    "   and item.i_rec_end_date is null",
                    "   and item.i_current_price = stg_item.i_current_price;",
                    "insert into {0}.item",
                    "(i_item_sk",
                    ",i_item_id",
                    ",i_rec_start_date",
                    ",i_rec_end_date",
                    ",i_current_price",
                    ",i_category",
                    ",i_product_name",
                    ")",
                    "with max_item_sk as",
                    "(select max(i_item_sk) max_item_sk",
                    "   from {0}.item)",
                    "select row_number() over (order by stg_item.i_item_id) +
 max_item_sk as i_item_sk",
```

```
                    "      ,stg_item.i_item_id",
                    "      ,trunc(sysdate) as i_rec_start_date",
                    "      ,null as i_rec_end_date",
                    "      ,stg_item.i_current_price",
                    "      ,stg_item.i_category",
                    "      ,stg_item.i_product_name",
                    "  from {0}.stg_item, {0}.item, max_item_sk",
                    " where item.i_item_id = stg_item.i_item_id",
                    "   and item.i_rec_end_date is null",
                    "   and item.i_current_price <> stg_item.i_current_price;",
                    "/* Sunset penultimate records that were inserted as type 2 */",
                    "update {0}.item",
                    "   set i_rec_end_date = trunc(sysdate)",
                    "  from {0}.stg_item",
                    " where item.i_item_id = stg_item.i_item_id",
                    "   and item.i_rec_end_date is null",
                    "   and item.i_current_price <> stg_item.i_current_price;",
                    "/* Commit and End transaction */",
                    "commit;",
                    "end transaction;"
                  ]
                }
              }
            },
            "ExecuteItemDimensionLoadJob": {
              "Type": "Task",
              "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
              "TimeoutSeconds": 180,
              "HeartbeatSeconds": 60,
              "Next": "GetItemDimensionLoadStatus",
              "InputPath": "$",
              "ResultPath": "$"
            },
            "GetItemDimensionLoadStatus": {
              "Type": "Task",
              "Next": "CheckItemDimensionLoadStatus",
              "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
              "TimeoutSeconds": 180,
              "HeartbeatSeconds": 60,
              "InputPath": "$",
              "ResultPath": "$.status"
            },
```

```
            "CheckItemDimensionLoadStatus": {
              "Type": "Choice",
              "Choices": [
                {
                  "Variable": "$.status",
                  "StringEquals": "FAILED",
                  "Next": "FailItemDimensionLoad"
                },
                {
                  "Variable": "$.status",
                  "StringEquals": "FINISHED",
                  "Next": "CompleteItemDimensionLoad"
                }
              ],
              "Default": "ItemWait"
            },
            "ItemWait": {
              "Type": "Wait",
              "Seconds": 5,
              "Next": "GetItemDimensionLoadStatus"
            },
            "CompleteItemDimensionLoad": {
              "Type": "Task",
              "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
              "TimeoutSeconds": 180,
              "HeartbeatSeconds": 60,
              "End": true
            },
            "FailItemDimensionLoad": {
              "Type": "Fail",
              "Cause": "ETL Workflow Failed",
              "Error": "Error"
            }
          }
        }
      ]
    },
    "InitializeSalesFactLoadJob": {
      "Type": "Pass",
      "Next": "ExecuteSalesFactLoadJob",
      "Result": {
        "input": {
          "redshift_cluster_id": "cfn36-redshiftcluster-AKIAI44QH8DHBEXAMPLE",
```

```
          "redshift_database": "dev",
          "redshift_user": "awsuser",
          "redshift_schema": "tpcds",
          "snapshot_date": "2003-01-02",
          "action": "load_sales_fact",
          "sql_statement": [
            "begin transaction;",
            "/* Create a stg_store_sales staging table */",
            "drop table if exists {0}.stg_store_sales;",
            "create table {0}.stg_store_sales",
            "(sold_date             date encode zstd",
            ",i_item_id             varchar(16) encode zstd",
            ",c_customer_id         varchar(16) encode zstd",
            ",ca_address_id         varchar(16) encode zstd",
            ",ss_ticket_number      integer encode zstd",
            ",ss_quantity           integer encode zstd",
            ",ss_net_paid           numeric(7,2) encode zstd",
            ",ss_net_profit         numeric(7,2) encode zstd",
            ")",
            "backup no",
            "diststyle even;",
            "/* Ingest data from source */",
            "insert into {0}.stg_store_sales",

"(sold_date,i_item_id,c_customer_id,ca_address_id,ss_ticket_number,ss_quantity,ss_net_paid,ss_
            "values",

"('2003-01-02','AAAAAAAAIFNAAAAA','AAAAAAAAOHNBAAAA','AAAAAAAAGBOFNIAA',1403191,13,5046.37,150

"('2003-01-02','AAAAAAAAIFNAAAAA','AAAAAAAAOHNBAAAA','AAAAAAAAGBOFNIAA',1403191,13,2103.72,-12

"('2003-01-02','AAAAAAAAILOBAAAA','AAAAAAAAOHNBAAAA','AAAAAAAAGBOFNIAA',1403191,13,959.10,-130

"('2003-01-02','AAAAAAAAILOBAAAA','AAAAAAAAHNAJAAAA','AAAAAAAAIAPCFNAA',1403191,13,962.65,-475

"('2003-01-02','AAAAAAAAMCOAAAAA','AAAAAAAAHNAJAAAA','AAAAAAAAIAPCFNAA',1201746,17,111.60,-241

"('2003-01-02','AAAAAAAAMCOAAAAA','AAAAAAAAHNAJAAAA','AAAAAAAAIAPCFNAA',1201746,17,4013.02,-11

"('2003-01-02','AAAAAAAAMCOAAAAA','AAAAAAAAMJCLACAA','AAAAAAAAMIFKEHAA',1201746,17,2689.12,-55

"('2003-01-02','AAAAAAAAMGOAAAAA','AAAAAAAAMJCLACAA','AAAAAAAAMIFKEHAA',193971,18,1876.89,-556
            "/* Delete any rows from target store_sales for the input date for
idempotency */",
```

```
            "delete from {0}.store_sales where ss_sold_date_sk in (select d_date_sk
 from {0}.date_dim where d_date='{1}');",
            "/* Insert data from staging table to the target table */",
            "insert into {0}.store_sales",
            "(ss_sold_date_sk",
            ",ss_item_sk",
            ",ss_customer_sk",
            ",ss_addr_sk",
            ",ss_ticket_number",
            ",ss_quantity",
            ",ss_net_paid",
            ",ss_net_profit",
            ")",
            "select date_dim.d_date_sk ss_sold_date_sk",
            "      ,item.i_item_sk ss_item_sk",
            "      ,customer.c_customer_sk ss_customer_sk",
            "      ,customer_address.ca_address_sk ss_addr_sk",
            "      ,ss_ticket_number",
            "      ,ss_quantity",
            "      ,ss_net_paid",
            "      ,ss_net_profit",
            "  from {0}.stg_store_sales as store_sales",
            "  inner join {0}.date_dim on store_sales.sold_date = date_dim.d_date",
            "  left join {0}.item on store_sales.i_item_id = item.i_item_id and
 item.i_rec_end_date is null",
            "  left join {0}.customer on store_sales.c_customer_id =
 customer.c_customer_id",
            "  left join {0}.customer_address on store_sales.ca_address_id =
 customer_address.ca_address_id;",
            "/* Drop staging table */",
            "drop table if exists {0}.stg_store_sales;",
            "/* Commit and End transaction */",
            "commit;",
            "end transaction;"
          ]
        }
      }
    },
    "ExecuteSalesFactLoadJob": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
```

```
      "Next": "GetSalesFactLoadStatus",
      "InputPath": "$",
      "ResultPath": "$"
    },
    "GetSalesFactLoadStatus": {
      "Type": "Task",
      "Next": "CheckSalesFactLoadStatus",
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
      "InputPath": "$",
      "ResultPath": "$.status"
    },
    "CheckSalesFactLoadStatus": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.status",
          "StringEquals": "FAILED",
          "Next": "FailSalesFactLoad"
        },
        {
          "Variable": "$.status",
          "StringEquals": "FINISHED",
          "Next": "SalesETLPipelineComplete"
        }
      ],
      "Default": "SalesWait"
    },
    "SalesWait": {
      "Type": "Wait",
      "Seconds": 5,
      "Next": "GetSalesFactLoadStatus"
    },
    "FailSalesFactLoad": {
      "Type": "Fail",
      "Cause": "ETL Workflow Failed",
      "Error": "Error"
    },
    "ClusterUnavailable": {
      "Type": "Fail",
      "Cause": "Redshift cluster is not available",
      "Error": "Error"
```

```
    },
    "SalesETLPipelineComplete": {
      "Type": "Pass",
      "Next": "ValidateSalesMetric",
      "Result": {
        "input": {
          "redshift_cluster_id": "cfn36-redshiftcluster-AKIAI44QH8DHBEXAMPLE",
          "redshift_database": "dev",
          "redshift_user": "awsuser",
          "redshift_schema": "tpcds",
          "snapshot_date": "2003-01-02",
          "action": "validate_sales_metric",
          "sql_statement": [
            "select 1/count(1) from {0}.store_sales where ss_sold_date_sk in (select
 d_date_sk from {0}.date_dim where d_date='{1}')"
          ]
        }
      }
    },
    "ValidateSalesMetric": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
      "Next": "GetValidateSalesMetricStatus",
      "InputPath": "$",
      "ResultPath": "$"
    },
    "GetValidateSalesMetricStatus": {
      "Type": "Task",
      "Next": "CheckValidateSalesMetricStatus",
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftDataApi-AIDACKCEVSQ6C2EXAMPLE",
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
      "InputPath": "$",
      "ResultPath": "$.status"
    },
    "CheckValidateSalesMetricStatus": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.status",
```

```
          "StringEquals": "FAILED",
          "Next": "FailSalesMetricValidation"
        },
        {
          "Variable": "$.status",
          "StringEquals": "FINISHED",
          "Next": "DataValidationComplete"
        }
      ],
      "Default": "SalesValidationWait"
    },
    "SalesValidationWait": {
      "Type": "Wait",
      "Seconds": 5,
      "Next": "GetValidateSalesMetricStatus"
    },
    "FailSalesMetricValidation": {
      "Type": "Fail",
      "Cause": "Data Validation Failed",
      "Error": "Error"
    },
    "DataValidationComplete": {
      "Type": "Pass",
      "Next": "InitializePauseCluster"
    },
    "InitializePauseCluster": {
      "Type": "Pass",
      "Next": "PauseCluster",
      "Result": {
        "input": {
          "redshift_cluster_id": "cfn36-redshiftcluster-AKIAI44QH8DHBEXAMPLE",
          "operation": "pause"
        }
      }
    },
    "PauseCluster": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftOperations-AKIAIOSFODNN7EXAMPLE",
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
      "Next": "PauseClusterWait",
      "InputPath": "$",
      "ResultPath": "$.clusterStatus",
```

```
      "Catch": [
        {
          "ErrorEquals": [
            "States.ALL"
          ],
          "Next": "ClusterPausedComplete"
        }
      ]
    },
    "InitializeCheckPauseCluster": {
      "Type": "Pass",
      "Next": "GetStateOfPausedCluster",
      "Result": {
        "input": {
          "redshift_cluster_id": "cfn36-redshiftcluster-AKIAI44QH8DHBEXAMPLE",
          "operation": "status"
        }
      }
    },
    "GetStateOfPausedCluster": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftOperations-AKIAIOSFODNN7EXAMPLE",
      "TimeoutSeconds": 180,
      "HeartbeatSeconds": 60,
      "Next": "IsClusterPaused",
      "InputPath": "$",
      "ResultPath": "$.clusterStatus"
    },
    "IsClusterPaused": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.clusterStatus",
          "StringEquals": "available",
          "Next": "InitializePauseCluster"
        },
        {
          "Variable": "$.clusterStatus",
          "StringEquals": "paused",
          "Next": "ClusterPausedComplete"
        },
        {
          "Variable": "$.clusterStatus",
```

```
          "StringEquals": "unavailable",
          "Next": "ClusterUnavailable"
        },
        {
          "Variable": "$.clusterStatus",
          "StringEquals": "resuming",
          "Next": "PauseClusterWait"
        }
      ]
    },
    "PauseClusterWait": {
      "Type": "Wait",
      "Seconds": 720,
      "Next": "InitializeCheckPauseCluster"
    },
    "ClusterPausedComplete": {
      "Type": "Pass",
      "End": true
    }
  }
}
```

Pour plus d'informations sur la configuration de l'IAM lors de l'utilisation de Step Functions avec d'autres AWS services, consultezPolitiques IAM pour les services intégrés.

Exemple IAM

Ces exemples de politiques AWS Identity and Access Management (IAM) générés par l'exemple de projet incluent le moindre privilège nécessaire pour exécuter la machine à états et les ressources associées. Nous vous recommandons de n'inclure que les autorisations nécessaires dans vos politiques IAM.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": [
                "arn:aws:lambda:us-east-1:111122223333:function:CFN36-RedshiftDataApi-
AIDACKCEVSQ6C2EXAMPLE",
```

```
                "arn:aws:lambda:us-east-1:111122223333:function:CFN36-
RedshiftOperations-AKIAIOSFODNN7EXAMPLE"
            ],
            "Effect": "Allow"
        }
    ]
}
```

Pour plus d'informations sur la configuration de l'IAM lors de l'utilisation de Step Functions avec d'autres AWS services, consultezPolitiques IAM pour les services intégrés.

Utilisation Step Functions et gestion AWS Batch des erreurs

Cet exemple de projet montre comment exécuter une AWS Batch tâche Step Functions à l'aide d'une machine à états dotée de capacités de gestion des erreurs.

Dans ce projet, Step Functions utilise une machine d'état pour appeler la tâche AWS Batch de manière synchrone. Il attend ensuite que la tâche réussisse ou échoue, réessaie et détecte les erreurs lorsqu'une tâche échoue, puis envoie un Amazon SNS sujet avec un message indiquant si la tâche a réussi ou échoué.

Étape 1 : créer la machine à états et provisionner les ressources

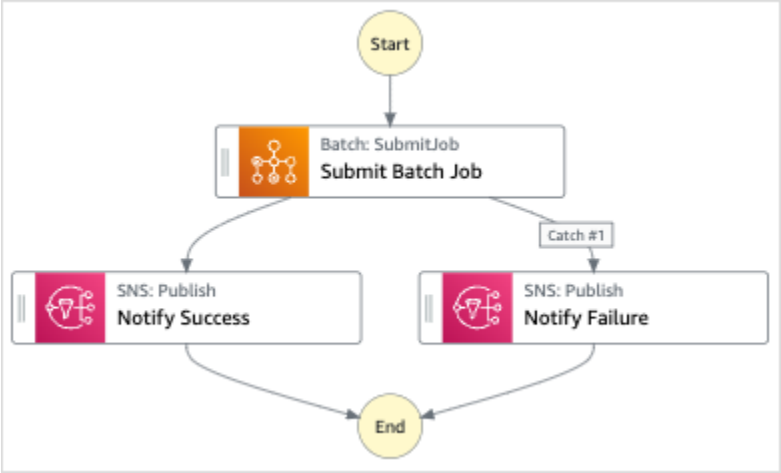
1. Ouvrez la console Step Functions et choisissez Create state machine.
2. Tapez **`Manage a batch job`** dans le champ de recherche, puis choisissez Gérer un traitement par lots dans les résultats de recherche renvoyés.
3. Choisissez Next (Suivant) pour continuer.
4. Step Functions répertorie les Services AWS éléments utilisés dans l'exemple de projet que vous avez sélectionné. Il montre également un graphique de flux de travail pour l'exemple de projet. Déployez ce projet sur votre site Compte AWS ou utilisez-le comme point de départ pour créer vos propres projets. Selon la façon dont vous souhaitez procéder, choisissez Exécuter une démo ou Construire à partir de celle-ci.

   Cet exemple de projet déploie les ressources suivantes :

   - Un AWS Batch travail
   - Une rubrique Amazon SNS
   - Une machine AWS Step Functions étatique

- Rôles associés AWS Identity and Access Management (IAM)

L'image suivante montre le graphique du flux de travail pour l'exemple de projet Gérer un travail par lots :



5. Choisissez Utiliser le modèle pour poursuivre votre sélection.
6. Effectuez l'une des actions suivantes :

- Si vous avez sélectionné Build on it, Step Functions crée le prototype de flux de travail pour l'exemple de projet que vous avez sélectionné. Step Functions ne déploie pas les ressources répertoriées dans la définition du flux de travail.

  Dans Workflow StudioMode de conception, glissez-déposez les états depuis le Navigateur d'états pour continuer à créer votre prototype de flux de travail. Vous pouvez également passer à un éditeur de code intégré similaire à VS Code pour mettre à jour la définition Amazon States Language (ASL) de votre machine à états dans la console Step Functions. Mode code Pour plus d'informations sur l'utilisation de Workflow Studio pour créer vos machines d'état, consultezUtilisation de Workflow Studio.

  > **Important**
  >
  > N'oubliez pas de mettre à jour l'espace réservé Amazon Resource Name (ARN) pour les ressources utilisées dans l'exemple de projet avant d'exécuter votre flux de travail.

- Si vous avez sélectionné Run a demo, Step Functions crée un exemple de projet en lecture seule qui utilise un AWS CloudFormation modèle pour déployer les AWS ressources répertoriées dans ce modèle sur votre. Compte AWS

> **Tip**
>
> Pour afficher la définition de la machine à états de l'exemple de projet, choisissez Code.

Lorsque vous êtes prêt, choisissez Déployer et exécuter pour déployer l'exemple de projet et créer les ressources.

La création de ces ressources et des autorisations IAM associées peut prendre jusqu'à 10 minutes. Pendant le déploiement de vos ressources, vous pouvez ouvrir le lien CloudFormation Stack ID pour voir quelles ressources sont mises en service.

Une fois que toutes les ressources de l'exemple de projet ont été créées, vous pouvez voir le nouvel exemple de projet répertorié sur la page State machines.

> **Important**
>
> Des frais standard peuvent s'appliquer pour chaque service utilisé dans le CloudFormation modèle.

Étape 2 : Exécuter la machine à états

1. Sur la page State machines, choisissez votre exemple de projet.
2. Sur la page d'exemple de projet, choisissez Démarrer l'exécution.
3. Dans la boîte de dialogue Démarrer l'exécution, procédez comme suit :

    1. (Facultatif) Pour identifier votre exécution, vous pouvez lui donner un nom dans le champ Nom. Par défaut, Step Functions génère automatiquement un nom d'exécution unique.

        > **Note**
        >
        > Step Functions vous permet de créer des noms pour les machines d'état, les exécutions et les activités, ainsi que des étiquettes contenant des caractères non ASCII. Ces noms non ASCII ne fonctionnent pas avec Amazon. CloudWatch Pour

être sûr de pouvoir suivre CloudWatch les métriques, choisissez un nom qui utilise uniquement des caractères ASCII.

2. (Facultatif) Dans la zone de saisie, entrez les valeurs d'entrée au format JSON pour exécuter votre flux de travail.

   Si vous avez choisi d'exécuter une démo, vous n'avez pas besoin de fournir d'entrée d'exécution.

   > **Note**
   >
   > Si le projet de démonstration que vous avez déployé contient des données d'entrée d'exécution préremplies, utilisez ces entrées pour exécuter la machine à états.

3. Choisissez Start execution (Démarrer l'exécution).
4. La console Step Functions vous dirige vers une page intitulée avec votre ID d'exécution. Cette page est connue sous le nom de page Détails de l'exécution. Sur cette page, vous pouvez consulter les résultats de l'exécution au fur et à mesure que l'exécution progresse ou une fois celle-ci terminée.

   Pour consulter les résultats de l'exécution, choisissez des états individuels dans la vue graphique, puis choisissez les onglets individuels du Détails de l'étape volet pour afficher les détails de chaque état, y compris les entrées, les sorties et la définition respectivement. Pour plus de détails sur les informations d'exécution que vous pouvez consulter sur la page Détails de l'exécution, voirPage de détails d'exécution — Vue d'ensemble de l'interface.

Exemple de code de machine d'état

Dans cet exemple de projet, la machine à états s'intègre AWS Batch à Amazon SNS en transmettant des paramètres directement à ces ressources.

Parcourez cet exemple de machine à états pour découvrir comment Step Functions contrôle AWS Batch Amazon SNS en vous connectant à l'Amazon Resource Name (ARN) `Resource` sur le terrain et en accédant `Parameters` à l'API du service.

Pour plus d'informations sur la manière de AWS Step Functions contrôler d'autres AWS services, consultezUtilisation AWS Step Functions avec d'autres services.

```
{
  "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
    }
  }
}
```

Exemple IAM

Cet exemple de politique AWS Identity and Access Management (IAM) généré par l'exemple de projet inclut le minimum de privilèges nécessaires pour exécuter la machine à états et les ressources associées. Nous vous recommandons de n'inclure que les autorisations nécessaires dans vos politiques IAM.

Example **`BatchJobNotificationAccessPolicy`**

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "sns:Publish"
            ],
            "Resource": [
                "arn:aws:sns:us-west-2:123456789012:StepFunctionsSample-
BatchJobManagement12345678-9abc-def0-1234-567890abcdef-SNSTopic-A2B3C4D5E6F7G"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "batch:SubmitJob",
                "batch:DescribeJobs",
                "batch:TerminateJob"
            ],
```

```
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "events:PutTargets",
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": [
                "arn:aws:events:us-west-2:123456789012:rule/
StepFunctionsGetEventsForBatchJobsRule"
            ],
            "Effect": "Allow"
        }
    ]
}
```

Pour plus d'informations sur la configuration de l'IAM lors de l'utilisation de Step Functions avec d'autres AWS services, consultezPolitiques IAM pour les services intégrés.

Répartissez un AWS Batch travail

Cet exemple de projet montre comment utiliser l'Mapétat de Step Functions pour répartir les AWS Batch tâches.

Dans ce projet, Step Functions utilise une machine à états pour appeler une fonction Lambda afin d'effectuer un prétraitement simple, puis invoque plusieurs tâches AWS Batch en parallèle en utilisant l'état. Map

Étape 1 : créer la machine à états et provisionner les ressources

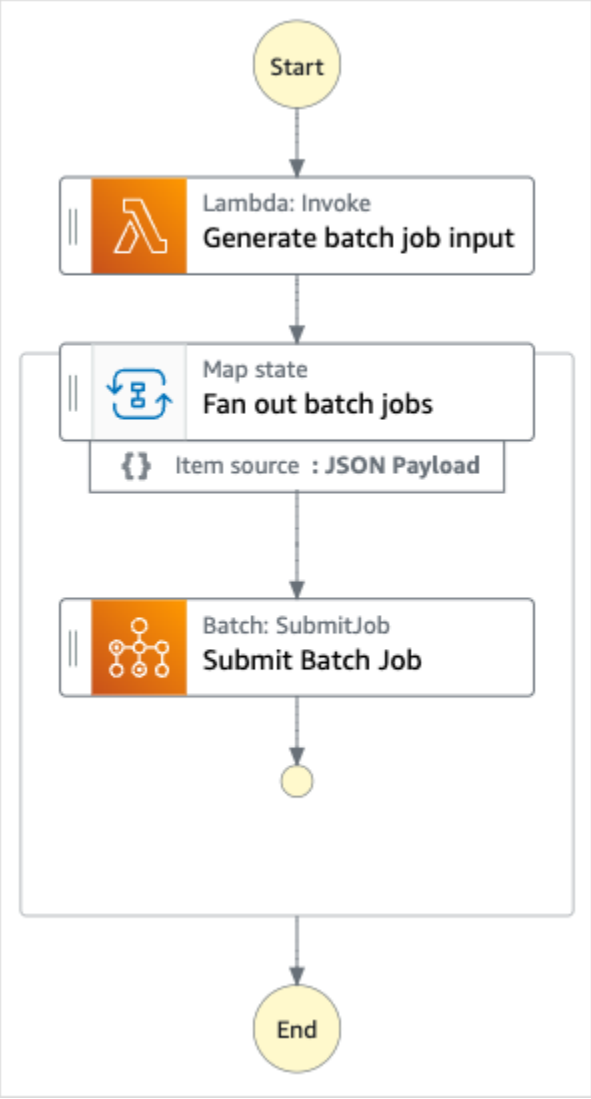
1. Ouvrez la console Step Functions et choisissez Create state machine.
2. Tapez **`Fan out a batch job`** dans la zone de recherche, puis choisissez Ventiler une tâche par lots dans les résultats de recherche renvoyés.
3. Choisissez Next (Suivant) pour continuer.
4. Step Functions répertorie les Services AWS éléments utilisés dans l'exemple de projet que vous avez sélectionné. Il montre également un graphique de flux de travail pour l'exemple de projet. Déployez ce projet sur votre site Compte AWS ou utilisez-le comme point de départ pour créer

vos propres projets. Selon la façon dont vous souhaitez procéder, choisissez Exécuter une démo ou Construire à partir de celle-ci.

Cet exemple de projet déploie les ressources suivantes :

- Une fonction Lambda
- Une file d'attente de AWS Batch tâches
- Une machine AWS Step Functions étatique
- Rôles associés AWS Identity and Access Management (IAM)

L'image suivante montre le graphique du flux de travail pour le projet d'exemple Fan out a batch job :



5. Choisissez Utiliser le modèle pour poursuivre votre sélection.

6. Effectuez l'une des actions suivantes :

   - Si vous avez sélectionné Build on it, Step Functions crée le prototype de flux de travail pour l'exemple de projet que vous avez sélectionné. Step Functions ne déploie pas les ressources répertoriées dans la définition du flux de travail.

     Dans Workflow StudioMode de conception, glissez-déposez les états depuis le Navigateur d'états pour continuer à créer votre prototype de flux de travail. Vous pouvez également passer à un éditeur de code intégré similaire à VS Code pour mettre à jour la définition Amazon States Language (ASL) de votre machine à états dans la console Step Functions. Mode code Pour plus d'informations sur l'utilisation de Workflow Studio pour créer vos machines d'état, consultezUtilisation de Workflow Studio.

     > **Important**
     >
     > N'oubliez pas de mettre à jour l'espace réservé Amazon Resource Name (ARN) pour les ressources utilisées dans l'exemple de projet avant d'exécuter votre flux de travail.

   - Si vous avez sélectionné Run a demo, Step Functions crée un exemple de projet en lecture seule qui utilise un AWS CloudFormation modèle pour déployer les AWS ressources répertoriées dans ce modèle sur votre. Compte AWS

     > **Tip**
     >
     > Pour afficher la définition de la machine à états de l'exemple de projet, choisissez Code.

     Lorsque vous êtes prêt, choisissez Déployer et exécuter pour déployer l'exemple de projet et créer les ressources.

     La création de ces ressources et des autorisations IAM associées peut prendre jusqu'à 10 minutes. Pendant le déploiement de vos ressources, vous pouvez ouvrir le lien CloudFormation Stack ID pour voir quelles ressources sont mises en service.

     Une fois que toutes les ressources de l'exemple de projet ont été créées, vous pouvez voir le nouvel exemple de projet répertorié sur la page State machines.

> **Important**
>
> Des frais standard peuvent s'appliquer pour chaque service utilisé dans le CloudFormation modèle.

Étape 2 : Exécuter la machine à états

1. Sur la page State machines, choisissez votre exemple de projet.
2. Sur la page d'exemple de projet, choisissez Démarrer l'exécution.
3. Dans la boîte de dialogue Démarrer l'exécution, procédez comme suit :

    1. (Facultatif) Pour identifier votre exécution, vous pouvez lui donner un nom dans le champ Nom. Par défaut, Step Functions génère automatiquement un nom d'exécution unique.

        > **Note**
        >
        > Step Functions vous permet de créer des noms pour les machines d'état, les exécutions et les activités, ainsi que des étiquettes contenant des caractères non ASCII. Ces noms non ASCII ne fonctionnent pas avec Amazon. CloudWatch Pour être sûr de pouvoir suivre CloudWatch les métriques, choisissez un nom qui utilise uniquement des caractères ASCII.

    2. (Facultatif) Dans la zone de saisie, entrez les valeurs d'entrée au format JSON pour exécuter votre flux de travail.

        Si vous avez choisi d'exécuter une démo, vous n'avez pas besoin de fournir d'entrée d'exécution.

        > **Note**
        >
        > Si le projet de démonstration que vous avez déployé contient des données d'entrée d'exécution préremplies, utilisez ces entrées pour exécuter la machine à états.

    3. Choisissez Start execution (Démarrer l'exécution).
    4. La console Step Functions vous dirige vers une page intitulée avec votre ID d'exécution. Cette page est connue sous le nom de page Détails de l'exécution. Sur cette page, vous pouvez

consulter les résultats de l'exécution au fur et à mesure que l'exécution progresse ou une fois celle-ci terminée.

Pour consulter les résultats de l'exécution, choisissez des états individuels dans la vue graphique, puis choisissez les onglets individuels du Détails de l'étape volet pour afficher les détails de chaque état, y compris les entrées, les sorties et la définition respectivement. Pour plus de détails sur les informations d'exécution que vous pouvez consulter sur la page Détails de l'exécution, voirPage de détails d'exécution — Vue d'ensemble de l'interface.

Exemple de code de machine d'état

Dans cet exemple de projet, la machine à états s'intègre AWS Batch à Amazon SNS en transmettant des paramètres directement à ces ressources.

Parcourez cet exemple de machine à états pour découvrir comment Step Functions contrôle AWS Batch Amazon SNS en vous connectant à l'Amazon Resource Name (ARN) `Resource` sur le terrain et en accédant `Parameters` à l'API du service.

Pour plus d'informations sur la manière de AWS Step Functions contrôler d'autres AWS services, consultezUtilisation AWS Step Functions avec d'autres services.

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

Exemple IAM

Ces exemples de politiques AWS Identity and Access Management (IAM) générés par l'exemple de projet incluent le minimum de privilèges nécessaires pour exécuter la machine à états et les ressources associées. Nous vous recommandons de n'inclure que les autorisations nécessaires dans vos politiques IAM.

Example **`BatchJobFanOutAccessPolicy`**

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "batch:SubmitJob",
                "batch:DescribeJobs",
                "batch:TerminateJob"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
```

```
        {
            "Action": [
                "events:PutTargets",
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": [
                "arn:aws:events:us-west-2:123456789012:rule/
StepFunctionsGetEventsForBatchJobsRule"
            ],
            "Effect": "Allow"
        }
    ]
}
```

Example **`InvokeGenerateBatchJobMapLambdaPolicy`**

```
{
    "Statement": [
        {
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": "arn:aws:lambda:us-
west-2:123456789012:function:StepFunctionsSample-BatchJobFa-
GenerateBatchJobMap-444455556666",
            "Effect": "Allow"
        }
    ]
}
```

Pour plus d'informations sur la configuration de l'IAM lors de l'utilisation de Step Functions avec d'autres AWS services, consultezPolitiques IAM pour les services intégrés.

AWS Batch avec Lambda

Cet exemple de projet montre comment utiliser Step Functions pour prétraiter des données à l'aide de AWS Lambda fonctions, puis orchestrer AWS Batch des tâches.

Dans ce projet, Step Functions utilise une machine à états pour appeler une fonction Lambda afin d'effectuer un prétraitement simple avant qu'une AWS Batch tâche ne soit soumise. Plusieurs tâches peuvent être invoquées en fonction du résultat ou du succès de la précédente.

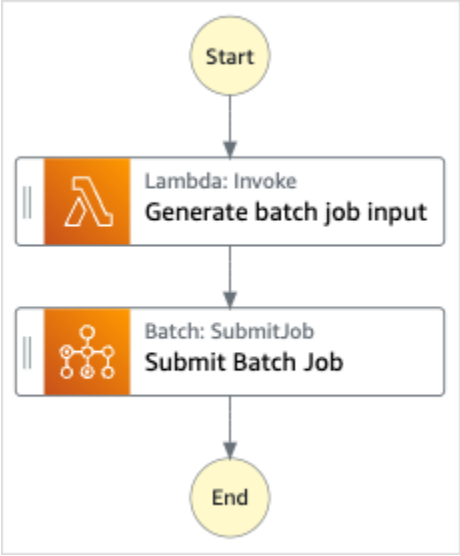
Étape 1 : créer la machine à états et provisionner les ressources

1. Ouvrez la console Step Functions et choisissez Create state machine.
2. Tapez **`Batch job with Lambda`** dans le champ de recherche, puis choisissez Batch job with dans les résultats Lambda de recherche renvoyés.
3. Choisissez Next (Suivant) pour continuer.
4. Step Functions répertorie les Services AWS éléments utilisés dans l'exemple de projet que vous avez sélectionné. Il montre également un graphique de flux de travail pour l'exemple de projet. Déployez ce projet sur votre site Compte AWS ou utilisez-le comme point de départ pour créer vos propres projets. Selon la façon dont vous souhaitez procéder, choisissez Exécuter une démo ou Construire à partir de celle-ci.

   Cet exemple de projet déploie les ressources suivantes :

   - Une fonction Lambda
   - Une tâche AWS Batch
   - Une machine AWS Step Functions étatique
   - Rôles associés AWS Identity and Access Management (IAM)

   L'image suivante montre le graphique du flux de travail pour le travail Batch avec Lambda un exemple de projet :

5. Choisissez Utiliser le modèle pour poursuivre votre sélection.
6. Effectuez l’une des actions suivantes :

- Si vous avez sélectionné Build on it, Step Functions crée le prototype de flux de travail pour l'exemple de projet que vous avez sélectionné. Step Functions ne déploie pas les ressources répertoriées dans la définition du flux de travail.

  Dans Workflow StudioMode de conception, glissez-déposez les états depuis le Navigateur d'états pour continuer à créer votre prototype de flux de travail. Vous pouvez également passer à un éditeur de code intégré similaire à VS Code pour mettre à jour la définition Amazon States Language (ASL) de votre machine à états dans la console Step Functions. Mode code Pour plus d'informations sur l'utilisation de Workflow Studio pour créer vos machines d'état, consultezUtilisation de Workflow Studio.

  > **Important**
  >
  > N'oubliez pas de mettre à jour l'espace réservé Amazon Resource Name (ARN) pour les ressources utilisées dans l'exemple de projet avant d'exécuter votre flux de travail.

- Si vous avez sélectionné Run a demo, Step Functions crée un exemple de projet en lecture seule qui utilise un AWS CloudFormation modèle pour déployer les AWS ressources répertoriées dans ce modèle sur votre. Compte AWS

  > **Tip**
  >
  > Pour afficher la définition de la machine à états de l'exemple de projet, choisissez Code.

  Lorsque vous êtes prêt, choisissez Déployer et exécuter pour déployer l'exemple de projet et créer les ressources.

  La création de ces ressources et des autorisations IAM associées peut prendre jusqu'à 10 minutes. Pendant le déploiement de vos ressources, vous pouvez ouvrir le lien CloudFormation Stack ID pour voir quelles ressources sont mises en service.

  Une fois que toutes les ressources de l'exemple de projet ont été créées, vous pouvez voir le nouvel exemple de projet répertorié sur la page State machines.

> **Important**
>
> Des frais standard peuvent s'appliquer pour chaque service utilisé dans le CloudFormation modèle.

Étape 2 : Exécuter la machine à états

1. Sur la page State machines, choisissez votre exemple de projet.
2. Sur la page d'exemple de projet, choisissez Démarrer l'exécution.
3. Dans la boîte de dialogue Démarrer l'exécution, procédez comme suit :

    1. (Facultatif) Pour identifier votre exécution, vous pouvez lui donner un nom dans le champ Nom. Par défaut, Step Functions génère automatiquement un nom d'exécution unique.

        > **Note**
        >
        > Step Functions vous permet de créer des noms pour les machines d'état, les exécutions et les activités, ainsi que des étiquettes contenant des caractères non ASCII. Ces noms non ASCII ne fonctionnent pas avec Amazon. CloudWatch Pour être sûr de pouvoir suivre CloudWatch les métriques, choisissez un nom qui utilise uniquement des caractères ASCII.

    2. (Facultatif) Dans la zone de saisie, entrez les valeurs d'entrée au format JSON pour exécuter votre flux de travail.

        Si vous avez choisi d'exécuter une démo, vous n'avez pas besoin de fournir d'entrée d'exécution.

        > **Note**
        >
        > Si le projet de démonstration que vous avez déployé contient des données d'entrée d'exécution préremplies, utilisez ces entrées pour exécuter la machine à états.

    3. Choisissez Start execution (Démarrer l'exécution).
    4. La console Step Functions vous dirige vers une page intitulée avec votre ID d'exécution. Cette page est connue sous le nom de page Détails de l'exécution. Sur cette page, vous pouvez

consulter les résultats de l'exécution au fur et à mesure que l'exécution progresse ou une fois celle-ci terminée.

Pour consulter les résultats de l'exécution, choisissez des états individuels dans la vue graphique, puis choisissez les onglets individuels du Détails de l'étape volet pour afficher les détails de chaque état, y compris les entrées, les sorties et la définition respectivement. Pour plus de détails sur les informations d'exécution que vous pouvez consulter sur la page Détails de l'exécution, voirPage de détails d'exécution — Vue d'ensemble de l'interface.

Exemple de code de machine d'état

Dans cet exemple de projet, la machine à états s'intègre AWS Batch à Amazon SNS en transmettant des paramètres directement à ces ressources.

Parcourez cet exemple de machine à états pour découvrir comment Step Functions contrôle AWS Batch Amazon SNS en vous connectant à l'Amazon Resource Name (ARN) `Resource` sur le terrain et en accédant `Parameters` à l'API du service.

Pour plus d'informations sur la manière de AWS Step Functions contrôler d'autres AWS services, consultezUtilisation AWS Step Functions avec d'autres services.

```
{
  "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
    }
  }
}
```

Exemple IAM

Ces exemples de politiques AWS Identity and Access Management (IAM) générés par l'exemple de projet incluent le minimum de privilèges nécessaires pour exécuter la machine à états et les ressources associées. Nous vous recommandons de n'inclure que les autorisations nécessaires dans vos politiques IAM.

Example **BatchJobWithLambdaAccessPolicy**

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "sns:Publish"
            ],
            "Resource": [
                "arn:aws:sns:us-west-2:123456789012:ManageBatchJob-SNSTopic-
JHLYYG7AZPZI"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "batch:SubmitJob",
                "batch:DescribeJobs",
                "batch:TerminateJob"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": [
```

```
                "events:PutTargets",
                "events:PutRule",
                "events:DescribeRule"
            ],
            "Resource": [
                "arn:aws:events:us-west-2:123456789012:rule/
StepFunctionsGetEventsForBatchJobsRule"
            ],
            "Effect": "Allow"
        }
    ]
}
```

Example **`InvokeGenerateBatchJobMapLambdaPolicy`**

```
{
    "Statement": [
        {
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": "arn:aws:lambda:us-
west-2:123456789012:function:StepFunctionsSample-BatchWithL-
GenerateBatchJobMap-444455556666",
            "Effect": "Allow"
        }
    ]
}
```

Pour plus d'informations sur la configuration de l'IAM lors de l'utilisation de Step Functions avec d'autres AWS services, consultezPolitiques IAM pour les services intégrés.

Réalisez un enchaînement d'instructions basé sur l'IA avec Amazon Bedrock

Cet exemple de projet montre comment vous pouvez l'intégrer Amazon Bedrock pour effectuer un chaînage d'instructions basé sur l'IA. Cet exemple de projet montre comment créer des chatbots de haute qualité en utilisantAmazon Bedrock. Le projet regroupe certaines instructions et les résout dans l'ordre dans lequel elles sont fournies. L'enchaînement de ces instructions augmente la capacité du modèle de langage utilisé à fournir une réponse parfaitement organisée.

Cet exemple de projet crée la machine d'état, les AWS ressources de support et configure les autorisations IAM associées. Explorez cet exemple de projet pour en savoir plus sur l'utilisation de l'intégration Amazon Bedrock optimisée des services avec les machines d'Step Functionsétat, ou utilisez-le comme point de départ pour vos propres projets.

Rubriques

AWS CloudFormationModèle et ressources supplémentaires

Vous utilisez un CloudFormation modèle pour déployer cet exemple de projet. Ce modèle crée les ressources suivantes dans votre Compte AWS :

- Une machine Step Functions étatique.
- Rôle d'exécution pour la machine à états. Ce rôle accorde les autorisations dont votre machine d'état a besoin pour accéder à Services AWS d'autres ressources telles que l'Amazon BedrockInvokeModelaction.

Prérequis

Cet exemple de projet utilise le modèle de langage large (LLM) de la commande Cohere. Pour exécuter correctement cet exemple de projet, vous devez ajouter l'accès à ce LLM depuis la Amazon Bedrock console. Pour ajouter l'accès au modèle, procédez comme suit :

1. Ouvrez la console Amazon Bedrock.
2. Dans le volet de navigation, choisissez Model Access.
3. Choisissez Gérer l'accès aux modèles.
4. Cochez la case à côté de Cohere.
5. Choisissez Demander l'accès. L'état d'accès du modèle Cohere indique que l'accès est accordé.

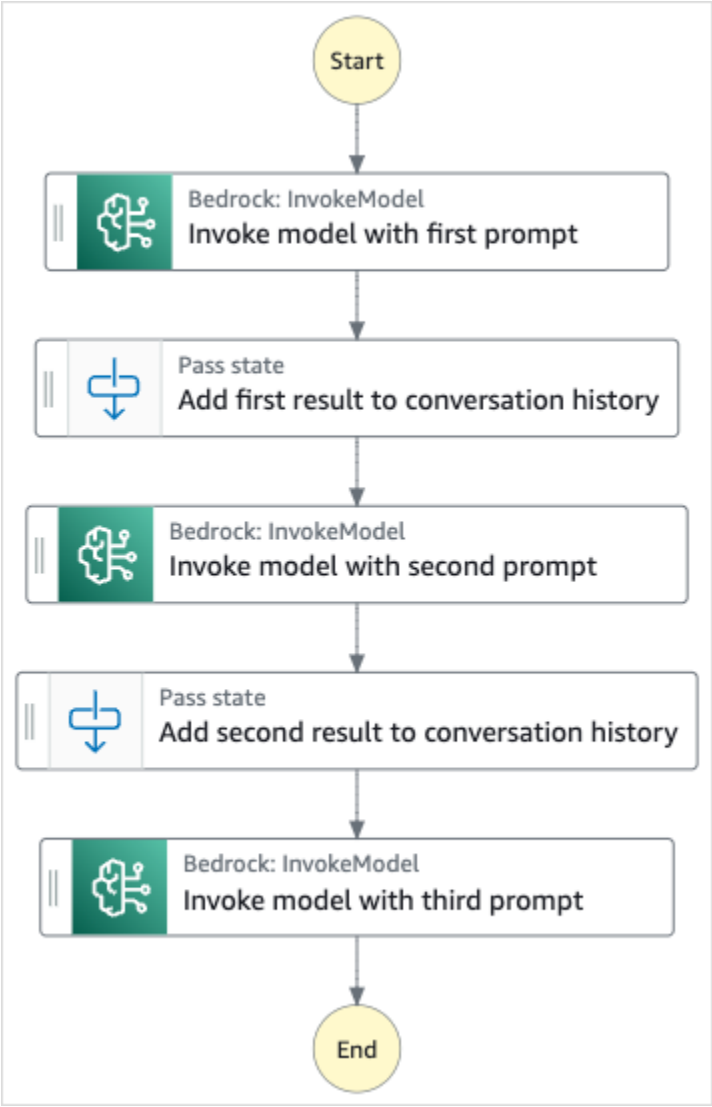
Étape 1 : créer la machine à états et provisionner les ressources

1. Ouvrez la console Step Functions et choisissez Create state machine.
2. Tapez **`bedrock`** dans le champ de recherche, puis choisissez Perform AI prompt-chaining with dans les résultats Bedrock de recherche renvoyés.
3. Choisissez Next (Suivant) pour continuer.
4. Step Functions répertorie les Services AWS éléments utilisés dans l'exemple de projet que vous avez sélectionné. Il montre également un graphique de flux de travail pour l'exemple de projet. Déployez ce projet sur votre site Compte AWS ou utilisez-le comme point de départ pour créer vos propres projets. Selon la façon dont vous souhaitez procéder, choisissez Exécuter une démo ou Construire à partir de celle-ci.

   Cet exemple de projet déploie les ressources suivantes :

   - Une machine AWS Step Functions étatique
   - Rôles associés AWS Identity and Access Management (IAM)

   L'image suivante montre le graphique du flux de travail pour le chaînage d'invites Perform AI avec Bedrock un exemple de projet :



5. Choisissez Utiliser le modèle pour poursuivre votre sélection.
6. Effectuez l'une des actions suivantes :

   - Si vous avez sélectionné Build on it, Step Functions crée le prototype de flux de travail pour l'exemple de projet que vous avez sélectionné. Step Functions ne déploie pas les ressources répertoriées dans la définition du flux de travail.

     Dans Workflow StudioMode de conception, glissez-déposez les états depuis le Navigateur d'états pour continuer à créer votre prototype de flux de travail. Vous pouvez également passer à un éditeur de code intégré similaire à VS Code pour mettre à jour la définition Amazon States Language (ASL) de votre machine à états dans la console Step Functions. Mode code Pour plus d'informations sur l'utilisation de Workflow Studio pour créer vos machines d'état, consultezUtilisation de Workflow Studio.

> **Important**
>
> N'oubliez pas de mettre à jour l'espace réservé Amazon Resource Name (ARN) pour les ressources utilisées dans l'exemple de projet avant d'exécuter votre flux de travail.

- Si vous avez sélectionné Run a demo, Step Functions crée un exemple de projet en lecture seule qui utilise un AWS CloudFormation modèle pour déployer les AWS ressources répertoriées dans ce modèle sur votre. Compte AWS

  > **Tip**
  >
  > Pour afficher la définition de la machine à états de l'exemple de projet, choisissez Code.

  Lorsque vous êtes prêt, choisissez Déployer et exécuter pour déployer l'exemple de projet et créer les ressources.

  La création de ces ressources et des autorisations IAM associées peut prendre jusqu'à 10 minutes. Pendant le déploiement de vos ressources, vous pouvez ouvrir le lien CloudFormation Stack ID pour voir quelles ressources sont mises en service.

  Une fois que toutes les ressources de l'exemple de projet ont été créées, vous pouvez voir le nouvel exemple de projet répertorié sur la page State machines.

  > **Important**
  >
  > Des frais standard peuvent s'appliquer pour chaque service utilisé dans le CloudFormation modèle.

Étape 2 : Exécuter la machine à états

1. Sur la page State machines, choisissez votre exemple de projet.
2. Sur la page d'exemple de projet, choisissez Démarrer l'exécution.
3. Dans la boîte de dialogue Démarrer l'exécution, procédez comme suit :

1. (Facultatif) Pour identifier votre exécution, vous pouvez lui donner un nom dans le champ Nom. Par défaut, Step Functions génère automatiquement un nom d'exécution unique.

   > **Note**
   >
   > Step Functions vous permet de créer des noms pour les machines d'état, les exécutions, les activités et les étiquettes contenant des caractères non ASCII. Ces noms non ASCII ne fonctionnent pas avec Amazon. CloudWatch Pour être sûr de pouvoir suivre CloudWatch les métriques, choisissez un nom qui utilise uniquement des caractères ASCII.

2. (Facultatif) Dans la zone de saisie, entrez les valeurs d'entrée au format JSON pour exécuter votre flux de travail.

   Si vous avez choisi d'exécuter une démo, vous n'avez pas besoin de fournir d'entrée d'exécution.

3. Choisissez Start execution (Démarrer l'exécution).

4. La console Step Functions vous dirige vers une page intitulée avec votre ID d'exécution. Cette page est connue sous le nom de page Détails de l'exécution. Sur cette page, vous pouvez consulter les résultats de l'exécution au fur et à mesure que l'exécution progresse ou une fois celle-ci terminée.

   Pour consulter les résultats de l'exécution, choisissez des états individuels dans la vue graphique, puis choisissez les onglets individuels du Détails de l'étape volet pour afficher les détails de chaque état, y compris les entrées, les sorties et la définition respectivement. Pour plus de détails sur les informations d'exécution que vous pouvez consulter sur la page Détails de l'exécution, voirPage de détails d'exécution — Vue d'ensemble de l'interface.

Quotas

AWS Step Functions impose des quotas sur la taille de certains paramètres de machine à états, tels que le nombre d'actions d'API pendant une certaine période ou le nombre de machines à états que vous pouvez définir. Bien que ces quotas soient conçus pour éviter qu'une machine d'état mal configurée utilise toutes les ressources du système, la plupart ne sont pas stricts.

Pour demander une augmentation du quota de service, vous pouvez effectuer l'une des opérations suivantes :

- Utilisez la console Service Quotas à l'adresse https://console.aws.amazon.com/servicequotas/home. Pour plus d'informations sur la demande d'augmentation de quota à l'aide de la console Service Quotas, voir Demande d'augmentation de quota dans le Guide de l'utilisateur de Service Quotas.
- Utilisez la page Support Center du AWS Management Console pour demander une augmentation du quota des ressources fournies par AWS Step Functions région. Pour plus d'informations, consultez la section Quotas du service AWS dans le Références générales AWS.

> **Note**
>
> Si une étape donnée de l'exécution de la machine d'état ou d'une activité est trop longue, vous pouvez configurer l'expiration de la machine d'état afin d'entraîner un événement d'expiration.

Rubriques

- Restrictions liées au balisage

Quotas généraux

Quota	Description
Noms dans Step Functions	Les noms des machines d'état, des exécutions et des tâches d'activité ne doivent pas dépasser 80 caractères. Ces noms doivent être uniques pour votre compte et votre AWS région, et ne doivent contenir aucun des éléments suivants : • Espace blanc • Caractères génériques () ? * • Caractères entre crochets (< > { } [ ]) • Caractères spéciaux (" # % \ ^ | ~ ` $ & , ; : /) • Caractères de contrôle (\\u0000- \\u001f ou \\u007f -\\u009f). Si votre machine à états est de type Express, vous pouvez attribuer le même nom à plusieurs exécutions de la machine à états. Step Functions génère un ARN d'exécution unique pour chaque exécution automatique d'Express State, même si plusieurs exécutions portent le même nom. Step Functions vous permet de créer des noms pour les machines d'état, les exécutions et les activités, ainsi que des étiquettes contenant des caractères non ASCII. Ces noms non ASCII ne fonctionnent pas avec Amazon. CloudWatch Pour être sûr de pouvoir suivre

Quota	Description
	CloudWatch les métriques, choisissez un nom qui utilise uniquement des caractères ASCII.

Quotas liés aux comptes

Ressource	Quota par défaut	Peut être augmenté jusqu'à
Nombre maximal de machines d'état enregistrées	10 000	25 000
Nombre maximal d'activités enregistrées	10 000	15 000
Taille maximum d'une requête	1 Mo par requête. Il s'agit de la taille totale des données par demande d'API Step Functions, y compris l'en-tête de la demande et toutes les autres données de demande associées.	Quota strict
Nombre maximal d'exécutions ouvertes par compte	1 000 000 d'exécutions pour chacune Compte AWS d'entre elles Région AWS. Un dépassement provoquer a une erreur `Execution LimitExceeded` . Cela ne s'applique pas aux flux de travail express.	Des millions
Nombre maximum d'exécutions de carte ouvertes Une exécution de carte ouverte est une exécution de	1 000 Ce quota s'applique à l'état de la carte distribuée.	Quota strict

Ressource	Quota par défaut	Peut être augmenté jusqu'à
carte qui a commencé, mais qui n'est pas encore terminée. Les courses de carte planifiées attendent lors de l'MapRunStartedévénement que le nombre total de courses de carte ouvertes soit inférieur au quota par défaut de 1 000.		
Durée maximale redrivesd'une course de carte.	1 000 Ce quota s'applique à l'état de la carte distribuée.	Quota strict
Nombre maximum d'exécutions parallèles d'enfants selon Map Run	10 000	Quota strict

Quotas liés à la tâche HTTP

Les tâches HTTP sont limitées à l'aide d'un schéma de bucket à jetons afin de maintenir la bande passante du Step Functions service.

Ressource	Taille de compartiment	Taux de remplissage par seconde
Tâche HTTP	300	300

Le tableau suivant répertorie le quota pour la durée d'une tâche HTTP.

Ressource	Quota par défaut
Durée de la tâche HTTP	60 secondes

Ressource	Quota par défaut
La durée d'une tâche HTTP fait référence au temps nécessaire à une tâche HTTP pour envoyer une requête HTTP et recevoir une réponse.	Il s'agit d'un quota strict qui ne peut pas être modifié.

Quotas liés à l'étranglement de l'État

Les transitions d'état de Step Functions sont limitées à l'aide d'un schéma de bucket à jetons afin de maintenir la bande passante du service. Les flux de travail standard et les flux de travail express ont une régulation de transition d'état différente. Les quotas des flux de travail standard sont des quotas souples et peuvent être augmentés.

> **Note**
>
> La limitation de l'indicateur de `StateTransition` service est signalée comme sur `ExecutionThrottled` Amazon. CloudWatch Pour plus d'informations, consultez la `ExecutionThrottled` CloudWatch métrique.

	Standard		Express	
Métrique du service	Taille de compartiment	Taux de remplissage par seconde	Taille de compartiment	Taux de remplissage par seconde
`StateTransition` — Dans l'est des États-Unis (Virginie du Nord), dans l'ouest des États-Unis (Oregon)	5 000	5 000	Illimité	Illimité

	Standard		Express	
Métrique du service	Taille de compartiment	Taux de remplissage par seconde	Taille de compartiment	Taux de remplissage par seconde
et en Europe (Irlande)				
`StateTransition` — Toutes les autres régions	800	800	Illimité	Illimité

Quotas liés à la limitation des actions des API

Certaines actions de l'API Step Functions sont limitées à l'aide d'un schéma de bucket à jetons afin de maintenir la bande passante du service. Ces quotas sont des quotas souples qui peuvent être augmentés.

> **Note**
>
> Les quotas de limitation sont établis par compte et par région. AWS
> AWS Step Functions peut augmenter à la fois la taille du seau et le taux de recharge à tout moment.

	Standard		Express	
Nom d'API	Taille de compartiment	Taux de remplissage par seconde	Taille de compartiment	Taux de remplissage par seconde
`StartExecution` — Dans l'est des États-Unis (Virginie	1 300	300	6 000	6 000

	Standard		Express	
Nom d'API	Taille de compartiment	Taux de remplissage par seconde	Taille de compartiment	Taux de remplissage par seconde
du Nord), dans l'ouest des États-Unis (Oregon) et en Europe (Irlande)				
`StartExecution` — Toutes les autres régions	800	150	6 000	6 000

Quota lié à l' TestState API

Nom d'API	Quota	Peut être augmenté jusqu'à
TestState	1 transaction par seconde (TPS)	Quota strict

Autres quotas

Ces quotas sont des quotas souples qui peuvent être augmentés.

	In US East (N. Virginia), US West (Oregon), and Europe (Ireland)		All other regions	
Nom d'API	Taille de compartiment	Taux de remplissage par seconde	Taille de compartiment	Taux de remplissage par seconde
`CreateActivity`	100	1	100	1
`CreateStateMachine`	100	1	100	1
`DeleteActivity`	100	1	100	1
`DeleteStateMachine`	100	1	100	1
`DescribeActivity`	200	1	200	1
`DescribeExecution`	300	15	250	10
`DescribeStateMachine`	200	20	200	20
`DescribeStateMachineForExecution`	200	1	200	1
`GetActivityTask`	3 000	500	1 500	300
`GetExecutionHistory`	400	20	400	20

	In US East (N. Virginia), US West (Oregon), and Europe (Ireland)		All other regions	
Nom d'API	Taille de compartiment	Taux de remplissage par seconde	Taille de compartiment	Taux de remplissage par seconde
`ListActivities`	100	10	100	5
`ListExecutions`	200	5	100	2
`ListStateMachines`	100	5	100	5
`ListTagsForResource`	100	1	100	1
`SendTaskFailure`	3 000	500	1 500	300
`SendTaskHeartbeat`	3 000	500	1 500	300
`SendTaskSuccess`	3 000	500	1 500	300
StartSync Execution	Les appels d'API d'exécution synchrone d'Express ne contribuent pas aux limites de capacité existantes du compte. Step Functions fournit des capacités à la demande et s'adapte automatiquement à une charge de travail soutenue. Les pics de charge de travail peuvent être limités jusqu'à ce que la capacité soit disponible. Si vous constatez un ralentissement, réessayez après un certain temps. Pour plus d'informations sur les flux de travail Synchrous Express, consultezFlux de travail express synchrones et asynchrones.			

	In US East (N. Virginia), US West (Oregon), and Europe (Ireland)		All other regions	
Nom d'API	Taille de compartiment	Taux de remplissage par seconde	Taille de compartiment	Taux de remplissage par seconde
`StopExecution`	1 000	200	500	25
`TagResource`	200	1	200	1
`UntagResource`	200	1	200	1
`UpdateStateMachine`	100	1	100	1

Quotas liés aux exécutions par les machines de l'État

Le tableau suivant décrit les quotas liés aux exécutions par des machines d'État. Les quotas d'exécution des machines à états sont des quotas stricts qui ne peuvent pas être modifiés, à l'exception du quota de durée de conservation de l'historique d'exécution.

Quota	Standard	Express
Durée d'exécution maximum	1 an. Si une exécution dure plus d'un an, elle échouera avec une `States.Timeout` erreur et émettra une `ExecutionsTimedOut` CloudWatch métrique.	5 minutes. Si une exécution dure plus de 5 minutes, elle échouera avec une `States.Timeout` erreur et émettra une `ExecutionsTimedOut` CloudWatch métrique.
Taille maximum de l'historique d'exécution	25 000 événements dans un historique d'exécution automatique à état unique.	Illimité.

Quota	Standard	Express
	Si l'historique des exécutions atteint cette limite, l'exécution échoue. Pour éviter ce problème, consultez Évitez d'atteindre le quota d'historique.	
Temps inactif d'exécution maximum	1 an (limité par le délai d'exécution maximal).	5 minutes (limité par le temps d'exécution maximal).
Durée de conservation de l'historique d'exécution	90 jours après la clôture de l'exécution. Passé ce délai, vous ne pouvez plus récupérer ou afficher l'historique d'exécution. Il n'y a pas de quota supplémentaire pour le nombre d'exécutions fermées que Step Functions conserve. Pour répondre aux exigences de conformité, organisationnelles ou réglementaires, vous pouvez réduire la période de conservation de l'historique d'exécution à 30 jours en envoyant une demande de quota. Pour ce faire, utilisez le AWS Support Center Console et créez un nouveau boîtier. La modification visant à réduire la période de conservation à 30 jours s'applique à chaque compte d'une région.	Pour consulter l'historique des exécutions, la journalisation Amazon CloudWatch Logs doit être configurée. Pour plus d'informations, consultez Journalisation à l'aideCloudWatchJournaux.

Quota	Standard	Express
redrivablePériode d'exécution RedrivableLa période fait référence au temps pendant lequel vous pouvez exécuter redriveun flux de travail standard donné. Cette période commence le jour où une machine d'État termine son exécution.	14 jours. Ce quota strict s'applique à l'état de la carte distribuée.	Redriven'est actuellement pas pris en charge pour les flux de travail Express.

Quotas liés à l'exécution des tâches

Le tableau suivant décrit les quotas relatifs aux exécutions de tâches. Ce sont tous des quotas stricts qui ne peuvent pas être modifiés.

Quota	Standard	Express
Durée maximum de l'exécution de tâche	1 an (limité par le délai maximal d'exécution)	5 minutes (contraintes par le temps d'exécution maximal)
Durée maximale pendant laquelle Step Functions conserve une tâche dans la file d'attente	1 an (limité par le délai maximal d'exécution)	5 minutes (contraintes par le temps d'exécution maximal)
Nombre maximal de sondeurs d'activité par Amazon Resource Name (ARN)	1 000 observateurs appelant `GetActivityTask` par ARN. Si cette limite est dépassée, une erreur est générée : « Le nombre maximal de programmes exécutant simultanément des	Ne s'applique pas aux workflows express.

Quota	Standard	Express
	tâches d'observation d'activit és a été atteint. »	
Taille d'entrée ou de sortie maximale pour une tâche, un état ou une exécution	256 Ko de données sous forme de chaîne codée en UTF-8. Ce quota affecte les tâches (activité, fonction Lambda ou service intégré), les résultats d'état ou d'exécuti on et les données d'entrée lors de la planification d'une tâche, de la saisie d'un état ou du démarrage d'une exécution.	256 Ko de données sous forme de chaîne codée en UTF-8. Ce quota affecte les tâches (activité, fonction Lambda ou service intégré), les résultats d'état ou d'exécuti on et les données d'entrée lors de la planification d'une tâche, de la saisie d'un état ou du démarrage d'une exécution.

Quotas liés aux versions et aux alias

Ressource	Quota par défaut
Nombre maximum de versions publiées de machines à états	1000 pour chaque machine à états. Pour demander une augmentation de cette limite souple, utilisez la page Support Center du AWS Management Console.
Nombre maximum d'alias de machine à états	100 pour chaque machine à états. Pour demander une augmentation de cette limite souple, utilisez la page Support Center du AWS Management Console.

Restrictions liées au balisage

Tenez compte de ces restrictions lorsque vous balisez des ressources Step Functions.

> **Note**
>
> Les restrictions de balises ne peuvent pas être augmentées comme les autres quotas.

Restriction	Description
Nombre maximal de balises par ressource	50
Longueur maximale de clé	128 caractères Unicode en UTF-8
Longueur maximale de valeur	256 caractères Unicode en UTF-8
Restriction de préfixe	N'utilisez pas le `aws:` préfixe dans les noms ou les valeurs de vos balises, car il est réservé à AWS l'usage. Vous ne pouvez pas modifier ou supprimer des noms ou valeurs de balise ayant ce préfixe. Les balises avec ce préfixe ne sont pas prises en comptes dans vos balises pour le quota de ressources.
Restrictions de caractères	Les balises doivent contenir uniquement des lettres Unicode, des chiffres, des espaces ou les symboles : `_ . : / = + - @`

Connexion et surveillance AWS Step Functions

La journalisation et la surveillance sont importantes pour garantir la fiabilité, la disponibilité et les performances de Step Functions et de vos AWS solutions. Plusieurs outils peuvent être utilisés avec Step Functions :

> **Tip**
>
> Pour déployer un exemple de flux de travail sur votre Compte AWS ordinateur et apprendre à surveiller les métriques, les journaux et les traces de l'exécution du flux de travail, consultez le module 12 - Observabilité de l' AWS Step Functions atelier.

Rubriques

Surveillance des fonctions Step Functions à l'aide de CloudWatch

La surveillance joue un rôle important dans le maintien de la fiabilité, de la disponibilité AWS Step Functions et des performances de vos AWS solutions. Vous devez collecter autant de données de surveillance auprès des AWS services que vous utilisez afin de pouvoir corriger les défaillances multipoints. Avant de commencer à surveiller Step Functions, vous devez créer un plan de surveillance répondant aux questions suivantes :

- Quels sont les objectifs de la surveillance ?
- Quelles sont les ressources à surveiller ?
- À quelle fréquence les ressources doivent-elles être surveillées ?

- Quels outils de surveillance utiliser ?
- Qui exécute les tâches de supervision ?
- Qui doit être informé en cas de problème ?

La prochaine étape consiste à établir une base pour les performances normales de dans votre environnement. Pour cela, vous devez mesurer les performances à différents moments et sous différentes conditions de charge. Lorsque vous surveillez Step Functions, pensez à stocker des données de surveillance historiques. Ces données peuvent constituer une référence à comparer avec des données de performances actuelles, afin d'identifier les modèles de performance normaux et les anomalies de performance, et de concevoir des solutions pour résoudre les problèmes.

Par exemple, avec Step Functions, vous pouvez surveiller le nombre d'activités ou de AWS Lambda tâches qui échouent en raison d'un arrêt du rythme cardiaque. Lorsque les performances se trouveront en dehors de la référence établie, vous devrez peut-être modifier votre intervalle de pulsations.

Pour établir une référence, vous devez au moins surveiller les métriques suivantes :

- `ActivitiesStarted`
- `ActivitiesTimedOut`
- `ExecutionsStarted`
- `ExecutionsTimedOut`
- `LambdaFunctionsStarted`
- `LambdaFunctionsTimedOut`

Les sections suivantes décrivent les métriques fournies par Step Functions à Amazon CloudWatch. Vous pouvez utiliser ces métriques pour suivre les machines d'état et les activités, et définir des alarmes sur les valeurs seuils. Vous pouvez consulter les statistiques à l'aide du AWS Management Console.

Indicateurs indiquant un intervalle de temps

Certaines des CloudWatch métriques de Step Functions sont des intervalles de temps, toujours mesurés en millisecondes. Ces métriques correspondent généralement aux étapes de votre exécution pour lesquelles vous pouvez définir les délais d'expiration de la machine à états, de l'activité et des fonctions Lambda, avec des noms descriptifs.

Par exemple, la métrique `ActivityRunTime` détermine le temps nécessaire pour qu'une activité soit terminée après le début de son exécution. Vous pouvez définir une valeur de délai d'attente pour la même période de temps.

Dans la CloudWatch console, vous pouvez obtenir les meilleurs résultats si vous choisissez la moyenne comme statistique d'affichage pour les mesures d'intervalle de temps.

Indicateurs indiquant un décompte

Certaines CloudWatch métriques de Step Functions indiquent les résultats sous forme de décompte. Par exemple, `ExecutionsFailed` enregistre le nombre d'exécutions de machine d'état ayant échoué.

Step Functions émet deux `ExecutionsStarted` métriques pour chaque exécution de State Machine. Cela fait que la SampleCountstatistique de la `ExecutionsStarted` métrique affiche la valeur 2 pour chaque exécution de la machine à états. La SampleCount statistique indique `ExecutionStarted=1` et `ExecutionStarted=0` quand l'exécution est terminée.

> **Tip**
>
> Nous vous recommandons de sélectionner Sum comme statistique d'affichage pour les métriques qui indiquent un nombre dans la CloudWatch console.

Métriques d'exécution

L'espace de `AWS/States` noms inclut les métriques suivantes pour toutes les exécutions de Step Functions. Il s'agit de statistiques sans dimension qui s'appliquent à l'ensemble de votre compte dans une région.

Métrique	Description
`OpenExecutionCount`	Nombre approximatif d'exécutions actuellement ouvertes : flux de travail actuellement en cours dans votre compte. L'objectif est de fournir un aperçu du moment où vos flux de travail approchent de la limite d'exécution maximale, afin d'éviter les ExecutionLimitExceedederreurs lors des appels `StartExec`

Métrique	Description
	`ution` ou `RedriveExecution` pour les flux de travail standard. La métrique dépend des transitions d'état du flux de travail actif. Ainsi, à de faibles niveaux, l'estimation peut ne pas correspon dre au nombre de flux de travail en cours observé.
`OpenExecutionLimit`	Nombre maximum d'exécutions ouvertes. Pour plus d'informa tions, consultez Quotas liés aux comptes. Cette limite ne s'applique pas aux flux de travail express.

Métriques d'exécution pour la machine à états avec version ou alias

Lorsque vous exécutez une exécution State Machine avec une version ou un alias, Step Functions émet les métriques suivantes. La `ExecutionThrottled` métrique ne sera émise qu'en cas d'exécution limitée. Ces métriques incluront un `StateMachineArn` pour identifier une machine à états spécifique.

Métrique	Description
`ExecutionTime`	Intervalle, en millisecondes, entre le début de l'exécution et le moment où elle se termine.
`ExecutionThrottled`	Nombre d'`StateEntered` événements et de nouvelles tentatives qui ont été limités. Ceci est lié aux limites `StateTran sition` . Pour plus d'informations, consultez Quotas liés à l'étranglement de l'État.
`ExecutionsAborted`	Nombre d'exécutions abandonnées ou interrompues.
`ExecutionsFailed`	Nombre d'exécutions ratées.
`ExecutionsStarted`	Nombre d'exécutions entamées.
`ExecutionsSucceeded`	Nombre d'exécutions terminées avec succès.

Métrique	Description
`ExecutionsTimedOut`	Nombre d'exécutions dont le délai est expiré pour une raison ou une autre.

Mesures d'exécution pour Express Workflows

L'espace de `AWS/States` noms inclut les métriques suivantes pour les exécutions de Step Functions Express Workflows.

Métrique	Description
`ExpressExecutionMemory`	Mémoire totale consommée par un flux de travail express.
`ExpressExecutionBilledDuration`	Durée pour laquelle un flux de travail express est facturé.
`ExpressExecutionBilledMemory`	Quantité de mémoire consommée pour laquelle un flux de travail express est facturé.

Redrivemétriques d'exécution pour les flux de travail standard

Lorsque vous exécutez redriveune machine à états, Step Functions émet les métriques suivantes.

Pour toutes les redriven exécutions, la `Executions*` métrique est émise. Supposons, par exemple, qu'une redriven exécution soit abandonnée. Cette exécution émettra des points de données non nuls pour les deux et. `RedrivenExecutionsAborted` `ExecutionsAborted`

Métrique	Description
`ExecutionsRedriven`	Nombre d'redrivenexécutions.
`RedrivenExecutionsAborted`	Nombre d'redrivenexécutions annulées ou terminées.

Métrique	Description
`RedrivenExecutionsTimedOut`	Nombre d'redrivenexécutions dont le délai est expiré pour une raison ou une autre.
`RedrivenExecutionsSucceeded`	Nombre d'redrivenexécutions réussies.
`RedrivenExecutionsFailed`	Nombre d'redrivenexécutions qui ont échoué.

Mesures d'exécution de Dimension for Step Functions

Dimension	Description
`StateMachineArn`	L'Amazon Resource Name (ARN) de la machine d'état pour l'exécution en question.

Dimensions pour les exécutions avec version

Dimension	Description
`StateMachineArn`	Le nom de ressource Amazon (ARN) de la machine d'état dont l'exécution a été lancée par une version.
`Version`	Version de la machine à états utilisée pour démarrer l'exécution.

Dimensions pour les exécutions avec un alias

Dimension	Description
`StateMachineArn`	Le nom de ressource Amazon (ARN) de la machine d'état dont l'exécution a été lancée par un alias.
`Alias`	Alias de machine à états utilisé pour démarrer l'exécution.

Mesures relatives au nombre de ressources pour les versions et les alias

L'espace de `AWS/States` noms inclut les métriques suivantes pour le nombre de versions et d'alias d'une machine à états.

Métrique	Description
AliasCount	Nombre d'alias créés pour la machine à états. Vous pouvez créer jusqu'à 100 alias pour chaque machine à états.
VersionCount	Nombre de versions publiées pour la machine d'état. Vous pouvez publier jusqu'à 1 000 versions d'une machine à états.

Dimension pour les mesures relatives au nombre de ressources pour les versions et les alias

Dimension	Description
`ResourceArn`	Le nom de ressource Amazon (ARN) de la machine à états avec une version ou un alias.

Métriques d'activité

L'espace de `AWS/States` noms inclut les métriques suivantes pour les activités de Step Functions.

Métrique	Description
`ActivityRunTime`	Intervalle, en millisecondes, entre le début de l'activité et le moment où elle se termine.
`ActivityScheduleTime`	Intervalle, en millisecondes, pendant lequel l'activité reste dans l'état planifié.

Métrique	Description
`ActivityTime`	Intervalle, en millisecondes, entre le moment où l'activité est planifiée et celui où elle se termine.
`ActivitiesFailed`	Nombre d'activités ayant échoué.
`ActivitiesHeartbeatTimedOut`	Nombre d'activités qui s'interrompent en raison d'un arrêt du rythme cardiaque.
`ActivitiesScheduled`	Nombre d'activités programmées.
`ActivitiesStarted`	Nombre d'activités démarrées.
`ActivitiesSucceeded`	Nombre d'activités achevées avec succès.
`ActivitiesTimedOut`	Nombre d'activités qui expirent à la fermeture.

Mesures d'activité de Dimension for Step Functions

Dimension	Description
`ActivityArn`	ARN de l'activité.

Métriques de la fonction Lambda

L'espace de `AWS/States` noms inclut les métriques suivantes pour les fonctions Lambda Step Functions.

Métrique	Description
`LambdaFunctionRunTime`	Intervalle, en millisecondes, entre le moment où la fonction Lambda démarre et celui où elle se ferme.
`LambdaFunctionScheduleTime`	Intervalle, en millisecondes, pendant lequel la fonction Lambda reste dans l'état planifié.

Métrique	Description
`LambdaFunctionTime`	Intervalle, en millisecondes, entre le moment où la fonction Lambda est planifiée et le moment où elle se ferme.
`LambdaFunctionsFailed`	Nombre de fonctions Lambda ayant échoué.
`LambdaFunctionsScheduled`	Nombre de fonctions Lambda planifiées.
`LambdaFunctionsStarted`	Nombre de fonctions Lambda démarrées.
`LambdaFunctionsSucceeded`	Nombre de fonctions Lambda terminées avec succès.
`LambdaFunctionsTimedOut`	Nombre de fonctions Lambda qui expirent à la fermeture.

Dimension for Step Functions > Métriques de la fonction Lambda

Dimension	Description
`LambdaFunctionArn`	L'ARN de la fonction Lambda.

> **Note**
>
> Les métriques de fonction Lambda sont émises pour les états de tâche qui spécifient l'ARN de la fonction Lambda dans le champ. `Resource` États de tâches qui utilisent plutôt `"Resource": "arn:aws:states:::lambda:invoke"` des métriques d'intégration de services d'émission. Pour plus d'informations, consultez Invoquez Lambda avec Step Functions.

Métriques d'intégration de services

L'espace de `AWS/States` noms inclut les métriques suivantes pour les intégrations de services Step Functions. Pour plus d'informations, consultez Utilisation AWS Step Functions avec d'autres services.

Métrique	Description
`ServiceIntegration RunTime`	Intervalle, en millisecondes, entre le début de la tâche de service et le moment où elle se termine.
`ServiceIntegration ScheduleTime`	Intervalle, en millisecondes, pendant lequel la tâche de service reste dans l'état planifié.
`ServiceIntegration Time`	Intervalle, en millisecondes, entre le moment où la tâche de service est planifiée et celui où elle se termine.
`ServiceIntegration sFailed`	Nombre de tâches de service ayant échoué.
`ServiceIntegration sScheduled`	Nombre de tâches de service planifiées.
`ServiceIntegration sStarted`	Nombre de tâches de service démarrées.
`ServiceIntegration sSucceeded`	Nombre de tâches de service achevées avec succès.
`ServiceIntegration sTimedOut`	Nombre de tâches de service qui expirent à la clôture.

Mesures d'intégration du service Dimension for Step Functions

Dimension	Description
`ServiceIntegration ResourceArn`	ARN de ressource du service intégré.

Métriques de service

L'espace de `AWS/States` noms inclut les métriques suivantes pour le service Step Functions.

Métrique	Description
`ThrottledEvents`	Nombre de demandes qui ont été limitées.
`ProvisionedBucketSize`	Nombre de demandes disponibles par seconde.
`ProvisionedRefillRate`	Nombre de demandes autorisées dans le compartiment par seconde.
`ConsumedCapacity`	Nombre de demandes par seconde.

Mesures du service Dimension for Step Functions

Dimension	Description
`ServiceMetric`	Données de filtres pour afficher les métriques de transitions d'état.

Métriques API

L'espace de `AWS/States` noms inclut les métriques suivantes pour l'API Step Functions.

Métrique	Description
`ThrottledEvents`	Nombre de demandes qui ont été limitées.
`ProvisionedBucketSize`	Nombre de demandes disponibles par seconde.
`ProvisionedRefillRate`	Nombre de demandes autorisées dans le compartiment par seconde.

Métrique	Description
`ConsumedCapacity`	Nombre de demandes par seconde.

Mesures de l'API Dimension for Step Functions

Dimension	Description
`APIName`	Filtre les données sur une API correspondant au nom d'API spécifié.

Livraison des CloudWatch indicateurs les plus efficaces

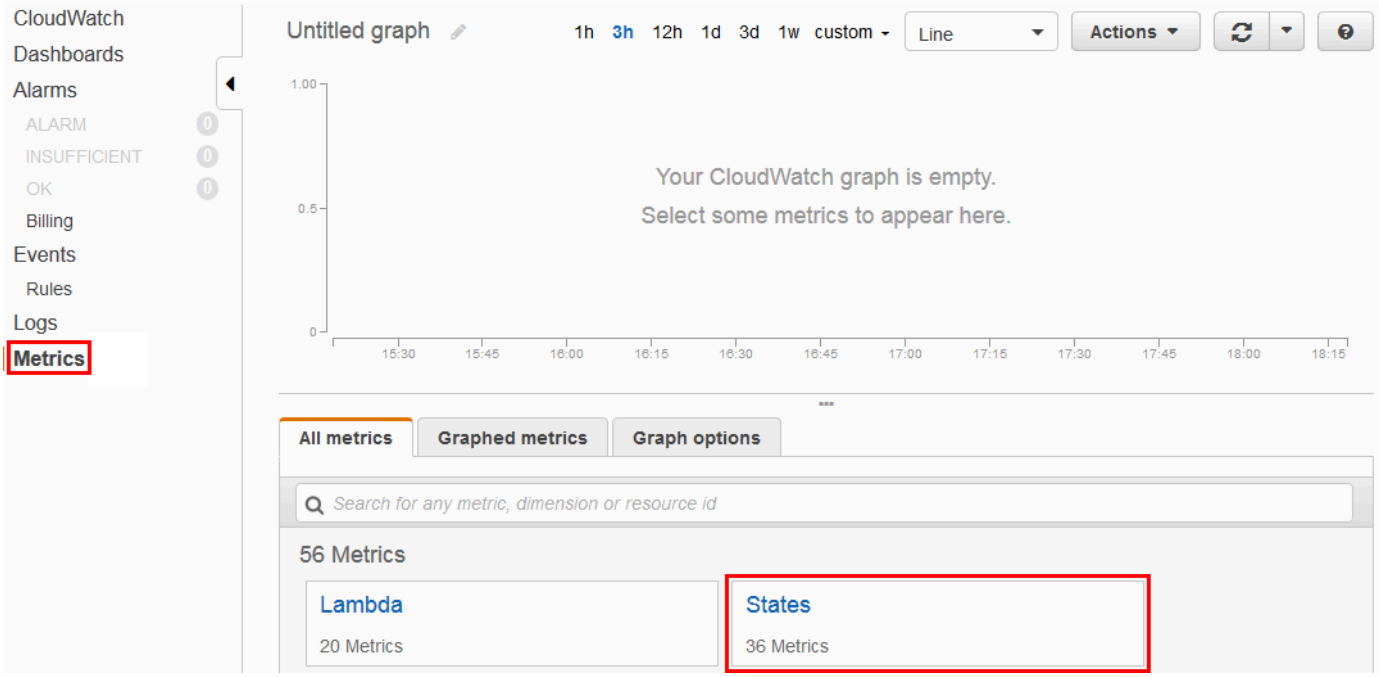
Les métriques CloudWatch sont fournies sur la base du meilleur effort.

L'exhaustivité et la ponctualité des métriques ne sont pas garanties. Le point de données d'une demande particulière doit être retourné avec un horodatage ultérieur au moment du traitement de la demande. Le point de données peut être retardé d'une minute avant d'être disponible CloudWatch ou il peut ne pas être livré du tout. CloudWatchles métriques de demande vous donnent une idée des exécutions par State Machine en temps quasi réel. Il ne s'agit pas d'un compte rendu complet de tous les indicateurs liés à l'exécution.

Compte tenu de la nature optimale de cette fonctionnalité, les rapports disponibles sur le tableau de bord de gestion de la facturation et des coûts peuvent inclure une ou plusieurs demandes d'accès qui n'apparaissent pas dans les statistiques d'exécution.

Afficher les métriques pour Step Functions

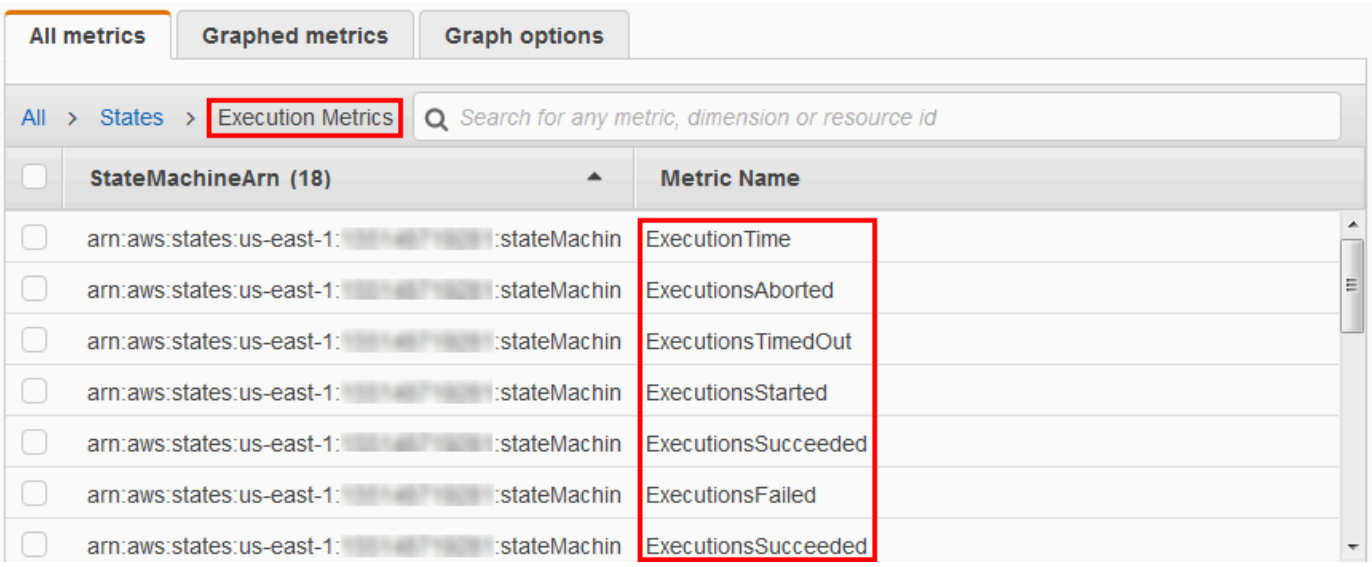
1. Connectez-vous à la CloudWatch console AWS Management Console et ouvrez-la.
2. Choisissez Metrics (Métriques), puis sous l'onglet All Metrics (Toutes les métriques), choisissez States (États).



Si vous avez effectué des exécutions récemment, vous verrez jusqu'à quatre types de métriques :

- Execution Metrics (Métriques de l'exécution)
- Métriques de la fonction de l'activité
- Métriques de la fonction Lambda
- Métriques d'intégration des services

3. Choisissez un type de métrique pour afficher une liste de métriques.

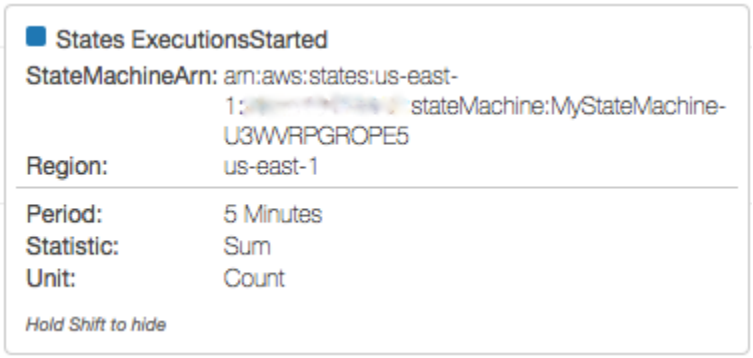


- Pour trier vos statistiques par nom de métrique ou utilisez StateMachineArnles en-têtes de colonne.
- Pour afficher les graphiques d'une métrique, cochez la case en regard de la métrique sur la liste. Vous pouvez modifier les paramètres du graphique à l'aide des contrôles des plages de temps au-dessus de l'affichage du graphique.

  Vous pouvez choisir des plages de temps personnalisées à l'aide de valeurs relatives ou absolues (jours et heures spécifiques). Vous pouvez également utiliser la liste déroulante pour afficher les valeurs sous forme de lignes, de zones empilées ou de chiffres (valeurs).
- Pour afficher les détails relatifs à un graphique, survolez le code de couleur de la métrique qui apparaît sous le graphique.

  Les détails de la métrique s'affichent.

Pour plus d'informations sur l'utilisation des CloudWatch métriques, consultez la section Utilisation d'Amazon CloudWatch Metrics dans le guide de CloudWatch l'utilisateur Amazon.

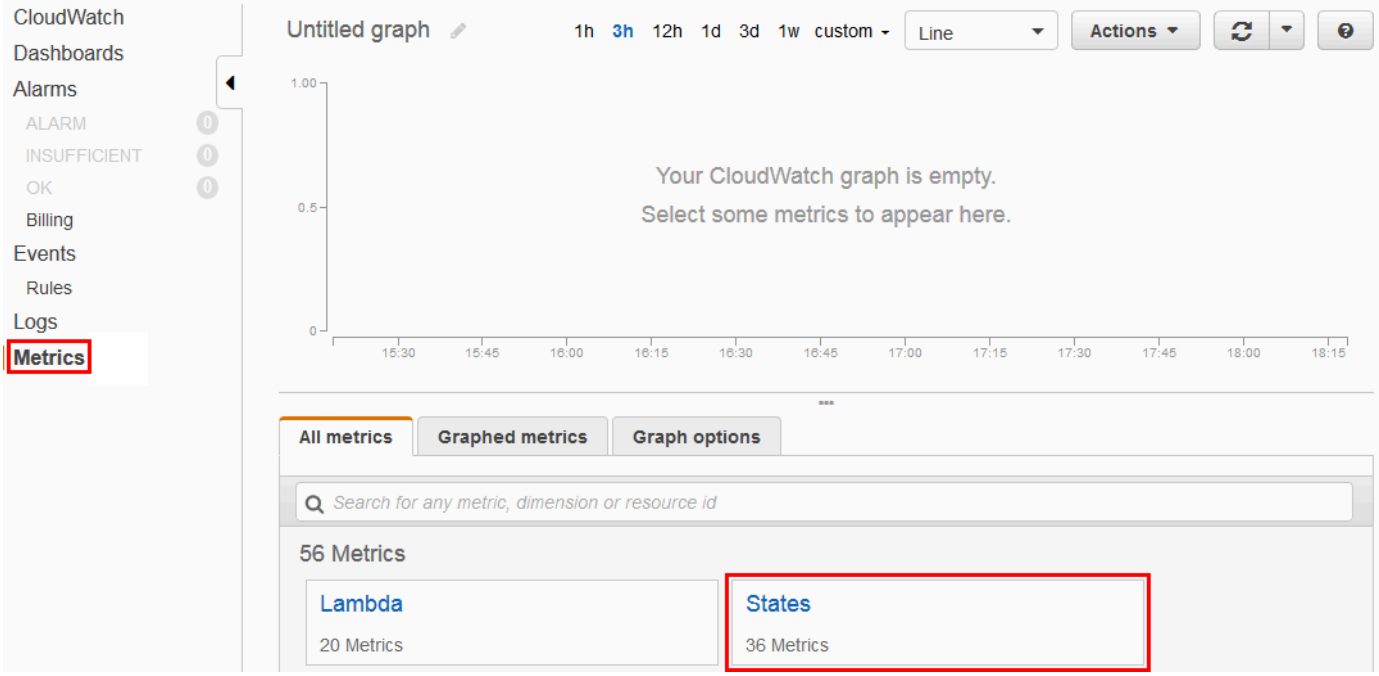
Configuration des alarmes pour Step Functions

Vous pouvez utiliser les CloudWatch alarmes Amazon pour effectuer des actions. Par exemple, si vous souhaitez savoir quand un seuil d'alarme est atteint, vous pouvez configurer une alarme pour envoyer une notification à une rubrique Amazon SNS ou pour envoyer un e-mail lorsque la `StateMachinesFailed` métrique dépasse un certain seuil.

Pour configurer une alarme sur une métrique

1. Connectez-vous à la CloudWatch console AWS Management Console et ouvrez-la.

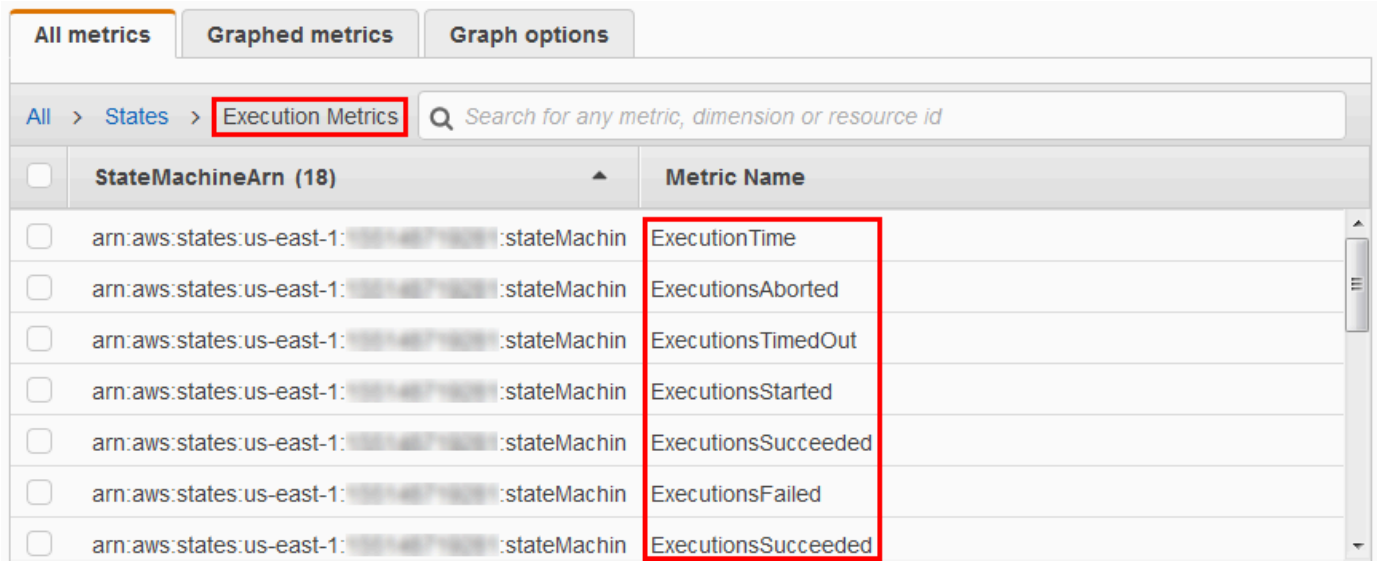
2. Choisissez Metrics (Métriques), puis sous l'onglet All Metrics (Toutes les métriques), choisissez States (États).



Si vous avez effectué des exécutions récemment, vous verrez jusqu'à quatre types de métriques :

- Execution Metrics (Métriques de l'exécution)
- Métriques de la fonction de l'activité
- Métriques de la fonction Lambda
- Métriques d'intégration des services

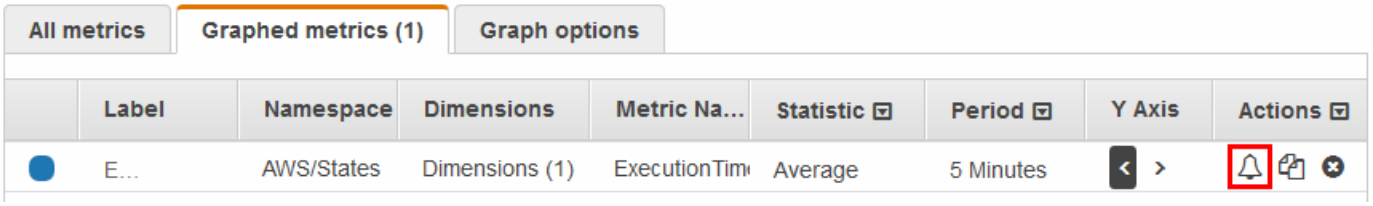
3. Choisissez un type de métrique pour afficher une liste de métriques.



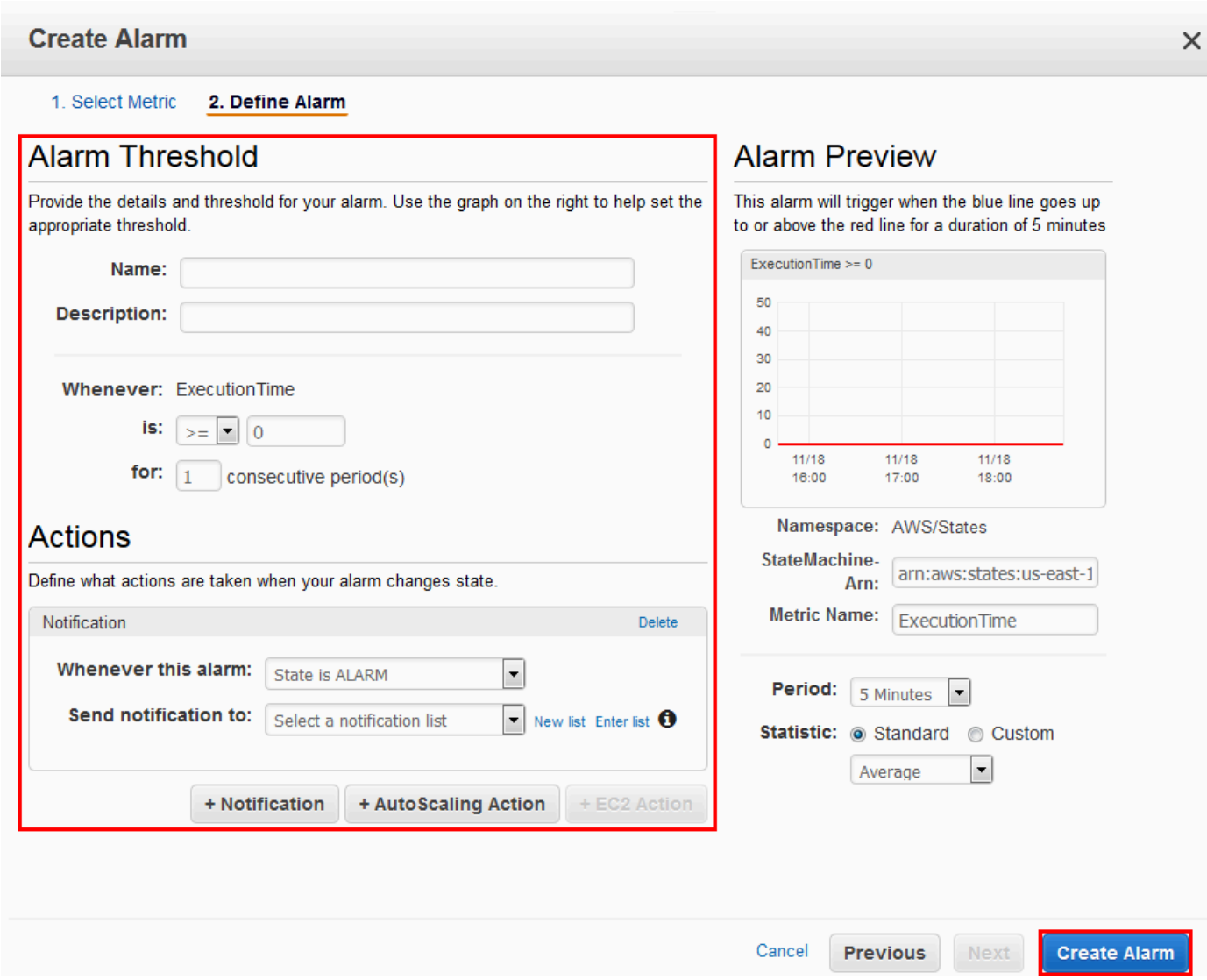
4. Choisissez une métrique, puis choisissez Graphed metrics (Graphique des métriques).
5. Choisissez

   en regard d'une métrique sur la liste.



   La page Create Alarm (Créer une alarme) s'affiche.



6. Entrez les valeurs des champs Alarm threshold (Seuil d'alarme) et Actions, puis choisissez Créer une alarme.

Pour plus d'informations sur le paramétrage et l'utilisation des CloudWatch alarmes, consultez la section Création d' CloudWatch alarmes Amazon dans le guide de CloudWatch l'utilisateur Amazon.

EventBridge (CloudWatch Evénements) pour les changements de statut d'exécution de Step Functions

Amazon EventBridge est un AWS service qui vous permet de réagir aux changements d'état d'une AWS ressource. Par exemple, vous pouvez réagir aux changements de statut d'exécution d'un flux de travail standard Step Functions en EventBridge utilisant les deux méthodes suivantes :

- Vous pouvez configurer EventBridge des règles pour réagir aux événements émis lorsque l'état d'exécution d'une machine d'état Step Functions change. Cela vous permet de surveiller votre flux de travail sans avoir à interroger constamment à l'aide de l'API DescribeExecution. En fonction de l'évolution des exécutions de machines à états, vous pouvez utiliser une EventBridge cible pour démarrer de nouvelles exécutions de machines à états, appeler des AWS Lambda fonctions, publier des messages sur les rubriques Amazon Simple Notification Service (Amazon SNS), etc.
- Vous pouvez également configurer une machine d'état Step Functions en tant que cible dans EventBridge. Cela vous permet de déclencher l'exécution d'un flux de travail Step Functions en réponse à un événement provenant d'un autre AWS service.

  Pour plus d'informations, consultez le guide de EventBridge l'utilisateur Amazon.

Toutefois, les flux de travail express n'émettent pas d'événements vers EventBridge. Pour surveiller l'exécution d'un flux de travail express, vous pouvez utiliser CloudWatch les journaux. Pour ce faire, sur la page State Machine Execution Details, sélectionnez les onglets Monitoring et Logging. Dans l'onglet Surveillance, vous pouvez consulter les CloudWatch mesures relatives aux événements, tels que la durée d'exécution, les erreurs d'exécution et la mémoire facturée. Dans l'onglet Journalisation, vous pouvez consulter les journaux récents et la configuration de journalisation.

> **Tip**
>
> Pour déployer un exemple d'Express Workflow sur votre site Compte AWS et apprendre à surveiller Express Workflows, consultez le module Monitoring Express Workflows de The AWS Step Functions Workshop.

EventBridge charges utiles

Un EventBridge événement peut contenir une propriété d'entrée dans sa définition. Pour certains événements, un EventBridge événement peut également contenir une propriété de sortie dans sa définition.

- Si l'entrée échappée et la sortie échappée combinées envoyées EventBridge à plus de 248 Ko, l'entrée sera exclue. De même, si la sortie échappée dépasse 248 Ko, la sortie sera exclue. Cela est dû aux quotas d' EventBridgeévénements.

- Vous pouvez déterminer si une charge utile a été tronquée à l'aide des propriétés `inputDetails` et`outputDetails`. Pour plus d'informations, consultez le type de `CloudWatchEventsExecutionDataDetails` données.
- Pour les flux de travail standard, vous pouvez voir l'intégralité des entrées et sorties en utilisant DescribeExecution.
- `DescribeExecution`n'est pas disponible pour Express Workflows. Si vous souhaitez voir l'intégralité des entrées/sorties, vous pouvez intégrer votre flux de travail express à un flux de travail standard. Une autre option consiste à utiliser les ARN d'Amazon S3. Pour plus d'informations sur l'utilisation des ARN, consultezthe section called “Utilisez les ARN d'Amazon S3 au lieu de transmettre des charges utiles importantes”.

Rubriques

Exemples d'événements Step Functions

Voici des exemples de Step Functions envoyant des événements à EventBridge :

Rubriques

Dans chaque cas, la section `detail` des données de l'événement fournit les mêmes informations que l'API `DescribeExecution`. Le champ `status` indique l'état de l'exécution au moment où l'événement a été envoyé, `RUNNING`, `SUCCEEDED`, `FAILED`, `TIMED_OUT` ou `ABORTED`, selon l'événement émis.

Exécution commencée

```
{
    "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
    }
}
```

Exécution réussie

```
{
    "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
        }
    }
}
```

Échec de l'exécution

```
{
    "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
    }
}
```

Le délai d'exécution a expiré

```
{
    "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
    }
}
```

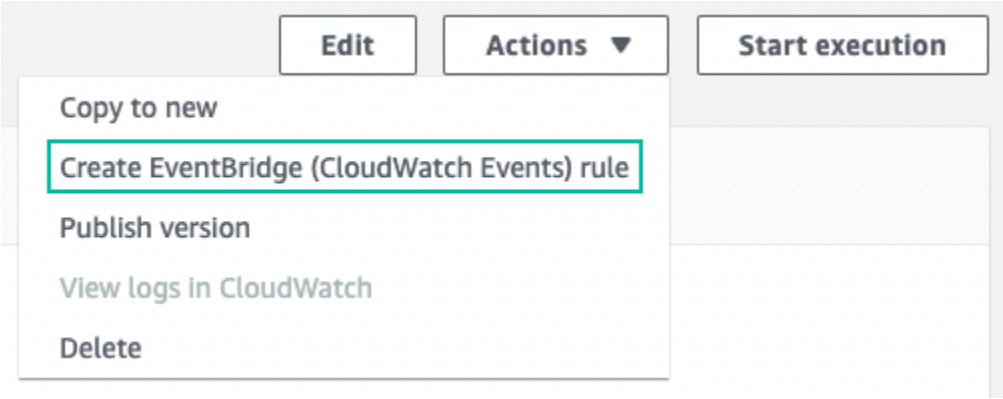
Exécution interrompue

```
{
    "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
    }
}
```

Routage d'un événement Step Functions vers EventBridge la EventBridge console

Utilisez les instructions suivantes pour savoir comment déclencher l'exécution d'une machine d'état Step Functions chaque fois qu'une machine d'état Step Functions spécifique s'exécute correctement. Vous utilisez la EventBridge console Amazon pour spécifier la machine à états dont vous souhaitez déclencher l'exécution.

1. Sur la page Détails d'une machine à états, choisissez Actions, puis choisissez la règle Create EventBridge (CloudWatch Events).



Vous pouvez également ouvrir la EventBridge console à l'adresse https://console.aws.amazon.com/events/. Dans le volet de navigation, sélectionnez Règles sous Bus.

2. Choisissez Créer une règle. Cela ouvre la page détaillée de définition de la règle.
3. Entrez un nom pour votre règle (par exemple,*`StepFunctionsEventRule`*) et entrez éventuellement une description pour la règle.
4. Pour le bus d'événements et le type de règle, conservez les sélections par défaut.
5. Choisissez Suivant. Cela ouvre la page Créer un modèle d'événement.
6. Sous Source d'événement, conservez la sélection par défaut d'AWS événements ou d'événements EventBridge partenaires.
7. Conservez les sélections par défaut pour les sections Exemple d'événement et Méthode de création.
8. Sous Modèle d'événement, procédez comme suit :

   a. Dans la liste déroulante Source de l'événement, conservez la sélection de AWS services par défaut.

   b. Dans la liste déroulante des AWS services, sélectionnez Step Functions.

   c. Dans la liste déroulante des types d'événements, sélectionnez Step Functions Execution Status Change.

   d. (Facultatif) Configurez un statut, un nom de ressource Amazon (ARN) de la machine à états ou un ARN d'exécution spécifique. Pour cette procédure, choisissez Statut (s) spécifique (s), puis sélectionnez SUCCEDED dans la liste déroulante.
9. Choisissez Suivant. Cela ouvre la page Sélectionner une ou plusieurs cibles.
10. Sous Types de cibles, conservez la sélection de AWS service par défaut.

11. Dans la liste déroulante Sélectionnez une cible, choisissez un AWS service. Par exemple, vous pouvez lancer une fonction Lambda ou exécuter une machine d'état Step Functions. Pour cette procédure, choisissez Step Functions state machine.
12. Dans la liste déroulante State machine, choisissez une machine State.
13. Sous Rôle d'exécution, conservez la sélection par défaut de Créer un nouveau rôle pour cette ressource spécifique.
14. Choisissez Suivant. Cela ouvre la page Configurer les balises.
15. sélectionnez à nouveau Next. Cela ouvre la page Réviser et créer.
16. Consultez les détails de la règle et choisissez Create rule (Créer une règle).

    La règle est créée et la page Règles s'affiche, répertoriant toutes vos EventBridge règles Amazon.

Enregistrement des appels d'API avec AWS CloudTrail

AWS Step Functions est intégré à AWS CloudTrailun service qui fournit un enregistrement des actions entreprises par un utilisateur, un rôle ou un Service AWS. CloudTrail capture tous les appels d'API Step Functions sous forme d'événements. Les appels capturés incluent des appels provenant de la Step Functions console et des appels de code vers les opérations de l' Step Functions API. À l'aide des informations collectées par CloudTrail, vous pouvez déterminer la demande qui a été faite Step Functions, l'adresse IP à partir de laquelle la demande a été faite, la date à laquelle elle a été faite et des informations supplémentaires.

Chaque événement ou entrée de journal contient des informations sur la personne ayant initié la demande. Les informations relatives à l’identité permettent de déterminer :

- Si la demande a été effectuée avec des informations d’identification d’utilisateur root ou d’utilisateur root.
- Si la demande a été faite au nom d'un utilisateur de l'IAM Identity Center.
- Si la demande a été effectuée avec les informations d’identification de sécurité temporaires d’un rôle ou d’un utilisateur fédéré.
- Si la requête a été effectuée par un autre Service AWS.

CloudTrail est actif dans votre compte Compte AWS lorsque vous créez le compte et vous avez automatiquement accès à l'historique des CloudTrail événements. L'historique des CloudTrail

événements fournit un enregistrement consultable, consultable, téléchargeable et immuable des 90 derniers jours des événements de gestion enregistrés dans un. Région AWS Pour plus d'informations, consultez la section Utilisation de l'historique des CloudTrail événements dans le guide de AWS CloudTrail l'utilisateur. La consultation de CloudTrail l'historique des événements est gratuite.

Pour un enregistrement continu des événements de vos 90 Compte AWS derniers jours, créez un magasin de données sur les événements de Trail ou CloudTrailLake.

CloudTrail sentiers

Un suivi permet CloudTrail de fournir des fichiers journaux à un compartiment Amazon S3. Tous les sentiers créés à l'aide du AWS Management Console sont multirégionaux. Vous pouvez créer un parcours à région unique ou multirégionale à l'aide du. AWS CLI Il est recommandé de créer un parcours multirégional, car vous capturez l'activité dans l'ensemble Régions AWS de votre compte. Si vous créez un parcours à région unique, vous ne pouvez voir que les événements enregistrés dans le parcours. Région AWS Pour plus d'informations sur les sentiers, consultez les sections Création d'un sentier pour votre organisation Compte AWS et Création d'un sentier pour une organisation dans le guide de AWS CloudTrail l'utilisateur.

Vous pouvez envoyer une copie de vos événements de gestion en cours dans votre compartiment Amazon S3 gratuitement CloudTrail en créant un journal. Toutefois, des frais de stockage Amazon S3 sont facturés. Pour plus d'informations sur la CloudTrail tarification, consultez la section AWS CloudTrail Tarification. Pour obtenir des informations sur la tarification Amazon S3, consultez Tarification Amazon S3.

CloudTrail Stockages de données sur les événements du lac

CloudTrail Lake vous permet d'exécuter des requêtes SQL sur vos événements. CloudTrail Lake convertit les événements existants au format JSON basé sur les lignes au format Apache ORC. ORC est un format de stockage en colonnes qui est optimisé pour une récupération rapide des données. Les événements sont agrégés dans des magasins de données d'événement. Ceux-ci constituent des collections immuables d'événements basées sur des critères que vous sélectionnez en appliquant des sélecteurs d'événements avancés. Les sélecteurs que vous appliquez à un magasin de données d'événement contrôlent les événements qui persistent et que vous pouvez interroger. Pour plus d'informations sur CloudTrail Lake, consultez la section Travailler avec AWS CloudTrail Lake dans le guide de AWS CloudTrail l'utilisateur.

CloudTrail Les stockages et requêtes de données sur les événements de Lake entraînent des coûts. Lorsque vous créez un magasin de données d'événement, vous choisissez l'option

de tarification que vous voulez utiliser pour le magasin de données d'événement. L'option de tarification détermine le coût d'ingestion et de stockage des événements, ainsi que les périodes de conservation par défaut et maximale pour le magasin de données d'événement. Pour plus d'informations sur la CloudTrail tarification, consultez la section AWS CloudTrail Tarification.

Événements liés aux données dans CloudTrail

Les événements de données fournissent des informations sur les opérations de ressources effectuées sur ou dans une ressource (par exemple, lecture ou écriture de données dans un objet Amazon S3). Ils sont également connus sous le nom opérations de plans de données. Les événements de données sont souvent des activités dont le volume est élevé. Par défaut, CloudTrail n'enregistre pas les événements liés aux données. L'historique des CloudTrail événements n'enregistre pas les événements liés aux données.

Des frais supplémentaires s'appliquent pour les événements de données. Pour plus d'informations sur la CloudTrail tarification, consultez la section AWS CloudTrail Tarification.

Vous pouvez enregistrer les événements de données pour les types de Step Functions ressources à l'aide de la CloudTrail console ou AWS CLI des opérations de CloudTrail l'API. Pour plus d'informations sur la façon de consigner les événements liés aux données, consultez les sections Enregistrement des événements liés aux données avec le AWS Management Console et Enregistrement des événements liés aux données avec le AWS Command Line Interface dans le Guide de AWS CloudTrail l'utilisateur.

Le tableau suivant répertorie les types de Step Functions ressources pour lesquels vous pouvez enregistrer des événements de données. La colonne Type d'événement de données indique la valeur à choisir dans la liste des types d'événements de données de la CloudTrail console. La colonne de valeur resources.type indique la **`resources.type`** valeur que vous devez spécifier lors de la configuration de sélecteurs d'événements avancés à l'aide des API or. AWS CLI CloudTrail La CloudTrail colonne Data APIs logged to indique les appels d'API enregistrés CloudTrail pour le type de ressource.

Vous pouvez configurer des sélecteurs d'événements avancés pour filtrer les `eventName`, `readOnly`, et `resources.ARN` des champs pour enregistrer uniquement les événements importants pour vous. Pour plus d'informations sur ces champs, consultez AdvancedFieldSelectorla référence de l'AWS CloudTrail API.

Type d'événement de données	valeur resources.type	API de données connectées à CloudTrail
Machine d'état Step Functions	`AWS::StepFunctions::StateMachine`	• InvokeHTTPEndpoint

Événements de gestion dans CloudTrail

Les événements de gestion fournissent des informations sur les opérations de gestion effectuées sur les ressources de votre Compte AWS. Ils sont également connus sous le nom opérations de plan de contrôle. Par défaut, CloudTrail enregistre les événements de gestion.

State Machine

- CreateStateMachine
- ListStateMachines
- DescribeStateMachine
- UpdateStateMachine
- DeleteStateMachine
- ValidateStateMachineDefinition
- TestState

Alias de la machine à états

- CreateStateMachineAlias
- ListStateMachineAliases
- DescribeStateMachineAlias
- UpdateStateMachineAlias
- DeleteStateMachineAlias

Version de la machine à états

- ListStateMachineVersions

- PublishStateMachineVersion
- DeleteStateMachineVersion

Exécutions

- StartExecution
- StartSyncExecution
- RedriveExecution
- ListExecutions
- DescribeExecution
- GetExecutionHistory
- DescribeStateMachineForExecution
- StopExecution

Activité

- CreateActivity
- ListActivities
- DescribeActivity
- DeleteActivity
- GetActivityTask

Jeton de tâche

- SendTaskSuccess
- SendTaskHeartbeat
- SendTaskFailure

MapRun

- ListMapRuns
- DescribeMapRun
- UpdateMapRun

Balises

- ListTagsForResource
- TagResource
- UntagResource

Exemples d'événements

Un événement représente une demande unique provenant de n'importe quelle source et inclut des informations sur l'opération d'API demandée, la date et l'heure de l'opération, les paramètres de la demande, etc. CloudTrail les fichiers journaux ne constituent pas une trace ordonnée des appels d'API publics. Les événements n'apparaissent donc pas dans un ordre spécifique.

L'exemple suivant montre un événement de CloudTrail données qui démontre`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"
}
```

L'exemple suivant montre un événement CloudTrail de gestion illustrant l'`CreateStateMachine`opération.

```
{
    "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"
}
```


Pour plus d'informations sur le contenu des CloudTrail enregistrements, voir le contenu des CloudTrail enregistrements dans le Guide de AWS CloudTrail l'utilisateur.

Journalisation à l'aideCloudWatchJournaux

Les flux de travail standard enregistrent l'historique des exécutions dansAWS Step Functions, bien que vous puissiez éventuellement configurer la journalisation sur AmazonCloudWatchJournaux.

Contrairement aux workflows standard, les workflows express n'enregistrent pas l'historique d'exécution dans AWS Step Functions. Pour voir l'historique des exécutions et les résultats d'un flux de travail Express, vous devez configurer la journalisation sur AmazonCloudWatchJournaux. La publication des journaux ne bloque pas ou ne ralentit pas les exécutions.

> **Note**
>
> Lorsque vous configurez la journalisation,CloudWatchEnregistre les fraiss'appliquera et vous serez facturé au tarif des journaux vendus. Pour plus d'informations, voirJournaux vendussous leJournauxonglet surCloudWatchPage de tarification.

Configurer la journalisation

Lorsque vous créez un flux de travail standard à l'aide de la console Step Functions, il n'est pas configuré pour activer la journalisation dansCloudWatchJournaux. Un flux de travail Express créé à l'aide de la console Step Functions sera configuré par défaut pour permettre la journalisation dansCloudWatchJournaux.

Pour les flux de travail Express, Step Functions peut créer un rôle avec les éléments nécessairesAWS Identity and Access Managementpolitique (IAM) pourCloudWatchJournaux. Si vous créez un flux de travail standard ou un flux de travail Express à l'aide de l'API, de la CLI ouAWS CloudFormation, Step Functions n'activera pas la journalisation par défaut et vous devrez vous assurer que votre rôle dispose des autorisations nécessaires.

Pour chaque exécution démarrée depuis la console, Step Functions fournit un lien versCloudWatchJournaux, configurés avec le filtre approprié pour récupérer les événements du journal spécifiques à cette exécution.

Pour configurer la journalisation, vous pouvez transmettre leLoggingConfigurationparamètre lors de l'utilisationCreateStateMachineouUpdateStateMachine. Vous pouvez approfondir l'analyse de vos données dansCloudWatchEnregistre à l'aide deCloudWatchInformations sur les journaux. Pour plus d'informations, voirAnalyse des données du journal avecCloudWatchInformations sur les journaux.

CloudWatchEnregistre les charges utiles

Les événements de l'historique d'exécution peuvent contenir des propriétés d'entrée ou de sortie dans leurs définitions. Si une entrée échappée ou une sortie échappée est envoyée àCloudWatchLes journaux dépassent 248 Ko ; ils seront tronqués en raison deCloudWatchConsigne les quotas.

- Vous pouvez déterminer si une charge utile a été tronquée en consultant le`inputDetails`et`outputDetails`propriétés. Pour plus d'informations, consultez le`HistoryEventExecutionDataDetails`Type de données.
- Pour les flux de travail standard, vous pouvez consulter l'historique complet des exécutions en utilisant`GetExecutionHistory`.
- `GetExecutionHistory`n'est pas disponible pour Express Workflows. Si vous souhaitez voir l'intégralité des entrées et des sorties, vous pouvez utiliser les ARN d'Amazon S3. Pour plus d'informations, veuillez consulter the section called “Utilisez les ARN d'Amazon S3 au lieu de transmettre des charges utiles importantes”.

Politiques IAM pour la connexion àCloudWatchJournaux

Vous devrez également configurer le rôle IAM d'exécution de votre machine d'état pour disposer de l'autorisation appropriée pour vous connecter àCloudWatchJournaux comme indiqué dans l'exemple suivant.

Exemple de politique IAM

Voici un exemple de stratégie que vous pouvez utiliser pour configurer vos autorisations. Comme indiqué dans l'exemple suivant, vous devez spécifier*dans le`Resource`champ parce queCloudWatchActions d'API, telles queCreateLogDeliveryetDescribeLogGroups, ne prend pas en chargeTypes de ressources définis parAmazon CloudWatch Logs. Pour plus d'informations, voirActions définies parAmazon CloudWatch Logs.

- Pour plus d'informations surCloudWatchressources, voirCloudWatch Logsressources et opérationsdans leAmazonCloudWatchGuide de l'utilisateur.
- Pour plus d'informations sur les autorisations dont vous avez besoin pour configurer l'envoi de journaux àCloudWatchJournaux, voirAutorisations utilisateurdans la section intituléeJournaux envoyés àCloudWatch 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": "*"
        }
    ]
}
```

Impossible d'accéder auCloudWatchJournaux

Si vous ne parvenez pas à accéder auCloudWatchJournaux, assurez-vous d'avoir effectué les opérations suivantes :

1. Vous avez configuré le rôle IAM d'exécution de votre machine d'état pour disposer de l'autorisation appropriée pour vous connecter àCloudWatchJournaux.

   Si vous utilisez leCreateStateMachineouUpdateStateMachinerequêtes, assurez-vous d'avoir spécifié le rôle IAM dans`roleArn`paramètre contenant les autorisations comme indiqué dans leexemple précédent.
2. J'ai vérifié leCloudWatchLa politique en matière de ressources des journaux ne dépasse pas la limite de 5 120 caractères pourCloudWatchEnregistre les politiques relatives aux ressources.

   Si vous avez dépassé la limite de caractères, supprimez les autorisations inutiles duCloudWatchConsigne la politique en matière de ressources, ou préfixez le nom du groupe de journaux avec`/aws/vendedlogs`, qui accordera des autorisations au groupe de journaux sans ajouter de caractères supplémentaires à la politique de ressources. Lorsque vous créez un groupe de journaux dans la console Step Functions, les noms des groupes de journaux sont préfixés par`/aws/vendedlogs/states`. Pour plus d'informations, veuillez consulter Politique relative CloudWatch aux ressources et restrictions de taille d'Amazon Logs.

Niveaux de journalisation

Vous pouvez choisir parmi `OFF`, `ALL`, `ERROR`, ou `FATAL`. Aucun journal des types d'événements lorsqu'il est défini sur `OFF` et tous les types d'événements le font lorsqu'il est défini sur `ALL`. Pour `ERROR` et `FATAL`, voir le tableau suivant.

Pour plus d'informations sur les données d'exécution affichées pour les exécutions d'Express Workflow basées sur celles-ciNiveaux de journalisation, voirExécutions de flux de travail standard et express dans la console.

Type d'événement	**`ALL`**	**`ERROR`**	**`FATAL`**	**`OFF`**
ChoiceStateEntered	✓			
ChoiceStateExited	✓			
ExecutionAborted	✓	✓	✓	

Type d'événement	ALL	ERROR	FATAL	OFF
ExecutionFailed	✓	✓	✓	
ExecutionStarted	✓			
Execution Succeeded	✓			
Execution TimedOut	✓	✓	✓	
FailStateEntered	✓	✓		
LambdaFunctionFailed	✓	✓		
LambdaFunctionScheduled	✓			
LambdaFunctionScheduleFailed	✓	✓		
LambdaFunctionStarted	✓			
LambdaFunctionStartFailed	✓	✓		
LambdaFunctionSucceeded	✓			
LambdaFunctionTimedOut	✓	✓		
MapIterationAborted	✓	✓		

Type d'événement	ALL	ERROR	FATAL	OFF
MapIterationFailed	✓	✓		
MapIterationStarted	✓			
MapIterationSucceeded	✓			
MapRunAborted	✓	✓		
MapRunFailed	✓	✓		
MapStateAborted	✓	✓		
MapStateEntered	✓			
MapStateExited	✓			
MapStateFailed	✓	✓		
MapStateStarted	✓			
MapStateSucceeded	✓			
ParallelStateAborted	✓	✓		
ParallelStateEntered	✓			
ParallelStateExited	✓			

Type d'événement	ALL	ERROR	FATAL	OFF
ParallelStateFailed	✓	✓		
ParallelStateStarted	✓			
ParallelStateSucceeded	✓			
PassStateEntered	✓			
PassStateExited	✓			
SucceedStateEntered	✓			
SucceedStateExited	✓			
TaskFailed	✓	✓		
TaskScheduled	✓			
TaskStarted	✓			
TaskStartFailed	✓	✓		
TaskStateAborted	✓	✓		
TaskStateEntered	✓			
TaskStateExited	✓			

Type d'événement	ALL	ERROR	FATAL	OFF
TaskSubmitFailed	✓	✓		
TaskSubmitted	✓			
TaskSucceeded	✓			
TaskTimedOut	✓	✓		
WaitStateAborted	✓	✓		
WaitStateEntered	✓			
WaitStateExited	✓			

AWS X-Ray et Step Functions

Vous pouvez l'utiliser AWS X-Raypour visualiser les composants de votre machine d'état, identifier les goulots d'étranglement liés aux performances et résoudre les demandes qui ont entraîné une erreur. Votre machine d'état envoie des données de suivi à X-Ray, et X-Ray traite les données pour générer une carte des services et des résumés de suivi consultables.

Lorsque X-Ray est activé pour votre machine d'état, vous pouvez suivre les demandes telles qu'elles sont exécutées dans Step Functions, dans toutes les AWS régions où X-Ray est disponible. Cela vous donne un aperçu détaillé de l'ensemble d'une demande Step Functions. Step Functions enverra des traces à X-Ray pour les exécuter par State Machine, même si aucun identifiant de trace n'est transmis par un service en amont. Vous pouvez utiliser une carte des services X-Ray pour visualiser la latence d'une demande, y compris les AWS services intégrés à X-Ray. Vous pouvez également configurer des règles d'échantillonnage pour indiquer à X-Ray quelles demandes enregistrer et à quelles fréquences d'échantillonnage, selon les critères que vous spécifiez.

Lorsque X-Ray n'est pas activé pour votre machine à états et qu'un service en amont ne transmet pas d'identifiant de trace, Step Functions n'envoie pas de traces à X-Ray pour les exécutions par

machine à états. Toutefois, si un identifiant de trace est transmis par un service en amont, Step Functions enverra ensuite des traces à X-Ray pour les exécuter par des machines à états.

Vous pouvez l'utiliser AWS X-Ray avec Step Functions dans les régions où les deux sont compatibles. Consultez les pages relatives aux points de terminaison et aux quotas de Step Functions et de X-Ray pour plus d'informations sur le support régional pour X-Ray et Step Functions.

> **Quotas combinés pour X-Ray et Step Functions**
>
> Vous pouvez ajouter des données à une trace pendant sept jours au maximum et interroger les données de trace remontant à trente jours, soit la durée pendant laquelle X-Ray stocke les données de trace. Vos traces seront soumises à des quotas X-Ray. Outre les autres quotas, X-Ray fournit une taille de trace minimale garantie de 100 Ko pour les machines à états Step Functions. Si plus de 100 Ko de données de suivi sont fournies à X-Ray, cela peut entraîner le gel du tracé. Consultez la section Quotas de service de la page Points de terminaison et quotas de X-Ray pour plus d'informations sur les autres quotas pour X-Ray.

> **Important**
>
> Step Functions ne prend pas en charge le suivi X-Ray pour les exécutions de flux de travail secondaires lancées par un état de carte distribuée, car il est facile de dépasser la limite de taille du document Trace pour de telles exécutions.

Rubriques

- Que se passe-t-il s'il n'y a aucune donnée dans la carte de suivi ou la carte des services ?

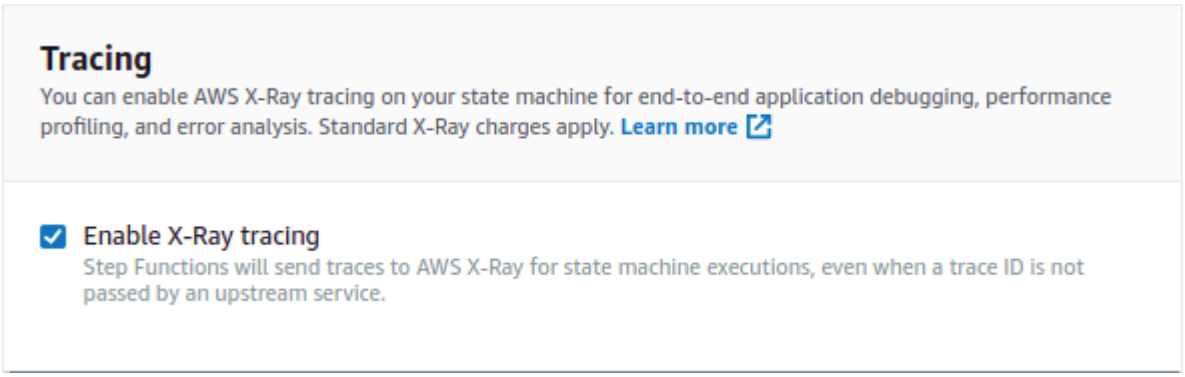
Installation et configuration

Activer le traçage X-Ray lors de la création d'une machine à états

Vous pouvez activer le suivi X-Ray lors de la création d'une nouvelle machine à états en sélectionnant Activer le traçage X-Ray sur la page Spécifier les détails.

1. Ouvrez la console Step Functions et choisissez Create state machine.
2. Sur la page Choisir une méthode de création, choisissez une option appropriée pour créer votre machine à états. Si vous choisissez Run a sample project, vous ne pouvez pas activer le suivi X-Ray lors de la création de la machine d'état, et vous devrez activer le suivi X-Ray une fois que votre machine d'état aura été créée. Pour plus d'informations sur l'activation de X-Ray dans une machine à états existante, consultezActiver X-Ray dans une machine à états existante.

   Choisissez Suivant.
3. Sur la page Spécifier les détails, configurez votre machine d'état.
4. Choisissez Enable X-Ray Tracing.



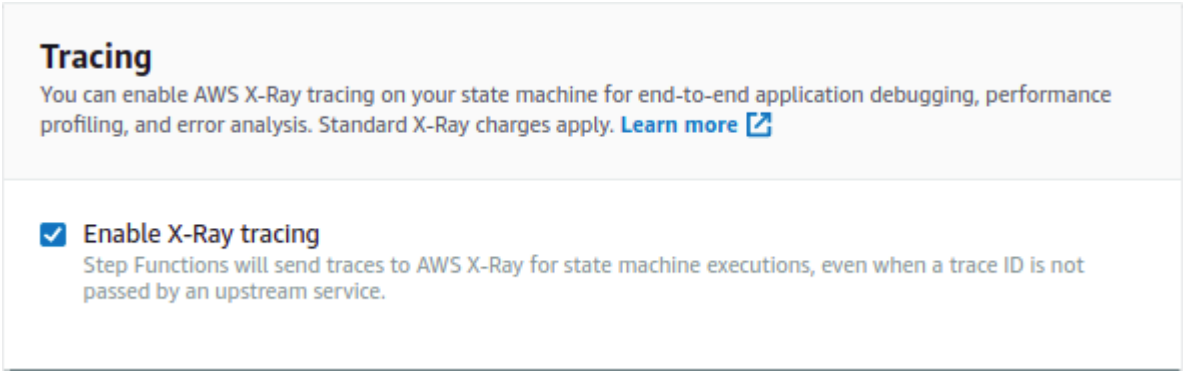
Votre machine à états Step Functions va désormais envoyer des traces à X-Ray pour les exécutions par machine à états.

> **Note**
>
> Si vous choisissez d'utiliser un rôle IAM existant, vous devez vous assurer que les écritures X-Ray sont autorisées. Pour plus d'informations sur les autorisations dont vous avez besoin, consultez les politiques IAM pour X-Ray.

Activer X-Ray dans une machine à états existante

Pour activer X-Ray sur une machine à états existante, procédez comme suit :

1. Dans la console Step Functions, sélectionnez la machine à états pour laquelle vous souhaitez activer le suivi.
2. Choisissez Modifier.
3. Choisissez Enable X-Ray Tracing.



   Vous verrez une notification vous indiquant que vous devrez peut-être apporter des modifications supplémentaires.

> **Note**
>
> Lorsque vous activez X-Ray pour une machine à états existante, vous devez vous assurer que vous disposez d'une politique IAM qui accorde des autorisations suffisantes pour permettre à X-Ray d'effectuer des traces. Vous pouvez soit en ajouter un manuellement, soit en générer un. Pour plus d'informations, consultez la section relative à la politique IAM pourPolitiques IAM pour AWS X-Ray.

4. (Facultatif) Générez automatiquement un nouveau rôle pour votre machine d'État afin d'inclure les autorisations X-Ray.
5. Choisissez Enregistrer.

Configurer le traçage X-Ray pour Step Functions

Lorsque vous exécutez pour la première fois une machine à états avec le suivi X-Ray activé, elle utilise les valeurs de configuration par défaut pour le suivi X-Ray. AWS X-Ray ne collecte pas de données pour chaque demande envoyée à une application. Il collecte plutôt des données pour un

nombre de demandes statistiquement significatif. Par défaut, la première demande est enregistrée chaque seconde, et cinq pour cent des demandes supplémentaires. Une demande par seconde est le réservoir. Ceci garantit qu'au moins une trace est enregistrée chaque seconde aussi longtemps que le service traite les demandes. 5 % est la fréquence à laquelle les demandes supplémentaires au-delà du réservoir sont échantillonnées.

Afin de ne pas encourir de frais de service lors de la mise en route, le taux d'échantillonnage par défaut est prudent. Vous pouvez configurer X-Ray pour modifier la règle d'échantillonnage par défaut et configurer des règles supplémentaires qui appliquent l'échantillonnage en fonction des propriétés du service ou de la demande.

Par exemple, vous souhaiterez peut-être désactiver l'échantillonnage et suivre toutes les demandes d'appels qui modifient l'état, le descripteur Comptes AWS ou les transactions. Pour les appels en lecture seule à volume élevé, tels que les sondages d'antécédents, les vérifications de santé ou la maintenance de la connexion, vous pouvez effectuer un échantillonnage à faible débit tout en obtenant suffisamment de données pour détecter les problèmes qui se produisent.

Pour configurer une règle d'échantillonnage pour votre machine à états :

1. Accédez à la console X-Ray.
2. Choisissez Sampling (Échantillonnage).
3. Pour créer une règle, choisissez Create sampling rule (Créer une règle d’échantillonnage).

   Pour modifier une règle, choisissez le nom d'une règle.

   Pour supprimer une règle, choisissez une règle et utilisez le menu Actions pour la supprimer.

Certaines parties des règles d'échantillonnage existantes, telles que le nom et la priorité, ne peuvent pas être modifiées. Ajoutez ou clonez plutôt une règle existante, apportez les modifications souhaitées, puis utilisez la nouvelle règle.

Pour des informations détaillées sur les règles d'échantillonnage X-Ray et sur la façon de configurer les différents paramètres, voir Configuration des règles d'échantillonnage dans la console X-Ray.

Intégrez les services en amont

Pour intégrer l'exécution des flux de travail Step Functions, tels que les flux de travail Express, Synchrone et Standard, à un service en amont, vous devez définir le`traceHeader`. Cela se fait automatiquement pour vous si vous utilisez une API HTTP dans API Gateway. Toutefois,

si vous utilisez une fonction Lambda et/ou un SDK, vous devez définir vous-même les appels d'`StartSyncExecution`API `traceHeader` on the `StartExecution`or.

Vous devez spécifier le `traceHeader` format sous la forme`\p{ASCII}#`. En outre, pour permettre à Step Functions d'utiliser le même identifiant de trace, vous devez spécifier le format sous la forme`Root={TRACE_ID};Sampled={1 or 0}`. Si vous utilisez une fonction Lambda, remplacez le par l'ID `TRACE_ID` de trace dans votre segment actuel et définissez le champ Sampled comme 1 si votre mode d'échantillonnage était vrai et `0` si votre mode d'échantillonnage était faux. La fourniture de l'ID de trace dans ce format garantit que vous obtiendrez une trace complète.

Voici un exemple écrit en Python pour montrer comment spécifier le`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)
```

Concepts

La console X-Ray

La AWS X-Ray console vous permet de visualiser les cartes de service et les traces des demandes traitées par vos applications. Vous pouvez accéder à la console pour consulter les informations détaillées collectées par X-Ray lorsqu'il est activé pour votre machine à états.

Consultez Affichage de la console X-Ray pour savoir comment accéder à la console X-Ray pour les exécutions de vos machines à états.

Pour des informations détaillées sur la console X-Ray, consultez la documentation de la console X-Ray.

Segments, sous-segments et traces

Un segment enregistre les informations relatives à une demande adressée à votre machine d'état. Il contient des informations telles que le travail effectué par votre machine d'état, et peut également contenir des sous-segments contenant des informations sur les appels en aval.

Une trace rassemble tous les segments générés par une seule demande.

Echantillonnage

Pour garantir un suivi efficace et fournir un échantillon représentatif des demandes traitées par votre application, X-Ray applique un algorithme d'échantillonnage afin de déterminer quelles demandes sont suivies. Cela peut être modifié en modifiant les règles d'échantillonnage.

Métriques

Pour votre machine à états, X-Ray mesurera le temps d'invocation, le temps de transition entre états, le temps d'exécution global de Step Functions et les variations de ce temps d'exécution. Ces informations sont accessibles via la console X-Ray.

Analyse

La console AWS X-Ray Analytics est un outil interactif permettant d'interpréter les données de suivi. Vous pouvez affiner l’ensemble de données actif en appliquant des filtres de plus en plus précis. Pour ce faire, cliquez sur les graphiques, les panneaux de métriques et les champs associés à l’ensemble de suivis sélectionné. Cela vous permet d'analyser les performances de votre machine à états, de localiser et d'identifier rapidement les problèmes de performances.

Pour des informations détaillées sur les analyses X-Ray, voir Interaction avec la console AWS X-Ray Analytics

Intégrations des services Step Functions et X-Ray

Certains des AWS services intégrés à Step Functions fournissent une intégration AWS X-Ray en ajoutant un en-tête de suivi aux demandes, en exécutant le daemon X-Ray ou en prenant des décisions d'échantillonnage et en téléchargeant les données de suivi vers X-Ray. Les autres doivent être instrumentés à l'aide du AWS X-Ray SDK. Quelques-uns ne prennent pas encore en charge

l'intégration de X-Ray. L'intégration de X-Ray est nécessaire pour fournir des données de suivi complètes lors de l'utilisation d'une intégration de service avec Step Functions

Support pour Native X-Ray

Les intégrations de services avec le support natif de X-Ray incluent :

- Amazon Simple Notification Service
- Amazon Simple Queue Service
- AWS Lambda
- AWS Step Functions

Instrumentation requise

Intégrations de services nécessitant une instrumentation X-Ray :

- Amazon Elastic Container Service
- AWS Batch
- AWS Fargate

Trace côté client uniquement

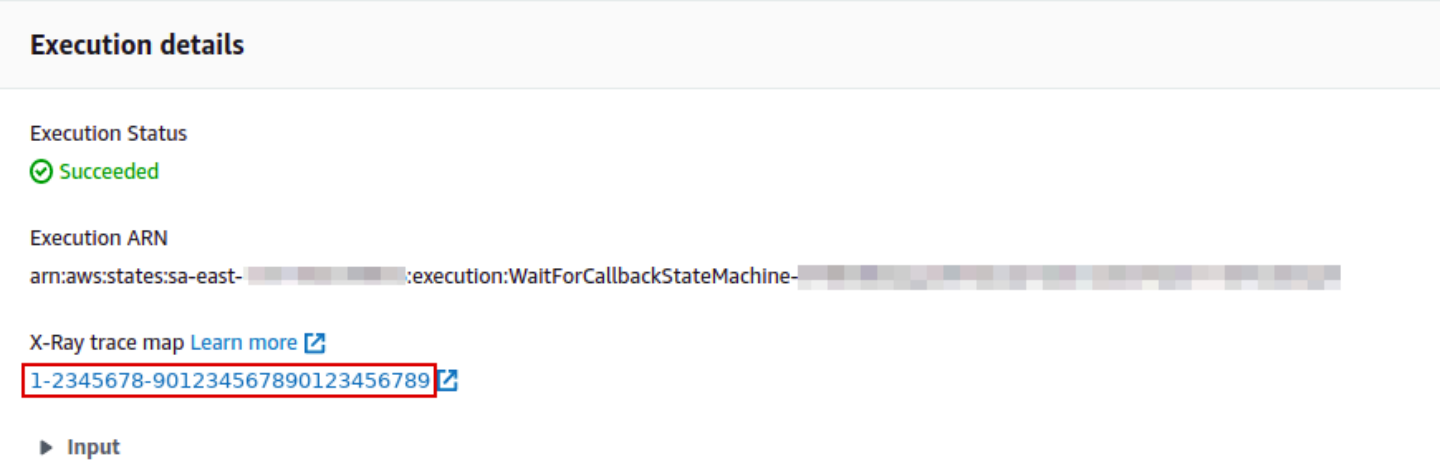
Les autres intégrations de services ne prennent pas en charge les traces X-Ray. Cependant, les traces côté client peuvent toujours être collectées :

- Amazon DynamoDB
- Amazon EMR
- Amazon SageMaker
- AWS CodeBuild
- AWS Glue

Affichage de la console X-Ray

X-Ray reçoit les données des services sous forme de segments. X-Ray regroupe les segments ayant une requête commune dans des traces. X-Ray traite les traces pour générer un graphe de service fournissant une représentation visuelle de votre application.

Après avoir démarré l'exécution de votre machine à états, vous pouvez consulter ses traces X-Ray en cliquant sur le lien X-Ray Trace Map dans la section Détails de l'exécution.



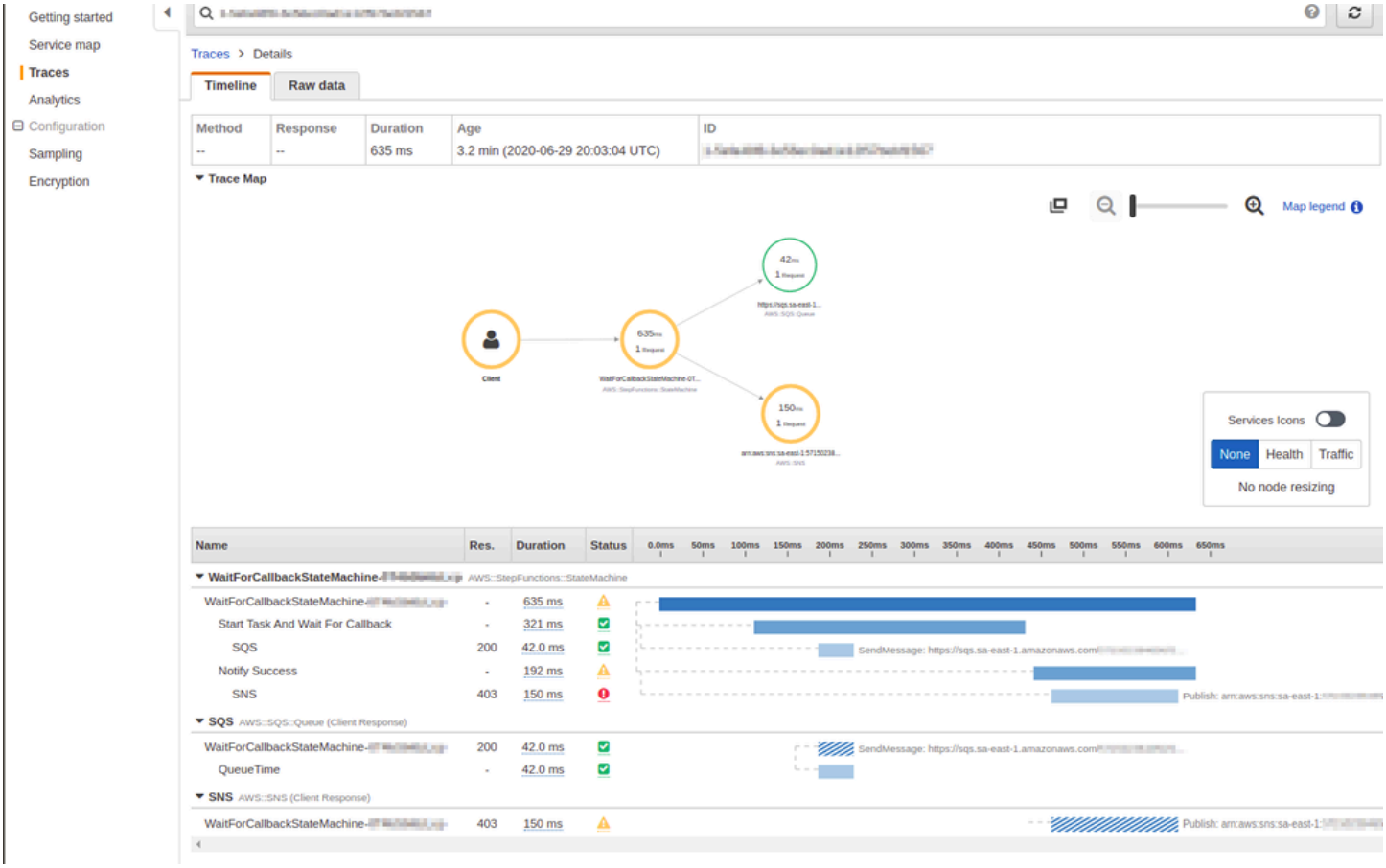
Une fois que vous avez activé X-Ray pour votre machine d'état, vous pouvez consulter les informations de suivi de ses exécutions dans la console X-Ray.

Afficher les informations de traçage X-Ray pour Step Functions

Les étapes suivantes illustrent le type d'informations que vous pouvez voir dans la console après avoir activé X-Ray et exécuté une exécution. Les traces X-Ray de l'Exemple de modèle de rappel (Amazon SQS, Amazon SNS, Lambda) exemple de projet sont affichées.

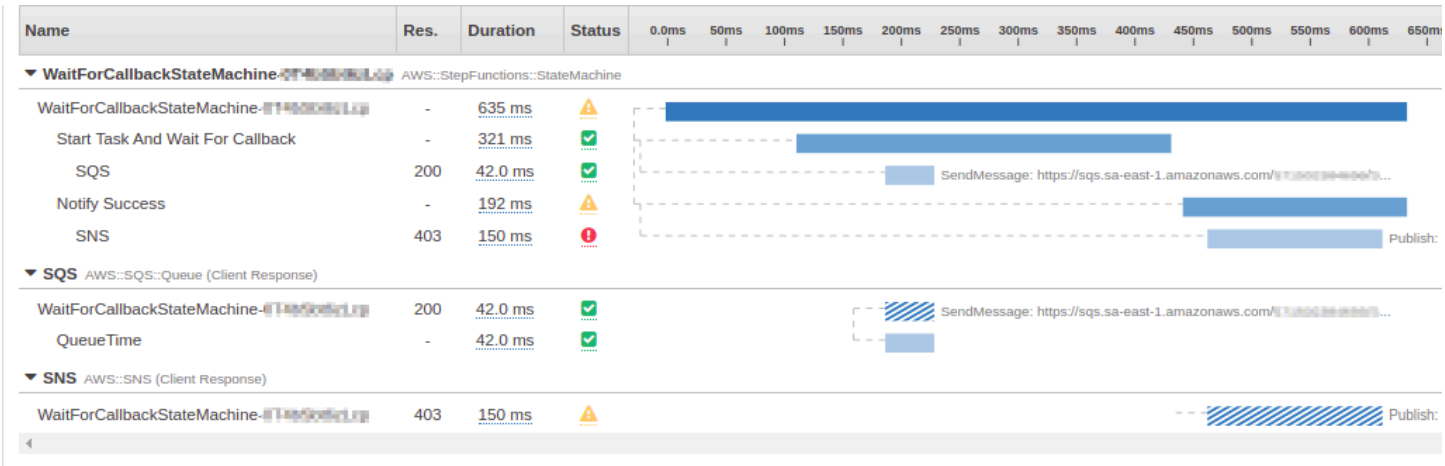
Suivis

Une fois l'exécution terminée, vous pouvez accéder à la console X-Ray, où vous verrez la page X-Ray Traces. Cela affiche une vue d'ensemble de la carte des services ainsi que des informations de suivi et de segment pour votre machine à états.

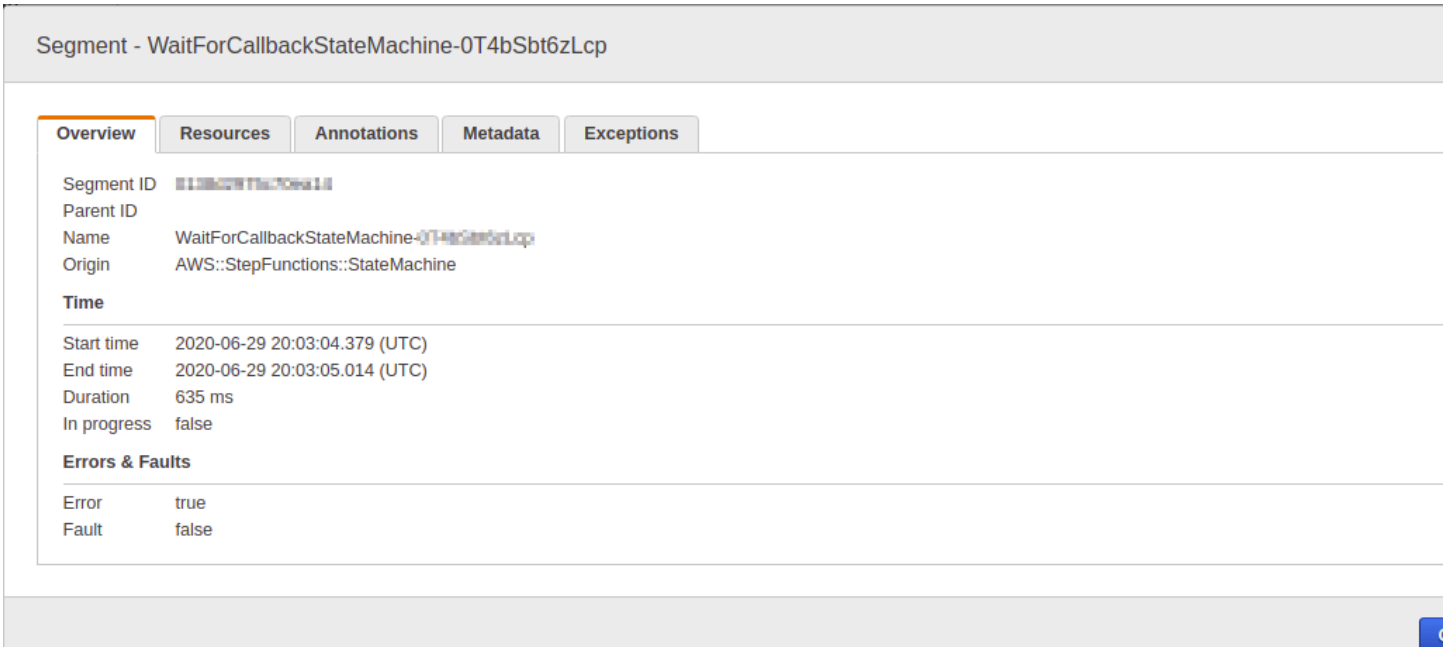


Cartographie des services

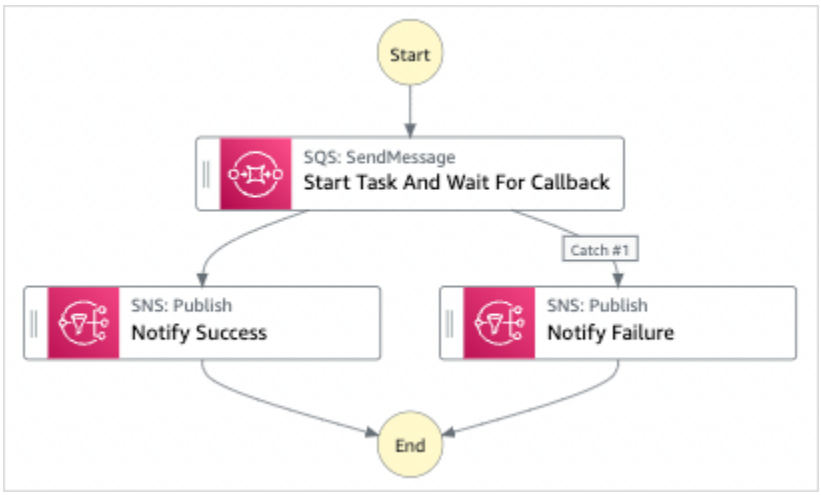
La carte des services de la console X-Ray vous permet d'identifier les services présentant des erreurs, des connexions présentant une latence élevée, ou de consulter les traces des demandes infructueuses.



Sur la carte de suivi, vous pouvez choisir un nœud de service pour afficher les demandes relatives à ce nœud, ou une limite entre deux nœuds pour afficher les demandes ayant transité par cette connexion. Ici, le `WaitForCallBack` nœud a été sélectionné et vous pouvez consulter des informations supplémentaires sur son exécution et son état de réponse.



Vous pouvez voir comment la carte du service X-Ray est corrélée à la machine à états. Il existe un nœud de carte des services pour chaque intégration de services appelé par Step Functions, à condition qu'il soit compatible avec X-Ray.



Segments et sous-segments

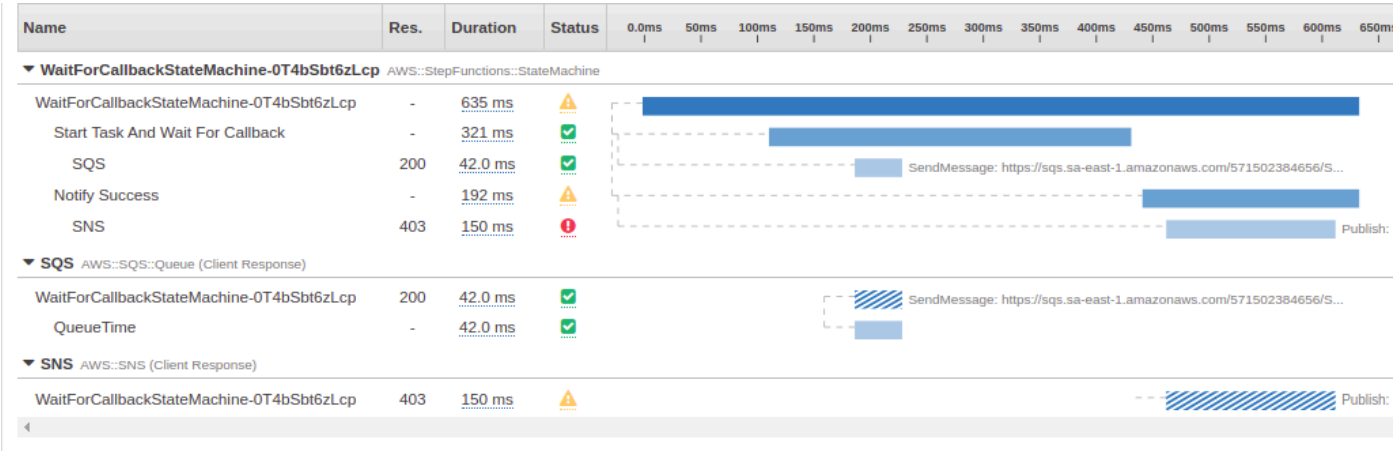
Une trace est un ensemble de segments générés par une seule demande. Chaque segment fournit le nom de la ressource, des détails sur la demande et des détails sur le travail effectué. Sur la page

Traces, vous pouvez voir les segments et, s'ils sont développés, les sous-segments correspondants. Vous pouvez choisir un segment ou un sous-segment pour afficher des informations détaillées le concernant.

Choisissez chacun des onglets pour voir comment les informations relatives aux segments et sous-segments sont affichées.

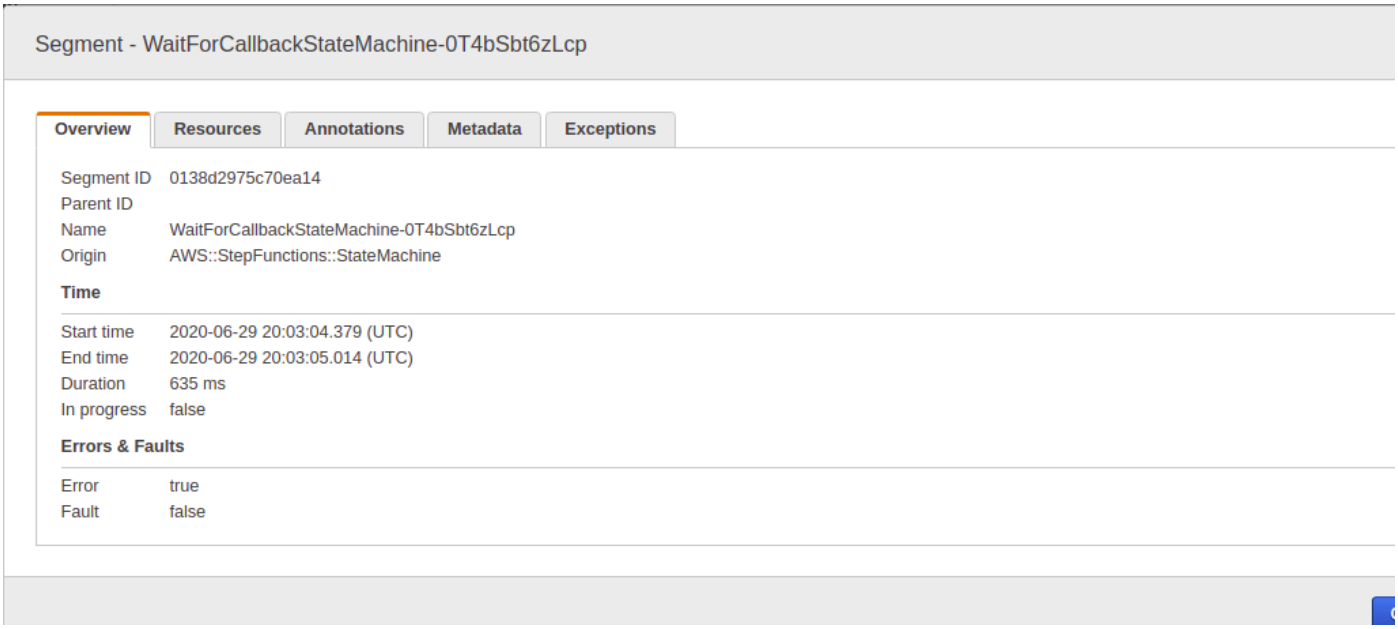
Overview of Segments

Vue d'ensemble des segments et sous-segments de cette machine à états. Il existe un segment différent pour chaque nœud sur la carte des services.



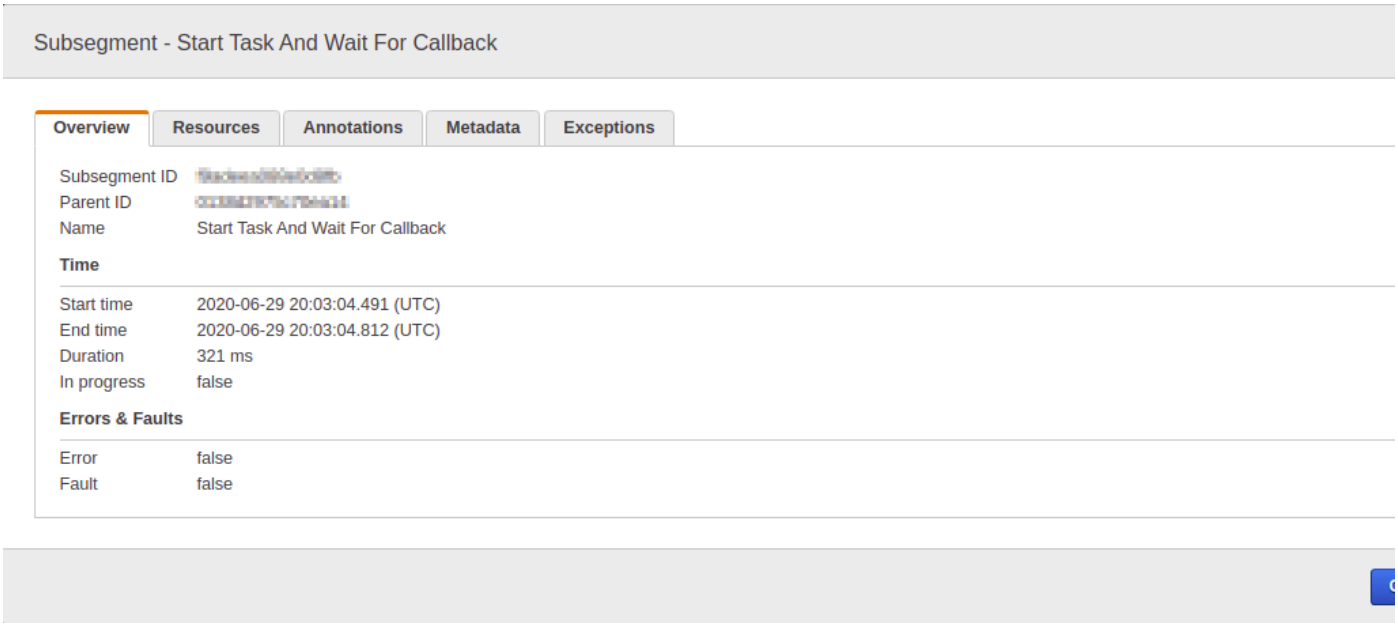
View segment detail

Le choix d'un segment fournit le nom de la ressource, des détails sur la demande et des détails sur le travail effectué.



View subsegment detail

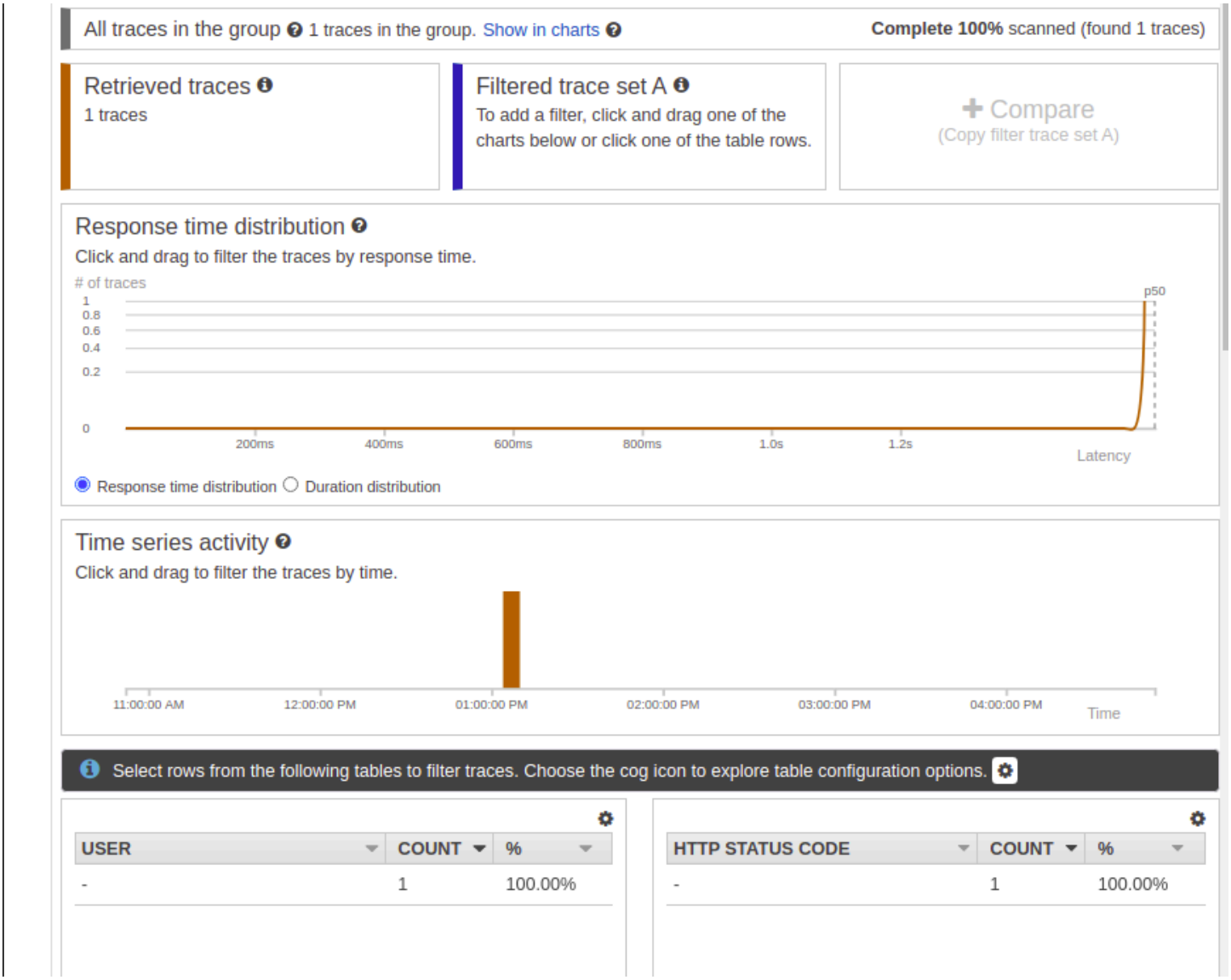
Un segment peut décomposer les données concernant le travail effectué en sous-segments. Le choix d'un sous-segment vous permet d'afficher des informations et des détails de chronométrage plus précis. Un sous-segment peut contenir des informations supplémentaires sur un appel à un AWS service, à une API HTTP externe ou à une base de données SQL.



Analyse

La console AWS X-Ray Analytics est un outil interactif permettant d'interpréter les données de suivi. Vous pouvez l'utiliser pour comprendre plus facilement les performances de votre machine à états. La console vous permet d'explorer, d'analyser et de visualiser les suivis grâce à des graphiques interactifs sur les temps de réponse et la chronologie. Cela peut vous aider à identifier rapidement les problèmes de performances et de latence.

Vous pouvez affiner l’ensemble de données actif en appliquant des filtres de plus en plus précis. Pour ce faire, cliquez sur les graphiques, les panneaux de métriques et les champs associés à l’ensemble de suivis sélectionné.

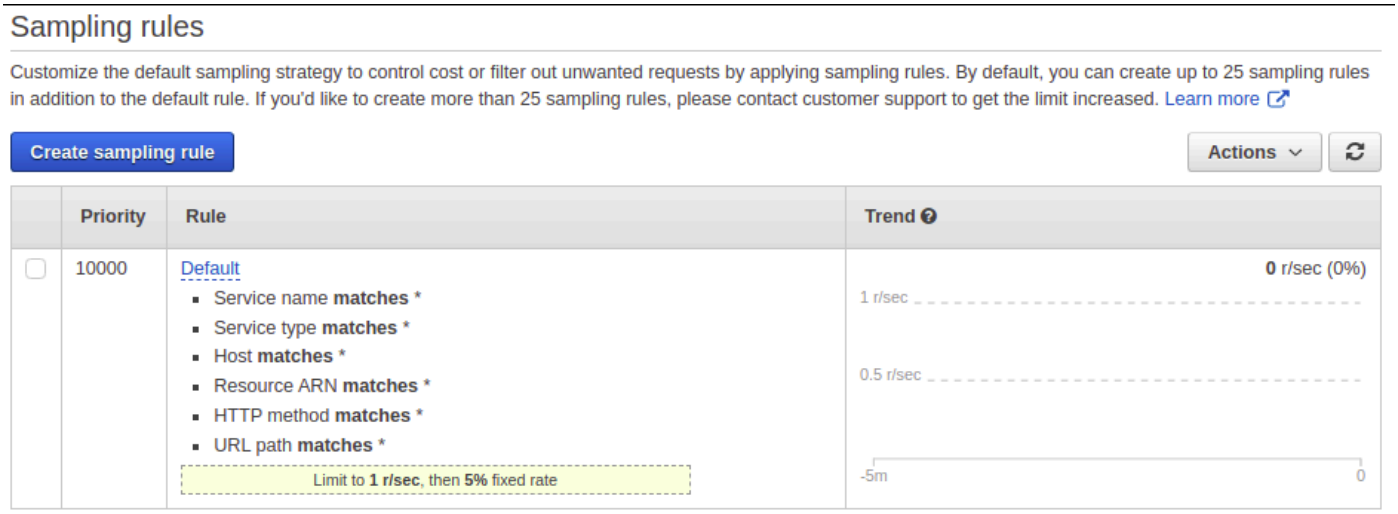


Configuration

Vous pouvez configurer les options d'échantillonnage et de chiffrement depuis la console X-Ray.

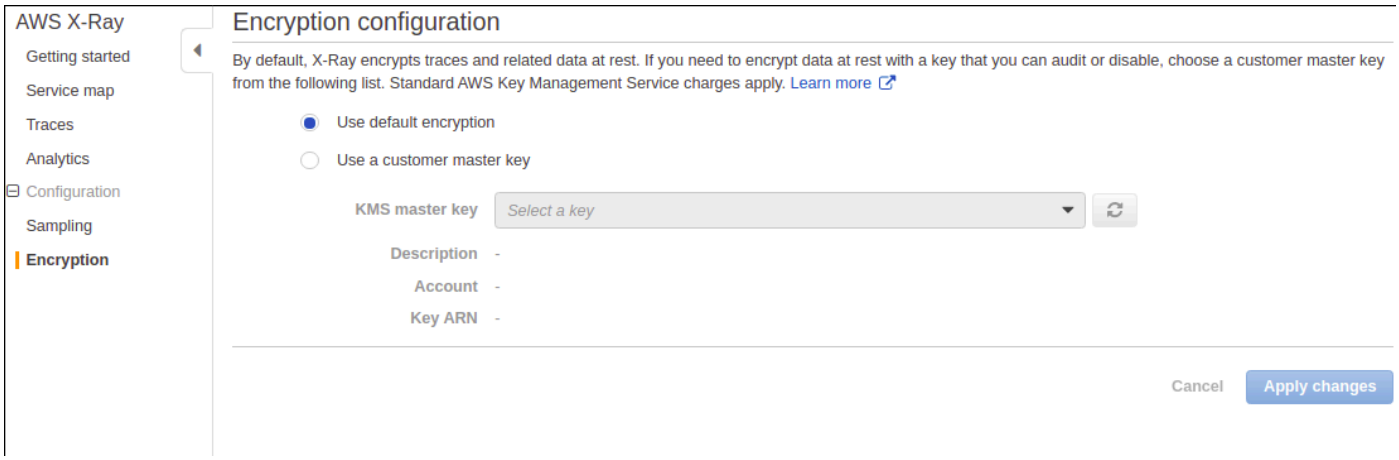
Sampling

Choisissez Échantillonnage pour afficher les détails relatifs à la fréquence d'échantillonnage et à la configuration. Vous pouvez modifier les règles d'échantillonnage pour contrôler la quantité de données que vous enregistrez et modifier le comportement d'échantillonnage en fonction de vos besoins spécifiques.



Encryption

Choisissez Chiffrement pour modifier les paramètres de chiffrement. Vous pouvez utiliser le paramètre par défaut, dans lequel X-Ray chiffre les traces et les dates au repos, ou, si nécessaire, vous pouvez choisir une clé principale client. AWS KMSLes frais standard s'appliquent dans ce dernier cas.



Que se passe-t-il s'il n'y a aucune donnée dans la carte de suivi ou la carte des services ?

Si vous avez activé X-Ray, mais que vous ne voyez aucune donnée dans la console X-Ray, vérifiez que :

- Vos rôles IAM sont correctement configurés pour permettre l'écriture dans X-Ray.
- Les règles d'échantillonnage permettent l'échantillonnage des données.

- Comme il peut s'écouler un court délai avant que les rôles IAM nouvellement créés ou modifiés ne soient appliqués, vérifiez à nouveau le tracé ou les cartes des services après quelques minutes.
- Si Data Not Found apparaît dans le panneau X-Ray Traces, vérifiez les paramètres de votre compte IAM et assurez-vous qu'ils AWS Security Token Service sont activés pour la région souhaitée. Pour plus d'informations, consultez la section Activation et désactivation AWS STS dans et Région AWS dans le guide de l'utilisateur IAM.

Utilisation d’Notifications des utilisateurs AWS avec AWS Step Functions

Vous pouvez l'utiliser Notifications des utilisateurs AWSpour configurer des canaux de diffusion afin d'être informé AWS Step Functions des événements. Vous recevez une notification lorsqu'un événement correspond à une règle que vous avez spécifiée. Vous pouvez recevoir des notifications relatives à des événements via plusieurs canaux, notamment des e-mails, des notifications de chat AWS Chatbot ou des notifications push AWS Console Mobile Application. Vous pouvez également voir les notifications dans le Centre de notifications de la console. Notifications des utilisateurs prend en charge l’agrégation, ce qui peut réduire le nombre de notifications que vous recevez lors d’événements spécifiques.

Sécurité dans AWS Step Functions

Cette section fournit des informations sur AWS Step Functions la sécurité et l'authentification.

Step Functions utilise IAM pour contrôler l'accès à d'autres AWS services et ressources. Pour obtenir un vue d'ensemble du fonctionnement d'IAM, consultez Présentation de la gestion des accès dans le Guide de l'utilisateur IAM. Pour obtenir une présentation des informations d'identification de sécurité, consultez Informations d'identification de sécuritéAWS dans le Référence générale d'Amazon Web Services.

Protection des données dans AWS Step Functions

Le modèle de responsabilité AWS partagée de s'applique à la protection des données dans AWS Step Functions. Comme décrit dans ce modèle, AWS est chargé de protéger l'infrastructure mondiale qui gère tous les AWS Cloud. La gestion du contrôle de votre contenu hébergé sur cette infrastructure relève de votre responsabilité. Vous êtes également responsable des tâches de configuration et de gestion de la sécurité des Services AWS que vous utilisez. Pour plus d'informations sur la confidentialité des données, consultez Questions fréquentes (FAQ) sur la confidentialité des données. Pour en savoir plus sur la protection des données en Europe, consultez le billet de blog Modèle de responsabilité partagée AWS et RGPD (Règlement général sur la protection des données) sur le Blog de sécuritéAWS .

À des fins de protection des données, nous vous recommandons de protéger les Compte AWS informations d'identification et de configurer les utilisateurs individuels avec AWS IAM Identity Center ou AWS Identity and Access Management (IAM). Ainsi, chaque utilisateur se voit attribuer uniquement les autorisations nécessaires pour exécuter ses tâches. Nous vous recommandons également de sécuriser vos données comme indiqué ci-dessous :

- Utilisez l'authentification multifactorielle (MFA) avec chaque compte.
- Utilisez le protocole SSL/TLS pour communiquer avec les ressources. AWS Nous exigeons TLS 1.2 et recommandons TLS 1.3.
- Configurez l'API et la journalisation de l'activité des utilisateurs avec AWS CloudTrail.
- Utilisez des solutions de AWS chiffrement, ainsi que tous les contrôles de sécurité par défaut qu'ils contiennent Services AWS.
- Utilisez des services de sécurité gérés avancés tels qu'Amazon Macie, qui contribuent à la découverte et à la sécurisation des données sensibles stockées dans Amazon S3.

- Si vous avez besoin de modules cryptographiques validés par la norme FIPS 140-2 pour accéder AWS via une interface de ligne de commande ou une API, utilisez un point de terminaison FIPS. Pour plus d'informations sur les points de terminaison FIPS (Federal Information Processing Standard) disponibles, consultez Federal Information Processing Standard (FIPS) 140-2 (Normes de traitement de l'information fédérale).

Nous vous recommandons fortement de ne jamais placer d'informations confidentielles ou sensibles, telles que les adresses e-mail de vos clients, dans des balises ou des champs de texte libre tels que le champ Name (Nom). Cela inclut lorsque vous travaillez avec Step Functions ou d'autres outils Services AWS à l'aide de la console, de l'API ou AWS des SDK. AWS CLI Toutes les données que vous entrez dans des balises ou des champs de texte de forme libre utilisés pour les noms peuvent être utilisées à des fins de facturation ou dans les journaux de diagnostic. Si vous fournissez une adresse URL à un serveur externe, nous vous recommandons fortement de ne pas inclure d'informations d'identification dans l'adresse URL permettant de valider votre demande adressée à ce serveur.

Chiffrement dans AWS Step Functions

Chiffrement au repos

Step Functions chiffre toujours vos données au repos. Les données entrantes AWS Step Functions sont cryptées au repos à l'aide d'un chiffrement transparent côté serveur. Cela réduit la lourdeur opérationnelle et la complexité induites par la protection des données sensibles. Le chiffrement au repos vous permet de créer des applications sensibles en matière de sécurité qui sont conformes aux exigences réglementaires et de chiffrement

Chiffrement en transit

Step Functions chiffre les données en transit entre le service et d'autres AWS services intégrés (voirUtilisation AWS Step Functions avec d'autres services). Toutes les données transmises entre Step Functions et les services intégrés sont cryptées à l'aide du protocole TLS (Transport Layer Security).

Identity and Access Management dans AWS Step Functions

L'accès à AWS Step Functions nécessite des informations d'identification qui AWS peuvent être utilisées pour authentifier vos demandes. Ces informations d'identification doivent être autorisées

à accéder aux AWS ressources, par exemple pour récupérer des données d'événements à partir d'autres AWS ressources. Les sections suivantes fournissent des détails sur la façon dont vous pouvez utiliser AWS Identity and Access Management (IAM) et Step Functions pour sécuriser vos ressources en contrôlant qui peut y accéder.

AWS Identity and Access Management (IAM) est un outil Service AWS qui permet à un administrateur de contrôler en toute sécurité l'accès aux AWS ressources. Les administrateurs IAM contrôlent qui peut être authentifié (connecté) et autorisé (autorisé) à utiliser Step Functions les ressources. IAM est un Service AWS outil que vous pouvez utiliser sans frais supplémentaires.

Public ciblé

La façon dont vous utilisez AWS Identity and Access Management (IAM) varie en fonction du travail que vous effectuez. Step Functions

Utilisateur du service : si vous utilisez le Step Functions service pour effectuer votre travail, votre administrateur vous fournit les informations d'identification et les autorisations dont vous avez besoin. Au fur et à mesure que vous utilisez de nouvelles Step Functions fonctionnalités pour effectuer votre travail, vous aurez peut-être besoin d'autorisations supplémentaires. En comprenant bien la gestion des accès, vous saurez demander les autorisations appropriées à votre administrateur. Si vous ne pouvez pas accéder à une fonctionnalité dans Step Functions, consultez Résolution des problèmes AWS Step Functions d'identité et d'accès.

Administrateur du service — Si vous êtes responsable des Step Functions ressources de votre entreprise, vous avez probablement un accès complet à Step Functions. C'est à vous de déterminer les Step Functions fonctionnalités et les ressources auxquelles les utilisateurs de votre service doivent accéder. Vous devez ensuite soumettre les demandes à votre administrateur IAM pour modifier les autorisations des utilisateurs de votre service. Consultez les informations sur cette page pour comprendre les concepts de base d’IAM. Pour en savoir plus sur la manière dont votre entreprise peut utiliser IAM avec Step Functions, voirComment AWS Step Functions fonctionne avec IAM.

Administrateur IAM – Si vous êtes un administrateur IAM, vous souhaiterez peut-être en savoir plus sur la façon d’écrire des politiques pour gérer l’accès à Step Functions. Pour consulter des exemples de politiques Step Functions basées sur l'identité que vous pouvez utiliser dans IAM, consultez. Exemples de politiques basées sur l'identité pour AWS Step Functions

Authentification par des identités

L'authentification est la façon dont vous vous connectez à AWS l'aide de vos informations d'identification. Vous devez être authentifié (connecté à AWS) en tant qu'utilisateur IAM ou en assumant un rôle IAM. Utilisateur racine d'un compte AWS

Vous pouvez vous connecter en AWS tant qu'identité fédérée en utilisant les informations d'identification fournies par le biais d'une source d'identité. AWS IAM Identity Center Les utilisateurs (IAM Identity Center), l'authentification unique de votre entreprise et vos informations d'identification Google ou Facebook sont des exemples d'identités fédérées. Lorsque vous vous connectez avec une identité fédérée, votre administrateur aura précédemment configuré une fédération d'identités avec des rôles IAM. Lorsque vous accédez à AWS l'aide de la fédération, vous assumez indirectement un rôle.

Selon le type d'utilisateur que vous êtes, vous pouvez vous connecter au portail AWS Management Console ou au portail AWS d'accès. Pour plus d'informations sur la connexion à AWS, consultez la section Comment vous connecter à votre compte Compte AWS dans le guide de Connexion à AWS l'utilisateur.

Si vous y accédez AWS par programmation, AWS fournit un kit de développement logiciel (SDK) et une interface de ligne de commande (CLI) pour signer cryptographiquement vos demandes à l'aide de vos informations d'identification. Si vous n'utilisez pas d' AWS outils, vous devez signer vous-même les demandes. Pour plus d'informations sur l'utilisation de la méthode recommandée pour signer vous-même les demandes, consultez la section Signature des demandes AWS d'API dans le guide de l'utilisateur IAM.

Quelle que soit la méthode d'authentification que vous utilisez, vous devrez peut-être fournir des informations de sécurité supplémentaires. Par exemple, il vous AWS recommande d'utiliser l'authentification multifactorielle (MFA) pour renforcer la sécurité de votre compte. Pour en savoir plus, consultez Authentification multifactorielle dans le Guide de l'utilisateur AWS IAM Identity Center et Utilisation de l'authentification multifactorielle (MFA) dans l'interface AWS dans le Guide de l'utilisateur IAM.

Compte AWS utilisateur root

Lorsque vous créez un Compte AWS, vous commencez par une identité de connexion unique qui donne un accès complet à toutes Services AWS les ressources du compte. Cette identité est appelée utilisateur Compte AWS root et est accessible en vous connectant avec l'adresse e-mail et le mot de passe que vous avez utilisés pour créer le compte. Il est vivement recommandé de ne pas

utiliser l'utilisateur racine pour vos tâches quotidiennes. Protégez vos informations d'identification d'utilisateur racine et utilisez-les pour effectuer les tâches que seul l'utilisateur racine peut effectuer. Pour obtenir la liste complète des tâches qui vous imposent de vous connecter en tant qu'utilisateur root, consultez Tâches nécessitant des informations d'identification d'utilisateur root dans le Guide de l'utilisateur IAM.

Identité fédérée

La meilleure pratique consiste à obliger les utilisateurs humains, y compris ceux qui ont besoin d'un accès administrateur, à utiliser la fédération avec un fournisseur d'identité pour accéder à l'aide Services AWS d'informations d'identification temporaires.

Une identité fédérée est un utilisateur de l'annuaire des utilisateurs de votre entreprise, d'un fournisseur d'identité Web AWS Directory Service, du répertoire Identity Center ou de tout utilisateur qui y accède à l'aide des informations d'identification fournies Services AWS par le biais d'une source d'identité. Lorsque des identités fédérées y accèdent Comptes AWS, elles assument des rôles, qui fournissent des informations d'identification temporaires.

Pour une gestion des accès centralisée, nous vous recommandons d'utiliser AWS IAM Identity Center. Vous pouvez créer des utilisateurs et des groupes dans IAM Identity Center, ou vous pouvez vous connecter et synchroniser avec un ensemble d'utilisateurs et de groupes dans votre propre source d'identité afin de les utiliser dans toutes vos applications Comptes AWS et applications. Pour obtenir des informations sur IAM Identity Center, consultez Qu'est-ce que IAM Identity Center ? dans le Guide de l'utilisateur AWS IAM Identity Center .

Utilisateurs et groupes IAM

Un utilisateur IAM est une identité au sein de vous Compte AWS qui possède des autorisations spécifiques pour une seule personne ou une seule application. Dans la mesure du possible, nous vous recommandons de vous appuyer sur des informations d'identification temporaires plutôt que de créer des utilisateurs IAM ayant des informations d'identification à long terme tels que les clés d'accès. Toutefois, si certains cas d'utilisation spécifiques nécessitent des informations d'identification à long terme avec les utilisateurs IAM, nous vous recommandons de faire pivoter les clés d'accès. Pour plus d'informations, consultez Rotation régulière des clés d'accès pour les cas d'utilisation nécessitant des informations d'identification dans le Guide de l'utilisateur IAM.

Un groupe IAM est une identité qui concerne un ensemble d'utilisateurs IAM. Vous ne pouvez pas vous connecter en tant que groupe. Vous pouvez utiliser les groupes pour spécifier des autorisations pour plusieurs utilisateurs à la fois. Les groupes permettent de gérer plus facilement les autorisations

pour de grands ensembles d'utilisateurs. Par exemple, vous pouvez avoir un groupe nommé IAMAdmins et accorder à ce groupe les autorisations d'administrer des ressources IAM.

Les utilisateurs sont différents des rôles. Un utilisateur est associé de manière unique à une personne ou une application, alors qu'un rôle est conçu pour être endossé par tout utilisateur qui en a besoin. Les utilisateurs disposent d'informations d'identification permanentes, mais les rôles fournissent des informations d'identification temporaires. Pour en savoir plus, consultez Quand créer un utilisateur IAM (au lieu d'un rôle) dans le Guide de l'utilisateur IAM.

Rôles IAM

Un rôle IAM est une identité au sein de vous Compte AWS dotée d'autorisations spécifiques. Le concept ressemble à celui d'utilisateur IAM, mais le rôle IAM n'est pas associé à une personne en particulier. Vous pouvez assumer temporairement un rôle IAM dans le en AWS Management Console changeant de rôle. Vous pouvez assumer un rôle en appelant une opération d' AWS API AWS CLI ou en utilisant une URL personnalisée. Pour plus d'informations sur les méthodes d'utilisation des rôles, consultez Utilisation de rôles IAM dans le Guide de l'utilisateur IAM.

Les rôles IAM avec des informations d'identification temporaires sont utiles dans les cas suivants :

- Accès utilisateur fédéré – Pour attribuer des autorisations à une identité fédérée, vous créez un rôle et définissez des autorisations pour le rôle. Quand une identité externe s'authentifie, l'identité est associée au rôle et reçoit les autorisations qui sont définies par celui-ci. Pour obtenir des informations sur les rôles pour la fédération, consultez Création d'un rôle pour un fournisseur d'identité tiers (fédération) dans le Guide de l'utilisateur IAM. Si vous utilisez IAM Identity Center, vous configurez un jeu d'autorisations. IAM Identity Center met en corrélation le jeu d'autorisations avec un rôle dans IAM afin de contrôler à quoi vos identités peuvent accéder après leur authentification. Pour plus d'informations sur les jeux d'autorisations, consultez la rubrique Jeux d'autorisations dans le Guide de l'utilisateur AWS IAM Identity Center .
- Autorisations d'utilisateur IAM temporaires : un rôle ou un utilisateur IAM peut endosser un rôle IAM pour profiter temporairement d'autorisations différentes pour une tâche spécifique.
- Accès intercompte : vous pouvez utiliser un rôle IAM pour permettre à un utilisateur (principal de confiance) d'un compte différent d'accéder aux ressources de votre compte. Les rôles constituent le principal moyen d'accorder l'accès intercompte. Toutefois, dans certains Services AWS cas, vous pouvez associer une politique directement à une ressource (au lieu d'utiliser un rôle comme proxy). Pour en savoir plus sur la différence entre les rôles et les politiques basées sur les ressources pour l'accès intercompte, consultez Différence entre les rôles IAM et les politiques basées sur les ressources dans le Guide de l'utilisateur IAM.

- Accès multiservices — Certains Services AWS utilisent des fonctionnalités dans d'autres Services AWS. Par exemple, lorsque vous effectuez un appel dans un service, il est courant que ce service exécute des applications dans Amazon EC2 ou stocke des objets dans Amazon S3. Un service peut le faire en utilisant les autorisations d'appel du principal, un rôle de service ou un rôle lié au service.
  - Sessions d'accès direct (FAS) : lorsque vous utilisez un utilisateur ou un rôle IAM pour effectuer des actions AWS, vous êtes considéré comme un mandant. Lorsque vous utilisez certains services, vous pouvez effectuer une action qui initie une autre action dans un autre service. FAS utilise les autorisations du principal appelant et Service AWS, associées Service AWS à la demande, pour adresser des demandes aux services en aval. Les demandes FAS ne sont effectuées que lorsqu'un service reçoit une demande qui nécessite des interactions avec d'autres personnes Services AWS ou des ressources pour être traitée. Dans ce cas, vous devez disposer d'autorisations nécessaires pour effectuer les deux actions. Pour plus de détails sur la politique relative à la transmission de demandes FAS, consultez Sessions de transmission d'accès.
  - Rôle de service : il s'agit d'un rôle IAM attribué à un service afin de réaliser des actions en votre nom. Un administrateur IAM peut créer, modifier et supprimer une fonction du service à partir d'IAM. Pour plus d'informations, consultez Création d'un rôle pour la délégation d'autorisations à un Service AWS dans le Guide de l'utilisateur IAM.
  - Rôle lié à un service — Un rôle lié à un service est un type de rôle de service lié à un. Service AWS Le service peut endosser le rôle afin d'effectuer une action en votre nom. Les rôles liés à un service apparaissent dans votre Compte AWS répertoire et appartiennent au service. Un administrateur IAM peut consulter, mais ne peut pas modifier, les autorisations concernant les rôles liés à un service.
- Applications exécutées sur Amazon EC2 : vous pouvez utiliser un rôle IAM pour gérer les informations d'identification temporaires pour les applications qui s'exécutent sur une instance EC2 et qui envoient des demandes d'API. AWS CLI AWS Cette solution est préférable au stockage des clés d'accès au sein de l'instance EC2. Pour attribuer un AWS rôle à une instance EC2 et le mettre à la disposition de toutes ses applications, vous devez créer un profil d'instance attaché à l'instance. Un profil d'instance contient le rôle et permet aux programmes qui s'exécutent sur l'instance EC2 d'obtenir des informations d'identification temporaires. Pour plus d'informations, consultez Utilisation d'un rôle IAM pour accorder des autorisations à des applications s'exécutant sur des instances Amazon EC2 dans le Guide de l'utilisateur IAM.

Pour savoir dans quel cas utiliser des rôles ou des utilisateurs IAM, consultez Quand créer un rôle IAM (au lieu d'un utilisateur) dans le Guide de l'utilisateur IAM.

Gestion des accès à l’aide de politiques

Vous contrôlez l'accès en AWS créant des politiques et en les associant à AWS des identités ou à des ressources. Une politique est un objet AWS qui, lorsqu'il est associé à une identité ou à une ressource, définit leurs autorisations. AWS évalue ces politiques lorsqu'un principal (utilisateur, utilisateur root ou session de rôle) fait une demande. Les autorisations dans les politiques déterminent si la demande est autorisée ou refusée. La plupart des politiques sont stockées AWS sous forme de documents JSON. Pour plus d’informations sur la structure et le contenu des documents de politique JSON, consultez Vue d’ensemble des politiques JSON dans le Guide de l’utilisateur IAM.

Les administrateurs peuvent utiliser les politiques AWS JSON pour spécifier qui a accès à quoi. C’est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

Par défaut, les utilisateurs et les rôles ne disposent d’aucune autorisation. Pour octroyer aux utilisateurs des autorisations d’effectuer des actions sur les ressources dont ils ont besoin, un administrateur IAM peut créer des politiques IAM. L’administrateur peut ensuite ajouter les politiques IAM aux rôles et les utilisateurs peuvent assumer les rôles.

Les politiques IAM définissent les autorisations d’une action, quelle que soit la méthode que vous utilisez pour exécuter l’opération. Par exemple, supposons que vous disposiez d’une politique qui autorise l’action `iam:GetRole`. Un utilisateur appliquant cette politique peut obtenir des informations sur le rôle à partir de AWS Management Console AWS CLI, de ou de l' AWS API.

Politiques basées sur l’identité

Les politiques basées sur l’identité sont des documents de politique d’autorisations JSON que vous pouvez attacher à une identité telle qu’un utilisateur, un groupe d’utilisateurs ou un rôle IAM. Ces politiques contrôlent quel type d’actions des utilisateurs et des rôles peuvent exécuter, sur quelles ressources et dans quelles conditions. Pour découvrir comment créer une politique basée sur l’identité, consultez Création de politiques IAM dans le Guide de l’utilisateur IAM.

Les politiques basées sur l’identité peuvent être classées comme des politiques en ligne ou des politiques gérées. Les politiques en ligne sont intégrées directement à un utilisateur, groupe ou rôle. Les politiques gérées sont des politiques autonomes que vous pouvez associer à plusieurs utilisateurs, groupes et rôles au sein de votre Compte AWS. Les politiques gérées incluent les politiques AWS gérées et les politiques gérées par le client. Pour découvrir comment choisir entre

une politique gérée et une politique en ligne, consultez Choix entre les politiques gérées et les politiques en ligne dans le Guide de l'utilisateur IAM.

politiques basées sur les ressources

Les politiques basées sur les ressources sont des documents de politique JSON que vous attachez à une ressource. Des politiques basées sur les ressources sont, par exemple, les politiques de confiance de rôle IAM et des politiques de compartiment. Dans les services qui sont compatibles avec les politiques basées sur les ressources, les administrateurs de service peuvent les utiliser pour contrôler l'accès à une ressource spécifique. Pour la ressource dans laquelle se trouve la politique, cette dernière définit quel type d'actions un principal spécifié peut effectuer sur cette ressource et dans quelles conditions. Vous devez spécifier un principal dans une politique basée sur les ressources. Les principaux peuvent inclure des comptes, des utilisateurs, des rôles, des utilisateurs fédérés ou. Services AWS

Les politiques basées sur les ressources sont des politiques en ligne situées dans ce service. Vous ne pouvez pas utiliser les politiques AWS gérées par IAM dans une stratégie basée sur les ressources.

Listes de contrôle d'accès (ACL)

Les listes de contrôle d'accès (ACL) vérifie quels principals (membres de compte, utilisateurs ou rôles) ont l'autorisation d'accéder à une ressource. Les listes de contrôle d'accès sont similaires aux politiques basées sur les ressources, bien qu'elles n'utilisent pas le format de document de politique JSON.

Amazon S3 et Amazon VPC sont des exemples de services qui prennent en charge les ACL. AWS WAF Pour en savoir plus sur les listes de contrôle d'accès, consultez Vue d'ensemble des listes de contrôle d'accès (ACL) dans le Guide du développeur Amazon Simple Storage Service.

Autres types de politique

AWS prend en charge d'autres types de politiques moins courants. Ces types de politiques peuvent définir le nombre maximum d'autorisations qui vous sont accordées par des types de politiques plus courants.

- Limite d'autorisations : une limite d'autorisations est une fonctionnalité avancée dans laquelle vous définissez le nombre maximal d'autorisations qu'une politique basée sur l'identité peut accorder à une entité IAM (utilisateur ou rôle IAM). Vous pouvez définir une limite d'autorisations pour

une entité. Les autorisations en résultant représentent la combinaison des politiques basées sur l'identité d'une entité et de ses limites d'autorisation. Les politiques basées sur les ressources qui spécifient l'utilisateur ou le rôle dans le champ `Principal` ne sont pas limitées par les limites d'autorisations. Un refus explicite dans l'une de ces politiques remplace l'autorisation. Pour plus d'informations sur les limites d'autorisations, consultez Limites d'autorisations pour des entités IAM dans le Guide de l'utilisateur IAM.

- Politiques de contrôle des services (SCP) — Les SCP sont des politiques JSON qui spécifient les autorisations maximales pour une organisation ou une unité organisationnelle (UO) dans. AWS Organizations AWS Organizations est un service permettant de regrouper et de gérer de manière centralisée Comptes AWS les multiples propriétés de votre entreprise. Si vous activez toutes les fonctionnalités d'une organisation, vous pouvez appliquer les politiques de contrôle des services (SCP) à l'un ou à l'ensemble de vos comptes. Le SCP limite les autorisations pour les entités figurant dans les comptes des membres, y compris chacune Utilisateur racine d'un compte AWS d'entre elles. Pour plus d'informations sur les organisations et les SCP, consultez Fonctionnement des SCP dans le Guide de l'utilisateur AWS Organizations .
- Politiques de séance : les politiques de séance sont des politiques avancées que vous utilisez en tant que paramètre lorsque vous créez par programmation une séance temporaire pour un rôle ou un utilisateur fédéré. Les autorisations de séance en résultant sont une combinaison des politiques basées sur l'identité de l'utilisateur ou du rôle et des politiques de séance. Les autorisations peuvent également provenir d'une politique basée sur les ressources. Un refus explicite dans l'une de ces politiques annule l'autorisation. Pour plus d'informations, consultez politiques de séance dans le Guide de l'utilisateur IAM.

Plusieurs types de politique

Lorsque plusieurs types de politiques s'appliquent à la requête, les autorisations en résultant sont plus compliquées à comprendre. Pour savoir comment AWS déterminer s'il faut autoriser une demande lorsque plusieurs types de politiques sont impliqués, consultez la section Logique d'évaluation des politiques dans le guide de l'utilisateur IAM.

Contrôle d'accès

Vous pouvez disposer d'informations d'identification valides pour authentifier vos demandes, mais vous ne pouvez pas créer de ressources Step Functions ou y accéder sans autorisation. Par exemple, vous devez être autorisé à invoquer AWS Lambda les cibles Amazon Simple Notification Service (Amazon SNS) et Amazon Simple Queue Service (Amazon SQS) associées à vos règles Step Functions.

Les sections suivantes décrivent comment gérer les autorisations pour Step Functions.

- Création d'un rôle IAM pour votre machine d'état
- Création d'autorisations IAM granulaires pour les utilisateurs non administrateurs
- Points de terminaison Amazon VPC pour Step Functions
- Politiques IAM pour les services intégrés
- Politiques IAM pour l'utilisation de l'état de la carte distribuée

Actions politiques pour Step Functions

Prend en charge les actions de politique	Oui

Les administrateurs peuvent utiliser les politiques AWS JSON pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

L'élément `Action` d'une politique JSON décrit les actions que vous pouvez utiliser pour autoriser ou refuser l'accès à une politique. Les actions de stratégie portent généralement le même nom que l'opération AWS d'API associée. Il existe quelques exceptions, telles que les actions avec autorisations uniquement qui n'ont pas d'opération API correspondante. Certaines opérations nécessitent également plusieurs actions dans une politique. Ces actions supplémentaires sont nommées actions dépendantes.

Intégration d'actions dans une stratégie afin d'accorder l'autorisation d'exécuter les opérations associées.

Pour consulter la liste des Step Functions actions, voir Ressources définies par AWS Step Functions dans la référence d'autorisation de service.

Les actions de politique en Step Functions cours utilisent le préfixe suivant avant l'action :

```
states
```

Pour indiquer plusieurs actions dans une seule déclaration, séparez-les par des virgules.

```
"Action": [
```

```
      "states:action1",
      "states:action2"
         ]
```

Pour consulter des exemples de politiques Step Functions basées sur l'identité, consultez. Exemples de politiques basées sur l'identité pour AWS Step Functions

Ressources politiques pour Step Functions

Prend en charge les ressources de politique	Oui

Les administrateurs peuvent utiliser les politiques AWS JSON pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

L'élément de politique JSON `Resource` indique le ou les objets auxquels l'action s'applique. Les instructions doivent inclure un élément `Resource` ou `NotResource`. Il est recommandé de définir une ressource à l'aide de son Amazon Resource Name (ARN). Vous pouvez le faire pour des actions qui prennent en charge un type de ressource spécifique, connu sous la dénomination autorisations de niveau ressource.

Pour les actions qui ne sont pas compatibles avec les autorisations de niveau ressource, telles que les opérations de liste, utilisez un caractère générique (*) afin d'indiquer que l'instruction s'applique à toutes les ressources.

```
"Resource": "*"
```

Pour consulter la liste des types de Step Functions ressources et de leurs ARN, consultez la section Actions définies par AWS Step Functions dans la référence d'autorisation de service. Pour savoir avec quelles actions vous pouvez spécifier l'ARN de chaque ressource, voir Ressources définies par AWS Step Functions.

Pour consulter des exemples de politiques Step Functions basées sur l'identité, consultez. Exemples de politiques basées sur l'identité pour AWS Step Functions

Clés de conditions de politique pour Step Functions

Prend en charge les clés de condition de politique spécifiques au service	Oui

Les administrateurs peuvent utiliser les politiques AWS JSON pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

L'élément `Condition` (ou le bloc `Condition`) vous permet de spécifier des conditions lorsqu'une instruction est appliquée. L'élément `Condition` est facultatif. Vous pouvez créer des expressions conditionnelles qui utilisent des opérateurs de condition, tels que les signes égal ou inférieur à, pour faire correspondre la condition de la politique aux valeurs de la demande.

Si vous spécifiez plusieurs éléments `Condition` dans une instruction, ou plusieurs clés dans un seul élément `Condition`, AWS les évalue à l'aide d'une opération `AND` logique. Si vous spécifiez plusieurs valeurs pour une seule clé de condition, AWS évalue la condition à l'aide d'une `OR` opération logique. Toutes les conditions doivent être remplies avant que les autorisations associées à l'instruction ne soient accordées.

Vous pouvez aussi utiliser des variables d'espace réservé quand vous spécifiez des conditions. Par exemple, vous pouvez accorder à un utilisateur IAM l'autorisation d'accéder à une ressource uniquement si elle est balisée avec son nom d'utilisateur IAM. Pour plus d'informations, consultez Éléments d'une politique IAM : variables et identifications dans le Guide de l'utilisateur IAM.

AWS prend en charge les clés de condition globales et les clés de condition spécifiques au service. Pour voir toutes les clés de condition AWS globales, voir les clés de contexte de condition AWS globales dans le guide de l'utilisateur IAM.

Pour consulter la liste des clés de Step Functions condition, voir Clés de condition pour AWS Step Functions la référence d'autorisation de service. Pour savoir avec quelles actions et ressources vous pouvez utiliser une clé de condition, voir Ressources définies par AWS Step Functions.

Pour consulter des exemples de politiques Step Functions basées sur l'identité, consultez. Exemples de politiques basées sur l'identité pour AWS Step Functions

ACL dans Step Functions

Prend en charge les listes ACL	Non

Les listes de contrôle d'accès (ACL) vérifient quels principaux (membres de compte, utilisateurs ou rôles) ont l'autorisation d'accéder à une ressource. Les listes de contrôle d'accès sont similaires aux politiques basées sur les ressources, bien qu'elles n'utilisent pas le format de document de politique JSON.

ABAC avec Step Functions

Prise en charge d'ABAC (identifications dans les politiques)	Partielle

Le contrôle d'accès basé sur les attributs (ABAC) est une politique d'autorisation qui définit des autorisations en fonction des attributs. Dans AWS, ces attributs sont appelés balises. Vous pouvez associer des balises aux entités IAM (utilisateurs ou rôles) et à de nombreuses AWS ressources. L'étiquetage des entités et des ressources est la première étape d'ABAC. Vous concevez ensuite des politiques ABAC pour autoriser des opérations quand l'identification du principal correspond à celle de la ressource à laquelle il tente d'accéder.

L'ABAC est utile dans les environnements qui connaissent une croissance rapide et pour les cas où la gestion des politiques devient fastidieuse.

Pour contrôler l'accès basé sur des étiquettes, vous devez fournir les informations d'étiquette dans l'élément de condition d'une politique utilisant les clés de condition `aws:ResourceTag/key-name`, `aws:RequestTag/key-name` ou `aws:TagKeys`.

Si un service prend en charge les trois clés de condition pour tous les types de ressources, alors la valeur pour ce service est Oui. Si un service prend en charge les trois clés de condition pour certains types de ressources uniquement, la valeur est Partielle.

Pour plus d'informations sur l'ABAC, consultez Qu'est-ce que le contrôle d'accès basé sur les attributs (ABAC) ? dans le Guide de l'utilisateur IAM. Pour accéder à un didacticiel décrivant les étapes de configuration de l'ABAC, consultez Utilisation du contrôle d'accès par attributs (ABAC) dans le Guide de l'utilisateur IAM.

Utilisation d'informations d'identification temporaires avec Step Functions

Prend en charge les informations d’identification temporaires	Oui

Certains Services AWS ne fonctionnent pas lorsque vous vous connectez à l'aide d'informations d'identification temporaires. Pour plus d'informations, y compris celles qui Services AWS fonctionnent avec des informations d'identification temporaires, consultez Services AWS la section relative à l'utilisation d'IAM dans le guide de l'utilisateur d'IAM.

Vous utilisez des informations d'identification temporaires si vous vous connectez à l' AWS Management Console aide d'une méthode autre qu'un nom d'utilisateur et un mot de passe. Par exemple, lorsque vous accédez à AWS l'aide du lien d'authentification unique (SSO) de votre entreprise, ce processus crée automatiquement des informations d'identification temporaires. Vous créez également automatiquement des informations d’identification temporaires lorsque vous vous connectez à la console en tant qu’utilisateur, puis changez de rôle. Pour plus d’informations sur le changement de rôle, consultez Changement de rôle (console) dans le Guide de l’utilisateur IAM.

Vous pouvez créer manuellement des informations d'identification temporaires à l'aide de l' AWS API AWS CLI or. Vous pouvez ensuite utiliser ces informations d'identification temporaires pour y accéder AWS. AWS recommande de générer dynamiquement des informations d'identification temporaires au lieu d'utiliser des clés d'accès à long terme. Pour plus d’informations, consultez Informations d’identification de sécurité temporaires dans IAM.

Autorisations principales interservices pour Step Functions

Prend en charge les sessions d'accès direct (FAS)	Oui

Lorsque vous utilisez un utilisateur ou un rôle IAM pour effectuer des actions AWS, vous êtes considéré comme un mandant. Lorsque vous utilisez certains services, vous pouvez effectuer une action qui initie une autre action dans un autre service. FAS utilise les autorisations du principal appelant et Service AWS, associées Service AWS à la demande, pour adresser des demandes aux services en aval. Les demandes FAS ne sont effectuées que lorsqu'un service reçoit une demande qui nécessite des interactions avec d'autres personnes Services AWS ou des ressources pour

être traitée. Dans ce cas, vous devez disposer d'autorisations nécessaires pour effectuer les deux actions. Pour plus de détails sur une politique lors de la formulation de demandes FAS, consultez Transmission des sessions d'accès.

Fonctions du service pour Step Functions

Prend en charge les fonctions du service	Oui

Une fonction de service est un rôle IAM qu'un service endosse pour accomplir des actions en votre nom. Un administrateur IAM peut créer, modifier et supprimer une fonction du service à partir d'IAM. Pour plus d'informations, consultez Création d'un rôle pour la délégation d'autorisations à un Service AWS dans le Guide de l'utilisateur IAM.

> **Warning**
>
> La modification des autorisations associées à un rôle de service peut perturber Step Functions les fonctionnalités. Modifiez les rôles de service uniquement lorsque Step Functions vous recevez des instructions à cet effet.

Rôles liés à un service pour Step Functions

Prend en charge les rôles liés à un service	Non

Un rôle lié à un service est un type de rôle de service lié à un. Service AWS Le service peut endosser le rôle afin d'effectuer une action en votre nom. Les rôles liés à un service apparaissent dans votre Compte AWS répertoire et appartiennent au service. Un administrateur IAM peut consulter, mais ne peut pas modifier, les autorisations concernant les rôles liés à un service.

Pour plus d'informations sur la création ou la gestion des rôles liés à un service, consultez Services AWS qui fonctionnent avec IAM. Recherchez un service dans le tableau qui inclut un `Yes` dans la colonne Rôle lié à un service. Choisissez le lien Oui pour consulter la documentation du rôle lié à ce service.

Comment AWS Step Functions fonctionne avec IAM

Avant d'utiliser IAM pour gérer l'accès à Step Functions, découvrez les fonctionnalités IAM disponibles. Step Functions

Fonctionnalités IAM que vous pouvez utiliser avec AWS Step Functions

Fonction IAM	Step Functions soutien
Politiques basées sur l’identité	Oui
Politiques basées sur les ressources	Non
Actions de politique	Oui
Ressources de politique	Oui
Clés de condition de politique (spécifiques au service)	Oui
ACL	Non
ABAC (identifications dans les politiques)	Partielle
Informations d’identification temporaires	Oui
Autorisations de principal	Oui
Fonctions de service	Oui
Rôles liés à un service	Non

Pour obtenir une vue d'ensemble de la façon dont Step Functions les autres AWS services fonctionnent avec la plupart des fonctionnalités IAM, consultez la section AWS Services compatibles avec IAM dans le Guide de l'utilisateur IAM.

Exemples de politiques basées sur l'identité pour AWS Step Functions

Par défaut, les utilisateurs et les rôles ne sont pas autorisés à créer ou modifier les ressources Step Functions . Ils ne peuvent pas non plus effectuer de tâches à l'aide de l'API AWS Management Console, AWS Command Line Interface (AWS CLI) ou de AWS l'API. Pour octroyer aux utilisateurs

des autorisations d'effectuer des actions sur les ressources dont ils ont besoin, un administrateur IAM peut créer des politiques IAM. L'administrateur peut ensuite ajouter les politiques IAM aux rôles et les utilisateurs peuvent assumer les rôles.

Pour apprendre à créer une politique basée sur l'identité IAM à l'aide de ces exemples de documents de politique JSON, consultez Création de politiques dans l'onglet JSON dans le Guide de l'utilisateur IAM.

Pour plus de détails sur les actions et les types de ressources définis par Step Functions, y compris le format des ARN pour chacun des types de ressources, voir Actions, ressources et clés de condition AWS Step Functions dans la référence d'autorisation de service.

Rubriques

- Bonnes pratiques en matière de politiques
- Utilisation de la console Step Functions
- Autorisation accordée aux utilisateurs pour afficher leurs propres autorisations

Bonnes pratiques en matière de politiques

Les politiques basées sur l'identité déterminent si quelqu'un peut créer, accéder ou supprimer Step Functions des ressources dans votre compte. Ces actions peuvent entraîner des frais pour votre Compte AWS. Lorsque vous créez ou modifiez des politiques basées sur l'identité, suivez ces instructions et recommandations :

- Commencez AWS par les politiques gérées et passez aux autorisations du moindre privilège : pour commencer à accorder des autorisations à vos utilisateurs et à vos charges de travail, utilisez les politiques AWS gérées qui accordent des autorisations pour de nombreux cas d'utilisation courants. Ils sont disponibles dans votre Compte AWS. Nous vous recommandons de réduire davantage les autorisations en définissant des politiques gérées par les AWS clients spécifiques à vos cas d'utilisation. Pour plus d'informations, consultez politiques gérées par AWS ou politiques gérées par AWS pour les activités professionnelles dans le Guide de l'utilisateur IAM.
- Accorder les autorisations de moindre privilège : lorsque vous définissez des autorisations avec des politiques IAM, accordez uniquement les autorisations nécessaires à l'exécution d'une seule tâche. Pour ce faire, vous définissez les actions qui peuvent être entreprises sur des ressources spécifiques dans des conditions spécifiques, également appelées autorisations de moindre privilège. Pour plus d'informations sur l'utilisation de IAM pour appliquer des autorisations, consultez politiques et autorisations dans IAM dans le Guide de l'utilisateur IAM.

- Utiliser des conditions dans les politiques IAM pour restreindre davantage l'accès : vous pouvez ajouter une condition à vos politiques afin de limiter l'accès aux actions et aux ressources. Par exemple, vous pouvez écrire une condition de politique pour spécifier que toutes les demandes doivent être envoyées via SSL. Vous pouvez également utiliser des conditions pour accorder l'accès aux actions de service si elles sont utilisées par le biais d'un service spécifique Service AWS, tel que AWS CloudFormation. Pour plus d'informations, consultez Conditions pour éléments de politique JSON IAM dans le Guide de l'utilisateur IAM.
- Utilisez IAM Access Analyzer pour valider vos politiques IAM afin de garantir des autorisations sécurisées et fonctionnelles : IAM Access Analyzer valide les politiques nouvelles et existantes de manière à ce que les politiques IAM respectent le langage de politique IAM (JSON) et les bonnes pratiques IAM. IAM Access Analyzer fournit plus de 100 vérifications de politiques et des recommandations exploitables pour vous aider à créer des politiques sécurisées et fonctionnelles. Pour plus d'informations, consultez Validation de politique IAM Access Analyzer dans le Guide de l'utilisateur IAM.
- Exiger l'authentification multifactorielle (MFA) : si vous avez un scénario qui nécessite des utilisateurs IAM ou un utilisateur root, activez l'authentification MFA pour une sécurité accrue. Compte AWS Pour exiger le MFA lorsque des opérations d'API sont appelées, ajoutez des conditions MFA à vos politiques. Pour plus d'informations, consultez Configuration de l'accès aux API protégé par MFA dans le Guide de l'utilisateur IAM.

Pour plus d'informations sur les bonnes pratiques dans IAM, consultez Bonnes pratiques de sécurité dans IAM dans le Guide de l'utilisateur IAM.

Utilisation de la console Step Functions

Pour accéder à la AWS Step Functions console, vous devez disposer d'un ensemble minimal d'autorisations. Ces autorisations doivent vous permettre de répertorier et d'afficher les détails Step Functions des ressources de votre Compte AWS. Si vous créez une stratégie basée sur l'identité qui est plus restrictive que l'ensemble minimum d'autorisations requis, la console ne fonctionnera pas comme prévu pour les entités (utilisateurs ou rôles) tributaires de cette stratégie.

Il n'est pas nécessaire d'accorder des autorisations de console minimales aux utilisateurs qui appellent uniquement l'API AWS CLI ou l' AWS API. Autorisez plutôt l'accès à uniquement aux actions qui correspondent à l'opération d'API qu'ils tentent d'effectuer.

Pour garantir que les utilisateurs et les rôles peuvent toujours utiliser la Step Functions console, associez également la politique Step Functions `ConsoleAccess` ou la politique `ReadOnly` AWS

gérée aux entités. Pour plus d'informations, consultez Ajout d'autorisations à un utilisateur dans le Guide de l'utilisateur IAM.

Autorisation accordée aux utilisateurs pour afficher leurs propres autorisations

Cet exemple montre comment créer une politique qui permet aux utilisateurs IAM d'afficher les politiques en ligne et gérées attachées à leur identité d'utilisateur. Cette politique inclut les autorisations permettant d'effectuer cette action sur la console ou par programmation à l'aide de l'API AWS CLI or AWS .

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Politiques basées sur l'identité pour Step Functions

Prend en charge les politiques basées sur l'identité	Oui

Les politiques basées sur l'identité sont des documents de politique d'autorisations JSON que vous pouvez attacher à une identité telle qu'un utilisateur, un groupe d'utilisateurs ou un rôle IAM. Ces politiques contrôlent quel type d'actions des utilisateurs et des rôles peuvent exécuter, sur quelles ressources et dans quelles conditions. Pour découvrir comment créer une politique basée sur l'identité, consultez Création de politiques IAM dans le Guide de l'utilisateur IAM.

Avec les politiques IAM basées sur l'identité, vous pouvez spécifier des actions et ressources autorisées ou refusées, ainsi que les conditions dans lesquelles les actions sont autorisées ou refusées. Vous ne pouvez pas spécifier le principal dans une politique basée sur une identité car celle-ci s'applique à l'utilisateur ou au rôle auquel elle est attachée. Pour découvrir tous les éléments que vous utilisez dans une politique JSON, consultez Références des éléments de politique JSON IAM dans le Guide de l'utilisateur IAM.

Exemples de politiques basées sur l'identité pour Step Functions

Pour consulter des exemples de politiques Step Functions basées sur l'identité, consultez. Exemples de politiques basées sur l'identité pour AWS Step Functions

Politiques basées sur les ressources au sein de Step Functions

Prend en charge les politiques basées sur les ressources	Non

Les politiques basées sur les ressources sont des documents de politique JSON que vous attachez à une ressource. Des politiques basées sur les ressources sont, par exemple, les politiques de confiance de rôle IAM et des politiques de compartiment. Dans les services qui sont compatibles avec les politiques basées sur les ressources, les administrateurs de service peuvent les utiliser pour contrôler l'accès à une ressource spécifique. Pour la ressource dans laquelle se trouve la politique, cette dernière définit quel type d'actions un principal spécifié peut effectuer sur cette ressource et dans quelles conditions. Vous devez spécifier un principal dans une politique basée sur les

ressources. Les principaux peuvent inclure des comptes, des utilisateurs, des rôles, des utilisateurs fédérés ou. Services AWS

Pour permettre un accès intercompte, vous pouvez spécifier un compte entier ou des entités IAM dans un autre compte en tant que principal dans une politique basée sur les ressources. L'ajout d'un principal entre comptes à une politique basée sur les ressources ne représente qu'une partie de l'instauration de la relation d'approbation. Lorsque le principal et la ressource sont différents Comptes AWS, un administrateur IAM du compte sécurisé doit également accorder à l'entité principale (utilisateur ou rôle) l'autorisation d'accéder à la ressource. Pour ce faire, il attache une politique basée sur une identité à l'entité. Toutefois, si une politique basée sur des ressources accorde l'accès à un principal dans le même compte, aucune autre politique basée sur l'identité n'est requise. Pour plus d'informations, consultez Différence entre les rôles IAM et les politiques basées sur une ressource dans le Guide de l'utilisateur IAM.

AWS politiques gérées pour AWS Step Functions

Une politique AWS gérée est une politique autonome créée et administrée par AWS. AWS les politiques gérées sont conçues pour fournir des autorisations pour de nombreux cas d'utilisation courants afin que vous puissiez commencer à attribuer des autorisations aux utilisateurs, aux groupes et aux rôles.

N'oubliez pas que les politiques AWS gérées peuvent ne pas accorder d'autorisations de moindre privilège pour vos cas d'utilisation spécifiques, car elles sont accessibles à tous les AWS clients. Nous vous recommandons de réduire encore les autorisations en définissant des politiques gérées par le client qui sont propres à vos cas d'utilisation.

Vous ne pouvez pas modifier les autorisations définies dans les politiques AWS gérées. Si les autorisations définies dans une politique AWS gérée sont AWS mises à jour, la mise à jour affecte toutes les identités principales (utilisateurs, groupes et rôles) auxquelles la politique est attachée. AWS est le plus susceptible de mettre à jour une politique AWS gérée lorsqu'une nouvelle politique Service AWS est lancée ou lorsque de nouvelles opérations d'API sont disponibles pour les services existants.

Pour plus d'informations, consultez la section Politiques gérées par AWS dans le Guide de l'utilisateur IAM.

AWS politique gérée : AWSStepFunctionsConsoleFullAccess

Vous pouvez associer la politique `AWSStepFunctionsConsoleFullAccess` à vos identités IAM.

Cette politique accorde des autorisations d'*`administrateur`* qui permettent à un utilisateur d'accéder à la console Step Functions. Pour bénéficier d'une expérience de console complète, un utilisateur peut également avoir besoin de PassRole l'autorisation iam : pour les autres rôles IAM pouvant être assumés par le service.

AWS politique gérée : AWSStepFunctionsReadOnlyAccess

Vous pouvez associer la politique `AWSStepFunctionsReadOnlyAccess` à vos identités IAM.

Cette politique accorde des autorisations *`en lecture seule`* qui permettent à un utilisateur ou à un rôle de répertorier et de décrire les machines à états, les activités, les exécutions, les activités, les balises MapRuns, ainsi que les alias et versions des machines à états. Cette politique accorde également l'autorisation de vérifier la syntaxe des définitions de machines à états que vous fournissez.

AWS politique gérée : AWSStepFunctionsFullAccess

Vous pouvez associer la politique `AWSStepFunctionsFullAccess` à vos identités IAM.

Cette politique accorde des autorisations *`complètes`* à un utilisateur ou à un rôle pour utiliser l'API Step Functions. Pour bénéficier d'un accès complet, un utilisateur doit disposer de PassRole l'autorisation *`iam :`* sur au moins un rôle IAM pouvant être assumé par le service.

Step Functions mises à jour des politiques AWS gérées

Consultez les détails des mises à jour des politiques AWS gérées Step Functions depuis que ce service a commencé à suivre ces modifications. Pour obtenir des alertes automatiques sur les modifications apportées à cette page, abonnez-vous au flux RSS de la page Step Functions Historique du document.

Modification	Description	Date
AWSStepFunctionsReadOnlyAccess – Mise à jour d'une stratégie existante	Step Functions a ajouté de nouvelles autorisations pour permettre d'appeler une action d'`states:ValidateSta`	25 avril 2024

Modification	Description	Date
	`teMachineDefinitio` n API afin de vérifier la syntaxe des définitions de machines à états que vous fournissez.	
AWSStepFunctionsReadOnlyAccess – Mise à jour d'une politique existante	Step Functions a ajouté de nouvelles autorisations pour permettre de répertorier et de lire les données relatives à : les balises (ListTagsForResource), les cartes distribuées (ListMapRuns, DescribeMapRun), les versions et les alias (DescribeStateMachineAlias, ListStateMachineAliases, ListStateMachineVersions).	2 avril 2024
Step Functions a commencé à suivre les modifications	Step Functions a commencé à suivre les modifications apportées AWS à ses politiques gérées.	2 avril 2024

Création d'un rôle IAM pour votre machine d'état

AWS Step Functions peut exécuter du code et accéder à AWS des ressources (par exemple en invoquant une AWS Lambda fonction). Pour maintenir la sécurité, vous devez accorder à Step Functions l'accès à ces ressources en utilisant un rôle IAM.

Ce guide vous permet de tirer parti des rôles IAM générés automatiquement qui sont valides pour la AWS région dans laquelle vous créez la machine à états. Tutoriels pour Step Functions Cependant, vous pouvez créer votre propre rôle IAM pour une machine à états.

Lorsque vous créez une politique IAM à utiliser par vos machines d'état, la politique doit inclure les autorisations que vous souhaitez que les machines d'état assument. Vous pouvez utiliser une

politique AWS gérée existante comme exemple ou vous pouvez créer une politique personnalisée à partir de zéro qui répond à vos besoins spécifiques. Pour plus d'informations, consultez la section Création de politiques IAM dans le Guide de l'utilisateur IAM

Pour créer votre propre rôle IAM pour une machine à états, suivez les étapes décrites dans cette section.

Dans cet exemple, vous créez un rôle IAM autorisé à appeler une fonction Lambda.

Création d'un rôle pour Step Functions

1. Connectez-vous à la console IAM, puis sélectionnez Rôles, Créer un rôle.
2. Sur la page Select trusted entity, sous AWS service, sélectionnez Step Functions dans la liste, puis choisissez Next : Permissions.
3. Dans la page Stratégie d'autorisations attachée, choisissez Suivant : Révision.
4. Sur la page Review (Révision), entrez `StepFunctionsLambdaRole` pour Role Name (Nom du rôle), puis choisissez Create role (Créer un rôle).

   Le rôle IAM apparaît dans la liste des rôles.

Pour plus d'informations sur les autorisations et les politiques IAM, consultez la section Gestion des accès dans le guide de l'utilisateur IAM.

Prévenir le problème des adjoints confus entre les services

Le problème de député confus est un problème de sécurité dans lequel une entité qui n’est pas autorisée à effectuer une action peut contraindre une entité plus privilégiée à le faire. En AWS, l'usurpation d'identité interservices peut entraîner la confusion des adjoints. L’usurpation d’identité entre services peut se produire lorsqu’un service (le service appelant) appelle un autre service (le service appelé). Ce type d’usurpation d’identité peut se produire entre comptes et entre services. Le service appelant peut être manipulé et ses autorisations utilisées pour agir sur les ressources d’un autre client auxquelles on ne serait pas autorisé d’accéder autrement.

Pour éviter toute confusion chez les adjoints, AWS fournit des outils qui vous aident à protéger vos données pour tous les services, les responsables de service ayant accès aux ressources de votre compte. Cette section se concentre sur la prévention de la confusion entre les services, en particulier à AWS Step Functions ; toutefois, vous pouvez en savoir plus à ce sujet dans la section du guide de l'utilisateur de l'IAM consacrée au problème de confusion chez les adjoints.

Nous vous recommandons d'utiliser les clés contextuelles de condition `aws:SourceAccount`globale `aws:SourceArn`et les clés contextuelles dans les politiques de ressources afin de limiter les Step Functions autorisations accordées à un autre service pour accéder à vos ressources. Utilisez `aws:SourceArn` si vous souhaitez qu'une seule ressource soit associée à l'accès entre services. Utilisez `aws:SourceAccount` si vous souhaitez autoriser l'association d'une ressource de ce compte à l'utilisation interservices.

Le moyen le plus efficace de se protéger contre le problème de député confus consiste à utiliser la clé de contexte de condition globale `aws:SourceArn` avec l'ARN complet de la ressource. Si vous ne connaissez pas l'ARN complet de la ressource, ou si vous spécifiez plusieurs ressources, utilisez la clé de condition contextuelle `aws:SourceArn` globale avec des caractères génériques (*) pour les parties inconnues de l'ARN. Par exemple, `arn:aws:states:*:111122223333:*`.

Voici un exemple de politique fiable qui montre comment utiliser `aws:SourceArn` et `aws:SourceAccount` avec Step Functions pour éviter la confusion liée aux adjoints.

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

Attacher une stratégie en ligne

Step Functions peut contrôler d'autres services directement dans un `Task` état. Ajoutez des politiques intégrées pour permettre à Step Functions d'accéder aux actions d'API des services que vous devez contrôler.

1. Ouvrez la console IAM, choisissez Roles, recherchez votre rôle Step Functions, puis sélectionnez ce rôle.
2. Sélectionnez Add inline policy (Ajouter une stratégie en ligne).
3. Utilisez l'Éditeur visuel dans l'onglet JSON pour créer des stratégies pour votre rôle.

Pour plus d'informations sur la manière de AWS Step Functions contrôler d'autres AWS services, consultezUtilisation AWS Step Functions avec d'autres services.

> **Note**
>
> Pour des exemples de politiques IAM créées par la console Step Functions, consultezPolitiques IAM pour les services intégrés.

Création d'autorisations IAM granulaires pour les utilisateurs non administrateurs

Les politiques gérées par défaut dans IAM, par exemple`ReadOnly`, ne couvrent pas entièrement tous les types d' AWS Step Functions autorisations. Cette section décrit ces différents types d'autorisations et fournit des exemples des configurations.

Step Functions propose quatre catégories d'autorisations. En fonction du type d'accès que vous souhaitez fournir à un utilisateur, vous pouvez contrôler l'accès en utilisant ces catégories d'autorisations.

Autorisations de niveau service

S'applique aux composants de l'API qui n'agissent pas sur une ressource spécifique.

Autorisations de niveau machine d'état

S'applique à tous les composants d'API qui agissent sur une machine d'état spécifique.

Autorisations de niveau exécution

S'applique à tous les composants d'API qui agissent sur une exécution spécifique.

Autorisations de niveau activité

S'applique à tous les composants d'API qui agissent sur une activité spécifique ou sur une instance d'activité particulière.

Autorisations de niveau service

Ce niveau d'autorisation s'applique à toutes les actions d'API qui n'agissent pas sur une ressource spécifique. Ceux-ci incluentCreateStateMachine, CreateActivityListStateMachines,ListActivities, etValidationStateMachineDefinition.

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

```
}
```

Autorisations de niveau machine d'état

Ce niveau d'autorisation s'applique à toutes les actions d'API qui agissent sur une machine d'état spécifique. Ces opérations d'API nécessitent l'Amazon Resource Name (ARN) de la machine d'état dans le cadre de la demande[DeleteStateMachine](), tel que[DescribeStateMachine](),[StartExecution](), et[ListExecutions]().

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:DescribeStateMachine",
        "states:StartExecution",
        "states:DeleteStateMachine",
        "states:ListExecutions",
        "states:UpdateStateMachine",
        "states:TestState",
        "states:RevealSecrets"
      ],
      "Resource": [
        "arn:aws:states:*:*:stateMachine:StateMachinePrefix*"
      ]
    }
  ]
}
```

Autorisations de niveau exécution

Ce niveau d'autorisation s'applique à toutes les actions d'API qui agissent sur une exécution spécifique. Ces actions d'API requièrent l'ARN de l'exécution dans la demande, comme par exemple [DescribeExecution](), [GetExecutionHistory]() et [StopExecution]().

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```
      "Action": [
        "states:DescribeExecution",
        "states:DescribeStateMachineForExecution",
        "states:GetExecutionHistory",
        "states:StopExecution"
      ],
      "Resource": [
        "arn:aws:states:*:*:execution:*:ExecutionPrefix*"
      ]
    }
  ]
}
```

Autorisations de niveau activité

Ce niveau d'autorisation s'applique à toutes les actions d'API qui agissent sur une activité spécifique ou sur une instance particulière de celle-ci. Ces opérations d'API nécessitent l'ARN de l'activité ou le jeton de l'instance dans le cadre de la demande, tel que DeleteActivityDescribeActivity,GetActivityTask, etSendTaskHeartbeat.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:DescribeActivity",
        "states:DeleteActivity",
        "states:GetActivityTask",
        "states:SendTaskHeartbeat"
      ],
      "Resource": [
        "arn:aws:states:*:*:activity:ActivityPrefix*"
      ]
    }
  ]
}
```

Accès à des ressources Comptes AWS dans d'autres flux de travail

Step Functions fournit un accès entre comptes aux ressources configurées selon différents flux Comptes AWS de travail. Grâce aux intégrations de services Step Functions, vous pouvez invoquer

n'importe quelle AWS ressource entre comptes, même si celle-ci Service AWS ne prend pas en charge les politiques basées sur les ressources ou les appels entre comptes.

Supposons, par exemple, que vous en Comptes AWS possédiez deux, appelées Développement et Tests, dans la même entité Région AWS. Grâce à l'accès entre comptes, votre flux de travail dans le compte de développement peut accéder à des ressources, telles que les compartiments Amazon S3, les tables Amazon DynamoDB et les fonctions Lambda disponibles dans le compte de test.

> **Important**
>
> Les politiques IAM basées sur les ressources et les rôles ne délèguent l'accès entre les comptes qu'au sein d'une seule partition. Par exemple, supposons que vous avez un compte dans la région USA Ouest (Californie du Nord) sur la partition `aws` standard. Vous avez également un compte dans la région Chine (Beijing) sur la partition `aws-cn`. Vous ne pouvez pas utiliser une politique basée sur les ressources d'Amazon S3 dans votre compte en Chine (Beijing) pour autoriser l'accès aux utilisateurs de votre compte `aws` standard.

Pour plus d'informations sur l'accès entre comptes, consultez la section Logique d'évaluation des politiques entre comptes dans le guide de l'utilisateur IAM.

Bien que chacun Compte AWS conserve un contrôle total sur ses propres ressources, Step Functions vous permet de réorganiser, d'échanger, d'ajouter ou de supprimer des étapes dans vos flux de travail sans avoir à personnaliser le moindre code. Vous pouvez le faire même si les processus changent ou que les applications évoluent.

Vous pouvez également invoquer des exécutions de machines à états imbriqués afin qu'elles soient disponibles sur différents comptes. Cela permet de séparer et d'isoler efficacement vos flux de travail. Lorsque vous utilisez le modèle d'intégration des `.sync`services dans vos flux de travail qui accèdent à un autre flux de travail Step Functions dans un autre compte, Step Functions utilise un sondage qui consomme le quota qui vous a été assigné. Pour plus d'informations, consultez Exécuter une tâche (.sync).

> **Note**
>
> Actuellement, l'intégration du AWS SDK entre régions et l'accès aux AWS ressources entre régions ne sont pas disponibles dans Step Functions.

Table des matières

Concepts clés de cette rubrique

Rôle d'exécution

Rôle IAM utilisé par Step Functions pour exécuter du code et accéder à AWS des ressources, telles que l'action Invoke de la AWS Lambda fonction.

Intégration des services

Les actions de l'API d'intégration du AWS SDK qui peuvent être appelées depuis un `Task` état dans vos flux de travail.

compte source

Celui Compte AWS qui possède la machine d'état et qui a commencé son exécution.

compte cible

Et Compte AWS vers lequel vous passez des appels entre comptes.

rôle cible

Rôle IAM dans le compte cible que la machine d'état assume pour effectuer des appels aux ressources détenues par le compte cible.

Exécuter un Job (.sync)

Modèle d'intégration de services utilisé pour appeler des services, tels que AWS Batch. Cela oblige également une machine à états Step Functions à attendre la fin d'une tâche avant de passer à l'état suivant. Pour indiquer que Step Functions doit attendre, ajoutez le `.sync` suffixe dans le `Resource` champ de votre définition `Task` d'état.

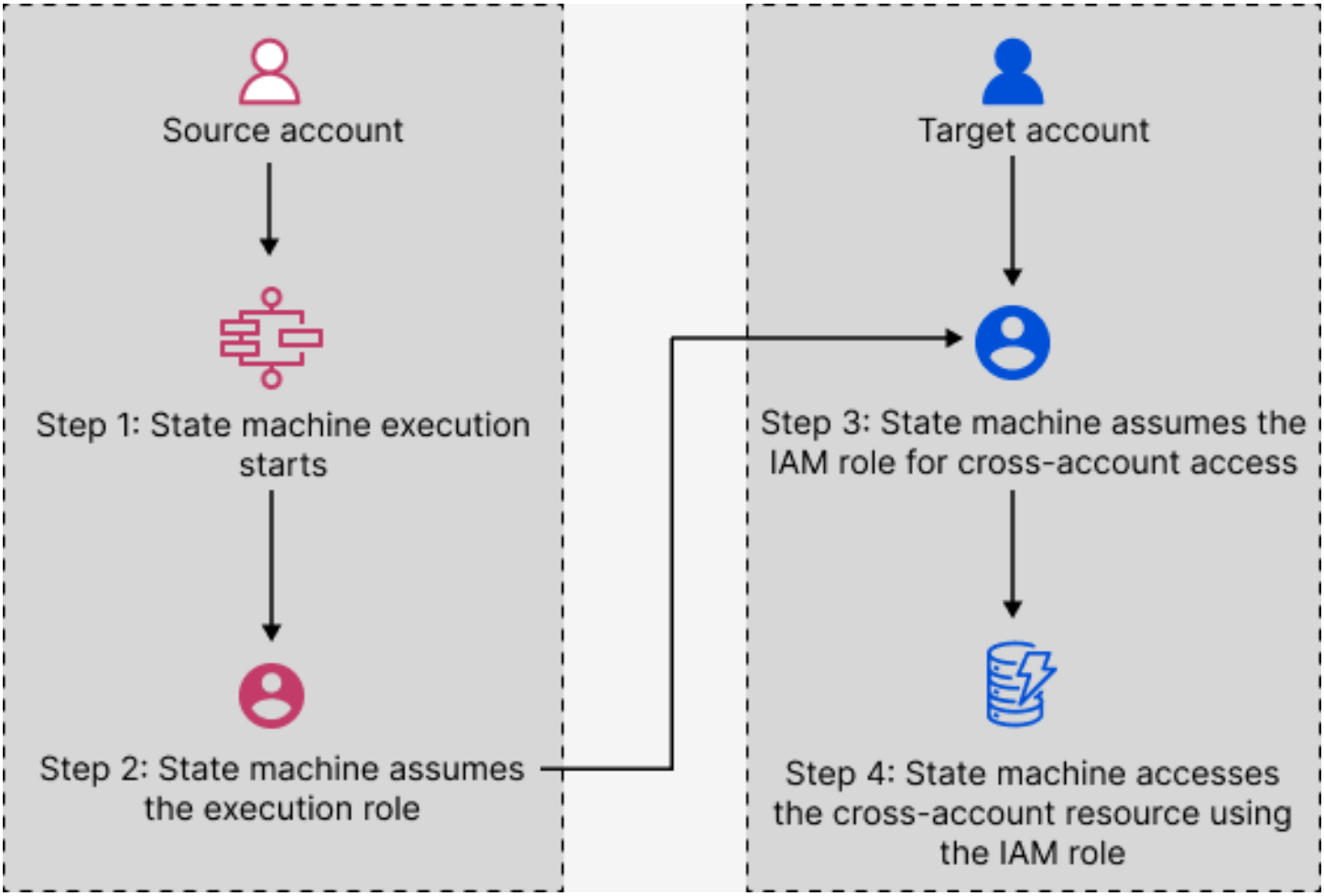
Invoquer des ressources entre comptes

Pour invoquer une ressource multi-comptes dans vos flux de travail, procédez comme suit :

1. Créez un rôle IAM dans le compte cible qui contient la ressource. Ce rôle accorde au compte source, contenant la machine d'état, l'autorisation d'accéder aux ressources du compte cible.
2. Dans la définition de `Task` l'état, spécifiez le rôle IAM cible à assumer par la machine d'état avant d'appeler la ressource entre comptes.
3. Modifiez la politique de confiance dans le rôle IAM cible pour permettre au compte source d'assumer temporairement ce rôle. La politique de confiance doit inclure l'Amazon Resource Name (ARN) de la machine d'état définie dans le compte source. Définissez également les autorisations appropriées dans le rôle IAM cible pour appeler la AWS ressource.
4. Mettez à jour le rôle d'exécution du compte source pour inclure l'autorisation requise pour assumer le rôle IAM cible.

Pour obtenir un exemple, consultez Tutoriel : Accès aux ressources multicomptes AWS.

> **Note**
>
> Vous pouvez configurer votre machine d'état pour qu'elle assume un rôle IAM afin d'accéder à des ressources à partir de plusieurs Comptes AWS. Cependant, une machine d'état ne peut assumer qu'un seul rôle IAM à la fois.



Tutoriel : Accès aux ressources multicomptes AWS

Grâce à la prise en charge de l'accès entre comptes dans Step Functions, vous pouvez partager des ressources configurées de différentes Comptes AWS manières. Dans ce didacticiel, nous vous expliquons comment accéder à une fonction Lambda entre comptes définie dans un compte appelé Production. Cette fonction est invoquée depuis une machine d'état dans un compte appelé Development. Dans ce didacticiel, le compte de développement est appelé compte source et le compte de production est le compte cible contenant le rôle IAM cible.

Pour commencer, dans la définition de votre `Task` État, vous spécifiez le rôle IAM cible que la machine d'État doit assumer avant d'appeler la fonction Lambda entre comptes. Modifiez ensuite la politique de confiance dans le rôle IAM cible pour permettre au compte source d'assumer temporairement le rôle cible. En outre, pour appeler la AWS ressource, définissez les autorisations appropriées dans le rôle IAM cible. Enfin, mettez à jour le rôle d'exécution du compte source pour spécifier l'autorisation requise pour assumer le rôle cible.

Vous pouvez configurer votre machine d'état pour qu'elle assume un rôle IAM afin d'accéder à des ressources à partir de plusieurs Comptes AWS. Cependant, une machine d'état ne peut assumer qu'un seul rôle IAM à la fois en fonction de la définition de `Task` l'état.

> **Note**
>
> Actuellement, l'intégration du AWS SDK entre régions et l'accès aux AWS ressources entre régions ne sont pas disponibles dans Step Functions.

Table des matières

Prérequis

- Ce didacticiel utilise l'exemple d'une fonction Lambda pour montrer comment configurer l'accès entre comptes. Vous pouvez utiliser n'importe quelle autre AWS ressource, mais assurez-vous de l'avoir configurée dans un autre compte.

  > **Important**
  >
  > Les politiques IAM basées sur les ressources et les rôles ne délèguent l'accès entre les comptes qu'au sein d'une seule partition. Par exemple, supposons que vous avez un compte dans la région USA Ouest (Californie du Nord) sur la partition `aws` standard. Vous avez également un compte dans la région Chine (Beijing) sur la partition `aws-cn`. Vous ne pouvez pas utiliser une politique basée sur les ressources d'Amazon S3 dans votre compte en Chine (Beijing) pour autoriser l'accès aux utilisateurs de votre compte `aws` standard.

- Notez le nom de ressource Amazon (ARN) de la ressource multicompte dans un fichier texte. Plus loin dans ce didacticiel, vous fournirez cet ARN dans la définition d'`Task`état de votre machine à états. Voici un exemple d'ARN de fonction Lambda :

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

- Assurez-vous d'avoir créé le rôle IAM cible que la machine d'état doit assumer.

Étape 1 : mettre à jour la définition de l'état de la tâche pour spécifier le rôle cible

Dans l'`Task`état de votre flux de travail, ajoutez un `Credentials` champ contenant l'identité que la machine d'état doit adopter avant d'appeler la fonction Lambda entre comptes.

La procédure suivante explique comment accéder à une fonction Lambda multi-comptes appelée. `Echo` Vous pouvez appeler n'importe quelle AWS ressource en suivant ces étapes.

1. Ouvrez la console Step Functions et choisissez Create state machine.
2. Sur la page Choisir une méthode de création, choisissez Concevez votre flux de travail visuellement et conservez toutes les sélections par défaut.
3. Pour ouvrir Workflow Studio, choisissez Next.
4. Dans l'onglet Actions, glissez et déposez un `Task` état sur le canevas. Cela appelle la fonction Lambda entre comptes qui utilise cet état. `Task`
5. Dans l'onglet Configuration, procédez comme suit :

   a. Renommez l'état en. **`Cross-account call`**

   b. Pour Nom de la fonction, choisissez Enter function name, puis entrez l'ARN de la fonction Lambda dans le champ. Par exemple, `arn:aws:lambda:us-east-2:111122223333:function:Echo`.

   c. Pour Fournir l'ARN du rôle IAM, spécifiez l'ARN du rôle IAM cible. Par exemple, `arn:aws:iam::111122223333:role/LambdaRole`.

   > **Tip**
   >
   > Vous pouvez également spécifier un chemin de référence vers une paire clé-valeur existante dans l'entrée JSON de l'État qui contient l'ARN du rôle IAM. Pour ce faire, choisissez Get IAM role ARN at runtime from state input. Pour un exemple de spécification d'une valeur à l'aide d'un chemin de référence, consultezSpécifier JSONPath comme ARN du rôle IAM.

6. Choisissez Suivant.

7. Sur la page du code généré par la révision, choisissez Next.
8. Sur la page Spécifier les paramètres de la machine à états, spécifiez les détails de la nouvelle machine à états, tels que le nom, les autorisations et le niveau de journalisation.
9. Choisissez Create state machine (Créer une machine d'état).
10. Notez l'ARN du rôle IAM de la machine à états et l'ARN de la machine à états dans un fichier texte. Vous devrez fournir ces ARN dans la politique de confiance du compte cible.

La définition de votre `Task` état doit maintenant ressembler à la définition suivante.

```
{
  "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
    }
  }
}
```

Étape 2 : Mettre à jour la politique de confiance du rôle cible

Le rôle IAM doit exister dans le compte cible et vous devez modifier sa politique de confiance pour permettre au compte source d'assumer ce rôle temporairement. En outre, vous pouvez contrôler qui peut assumer le rôle IAM cible.

Après avoir créé la relation de confiance, un utilisateur du compte source peut utiliser l'opération d'AssumeRoleAPI AWS Security Token Service (AWS STS). Cette opération fournit des informations d'identification de sécurité temporaires qui permettent d'accéder aux AWS ressources d'un compte cible.

1. Ouvrez la console IAM à l’adresse https://console.aws.amazon.com/iam/.

2. Dans le volet de navigation de la console, choisissez Rôles, puis utilisez la zone de recherche pour rechercher le rôle IAM cible. Par exemple, *LambdaRole*.
3. Choisissez l'onglet Trust relationships.
4. Choisissez Modifier la politique de confiance et collez la politique de confiance suivante. Assurez-vous de remplacer le Compte AWS numéro et l'ARN du rôle IAM. Le `sts:ExternalId` champ contrôle également qui peut assumer le rôle. Le nom de la machine à états ne doit inclure que des caractères pris en charge par l' AWS Security Token Service `AssumeRole`API. Pour plus d'informations, consultez AssumeRolela référence de AWS Security Token Service l'API.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:role/ExecutionRole"  // The source
 account's state machine execution role ARN
      },
      "Condition": {  // Control which account and state machine can assume the
 target IAM role

        "StringEquals": {
          "sts:ExternalId": "arn:aws:states:us-
east-1:123456789012:stateMachine:testCrossAccount"   //// ARN of the state machine
 that will assume the role.
        }
      }
    }
  ]
}
```

5. Gardez cette fenêtre ouverte et passez à l'étape suivante pour d'autres actions.

Étape 3 : ajouter l'autorisation requise dans le rôle cible

Les autorisations définies dans les politiques IAM déterminent si une demande spécifique est autorisée ou refusée. Le rôle IAM cible doit disposer de l'autorisation appropriée pour appeler la fonction Lambda.

1. Choisissez l’onglet Permissions (Autorisations).

2. Sélectionnez Ajouter des autorisations, puis Ajouter la politique.
3. Choisissez l'onglet JSON et remplacez le contenu existant par l'autorisation suivante. Assurez-vous de remplacer l'ARN de votre fonction Lambda.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "lambda:InvokeFunction",
      "Resource": "arn:aws:lambda:us-east-2:111122223333:function:Echo"  // The
 cross-account AWS resource being accessed
    }
  ]
}
```

4. Choisissez Examiner une politique.
5. Sur la page Révision de la politique, entrez le nom de l'autorisation, puis choisissez Créer une politique.

Étape 4 : Ajouter l'autorisation dans le rôle d'exécution pour assumer le rôle cible

Step Functions ne génère pas automatiquement la AssumeRolepolitique pour toutes les intégrations de services entre comptes. Vous devez ajouter l'autorisation requise dans le rôle d'exécution de la machine d'état pour lui permettre d'assumer un rôle IAM cible dans un ou plusieurs Comptes AWS rôles.

1. Ouvrez le rôle d'exécution de votre machine à états dans la console IAM à l'adresse https://console.aws.amazon.com/iam/. Pour cela :
   a. Ouvrez la machine à états que vous avez créée à l'étape 1 dans le compte source.
   b. Sur la page détaillée de la machine State, choisissez l'ARN du rôle IAM.
2. Dans l'onglet Autorisations, choisissez Ajouter des autorisations, puis choisissez Créer une politique intégrée.
3. Choisissez l'onglet JSON et remplacez le contenu existant par l'autorisation suivante. Assurez-vous de remplacer l'ARN de votre fonction Lambda.

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

```
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Resource": "arn:aws:iam::111122223333:role/LambdaRole"  // The target role
 to be assumed
    }
  ]
}
```

4. Choisissez Examiner une politique.
5. Sur la page Révision de la politique, entrez le nom de l'autorisation, puis choisissez Créer une politique.

Accès entre comptes pour le modèle d'intégration .sync

Lorsque vous utilisez les modèles d'intégration des `.sync` services dans vos flux de travail, Step Functions interroge la ressource inter-comptes invoquée pour confirmer que la tâche est terminée. Cela entraîne un léger décalage entre le moment où la tâche est réellement terminée et le moment où Step Functions reconnaît que la tâche est terminée. Le rôle IAM cible a besoin des autorisations requises pour effectuer un `.sync` appel afin de terminer cette boucle d'interrogation. Pour ce faire, le rôle IAM cible doit disposer d'une politique de confiance qui autorise le compte source à l'assumer. En outre, le rôle IAM cible a besoin des autorisations requises pour terminer la boucle d'interrogation.

> **Note**
>
> Pour les flux de travail express imbriqués, `arn:aws:states:::states:startExecution.sync` n'est actuellement pas pris en charge. Utilisez `arn:aws:states:::aws-sdk:sfn:startSyncExecution` à la place.

Mise à jour de la politique de confiance pour les appels .sync

Mettez à jour la politique de confiance de votre rôle IAM cible comme indiqué dans l'exemple suivant. Le `sts:ExternalId` champ contrôle également qui peut assumer le rôle. Le nom de la machine à états ne doit inclure que des caractères pris en charge par l' AWS Security Token Service `AssumeRole`API. Pour plus d'informations, consultez AssumeRolela référence de AWS Security Token Service l'API.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Principal": {
        "AWS": "arn:aws:iam::sourceAccountID:role/InvokeRole",
      },
      "Condition": {
        "StringEquals": {
          "sts:ExternalId": "arn:aws:states:us-
east-2:sourceAccountID:stateMachine:stateMachineName"
        }
      }
    }
  ]
}
```

Autorisations requises pour les appels .sync

Pour accorder les autorisations requises pour votre machine d'état, mettez à jour les autorisations requises pour le rôle IAM cible. Pour plus d'informations, consultez the section called "Politiques IAM pour les services intégrés". Les EventBridge autorisations Amazon indiquées dans les exemples de politiques ne sont pas requises. Par exemple, pour démarrer une machine à états, ajoutez les autorisations suivantes.

```
{
  "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:*"
      ]
    }
  ]
}
```

Points de terminaison Amazon VPC pour Step Functions

Si vous utilisez Amazon Virtual Private Cloud (Amazon VPC) pour héberger vos AWS ressources, vous pouvez établir une connexion entre votre Amazon VPC et les flux de travail. AWS Step Functions Vous pouvez utiliser cette connexion avec vos flux de travail Step Functions sans passer par l'Internet public. Les points de terminaison Amazon VPC sont pris en charge par les flux de travail standard, les flux de travail express et les flux de travail express synchrones.

Amazon VPC vous permet de lancer AWS des ressources dans un réseau virtuel personnalisé. Vous pouvez utiliser un VPC pour contrôler vos paramètres réseau, tels que la plage d'adresses IP, les sous-réseaux, les tables de routage et les passerelles réseau. Pour plus d'informations sur les VPC, consultez le Guide de l'utilisateur Amazon VPC.

Pour connecter votre Amazon VPC à Step Functions, vous devez d'abord définir un point de terminaison VPC d'interface, qui vous permet de connecter votre VPC à d'autres services. AWS Le point de terminaison assure une connectivité évolutive et fiable, sans qu'une passerelle Internet, une instance NAT (Network Address Translation) ou une connexion VPN ne soit nécessaire. Pour de plus amples informations, consultez Points de terminaison VPC (AWS PrivateLink) dans le Guide de l'utilisateur Amazon VPC.

Création du point de terminaison

Vous pouvez créer un AWS Step Functions point de terminaison dans votre VPC à l'aide du AWS Management Console, the AWS Command Line Interface (AWS CLI), d'un AWS SDK, de l' AWS Step Functions API ou. AWS CloudFormation

Pour plus d'informations sur la création et la configuration d'un point de terminaison à l'aide de la console Amazon VPC ou de la AWS CLI, consultez la section Création d'un point de terminaison d'interface dans le Guide de l'utilisateur Amazon VPC.

> **Note**
>
> Lorsque vous créez un point de terminaison, spécifiez Step Functions comme service auquel vous souhaitez que votre VPC se connecte. Dans la console Amazon VPC, les noms des services varient en fonction de la AWS région. Par exemple, si vous choisissez US East (Virginie du Nord), le nom du service pour Standard Workflows et Express Workflows est com.amazonaws.us-east-1.states, et le nom du service pour Synchronous Express Workflows est com.amazonaws.us-east-1.sync-states.

> **Note**
>
> Il est possible d'utiliser des points de terminaison VPC sans remplacer le point de terminaison dans le SDK via un DNS privé. Toutefois, si vous souhaitez remplacer le point de terminaison dans le SDK pour Synchronous Express Workflows, vous devez définir `DisableHostPrefixInjection` la configuration sur. `true` Exemple (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();
> ```

Pour plus d'informations sur la création et la configuration d'un point de terminaison à l'aide de AWS CloudFormation, consultez la ressource AWS : :EC2 : :VPCEndpoint dans le guide de l'utilisateur.AWS CloudFormation

Politiques relatives aux terminaux Amazon VPC

Pour contrôler l'accès à la connectivité à Step Functions, vous pouvez associer une politique de point de terminaison AWS Identity and Access Management (IAM) lors de la création d'un point de terminaison Amazon VPC. Vous pouvez créer des règles IAM complexes en associant plusieurs politiques de point de terminaison. Pour plus d’informations, consultez :

- Politiques relatives aux terminaux Amazon Virtual Private Cloud pour Step Functions
- Création d'autorisations IAM granulaires pour les utilisateurs non administrateurs
- Contrôle de l'accès aux services avec les points de terminaison d'un VPC

Politiques relatives aux terminaux Amazon Virtual Private Cloud pour Step Functions

Vous pouvez créer une politique de point de terminaison Amazon VPC pour Step Functions dans laquelle vous spécifiez les éléments suivants :

- Le principal qui peut exécuter des actions.
- Les actions qui peuvent être effectuées.
- La ressource sur laquelle les actions peuvent être effectuées.

L'exemple suivant montre une politique de point de terminaison Amazon VPC qui permet à un utilisateur de créer des machines d'état et refuse à tous les autres utilisateurs l'autorisation de supprimer des machines d'état. L'exemple de stratégie accorde également l'autorisation d'exécution à tous les utilisateurs .

```
{
    "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": "*"
        }
    ]
}
```

Pour plus d'informations sur la création de stratégies de point de terminaison, consultez les rubriques suivantes :

- Création d'autorisations IAM granulaires pour les utilisateurs non administrateurs
- Contrôle de l'accès aux services avec les points de terminaison d'un VPC

Politiques IAM pour les services intégrés

Lorsque vous créez une machine à états dans la AWS Step Functions console, Step Functions produit une politique AWS Identity and Access Management (IAM) basée sur les ressources utilisées dans la définition de votre machine à états, comme suit :

- Si votre machine d'état utilise l'une des intégrations optimisées, Step Functions créera une politique avec les autorisations et les rôles nécessaires pour votre machine d'état. (Exception : MediaConvert l'intégration nécessite que vous définissiez manuellement les autorisations — voirPolitiques IAM pour AWS Elemental MediaConvert.)
- Si votre machine d'état utilise l'une des intégrations du AWS SDK, un rôle IAM avec des autorisations partielles sera créé. Vous pouvez ensuite utiliser la console IAM pour ajouter les politiques de rôle manquantes.

Les exemples suivants montrent comment Step Functions génère une politique IAM basée sur la définition de votre machine à états. Les éléments dans l'exemple de code tels que *`[[resourceName]]`* sont remplacés par les ressources statiques répertoriées dans la définition de votre machine d'état. Si vous disposez de plusieurs ressources statiques, le rôle IAM comportera une entrée pour chacune d'entre elles.

Ressources dynamiques ou statiques

Les ressources statiques sont définies directement dans l'état de tâche de votre machine d'état. Lorsque vous incluez les informations relatives aux ressources que vous souhaitez appeler directement dans l'état de vos tâches, Step Functions crée un rôle IAM uniquement pour ces ressources.

Les ressources dynamiques sont celles qui sont transmises à votre état d'entrée et accessibles à l'aide d'un chemin (voir Chemins). Si vous transmettez des ressources dynamiques à votre tâche, Step Functions créera une politique plus privilégiée qui spécifie :`"Resource": "*"`.

Autorisations supplémentaires pour les tâches utilisant le modèle Run a Job

Pour les tâches qui utilisent le modèle Run a Job (celles qui se terminent par`.sync`), des autorisations supplémentaires sont nécessaires pour surveiller les actions d'API des services connectés et recevoir une réponse à partir de celles-ci. Les politiques associées incluent davantage d'autorisations que pour les tâches qui utilisent les modèles Request Response ou Wait for Callback. Consultez Modèles d'intégration des services pour plus d'informations sur les tâches synchrones.

> **Note**
>
> Vous devez fournir des autorisations supplémentaires pour les intégrations de services qui prennent en charge le modèle Run a Job (.sync).

Step Functions utilise deux méthodes pour surveiller l'état d'une tâche lorsqu'elle est exécutée sur un service connecté : les sondages et les événements.

Le sondage nécessite une autorisation `Describe` ou des actions d'`Get`API, telles que `ecs:DescribeTasks` ou`glue:GetJobRun`. Si ces autorisations sont absentes de votre rôle, Step Functions ne sera peut-être pas en mesure de déterminer le statut de votre tâche. Cela est dû au fait que certaines intégrations du service Run a Job (.sync) ne prennent pas en charge les EventBridge événements, et certains services n'envoient des événements que dans la mesure du possible.

Les événements envoyés par les AWS services à Amazon EventBridge sont dirigés vers Step Functions à l'aide d'une règle gérée et nécessitent des autorisations pour `events:PutTargets`et`events:PutRule`, et`events:DescribeRule`. Si ces autorisations ne sont pas associées à votre rôle, Step Functions ne sera peut-être pas informée de la fin de votre tâche pendant un certain temps. Pour plus d'informations sur les EventBridge événements, consultez la section Événements liés AWS aux services.

> **Note**
>
> Pour les tâches Run a Job (.sync) qui prennent en charge à la fois les sondages et les événements, votre tâche peut toujours s'exécuter correctement à l'aide d'événements.

Cela peut se produire même si votre rôle ne dispose pas des autorisations requises pour le sondage. Dans ce cas, vous ne remarquerez peut-être pas immédiatement que les autorisations de vote sont incorrectes ou manquantes. Dans les rares cas où l'événement ne parvient pas à être transmis ou traité par Step Functions, votre exécution peut être bloquée. Pour vérifier que vos autorisations d'interrogation sont correctement configurées, vous pouvez exécuter une exécution dans un environnement sans EventBridge événements de la manière suivante :

- Supprimez la règle gérée de EventBridge, qui est chargée de transférer les événements vers Step Functions. Cette règle gérée est partagée par toutes les machines d'état de votre compte. Vous devez donc effectuer cette action uniquement dans un compte de test ou de développement afin d'éviter tout impact involontaire sur les autres machines d'état. Vous pouvez identifier la règle gérée spécifique à supprimer en inspectant le `Resource` champ utilisé `events:PutRule` dans le modèle de politique du service cible. La règle gérée sera recréée la prochaine fois que vous créerez ou mettrez à jour une machine à états utilisant cette intégration de services. Pour plus d'informations sur la suppression de EventBridge règles, consultez la section Désactivation ou suppression d'une règle.
- Effectuez le test avec Step Functions Local, qui ne prend pas en charge l'utilisation d'événements pour effectuer des tâches Run a Job (.sync). Pour utiliser Step Functions Local, assumez le rôle IAM utilisé par votre machine d'état. Vous devrez peut-être modifier la relation de confiance. Définissez les variables`AWS_ACCESS_KEY_ID`,`AWS_SECRET_ACCESS_KEY`, et d'`AWS_SESSION_TOKEN`environnement sur les valeurs du rôle assumé, puis lancez Step Functions Local à l'aide dejava -jar StepFunctionsLocal.jar. Enfin, utilisez le --endpoint-url paramètre AWS CLI with pour créer une machine à états, démarrer une exécution et obtenir l'historique des exécutions. Pour plus d'informations, consultez Tester les machines d'état localement.

Si une tâche utilisant le modèle Run a Job (.sync) est arrêtée, Step Functions fera de son mieux pour annuler la tâche. Cela nécessite une autorisation pour`Cancel`, `StopTerminate`, ou des actions `Delete` d'API, telles que `batch:TerminateJob` ou`eks:DeleteCluster`. Si ces autorisations ne sont pas associées à votre rôle, Step Functions ne sera pas en mesure d'annuler votre tâche et vous risquez de devoir payer des frais supplémentaires pendant son exécution. Pour plus d'informations sur l'arrêt des tâches, consultez Run a Job.

Modèles de politiques utilisés pour créer des rôles IAM

Les rubriques suivantes incluent les modèles de politique utilisés lorsque vous choisissez que Step Functions crée un nouveau rôle pour vous.

> **Note**
>
> Consultez ces modèles pour comprendre comment Step Functions crée vos politiques IAM, et pour illustrer comment créer manuellement des politiques IAM pour Step Functions lorsque vous travaillez avec d'autres AWS services. Pour plus d'informations sur les intégrations de services Step Functions, consultezUtilisation AWS Step Functions avec d'autres services.

Rubriques

- Politiques IAM pour AWS Step Functions
- Politiques IAM pour AWS X-Ray
- Activités ou aucune tâche

Politiques IAM pour Amazon API Gateway

Les exemples de modèles suivants montrent comment AWS Step Functions générer des politiques IAM en fonction des ressources contenues dans la définition de votre machine d'état. Pour plus d'informations, consultez Politiques IAM pour les services intégrés et Modèles d'intégration des services.

Ressources :

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "execute-api:Invoke"
            ],
            "Resource": [
                "arn:[[region]]:[[accountId]]:*"
            ]
        }
    ]
}
```

L'exemple de code suivant montre une politique de ressources pour appeler 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>"
                    ]
                }
            }
        }
    ]
}
```

Politiques IAM pour Amazon Athena

Les exemples de modèles suivants montrent comment AWS Step Functions générer des politiques IAM en fonction des ressources contenues dans la définition de votre machine d'état. Pour plus d'informations, consultez Politiques IAM pour les services intégrés et Modèles d'intégration des services.

`StartQueryExecution`

Ressources statiques

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": [
            "*"
        ]
    }
]
}
```

Ressources dynamiques

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

Ressources

```
{
    "Version": "2012-10-17",
    "Statement":[
    {
        "Effect": "Allow",
        "Action": [
            "athena:stopQueryExecution"
        ],
        "Resource": [
            "arn:aws:athena:{{region}}:{{accountId}}:workgroup/*"
        ]
    }
    ]
}
```

GetQueryExecution

Ressources

```
{
    "Version": "2012-10-17",
    "Statement":[
    {
        "Effect": "Allow",
        "Action": [
            "athena:getQueryExecution"
        ],
        "Resource": [
            "arn:aws:athena:{{region}}:{{accountId}}:workgroup/*"
        ]
    }
    ]
}
```

GetQueryResults

Ressources

```
{
    "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:::*"
        ]
    }
    ]
}
```

Politiques IAM pour AWS Batch

Les exemples de modèles suivants montrent comment AWS Step Functions générer des politiques IAM en fonction des ressources contenues dans la définition de votre machine d'état. Pour plus d'informations, consultez Politiques IAM pour les services intégrés et Modèles d'intégration des services.

Étant donné qu'il AWS Batch fournit une prise en charge partielle du contrôle d'accès au niveau des ressources, vous devez utiliser. `"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

Lorsque vous créez une machine d'état à l'aide de la console, un rôle d'exécution est Step Functions automatiquement créé pour votre machine d'état avec le moins de privilèges requis. Ces IAM rôles générés automatiquement sont valides pour la machine à états Région AWS dans laquelle vous créez la machine à états.

Les exemples de modèles suivants montrent comment AWS Step Functions générer des politiques IAM en fonction des ressources contenues dans la définition de votre machine d'état. Pour plus d'informations, consultez Politiques IAM pour les services intégrés et Modèles d'intégration des services.

Lorsque vous créez des IAM politiques, nous vous recommandons de ne pas y inclure de caractères génériques. En tant que bonne pratique en matière de sécurité, vous devez limiter autant que possible vos politiques. Vous ne devez utiliser des politiques dynamiques que lorsque certains paramètres d'entrée ne sont pas connus pendant l'exécution.

Dans cette rubrique

- IAMexemples de politiques pour Amazon Bedrock l'intégration avec Step Functions

IAMexemples de politiques pour Amazon Bedrock l'intégration avec Step Functions

La section suivante décrit les IAM autorisations dont vous avez besoin en fonction de l'Amazon BedrockAPI que vous utilisez pour une fondation ou un modèle provisionné spécifique. Cette section contient également des exemples de politiques qui accordent un accès complet.

N'oubliez pas de remplacer le texte *en italique* par les informations spécifiques à votre ressource.

- IAMexemple de politique pour accéder à un modèle de fondation spécifique en utilisant InvokeModel
- IAMexemple de politique pour accéder à un modèle provisionné spécifique en utilisant InvokeModel
- Exemple de IAM politique d'accès complet à utiliser InvokeModel
- IAMexemple de politique pour accéder à un modèle de fondation spécifique en tant que modèle de base
- IAMexemple de politique pour accéder à un modèle personnalisé spécifique en tant que modèle de base
- Exemple de IAM politique d'accès complet pour utiliser CreateModelCustomizationJob .sync
- IAMexemple de politique pour accéder à un modèle de base spécifique à l'aide de CreateModelCustomizationJob .sync
- IAMexemple de politique pour accéder à un modèle personnalisé à l'aide de CreateModelCustomizationJob .sync
- Exemple de IAM politique d'accès complet pour utiliser CreateModelCustomizationJob .sync

IAMexemple de politique pour accéder à un modèle de fondation spécifique en utilisant InvokeModel

Voici un exemple de IAM politique pour une machine à états qui accède à un modèle de base spécifique nommé à `amazon.titan-text-express-v1` l'aide de l'action d'InvokeModelAPI.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Sid": "InvokeModel1",
            "Action": [
                "bedrock:InvokeModel"
            ],
            "Resource": [
                "arn:aws:bedrock:us-east-2::foundation-model/amazon.titan-text-express-v1"
            ]
```

```
        }
    ]
}
```

IAMexemple de politique pour accéder à un modèle provisionné spécifique en utilisant InvokeModel

Voici un exemple de IAM politique pour une machine à états qui accède à un modèle provisionné spécifique nommé à `c2oi931ulksx` l'aide de l'action d'InvokeModelAPI.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Sid": "InvokeModel1",
      "Action": [
        "bedrock:InvokeModel"
      ],
      "Resource": [
        "arn:aws:bedrock:us-east-2:123456789012:provisioned-model/c2oi931ulksx"
      ]
    }
  ]
}
```

Exemple de IAM politique d'accès complet à utiliser InvokeModel

Voici un exemple de IAM politique pour une machine à états qui fournit un accès complet lorsque vous utilisez l'action InvokeModelAPI.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Sid": "InvokeModel1",
            "Action": [
                "bedrock:InvokeModel"
            ],
            "Resource": [
                "arn:aws:bedrock:us-east-2::foundation-model/*",
                "arn:aws:bedrock:us-east-2:123456789012:provisioned-model/*"
```

```
            ]
        }
    ]
}
```

IAMexemple de politique pour accéder à un modèle de fondation spécifique en tant que modèle de base

Voici un exemple de IAM politique permettant à une machine à états d'accéder à un modèle de base spécifique `amazon.titan-text-express-v1` nommé modèle de base à l'aide de l'action CreateModelCustomizationJobAPI.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Sid": "CreateModelCustomizationJob1",
            "Action": [
                "bedrock:CreateModelCustomizationJob"
            ],
            "Resource": [
                "arn:aws:bedrock:us-east-2::foundation-model/amazon.titan-text-express-
v1",
                "arn:aws:bedrock:us-east-2:123456789012:custom-model/*",
                "arn:aws:bedrock:us-east-2:123456789012:model-customization-job/*"
            ]
        },
        {
            "Effect": "Allow",
            "Sid": "CreateModelCustomizationJob2",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::123456789012:role/myRole"
            ]
        }
    ]
}
```

IAMexemple de politique pour accéder à un modèle personnalisé spécifique en tant que modèle de base

Voici un exemple de IAM politique permettant à une machine à états d'accéder à un modèle personnalisé spécifique en tant que modèle de base à l'aide de l'action CreateModelCustomizationJobAPI.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Sid": "CreateModelCustomizationJob1",
            "Action": [
                "bedrock:CreateModelCustomizationJob"
            ],
            "Resource": [
                "arn:aws:bedrock:us-east-2:123456789012:custom-model/*",
                "arn:aws:bedrock:us-east-2:123456789012:model-customization-job/*"
            ]
        },
        {
            "Effect": "Allow",
            "Sid": "CreateModelCustomizationJob2",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::123456789012:role/[[roleName]]"
            ]
        }
    ]
}
```

Exemple de IAM politique d'accès complet pour utiliser CreateModelCustomizationJob .sync

Voici un exemple de IAM politique pour une machine à états qui fournit un accès complet lorsque vous utilisez l'action CreateModelCustomizationJobAPI.

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

```
            "Effect": "Allow",
            "Sid": "CreateModelCustomizationJob1",
            "Action": [
                "bedrock:CreateModelCustomizationJob"
            ],
            "Resource": [
                "arn:aws:bedrock:us-east-2::foundation-model/*",
                "arn:aws:bedrock:us-east-2:123456789012:custom-model/*",
                "arn:aws:bedrock:us-east-2:123456789012:model-customization-job/*"
            ]
        },
        {
            "Effect": "Allow",
            "Sid": "CreateModelCustomizationJob2",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::123456789012:role/myRole"
            ]
        }
    ]
}
```

IAMexemple de politique pour accéder à un modèle de base spécifique à l'aide de CreateModelCustomizationJob .sync

Voici un exemple de IAM politique permettant à une machine d'état d'accéder à un modèle de base spécifique nommé à `amazon.titan-text-express-v1` l'aide de l'action d'API CreateModelCustomizationJob.sync.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Sid": "CreateModelCustomizationJob1",
            "Action": [
                "bedrock:CreateModelCustomizationJob"
            ],
            "Resource": [
                "arn:aws:bedrock:us-east-2::foundation-model/amazon.titan-text-express-
v1",
```

```
                "arn:aws:bedrock:us-east-2:123456789012:custom-model/*",
                "arn:aws:bedrock:us-east-2:123456789012:model-customization-job/*"
            ]
        },
        {
            "Effect": "Allow",
            "Sid": "CreateModelCustomizationJob2",
            "Action": [
                "bedrock:GetModelCustomizationJob",
                "bedrock:StopModelCustomizationJob"
            ],
            "Resource": [
                "arn:aws:bedrock:us-east-2:123456789012:model-customization-job/*"
            ]
        },
        {
            "Effect": "Allow",
            "Sid": "CreateModelCustomizationJob3",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::123456789012:role/myRole"
            ]
        }
    ]
}
```

IAMexemple de politique pour accéder à un modèle personnalisé à l'aide de CreateModelCustomizationJob .sync

Voici un exemple de IAM politique permettant à une machine d'état d'accéder à un modèle personnalisé à l'aide de l'action d'API CreateModelCustomizationJob.sync.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Sid": "CreateModelCustomizationJob1",
            "Action": [
                "bedrock:CreateModelCustomizationJob"
            ],
```

```
            "Resource": [
                "arn:aws:bedrock:us-east-2:123456789012:custom-model/*",
                "arn:aws:bedrock:us-east-2:123456789012:model-customization-job/*"
            ]
        },
        {
            "Effect": "Allow",
            "Sid": "CreateModelCustomizationJob2",
            "Action": [
                "bedrock:GetModelCustomizationJob",
                "bedrock:StopModelCustomizationJob"
            ],
            "Resource": [
                "arn:aws:bedrock:us-east-2:123456789012:model-customization-job/*"
            ]
        },
        {
            "Effect": "Allow",
            "Sid": "CreateModelCustomizationJob3",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::123456789012:role/myRole"
            ]
        }
    ]
}
```

Exemple de IAM politique d'accès complet pour utiliser CreateModelCustomizationJob .sync

Voici un exemple de IAM politique pour une machine à états qui fournit un accès complet lorsque vous utilisez l'action d'API CreateModelCustomizationJob.sync.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Sid": "CreateModelCustomizationJob1",
            "Action": [
                "bedrock:CreateModelCustomizationJob"
            ],
```

```
            "Resource": [
                "arn:aws:bedrock:us-east-2::foundation-model/*",
                "arn:aws:bedrock:us-east-2:123456789012:custom-model/*",
                "arn:aws:bedrock:us-east-2:123456789012:model-customization-job/*"
            ]
        },
        {
            "Effect": "Allow",
            "Sid": "CreateModelCustomizationJob2",
            "Action": [
                "bedrock:GetModelCustomizationJob",
                "bedrock:StopModelCustomizationJob"
            ],
            "Resource": [
                "arn:aws:bedrock:us-east-2:123456789012:model-customization-job/*"
            ]
        },
        {
            "Effect": "Allow",
            "Sid": "CreateModelCustomizationJob3",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::123456789012:role/myRole"
            ]
        }
    ]
}
```

Politiques IAM pour AWS CodeBuild

Les exemples de modèles suivants montrent comment AWS Step Functions générer des politiques IAM en fonction des ressources contenues dans la définition de votre machine d'état. Pour plus d’informations, consultez Politiques IAM pour les services intégrés et Modèles d'intégration des services.

Ressources :

```
{
    "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`

Ressources statiques

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]]"
      ]
    }
  ]
}
```

Ressources dynamiques

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

Ressources statiques

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codebuild:StopBuild"
      ],
      "Resource": [
        "arn:aws:codebuild:[[region]]:[[accountId]]:project/[[projectName]]"
      ]
    }
  ]
}
```

Ressources dynamiques

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codebuild:StopBuild"
      ],
      "Resource": [
        "arn:aws:codebuild:[[region]]:*:project/*"
      ]
    }
  ]
```

```
}
```

`BatchDeleteBuilds`

Ressources statiques

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codebuild:BatchDeleteBuilds"
      ],
      "Resource": [
        "arn:aws:codebuild:[[region]]:[[accountId]]:project/[[projectName]]"
      ]
    }
  ]
}
```

Ressources dynamiques

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codebuild:BatchDeleteBuilds"
      ],
      "Resource": [
        "arn:aws:codebuild:[[region]]:*:project/*"
      ]
    }
  ]
}
```

`BatchGetReports`

Ressources statiques

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codebuild:BatchGetReports"
      ],
      "Resource": [
        "arn:aws:codebuild:[[region]]:[[accountId]]:report-group/[[reportName]]"
      ]
    }
  ]
}
```

Ressources dynamiques

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "codebuild:BatchGetReports"
      ],
      "Resource": [
        "arn:aws:codebuild:[[region]]:*:report-group/*"
      ]
    }
  ]
}
```

`StartBuildBatch`

Ressources statiques

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]]"
            ]
        }
    ]
}
```

Ressources dynamiques

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

Ressources statiques

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codebuild:StopBuildBatch"
            ],
            "Resource": [
                "arn:aws:codebuild:[[region]]:[[accountId]]:project/[[projectName]]"
            ]
        }
    ]
}
```

Ressources dynamiques

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codebuild:StopBuildBatch"
            ],
            "Resource": [
                "arn:aws:codebuild:[[region]]:[[accountId]]:project/*"
            ]
        }
    ]
```

```
}
```

`RetryBuildBatch`

Ressources statiques

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]]"
            ]
        }
    ]
}
```

Ressources dynamiques

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`

Ressources statiques

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codebuild:DeleteBuildBatch"
            ],
            "Resource": [
                "arn:aws:codebuild:[[region]]:[[accountId]]:project/[[projectName]]"
            ]
        }
    ]
}
```

Ressources dynamiques

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "codebuild:DeleteBuildBatch"
            ],
            "Resource": [
                "arn:aws:codebuild:[[region]]:[[accountId]]:project/*"
            ]
        }
    ]
}
```

Politiques IAM pour Amazon DynamoDB

Les exemples de modèles suivants montrent comment AWS Step Functions générer des politiques IAM en fonction des ressources contenues dans la définition de votre machine d'état. Pour plus d'informations, consultez Politiques IAM pour les services intégrés et Modèles d'intégration des services.

Ressources statiques

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "dynamodb:GetItem",
                "dynamodb:PutItem",
                "dynamodb:UpdateItem",
                "dynamodb:DeleteItem"
            ],
            "Resource": [
                "arn:aws:dynamodb:[[region]]:[[accountId]]:table/[[tableName]]"
            ]
        }
    ]
}
```

Ressources dynamiques

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "dynamodb:GetItem",
                "dynamodb:PutItem",
                "dynamodb:UpdateItem",
                "dynamodb:DeleteItem"
            ],
            "Resource": "*"
        }
    ]
}
```

Pour plus d'informations sur les politiques IAM pour toutes les actions d'API DynamoDB, consultez la section Politiques IAM avec DynamoDB dans le manuel du développeur Amazon DynamoDB. En outre, pour plus d'informations sur les politiques IAM pour PartiQL pour DynamoDB, consultez la section Politiques IAM avec PartiQL pour DynamoDB dans le manuel du développeur Amazon DynamoDB.

Politiques IAM pour Amazon ECS/AWS Fargate

Les exemples de modèles suivants montrent comment AWS Step Functions générer des politiques IAM en fonction des ressources contenues dans la définition de votre machine d'état. Pour plus d'informations, consultez Politiques IAM pour les services intégrés et Modèles d'intégration des services.

Comme la valeur de n'`TaskId`est pas connue tant que la tâche n'est pas soumise, Step Functions crée une `"Resource": "*"` politique plus privilégiée.

> **Note**
>
> Vous ne pouvez arrêter que les tâches Amazon Elastic Container Service (Amazon ECS) lancées par Step Functions, malgré `"*"` la politique IAM.

Run a Job (.sync)

Ressources statiques

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

Ressources dynamiques

```
{
    "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)

Ressources statiques

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:RunTask"
            ],
            "Resource": [
                "arn:aws:ecs:[[region]]:
[[accountId]]:task-definition/[[taskDefinition]]:[[revisionNumber]]"
            ]
        }
    ]
}
```

Ressources dynamiques

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecs:RunTask"
            ],
            "Resource": "*"
        }
    ]
}
```

Si vos tâches Amazon ECS planifiées nécessitent l'utilisation d'un rôle d'exécution de tâche, d'un rôle de tâche ou d'un remplacement de rôle de tâche, vous devez ajouter des `iam:PassRole`

autorisations pour chaque rôle d'exécution de tâche, rôle de tâche ou remplacement de rôle de tâche au rôle CloudWatch Events IAM de l'entité appelante, qui est dans ce cas Step Functions.

Politiques IAM pour Amazon EKS

Les exemples de modèles suivants montrent comment AWS Step Functions générer des politiques IAM en fonction des ressources contenues dans la définition de votre machine d'état. Pour plus d'informations, consultez Politiques IAM pour les services intégrés et Modèles d'intégration des services.

`CreateCluster`

Ressources

```
{
    "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`

Ressources

```
{
    "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`

Ressources

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks:DeleteCluster",
                "eks:DescribeCluster"
            ],
            "Resource": [
                "arn:aws:eks:sa-east-1:444455556666:cluster/ExampleCluster"
            ]
        }
    ]
}
```

`DeleteNodegroup`

Ressources

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks:DeleteNodegroup",
```

```
                "eks:DescribeNodegroup"
            ],
            "Resource": [
                "arn:aws:eks:sa-east-1:444455556666:nodegroup/ExampleCluster/
ExampleNodegroup/*"
            ]
        }
    ]
}
```

Pour plus d'informations sur l'utilisation d'Amazon EKS avec Step Functions, consultezAppelez Amazon EKS avec Step Functions.

Politiques IAM pour Amazon EMR

Les exemples de modèles suivants montrent comment AWS Step Functions générer des politiques IAM en fonction des ressources contenues dans la définition de votre machine d'état. Pour plus d'informations, consultez Politiques IAM pour les services intégrés et Modèles d'intégration des services.

`addStep`

Ressources statiques

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "elasticmapreduce:AddJobFlowSteps",
        "elasticmapreduce:DescribeStep",
        "elasticmapreduce:CancelSteps"
      ],
      "Resource": [
        "arn:aws:elasticmapreduce:[[region]]:[[accountId]]:cluster/[[clusterId]]"
      ]
    }
  ]
}
```

Ressources dynamiques

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "elasticmapreduce:AddJobFlowSteps",
        "elasticmapreduce:DescribeStep",
        "elasticmapreduce:CancelSteps"
      ],
      "Resource": "arn:aws:elasticmapreduce:*:*:cluster/*"
    }
  ]
}
```

`cancelStep`

Ressources statiques

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "elasticmapreduce:CancelSteps",
            "Resource": [

 "arn:aws:elasticmapreduce:[[region]]:[[accountId]]:cluster/[[clusterId]]"
            ]
        }
    ]
}
```

Ressources dynamiques

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "elasticmapreduce:CancelSteps",
            "Resource": "arn:aws:elasticmapreduce:*:*:cluster/*"
```

```
        }
    ]
}
```

createCluster

Ressources statiques

```
{
  "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

Ressources statiques

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "elasticmapreduce:SetTerminationProtection",
            "Resource": [

 "arn:aws:elasticmapreduce:[[region]]:[[accountId]]:cluster/[[clusterId]]"
```

```
            ]
        }
    ]
}
```

Ressources dynamiques

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "elasticmapreduce:SetTerminationProtection",
            "Resource": "arn:aws:elasticmapreduce:*:*:cluster/*"
        }
    ]
}
```

`modifyInstanceFleetByName`

Ressources statiques

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elasticmapreduce:ModifyInstanceFleet",
                "elasticmapreduce:ListInstanceFleets"
            ],
            "Resource": [

 "arn:aws:elasticmapreduce:[[region]]:[[accountId]]:cluster/[[clusterId]]"
            ]
        }
    ]
}
```

Ressources dynamiques

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elasticmapreduce:ModifyInstanceFleet",
                "elasticmapreduce:ListInstanceFleets"
            ],
            "Resource": "arn:aws:elasticmapreduce:*:*:cluster/*"
        }
    ]
}
```

`modifyInstanceGroupByName`

Ressources statiques

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elasticmapreduce:ModifyInstanceGroups",
                "elasticmapreduce:ListInstanceGroups"
            ],
            "Resource": [

 "arn:aws:elasticmapreduce:[[region]]:[[accountId]]:cluster/[[clusterId]]"
            ]
        }
    ]
}
```

Ressources dynamiques

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elasticmapreduce:ModifyInstanceGroups",
```

```
                "elasticmapreduce:ListInstanceGroups"
            ],
            "Resource": "*"
        }
    ]
}
```

`terminateCluster`

Ressources statiques

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "elasticmapreduce:TerminateJobFlows",
        "elasticmapreduce:DescribeCluster"
      ],
      "Resource": [
        "arn:aws:elasticmapreduce:[[region]]:[[accountId]]:cluster/[[clusterId]]"
      ]
    }
  ]
}
```

Ressources dynamiques

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "elasticmapreduce:TerminateJobFlows",
        "elasticmapreduce:DescribeCluster"
      ],
      "Resource": "arn:aws:elasticmapreduce:*:*:cluster/*"
    }
  ]
}
```

Politiques IAM pour Amazon EMR sur EKS

Les exemples de modèles suivants montrent comment AWS Step Functions générer des politiques IAM en fonction des ressources contenues dans la définition de votre machine d'état. Pour plus d'informations, consultez Politiques IAM pour les services intégrés et Modèles d'intégration des services.

`CreateVirtualCluster`

Ressources

```
{
    "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`

Ressources statiques

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]]"
      ]
    }
  ]
}
```

Ressources dynamiques

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`

Ressources statiques

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]]"
          ]
        }
```

```
      }
    }
  ]
}
```

Ressources dynamiques

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

Stratégies IAM pour Amazon EMR Serverless

Lorsque vous créez une machine d'état à l'aide de la console, un rôle d'exécution est Step Functions automatiquement créé pour votre machine d'état avec le moins de privilèges requis. Ces IAM rôles générés automatiquement sont valides pour la machine à états Région AWS dans laquelle vous créez la machine à états.

Les exemples de modèles suivants montrent comment AWS Step Functions générer des politiques IAM en fonction des ressources contenues dans la définition de votre machine d'état. Pour plus d'informations, consultez Politiques IAM pour les services intégrés et Modèles d'intégration des services.

Lorsque vous créez des IAM politiques, nous vous recommandons de ne pas y inclure de caractères génériques. En tant que bonne pratique en matière de sécurité, vous devez limiter autant que possible vos politiques. Vous ne devez utiliser des politiques dynamiques que lorsque certains paramètres d'entrée ne sont pas connus pendant l'exécution.

En outre, les utilisateurs administrateurs doivent faire preuve de prudence lorsqu'ils accordent à des utilisateurs non administrateurs des rôles d'exécution pour exécuter les machines d'état. Nous vous recommandons d'inclure les politiques PassRole dans les rôles d'exécution si vous créez vous-même

des politiques. Nous vous recommandons également d'ajouter les clés de `aws:SourceAccount` contexte `aws:SourceARN` et dans les rôles d'exécution.

Dans cette rubrique

Exemples de politiques IAM pour l'intégration EMR Serverless avec Step Functions

Exemple de politique IAM pour CreateApplication

Voici un exemple de politique IAM pour une machine à états dotée d'un CreateApplication État de la tâche état.

> **Note**
>
> Vous devez spécifier les CreateServiceLinkedRole autorisations dans vos politiques IAM lors de la création de la toute première application de votre compte. Par la suite, il n'est pas nécessaire d'ajouter cette autorisation. Pour plus d'informations sur CreateServiceLinkedRole, consultez CreateServiceLinkedRolele site https://docs.aws.amazon.com/IAM/latest/APIReference/.

Les ressources statiques et dynamiques pour les politiques suivantes sont identiques.

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"
                }
            }
        }
    ]
}
```

Exemple de politique IAM pour StartApplication

Ressources statiques

Vous trouverez ci-dessous des exemples de politique IAM pour les ressources statiques lorsque vous utilisez une machine à états dotée d'un StartApplication État de la tâche état.

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]]"
             ]
        }
    ]
}
```

Ressources dynamiques

Vous trouverez ci-dessous des exemples de politique IAM pour les ressources dynamiques lorsque vous utilisez une machine à états dotée d'un StartApplication [État de la tâche](#) état.

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/*"
             ]
        }
    ]
}
```

Exemple de politique IAM pour StopApplication

Ressources statiques

Vous trouverez ci-dessous des exemples de politique IAM pour les ressources statiques lorsque vous utilisez une machine à états dotée d'un StopApplication État de la tâche état.

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]]"
            ]
        }
    ]
}
```

Ressources dynamiques

Vous trouverez ci-dessous des exemples de politique IAM pour les ressources dynamiques lorsque vous utilisez une machine à états dotée d'un StopApplication [État de la tâche](#) état.

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/*"
            ]
        }
    ]
}
```

Exemple de politique IAM pour DeleteApplication

Ressources statiques

Vous trouverez ci-dessous des exemples de politique IAM pour les ressources statiques lorsque vous utilisez une machine à états dotée d'un DeleteApplication État de la tâche état.

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]]"
            ]
        }
    ]
}
```

Ressources dynamiques

Vous trouverez ci-dessous des exemples de politique IAM pour les ressources dynamiques lorsque vous utilisez une machine à états dotée d'un DeleteApplication État de la tâche état.

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/*"
            ]
        }
    ]
}
```

Exemple de politique IAM pour StartJobRun

Ressources statiques

Vous trouverez ci-dessous des exemples de politique IAM pour les ressources statiques lorsque vous utilisez une machine à états dotée d'un StartJobRun État de la tâche état.

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"
                }
            }
        }
    ]
}
```

Ressources dynamiques

Vous trouverez ci-dessous des exemples de politique IAM pour les ressources dynamiques lorsque vous utilisez une machine à états dotée d'un StartJobRun [État de la tâche](#) état.

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"
                }
            }
        }
    ]
}
```

Exemple de politique IAM pour CancelJobRun

Ressources statiques

Vous trouverez ci-dessous des exemples de politique IAM pour les ressources statiques lorsque vous utilisez une machine à états dotée d'un CancelJobRun [État de la tâche] état.

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]]"
            ]
        }
    ]
}
```

Ressources dynamiques

Vous trouverez ci-dessous des exemples de politique IAM pour les ressources dynamiques lorsque vous utilisez une machine à états dotée d'un CancelJobRun État de la tâche état.

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/*"
            ]
        }
    ]
}
```

Politiques IAM pour Amazon EventBridge

Les exemples de modèles suivants montrent comment AWS Step Functions générer des politiques IAM en fonction des ressources contenues dans la définition de votre machine d'état. Pour plus

d'informations, consultez Politiques IAM pour les services intégrés et Modèles d'intégration des services.

`PutEvents`

Ressources statiques

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "events:PutEvents"
            ],
            "Resource": [
                "arn:aws:events:us-east-1:123456789012:event-bus/stepfunctions-
sampleproject-eventbus"
            ],
            "Effect": "Allow"
        }
    ]
}
```

Ressources dynamiques

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

Pour plus d'informations sur l'utilisation EventBridge avec Step Functions, consultezAppelez EventBridge avec Step Functions.

Politiques IAM pour AWS Lambda

Les exemples de modèles suivants montrent comment AWS Step Functions générer des politiques IAM en fonction des ressources contenues dans la définition de votre machine d'état. Pour plus d'informations, consultez Politiques IAM pour les services intégrés et Modèles d'intégration des services.

AWS Step Functions génère une politique IAM basée sur la définition de votre machine à états. Pour une machine à états avec deux états de AWS Lambda tâche qui appellent `function1` et`function2`, une politique avec `lambda:Invoke` des autorisations pour les deux fonctions doit être utilisée.

Voici un exemple :

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

Politiques IAM pour AWS Elemental MediaConvert

Les exemples de modèles suivants montrent comment AWS Step Functions vous devez configurer vos politiques IAM en fonction des ressources contenues dans la définition de votre machine à états. Vous pouvez utiliser la console IAM pour ajouter les politiques de rôle manquantes. Pour plus d'informations, consultez Politiques IAM pour les services intégrés et Modèles d'intégration des services.

Étant donné qu'il MediaConvert fournit une prise en charge partielle du contrôle d'accès au niveau des ressources, vous devez utiliser. `"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": "*"
        }
    ]
}
```

Politiques IAM pour AWS Glue

Les exemples de modèles suivants montrent comment AWS Step Functions générer des politiques IAM en fonction des ressources contenues dans la définition de votre machine d'état. Pour plus d'informations, consultez Politiques IAM pour les services intégrés et Modèles d'intégration des services.

AWS Glue ne dispose pas d'un contrôle basé sur les ressources.

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": "*"
        }
    ]
}
```

Politiques IAM pour AWS Glue DataBrew

Les exemples de modèles suivants montrent comment AWS Step Functions générer des politiques IAM en fonction des ressources contenues dans la définition de votre machine d'état. Pour plus d'informations, consultez Politiques IAM pour les services intégrés et Modèles d'intégration des services.

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/*"
            ]
        }
    ]
}
```

Politiques IAM pour Amazon SageMaker

Les exemples de modèles suivants montrent comment AWS Step Functions générer des politiques IAM en fonction des ressources contenues dans la définition de votre machine d'état. Pour plus d'informations, consultez Politiques IAM pour les services intégrés et Modèles d'intégration des services.

> **Note**
>
> Pour ces exemples, il *[[roleArn]]* fait référence à l'Amazon Resource Name (ARN) du rôle IAM SageMaker utilisé pour accéder aux artefacts du modèle et aux images docker à des fins de déploiement sur des instances de calcul ML ou pour des tâches de transformation par lots. Pour plus d'informations, consultez Amazon SageMaker Roles.

`CreateTrainingJob`

Ressources statiques

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"
```

```
        }
      }
    }
  ]
}
```

Ressources dynamiques

.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 ne créera pas automatiquement de politique `CreateTransformJob` lorsque vous créez une machine à états intégrée à SageMaker. Vous devez associer une politique intégrée au rôle créé en vous basant sur l'un des exemples IAM suivants.

Ressources statiques

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"
        }
      }
    }
  ]
}
```

Ressources dynamiques

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"
        }
      }
    }
  ]
}
```

Politiques IAM pour Amazon SNS

Les exemples de modèles suivants montrent comment AWS Step Functions générer des politiques IAM en fonction des ressources contenues dans la définition de votre machine d'état. Pour plus d'informations, consultez Politiques IAM pour les services intégrés et Modèles d'intégration des services.

Ressources statiques

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sns:Publish"
            ],
            "Resource": [
                "arn:aws:sns:[[region]]:[[accountId]]:[[topicName]]"
            ]
        }
    ]
}
```

Ressources basées sur un chemin, ou publiées dans *TargetArn* ou *PhoneNumber*

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

```
        {
            "Effect": "Allow",
            "Action": [
                "sns:Publish"
            ],
            "Resource": "*"
        }
    ]
}
```

Politiques IAM pour Amazon SQS

Les exemples de modèles suivants montrent comment AWS Step Functions générer des politiques IAM en fonction des ressources contenues dans la définition de votre machine d'état. Pour plus d'informations, consultez Politiques IAM pour les services intégrés et Modèles d'intégration des services.

Ressources statiques

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sqs:SendMessage"
            ],
            "Resource": [
                "arn:aws:sqs:[[region]]:[[accountId]]:[[queueName]]"
            ]
        }
    ]
}
```

Ressources dynamiques

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": [
                "sqs:SendMessage"
            ],
            "Resource": "*"
        }
    ]
}
```

Politiques IAM pour AWS Step Functions

Pour une machine à états nécessitant l'exécution d'un seul flux de travail imbriqué, utilisez une politique IAM qui limite les autorisations à cette machine à états. `StartExecution`

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "states:StartExecution"
            ],
            "Resource": [

 "arn:aws:states:[[region]]:[[accountId]]:stateMachine:[[stateMachineName]]"
            ]
        }
    ]
}
```

Pour plus d’informations, consultez les ressources suivantes :

- Utilisation AWS Step Functions avec d'autres services
- Transmettre des paramètres à une API de service
- Gérez AWS Step Functions les exécutions en tant que service intégré

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]]"
            ]
        }
    ]
}
```

Pour de plus amples informations sur les exécutions de workflows imbriqués, veuillez consulter Démarrer les exécutions de flux de travail à partir d'un état de tâche.

Politiques IAM pour AWS X-Ray

Les exemples de modèles suivants montrent comment AWS Step Functions générer des politiques IAM en fonction des ressources contenues dans la définition de votre machine d'état. Pour plus d'informations, consultez Politiques IAM pour les services intégrés et Modèles d'intégration des services.

Pour activer le suivi X-Ray, vous aurez besoin d'une politique IAM avec les autorisations appropriées pour autoriser le suivi. Si votre machine d'état utilise d'autres services intégrés, vous aurez peut-être besoin de politiques IAM supplémentaires. Consultez les politiques IAM relatives à vos intégrations de services spécifiques.

Lorsque vous créez une machine à états avec le suivi X-Ray activé, une politique IAM est automatiquement créée.

> **Note**
>
> Si vous activez le suivi X-Ray pour une machine à états existante, vous devez vous assurer d'ajouter une politique avec des autorisations suffisantes pour activer le traçage X-Ray.

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

```
            "Effect": "Allow",
            "Action": [
                "xray:PutTraceSegments",
                "xray:PutTelemetryRecords",
                "xray:GetSamplingRules",
                "xray:GetSamplingTargets"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Pour plus d'informations sur l'utilisation de X-Ray with Step Functions, consultezAWS X-Ray et Step Functions.

Activités ou aucune tâche

Pour une machine à états qui n'a que `Activity` des tâches, ou aucune tâche du tout, utilisez une politique IAM qui refuse l'accès à toutes les actions et ressources.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "*",
            "Resource": "*"
        }
    ]
}
```

Pour de plus amples informations sur l'utilisation des tâches `Activity` , veuillez consulter Activités.

Politiques IAM pour l'utilisation de l'état de la carte distribuée

Lorsque vous créez des flux de travail avec la console Step Functions, Step Functions peut générer automatiquement des politiques IAM en fonction des ressources figurant dans votre définition de flux de travail. Ces politiques incluent le minimum de privilèges nécessaires pour permettre au rôle

de machine d'état d'invoquer l'action d'`StartExecution`API pour l'état de la carte distribuée. Ces politiques incluent également le minimum de privilèges nécessaires aux Step Functions pour accéder aux AWS ressources, telles que les buckets et les objets Amazon S3 et les fonctions Lambda. Nous vous recommandons vivement de n'inclure que les autorisations nécessaires dans vos politiques IAM. Par exemple, si votre flux de travail inclut un `Map` état en mode distribué, limitez vos politiques au compartiment et au dossier Amazon S3 spécifiques qui contiennent votre ensemble de données.

> **Important**
>
> Si vous spécifiez un compartiment et un objet Amazon S3, ou un préfixe, avec un chemin de référence vers une paire clé-valeur existante dans l'entrée d'état de votre carte distribuée, assurez-vous de mettre à jour les politiques IAM pour votre flux de travail. Élargissez les politiques jusqu'au bucket et aux noms d'objets auxquels le chemin aboutit au moment de l'exécution.

Dans cette rubrique :

- Exemple de politique IAM pour exécuter un état de carte distribuée
- Exemple de politique IAM pour redriving une carte distribuée
- Exemples de politiques IAM pour lire les données des ensembles de données Amazon S3
- Exemple de politique IAM pour écrire des données dans un compartiment Amazon S3

Exemple de politique IAM pour exécuter un état de carte distribuée

Lorsque vous incluez un état de carte distribuée dans vos flux de travail, Step Functions a besoin des autorisations appropriées pour permettre au rôle de machine à états d'invoquer l'action d'`StartExecution`API pour l'état de carte distribuée.

L'exemple de politique IAM suivant accorde le minimum de privilèges requis à votre rôle de machine d'état pour exécuter l'état de carte distribuée.

> **Note**
>
> Assurez-vous de *`stateMachineName`* remplacer par le nom de la machine à états dans laquelle vous utilisez l'état Distributed Map. Par exemple, `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:*"
    }
  ]
}
```

Exemple de politique IAM pour redriving une carte distribuée

Vous pouvez redémarrer les exécutions infructueuses d'un flux de travail enfant dans un flux de travail Map Run par redrivingvotre flux de travail parent. Un flux de travail redriven parent redrives contenant tous les états infructueux, y compris la carte distribuée. Assurez-vous que votre rôle d'exécution dispose du minimum de privilèges nécessaires pour lui permettre d'invoquer l'action d'`RedriveExecution`API sur le flux de travail parent.

L'exemple de politique IAM suivant accorde le minimum de privilèges requis à votre rôle de machine d'état pour redriving un état de carte distribuée.

> **Note**
>
> Assurez-vous de *`stateMachineName`* remplacer par le nom de la machine à états dans laquelle vous utilisez l'état Distributed Map. Par exemple, `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:*"
    }
  ]
}
```

Exemples de politiques IAM pour lire les données des ensembles de données Amazon S3

Les exemples de politique IAM suivants accordent le minimum de privilèges requis pour accéder à vos ensembles de données Amazon S3 à l'aide des actions ListObjectsV2 et GetObjectAPI.

Example Politique IAM pour les objets Amazon S3 en tant que jeu de données

L'exemple suivant montre une politique IAM qui accorde le moins de privilèges pour accéder aux objets organisés *processImages* dans un compartiment Amazon S3 nommé*myBucket*.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::myBucket"
            ],
            "Condition": {
                "StringLike": {
                    "s3:prefix": [
                        "processImages"
                    ]
                }
```

```
            }
        }
    ]
}
```

Example Politique IAM pour un fichier CSV en tant que jeu de données

L'exemple suivant montre une politique IAM qui accorde le moins de privilèges pour accéder à un fichier CSV nommé*ratings.csv*.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::myBucket/csvDataset/ratings.csv"
            ]
        }
    ]
}
```

Example Politique IAM pour un inventaire Amazon S3 sous forme de jeu de données

L'exemple suivant montre une politique IAM qui accorde le moins de privilèges pour accéder à un rapport d'inventaire Amazon S3.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::destination-prefix/source-bucket/config-ID/YYYY-MM-DDTHH-
MMZ/manifest.json",
                "arn:aws:s3:::destination-prefix/source-bucket/config-ID/data/*"
```

```
            ]
        }
    ]
}
```

Exemple de politique IAM pour écrire des données dans un compartiment Amazon S3

L'exemple de politique IAM suivant accorde le minimum de privilèges requis pour écrire les résultats de l'exécution du flux de travail de votre enfant dans un dossier nommé *CSVjobs* dans un compartiment Amazon S3 à l'aide de l'PutObjectaction API.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListMultipartUploadParts",
                "s3:AbortMultipartUpload"
            ],
            "Resource": [
                "arn:aws:s3:::resultBucket/csvJobs/*"
            ]
        }
    ]
}
```

Autorisations IAM pour le compartiment Amazon S3 AWS KMS key chiffré

L'état de la carte distribuée utilise des téléchargements partitionnés pour écrire les résultats de l'exécution du flux de travail enfant dans un compartiment Amazon S3. Si le compartiment est chiffré à l'aide d'une AWS Key Management Service (AWS KMS) clé, vous devez également inclure des autorisations dans votre IAM politique pour effectuer les `kms:GenerateDataKey` actions `kms:Decryptkms:Encrypt`, et sur la clé. Ces autorisations sont requises, car Simple Storage Service (Amazon S3) doit déchiffrer et lire les données des parties de fichier chiffrées avant de terminer le chargement partitionné.

L'exemple de politique IAM suivant accorde l'autorisation à `kms:Decryptkms:Encrypt`, et aux `kms:GenerateDataKey` actions sur la clé utilisée pour chiffrer votre compartiment Amazon S3.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "kms:Decrypt",
      "kms:Encrypt",
      "kms:GenerateDataKey"
    ],
    "Resource": [
      "arn:aws:kms:us-east-1:123456789012:key/111aa2bb-333c-4d44-5555-a111bb2c33dd"
    ]
  }
}
```

Pour plus d'informations, consultez Chargement d'un fichier volumineux vers Amazon S3 avec chiffrement à l'aide d'une AWS KMS key que vous trouverez dans le AWS Centre de connaissances.

Si votre utilisateur ou votre rôle IAM est Compte AWS identique auKMS key, vous devez disposer de ces autorisations sur la politique clé. Si votre utilisateur ou rôle IAM appartient à un compte différent duKMS key, vous devez disposer des autorisations à la fois sur la politique clé et sur votre utilisateur ou rôle IAM.

Stratégies basées sur des balises

Step Functions prend en charge les politiques basées sur les balises. Par exemple, vous pouvez restreindre l'accès à toutes les ressources Step Functions qui incluent une balise avec la clé `environment` et la valeur`production`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "states:TagResource",
                "states:UntagResource",
                "states:DeleteActivity",
                "states:DeleteStateMachine",
                "states:StopExecution"
            ],
```

```
            "Resource": "*",
            "Condition": {
                "StringEquals": {"aws:ResourceTag/environment": "production"}
            }
        }
    ]
}
```

Cette stratégie refusera (`Deny`) la possibilité de supprimer des machines d'état ou des activités, d'arrêter des exécutions et d'ajouter ou de supprimer de nouvelles balises pour toutes les ressources qui ont été balisées en tant que `environment/production`.

Pour l'autorisation basée sur des balises, les ressources d'exécution de la machine d'état, comme indiqué dans l'exemple suivant, héritent des balises associées à une machine d'état.

```
arn:<partition>:states:<Region>:<account-id>:execution:<StateMachineName>:<ExecutionId>
```

Lorsque vous appelez DescribeExecutionou utilisez d'autres API dans lesquelles vous spécifiez l'ARN de la ressource d'exécution, Step Functions utilise les balises associées à la machine d'état pour accepter ou refuser la demande tout en effectuant une autorisation basée sur des balises. Cela vous permet d'autoriser ou de refuser l'accès aux exécutions par machine d'état au niveau de la machine d'état.

Pour plus d'informations sur le balisage, consultez les ressources suivantes :

- Fonctions de balisage en étapes
- Contrôle de l'accès à l'aide de balises IAM

Résolution des problèmes AWS Step Functions d'identité et d'accès

Utilisez les informations suivantes pour vous aider à diagnostiquer et à résoudre les problèmes courants que vous pouvez rencontrer lors de l'utilisation de Step Functions et d'IAM.

Rubriques

- Je ne suis pas autorisé à effectuer une action dans Step Functions
- Je ne suis pas autorisé à effectuer iam : PassRole
- Je souhaite permettre à des personnes extérieures Compte AWS à moi d'accéder à mes ressources Step Functions

Je ne suis pas autorisé à effectuer une action dans Step Functions

Si vous recevez une erreur selon laquelle vous n'êtes pas autorisé à effectuer une action, vos stratégies doivent être mises à jour afin de vous permettre d'effectuer l'action.

L'exemple d'erreur suivant se produit quand l'utilisateur `mateojackson` tente d'utiliser la console pour afficher des informations détaillées sur une ressource *`my-example-widget`* fictive, mais ne dispose pas des autorisations `states:`*`GetWidget`* fictives.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 states:GetWidget on resource: my-example-widget
```

Dans ce cas, la stratégie de Mateo doit être mise à jour pour l'autoriser à accéder à la ressource *`my-example-widget`* à l'aide de l'action `states:`*`GetWidget`*.

Si vous avez besoin d'aide, contactez votre AWS administrateur. Votre administrateur vous a fourni vos informations d’identification de connexion.

Je ne suis pas autorisé à effectuer iam : PassRole

Si vous recevez un message d'erreur indiquant que vous n'êtes pas autorisé à effectuer l'`iam:PassRole`action, vos politiques doivent être mises à jour pour vous permettre de transmettre un rôle à Step Functions.

Certains vous Services AWS permettent de transmettre un rôle existant à ce service au lieu de créer un nouveau rôle de service ou un rôle lié à un service. Pour ce faire, un utilisateur doit disposer des autorisations nécessaires pour transmettre le rôle au service.

L'exemple d'erreur suivant se produit lorsqu'un utilisateur IAM nommé `marymajor` essaie d'utiliser la console pour effectuer une action dans Step Functions. Toutefois, l’action nécessite que le service ait des autorisations accordées par un rôle de service. Mary ne dispose pas des autorisations nécessaires pour transférer le rôle au service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Dans ce cas, les politiques de Mary doivent être mises à jour pour lui permettre d’exécuter l’action `iam:PassRole`.

Si vous avez besoin d'aide, contactez votre AWS administrateur. Votre administrateur vous a fourni vos informations d’identification de connexion.

Je souhaite permettre à des personnes extérieures Compte AWS à moi d'accéder à mes ressources Step Functions

Vous pouvez créer un rôle que les utilisateurs provenant d'autres comptes ou les personnes extérieures à votre organisation pourront utiliser pour accéder à vos ressources. Vous pouvez spécifier qui est autorisé à assumer le rôle. Pour les services qui prennent en charge les politiques basées sur les ressources ou les listes de contrôle d'accès (ACL), vous pouvez utiliser ces politiques pour donner l'accès à vos ressources.

Pour en savoir plus, consultez les éléments suivants :

- Pour savoir si Step Functions prend en charge ces fonctionnalités, consultezComment AWS Step Functions fonctionne avec IAM.
- Pour savoir comment fournir l'accès à vos ressources sur celles Comptes AWS que vous possédez, consultez la section Fournir l'accès à un utilisateur IAM dans un autre utilisateur Compte AWS que vous possédez dans le Guide de l'utilisateur IAM.
- Pour savoir comment fournir l'accès à vos ressources à des tiers Comptes AWS, consultez la section Fournir un accès à des ressources Comptes AWS détenues par des tiers dans le guide de l'utilisateur IAM.
- Pour savoir comment fournir un accès par le biais de la fédération d'identité, consultez Fournir un accès à des utilisateurs authentifiés en externe (fédération d'identité) dans le Guide de l'utilisateur IAM.
- Pour découvrir quelle est la différence entre l'utilisation des rôles et l'utilisation des politiques basées sur les ressources pour l'accès entre comptes, consultez Différence entre les rôles IAM et les politiques basées sur les ressources dans le Guide de l'utilisateur IAM.

Journalisation et surveillance

Pour plus d'informations sur la connexion et la surveillance AWS Step Functions, consultez la Journalisation et surveillance section.

Validation de conformité pour AWS Step Functions

Des auditeurs tiers évaluent la sécurité et AWS Step Functions la conformité de plusieurs programmes de AWS conformité. Il s'agit notamment des certifications SOC, PCI, FedRAMP, HIPAA et d'autres.

Pour une liste des AWS services concernés par des programmes de conformité spécifiques, voir AWS Services concernés par programme de conformitéAWS . Pour des informations générales, voir Programmes de AWS conformité Programmes AWS de .

Vous pouvez télécharger des rapports d'audit tiers à l'aide de AWS Artifact. Pour plus d'informations, voir Téléchargement de rapports dans AWS Artifact .

Lorsque vous utilisez Step Functions, votre responsabilité en matière de conformité dépend de la sensibilité de vos données, des objectifs de conformité de votre entreprise et des lois et réglementations applicables. AWS fournit les ressources suivantes pour faciliter la mise en conformité :

- Guides de démarrage rapide sur la sécurité et la conformité Guides sur la sécurité et la conformité — Ces guides de déploiement abordent les considérations architecturales et fournissent les étapes à suivre pour déployer des environnements de base axés sur la sécurité et la conformité sur. AWS
- Architecture axée sur la sécurité et la conformité HIPAA sur Amazon Web Services : ce livre blanc décrit comment les entreprises peuvent créer des applications AWS conformes à la loi HIPAA.
- AWS Ressources de https://aws.amazon.com/compliance/resources/ de conformité — Cette collection de classeurs et de guides peut s'appliquer à votre secteur d'activité et à votre région.
- Évaluation des ressources à l'aide des règles du guide du AWS Config développeur : le AWS Config service évalue dans quelle mesure les configurations de vos ressources sont conformes aux pratiques internes, aux directives du secteur et aux réglementations.
- AWS Security Hub— Ce AWS service fournit une vue complète de l'état de votre sécurité interne, AWS ce qui vous permet de vérifier votre conformité aux normes et aux meilleures pratiques du secteur de la sécurité.

Résilience dans AWS Step Functions

L'infrastructure AWS mondiale est construite autour des AWS régions et des zones de disponibilité. AWS Les régions fournissent plusieurs zones de disponibilité physiquement séparées et isolées, connectées par un réseau à faible latence, à haut débit et hautement redondant. Avec les zones de disponibilité, vous pouvez concevoir et exploiter des applications et des bases de données qui basculent automatiquement d’une zone à l’autre sans interruption. Les zones de disponibilité sont davantage disponibles, tolérantes aux pannes et ont une plus grande capacité de mise à l’échelle que les infrastructures traditionnelles à un ou plusieurs centres de données.

Pour plus d'informations sur AWS les régions et les zones de disponibilité, consultez la section Infrastructure AWS mondiale.

Outre l'infrastructure AWS mondiale, Step Functions propose plusieurs fonctionnalités pour répondre à vos besoins en matière de résilience et de sauvegarde des données.

Sécurité de l'infrastructure dans AWS Step Functions

En tant que service géré, AWS Step Functions il est protégé par la sécurité du réseau AWS mondial. Pour plus d'informations sur les services AWS de sécurité et sur la manière dont AWS l'infrastructure est protégée, consultez la section Sécurité du AWS cloud. Pour concevoir votre AWS environnement en utilisant les meilleures pratiques en matière de sécurité de l'infrastructure, consultez la section Protection de l'infrastructure dans le cadre AWS bien architecturé du pilier de sécurité.

Vous utilisez des appels d'API AWS publiés pour accéder Step Functions via le réseau. Les clients doivent prendre en charge les éléments suivants :

- Protocole TLS (Transport Layer Security). Nous exigeons TLS 1.2 et recommandons TLS 1.3.
- Ses suites de chiffrement PFS (Perfect Forward Secrecy) comme DHE (Ephemeral Diffie-Hellman) ou ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). La plupart des systèmes modernes tels que Java 7 et les versions ultérieures prennent en charge ces modes.

En outre, les demandes doivent être signées à l’aide d’un ID de clé d’accès et d’une clé d’accès secrète associée à un principal IAM. Vous pouvez également utiliser AWS Security Token Service (AWS STS) pour générer des informations d’identification de sécurité temporaires et signer les demandes.

Vous pouvez appeler les opérations d' AWS API depuis n'importe quel emplacement réseau, mais les politiques d'accès basées sur les ressources Step Functions ne sont pas prises en charge, qui peuvent inclure des restrictions basées sur l'adresse IP source. Vous pouvez utiliser des stratégies Step Functions pour contrôler l'accès depuis des points de terminaison Amazon Virtual Private Cloud (Amazon VPC) spécifiques ou depuis des VPC spécifiques. En fait, cela isole l'accès réseau à une Step Functions ressource donnée uniquement du VPC spécifique au sein AWS du réseau.

Analyse de configuration et de vulnérabilité dans AWS Step Functions

La configuration et les contrôles informatiques sont une responsabilité partagée entre vous AWS et vous, notre client. Pour plus d'informations, consultez le modèle de responsabilité AWS partagée.

Migration des charges de travail depuis Step AWS Data Pipeline Functions

AWS a lancé le AWS Data Pipeline service en 2012. À l'époque, les clients recherchaient un service leur permettant d'utiliser diverses options de calcul pour déplacer des données entre différentes sources de données. Les besoins en matière de transfert de données ayant évolué au fil du temps, les solutions à ces besoins ont également évolué. Vous avez désormais la possibilité de choisir la solution qui répond le mieux aux besoins de votre entreprise. Par exemple, vous pouvez effectuer l'une des opérations suivantes :

- Utilisez Step Functions pour orchestrer des flux de travail entre plusieurs Services AWS.
- Utilisez Amazon Managed Workflows for Apache Airflow (Amazon MWAA) pour gérer l'orchestration des flux de travail pour Apache Airflow.
- AWS Glue À utiliser pour exécuter et orchestrer les applications Apache Spark.

Vous pouvez migrer les cas d' AWS Data Pipeline utilisation typiques vers Step Functions ou Amazon MWAA. AWS Glue L'option que vous choisissez dépend de votre charge de travail actuelle AWS Data Pipeline. Cette rubrique explique comment migrer de Step Functions AWS Data Pipeline vers Step Functions.

Rubriques

Migration des charges de travail depuis AWS Data Pipeline

Step Functions est un service d'orchestration sans serveur qui vous permet de créer des flux de travail pour des applications critiques pour l'entreprise. Avec Workflow Studio de Step Functions, vous pouvez créer des flux de travail et les intégrer à plus de 11 000 actions d'API parmi plus de 250. Services AWS Cela inclut Services AWS notamment AWS Lambda Amazon EMR et Amazon DynamoDB. Vous pouvez également utiliser Step Functions pour orchestrer des pipelines de traitement des données, gérer les erreurs et utiliser des limites de régulation sur le sous-

jacent. Services AWS Vous pouvez créer des flux de travail qui traitent et publient des modèles d'apprentissage automatique, orchestrent des microservices et gèrent des flux de travail d'extraction, de transformation et de chargement (ETL) avec. AWS Glue Vous pouvez également créer des flux de travail automatisés de longue durée pour les applications qui nécessitent une interaction humaine.

Step Functions est un service entièrement géré fourni par AWS. Cela signifie qu'il AWS gère des tâches telles que la maintenance de l'infrastructure, l'application de correctifs aux travailleurs et la gestion des mises à jour des versions du système d'exploitation pour vous.

Lorsque votre cas d'utilisation répond aux conditions suivantes, nous vous recommandons de passer AWS Data Pipeline à Step Functions :

- Vous préférez un service d'orchestration de flux de travail sans serveur à haute disponibilité.
- Vous avez besoin d'une solution qui se base sur la granularité de l'exécution d'une seule tâche.
- Vos charges de travail impliquent l'orchestration de tâches pour plusieurs autres entreprises Services AWS, telles qu'Amazon EMR, Lambda AWS Glue ou DynamoDB.
- Vous avez besoin d'une solution low-code avec un concepteur drag-and-drop visuel pour la création de flux de travail. Cette solution ne devrait pas nécessiter l'apprentissage de concepts de programmation complexes et inconnus.
- Vous avez besoin d'un service qui s'intègre à plus de 250 Services AWS applications couvrant plus de 11 000 actions d'API. Ce service doit également s'intégrer aux services et activités personnalisés extérieurs à AWS.

Cartographie conceptuelle entre Step Functions et AWS Data Pipeline

AWS Data Pipeline et Step Functions partagent certains concepts communs. Par exemple, pour définir vos flux de travail, vous utilisez le format JSON à la fois dans Step Functions AWS Data Pipeline et dans Step Functions. Dans Step FunctionsAmazon States Language, vous utilisez un langage structuré basé sur JSON. Vous utilisez Amazon States Language (ASL) pour définir vos flux de travail et alterner entre les représentations textuelles et visuelles de votre flux de travail. Ce format basé sur JSON permet de simplifier le stockage de vos flux de travail dans un outil de contrôle de source. Il vous permet également de gérer plusieurs versions de vos flux de travail, de contrôler leur accès ou d'automatiser leur orchestration à l'aide de méthodes CI/CD.

Le tableau suivant décrit le mappage entre les principaux concepts utilisés dans les deux services. La colonne des concepts du pipeline de données sur la gauche répertorie les concepts dans AWS

Data Pipeline, tandis que la colonne des concepts Step Functions sur la droite répertorie les concepts équivalents dans Step Functions.

Concepts de pipeline de données	Concepts de Step Functions
Pipelines	Flux de travail
Définition du pipeline	Amazon States Language(ASL)
Activités	States et État de la tâche
instances	Exécutions
Tentatives	Catchers et retriers
Calendrier du pipeline	• Exécutions avec Amazon EventBridge Scheduler • Événements déclenchés par le biais de EventBridge Pipes
Expressions et fonctions du pipeline	• Fonctions intrinsèques • Fonctions Lambda utilisant l'intégration de services

Exemples de projets Step Functions

Pour une présentation de Step Functions, regardez la vidéo suivante :

Commencer à utiliser AWS Step Functions pour l'orchestration des services

La liste suivante présente quelques exemples de projets qui implémentent les cas d' AWS Data Pipeline utilisation les plus courants avec Step Functions. Vous pouvez utiliser ces exemples de projets comme référence pour AWS Data Pipeline migrer depuis Step Functions. Vous pouvez également les utiliser comme modèle pour créer vos propres flux de travail et les intégrer à ceux pris en charge en Services AWS fonction de votre cas d'utilisation.

- Gérer une offre d'emploi Amazon EMR
- Exécuter une tâche de traitement de données sur Amazon EMR Serverless

- Exécution de jobs Hive/Pig/Hadoop
- Interrogez de grands ensembles de données (Amazon Athena, Amazon S3 AWS Glue, Amazon SNS)
- Exécutez des flux de travail ETL/ELT à l'aide d'Amazon Redshift
- Orchestration AWS Glue des crawlers
- Exécuter un script shell avec Step Functions

Pour en savoir plus sur Step Functions, consultez les rubriques et ressources suivantes :

- Tutoriels pour Step Functions
- Exemples de projets pour Step Functions
- L' AWS Step Functions atelier

Comparaison des prix

AWS Data Pipeline est établi en fonction du nombre de pipelines et de leur niveau d'utilisation. Les activités organisées plus d'une fois par jour (fréquence élevée) sont facturées 1$ par mois et par activité. Les activités organisées une fois par jour ou moins (basse fréquence) sont proposées au prix de 0,60$ par mois et par activité. Le prix des pipelines inactifs est de 1$ par pipeline. Pour plus d'informations sur les tarifs, consultez la page AWS Data Pipeline des tarifs.

Step Functions propose deux types de flux de travail : Standard et Express. Chaque type de flux de travail possède un modèle de tarification différent. Cette comparaison est basée sur le flux de travail standard, car il correspond le mieux aux cas d'utilisation courants de AWS Data Pipeline. Les flux de travail standard sont proposés au prix de 0,025$ pour 1 000 transitions d'état. Les machines d'état inactives sont gratuites ; vous ne payez que pour ce que vous utilisez. Pour plus d'informations sur les tarifs, consultez la page AWS Step Functions des tarifs.

Résolution des problèmes

Si vous rencontrez des difficultés lors de l'utilisation de Step Functions, utilisez les ressources de dépannage suivantes.

Rubriques

Résolution de problème généraux

Je ne parviens pas à créer une machine d'État.

Le rôle IAM associé à la machine d'état ne dispose peut-être pas d'autorisations suffisantes. Vérifiez les autorisations du rôle IAM, notamment pour les tâches d'intégration des AWS services, X-Ray et la CloudWatch journalisation. Des autorisations supplémentaires sont requises pour les états des `.sync` tâches.

Je ne parviens pas à utiliser a JsonPath pour référencer la sortie de la tâche précédente.

Pour aJsonPath, une clé JSON doit se terminer par`.$`. Cela signifie que a ne JsonPath peut être utilisé que dans une paire clé-valeur. Si vous souhaitez utiliser un JsonPath autre emplacement, tel qu'un tableau, vous pouvez utiliser des fonctions intrinsèques. Par exemple, vous pouvez utiliser quelque chose de similaire à ce qui suit :

Résultat de la tâche A :

```
{
    "sample": "test"
}
```

Tâche B :

```
{
    "JsonPathSample.$": "$.sample"
}
```

> **Tip**
>
> Utilisez le simulateur de flux de données de la console Step Functions pour tester la syntaxe du chemin JSON, pour mieux comprendre comment les données sont manipulées au sein d'un état et pour voir comment les données sont transmises entre les états.

Il y a eu un retard dans les transitions entre les États.

Pour les flux de travail standard, le nombre de transitions d'état est limité. Lorsque vous dépassez la limite de transition d'état, Step Functions retarde les transitions d'état jusqu'à ce que le bucket correspondant au quota soit rempli. La limitation des limites de transition d'état peut être surveillée en consultant la `ExecutionThrottled` métrique dans la Métriques d'exécution section de la page CloudWatch Mesures.

Lorsque je lance de nouvelles exécutions de flux de travail standard, elles échouent avec l'**`ExecutionLimitExceeded`**erreur.

Step Functions a une limite de 1 000 000 d'exécutions ouvertes pour chacune Compte AWS d'entre ellesRégion AWS. Si vous dépassez cette limite, Step Functions génère une `ExecutionLimitExceeded` erreur. Cette limite ne s'applique pas aux flux de travail Express. Vous pouvez utiliser les calculs de CloudWatch métriques suivants dans le guide de CloudWatch l'utilisateur d'Amazon pour estimer le nombre d'exécutions ouvertes : `ExecutionsStarted - (ExecutionsSucceeded + ExecutionsTimedOut + ExecutionsFailed + ExecutionsAborted)`

Une défaillance sur une branche dans un état parallèle entraîne l'échec de l'ensemble de l'exécution.

Il s'agit d'un comportement attendu. Pour éviter de rencontrer des défaillances lors de l'utilisation d'un état parallèle, configurez Step Functions de manière à détecter les erreurs provenant de chaque branche.

Résolution des problèmes liés aux intégrations de services

Ma tâche est terminée dans le service en aval, mais dans Step Functions, l'état de la tâche reste « En cours » ou son achèvement est retardé.

Pour les modèles d'intégration des `.sync` services, Step Functions utilise des EventBridge règles, des API en aval ou une combinaison des deux pour détecter l'état des tâches en aval. Pour certains services, Step Functions ne crée pas de EventBridge règles à surveiller. Par exemple, pour l'intégration des AWS Glue services, Step Functions passe un `glue:GetJobRun` appel au lieu d'utiliser des EventBridge règles. En raison de la fréquence des appels d'API, il existe une différence entre le temps d'exécution de la tâche en aval et le temps d'exécution de la tâche Step Functions. Step Functions nécessite des autorisations IAM pour gérer les EventBridge règles et passer des appels vers le service en aval. Pour plus d'informations sur la manière dont des autorisations insuffisantes sur votre rôle d'exécution peuvent affecter l'exécution des tâches, consultezAutorisations supplémentaires pour les tâches utilisant le modèle Run a Job.

Je souhaite renvoyer une sortie JSON à partir d'une exécution de machine à états imbriqués.

Il existe deux intégrations de services synchrones Step Functions pour Step Functions : `startExecution.sync` et. `startExecution.sync:2` Les deux attendent que la machine à états imbriqués soit terminée, mais ils renvoient des `Output` formats différents. Vous pouvez l'utiliser `startExecution.sync:2` pour renvoyer une sortie JSON sous`Output`.

Je n'arrive pas à invoquer une fonction Lambda depuis un autre compte.

Accès à la fonction Lambda avec prise en charge multicompte

Si l'accès aux AWS ressources entre comptes est disponible dans votre région, utilisez la méthode suivante pour appeler une fonction Lambda depuis un autre compte.

Pour appeler une ressource multicompte dans vos flux de travail, procédez comme suit :

1. Créez un rôle IAM dans le compte cible qui contient la ressource. Ce rôle accorde au compte source, qui contient la machine d'état, l'autorisation d'accéder aux ressources du compte cible.
2. Dans la définition de `Task` l'État, spécifiez le rôle IAM cible que doit assumer la machine d'État avant d'appeler la ressource multicomptes.

3. Modifiez la politique de confiance dans le rôle IAM cible pour permettre au compte source d'assumer temporairement ce rôle. La politique de confiance doit inclure le nom de ressource Amazon (ARN) de la machine d'état définie dans le compte source. Définissez également les autorisations appropriées dans le rôle IAM cible pour appeler la AWS ressource.
4. Mettez à jour le rôle d'exécution du compte source pour inclure l'autorisation requise pour assumer le rôle IAM cible.

Pour voir un exemple, consultez Tutoriel : Accès aux ressources multicomptes AWS.

> **Note**
>
> Vous pouvez configurer votre machine d'État pour qu'elle assume un rôle IAM pour accéder à des ressources provenant de plusieurs Comptes AWS sources. Toutefois, une machine d'état ne peut assumer qu'un seul rôle IAM à la fois.

Pour un exemple de définition d'`Task`état qui spécifie une ressource multicompte, consultezExemples de champs d'informations d'identification de l'état de la tâche.

Accès à la fonction Lambda sans prise en charge multicompte

Si l'accès aux AWS ressources entre comptes n'est pas disponible dans votre région, utilisez la méthode suivante pour appeler une fonction Lambda depuis un autre compte.

Dans le `Resource` champ `Task` État, utilisez `arn:aws:states:::lambda:invoke` et transmettez les paramètres `FunctionArn` d'entrée. Le rôle IAM associé à la machine d'état doit disposer des autorisations appropriées pour invoquer des fonctions Lambda multicomptes :. `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
```

```
      }
   }
}
```

Je ne parviens pas à voir les jetons de tâche transmis par **`.waitForTaskToken`** les États.

Dans le `Parameters` champ `Task` État, vous devez transmettre un jeton de tâche. Par exemple, vous pouvez utiliser un code similaire au code suivant.

```
{
   "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**
>
> Vous pouvez essayer de l'utiliser `.waitForTaskToken` avec n'importe quelle action d'API. Cependant, certaines API ne disposent pas de paramètres appropriés.

Activités de résolution des problèmes

L'exécution de ma machine d'état est bloquée à un état d'activité.

L'état d'une tâche d'activité ne démarre que lorsque vous interrogez un jeton de tâche à l'aide de l'action de l'GetActivityTaskAPI. La meilleure pratique consiste à ajouter un délai d'expiration au

niveau de la tâche afin d'éviter un blocage de l'exécution. Pour plus d'informations, veuillez consulter Utilisez les délais d'attente pour éviter les exécutions bloquées.

Si votre machine d'État est bloquée lors de l'ActivityScheduledévénement, cela indique que votre parc de travailleurs présente des problèmes ou est sous-dimensionné. Vous devez surveiller la `ActivityScheduleTime`CloudWatchmétrique et régler une alarme lorsque ce délai augmente. Toutefois, pour expirer les exécutions bloquées de la machine à état pendant laquelle l'`Activity`état ne passe pas à l'`ActivityStarted`état, définissez un délai d'expiration au niveau de la machine d'état. Pour ce faire, spécifiez un `TimeoutSeconds` champ au début de la définition de la machine d'état, en dehors du `States` champ.

Mon agent d'activité expire en attendant un jeton de tâche.

Les travailleurs utilisent l'action de l'GetActivityTaskAPI pour récupérer une tâche avec l'ARN d'activité spécifié qui est planifiée pour être exécutée par une machine en cours d'exécution. `GetActivityTask`lance un long sondage afin que le service garde la connexion HTTP ouverte et réponde dès qu'une tâche est disponible. La durée maximale pendant laquelle le service conserve la demande avant de répondre est de 60 secondes. Si aucune tâche n'est disponible dans les 60 secondes, le sondage renvoie a `taskToken` avec une chaîne nulle. Pour éviter ce délai, configurez un socket côté client avec un délai d'au moins 65 secondes dans le AWS SDK ou dans le client que vous utilisez pour effectuer l'appel d'API.

Résolution des problèmes avec Express Workflows

Mon application expire avant de recevoir une réponse d'un appel d'`StartSyncExecution`API.

Configurez un délai d'expiration du socket côté client dans le AWS SDK ou le client que vous utilisez pour effectuer l'appel d'API. Pour recevoir une réponse, le délai d'attente doit avoir une valeur supérieure à la durée des exécutions d'Express Workflow.

Je ne parviens pas à consulter l'historique des exécutions afin de résoudre les défaillances d'Express Workflow.

Express Workflows n'enregistre pas l'historique des exécutions dansAWS Step Functions. Au lieu de cela, vous devez activer la CloudWatch journalisation. Une fois la journalisation activée, vous pouvez utiliser les requêtes CloudWatch Logs Insights pour passer en revue vos exécutions

d'Express Workflow. Vous pouvez également consulter l'historique des exécutions d'Express Workflow sur la console Step Functions en cliquant sur le bouton Activer dans l'onglet Exécutions. Pour plus d'informations, veuillez consulter Affichage et débogage des exécutions sur la console Step Functions.

Pour répertorier les exécutions en fonction de leur durée :

```
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
```

Pour répertorier les exécutions qui ont échoué et qui ont été annulées :

```
fields ispresent(execution_arn) as isRes | filter type in ["ExecutionFailed",
 "ExecutionAborted", "ExecutionTimedOut"]
```

Informations connexes

Le tableau suivant répertorie les ressources connexes qui peuvent vous être utiles lors de l'utilisation de ce service.

Ressource	Description
Référence API AWS Step Functions	Descriptions des actions, paramètres et types de données d'API, et liste des erreurs renvoyées par le service.
AWS Step FunctionsRéférence de ligne de commande	Descriptions des commandes de l'AWS CLI que vous pouvez utiliser avec AWS Step Functions.
Informations sur le produit pour Step Functions	La page Web principale contenant des informations sur Step Functions.
Forums de discussion	Un forum communautaire permettant aux développeurs de discuter de questions techniques liées à Step Functions et à d'autres AWS services.
AWS SupportInformations	La principale page Web contenant des informations sur AWS Support un canal d'ne-on-oneassistance à réponse rapide qui vous aide à créer et à exécuter des applications sur des services AWS d'infrastructure.

Derniers lancements de fonctionnalités

Le tableau suivant répertorie les régions dans lesquelles les nouvelles fonctionnalités de Step Functions sont disponibles.

Date de lancement	Nom de la fonctionnalité	Régions disponibles
26 novembre 2023	Appelez des points de terminaison HTTPS publics et testez des états individuels	• USA Est (Virginie du Nord) – us-east-1 • USA Ouest (Oregon) – us-west-2 • USA Est (Ohio) – us-east-2 • Europe (Irlande) – eu-west-1 • Europe (Francfort) – eu-central-1 • Europe (Stockholm) – eu-north-1 • Asie-Pacifique (Sydney) – ap-southeast-2 • Asie-Pacifique (Tokyo) – ap-northeast-1 • Asie-Pacifique (Singapour) – ap-southeast-1
15 novembre 2023	Redriveexécutions	Pour une liste complète des pays Régions AWS dans lesquels cette fonctionnalité est disponible, consultez les options de la liste déroulant e Région sur la page intitulée Services AWSPar Région.
12 octobre 2023	Intégration optimisée pour Amazon EMR Serverless	Pour une liste complète des pays Régions AWS dans lesquels cette fonctionnalité

Date de lancement	Nom de la fonctionnalité	Régions disponibles
		est disponible, consultez les options de la liste déroulant e Région sur la page intitulée Services AWSPar Région.
07 septembre 2023	Gestion améliorée des erreurs	Pour une liste complète des pays Régions AWS dans lesquels cette fonctionnalité est disponible, consultez les options de la liste déroulant e Région sur la page intitulée Services AWSPar Région.
31 août 2023	Améliorations apportées à Workflow Studio pour une expérience de création rationalisée	Pour une liste complète des pays Régions AWS dans lesquels cette fonctionnalité est disponible, consultez les options de la liste déroulant e Région sur la page intitulée Services AWSPar Région.
22 juin 2023	Versions et alias	Pour une liste complète des pays Régions AWS dans lesquels cette fonctionnalité est disponible, consultez les options de la liste déroulant e Région sur la page intitulée Services AWSPar Région.
16 juin 2023	Nouvelles AWS intégrations de SDK	Pour une liste complète des pays Régions AWS dans lesquels cette fonctionnalité est disponible, consultez les options de la liste déroulant e Région sur la page intitulée Services AWSPar Région.

Date de lancement	Nom de la fonctionnalité	Régions disponibles
01 décembre 2022	Orchestrez des workflows parallèles à grande échelle pour le traitement des données avec Distributed Map State	Pour une liste complète des pays Régions AWS dans lesquels cette fonctionnalité est disponible, consultez les options de la liste déroulant e Région sur la page intitulée Services AWSPar Région.

Historique du document

Cette section répertorie les principales modifications apportées au guide du AWS Step Functions développeur.

Modification	Description	Date de modification
Mises à jour	AWS mises à jour des politiques gérées - nouvelle autorisation : `states:ValidateStateMachineDefinition` Ajout d'informations sur les nouvelles autorisations permettant de vérifier la syntaxe d'une machine à états que vous fournissez. Pour en savoir plus, veuillez consulter la section AWS politiques gérées pour AWS Step Functions.	29 avril 2024
Nouvelle fonctionnalité	Step Functions ajoute une intégration optimisée pour AWS Elemental MediaConvert AWS Elemental MediaConvert fournit un transcodage de fichiers vidéo et audio de qualité télévisuelle, que les clients peuvent automatiser à l'aide d'un code adapté à leurs flux de travail multimédia. Grâce à l'intégration optimisée pour AWS Step Functions in MediaConvert, il est désormais possible d'orchestrer à l'aide de l'outil visuel low-code Workflow Studio. Pour en savoir plus, consultez la documentation relative à Manage AWS Elemental MediaConvert with Step Functions.	12 avril 2024
Mises à jour	AWS mises à jour de politiques gérées - Mise à jour d'une politique existante : `AWSStepFunctionsReadOnlyAccess` Ajout d'informations sur les nouvelles autorisations en lecture seule pour les balises, les cartes distribuées, les versions et les alias. Pour en savoir plus, veuillez consulter	2 avril 2024

Modification	Description	Date de modification
	la section AWS politiques gérées pour AWS Step Functions .	
Mises à jour	Step Functions ajoute la prise en charge des métriques Open Workflow Grâce aux métriques des flux de travail ouverts, vous avez désormais une visibilité au niveau du compte sur le nombre de flux de travail standard en cours ainsi que sur votre limite de flux de travail ouverts. Vous pouvez gérer les charges de travail dans tous les flux de travail, quel que soit leur mode de démarrage, afin de garantir le bon déroulement des opérations de flux de travail. Vous pouvez définir des CloudWatch alarmes pour surveiller vos flux de travail et recevoir des alertes de manière proactive lorsque vous approchez de vos limites. Une fois alerté, vous pouvez gérer efficacement vos flux de travail en prenant des mesures telles que l'arrêt de flux de travail spécifiques ou la demande d'augmentation de limite. Les métriques de flux de travail ouverts peuvent être utilisées dans CloudWatch les flux de travail standard sans qu'aucune configuration supplémentaire ne soit requise. Pour en savoir plus, veuillez consulter la section Métriques d'exécution.	29 février 2024
Mises à jour	Ajouts et mises à jour relatifs à l'intégration des services. Pour obtenir la liste des intégrations de AWS SDK nouvelles et mises à jour, consultez. Journal des modifications pour les intégrations de AWS SDK prises en charge Pour la liste complète des services, voirIntégrations de services AWS SDK prises en charge.	18 janvier 2024

Modification	Description	Date de modification
Nouvelle fonctionnalité	Utilisez Workflow Studio Application Composer pour créer des flux de travail sans serveur à l'aide de AWS CloudFormation modèles. Pour plus d'informations, consultez Utilisation de Workflow Studio dans Application Composer.	27 novembre 2023
Nouvelle fonctionnalité	Step Functionsvous permet désormais d'invoquer directement des points de terminaison HTTPS publics et de tester des états individuels à l'aide d'une nouvelle API Test State. Pour plus d'informations, consultez : • Appelez des API tierces • Utilisation de TestState l'API pour tester un état	26 novembre 2023
Nouvelle fonctionnalité	Step Functions s'intègre désormais à Amazon Bedrock Pour plus d'informations, consultez les rubriques suivantes : • Appelez Amazon Bedrock avec Step Functions • Autorisations IAM pour Amazon Bedrock • Réalisez un enchaînement d'instructions basé sur l'IA avec Amazon Bedrock • Utilisation AWS Step Functions avec d'autres services	26 novembre 2023
Nouvelle fonctionnalité	Step Functions vous permet désormais d'exécuter des redrive workflows de type Standard depuis leur point d'échec. Pour plus d'informations, consultez Redriving exécutions et RedrivingCartes parcourues.	15 novembre 2023
Mise à jour relative à la documentation uniquement	Publication d'une nouvelle rubrique qui explique comment exécuter des machines à états selon un calendrier en utilisantAmazon EventBridge Scheduler. Pour plus d'informations, consultez Utilisation d'Amazon EventBridge Scheduler avec AWS Step Functions.	16 octobre 2023

Modification	Description	Date de modification
Nouvelle fonctionnalité	Step Functions s'intègre désormais à Amazon EMR Serverless Pour plus d'informations, consultez les rubriques suivantes : • Appelez Amazon EMR Serverless avec Step Functions • Exécuter une EMR Serverless tâche • Intégrations optimisées pour Step Functions • Utilisation AWS Step Functions avec d'autres services	12 octobre 2023
Mise à jour relative à la documentation uniquement	Ajout d'informations sur l'exécution de machines à états selon un calendrier à l'aide deAmazon EventBridge Scheduler. Pour plus d'informations, consultez Utilisation du EventBridge planificateur.	05 octobre 2023
Mettre à jour	Réorganisation et mise à jour des rubriques relatives à l'état de la carte distribuée pour des raisons de clarté et de concision, et pour établir une carte de parcours claire pour les nouveaux utilisateurs. Pour plus d'informations, consultez Utilisation de l'état de la carte en mode distribué pour orchestrer des charges de travail parallèles à grande échelle.	6 octobre 2023
Correctifs	Exemples de code corrigés dans un didacticiel pour utiliser la AWS CDK v2. Pour plus d'informations, consultez Création d'une machine à Lambda états pour Step Functions utiliser AWS CDK.	19 septembre 202
Mettre à jour	Ajout d'informations sur les fonctionnalités améliorées de gestion des erreurs introduites par Step Functions pour identifier clairement les erreurs et implémenter les nouvelles tentatives avec un meilleur contrôle. Pour plus d'informations, consultez Fail et Réessayer après une erreur.	07 septembre 2023

Modification	Description	Date de modification
Mettre à jour	Step Functions a apporté des améliorations à Workflow Studio afin de rationaliser l'expérience de création de flux de travail. Pour plus d'informations, consultez AWS Step Functions Studio de flux de travail.	31 août 2023
Mise à jour relative à la documentation uniquement	Des informations ont été ajoutées environ deux fois le nombre réel de mesures indiqué pour la `ExecutionsStarted` métrique. Pour plus d'informations, consultez Indicateurs indiquant un décompte.	25 juillet 2023
Mise à jour relative à la documentation uniquement	Step Functions a ajouté deux nouveaux exemples de projets illustrant les cas d'utilisation courants suivants de l'état de la carte distribuée : • Traitement d'un fichier CSV • Traitement des données dans un compartiment Amazon S3	17 juillet 2023
Mise à jour relative à la documentation uniquement	Publication d'une nouvelle rubrique sur le déploiement de machines d'état à l'aide de Terraform. Pour plus d'informations, consultez Déploiement de machines d'état à l'aide de Terraform.	5 juillet 2023
Mise à jour relative à la documentation uniquement	Les procédures suivantes ont été mises à jour pour tenir compte des modifications apportées à l' EventBridge interface Amazon. • Routage d'un événement Step Functions vers EventBridge • Démarrage d'une exécution State Machine en réponse à des événements Amazon S3	26 juin 2023

Modification	Description	Date de modification
Nouvelle fonctionnalité	Step Functions permet désormais de créer plusieurs versions de machines à états et des alias pour améliorer la résilience lors du déploiement de flux de travail sans serveur. Pour plus d'informations, consultez Gérez les déploiements continus avec des versions et des alias.	22 juin 2023
Mise à jour relative à la documentation uniquement	Amélioration de la description `TimeoutSeconds` et des `HeartbeatSeconds` champs pour décrire en quoi ils sont différents les uns des autres. Pour plus d'informations, consultez Champs d'état des tâches.	22 juin 2023
Mise à jour relative à la documentation uniquement	Publication d'une nouvelle section qui décrit comment aplatir un tableau de tableaux généralement renvoyé comme résultat pour les états Parallel et Map. Pour plus d'informations, consultez Aplatir un tableau de tableaux.	20 juin 2023
Mettre à jour	Step Functions a étendu la prise en charge des intégrations de AWS SDK en ajoutant sept Services AWS et 468 nouvelles actions d'API. Pour plus d'informations, consultez Intégrations de services AWS SDK prises en charge et Journal des modifications pour les intégrations de AWS SDK prises en charge.	16 juin 2023
Mise à jour relative à la documentation uniquement	Publication d'une nouvelle rubrique Régions AWS répertoriant les fonctionnalités de Step Functions récemment lancées. Pour plus d'informations, consultez Derniers lancements de fonctionnalités.	16 juin 2023
Mise à jour relative à la documentation uniquement	Step Functions inclut désormais une section sur Notifications des utilisateurs AWS, et Service AWS qui sert d'emplacement central pour vos AWS notifications dans le AWS Management Console. Pour plus d'informations, consultez Utilisation Notifications des utilisateurs AWS avec Step Functions.	4 mai 2023

Modification	Description	Date de modification
Mise à jour relative à la documentation uniquement	Ajout d'une nouvelle section qui explique les autorisations nécessaires pour écrire les résultats d'exécution d'un flux de travail enfant dans un compartiment Amazon S3 chiffré à l'aide d'une AWS Key Management Service (AWS KMS) clé. Pour plus d'informations, consultez Autorisations IAM pour le compartiment Amazon S3 AWS KMS key chiffré.	29 avril 2023
Mise à jour relative à la documentation uniquement	Ajout d'une nouvelle rubrique expliquant la fonctionnalité de simulateur de flux de données. Pour plus d'informations, consultez Simulateur de flux de données.	14 avril 2023
Mise à jour des quotas	Ajout d'informations sur le quota par défaut de 1 000 pour les Map Runs ouverts dans chaque compte. Pour plus d'informations, consultez Quotas liés aux comptes.	05 avril 2023
Mise à jour relative à la documentation uniquement	Ajout d'une rubrique qui décrit à quel moment migrer les AWS Data Pipeline charges de travail vers Step Functions . Cette rubrique fournit également une liste d'exemple s expliquant comment effectuer la migration. Pour plus d'informations, consultez Migration des charges de travail depuis Step AWS Data Pipeline Functions.	30 mars 2023
Mise à jour relative à la documentation uniquement	Ajout d'une note concernant l'indisponibilité du traçage par rayons X pour l'état de la carte distribuée. Pour plus d'informations, consultez AWS X-Ray et Step Functions.	21 mars 2023
Mise à jour relative à la documentation uniquement	Ajout d'informations sur la façon dont Step Functions gère les autorisations basées sur les balises. Pour plus d'informations, consultez Fonctions de balisage en étapes et Stratégies basées sur des balises.	15 mars 2023

Modification	Description	Date de modification
Mise à jour relative à la documentation uniquement	Ajout d'informations sur la façon dont Step Functions analyse les fichiers CSV utilisés comme entrée dans l'état de la carte distribuée. Pour plus d'informations, consultez Fichier CSV dans un compartiment Amazon S3.	14 mars 2023
Mise à jour relative à la documentation uniquement	Ajout d'informations sur la façon dont Step Functions gère les appels entre comptes pour le modèle Run a Job (.sync). Pour plus d'informations, consultez Run a Job (.sync).	01 mars 2023
Mise à jour relative à la documentation uniquement	Réduisez la période de conservation de l'historique de vos exécutions de flux de travail terminées de 90 jours à 30 jours. Pour plus d'informations sur l'ajustement de la période de conservation, reportez-vous Garanties d'exécution aux sections etQuotas liés aux exécutions par les machines de l'État.	21 février 2023
Mettre à jour	Step Functions a étendu la prise en charge des intégrations de AWS SDK en ajoutant 35 AWS services et 1 100 nouvelles actions d'API. Pour plus d'informations, consultez Intégrations de services AWS SDK prises en charge et Journal des modifications pour les intégrations de AWS SDK prises en charge.	17 février 2023
Mise à jour relative à la documentation uniquement	A publié une série de didacticiels Getting Started qui vous explique le processus de création d'un flux de travail pour une demande de carte de crédit à l'aide de Step Functions . Pour plus d'informations, consultez Démarrer avec AWS Step Functions.	30 décembre 2022

Modification	Description	Date de modification
Nouvelle fonctionnalité	Step Functions permet d'orchestrer des flux de travail parallèles à grande échelle pour le traitement des données à l'aide d'un nouveau mode distribué pour `Map` l'état. Pour plus d'informations, consultez Utilisation de l'état de la carte en mode distribué pour orchestrer des charges de travail parallèles à grande échelle.	01 décembre 2022
Nouvelle fonctionnalité	Step Functions prend désormais en charge l'accès aux AWS ressources inter-comptes configurées dans d'autres comptes. Pour plus d'informations, veuillez consulter la rubrique • Accès à des ressources Comptes AWS dans d'autres flux de travail • Tutoriel : Accès aux ressources multicomptes AWS • Task state	18 novembre 2022
Mettre à jour	Step Functions propose désormais une nouvelle expérience de console permettant de visualiser et de déboguer les exécutions de flux de travail Express. Pour plus d'informations, consultez : • Exécutions de flux de travail standard et express dans la console • Affichage et débogage des exécutions sur la console Step Functions	18 octobre 2022
Mettre à jour	Ajout du support permettant de spécifier éventuellement le `ExecutionRoleArn` paramètre lors de l'utilisation des `addStep.sync` API `addStep` et pour l'intégration des services optimisés Amazon EMR. Pour plus d'informations, consultez Appelez Amazon EMR avec Step Functions.	20 septembre 2022

Modification	Description	Date de modification
Mise à jour relative à la documentation uniquement	Ajout d'une nouvelle rubrique qui fournit des recommandations sur l'optimisation des coûts lors de la création de flux de travail sans serveur à l'aide de Step Functions. Pour plus d'informations, consultez Optimisation des coûts à l'aide d'Express Workflows.	15 septembre 202
Mettre à jour	Step Functions prend en charge 14 nouvelles fonctions intrinsèques pour effectuer des tâches de traitement des données, telles que les manipulations de tableaux, le codage et le décodage des données, les calculs de hachage, la manipulation de données JSON, les opérations de fonctions mathématiques et la génération d'identifiants uniques. Mise à jour relative à la documentation uniquement : Vous avez regroupé toutes les fonctions intrinsèques existantes et récemment introduites dans les catégories suivantes en fonction du type de tâche de traitement des données qu'elles vous aident à effectuer : • Intrinsèques pour les réseaux • Intrinsèques pour le codage et le décodage des données • Intrinsèque pour le calcul du hachage • Intrinsèques pour la manipulation des données JSON • Intrinsèques pour les opérations mathématiques • Intrinsèque pour le fonctionnement des chaînes • Intrinsèque pour la génération d'identifiants uniques • Intrinsèque pour un fonctionnement générique Pour plus d'informations, consultez Fonctions intrinsèques.	31 août 2022

Modification	Description	Date de modification
Mettre à jour	Step Functions a étendu la prise en charge des intégrations de AWS SDK en ajoutant trois AWS services supplémentaires : AWS Billing ConductorAmazon GameSparks, et. Amazon Pinpoint SMS and Voice V2 Pour plus d'informations, consultez Journal des modifications pour les intégrations de AWS SDK prises en charge.	26 juillet 2022
Mise à jour relative à la documentation uniquement	Ajout d'une nouvelle rubrique pour inclure un résumé de toutes les mises à jour apportées aux intégrations de AWS SDK prises en charge par Step Functions. Pour plus d'informations, consultez Journal des modifications pour les intégrations de AWS SDK prises en charge.	26 juillet 2022
Mise à jour relative à la documentation uniquement	AWS Step Functions Le guide du développeur inclut désormais des détails sur les métriques d'exécution émises spécifiquement pour Express Workflows. Pour plus d'informations, consultez Mesures d'exécution pour Express Workflows.	9 juin 2022

Modification	Description	Date de modification
Mettre à jour	Améliorations apportées à la console Step Functions La console comporte désormais une page de détails d'exécution repensée qui inclut les améliorations suivantes : • Possibilité d'identifier la raison d'un échec d'exécution en un coup d'œil. • Deux nouveaux modes de visualisation pour votre machine à états : affichage sous forme de tableau et affichage d'événements. Ces vues vous permettent également d'appliquer des filtres pour n'afficher que les informations qui vous intéressent. En outre, vous pouvez trier le contenu de l'affichage des événements en fonction des horodatages des événements. • Basculez entre les différentes itérations d'`Map`état dans le mode d'affichage graphique à l'aide d'une liste déroulant e ou dans l'arborescence du mode d'affichage tabulaire pour les `Map` états. • Consultez des informations détaillées sur chaque état du flux de travail, y compris le chemin complet de transfert des données d'entrée et de sortie et réessayez les tentatives pour les `Parallel` états `Task` ou. • Diverses améliorations, notamment la possibilité de copier le nom de ressource Amazon d'exécution de la machine à états, d'afficher le nombre total de transitio ns entre les machines à états et d'exporter les détails de l'exécution au format JSON. Mises à jour relatives uniquement à la documentation	09 mai 2022

Modification	Description	Date de modification
	Ajout d'une nouvelle rubrique expliquant les différents types d'informations affichées sur la page Détails de l'exécution. Un didacticiel a également été ajouté pour montrer comment examiner ces informations. Pour plus d’informations, consultez : • Affichage et débogage des exécutions sur la console Step Functions • Tutoriel : Examen des exécutions par des machines à états à l'aide de la console Step Functions	
Mettre à jour	Step Functions propose désormais une solution pour éviter le problème de sécurité secondaire confus, qui survient lorsqu'une entité (un service ou un compte) est contrainte par une autre entité d'effectuer une action. Pour plus d’informations, consultez : • Prévenir le problème des adjoints confus entre les services	2 mai 2022

Modification	Description	Date de modification

Mettre à jour	• Step Functions a étendu la prise en charge des intégrations de AWS SDK en ajoutant 21 services supplémentaires AWS . Pour plus d'informations, consultez Intégrations de services AWS SDK prises en charge.
	• Mises à jour relatives à la documentation uniquement :
	  • Ajout d'une liste de tous les préfixes d'exception présents dans les exceptions générées lorsque vous effectuez par erreur une intégration de service AWS SDK avec Step Functions. Pour plus d'informations, consultez Intégrations de services AWS SDK prises en charge.
	  • Ajout d'une liste de toutes les actions d'API non prises en charge pour les intégrations de AWS SDK prises en charge. Pour plus d'informations, consultez Actions d'API non prises en charge pour les services pris en charge.
	  • Ajout d'une liste de toutes les intégrations de AWS SDK prises en charge qui sont désormais obsolètes . Pour plus d'informations, consultez Intégrations de services AWS SDK obsolètes.
		19 avril 2022

Nouvelle fonctionnalité	Step Functions Local prend désormais en charge l'intégration du AWS SDK et la simulation des intégrations de services. Pour plus d’informations, consultez : • Utilisation d'intégrations de services simulées	28 janvier 2022

Modification	Description	Date de modification
Nouvelle fonctionnalité	AWS Step Functions prend désormais en charge la création d'une API REST Amazon API Gateway avec une machine à états express synchrone comme intégration principale à l'aide du. AWS Cloud Development Kit (AWS CDK) Pour plus d'informations, consultez : • Création d'une API REST API Gateway avec une machine synchrone Express State à l'aide du AWS CDK	10 décembre 2021
Mettre à jour	Step Functions a ajouté trois nouveaux exemples de projets qui démontrent l'intégration de Step Functions et de la console améliorée d'Amazon Athena. Pour plus d'informations, consultez : • Exécuter plusieurs requêtes (Amazon Athena, Amazon SNS) • Interrogez de grands ensembles de données (Amazon Athena, Amazon S3 AWS Glue, Amazon SNS) • Maintenir les données à jour (Amazon Athena, Amazon S3,) AWS Glue	22 novembre 2021
Nouvelle fonctionnalité	Step Functions a ajouté la prise en charge des points de terminaison Amazon VPC pour les flux de travail synchrones Express. Pour plus d'informations, consultez : • Points de terminaison Amazon VPC pour Step Functions	15 novembre 2021

Modification	Description	Date de modification
Mettre à jour	AWS Step Functions a ajouté trois nouveaux exemples de projets qui montrent comment utiliser l' AWS Batch intégration Step Functions. Pour plus d'informations, consultez : • Répartissez un AWS Batch travail • AWS Batch avec Lambda • Utilisation Step Functions et gestion AWS Batch des erreurs	14 octobre 2021
Nouvelle fonctionnalité	AWS Step Functions a ajouté des intégrations de AWS SDK, vous permettant d'utiliser les actions de l'API pour l'ensemble des plus de deux cents AWS services. Pour plus d'informations, consultez : • AWS Intégrations de services SDK • Collectez des informations sur le compartiment Amazon S3 à l'aide des AWS intégrations de services du SDK	30 septembre 202
Nouvelle fonctionnalité	AWS Step Functions a ajouté un concepteur visuel de flux de travail, le AWS Step Functions Workflow Studio. Pour plus d'informations, consultez : • AWS Step Functions Studio de flux de travail • Apprenez à utiliser le AWS Step Functions Workflow Studio	17 juin 2021
Mettre à jour	AWS Step Functions a ajouté quatre nouvelles API`StartBuildBatch` ,`StopBuildBatch` ,`DeleteBuildBatch` , `RetryBuildBatch` et à l' CodeBuild intégration. Pour plus d'informations, consultez : • Appelez AWS CodeBuild avec Step Functions	4 juin 2021

Modification	Description	Date de modification
Nouvelle fonctionnalité	AWS Step Functions s'intègre désormais à Amazon EventBridge. Pour plus d'informations, consultez : • Appelez EventBridge avec Step Functions • Politiques IAM pour Step Functions et Politiques IAM pour Amazon EventBridge • Un exemple de projet qui montre comment Envoyer un événement personnalisé à EventBridge	14 mai 2021
Mettre à jour	AWS Step Functions a ajouté un nouvel exemple de projet qui montre comment utiliser Step Functions et l'API Amazon Redshift Data pour exécuter un flux de travail ETL/ELT. Pour plus d'informations, consultez : • Exécutez des flux de travail ETL/ELT à l'aide d'Amazon Redshift (Lambda, API de données Amazon Redshift)	16 avril 2021
Nouvelle fonctionnalité	AWS Step Functions dispose d'un nouveau simulateur de flux de données dans la console. Pour plus d'informations, consultez : • Console Step Functions	08 avril 2021
Nouvelle fonctionnalité	AWS Step Functions s'intègre désormais à Amazon EMR sur EKS. Pour plus d'informations, consultez : • Appelez Amazon EMR sur EKS avec AWS Step Functions	29 mars 2021

Modification	Description	Date de modification
Mettre à jour	Le support YAML pour les définitions de machines à états a été ajouté à AWS Toolkit for Visual Studio Code et AWS CloudFormation. Pour plus d'informations, consultez : • Support du format de définition • AWS Toolkit for Visual Studio Code	4 mars 2021
Nouvelle fonctionnalité	AWS Step Functions s'intègre désormais à AWS Glue DataBrew. Pour plus d'informations, consultez : • Gérez les AWS Glue DataBrew tâches avec Step Functions • Qu'est-ce que c'est AWS Glue DataBrew ? dans le guide DataBrew du développeur.	6 janvier 2021
Nouvelle fonctionnalité	AWS Step Functions Les flux de travail express synchrones sont désormais disponibles, ce qui vous permet d'orchestrer facilement des microservices. Pour plus d'informations, consultez : • Flux de travail express synchrones et asynchrones • Un exemple de projet qui montre comment Invoquer des flux de travail express synchrones • La documentation de StartSyncExecutionl'API.	24 novembre 2020
Nouvelle fonctionnalité	AWS Step Functions s'intègre désormais à Amazon API Gateway. Pour plus d'informations, consultez : • Call API Gateway avec Step Functions • Politiques IAM pour Step Functions et Politiques IAM pour Amazon API Gateway • Un exemple de projet qui montre comment Passez un appel à API Gateway	17 novembre 2020

Modification	Description	Date de modification
Nouvelle fonctionnalité	AWS Step Functions s'intègre désormais à Amazon Elastic Kubernetes Service. Pour plus d'informations, consultez : • Appelez Amazon EKS avec Step Functions • Politiques IAM pour Step Functions et Politiques IAM pour Amazon EKS • Un exemple de projet qui montre comment Gérer un cluster Amazon EKS	16 novembre 2020
Nouvelle fonctionnalité	AWS Step Functions s'intègre désormais à Amazon Athena. Pour plus d'informations, consultez : • Appelez Athéna avec Step Functions • Politiques IAM pour Step Functions et Politiques IAM pour Amazon Athena • Un exemple de projet qui montre comment Lancer une requête Athena	22 octobre 2020
Nouvelle fonctionnalité	AWS Step Functions prend désormais en charge le suivi des end-to-end flux de travail avec AWS X-Ray, ce qui vous donne une visibilité complète sur les exécutions par State Machine et facilite l'analyse et le débogage de vos applications distribuées. Pour plus d'informations, consultez : • AWS X-Ray et Step Functions • Politiques IAM pour Step Functions et Politiques IAM pour AWS X-Ray • AWS Step Functions API Reference • TracingConfiguration	14 septembre 2020

Modification	Description	Date de modification

Mettre à jour	AWS Step Functions prend désormais en charge des tailles de charge utile allant jusqu'à 256 Ko de données sous forme de chaîne codée en UTF-8. Cela vous permet de traiter des charges utiles plus importantes dans les flux de travail Standard et Express.
	
	Vos machines d'état existantes n'ont pas besoin d'être modifiées pour utiliser des charges utiles plus important es. Cependant, vous devrez effectuer une mise à jour vers les dernières versions du SDK Step Functions et de Local Runner pour utiliser les API mises à jour. Pour plus d'informations, consultez :
	
	• Quotas
	• the section called “Utilisez les ARN d'Amazon S3 au lieu de transmettre des charges utiles importantes”
	• `States.DataLimitExceeded`
	• the section called “CloudWatchEnregistre les charges utiles”
	• the section called “EventBridge charges utiles”
	• AWS Step Functions API Reference
	
	  • CloudWatchEventsExecutionDataDetails
	  • HistoryEventExecutionDataDetails
	  • GetExecutionHistory
	  • ActivityScheduledEventDetails
	  • ActivitySucceededEventDetails
	  • CloudWatchEventsExecutionDataDetails
	  • ExecutionSucceededEventDetails
	  • LambdaFunctionScheduledEventDetails
	  • ExecutionSucceededEventDetails
		3 septembre 2020

Modification	Description	Date de modification
	• StateEnteredEventDetails • StateExitedEventDetails • TaskSubmittedEventDetails • TaskSucceededEventDetails	

Modification	Description	Date de modification

Mettre à jour	La langue d'Amazon States a été mise à jour comme suit :
	• Règles Choicea ajouté
	  • Un opérateur de comparaison nul,`IsNull`. `IsNull`teste par rapport à la valeur nulle JSON et peut être utilisé pour détecter si la sortie d'un état précédent est nulle ou non.
	  • Quatre autres nouveaux opérateurs ont été ajoutés IsBoolean, IsNumeric, IsString et IsTimestamp.
	  • Test de l'existence ou de l'inexistence d'un champ à l'aide de l'`IsPresent` opérateur. `IsPresent` peut être utilisé pour éviter les `States.Runtime` erreurs lors d'une tentative d'accès à une clé inexistante.
	  • Correspondance de motifs génériques pour permettre la comparaison de chaînes avec des modèles contenant un ou plusieurs caractères génériques.
	  • Comparaison entre deux variables pour les opérateurs de comparaison pris en charge.
	• Les valeurs du délai d'expiration et du rythme cardiaque d'un `Task` état peuvent désormais être fournies dynamiquement à partir de l'entrée d'état au lieu d'une valeur fixe à l'aide des champs `TimeoutSecondsPath` et`HeartbeatSecondsPath` . Consultez l'État de la tâcheÉtat pour plus d'informations.
	• Le nouveau ResultSelector champ permet de manipuler le résultat d'un état avant `ResultPath` son application. Le `ResultSelector` champ est un champ facultatif dans les État de la tâche états MapParallèle, et.
	• Fonctions intrinsèquesont été ajoutés pour permettre les opérations de base sans `Task` états. Les fonctions
		13 août 2020

Modification	Description	Date de modification
	intrinsèques peuvent être utilisées dans les `ResultSelector` champs `Parameters` et.	
Mettre à jour	AWS Step Functions prend désormais en charge l'appel SageMaker `CreateProcessingJob` d'API Amazon. Pour plus d'informations, consultez : • Gérez SageMaker avec Step Functions • Prétraitez les données et entraînez un modèle d'apprentissage automatique, un exemple de projet qui illustre`CreateProcessingJob` .	4 août 2020
Nouvelle fonctionnalité	AWS Step Functions est désormais pris en charge par AWS Serverless Application Model, ce qui facilite l'intégration de l'orchestration des flux de travail dans vos applications sans serveur. Pour plus d'informations, consultez : • AWS Step Functions et AWS SAM • AWS::Serverless::StateMachine • AWS SAM Modèles de politiques	27 mai 2020
Nouvelle fonctionnalité	AWS Step Functions a introduit une nouvelle invocation synchrone pour imbriquer les exécutions de Step Functions . Le nouvel appel, `arn:aws:states:::states:startExecution.sync:2` , renvoie un objet JSON. L'appel d'origine `arn:aws:states:::states:startExecution.sync` , reste pris en charge et renvoie une chaîne JSON d'échappement. Pour plus d'informations, consultez : • Gérez AWS Step Functions les exécutions en tant que service intégré	19 mai 2020

Modification	Description	Date de modification
Nouvelle fonctionnalité	AWS Step Functions s'intègre désormais à AWS CodeBuild . Pour plus d'informations, consultez : • Utilisation AWS Step Functions avec d'autres services • Appelez AWS CodeBuild avec Step Functions • Intégrations optimisées pour Step Functions	5 mai 2020
Nouvelle fonctionnalité	Step Functions est désormais pris en charge dans AWS Toolkit for Visual Studio Code, ce qui facilite la création et la visualisation de flux de travail basés sur des machines à états sans quitter votre éditeur de code.	31 mars 2020
Mettre à jour	Vous pouvez désormais configurer la journalisation sur Amazon CloudWatch Logs pour les flux de travail standard. Pour plus d'informations, consultez : • Journalisation à l'aideCloudWatchJournaux	25 février 2020
Nouvelle fonctionnalité	AWS Step Functions est désormais accessible sans adresse IP publique, directement depuis Amazon Virtual Private Cloud (VPC). Pour plus d'informations, consultez : • Points de terminaison Amazon VPC pour Step Functions	23 décembre 2019

Modification	Description	Date de modification

Nouvelle fonctionnalité	Les workflows express sont un nouveau type de flux de travail, adapté aux charges de travail de traitement d'événements à volume élevé telles que l'ingestion de données IoT, le traitement et la transformation des données en continu et les backends d'applications mobiles.
	
	Pour de plus amples informations, veuillez consulter les rubriques nouvelles et mises à jour suivantes.
	
	• Flux de travail standard ou express
	  • Garanties d'exécution
	• Utilisation AWS Step Functions avec d'autres services
	  • Intégrations optimisées pour Step Functions
	• Traitement de gros volumes de messages depuis Amazon SQS (Express Workflows)
	• Exemple de point de contrôle sélectif (workflows express)
	• Quotas
	  • Quotas
	• Journalisation à l'aideCloudWatchJournaux
	• AWS Step Functions API Reference
	  • CreateStateMachine
	  • UpdateStateMachine
	  • DescribeStateMachine
	  • DescribeStateMachineForExecution
	  • StopExecution
	  • DescribeExecution
	  • GetExecutionHistory
	  • ListExecutions
	  • ListStateMachines
		3 décembre 2019

Modification	Description	Date de modification
	• StartExecution • CloudWatchLogsLogGroup • LogDestination • LoggingConfiguration	
Nouvelle fonctionnalité	AWS Step Functions s'intègre désormais à Amazon EMR. Pour plus d'informations, consultez : • Utilisation AWS Step Functions avec d'autres services • Appelez Amazon EMR avec Step Functions • Intégrations optimisées pour Step Functions	19 novembre 2019

Mettre à jour	AWS Step Functions a publié le SDK AWS Step Functions Data Science. Pour plus d'informations, consultez les rubriques suivantes.
	• Projet sur Github
	• Documentation des kits SDK
	• Les exemples de blocs-notes suivants, disponibles dans la SageMakerconsole et dans le GitHub projet associé.
	  • `hello_world_workflow.ipynb`
	  • `machine_learning_workflow_abalone.ipynb`
	  • `training_pipeline_pytorch_mnist.ipynb`
		7 novembre 2019

Modification	Description	Date de modification
Mettre à jour	Step Functions prend désormais en charge davantage d'actions d'API pour Amazon SageMaker et inclut deux nouveaux exemples de projets pour démontrer cette fonctionnalité. Pour plus d'informations, consultez les rubriques suivantes. • Gérez SageMaker avec Step Functions • Utilisation AWS Step Functions avec d'autres services • Former un modèle de Machine Learning • Régler un modèle de Machine Learning	3 octobre 2019
Nouvelle fonctionnalité	Step Functions permet de démarrer de nouvelles exécutions de flux de travail en `StartExecution` faisant appel à une API de service intégrée. Consultez : • Démarrer les exécutions de flux de travail à partir d'un état de tâche • Gérez AWS Step Functions les exécutions en tant que service intégré • Utilisation AWS Step Functions avec d'autres services • Politiques IAM pour le démarrage des exécutions de flux de travail Step Functions	12 août 2019

Modification	Description	Date de modification
Nouvelle fonctionnalité	Step Functions inclut la possibilité de transmettre un jeton de tâche aux services intégrés et de suspendre l'exécution jusqu'à ce que ce jeton de tâche soit renvoyé avec `SendTaskSuccess` ou `SendTaskFailure` . Consultez : • Modèles d'intégration des services • Attendre un rappel avec le jeton de tâche • Exemple de modèle de rappel (Amazon SQS, Amazon SNS, Lambda) • Intégrations optimisées pour Step Functions • Déployer un exemple de projet d'approbation humaine • Métriques d'intégration de services Step Functions permet désormais d'accéder aux informations dynamiques concernant votre exécution en cours directement dans le `"Parameters"` champ de définition d'un état. Consultez : • Objet Contexte • Transmettre les nœuds d’objet de contexte comme paramètres	23 mai 2019
Nouvelle fonctionnalité	Step Functions prend en charge les CloudWatch événements pour les changements de statut d'exécution, voir : • EventBridge (CloudWatch Evénements) pour les changements de statut d'exécution de Step Functions • Guide de l'utilisateur d'Amazon CloudWatch Events	8 mai 2019

Modification	Description	Date de modification
Nouvelle fonctionnalité	Step Functions prend en charge les autorisations IAM à l'aide de balises. Pour plus d'informations, consultez : • Fonctions de balisage en étapes • Stratégies basées sur des balises	5 mars 2019
Nouvelle fonctionnalité	Step Functions Local est désormais disponible. Vous pouvez exécuter Step Functions sur votre machine locale à des fins de test et de développement. Step Functions Local est disponible au téléchargement sous forme d'application Java ou d'image Docker. veuillez consulter Tester les machines d'état localement.	4 février 2019
Nouvelle fonctionnalité	AWS Step Functions est désormais disponible dans les régions de Pékin et de Ningxia. veuillez consulter Régions prises en charge.	15 janvier 2018
Nouvelle fonctionnalité	Step Functions prend en charge le balisage des ressources pour faciliter le suivi de votre allocation des coûts. Vous pouvez baliser des machines d'état sur la page Details (Détails), ou via les actions d'API. veuillez consulter Fonctions de balisage en étapes.	7 janvier 2019
Nouvelle fonctionnalité	AWS Step Functions est désormais disponible en Europe (Paris) et en Amérique du Sud (São Paulo). veuillez consulter Régions prises en charge.	13 décembre 2018
Nouvelle fonctionnalité	AWS Step Functions est désormais disponible dans la région Europe (Stockholm). Consultez Régions prises en charge pour obtenir une liste des régions prises en charge.	12 décembre 2018

Modification	Description	Date de modification
Nouvelle fonctionnalité	Step Functions s'intègre désormais à certains AWS services. Vous pouvez désormais appeler et transmettre des paramètres directement à l'API de ces services intégrés à partir d'un état de tâche dans le langage Amazon States Language. Pour plus d'informations, consultez : • Utilisation AWS Step Functions avec d'autres services • Transmettre des paramètres à une API de service • Intégrations optimisées pour Step Functions	29 novembre 2018
Mettre à jour	Amélioration de la description de `TimeoutSeconds` et `HeartbeatSeconds` dans la documentation des états de tâche. veuillez consulter État de la tâche.	24 octobre 2018
Mettre à jour	Amélioration de la description de la limite Taille maximale de l'historique d'exécution et fourniture d'un lien vers les bonnes pratiques liées à la rubrique. • Quotas liés aux exécutions par les machines de l'État • Évitez d'atteindre le quota d'historique	17 octobre 2018
Mettre à jour	Ajout d'un nouveau didacticiel à la AWS Step Functions documentation : voirDémarrage d'une exécution State Machine en réponse à des événements Amazon S3.	25 septembre 2018
Mettre à jour	Suppression de l'entrée Nombre maximal d'exécutions affichées dans la console Step Function de la documentation relative aux limites. veuillez consulter Quotas.	13 septembre 2018
Mettre à jour	Ajout d'une rubrique sur les meilleures pratiques à la AWS Step Functions documentation sur l'amélioration de la latence lors du sondage pour les tâches d'activité. veuillez consulter Évitez la latence lorsque vous interrogez pour des tâches d'activité.	30 août 2018

Modification	Description	Date de modification
Mettre à jour	Amélioration de la AWS Step Functions rubrique sur les activités et les travailleurs actifs. veuillez consulter Activités .	29 août 2018
Mettre à jour	Amélioration de la AWS Step Functions rubrique sur CloudTrail l'intégration. veuillez consulter Enregistrement des appels d'API avec AWS CloudTrail.	7 août 2018
Mettre à jour	Des exemples JSON ont été ajoutés au AWS CloudFormation didacticiel. veuillez consulter Création d'une machine à états Lambda pour Step Functions à l'aide de AWS CloudFormation.	23 juin 2018
Mettre à jour	Ajout d'une nouvelle rubrique sur la gestion des erreurs de service Lambda. veuillez consulter Gérer les exceptions du service Lambda.	le 20 juin 2018
Nouvelle fonctionnalité	AWS Step Functions est désormais disponible dans la région Asie-Pacifique (Mumbai). Consultez Régions prises en charge pour obtenir une liste des régions prises en charge.	28 juin 2018
Nouvelle fonctionnalité	AWS Step Functions est désormais disponible dans la région AWS GovCloud (ouest des États-Unis). Consultez Régions prises en charge pour obtenir une liste des régions prises en charge. Pour plus d'informations sur l'utilisation de Step Functions dans la région AWS GovCloud (ouest des États-Unis), consultez AWS GovCloud (US).	28 juin 2018
Mettre à jour	Amélioration de la documentation relative au traitement des erreurs pour les états `Parallel`. veuillez consulter Gestion des erreurs.	le 20 juin 2018

Modification	Description	Date de modification
Mettre à jour	Documentation améliorée sur le traitement des entrées et des sorties dans Step Functions. Apprenez à utiliser `InputPath` , `ResultPath` et `OutputPath` pour contrôler le flux de JSON par le biais de vos flux de travail, des états et des tâches. Consultez : • Traitement des entrées et des sorties dans Step Functions • ResultPath	7 juin 2018
Mettre à jour	Amélioration des exemples de code pour les états Parallel. veuillez consulter Parallèle.	4 juin 2018
Nouvelle fonctionnalité	Vous pouvez désormais surveiller les métriques des API et des services dans CloudWatch. veuillez consulter Surveillance des fonctions Step Functions à l'aide de CloudWatch.	25 mai 2018
Mettre à jour	Les limitations ont été augmentées pour `StartExecution` , `StopExecution` et `StateTransition` dans les régions suivantes : • USA Est (Virginie du Nord) • USA Ouest (Oregon) • Europe (Irlande) Pour plus d'informations, consultez Quotas.	16 mai 2018
Nouvelle fonctionnalité	AWS Step Functions est désormais disponible dans les régions de l'ouest des États-Unis (Californie du Nord) et de l'Asie-Pacifique (Séoul). Consultez Régions prises en charge pour obtenir une liste des régions prises en charge.	5 mai 2018

Modification	Description	Date de modification
Mettre à jour	Mise à jour des procédures et des images pour correspondre aux modifications de l'interface.	25 avril 2018
Mettre à jour	Ajout d'un didacticiel qui montre comment démarrer une nouvelle exécution pour poursuivre votre travail. veuillez consulter Poursuite des exécutions de flux de travail de longue durée en tant que nouvelle exécution. Ce didacticiel décrit un modèle de conception qui peut vous aider à éviter certaines restrictions de service. veuillez consulter Évitez d'atteindre le quota d'historique.	19 avril 2018
Mettre à jour	Amélioration de la présentation de la documentation des états par l'ajout d'informations conceptuelles sur les machines d'état. veuillez consulter States.	9 mars 2018
Mettre à jour	Outre le HTML, le PDF et le Kindle, le guide du AWS Step Functions développeur est disponible sur GitHub. Pour laisser un commentaire, cliquez GitHub sur l'icône dans le coin supérieur droit. 	2 mars 2018
Mettre à jour	Ajout d'une rubrique décrivant d'autres ressources relatives à Step Functions. veuillez consulter Informations connexes.	20 février 2018

Modification	Description	Date de modification

Nouvelle fonctionnalité	• Lorsque vous créez une nouvelle machine à états, vous devez reconnaître que cela AWS Step Functions créera un rôle IAM qui permettra d'accéder à vos fonctions Lambda.
	• Mise à jour des didacticiels suivants pour refléter les modifications mineures apportées au flux de création de machine d'état :
	  • Création d'une machine d'état Step Functions utilisant Lambda
	  • Création d'une machine à états d'activité à l'aide de Step Functions
	  • Gestion des conditions d'erreur à l'aide d'une machine à états Step Functions
	  • Itérer une boucle avec Lambda
		19 février 2018

Mettre à jour	Ajout d'une rubrique qui décrit un exemple de travail d'activité écrit en Ruby. Cette implémentation peut être utilisée pour créer directement un travail d'activité Ruby ou un modèle de conception en vue de la création d'un travail d'activité dans un autre langage. veuillez consulter Exemple d'activité de travail en Ruby.	6 février 2018
Mettre à jour	Ajout d'un nouveau didacticiel décrivant un modèle de conception utilisant une fonction Lambda pour itérer un décompte. veuillez consulter Création d'une machine d'état Step Functions utilisant Lambda.	31 janvier 2018

Modification	Description	Date de modification
Mettre à jour	Contenu mis à jour sur les autorisations IAM à inclure `DescribeStateMachineForExecution` et les `UpdateStateMachine` API. veuillez consulter Création d'autorisations IAM granulaires pour les utilisateurs non administrateurs.	26 janvier 2018
Mettre à jour	Nouvelles régions disponibles ajoutées : Canada (Centre), Asie-Pacifique (Singapour). veuillez consulter Régions prises en charge.	25 janvier 2018
Mettre à jour	Tutoriels et procédures mis à jour pour refléter le fait qu'IAM vous permet de sélectionner Step Functions comme rôle.	24 janvier 2018
Mettre à jour	Ajout d'une rubrique Best Practices qui suggère de ne pas transmettre de charges utiles volumineuses entre états. veuillez consulter Utilisez les ARN d'Amazon S3 au lieu de transmettre des charges utiles importantes.	23 janvier 2018
Mettre à jour	Procédures corrigées pour correspondre à l'interface mise à jour de création d'une machine d'état : • Création d'une machine d'état Step Functions utilisant Lambda • Création d'une machine à états d'activité à l'aide de Step Functions • Gestion des conditions d'erreur à l'aide d'une machine à états Step Functions	17 janvier 2018

Modification	Description	Date de modification
Nouvelle fonction	Vous pouvez utiliser des exemples de projets pour rapidement mettre en service des machines d'état et toutes les ressources AWS associées. Voir Exemples de projets pour Step Functions, Les exemples de projets disponibles incluent : • Sondage pour connaître le statut du poste (Lambda,) AWS Batch • Minuteur de tâches (Lambda, Amazon SNS) **Note** Ces exemples de projet et la documentation associée remplacent les didacticiels qui décrivaient la mise en œuvre de la même fonctionnalité.	11 janvier 2018
Mettre à jour	Ajout d'une section Bonnes pratiques qui contient des informations permettant d'éviter le blocage des exécutions. veuillez consulter Meilleures pratiques pour Step Functions.	5 janvier 2018
Mettre à jour	Ajout d'une remarque sur la façon dont les nouvelles tentatives peuvent affecter les prix : **Note** Les nouvelles tentatives sont traitées comme des transitions d'état. Pour plus d'informations sur l'impact des transitions entre États sur la facturation, consultez Step Functions Pricing.	8 décembre 2017

Modification	Description	Date de modification
Mettre à jour	Ajout d'informations sur les noms de ressources. **Note** Step Functions vous permet de créer des noms pour les machines d'état, les exécutions et les activités, ainsi que des étiquettes contenant des caractères non ASCII. Ces noms non ASCII ne fonctionnent pas avec Amazon. CloudWatch Pour être sûr de pouvoir suivre CloudWatch les métriques, choisissez un nom qui utilise uniquement des caractères ASCII.	6 décembre 2017
Mettre à jour	Amélioration des informations générales sur la sécurité et ajout d'une rubrique sur les autorisations IAM granulaires. Consultez Sécurité dans AWS Step Functions et Création d'autorisations IAM granulaires pour les utilisateurs non administrateurs.	27 novembre 2017
Nouvelle fonction	Vous pouvez mettre à jour une machine d'état existante. Voir Mettre à jour votre machine d'état.	15 novembre 2017

Modification	Description	Date de modification
Mettre à jour	Ajout d'une note pour clarifier les erreurs `Lambda.Unknown` et d'un lien vers la documentation Lambda dans les sections suivantes : • Noms des erreurs • Étape 3 : Création d'une machine à états avec un champ Catch **Note** Les erreurs non gérées dans Lambda sont signalées `Lambda.Unknown` comme dans le résultat d'erreur. Il s'agit notamment out-of-memory des erreurs et des délais d'expiration des fonctions. Vous pouvez effectuer une correspondance ou `States.TaskFailed` pour gérer ces erreurs. `Lambda.Unknown` `States.ALL` Lorsque Lambda atteint le nombre maximum d'appels, l'erreur est. `Lambda.TooManyRequestsException` Pour plus d'informations sur les erreurs liées aux fonctions Lambda, consultez la section Gestion des erreurs et tentatives automatiques dans le Guide du AWS Lambda développeur.	17 octobre 2017
Mettre à jour	Correction et clarification des instructions IAM et mise à jour des captures d'écran de tous les didacticiels.	11 octobre 2017

Modification	Description	Date de modification

Mettre à jour	• Ajout de nouvelles captures d'écran pour les résultats d'exécution de State Machine afin de refléter les modifications apportées à la console Step Functions. Réécrivez les instructions Lambda dans les didacticiels suivants pour refléter les modifications apportées à la console Lambda :
	  • Création d'une machine d'état Step Functions utilisant Lambda
	  • Création d'un observateur de statut de tâche
	  • Création d'un temporisateur de tâche
	  • Gestion des conditions d'erreur à l'aide d'une machine à états Step Functions
	• Correction et clarification des informations sur la création des machines d'état dans les sections suivantes :
	  • Création d'une machine à états d'activité à l'aide de Step Functions
		6 octobre 2017

Mettre à jour	Réécrivez les instructions IAM dans les sections suivantes pour refléter les modifications apportées à la console IAM : • Création d'un rôle IAM pour votre machine d'état • Création d'une machine d'état Step Functions utilisant Lambda • Création d'un observateur de statut de tâche • Création d'un temporisateur de tâche • Gestion des conditions d'erreur à l'aide d'une machine à états Step Functions • Création d'une API Step Functions à l'aide d'API Gateway	5 octobre 2017

Modification	Description	Date de modification
Mettre à jour	Réécriture de la section Données de la machine d'état.	28 septembre 2017
Nouvelle fonctionnalité	Les limites liées à la régulation des actions des API sont augmentées pour toutes les régions où Step Functions est disponible.	18 septembre 2017
Mettre à jour	• Correction et clarification des informations sur le démarrage de nouvelles exécutions dans tous les didacticiels. • Correction et clarification des informations dans la section Quotas liés aux comptes.	14 septembre 2017
Mettre à jour	Réécrivez les didacticiels suivants pour refléter les modifications apportées à la console Lambda : • Création d'une machine d'état Step Functions utilisant Lambda • Gestion des conditions d'erreur à l'aide d'une machine à états Step Functions • Création d'un observateur de statut de tâche	28 août 2017
Nouvelle fonctionnalité	Step Functions est disponible en Europe (Londres).	23 août 2017
Nouvelle fonctionnalité	Le flux de travail visuel des machines d'état vous permet de zoomer vers l'avant ou vers l'arrière, et de centrer le graphique.	le 21 août 2017

Modification	Description	Date de modification
Nouvelle fonctionnalité	**Important** Une exécution ne peut pas utiliser le nom d'une autre exécution pendant 90 jours. Lorsque vous effectuez plusieurs `StartExecution` appels portant le même nom, la nouvelle exécution ne s'exécute pas. Pour plus d'informations, consultez le paramètre de `name`requête de l'action d'`StartExecution` API dans la référence AWS Step Functions d'API.	18 août 2017
Mettre à jour	Ajout d'informations sur un autre moyen de transmettre l'ARN de la machine d'état au didacticiel Création d'une API Step Functions à l'aide d'API Gateway.	17 août 2017
Mettre à jour	Ajout du nouveau didacticiel Création d'un observateur de statut de tâche.	le 10 août 2017
Nouvelle fonctionnalité	• Step Functions émet la `ExecutionThrottled` CloudWatch métrique. Pour plus d'informations, consultez Surveillance des fonctions Step Functions à l'aide de CloudWatch. • Nouvelle section Quotas liés à l'étranglement de l'État ajoutée.	3 août 2017
Mettre à jour	Mise à jour des instructions dans la section Étape 1 : créer un rôle IAM pour API Gateway.	18 juillet 2017
Mettre à jour	Correction et clarification des informations dans la section Choice.	23 juin 2017

Modification	Description	Date de modification
Mettre à jour	Des informations sur l'utilisation des ressources sous d'autres AWS comptes ont été ajoutées aux didacticiels suivants : • Création d'une machine d'état Step Functions utilisant Lambda • Création d'une machine à états Lambda pour Step Functions à l'aide de AWS CloudFormation • Création d'une machine à états d'activité à l'aide de Step Functions • Gestion des conditions d'erreur à l'aide d'une machine à états Step Functions	22 juin 2017
Mettre à jour	Correction et clarification des informations dans les sections suivantes : • Gestion des conditions d'erreur à l'aide d'une machine à états Step Functions • States • Gestion des erreurs dans Step Functions	21 juin 2017
Mettre à jour	Tous les didacticiels ont été réécrits pour les adapter à l'actualisation de la console Step Functions.	12 juin 2017
Nouvelle fonctionnalité	Step Functions est disponible en Asie-Pacifique (Sydney).	8 juin 2017
Mettre à jour	La section Amazon States Language a été restructurée.	7 juin 2017
Mettre à jour	Correction et clarification des informations dans la section Création d'une machine à états d'activité à l'aide de Step Functions.	6 juin 2017
Mettre à jour	Exemples de code corrigés dans la section Exemples de machines à états utilisant Retry et Catch.	5 juin 2017

Modification	Description	Date de modification
Mettre à jour	Restructuration de ce guide en utilisant les normes de AWS documentation.	31 mai 2017
Mettre à jour	Correction et clarification des informations dans la section Parallèle.	25 mai 2017
Mettre à jour	Fusion des sections Chemins d'accès et filtres dans la section Traitement des entrées et des sorties dans Step Functions.	24 mai 2017
Mettre à jour	Correction et clarification des informations dans la section Surveillance des fonctions Step Functions à l'aide de CloudWatch.	15 mai 2017
Mettre à jour	Mise à jour du `GreeterActivities.java` code des travaux dans le didacticiel Création d'une machine à états d'activité à l'aide de Step Functions.	9 mai 2017
Mettre à jour	Ajout d'une vidéo de présentation à la section Qu'est-ce que c'est AWS Step Functions ?.	19 avril 2017
Mettre à jour	Correction et clarification des informations dans les didacticiels suivants : • Création d'une machine d'état Step Functions utilisant Lambda • Création d'une machine à états d'activité à l'aide de Step Functions • Gestion des conditions d'erreur à l'aide d'une machine à états Step Functions	19 avril 2017

Modification	Description	Date de modification
Mettre à jour	Des informations sur les modèles Lambda ont été ajoutées aux didacticiels Création d'une machine d'état Step Functions utilisant Lambda etGestion des conditions d'erreur à l'aide d'une machine à états Step Functions.	6 avril 2017
Mettre à jour	Modification de la limite « Taille maximale des données d'entrée/de résultat » en « Taille maximale des données d'entrée/de résultat pour une tâche, un état ou une exécution » (32 768 caractères). Pour plus d'informations, consultez Quotas liés à l'exécution des tâches.	31 mars 2017
Nouvelle fonctionnalité	• Step Functions prend en charge l'exécution de machines à états en définissant Step Functions comme cibles Amazon CloudWatch Events.	21 mars 2017
Nouvelle fonctionnalité	• Step Functions permet de gérer les erreurs de la fonction Lambda en tant que méthode de gestion des erreurs préférée. • Mise à jour du didacticiel Gestion des conditions d'erreur à l'aide d'une machine à états Step Functions et de la section Gestion des erreurs dans Step Functions.	16 mars 2017
Nouvelle fonctionnalité	Step Functions est disponible en Europe (Francfort).	7 mars 2017
Mettre à jour	Réorganisation des rubriques dans la table des matières et mise à jour des didacticiels suivants : • Création d'une machine d'état Step Functions utilisant Lambda • Création d'une machine à états d'activité à l'aide de Step Functions • Gestion des conditions d'erreur à l'aide d'une machine à états Step Functions	23 février 2017

Modification	Description	Date de modification
Nouvelle fonctionnalité	• La page State Machines de la console Step Functions inclut les boutons Copy to New et Delete. • Mise à jour des captures d'écran pour correspondre aux modifications de la console.	23 février 2017
Nouvelle fonctionnalité	• Step Functions prend en charge la création d'API à l'aide d'API Gateway. • Ajout du didacticiel Création d'une API Step Functions à l'aide d'API Gateway.	14 février 2017
Nouvelle fonctionnalité	• Step Functions prend en charge l'intégration avec AWS CloudFormation. • Ajout du didacticiel Création d'une machine à états Lambda pour Step Functions à l'aide de AWS CloudFormation.	10 février 2017
Mettre à jour	Clarification du comportement actuel des champs `ResultPath` et `OutputPath` par rapport aux états `Parallel`.	6 février 2017
Mettre à jour	• Clarification des restrictions de dénomination de machine d'état dans les didacticiels. • Correction de certains exemples de code.	5 janvier 2017
Mettre à jour	Exemples de fonctions Lambda mis à jour pour utiliser le dernier modèle de programmation.	9 décembre 2016
Première version	Version initiale de AWS Step Functions.	1er décembre 201

Les traductions sont fournies par des outils de traduction automatique. En cas de conflit entre le contenu d'une traduction et celui de la version originale en anglais, la version anglaise prévaudra.