

AWS Step Functions ワークショップ

2022/07/28

シニアエバンジェリスト 亀田

[Hello World]の作成

Lambda を使わずに Step Functions による Hello World です

1. Step Functions のマネージメントコンソールに移動します
2. [ステートマシンの作成]をおします
3. [ワークフローを視覚的に設計]を選びます

作成方法を選択

ワークフローを視覚的に設計 ☒

Step Functions Workflow Studio と一緒にワークフローをドラッグアンドドロップします。 **新規**

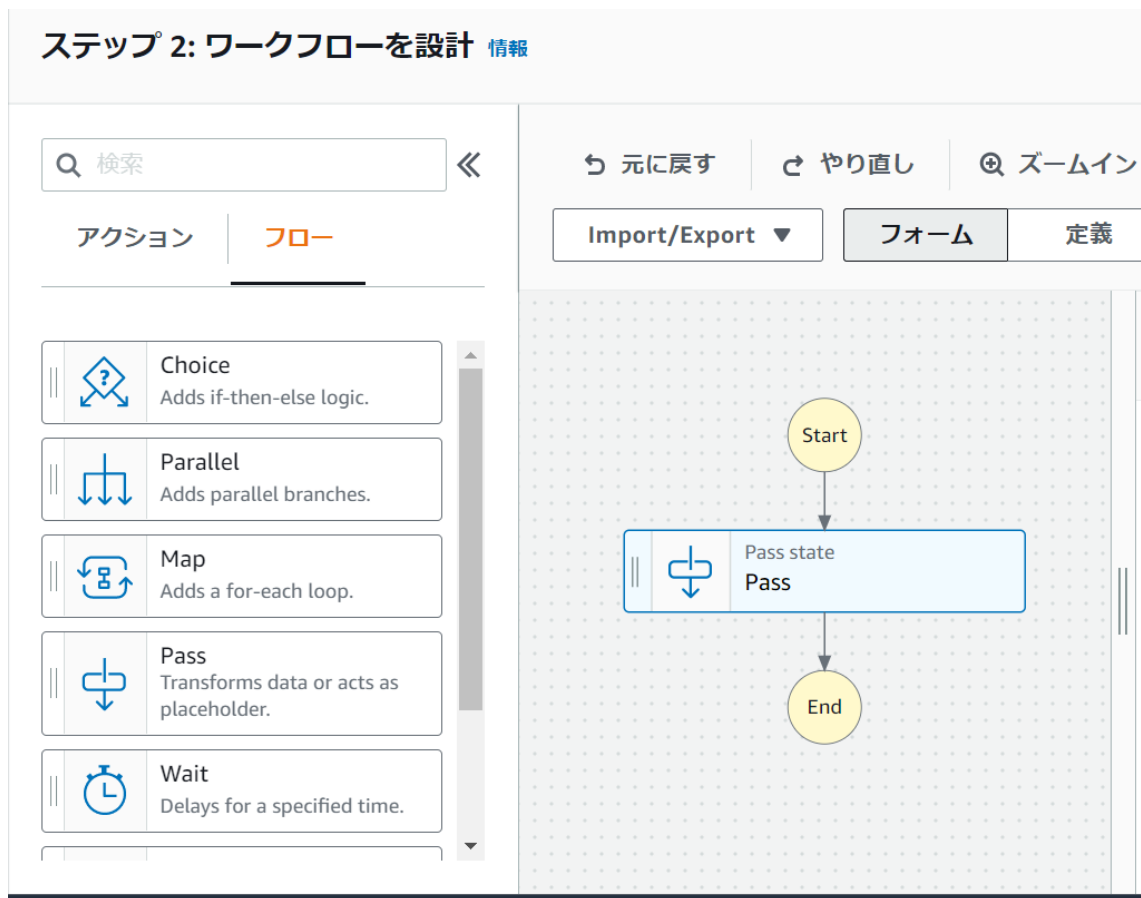
コードでワークフローを記述 ☐

Amazon ステートメント言語を使用してワークフローを作成します。コードスニペットを生成して、ワークフローステップを簡単に構築できます。

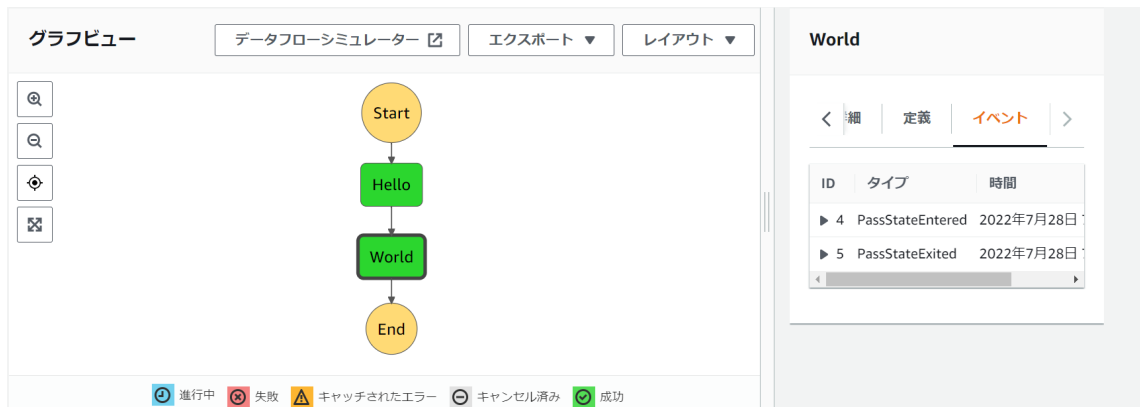
サンプルプロジェクトを実行 ☐

CloudFormation を使用すると、数分で完全に機能するサンプルプロジェクトをデプロイおよび実行できます。

4. [次へ]をおします
5. 左ペインから[フロー]のタブを選び、[Pass]を真ん中にドラッグします。以下のようになります



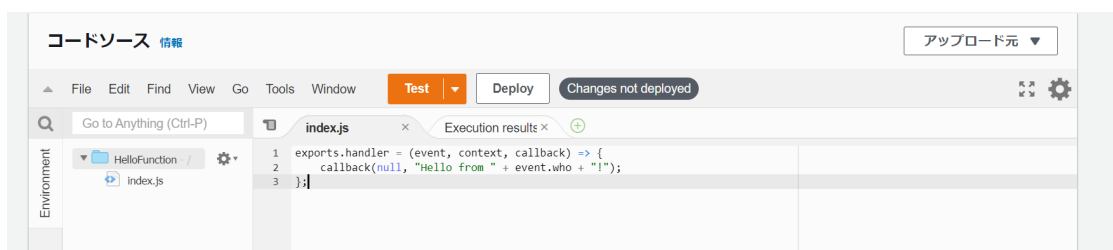
6. [次へ]をおします
7. コードスニペット部分を `commands.txt` の 1 番に置き換えます
8. [次へ]をおします
9. 名前に[HelloWorld]と入力します
10. [ステートマシンの作成]をおします
11. [実行の開始]をおします
12. もう一度[実行の開始]をおします
13. 実行結果が出力されますので、グラフビューなどを眺めてください。また[Table View][Event View]等何が出力されているかも確認してください



Lambda の起動

ここからの手順では Step Functions から Lambda 関数を起動するステートマシンを作成していきます

14. ブラウザ別タブで Lambda のマネージメントコンソールに移動します
15. [関数の作成]をおします
16. [HelloFunction]と関数名に入力します
17. [関数の作成]をおします
18. 関数が作成出来たら、コードソースの index.js を commands.txt の 2 番に置換して [Deploy]をおします








19. [Test]をおし、イベント名に[test]と入力します。[イベント JSON]に commands.txt3 番をコピーして、[保存]をおします
20. もう一度[Test]をおします。"Hello from AWS Step Functions!"と表示されたら成功です。
21. Step Functions の画面に戻り [ステートマシンの作成]をおします
22. [次へ]をおします
23. アクションから [Lambda Invoke]を真ん中にドラッグします

情報

アクション フロー

人気の高い順

- | | |
|-------------------------------------------------------------------------------------|--------------------------------------|
|  | AWS Lambda
Invoke |
|  | Amazon SNS
Publish |
|  | Amazon ECS
RunTask |
|  | AWS Step Functions
StartExecution |
|  | AWS Glue
StartJobRun |

[🔙 元に戻る](#)

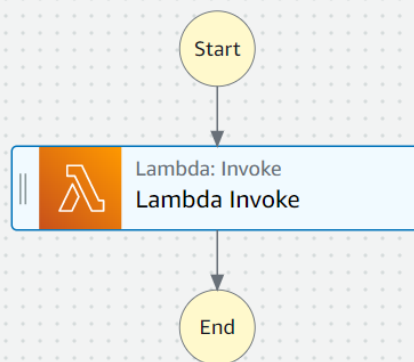
🔄 やり直し

🔍 **ズームイ**

Import/Export

フォーム

定義



ついでに、サポートされている AWS サービスを眺めてください。主要サービスがかなり多くサポートされていることがわかります。

24. 右の設定タブから[Function name]に先程作成した Lambda 関数をセットします

API パラメータ

☐ Edit as JSON

Function name

The Lambda function to invoke

▼



ペイロード

Lambda 関数に提供する JSON。

▼

25. [次へ]をおします

26. [次へ]をおします
27. 名前に[LambdaStateMachine]と入力し[ステートマシンの作成]をおします
28. [実行の開始]をおします。次のダイアログでも同様に[実行の開始]をおします
グラフビューの[Lambda Invoke]の個所が進行中を表す水色になっていますが、ほどなく成功を表す緑色に代わります。
29. 出力が[“Hello from undefined”]になっています。Lambda 関数に Input を渡さなかったせいです



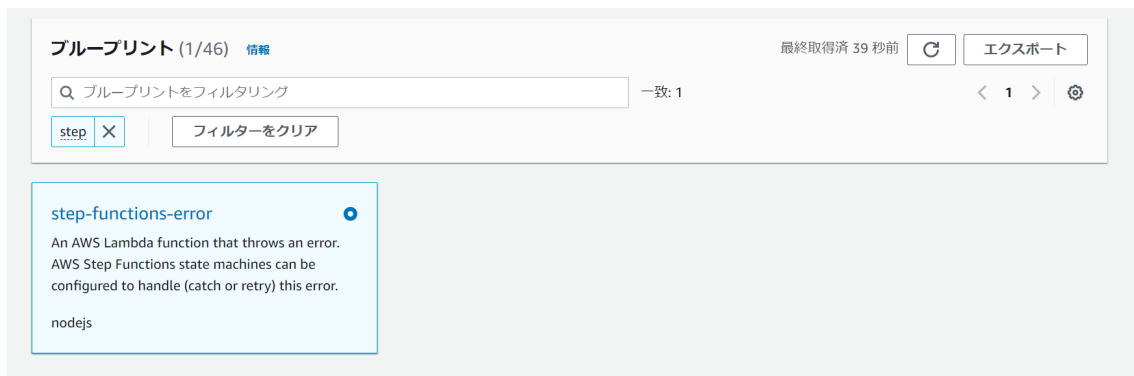
30. [新しい実行]のボタンをおします
31. 入力の commands.txt3 番で置換します（先ほど Lambda のテストにセットしたものと
同じものです）
32. [実行の開始]をおします
33. 今度は正しく出力されています



Catch によるエラー処理

次のステップでは意図的に実行が失敗する Lambda 関数を作成しエラーイベントを発生させ、Catch 処理により次の Lambda 関数を実行させます

34. Lambda マネージメントコンソールから[関数の作成]をおします
35. [設計図の使用]を選びます
36. [step-functions-error]のブループリントを選び[設定]をおします



37. 名前に[FailFunction]とつけます
38. [関数の作成]をおします
39. [Test]をおします
40. イベント名に[test]と入力し、[保存]をおします
41. 再度[Test]を実行します。以下のように意図したエラーが表示されます



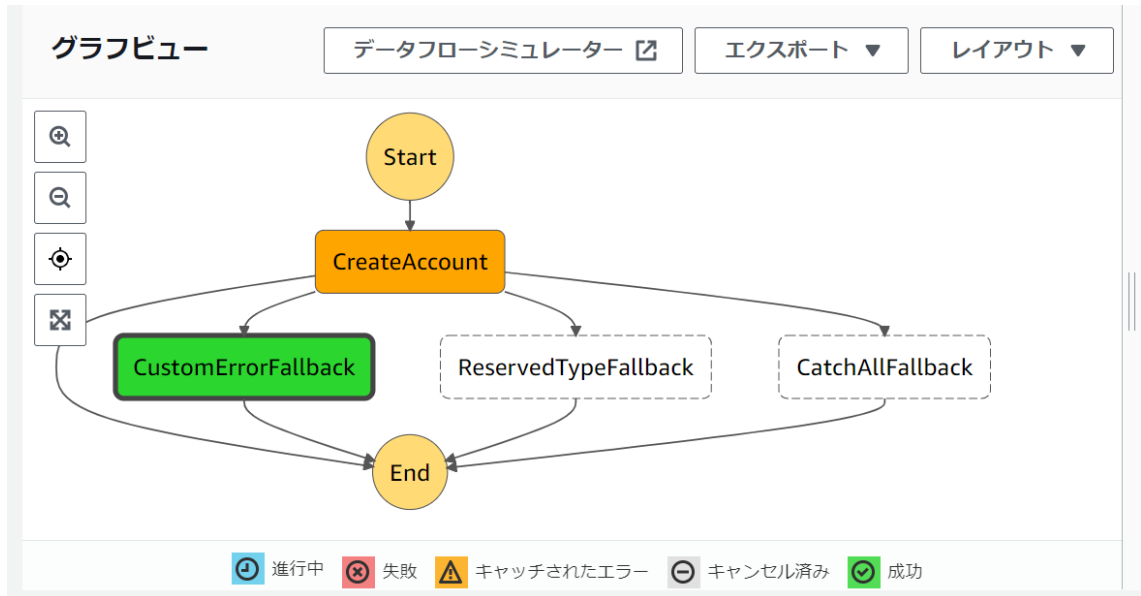
42. Lambda 関数の ARN をコピーしておきます



43. Step Functions のマネージメントコンソールから[ステートマシンの作成]をおします
44. [コードでワークフローを記述]をおします
45. Commands.txt の 4 番の内容に置換します。その際必ず ARN の値を先程コピーしたものに置き換えてください
46. このステートマシンでは[CreateAccount]をまず実行し、失敗したらエラーに応じて 3

つの処理のどれかを条件に応じて実行させます。[CreateAccount]を作成しておらず当然実行が失敗しますがこのサンプルでは[CustomErrorFallback]が先程の Lambda 関数にイベントとして引き渡され関数が実行されます。

47. [次へ]を押して[Catchfailure]を名前をつけ[ステートマシンの作成]をおします
48. [実行の開始]をおします。もう一度[実行の開始]をおします
49. 以下のようにエラーがキャッチされ Lambda 関数が実行されたことがわかります

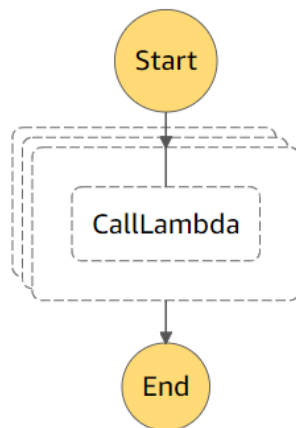


Map 機能にする Lambda 関数の複数回実行

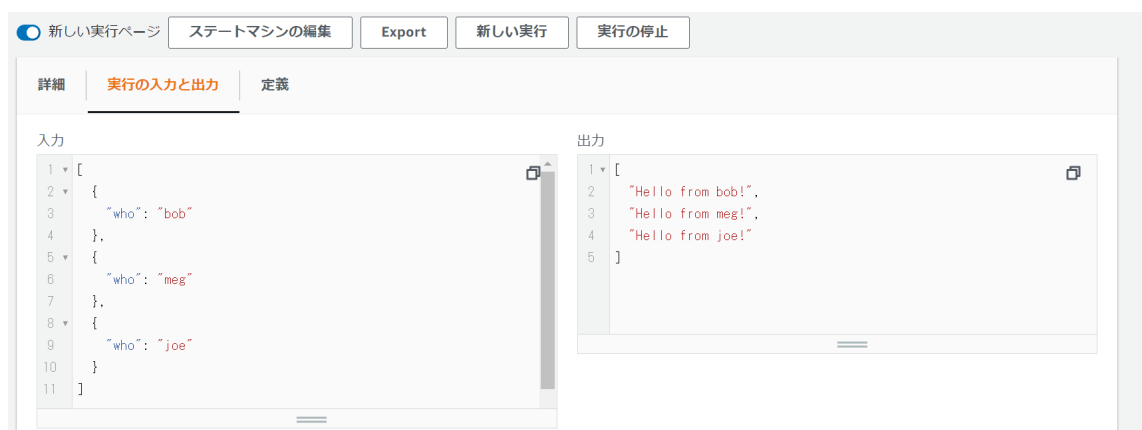
ステートマシンではあらかじめ実行回数がわからず、与えられたパラメータの数（行数）により任意の複数回 Lambda 関数を実行したいケースがあります。Map 機能では Lambda を任意の複数回実行し結果を結合して表示させることが可能です

50. [ステートマシンの作成]をおします
51. [コードでワークフローを記述]を選びます
52. Commands.txt の 5 番でコードを置換します。この際 Lambda 関数[Hellofunction]の ARN を必ず置換します

```
callLambda : {  
  "Type": "Task",  
  "Resource": "arn:aws:lambda:ap-northeast-  
1:294963776963:function:HelloFunction",  
  "End": true  
}
```



53. [次へ]をおします
54. 名前に[ExampleMapState]と入力し[ステートマシンの作成]をおします
55. [実行の開始]をおします。JSON の内容を commands.txt 6 番で置換し、[実行の開始]をおします
56. Lambda 関数が 3 回実行され出力が 3 行行われています



EvenBridge 連携

Step Functions は EventBridge と連携することで決められた時間での実行化が可能です

57. ブラウザの新しいタブで EventBridge を開きます
58. [ルールを作成]をおします
59. 名前に適当な文字を入力します
60. ルールタイプでスケジュールを選びます

ルールタイプ | 情報

☐ イベントパターンを持つルール
定義したイベントパターンとイベントが一致したときに実行されるルール。EventBridge は指定されたターゲットにイベントを送信します。

☒ スケジュール
スケジュールに従って実行されるルール

61. [次へ]をおします

62. Cron を以下のように 10-分後ぐらいに実行されるように設定します

Cron 式 | 情報

スケジュールの cron 式を定義

☐ cron (10 23 * * ? 2022)
分 時間 日付 月 曜日 年

以後 10 回のトリガー日

☐ ローカルタ... ▼

2022年7月28日 08:10 JST

2022年7月29日 08:10 JST

2022年7月30日 08:10 JST

2022年7月31日 08:10 JST

2022年8月1日 08:10 JST

2022年8月2日 08:10 JST

2022年8月3日 08:10 JST

2022年8月4日 08:10 JST

2022年8月5日 08:10 JST

2022年8月6日 08:10 JST

63. ターゲットを以下のように設定します。ステートマシン名は任意のもので大丈夫です

ターゲットタイプ

EventBridge イベントバス、EventBridge API の宛先 (SaaS パートナー)、または別の AWS のサービスをターゲットとして選択します。

- ☐ EventBridge イベントバス
- ☐ EventBridge API の宛先
- ☒ AWS のサービス

ターゲットを選択 [情報](#)

イベントがイベントパターンと一致したとき、またはスケジュールがトリガーされたときに呼び出すターゲットを選択します (ルールごとに 5 個のターゲットに制限されます)。

Step Functions ステートマシン ▼

ステートマシン

HelloWorld ▼

64. [次へ] を 2 回押します。その後[ルールの作成]をおします
65. ルールのモニタリングタブを開き実行されるのを待ちます。最大 1 分程度のずれがあります。あらにそこからグラウが表示されるまで数分の遅延が発生します。
66. 実行が確認出来たら[削除]をおしルールを削除しておきます

S3 との連携

次に S3 バケットのイベントを EventBridge で取得し StateMachine を起動します

67. S3 のマネージメントコンソールに移動し、[バケットを作成]をおし、適当名前でバケットを作成します
68. 先程作成したバケットのプロパティ画面に移動します

20220729hkameda [情報](#)

[オブジェクト](#) | [プロパティ](#) | [アクセス許可](#) | [メトリクス](#) | [管理](#) | [アクセスポイント](#)

バケットの概要

AWS リージョン アジアパシフィック (東京) ap-northeast-1	Amazon リソースネーム (ARN)  arn:aws:s3:::20220729hkameda	作成日 2022/07/29 06:48:14 AM JST
--------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------

69. イベント通知の個所で、[編集]ボタンをおします
70. [オン]を選び[変更の保存]をおします
71. EventBridge のマネージメントコンソールで[ルールを作成]をおします
72. 名前に[S3StepFunctions]と入力します

73. [次へ]をおします

74. イベントパターンの個所を以下のように設定します

イベントソース

ソースとして AWS のサービスまたは EventBridge パートナー

AWS のサービス ▼

AWS のサービス


イベントソースとしての AWS のサービスの名前

Simple Storage Service (S3) ▼

イベントタイプ

一致パターンのソースとしてのイベントのタイプ

Amazon S3 イベント通知 ▼

 S3 イベント通知は、イベント通知を EventBridge に発行するように S3 バケットを設定している場合にのみルールに一致します。 [詳細はこちら](#)。

☐ 任意のイベント

☒ 特定のイベント

Object Created ✕

☐ 任意のバケット

☒ 特定のバケット (名前別)

削除

75. [次へ]をおします

76. ターゲットの個所で、任意の Step Functions ステートマシンを選びます

ターゲット 1

ターゲットタイプ
EventBridge イベントバス、EventBridge API の宛先 (SaaS パートナー)、または別の AWS のサービスをターゲットとして選択します。

☐ EventBridge イベントバス

☐ EventBridge API の宛先

☒ AWS のサービス

ターゲットを選択 [情報](#)
イベントがイベントパターンと一致したとき、またはスケジュールがトリガーされたときに呼び出すターゲットを選択します (ルールごとに 5 個のターゲットに制限されます)。

ステートマシン

77. [次へ]を 2 回押し、[ルールの作成]をおします

78. 何でもよいのでファイルを S3 にブラウザ経由マネージメントコンソールからアップ

すると、EventBridge を介して StepFunctions のステートマシンが実行されます。
StepFunctions ステートマシンの画面で最新実行日時が確認できます

実行 ログ記録 定義 タグ				
実行 (5)				
<div>実行の検索</div> <div>ステータスでフィルタ</div>				
<div>実行の開始</div>				
<div>詳細を表示</div>				
<div>実行の停止</div>				
<div>1</div>				
名前	ステータス	開始	終了時間	
94c948c9-e791-7c94-9f34-ad65daea292e_9bebef71-45b6-20b6-724c-ef15d156f95e	成功	2022年7月29日 金曜日 午前 6:57:19.124	2022年7月29日 金曜日 午前 6:57:19.221	

EventBridge のモニタリングは 5 分程度遅延したのち反映されます

次に API Gateway 経由で StepFunctions のステートマシンを起動してみます

79. ブラウザ別タブで IAM マネージメントコンソールに移動します

80. [ロールを作成]をおします

81. ユースケースで API Gateway を選び、[次へ]を 2 回おします

ユースケース

EC2、Lambda、その他の AWS のサービスがこのアカウントでアクションを実行することを許可します。

一般的なユースケース

- ☐ EC2
Allows EC2 instances to call AWS services on your behalf.
- ☐ Lambda
Allows Lambda functions to call AWS services on your behalf.

他の AWS のサービスのユースケース:

- ☐ API Gateway
Allows API Gateway to push logs to CloudWatch Logs.

82. 名前に[APIGatewayToStepFunctions]と入力し[ロールを作成]をおします。以下に表示される ARN をコピーしておいてください

IAM > ロール > APIGatewayToStepFunctions

APIGatewayToStepFunctions

Allows API Gateway to push logs to CloudWatch Logs.

概要

作成日
July 29, 2022, 07:35 (UTC+09:00)

最後のアクティビティ
なし

ARN をコピーしました


arn:aws:iam::294963776963:role/APIGatewayToStepFunctions

最大セッション時間
1 時間

83. [許可の追加]から[AWSStepFunctionsFullAccess]を付与します



許可 | 信頼関係 | タグ | アクセスアドバイザー | セッションを取り消す


許可ポリシー (2)
最大 10 個の管理ポリシーを添付できます。

 シミュレート 削除

許可を追加 ▼

< 1 > 

<input type="checkbox"/>	ポリシー名 	タイプ	説明
<input type="checkbox"/>	 AmazonAPIGatewayPushToCloudWatchLogs	AWS 管理	Allows API Gateway to pus
<input type="checkbox"/>	 AWSStepFunctionsFullAccess	AWS 管理	An access policy for provic



84. ブラウザの別タブで API Gateway にアクセスします

85. [API の作成]をおします

86. REST API の[構築]をおします

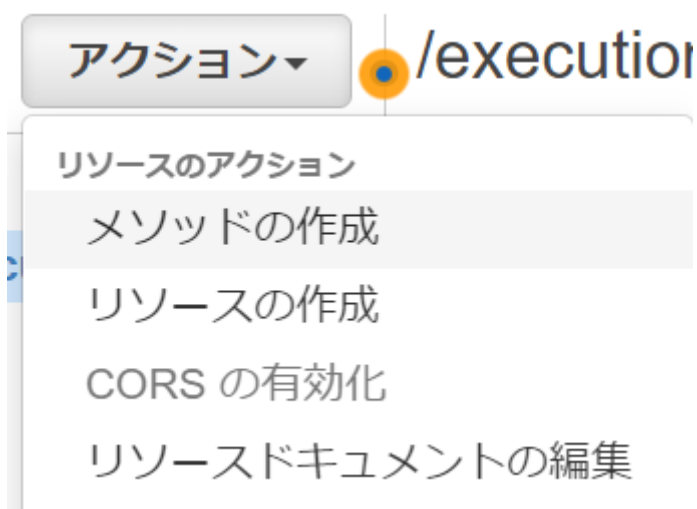
87. 名前に[StartExecutionAPI]と入力します

88. [API の作成]をおします

89. [アクション][リソースの作成]をおします



90. リソース名に[execution]と入力します
91. [リソースの作成]をおします
92. 今度はメソッドの作成を選び POST メソッドを作成します



93. 統合タイプで[AWS サービス]を選びます

- 統合タイプ**
- ☐ Lambda 関数 ⓘ
 - ☐ HTTP ⓘ
 - ☐ Mock ⓘ
 - ☒ AWS サービス ⓘ
 - ☐ VPC リンク ⓘ

94. 以下の画面に従い同じ設定を行います

AWS リージョン	ap-northeast-1	▼
AWS サービス	Step Functions	▼
AWS サブドメイン		
HTTP メソッド	POST	▼
アクションの種類	<input checked="" type="radio"/> アクション名の使用 <input type="radio"/> パス上書きの使用	
アクション	StartExecution	
実行ロール	arn:aws:iam::294963776963:role/service-role/Amazon_Ever ⓘ	
コンテンツの処理	パススルー	▼ ⓘ

IAM ロール ARN は先程作成したものを貼り付けてください

95. [保存]をおします

96. Commands.txt の 7 番を[リクエスト本文]に貼り付けて[テスト]をおします。この際
ARN は StepFunctions の HelloWorld Satemachine の ARN を入力してください

97. テストが以下のように戻りになれば設定完了です

リクエスト: /execution

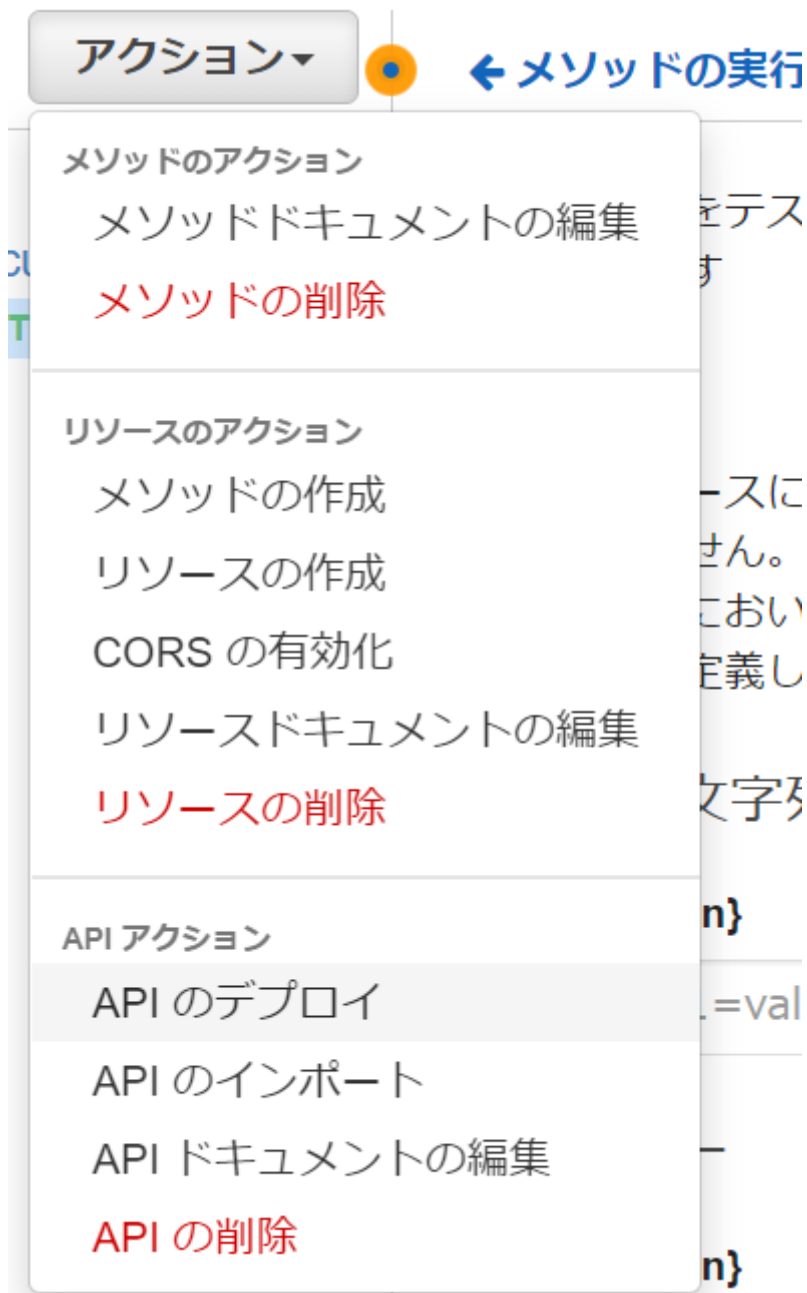
ステータス: 200

レーテンシー: 168 ms

レスポンス本文

```
{
  "executionArn": "arn:aws:states:ap-northeast-1:294963776963:execution:HelloWorld:MyExecution",
  "startDate": 1659048764.365
}
```

98. [アクション]から[API のデプロイ]をおします



99. 以下のように設定し[デプロイ]をおします

API のデプロイ

×

API がデプロイされるステージを選択します。たとえば、API のテスト版をベータという名前のステージにデプロイできます。

デプロイされるステージ	<div>[新しいステージ] ▼</div>
ステージ名*	<div>alpha</div>
ステージの説明	<div></div>
デプロイメントの説明	<div></div>

キャンセル

デプロイ

100. Curl で commands.txt 8 番を実行してみます。ARN、HTTPS 二つの値は皆さんの環境の値で置き換えてください（Already exists のエラーが戻ってきた人は MyExecution2 の値を適当に変えてください

おつかれさまでした！：

削除は以下を行ってください

S3 バケット

Step Functions ステートマシン

IAM ロール

EventBridge ルール

Lambda 関数

API Gateway