

Servelesspresso Compass Progress Report:

Overview:

Serverlesspresso

- Descrição: Uma cafeteria pop-up que oferece bebidas espresso premium em conferências e eventos, com uma equipe de elite de baristas preparada para servir até 1.000 bebidas por dia.

Objetivo do Projeto

- Aplicativo Serverless: O objetivo é construir um aplicativo serverless para aceitar pedidos e notificar os clientes quando suas bebidas estiverem prontas, garantindo que a solução seja robusta e escalável.

Funcionamento do Café

- Processo de Pedido:

- Monitores exibem um código QR que muda a cada 5 minutos.
- Clientes escaneiam o código QR para fazer pedidos, válidos para 10 bebidas por 5 minutos.
- O pedido é feito no aplicativo web, validado pelo backend e um número de pedido é criado.
- Baristas veem os pedidos em seu aplicativo e podem atualizar o status (em andamento, concluído ou cancelado).
- Clientes recebem atualizações sobre o status do pedido em seus dispositivos móveis e nos monitores.

Estrutura da Aplicação

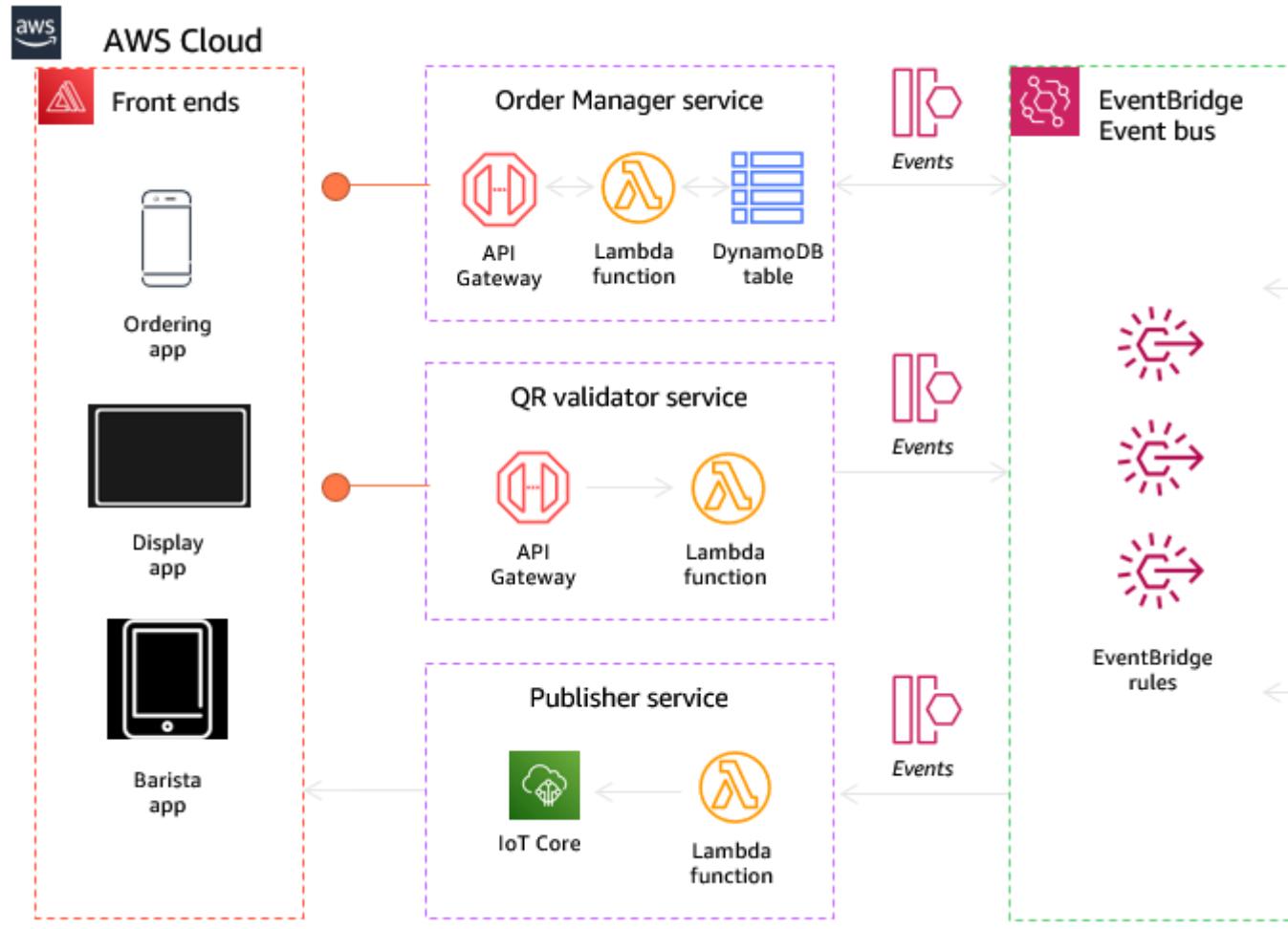
- Microsserviços: Vários microsserviços integrarão os frontends existentes com o backend serverless, utilizando AWS Step Functions para orquestração e Amazon EventBridge para coreografia.

Frontends

- Aplicativo de Exibição: Mostra um código QR para pedidos e exibe uma fila em tempo real de pedidos futuros e concluídos.
- Aplicativo Barista: Usado pelos baristas para alterar o status de pedidos e cancelar se necessário. As atualizações são propagadas para outros aplicativos.
- Aplicativo de Pedidos: Utilizado pelos clientes para fazer pedidos, projetado para dispositivos móveis.

Backend

- Arquitetura: Utiliza AWS Step Functions, Amazon EventBridge, AWS Lambda, Amazon API Gateway, Amazon S3, Amazon DynamoDB e Amazon Cognito. O JavaScript roda no frontend, interagindo com a API backend via API Gateway, enquanto o DynamoDB gerencia a persistência dos dados. Eventos são enviados de volta aos frontends usando AWS IoT Core e Lambda.



Custo:

- Conta Temporária da AWS: Se o workshop estiver sendo realizado em um evento da AWS, você receberá uma conta temporária, e não haverá custos. Os recursos serão excluídos automaticamente após o workshop, portanto, não é necessário fazer limpeza.
- Conta Pessoal da AWS: Se você estiver implantando o workshop em sua própria conta, será responsável pelos custos incorridos. A maioria dos serviços pode estar coberta pelo AWS Free Tier para contas qualificadas. Para minimizar custos após o workshop, é recomendado seguir as etapas de limpeza para excluir recursos.
- Custos Esperados em Produção: Para 1.000 clientes por dia, os custos diários estimados para executar o aplicativo são:
 - Console do AWS Amplify: \$0,28 (Livre)
 - Gateway de API da Amazon: \$0,01 (Livre)
 - Amazon Cognito: Livre

- Amazon DynamoDB: \$0,01 (Livre)
 - Amazon EventBridge: \$0,01 (Livre)
 - Núcleo de IoT da AWS: \$0,01 (Livre)
 - AWS Lambda: \$0,01 (Livre)
 - Funções de Etapa da AWS: \$0,29 (Livre)
- Nota: Essas estimativas são apenas um guia.
-

MÓ

Após concluir a seção *Configuração*, siga os módulos na ordem:

Módulo #	Recurso	Descrição
Esta seção	Introdução	Saiba mais sobre o aplicativo.
Configurar	Configurar	Pré-requisitos e requisitos.
1a	Construindo o fluxo de trabalho - Parte 1	Comece a criar o fluxo.
1b	Construindo o fluxo de trabalho - Parte 2	Conclua e teste o fluxo.
2	Roteamento de eventos com EventBridge	Usando eventos para...
3	Configurar os front ends	Crie um serviço que...

Configurar (Visão Geral):

O aplicativo Serverlesspresso consiste em três frontends e um aplicativo backend. Os frontends já estão construídos e implantados em uma conta da AWS, enquanto alguns recursos de backend serão pré-implantados durante as seções de configuração.

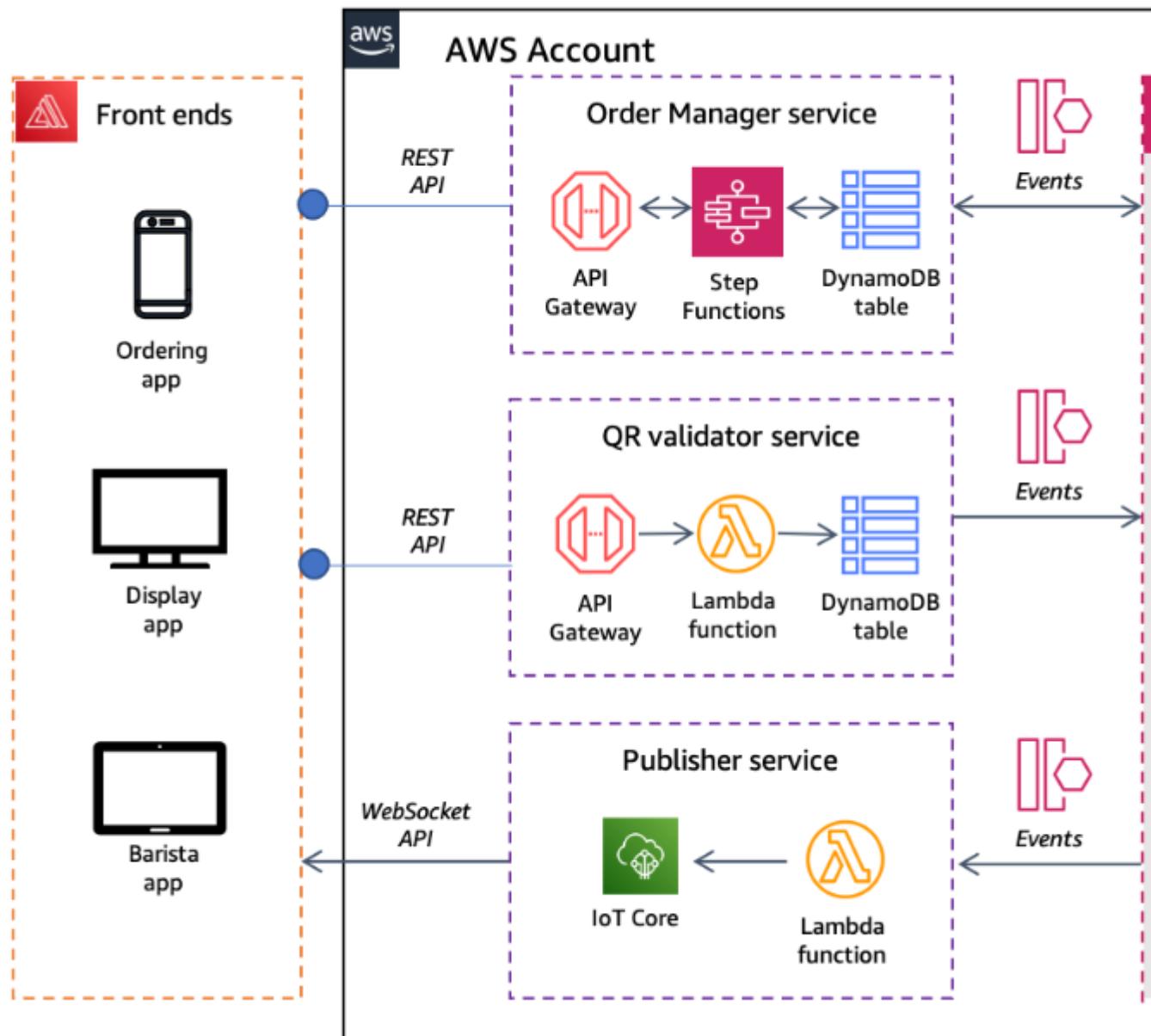
Componentes do Backend

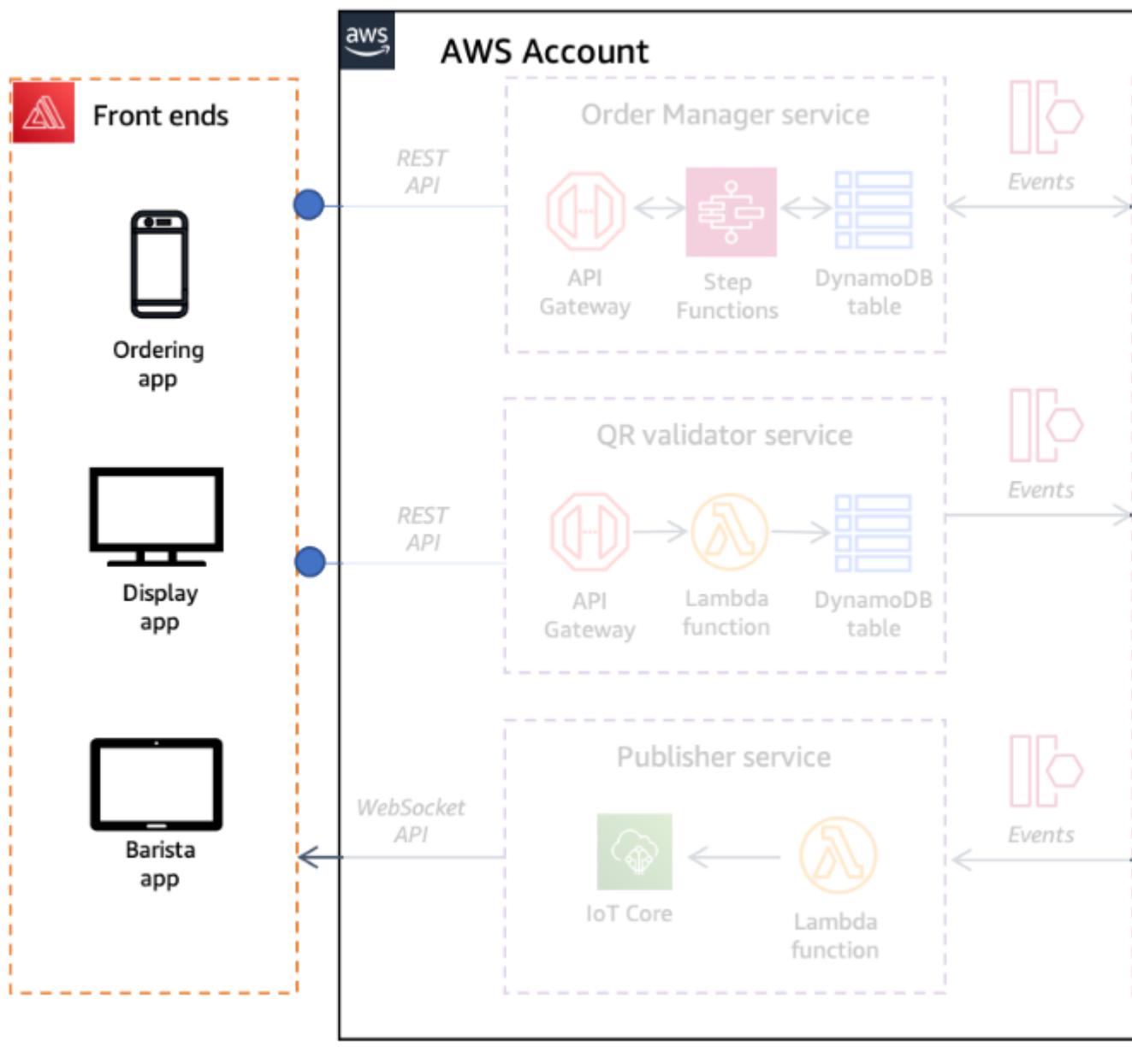
O backend é composto por um conjunto de microsserviços sem servidor:

- Microsserviço de Contagem: Usa uma tabela do Amazon DynamoDB para incrementar números de pedidos sequencialmente.
- Microsserviço OrderManager: Fornece uma API para enviar, atualizar e cancelar pedidos de café, utilizando uma tabela DynamoDB que contém o estado de cada pedido do cliente.

- Microsserviço Config: Utiliza uma tabela do DynamoDB para informações sobre itens de menu e status da loja, além de um recurso do Amazon API Gateway para acesso autenticado.
- Microsserviço Publisher: Roteia eventos para diferentes tópicos do IoT Core, publicando mensagens de eventos para aplicativos front-end.
- Microsserviço QR Validator: Fornece códigos QR para o aplicativo de exibição front-end, armazenados em uma tabela do DynamoDB e usados para validar cada pedido.
- Microsserviço OrderProcessor: Um fluxo de trabalho do AWS Step Functions que orquestra cada pedido do cliente do início ao fim.

A lógica de roteamento de eventos direciona eventos para o serviço downstream correto. Após a criação dos recursos de backend, a configuração do aplicativo front-end será atualizada para consultar o ponto de extremidade do API Gateway e exibir informações sobre o menu atual e o status dos pedidos.





Auto-Hospedado:

1. Inicie o Modelo AWS CloudFormation

Este workshop pode ser executado dentro de suas próprias contas AWS. Para acompanhar o workshop, será necessário instalar e configurar vários serviços da AWS. O objetivo é simplificar o provisionamento desses serviços.

Responsabilidade e Custos: Ao executar os modelos, você assume a responsabilidade pelo ciclo de vida e pelos custos associados ao provisionamento dos recursos. Siga as instruções de desmontagem para remover todos os recursos após a conclusão do workshop, evitando custos inesperados.

Uso do AWS CloudFormation: Utilizaremos o AWS CloudFormation para codificar nossa infraestrutura. Siga os passos abaixo para iniciar a pilha:

1. Selecione sua Região: Escolha a região preferida onde você implantará o modelo.
2. Clique no Link de Lançamento: Clique no link para criar a pilha em sua conta.
3. Região e Pilha de Lançamento:
 - Leste dos EUA (Norte da Virgínia): `us-east-1`
4. Insira um Nome de Pilha: Insira um nome para a pilha ou mantenha o nome padrão.
5. Marque as Caixas na Seção Capacidades: Aceite as capacidades necessárias.
6. Clique em Criar Pilha: Inicie a criação da pilha.

Região

Leste dos EUA (Norte da Virgínia) us-east-1

Quick create stack

Template

Template URL

<https://ee-assets-prod-us-east-1.s3.amazonaws.com/modules/2927cc5a8ed2422bbe540bafaa2cc381/v1/template.yaml>

Stack description

-

Stack name

Stack name

serverless-workshop

Stack name can include letters (A-Z and a-z), numbers (0-9), and dashes (-).

Parameters

Parameters are defined in your template and allow you to input custom values when you create or update a stack.

Cloud9CidrBlock

The CIDR block range for your Cloud9 IDE VPC

10.43.0.0/28

Capabilities and transforms

i Transforms might require access capabilities

A transform might add Identity and Access Management (IAM) resources that could provide entities access to make changes to your stack. If a transform adds IAM resources, you must acknowledge their capabilities to create or update them. Ensure that you want to create IAM resources, and that they have the minimum required permissions. In addition, if they have custom names, check that the names are valid within your AWS account. [Learn more](#)

- I acknowledge that AWS CloudFormation might create IAM resources.
- I acknowledge that AWS CloudFormation might create IAM resources with custom names.
- I acknowledge that AWS CloudFormation might require the following capability:
CAPABILITY_AUTO_EXPAND

Cancel

Create change set

1ª. Construindo o fluxo de trabalho – Parte 1:

Visão Geral

Por que você precisa de um fluxo de trabalho?

No aplicativo de pedidos de café, cada pedido do cliente passa por várias etapas:

1. Início do Pedido: A leitura inicial do código QR inicia o processo de pedido.
2. Verificação de Condições: O aplicativo verifica se a loja está aberta e se a fila do barista não está cheia (o barista pode atender até 20 bebidas por vez). Se a loja estiver fechada ou a fila estiver cheia, o processo de pedido é interrompido.
3. Aguardando Detalhes do Pedido: O aplicativo espera 5 minutos para que o cliente forneça detalhes do pedido. Se nada acontecer nesse período, o pedido expira.
4. Produção da Bebida: O barista tem 15 minutos para produzir a bebida. Se nada acontecer após esse tempo, o pedido expira.
5. Conclusão ou Cancelamento: O pedido é finalmente concluído ou cancelado pelo barista.

Cada pedido de bebida está em um ponto separado desse fluxo de trabalho. Tradicionalmente, implementar essa lógica no código resulta em ramificações lógicas complexas e depende de um banco de dados central para manter o controle do estado. Também é necessário um processo separado para lidar com timeouts.

Esse tipo de lógica de fluxo de trabalho é um exemplo de máquina de estado. Este workshop utiliza o AWS Step Functions para construir uma máquina de estado que gerencia todas essas etapas de forma independente e confiável, sem a necessidade de código complexo.

Como Funciona:

Os fluxos de trabalho do Step Functions são definidos usando a Amazon States Language (ASL), uma linguagem baseada em JSON que permite definir máquinas de estado, incluindo:

- Estados de Tarefa: Executam ações específicas.
- Estados de Escolha: Determinam a transição para os próximos estados.
- Estados de Falha: Interrompem a execução em caso de erro.

Os fluxos de trabalho são documentos JSON que podem ser versionados no GitHub e implantados com ferramentas como AWS SAM ou CDK.

O AWS Management Console oferece uma ferramenta visual chamada Workflow Studio, que simplifica a criação de fluxos de trabalho. Após a criação, você pode salvar para uso imediato na AWS Cloud ou exportar a definição em ASL. Este módulo utilizará o Workflow Studio para criar o fluxo de trabalho.

Criando o fluxo de trabalho:

Visão Geral

Neste módulo, você criará um novo fluxo de trabalho no AWS Step Functions e testará sua funcionalidade. Ao final, você terá um fluxo de trabalho que será utilizado para desenvolver a funcionalidade do aplicativo.

[Criando o Fluxo de Trabalho do Step Functions](#)

Instruções Passo a Passo:

1. Acesse o Console do Step Functions:

- No AWS Management Console, selecione Services e, em seguida, escolha Step Functions em Application Integration.

- Certifique-se de que sua região esteja correta.

2. Criar Máquina de Estado:

- Selecione Criar máquina de estado.

3. Etapa 1: Escolher Modo de Criação:

- Para Escolher modo de criação, selecione Projetar seu fluxo de trabalho visualmente.

- Para Tipo, selecione Padrão e clique em Avançar.

4. Etapa 2: Projetar o Fluxo de Trabalho com o Workflow Studio:

- Aba de Design: Escolha entre Actions (etapas com serviços AWS) e Flow (opções para lógica de fluxo de controle).

- Use a barra de ferramentas para desfazer ou refazer alterações.

- A visualização do fluxo de trabalho mostra o fluxo atual, onde você pode arrastar e soltar elementos.

- O painel direito exibe opções para o elemento selecionado.

5. Adicionar um Estado de Passagem:

- Selecione a aba Fluxo e arraste um estado de Passagem do menu à esquerda para a caixa que diz Arraste o primeiro estado aqui. Escolha Próximo.

6. Revisar Código Gerado:

- Na página Revisar código gerado, veja a definição do fluxo de trabalho em ASL (Amazon States Language) no painel esquerdo e o fluxograma visual à direita. Clique em Próximo.

7. Especificar Configurações da Máquina de Estado:

- Nome: Insira `OrderProcessorWorkflow`.

- Permissões: Escolha Choose an existing role e selecione a função que contém `01OrderProcessorRole`.

- Logging: Mantenha como OFF.

- Tracing: Mantenha desabilitado.

8. Criar Máquina de Estado:

- Depois de definir as opções, escolha **Criar máquina de estado.

Você criou seu primeiro fluxo de trabalho do Step Functions usando o Workflow Studio!

Testando o Fluxo de Trabalho do Step Functions

Instruções Passo a Passo:

1. Iniciar Execução:

- Na página do novo fluxo de trabalho, escolha Iniciar execução.

2. Editar JSON de Entrada:

- No pop-up Iniciar execução, edite o JSON de entrada para que o valor do comentário seja "Olá, mundo!". Clique em Iniciar execução.

3. Resultados da Execução:

- Após a conclusão, o console mostrará uma página de resultados:
 - Status de Execução: Mostra Succeeded.
 - Graph Inspector: Exibe o fluxo da execução, com o caminho destacado em verde.
 - Histórico de Eventos: Mostra cada evento durante a execução, incluindo cargas de entrada e saída

The screenshot shows the AWS Step Functions console interface. On the left, there's a sidebar with links like 'Step Functions', 'State machines' (which is highlighted in orange), 'Activities', 'Data flow simulator', 'Feature spotlight', 'Local Development', and 'Join our feedback panel'. The main content area has a breadcrumb navigation 'Step Functions > State machines'. It displays a summary 'State machines (1)' with a note that counts are based on the most recent 1000 executions. Below this are buttons for 'View details', 'Edit', and 'Copy to'. A search bar says 'Search for state machines'. At the bottom, there's a table header with columns 'Name', 'Type', and 'Create date'.

Step 1
Choose authoring
method

Step 2
Design workflow

Step 3 - optional
Review generated code

Step 4
Specify state machine
settings

Choose authoring method

Design your workflow visually

Drag and drop your workflow together with Step Functions Workflow Studio. [New](#)

Write yo

Author you
Language. V
to easily bu

Type

Standard

Durable, checkpointer workflows for machine learning, order fulfillment, IT/DevOps automation, ETL jobs, and other long-duration workloads.

► [Help me decide](#)

Step 2: Design workflow Info

The screenshot shows the AWS Step Functions Designer interface. At the top, there is a search bar labeled "Search" and a toolbar with "Undo" (highlighted with a red circle labeled "2"), "Redo", and "Zoom in". Below the toolbar, there are tabs for "Actions" (highlighted with a red circle labeled "1") and "Flow". The "Actions" tab is selected, showing a list of "Most Popular" actions:

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

Below this, under the "COMPUTE" section, there are links to "Amazon Data Lifecycle ...", "Amazon EBS", and "Amazon EC2". On the right side of the screen, there is a canvas area where a workflow is being designed. A yellow "Start" state is at the top, connected by a vertical arrow to a yellow "End" state at the bottom. A dashed box surrounds the vertical path between them, with the text "Drag first state here" inside. Red circles labeled "1", "2", and "3" are overlaid on the interface to indicate specific features or steps.

Step 2: Design workflow [Info](#)

Search Actions Flow

||  **Choice**
Adds if-then-else logic.

||  **Parallel**
Adds parallel branches.

||  **Map**
Adds a for-each loop.

||  **Pass**
Transforms data or acts as placeholder.

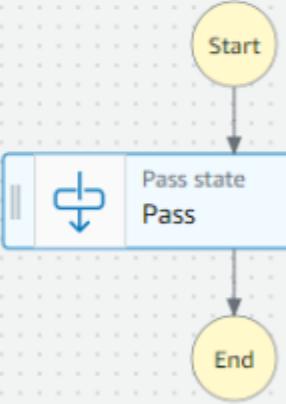
||  **Wait**
Delays for a specified time.

||  **Success**
Stops and marks as success.

||  **Fail**
Stops and marks as failure.

|| Undo || Redo || Zoom in || Z

Import/Export ▾ Form Definition



```
graph TD; Start((Start)) --> Pass[Pass state  
Pass]; Pass --> End((End))
```

Step 1

Choose authoring method

Step 2

Design workflow

Step 3 - optional

Review generated code

Step 4

Specify state machine settings

Review generated code - *optional*

Review your state machine's Amazon States Language (ASL) definition.

Definition

Generate code snippet ▾

Format JSON

```
1 {  
2     "Comment": "A description of my state machine",  
3     "StartAt": "Pass",  
4     "States": {  
5         "Pass": {  
6             "Type": "Pass",  
7             "End": true  
8         }  
9     }  
10 }
```

Name

State machine name

A

Must be 1-80 characters. Can use alphanumeric characters, dashes, or underscores.

Permissions

Execution role

The IAM role that defines which resources your state machine has permission to access during execution. To create a custom role, go to the [IAM console](#).

- Create new role
Let Step Functions create a new role for you based on your state machine's definition and configuration details.
- Choose an existing role
- Enter a role ARN

Existing roles

- mod-67b03f2bcecc4faf-01OrderProcessorRole-1SLJZSDG5H9EQ
- mod-67b03f2bcecc4faf-01OrderProcessorRole-1SLJZSDG5H9EQ
- mod-67b03f2bcecc4faf-02OrderManagerStateMachineRol-TM4P06OF7N2C

B

Logging

You can log your state machine's execution history to CloudWatch Logs. For Express state machines, you must enable logging to inspect and debug executions. CloudWatch Logs charges apply. [Learn more](#)

Log level

Indicates which execution history events to log

C

Tracing

You can enable AWS X-Ray tracing on your state machine for end-to-end application debugging, performance profiling, and error analysis. Standard X-Ray charges apply. [Learn more](#)

Enable X-Ray tracing

Step Functions will send traces to AWS X-Ray for state machine executions, even when a trace ID is not passed by an upstream service.

D

Step Functions

X

✓ State machine successfully created

State machines

Activities

Data flow simulator

Feature spotlight

Local Development

Join our feedback panel 

Step Functions > State machines > OrderProcessorWorkflow

OrderProcessorWorkflow

Details

ARN

arn:aws:states:us-east-
2::stateMachine:OrderProcessorWorkflow

IAM role ARN

arn:aws:iam:::role/severlesspresso-w
4-OrderProcessorRole-YNLAA4QXSKRF 

[Executions](#)

[Logging](#)

[Definition](#)

[Tags](#)

Executions (0)

Activities

Data flow s

Feature spe

Local Deve

Join our fe

Start execution

Start an execution using the latest definition of the state machine. [Learn more](#) 

Name - *optional*

01a44c3e-5281-2e5d-2201-016edfacdd20

Input - *optional*

Enter input values for this execution in JSON format

```
1 <pre>{</pre>
2   "Comment": "Hello, world!"</pre>
3 }</pre>
```

Open in a new browser tab

Step Functions

X

State machines

Activities

Data flow simulator

Feature spotlight

Local Development

Join our feedback panel

Step Functions > State machines > OrderProcessorWorkflow > 01a44c3e-5281-2e5d-2201-016edfacdd20

01a44c3e-5281-2e5d-2201-016edfacdd20

Details

Execution input

Execution output

Definition

Execution Status

Succeeded

Started

Dec 17, 2023

Execution ARN

arn:aws:states:ca-central-1:468083054740:execution:OrderProcessorWorkflow:01a44c3e-5281-2e5d-2201-016edfacdd20

End Time

Dec 17, 2023

Graph inspector



Detail

Select

In Progress Succeeded Failed Cancelled Caught Error

Execution event history

ID	Type	Step	Resource	Elapsed
▶ 1	ExecutionStarted	-	-	0
▶ 2	PassStateEntered	Pass	-	43

3

```

Step Functions X

State machines
Activities
Data flow simulator
Feature spotlight
Local Development
Join our feedback panel ↗

▼ 2 PassStateEntered Pass
1 { "name": "Pass", "input": { "Comment": "Hello, world!" }, "inputDetails": { "truncated": false } }
▼ 3 PassStateExited Pass
1 { "name": "Pass", "output": { "Comment": "Hello, world!" }, "outputDetails": { "truncated": false } }
▶ 4 ExecutionSucceeded Pass

```

A loja está aberta:

- Verificando se a Loja Está Aberta

Neste módulo, você modificará o fluxo de trabalho para verificar se a cafeteria está aberta e tomar as medidas necessárias. Após a conclusão, o fluxo de trabalho será executado caso a loja esteja aberta.

Redefinindo o Fluxo de Trabalho do Step Functions

Instruções Passo a Passo:

1. Acesse o Console do Step Functions:

- No AWS Management Console, selecione Services e depois Step Functions em Application Integration. Verifique se a região está correta.

2. Selecionar a Máquina de Estado:

- No menu à esquerda, selecione State machine e escolha `OrderProcessorWorkflow` na lista. Clique em Edit.

3. Abrir o Workflow Studio:

- Na próxima página, escolha Workflow Studio para abrir o fluxo de trabalho no designer.

4. Excluir o Estado de Passagem:

- Clique no estado de passagem adicionado anteriormente e escolha Delete na barra de ferramentas.

Consultando a Tabela do DynamoDB

Instruções Passo a Passo:

1. Adicionar Ação de Consulta:

- Com a aba Actions selecionada, escolha DynamoDB na categoria Database.
- Arraste a ação DynamoDB GetItem para o estado vazio no designer.

2. Configurar o Estado de Consulta:

- Selecione o estado adicionado. No painel de atributos à direita, na aba Configuration:
 - Para Nome do estado digite `DynamoDB Get Shop Status`.
 - Para Parâmetros de API, cole a seguinte consulta do DynamoDB:

```
```json
{
 "TableName": "serverlesspresso-config-table",
 "Key": {
 "PK": {
 "S": "config"
 }
 }
}
```
```

```

### 3. Modificar a Saída do Estado:

- Selecione a aba Output e marque a caixa Adicionar entrada original à saída usando ResultPath.
- Certifique-se de que Combinar entrada original com resultado esteja selecionado e insira `\$.GetStore` na caixa de texto.

Adicionando Lógica de Ramificação

Instruções Passo a Passo:

#### 1. Adicionar Estado de Escolha:

- Na aba Flow, arraste o estado Choice para baixo do estado DynamoDB GetItem.

## 2. Adicionar Ação de Eventos:

- Na aba Actions, pesquise por EventBridge e arraste a ação PutEvents para a caixa vazia Rule #1 sob o estado de escolha.

## 3. Adicionar Estado de Passagem:

- Arraste o estado Pass para a caixa vazia Padrão abaixo do estado de escolha.

## 4. Definir Lógica de Decisão:

- Clique no estado Choice para abrir os atributos no painel lateral direito. Para Rule #1, clique no ícone de edição.

- Selecione Adicionar condições.

- No painel Condições para a regra nº 1, especifique a regra:

- Para Não, selecione NÃO no menu suspenso.

- Para Variável, insira `\$.GetStore.Item.storeOpen.BOOL`.

- Para o Operador, selecione é igual a.

- Para Valor, selecione Constante booleana e escolha \*\*verdadeiro\*\* como valor.

- Selecione Salvar condições.

## 5. Nomear o Estado:

- Para o nome do estado, adicione `Shop Open?`.

## 6. Verificar a Definição ASL:

- Ative a definição da Amazon States Language (ASL) clicando no botão de alternância Definition acima do designer. Verifique a ASL, que deve parecer com:

```
```json
{
  "Comment": "A description of my state machine",
  "StartAt": "DynamoDB Get Shop Status",
  "States": {
    "DynamoDB Get Shop Status": {
      "Type": "Task",
      "Resource": "arn:aws:states:::dynamodb:getItem",
      "Parameters": {
        "TableName": "serverlesspresso-config-table",
        "Key": {
          "id": "1234567890"
        }
      }
    }
  }
}
```

```
"PK": {  
    "S": "config"  
},  
}  
,  
"ResultPath": "$.GetStore",  
"Next": "Shop Open?"  
},  
"Shop Open?": {  
    "Type": "Choice",  
    "Choices": [  
        {  
            "Not": {  
                "Variable": "$.GetStore.Item.storeOpen.BOOL",  
                "BooleanEquals": true  
            },  
            "Next": "PutEvents"  
        }  
    ],  
    "Default": "Pass"  
},  
"PutEvents": {  
    "Type": "Task",  
    "End": true,  
    "Parameters": {  
        "Entries": [  
            {}  
        ]  
    },  
    "Resource": "arn:aws:states:::aws-sdk:eventbridge:putEvents"  
},
```

```
"Pass": {  
    "Type": "Pass",  
    "End": true  
}  
}  
}  
}  
...  
}
```

- Clique em Aplicar e saia.

7. Salvar o Fluxo de Trabalho:

- Na página Editar OrderProcessorWorkflow, escolha Salvar.
- No pop-up da função IAM, escolha Save anyway.

Testando o Fluxo de Trabalho do Step Functions

Instruções Passo a Passo:

1. Iniciar Execução:

- Na página do novo fluxo de trabalho, escolha Iniciar execução e, em seguida, clique em Iniciar execução no pop-up.

2. Ver Resultados da Execução:

- Após a execução ser concluída, o console mostrará uma página de resultados. O lado esquerdo mostra o fluxo de execução com os estados destacados em verde.
- Escolha o status Shop Open? para ver os detalhes no lado direito.

3. Ver Entrada do Estado de Escolha:

- Selecione a entrada Step no lado direito para ver a carga de entrada para o estado de escolha. Se o atributo `storeOpen` for TRUE, o estado de escolha redirecionará para o estado Pass.

Step Functions

X

State machines

Activities

Data flow simulator

Feature spotlight

Local Development

Join our feedback panel 

Step Functions > State machines > OrderProcessorWorkflow

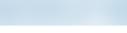
OrderProcessorWorkflow

Details

ARN

arn:aws:states:us-east-
2::stateMachine:OrderProcessorWorkflow

IAM role ARN

arn:aws:iam:::role/severlesspress-
4-OrderProcessorRole-YNLAA4QXSKRF 

[Executions](#)

[Logging](#)

[Definition](#)

Executions (1)

 *Search for executions*

Step Functions

State machines

Activities

Data flow simulator

Feature spotlight

Local Development

Join our feedback panel 

Edit OrderProcessorWorkflow [Info](#)

 Search

Actions

Flow

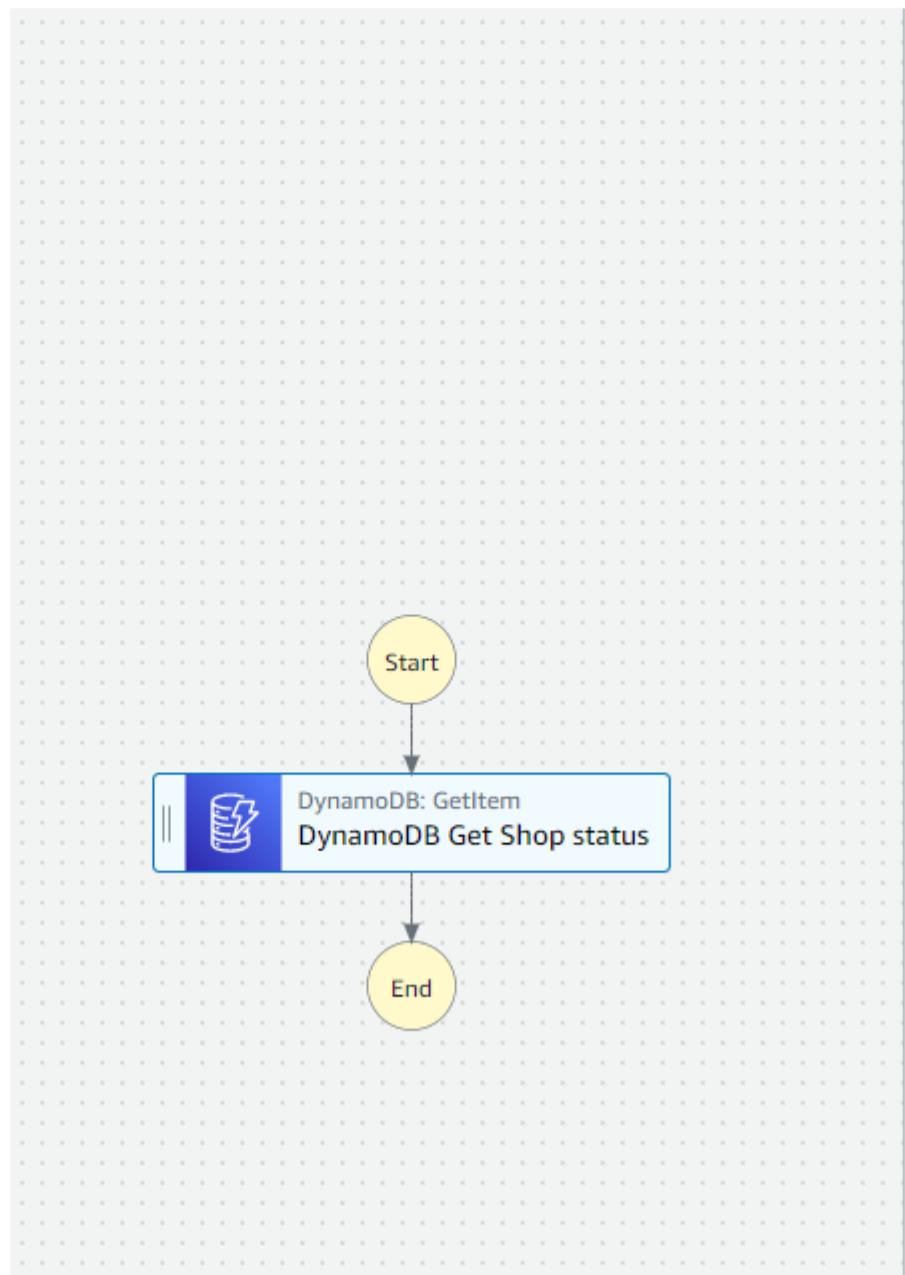
Most Popular

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

COMPUTE



```
{  
  "TableName": "serverlesspresso-config-table",  
  "Key": {  
    "PK": {  
      "S": "config"  
    }  
  }  
}
```



DynamoDB Get Shop Status

Configuration

Input

Output

Error handling

During execution, the Task state calls an API and the response goes into the task result. The result can be combined with filters before it is passed as output to the next state. [Info](#)

Transform result with ResultSelector - *optional* [Info](#)

Use the ResultSelector filter to construct a new JSON object using parts of the task result.

Add original input to output using ResultPath - *optional* [Info](#)

By default, a state sends its task result as output. Use the ResultPath filter to include the original input in the state's output.

Combine original input with result

`$.GetStore`

Must use valid JSONPath syntax.

Filter output with OutputPath - *optional* [Info](#)

Use the OutputPath filter to select a portion of the effective output to pass to the next state.



Edit OrderProcessorWorkflow [Info](#)

Search



Actions

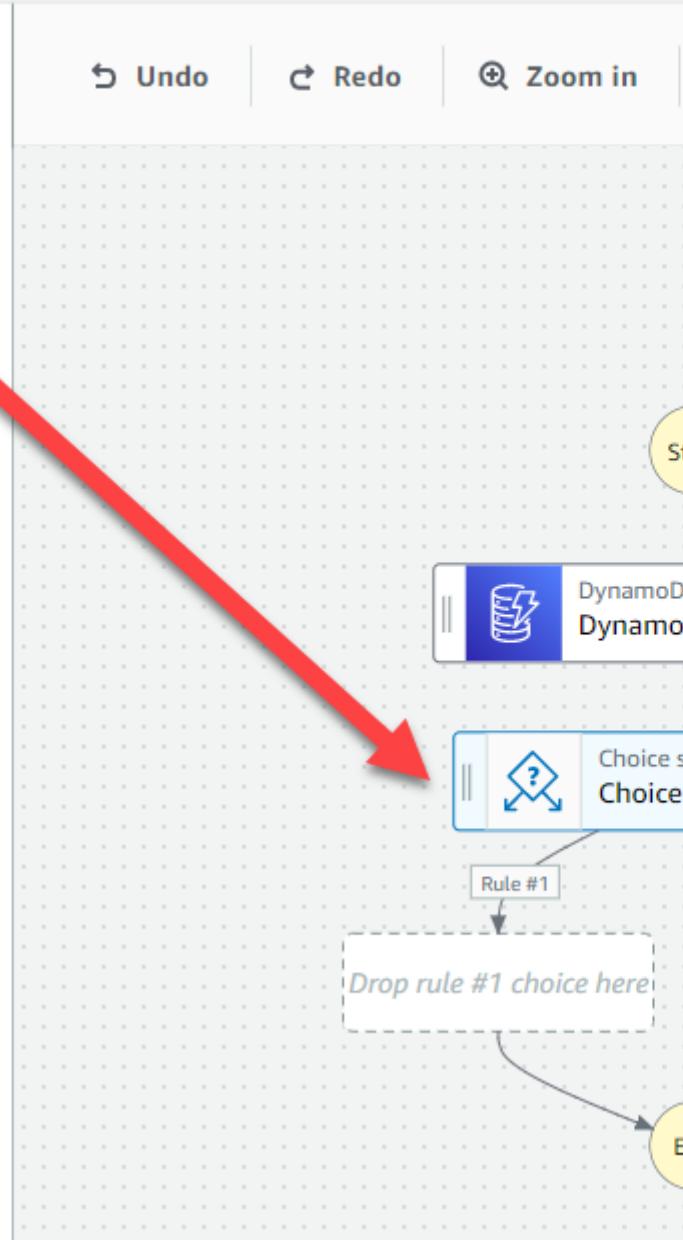
Flow

Undo

Redo

Zoom in

- Choice
Adds if-then-else logic.
- Parallel
Adds parallel branches.
- Map
Adds a for-each loop.
- Pass
Transforms data or acts as placeholder.
- Wait
Delays for a specified time.
- Success
Stops and marks as success.
- Fail
Stops and marks as failure.





Edit OrderProcessorWorkflow [Info](#)

eventbridge X «

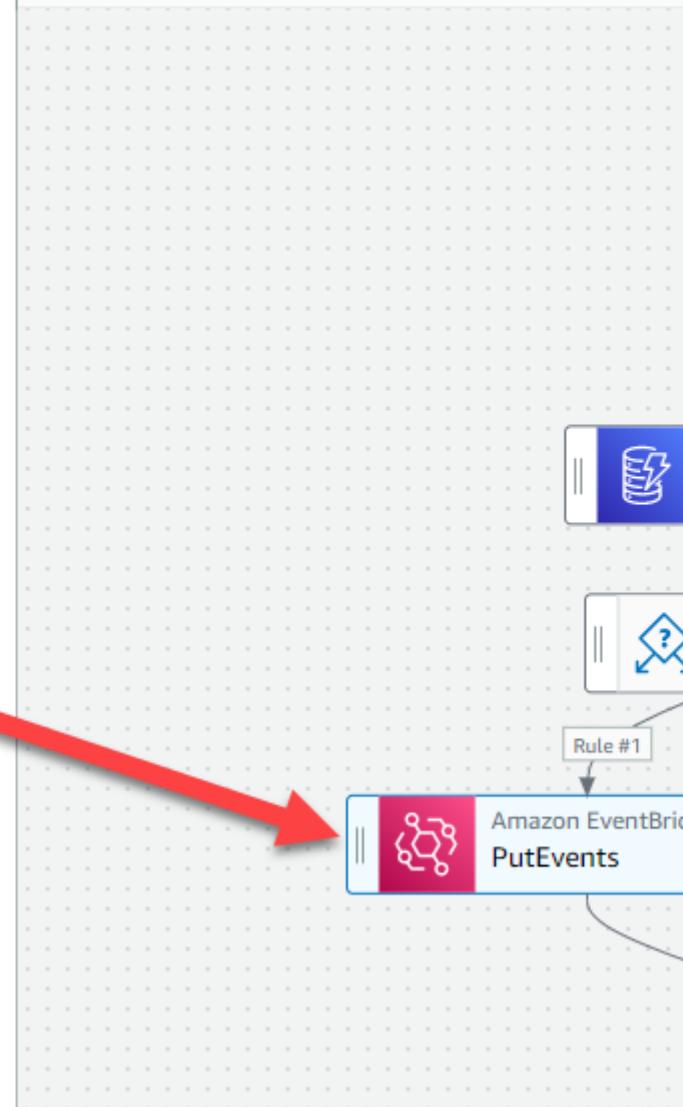
83 matches found

- || EventBridge Amazon EventBridge ListReplays
- || EventBridge Amazon EventBridge ListRuleNamesByTarget
- || EventBridge Amazon EventBridge ListRules
- || EventBridge Amazon EventBridge ListTagsForResource
- || EventBridge Amazon EventBridge ListTargetsByRule
- || EventBridge Amazon EventBridge PutEvents
- || EventBridge Amazon EventBridge PutPartnerEvents
- || EventBridge Amazon EventBridge PutPermission
- || EventBridge Amazon EventBridge PutRule

↶ Undo

↷ Redo

🔍 Zoom in





Edit OrderProcessorWorkflow [Info](#)

Search



Actions

Flow

Undo

Redo

Zoom in



Choice

Adds if-then-else logic.



Parallel

Adds parallel branches.



Map

Adds a for-each loop.



Pass

Transforms data or acts as placeholder.



Wait

Delays for a specified time.



Success

Stops and marks as success.



Fail

Stops and marks as failure.

Undo

Redo

Zoom in



Amazon EventBridge
PutEvents

Choice Rules

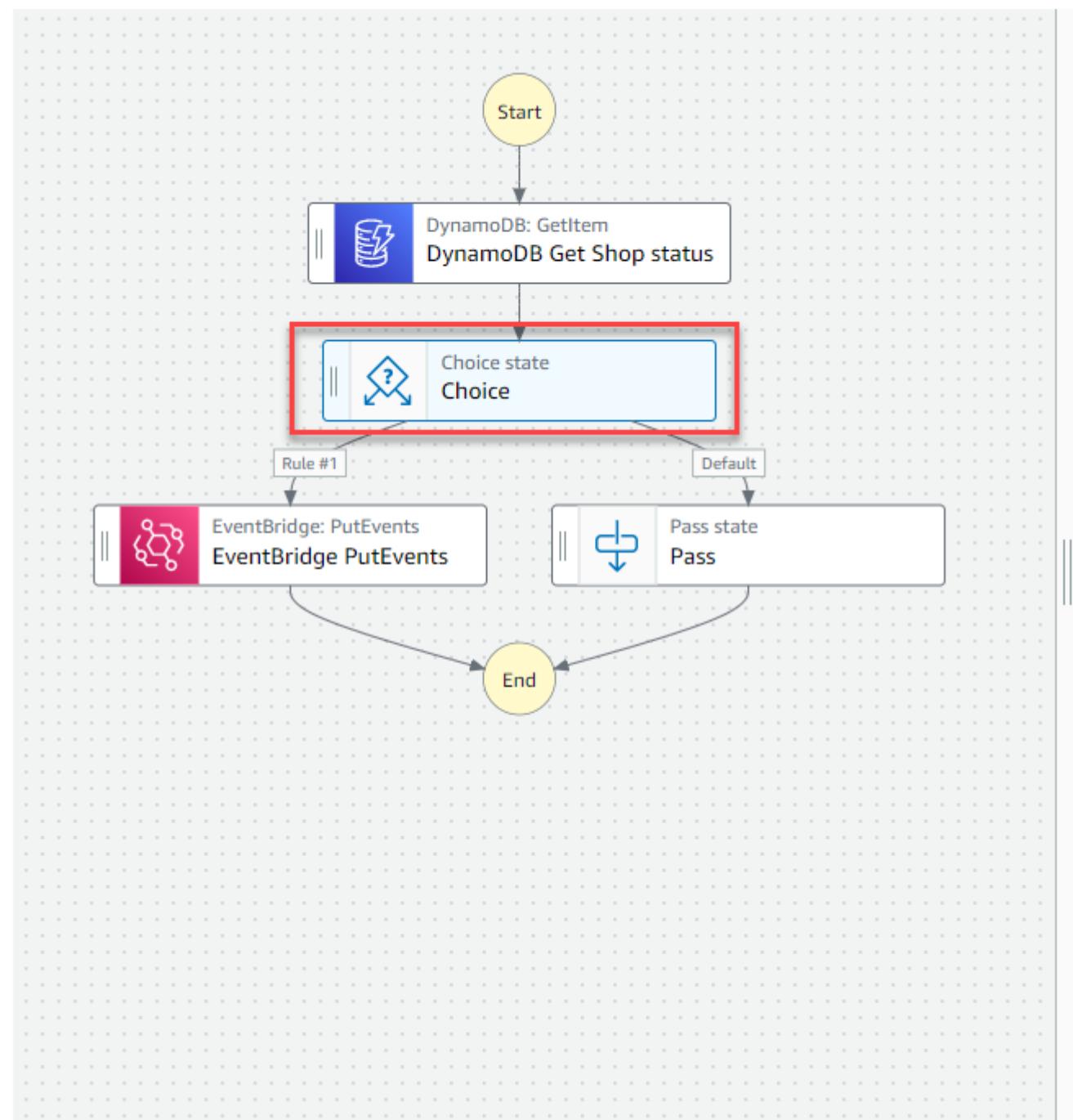
Choice rules let you create if-then-else logic to determine which state the workflow should transition to next.

Rule #1 (Edit)

Default rule (Edit)

Like an else statement, defines the next state when no rule is true.

+ Add new choice rule



Conditions for rule #1

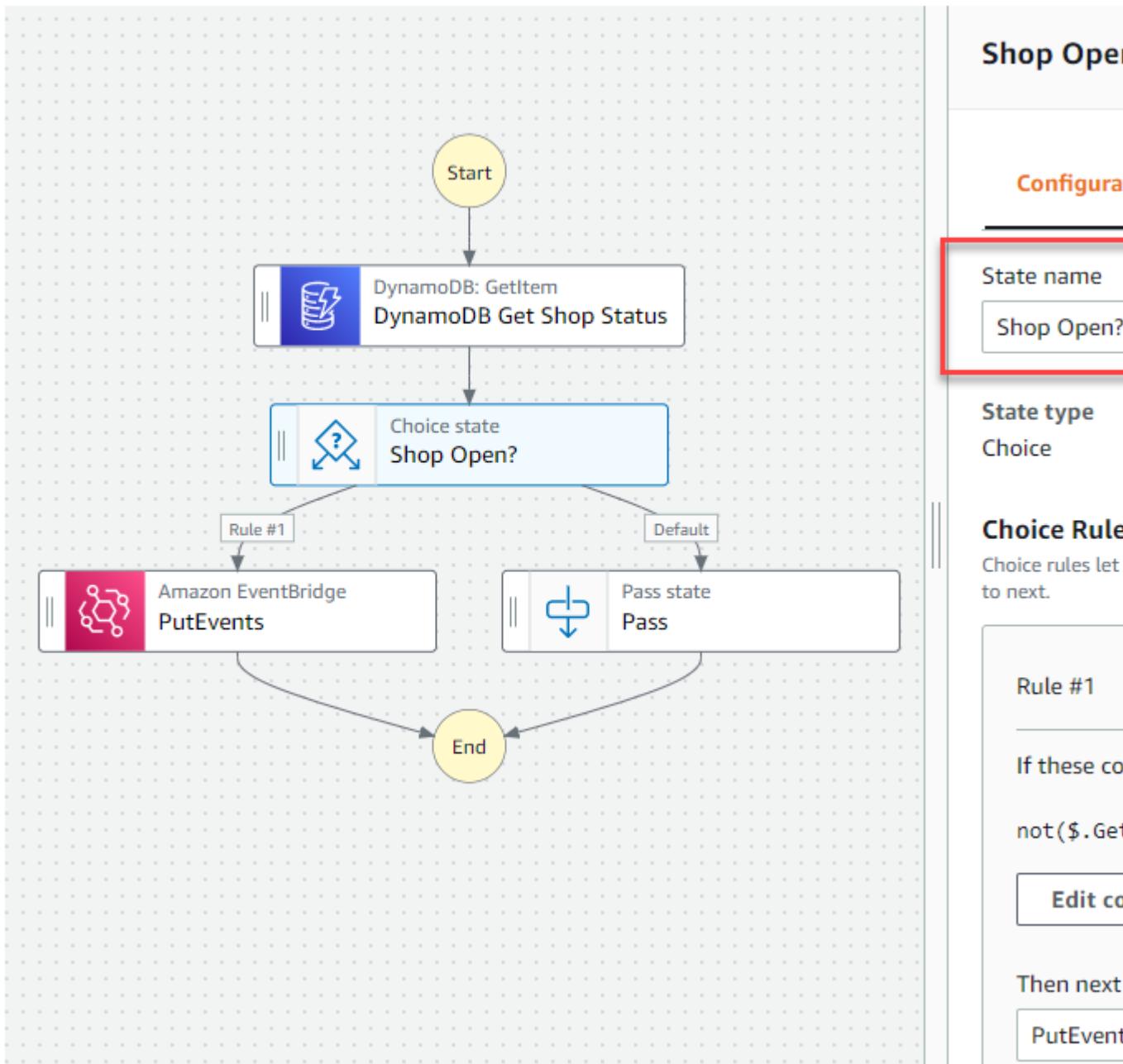
Choice rules contain conditional statements, which are used to evaluate one or more node values (called conditions).

Simple

Evaluates a single conditional statement.

Not	Variable	Operator	Value
NOT ▼	\$.GetStore.Item.store	is equal to	▼ Boolean constant

Must use JsonPath.



Edit OrderProcessorWorkflow [Info](#)

<>

Actions Flow

Choice
Adds if-then-else logic.

Parallel
Adds parallel branches.

Map
Adds a for-each loop.

Pass
Transforms data or acts as placeholder.

Wait
Delays for a specified time.

Success
Stops and marks as success.

Fail
Stops and marks as failure.

Import/Export ▾

Form

Definition

```
graph TD; Start((Start)) --> GetItem[DynamoDB: GetItem<br/>DynamoDB Get Shop Status]; GetItem --> Choice{Choice state<br/>Shop Open?}; Choice -- Rule #1 --> PutEvents[Amazon EventBridge<br/>PutEvents]; PutEvents --> End((End)); Choice --> Pass1[Pass<br/>Pass]; Pass1 --> End;
```

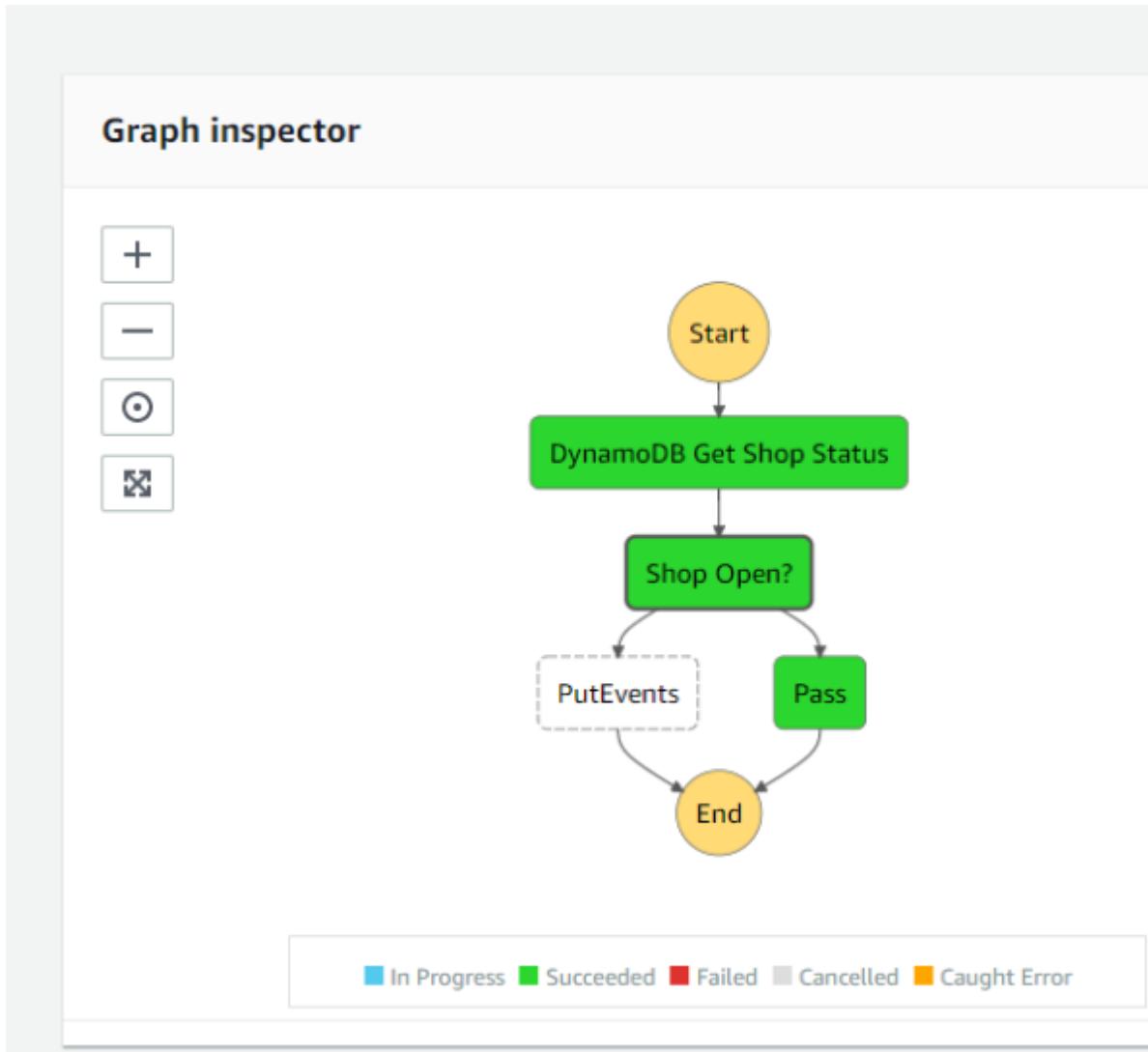
IAM role

X

⚠ The changes to your state machine may affect which resources it needs to access. To ensure your state machine has the right permissions, you might need to edit the current IAM role, create a new one, or select a different role.

Cancel

Save anyway



Há Capacidade:

- Verificando se Há Capacidade na Cafeteria

Neste módulo, você modificará o fluxo de trabalho para verificar se a fila na cafeteria tem capacidade para aceitar outro pedido. Você adicionará um estado que verifica se o número total de fluxos de trabalho abertos é maior do que um máximo configurável.

Listando as Execuções Ativas do Fluxo de Trabalho

Instruções Passo a Passo:

1. Acesse o Console do Step Functions:

- No AWS Management Console, selecione Services e depois Step Functions em Application Integration. Verifique se a região está correta.

2. Selecionar a Máquina de Estado:

- No menu à esquerda, selecione State machine e escolha `OrderProcessorWorkflow` na lista. Copie o valor ARN para um scratchpad - você precisará desse valor mais tarde. Clique em Edit.

3. Abrir o Workflow Studio:

- Na próxima página, escolha Workflow Studio para abrir o fluxo de trabalho no designer.

4. Adicionar Ação de Listagem de Execuções:

- Com a aba Actions selecionada, digite `listexecutions` na barra de pesquisa. Arraste a ação AWS Step Functions ListExecutions da lista para entre os estados Shop Open? e Pass no designer.

5. Configurar o Estado ListExecutions:

- Selecione o estado adicionado. No painel de atributos à direita, na aba Configuration:

- Para Nome do estado, digite `ListExecutions`.

- Para Parâmetros de API, cole o seguinte JSON, substituindo `YOUR_STATE_MACHINE_ARN` pelo ARN que você copiou anteriormente:

```
```json
{
 "StateMachineArn": "YOUR_STATE_MACHINE_ARN",
 "MaxResults": 100,
 "StatusFilter": "RUNNING"
}
```
``
```

6. Modificar a Saída do Estado:

- Selecione a aba Output e marque a caixa Adicionar entrada original à saída usando ResultPath.

- Certifique-se de que Combinar entrada original com resultado esteja selecionado e insira `\$.isCapacityAvailable` na caixa de texto.

Adicionando Lógica de Ramificação

Instruções Passo a Passo:

1. Adicionar Estado de Escolha:

- Na guia Flow, arraste o estado Choice para baixo do estado ListExecutions.

2. Definir a Lógica de Decisão:

- Clique no estado Choice para abrir seus atributos no painel lateral direito. Para a Regra nº 1, clique no ícone de edição.

- Escolha Adicionar condições.

- No painel Condições para a regra nº 1, especifique a regra:

- Para Não, deixe em branco no menu suspenso.

- Para Variável, insira `\$.isCapacityAvailable.Executions[20]`.
- Para Operador, selecione está presente.
- Selecione Salvar condições.

3. Definir o Próximo Estado:

- Para Então, defina o próximo estado como PutEvents.

4. Nomear o Estado:

- Para o nome do estado, digite `Is capacity available?`.

5. Verificar a Definição ASL:

- Ative a definição da Amazon States Language (ASL) clicando no botão de alternância Definition acima do designer. Verifique a ASL, que deve parecer com:

```
```json
{
 "Comment": "A description of my state machine",
 "StartAt": "DynamoDB Get Shop Status",
 "States": {
 "DynamoDB Get Shop Status": {
 "Type": "Task",
 "Resource": "arn:aws:states:::dynamodb:getItem",
 "Parameters": {
 "TableName": "serverlesspresso-config-table",
 "Key": {
 "PK": {
 "S": "config"
 }
 }
 }
 },
 "ResultPath": "$.GetStore",
 "Next": "Shop Open?"
 },
 "Shop Open?": {
 "Type": "Choice",
 }
}
```

```
"Choices": [
 {
 "Not": {
 "Variable": "$.GetStore.Item.storeOpen.BOOL",
 "BooleanEquals": true
 },
 "Next": "PutEvents"
 }
],
"Default": "ListExecutions"
},
"ListExecutions": {
 "Type": "Task",
 "Next": "Is capacity available?",
 "Parameters": {
 "StateMachineArn": "YOUR_STATE_MACHINE_ARN",
 "MaxResults": 100,
 "StatusFilter": "RUNNING"
 },
 "Resource": "arn:aws:states:::aws-sdk:sfn:listExecutions",
 "ResultPath": "$.isCapacityAvailable"
},
"Is capacity available?": {
 "Type": "Choice",
 "Choices": [
 {
 "Variable": "$.isCapacityAvailable.Executions[20]",
 "IsPresent": true,
 "Next": "PutEvents"
 }
],
}
```

```

 "Default": "Pass"
},
"PutEvents": {
 "Type": "Task",
 "End": true,
 "Parameters": {
 "Entries": [
 {}
]
 },
 "Resource": "arn:aws:states:::aws-sdk:eventbridge:putEvents"
},
"Pass": {
 "Type": "Pass",
 "End": true
}
}
}
```

```

- Clique em Aplicar e saia.

6. Salvar o Fluxo de Trabalho:

- Na página Editar OrderProcessorWorkflow, escolha *Salvar.
- No pop-up da função IAM, escolha Save anyway.

Testando o Fluxo de Trabalho do Step Functions

Instruções Passo a Passo:

1. Iniciar Execução:

- Na página do novo fluxo de trabalho, escolha Iniciar execução e, em seguida, clique em Iniciar execução no pop-up.

2. Ver Resultados da Execução:

- Após a execução ser concluída, o console mostrará uma página de resultados. O lado esquerdo mostra o fluxo de execução com os estados destacados em verde.

- Escolha o status Is capacity available? para ver os detalhes no lado direito.

3. Ver Saída do Estado de Escolha:

- Selecione a saída Step no lado direito para ver o caminho de saída para o estado de escolha. Se o array Executions no atributo `isCapacityAvailable` mostrar um item, isso significa que há uma execução ativa e, portanto, capacidade disponível, fazendo com que o fluxo de trabalho vá para o estado Pass.

Com isso, você configurou o fluxo de trabalho para verificar a capacidade disponível na cafeteria, adicionando a lógica necessária para gerenciar os pedidos conforme a capacidade.

The screenshot shows the AWS Step Functions console interface. On the left, there's a sidebar with links: 'Step Functions' (selected), 'State machines' (highlighted in orange), 'Activities', 'Data flow simulator', 'Feature spotlight', 'Local Development', and 'Join our feedback panel'. The main area is titled 'OrderProcessorWorkflow' under 'Details'. It shows the ARN: 'arn:aws:states:ca-central-1:...:stateMachine:OrderProcessorWorkflow' and the IAM role ARN: 'arn:aws:iam::...:role/service-role/StepFunctions-OrderProcessorWorkflow-role-689a97aa'. The ARN field is highlighted with a red box.

Edit OrderProcessorWorkflow [Info](#)

X <<

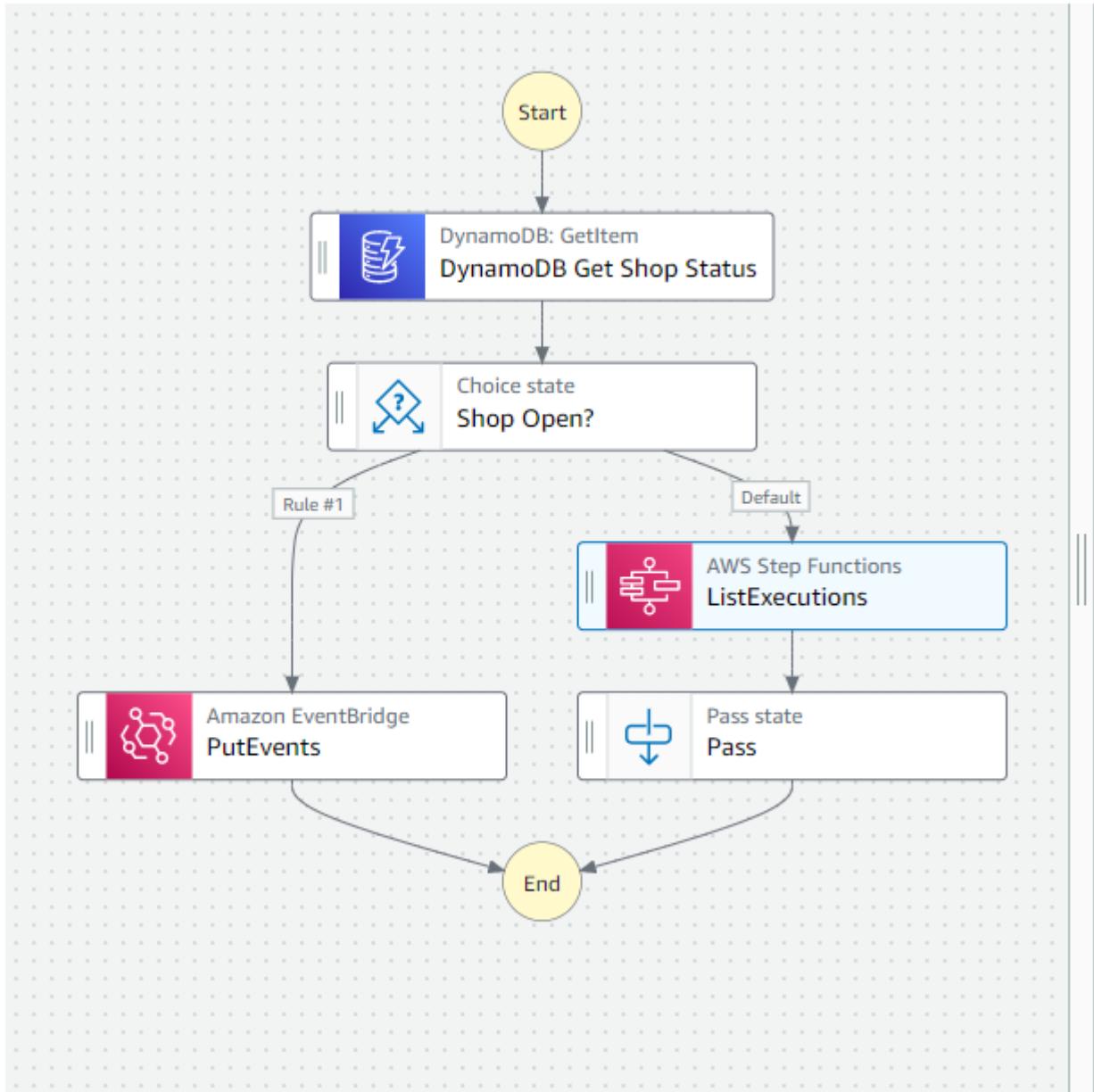
31 matches found

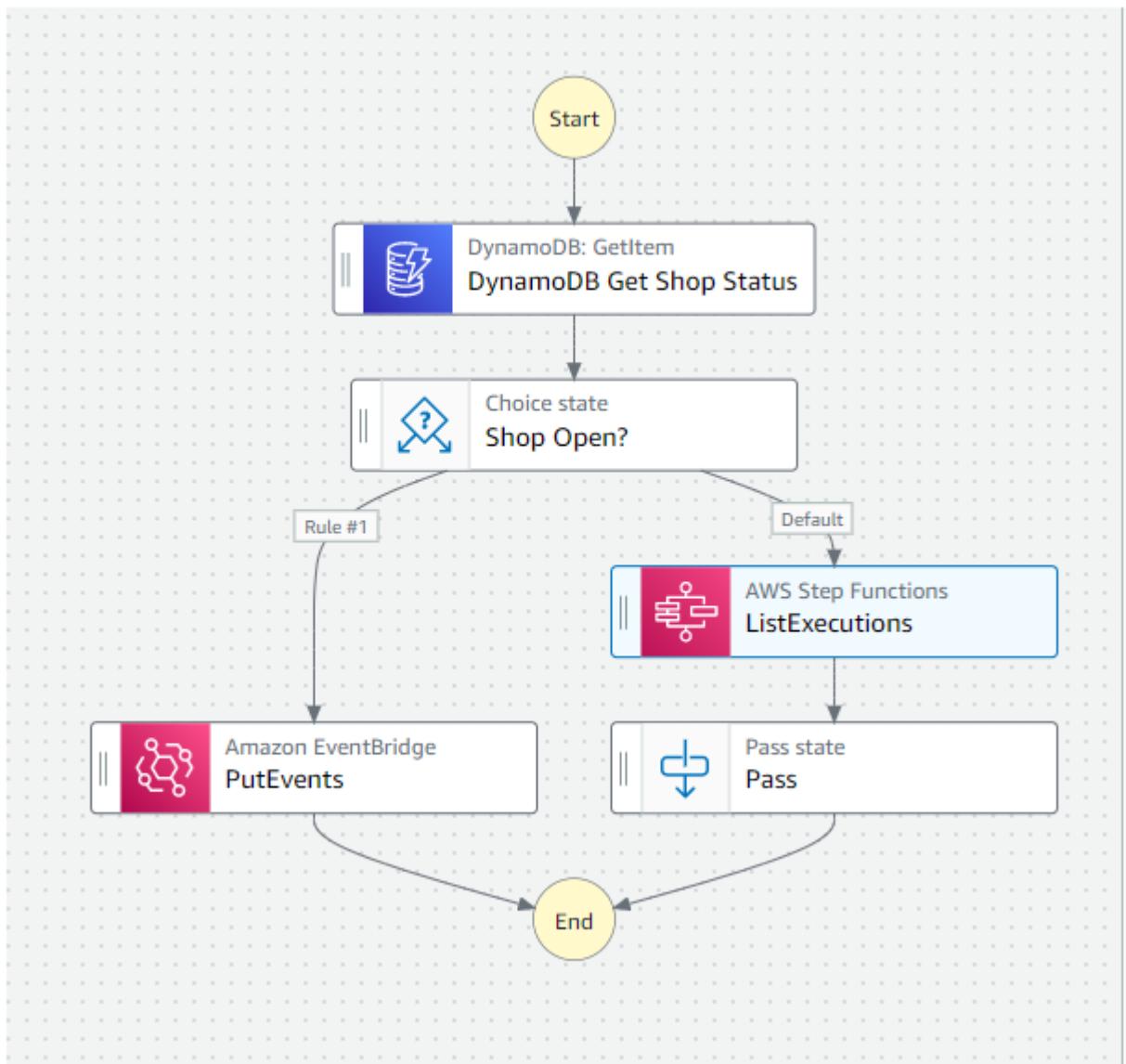
- AWS Step Functions
ListExecutions
- AWS Snow Device Manag...
ListExecutions
- DataSync
ListTaskExecutions
- CodePipeline
ListPipelineExecutions
- Amazon EMR
ListNotebookExecuti...
- SageMaker
ListPipelineExecutions
- Amazon Athena
ListQueryExecutions
- CodePipeline
ListActionExecutions

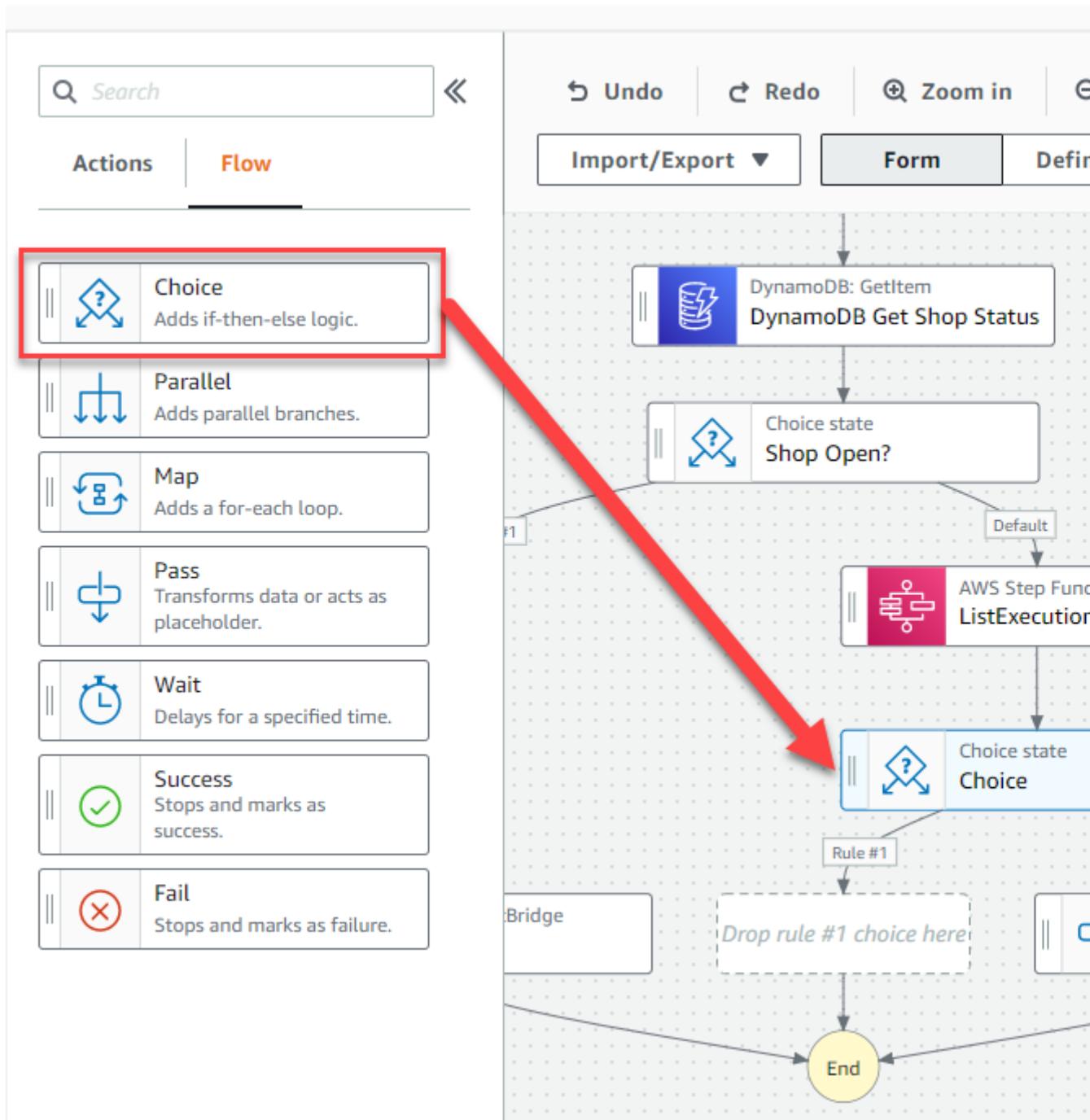
UndoRedoZoom inZoom out

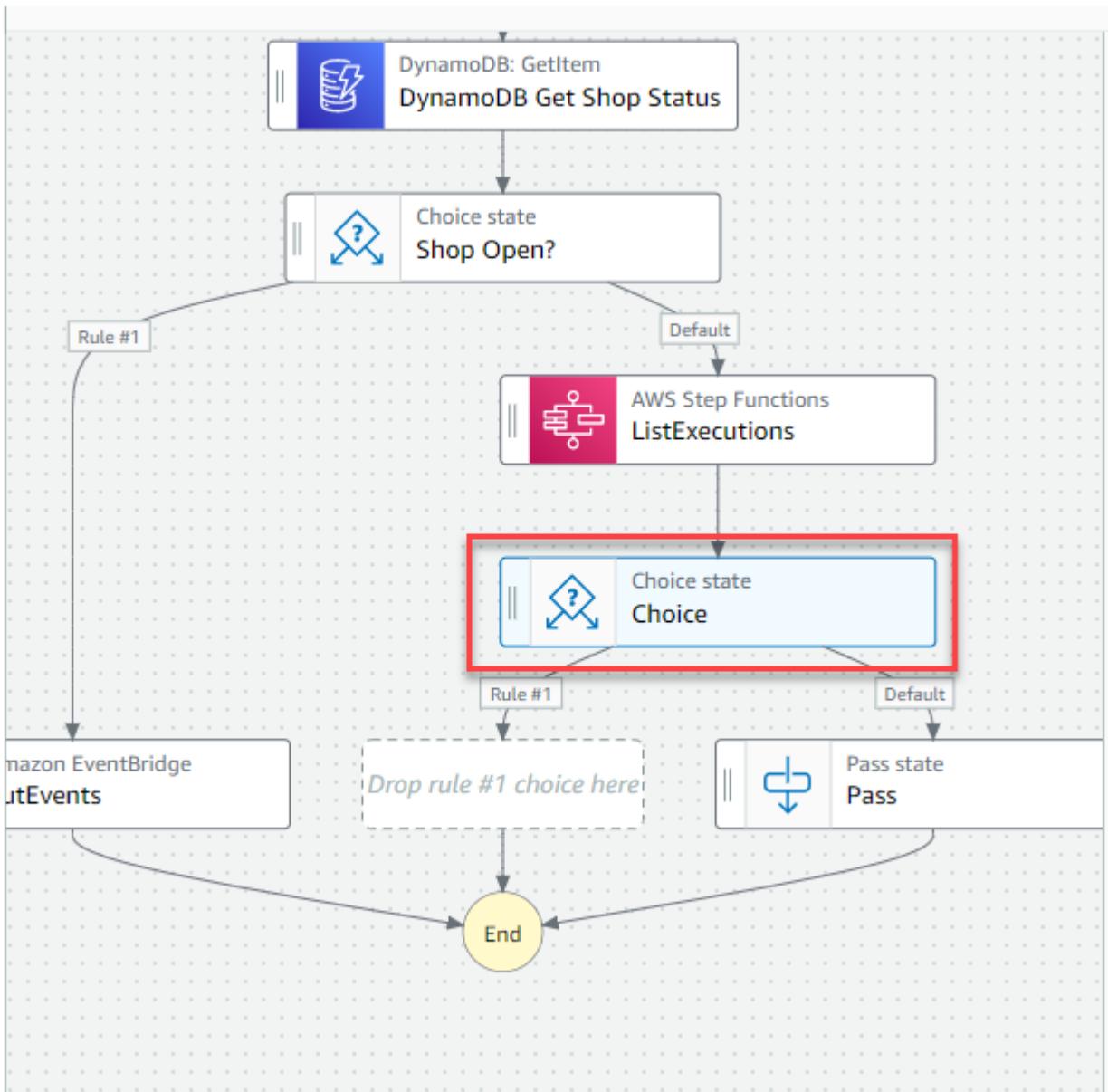
Import/ExportFormDefinition

```
graph TD; Rule1[Rule #1] --> PutEvents[Amazon EventBridge PutEvents]; PutEvents --> End((End))
```









Conditions for rule #1

Choice rules contain conditional statements, which are used to evaluate one or more node values (callouts).

Simple

Evaluates a single conditional statement.

Not

Variable

Operator

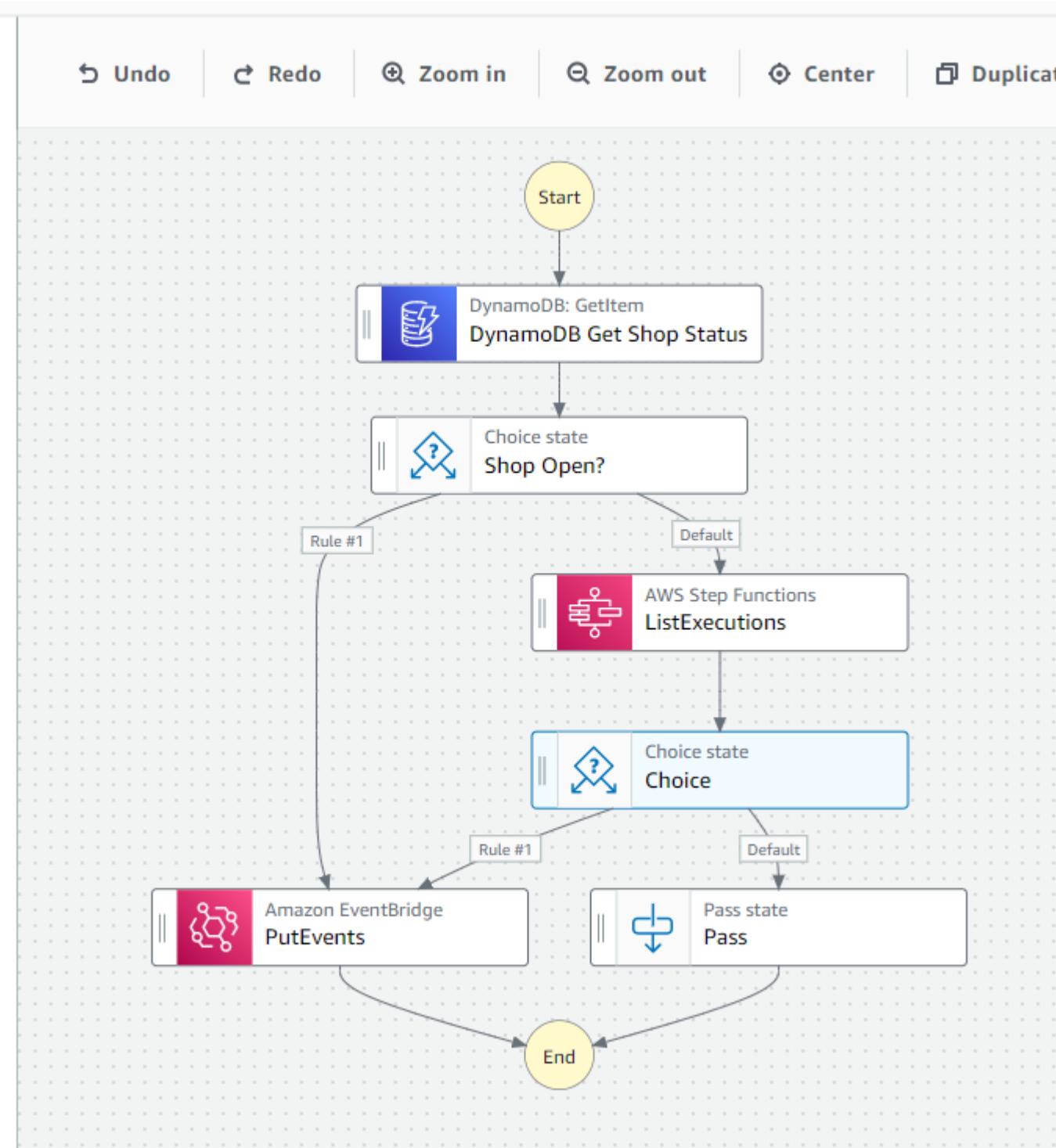
Value

\$.isCapacityAvailable

is present

No value required

Must use JsonPath.



Is capacity available?

Configuration Input Output

State name

Is capacity available?

State type

Choice

Choice Rules

Choice rules let you create if-then-else logic to determine which state the workflow should transition to next.

Edit OrderProcessorWorkflow

Definition

Define your workflow using [Amazon States Language](#). Test your data flow with the new [Data Flow Simulator](#).

[Generate code snippet](#)

Format JSON

```
1 *
2     "Comment": "A description of my state machine",
3     "StartAt": "DynamoDB Get Shop Status",
4     "States": {
5         "DynamoDB Get Shop Status": {
6             "Type": "Task",
7             "Resource": "arn:aws:states:::dynamodb:getItem",
8             "Parameters": {
9                 "TableName": "severlesspresso-workshop-core-2-ConfigTable-PLYSSLC9MLXY",
10                "Key": {
11                    "PK": {
12                        "S": "config"
13                    }
14                }
15            },
16            "ResultPath": "$.GetStore",
17            "Next": "Shop Open?"
18        },
19        "Shop Open?": {
20            "Type": "Choice",
21            "Choices": [
22                {
23                    "Not": {
```

IAM role

X

 The changes to your state machine may affect which resources it needs to access. To ensure your state machine has the right permissions, you might need to edit the current IAM role, create a new one, or select a different role.

Cancel

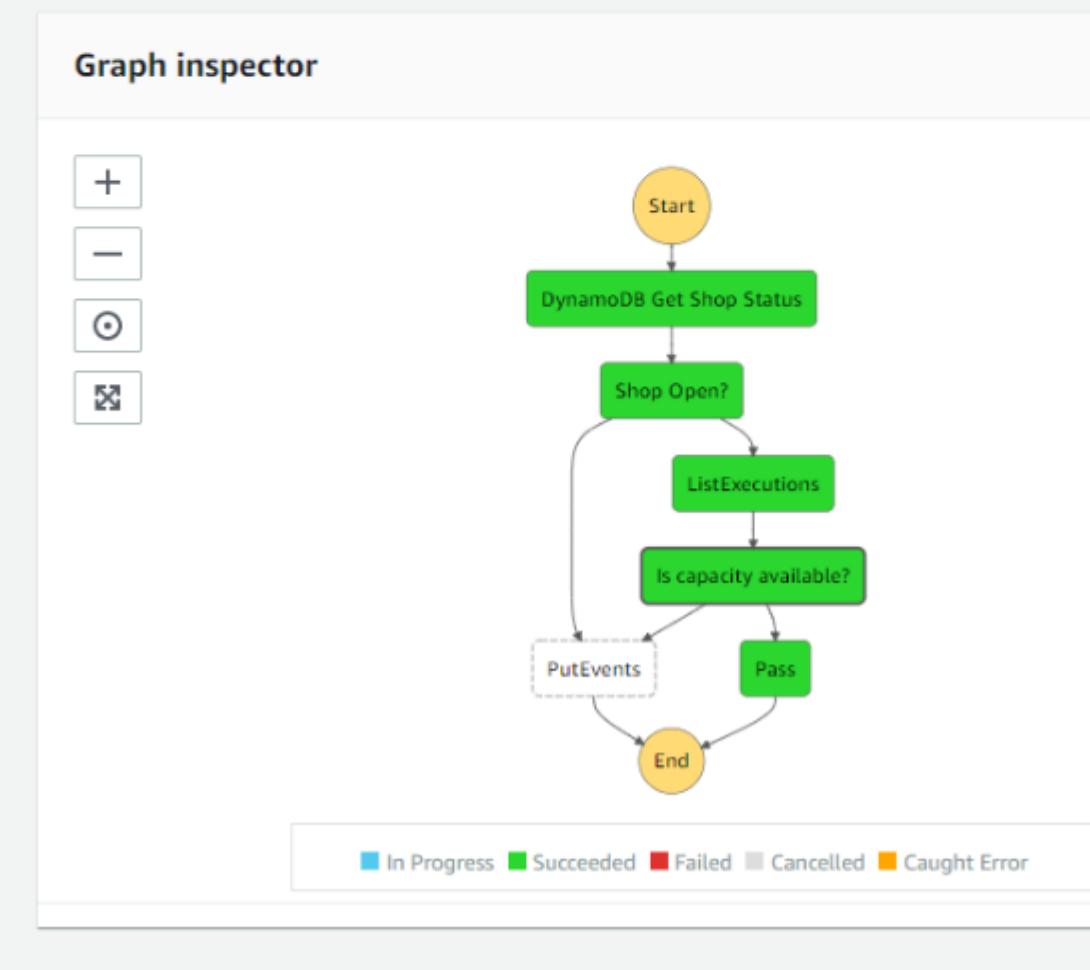
Save anyway

20955cb9-634a-0e47-61ba-24ca1c9aefe0

| Details | Execution input | Execution output | Definition |
|---------|-----------------|------------------|------------|
|---------|-----------------|------------------|------------|

Execution Status
 Succeeded

Execution ARN
arn:aws:states:ca-central-1:XXXXXXXXXX:execution:OrderProcessorWorkflow:20955cb9-634a-0e47-61ba-24ca1c9aefe0



Adicionando um Número de Pedido ao Fluxo de Trabalho

Neste módulo, você modificará o fluxo de trabalho para adicionar um número de pedido a cada pedido de café. Para isso, será adicionado um estado que incrementa um contador em uma tabela do DynamoDB e usa esse valor como o número do pedido.

Atualizando um Contador Atômico em uma Tabela do DynamoDB

Instruções Passo a Passo:

1. Acesse o Console do Step Functions:

- No AWS Management Console, selecione Services e, em seguida, Step Functions em Application Integration. Verifique se a região está correta.

2. Selecionar a Máquina de Estado:

- No menu à esquerda, selecione State machine e escolha `OrderProcessorWorkflow` na lista. Clique em Edit.

3. Abrir o Workflow Studio:

- Na próxima página, escolha Workflow Studio para abrir o fluxo de trabalho no designer.

4. Adicionar Ação de Atualização de Item:

- Com a aba Actions selecionada, digite `updateitem` na barra de pesquisa. Arraste a ação Amazon DynamoDB UpdateItem da lista para entre os estados Is capacity available? e Pass no designer.

5. Configurar o Estado Generate Order Number:

- Selecione o estado adicionado. No painel de atributos à direita, na aba Configuration:

- Para Nome do estado, digite `Generate Order Number`.

- Para Parâmetros de API, cole a seguinte consulta do DynamoDB:

```
```json
{
 "TableName": "serverlesspresso-counting-table",
 "Key": {
 "PK": {
 "S": "orderId"
 }
 },
 "UpdateExpression": "set IDvalue = IDvalue + :val",
 "ExpressionAttributeValues": {
 ":val": {
 "N": "1"
 }
 }
}
```

```

 "N": "1"
}
},
"ReturnValues": "UPDATED_NEW"
}
...

```

#### 6. Modificar a Saída do Estado:

- Selecione a aba Output e marque a caixa Transformar resultado com ResultSelector.
  - Na caixa de texto de valor, insira:
- ```

```json
{
 "orderNumber.$": "$.Attributes.IDvalue.N"
}
...

```
- Marque a caixa Adicionar entrada original à saída usando ResultPath.
  - Certifique-se de que o menu suspenso esteja definido como \*\*Combinar entrada original com resultado\*\* e insira `\$.Order.Payload` na caixa de texto.

#### 7. Verificar a Definição da Amazon States Language (ASL):

- Clique no botão de alternância Definition acima do designer. A ASL deve aparecer como:

```

```json
{
  "Comment": "A description of my state machine",
  "StartAt": "DynamoDB Get Shop Status",
  "States": {
    "DynamoDB Get Shop Status": {
      "Type": "Task",
      "Resource": "arn:aws:states:::dynamodb:getItem",
      "Parameters": {
        "TableName": "serverlesspresso-config-table",
        "Key": {
          "PK": {
            ...
```

```

```
 "S": "config"
 }
},
},
"ResultPath": "$.GetStore",
"Next": "Shop Open?"
},
"Shop Open?": {
 "Type": "Choice",
 "Choices": [
 {
 "Not": {
 "Variable": "$.GetStore.Item.storeOpen.BOOL",
 "BooleanEquals": true
 },
 "Next": "EventBridge PutEvents"
 }
],
 "Default": "ListExecutions"
},
"ListExecutions": {
 "Type": "Task",
 "Next": "Is capacity available?",
 "Parameters": {
 "StateMachineArn": "YOUR_STATE_MACHINE_ARN",
 "MaxResults": 100,
 "StatusFilter": "RUNNING"
 },
 "Resource": "arn:aws:states:::aws-sdk:sfn:listExecutions",
 "ResultPath": "$.isCapacityAvailable"
},
```

```
"Is capacity available?": {
 "Type": "Choice",
 "Choices": [
 {
 "Variable": "$.isCapacityAvailable[20]",
 "IsPresent": true,
 "Next": "EventBridge PutEvents"
 }
],
 "Default": "Generate Order Number"
},
"Generate Order Number": {
 "Type": "Task",
 "Resource": "arn:aws:states:::dynamodb:updateItem",
 "Parameters": {
 "TableName": "serverlesspresso-counting-table",
 "Key": {
 "PK": {
 "S": "orderID"
 }
 },
 "UpdateExpression": "set IDvalue = IDvalue + :val",
 "ExpressionAttributeValues": {
 ":val": {
 "N": "1"
 }
 },
 "ReturnValues": "UPDATED_NEW"
 },
 "Next": "Pass",
 "ResultPath": "$.Order.Payload",
```

```

"ResultSelector": {

 "orderNumber.$": "$.Attributes.IDvalue.N"

}

},

"EventBridge PutEvents": {

 "Type": "Task",

 "Resource": "arn:aws:states:::events:putEvents.waitForTaskToken",

 "Parameters": {

 "Entries": [

 {

 "Detail": {

 "Message": "Hello from Step Functions!",

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

 },
 {
 "DetailType": "MyDetailType",
 "EventBusName": "MyEventBusName",
 "Source": "MySource"
 }
]
 },
 "End": true
},

"Pass": {

 "Type": "Pass",

 "End": true
}
}
}
```

```

- Selecione Aplicar e saia.

8. Salvar o Fluxo de Trabalho:

- Na página Editar OrderProcessorWorkflow, escolha Salvar.
- No pop-up da função IAM, escolha Salvar mesmo assim.

Testando o Fluxo de Trabalho do Step Functions

Instruções Passo a Passo:

1. Iniciar Execução:

- Na página que mostra o novo fluxo de trabalho, escolha Iniciar execução e, em seguida, clique em Iniciar execução no pop-up.

2. Ver Resultados da Execução:

- Após a execução ser concluída, o console mostrará uma página de resultados. O lado esquerdo mostra o fluxo de execução com os estados destacados em verde.
- Escolha o estado Generate Order Number para ver os detalhes no lado direito.

3. Ver Saída do Estado:

- Selecione a saída Step no lado direito para ver o caminho de saída. A saída JSON deve mostrar um atributo Order com um Payload contendo um orderNumber de 1.

4. Repetir Execução:

- Selecione Iniciar execução novamente e repita as etapas 1 e 2. O orderNumber agora será 2. Cada vez que você inicia outra execução, o número do pedido é incrementado.

Para modificar o fluxo de trabalho do `OrderProcessorWorkflow` e adicionar um estado de espera para que o cliente envie o pedido de café e, em seguida, para que o barista faça o pedido, siga as instruções abaixo:

1. Adicionando um estado de retorno de chamada para pedidos de clientes

Instruções passo a passo:

1. Acesse o console do Step Functions:

- No AWS Management Console, selecione Services e, em seguida, escolha Step Functions em Application Integration. Certifique-se de que a região está correta.

2. Edite o fluxo de trabalho:

- No menu à esquerda, selecione State machine e escolha OrderProcessorWorkflow. Clique em Edit.

3. Acesse o Workflow Studio:

- Na próxima página, escolha Workflow Studio para abrir o fluxo de trabalho no designer.

4. Adicionar estado PutEvents:

- Com a aba Actions selecionada à esquerda, digite `putevents` na barra de pesquisa. Arraste a ação Amazon EventBridge PutEvents para entre os estados Is capacity available? e Generate Order Number no designer.

5. Configurar o estado:

- Com o estado selecionado, no painel de atributos à direita:

- Para Nome do estado, digite `Emit - Workflow Started TT`.

- Marque Aguardar retorno de chamada.

- Para Parâmetros de API, cole o seguinte JSON:

```
```json
{
 "Entries": [
 {
 "Detail": {
 "Message": "The workflow waits for your order to be submitted. It emits an event with a unique 'task token'. The token is stored in an Amazon DynamoDB table, along with your order ID.",
 "TaskToken.$": "$$.Task.Token",
 "orderId.$": "$.detail.orderId",
 "userId.$": "$.detail.userId"
 },
 "DetailType": "OrderProcessor.WorkflowStarted",
 "EventBusName": "Serverlesspresso",
 "Source": "awsserverlessda.serverlesspresso"
 }
]
}
```

```

6. Configurar a saída:

- Selecione a aba Output e marque a caixa Adicionar entrada original à saída usando ResultPath.

- No menu suspenso, selecione Descartar resultado e manter a entrada original.

7. Adicionar manipulação de erros:

- Selecione a aba Error handling.

- Em Catch errors, selecione Add new catcher.

- Para Comentário, insira `Customer timed out`.

- Para Erros, selecione `States.Timeout`.

- Para Estado de fallback, selecione Adicionar novo estado.
- Para Heartbeat, insira `900 segundos` (15 minutos).

8. Adicionar estado Pass:

- Com a aba Flow selecionada à esquerda, arraste a ação Pass para o espaço reservado no designer.
- Com o estado selecionado:
 - Para Nome do estado, digite `Customer timedout`.
 - Na guia Saída, insira `Customer timedout` (incluindo as aspas).
 - Marque Adicionar entrada original à saída usando ResultPath e insira `\$.cause`.

2. Adicionando um estado de retorno de chamada para os baristas

Instruções passo a passo:

1. Adicionar estado PutEvents para baristas:

- Com a aba Actions selecionada à esquerda, digite `putevents` na barra de pesquisa. Arraste a ação Amazon EventBridge PutEvents para entre os estados Generate Order Number e Pass no designer.

2. Configurar o estado:

- Com o estado selecionado:
 - Para Nome do estado, insira `Emitir - Aguardando conclusão TT`.
 - Marque Aguardar retorno de chamada.
 - Para Parâmetros de API, cole o seguinte JSON:

```
```json
{
 "Entries": [
 {
 "Detail": {
 "Message": "You pressed 'submit order'. The workflow resumes using the stored 'task token', it generates your order number. It then pauses again, emitting an event with a new 'task token'.",
 "TaskToken.$": "$$.Task.Token",
 "orderId.$": ".$.detail.orderId",
 "orderNumber.$": ".$.Order.Payload.orderNumber",
 "userId.$": ".$.detail.userId"
 }
 }
]
}
```

```

 "DetailType": "OrderProcessor.WaitingCompletion",
 "EventBusName": "Serverlesspresso",
 "Source": "awsserverlessda.serverlesspresso"
 }
]
}
```

```

3. Configurar a saída:

- Na aba Output, marque a caixa Adicionar entrada original à saída usando ResultPath.
- No menu suspenso, selecione Combinar entrada original com resultado** e insira `\$.order`.

4. Adicionar manipulação de erros:

- Na aba Error handling, adicione um novo catcher:
 - Para Comentário, insira `Barista timed out`.
 - Para Erros, selecione `States.Timeout`.
 - Para Estado de fallback, selecione Adicionar novo estado.
 - Para Heartbeat, insira `900 segundos` (15 minutos).

5. Adicionar estado Pass:

- Arraste a ação Pass para o espaço reservado no designer.
- Para Nome do estado, digite `Barista timedout`.
- Na guia Saída**, insira `'"Barista timedout"'` (incluindo as aspas).
- Marque Adicionar entrada original à saída usando ResultPath e insira `\$.cause`.

3. Salvar e testar

- Salvar as alterações:
 - Selecione Aplicar e sair. Na página de edição, clique em Salvar e confirme qualquer pop-up do IAM.Testando o fluxo de trabalho

1. Iniciar execução:

- Na página que mostra o novo fluxo de trabalho, clique em Iniciar execução.

- No pop-up, insira a seguinte carga JSON:

```
```json
```

```
{
```

```
 "detail": {
```

```
 "orderId": "1",
 "userId": "testuser"
}
}

```

```

2. Monitorar a execução:

- O console mostrará o status de execução. O estado azul indicará que o fluxo está suspenso, aguardando um retorno de chamada.

3. Capturar o TaskToken:

- No painel Histórico de eventos de execução, abra o evento TaskScheduled para `Emit - Workflow Started TT` e copie o valor `TaskToken`.

4. Retomar a execução:

- Use o comando abaixo no AWS CloudShell, substituindo `YOUR_TASK_TOKEN` pelo valor do token:

```
```bash
aws stepfunctions send-task-success --task-output '{"orderId":1}' --task-token
YOUR_TASK_TOKEN
```

```

5. Repetir para o barista:

- Novamente, capture o `TaskToken` do evento TaskScheduled para `Emit - Aguardando conclusão TT` e execute o mesmo comando `send-task-success` para retomar a execução.

Para implementar a funcionalidade de emissão de eventos no fluxo de trabalho do Step Functions, você seguirá as instruções para adicionar estados que emitem eventos relacionados a tempos limite, pedidos concluídos e verificações de disponibilidade da loja. Aqui estão as etapas detalhadas para cada parte do processo:

1. Emitindo um evento de tempo limite

- Navegue até o console do Step Functions:

- No AWS Management Console, acesse Services e selecione Step Functions em Application Integration.

- Certifique-se de que a região correta esteja selecionada.

- Edite o fluxo de trabalho:

- No menu à esquerda, selecione State machine e escolha OrderProcessorWorkflow. Clique em Edit.

- Abra o Workflow Studio:

- Selecione Workflow Studio para abrir o fluxo de trabalho no designer.

- Adicione um estado PutEvents:

- Com a aba Actions selecionada, pesquise por `putevents` e arraste a ação Amazon EventBridge PutEvents para entre os estados Customer timeout e End no designer.

- Configure o estado:

- Com o estado selecionado, no painel de atributos à direita:

- Nome do estado: `Emit - error timeout`.

- Desmarque Aguardar retorno de chamada.

- Em Parâmetros de API, cole o seguinte JSON:

```
```json
```

```
{
```

```
 "Entries": [
```

```
 {
```

```
 "Detail": {
```

"Message": "The order timed out. Step Functions waits a set amount of time (5 minutes for a customer, 15 minutes for a barista), no action was taken and so the order is ended.",

"userId.\$": "\$.detail.userId",

"orderId.\$": "\$.detail.orderId",

"cause.\$": "\$.cause"

```
 },
```

```
 "DetailType": "OrderProcessor.OrderTimeOut",
```

```
 "EventBusName": "Serverlesspresso",
```

```
 "Source": "awsserverlessda.serverlesspresso"
```

```
 }
```

```
]
```

```
}
```

```
```

```

- Conecte o estado de passagem do tempo limite do Barista:

- Conecte o estado de passagem do Barista ao estado de Emit - error timeout. Selecione o estado de tempo limite do Barista e altere o Next state para `Emit - error timeout`.

2. Emitindo um evento de pedido concluído

- Adicione outro estado PutEvents:

- Com a aba Actions selecionada, pesquise por `putevents` novamente e arraste a ação Amazon EventBridge PutEvents para entre os estados Pass e End no designer.

- Configure o estado:

- Com o estado selecionado, no painel de atributos à direita:

- Nome do estado: `Emitir - pedido concluído`.

- Desmarque Aguardar retorno de chamada.

- Em Parâmetros de API, cole o seguinte JSON:

```
```json
```

```
{
```

```
 "Entries": [
```

```
 {
```

```
 "Detail": {
```

"Message": "The order has reached the end of the workflow, and so a final event is emitted to alert other services to this.",

```
 "userId.$": "$.detail.userId",
```

```
 "orderId.$": "$.detail.orderId"
```

```
 },
```

```
 "DetailType": "OrderProcessor.orderFinished",
```

```
 "EventBusName": "Serverlesspresso",
```

```
 "Source": "awsserverlessda.serverlesspresso"
```

```
 }
```

```
]
```

```
}
```

```
...
```

### 3. Atualizando o evento de loja não pronta

- Atualize o estado PutEvents existente:

- Selecione o estado PutEvents que emite um evento quando a loja não está pronta (entre os estados Loja aberta? e End).

- Configure o estado:

- No painel de atributos à direita:

- Nome do estado: `Emitir - Loja não pronta`.

- Desmarque Aguardar retorno de chamada.

- Em Parâmetros de API, cole o seguinte JSON:

```
```json
{
  "Entries": [
    {
      "Detail": {
        "Message": "The Step functions workflow checks if the shop is open and has capacity to serve a new order by invoking a Lambda function that queries the Shop config service. The shop was not ready, and so a 'not ready' event is emitted to cancel the current order.",
        "userId.$": "$.detail.userId"
      },
      "DetailType": "OrderProcessor.ShopUnavailable",
      "EventBusName": "Serverlesspresso",
      "Source": "awsserverlessda.serverlesspresso"
    }
  ]
}
```
``
```

#### 4. Finalizando as alterações

- Salvar as alterações:
  - Selecione Aplicar e sair. Na página de edição, clique em Salvar.
  - No pop-up do IAM, escolha Salvar mesmo assim.

Para testar o fluxo de trabalho do seu aplicativo de pedidos de bebidas com diferentes estados da loja (aberta e fechada), siga as instruções abaixo. Essas etapas garantem que você possa observar como o fluxo de trabalho se comporta em cada situação e validar o comportamento do sistema.

#### 1. Testando o fluxo de trabalho com a loja aberta

- Acesse o console do Step Functions:
  - No AWS Management Console, vá para Services e selecione Step Functions em Application Integration.
  - Confirme que a região correta está selecionada.
- Inicie a execução do fluxo de trabalho:
  - Em State machines, selecione OrderProcessorWorkflow.

- Na página do fluxo de trabalho, clique em Start execution.

- Insira a carga JSON:

- No pop-up Iniciar execução, cole o seguinte JSON e clique em Iniciar execução:

```
```json
{
  "detail": {
    "orderId": "1",
    "userId": "testuser"
  }
}
```

```

- Verifique o resultado:

- O inspetor de gráfico mostrará o caminho do fluxo de trabalho seguido como resultado do estado aberto da loja.

## 2. Testando o fluxo de trabalho com a loja fechada

- Acesse o console do DynamoDB:

- No AWS Management Console, vá para Services e depois selecione DynamoDB em Database.

- Verifique se a região correta está selecionada.

- Atualize o estado da loja:

- No menu à esquerda, escolha Explore items no menu Tables e selecione serverlesspresso-config-table.

- Selecione o item de configuração no painel Itens retornados\*\*. Isso abrirá o editor de itens.

- Altere para a visualização JSON e desative a View DynamoDB JSON.

- Defina a loja para fechada, colando o seguinte JSON, que define `storeOpen` como `false`:

```
```json
{
  "PK": "config",
  "storeOpen": false,
  "maxOrdersPerUser": 1,
  "maxOrdersInQueue": 10
}
```

```

```

- Salve as alterações:
 - Clique em Salvar alterações para atualizar a tabela.
 - Inicie a execução do fluxo de trabalho novamente:
 - Retorne ao console do Step Functions e selecione OrderProcessorWorkflow novamente.
 - Clique em Start execution.
 - Insira a mesma carga JSON:
 - No pop-up **Iniciar execução, cole o mesmo JSON e clique em Iniciar execução:

```json

```
{
 "detail": {
 "orderId": "1",
 "userId": "testuser"
 }
}
```

```

- Verifique o resultado:
 - O inspetor de gráfico mostrará o caminho do fluxo de trabalho seguido como resultado do estado fechado da loja.
 - Expanda o evento com o tipo TaskScheduled para a etapa EventBridge PutEvents no painel Histórico de eventos de execução. Isso mostrará que o fluxo de trabalho emitiu um evento indicando que a loja está indisponível.

3. Restaure o estado da loja para aberto

- Acesse o console do DynamoDB novamente:
 - No AWS Management Console, vá para Services e depois selecione DynamoDB em Database.
 - Atualize o estado da loja de volta para aberta:
 - No menu à esquerda, escolha Explore items no menu Tables e selecione serverlesspresso-config-table.
 - Selecione o item de configuração no painel Itens retornados.
 - Altere para a visualização JSON e desative a View DynamoDB JSON.
 - Cole o seguinte JSON, que define `storeOpen` como `true`:

```json

```
{
 "PK": "config",
 "storeOpen": true,
 "maxOrdersPerUser": 1,
 "maxOrdersInQueue": 10
}
...
}
```

- Salve as alterações:

- Clique em Salvar alterações para atualizar a tabela.

---

## 2. Eventos de Roteamento:

Os eventos desempenham um papel crucial na arquitetura de sistemas distribuídos e na comunicação entre microsserviços, especialmente dentro do contexto da AWS. Vamos explorar os principais conceitos e características dos eventos, bem como como eles são utilizados com o Amazon EventBridge.

O que são eventos?

Um evento é um sinal de que o estado de um sistema mudou. Na AWS, um evento é representado como uma mensagem JSON que contém informações sobre a mudança de estado e, potencialmente, sobre o estado atual do sistema. Aqui estão algumas características fundamentais dos eventos:

- Fatos: Eventos são baseados em ações que aconteceram, como a criação de um novo pedido.
- Imutáveis: Uma vez que um evento é emitido, ele não pode ser alterado. Se um pedido for cancelado, isso gera um novo evento de cancelamento, em vez de modificar o evento original.
- Observáveis: Microsserviços podem se inscrever e escutar eventos dos quais têm interesse.
- Temporal: O momento em que um evento ocorre é importante e pode influenciar a lógica de negócios.

Estrutura de um Evento

Um evento é encapsulado em uma mensagem JSON com um formato específico:

- Envelope: Contém atributos fornecidos pelo EventBridge, que identificam a conta AWS de origem, timestamp, região de origem e recursos AWS.
- Source: Atributo que identifica a origem do evento, definido pelo aplicativo que cria o evento. Por exemplo, `awsserverlessda.serverlesspresso`.
- Detail-Type: Define o tipo de evento. Por exemplo, `OrderProcessor.WorkflowStarted` indica que um fluxo de trabalho foi iniciado.
- Detail: O payload JSON que contém informações detalhadas sobre o evento. Para eventos personalizados, isso pode ser qualquer JSON arbitrário.

O tamanho máximo da mensagem é de 256 KB.

### Roteamento de Eventos

O Amazon EventBridge é responsável por roteamento e distribuição de eventos:

1. Barramentos de Eventos: Os eventos são publicados em barramentos, incluindo barramentos personalizados configurados para sua carga de trabalho.
2. Regras de Roteamento: Os assinantes (serviços da AWS ou microsserviços) usam regras para filtrar os eventos que desejam receber, permitindo que apenas os eventos relevantes sejam processados.
3. Desacoplamento: Os produtores de eventos não precisam saber quem está ouvindo os eventos que publicam, e os assinantes não precisam saber sobre os produtores. Isso promove uma arquitetura orientada a eventos que facilita o desenvolvimento ágil, aumenta a extensibilidade e reduz a complexidade entre equipes de desenvolvimento.

### Benefícios da Arquitetura Orientada a Eventos

- Desacoplamento: Os serviços são menos dependentes uns dos outros, permitindo que mudanças em um serviço não impactem diretamente outros serviços.
- Escalabilidade: Sistemas podem ser escalados de forma independente, já que os produtores e consumidores não precisam ser alterados simultaneamente.
- Agilidade no Desenvolvimento: Facilita a implementação de novos recursos e funcionalidades, já que os serviços podem evoluir sem a necessidade de coordenação constante.

### Registrar tudo:

Para configurar o registro de todos os eventos do Serverlesspresso no Amazon CloudWatch Logs, você precisará criar uma regra no Amazon EventBridge e testá-la para garantir que os eventos sejam registrados corretamente. Aqui está um resumo do processo:

#### 1. Criando a Regra "Registrar tudo" no EventBridge

Objetivo: Configurar uma regra que registre todos os eventos emitidos no barramento de eventos personalizado "Serverlesspresso" no CloudWatch Logs.

#### Instruções:

##### 1. Acessar o Amazon EventBridge:

- No Console de Gerenciamento da AWS, vá para "Services" e selecione "EventBridge" em "Application Integration".

- Verifique se a região selecionada está correta.

##### 2. Criar a Regra:

- Selecione "Regras" e clique em "Criar regra".

- Na Etapa 1 do assistente:

- Nome: Insira 'logAll'.

- Barramento de eventos: Insira `Serverlesspresso`.

- Clique em "Avançar".

- Na Etapa 2:

- Fonte do evento: Selecione "Outro".

- Ignore o painel de eventos de amostra.

- No painel "Padrão de evento", cole o seguinte JSON:

```
```json
{
  "source": ["awsserverlessda.serverlesspresso"]
}
```

```

- Clique em "Avançar".

- Na Etapa 3:

- No painel "Destino 1", escolha "serviço AWS".

- No menu suspenso "Selecionar um destino", escolha "grupo de logs do CloudWatch".

- No campo "Grupo de Log", insira `serverlesspressoEventBus`.

- Clique em "Avançar".

- Na Etapa 4, apenas revise as configurações e clique em "Avançar".

- Na Etapa 5, verifique se o "Event bus" está definido como `Serverlesspresso` e clique em "Criar regra".

## 2. Testando a Regra "Log All" do EventBridge

Objetivo: Verificar se a regra criada está registrando corretamente todos os eventos no grupo de logs do CloudWatch.

Instruções:

### 1. Gerar Eventos com Step Functions:

- No Console do AWS, vá para "Step Functions" e selecione `OrderProcessorWorkflow`.

- Na parte superior da página, clique em "Iniciar execução".

- No pop-up "Iniciar execução", insira a seguinte carga JSON e clique em "Iniciar execução":

```
```json
{

```

```
  "detail": {

```

```
    "orderId": "2",

```

```
    "userId": "testuser2"  
}  
}  
...  
``
```

- Observe o status de execução e o caminho seguido no fluxo de trabalho.

2. Verificar os Eventos no CloudWatch Logs:

- No Console do AWS, vá para "CloudWatch" e selecione "Log groups" no menu à esquerda.
- Escolha o grupo de logs chamado `/aws/events/serverlesspressoEventBus`.
- Selecione o primeiro fluxo de log e expanda a primeira linha para visualizar as informações detalhadas do evento, como `TaskToken`, `detail-type`, e `source`.

Nova Ordem:

Para automatizar o início do fluxo de trabalho `OrderProcessor` com base em um evento emitido pelo serviço `Validator`, siga os passos abaixo para criar e testar uma nova regra no Amazon EventBridge:

1. Criando a Regra "Nova Ordem"

Objetivo: Configurar uma regra no EventBridge que escute eventos `Validator.NewOrder` e inicie automaticamente o fluxo de trabalho `OrderProcessorWorkflow`.

Instruções:

1. Acessar o Amazon EventBridge:

- No Console de Gerenciamento da AWS, vá para "Services" e selecione "EventBridge" em "Application Integration".

- Verifique se a região selecionada está correta.

2. Criar a Regra:

- Selecione "Regras" e clique em "Criar regra".
- Na Etapa 1 do assistente:
 - Nome: Insira `NewOrder`.
 - Barramento de eventos: Insira `Serverlesspresso`.
 - Clique em "Avançar".

- Na Etapa 2:

- Fonte do evento: Selecione "Outro".
- Ignore o painel de eventos de amostra.
- No painel "Padrão de evento", cole o seguinte JSON:

```
```json
```

```
{
 "detail-type": ["Validator.NewOrder"],
 "source": ["awsserverlessda.serverlesspresso"]
}
```
```

- Clique em "Avançar".

- Na Etapa 3:

- No painel "Destino 1", escolha "serviço AWS".

- No menu suspenso "Selecionar um destino", escolha "Máquina de estado de funções de etapa".

- No menu suspenso "State machine", escolha `OrderProcessorWorkflow`. Dica: você pode começar a digitar `OrderProcessor` na caixa de pesquisa para encontrá-lo.

- Para "Função de execução", certifique-se de que "Criar uma nova função para o recurso específico" esteja selecionado.

- Clique em "Avançar".

- Na Etapa 4, apenas revise as configurações e clique em "Avançar".

- Na Etapa 5, verifique se o "Event bus" está definido como `Serverlesspresso` e clique em "Criar regra".

2. Testando a Regra "Nova Ordem"

Objetivo: Verificar se a nova regra inicia o fluxo de trabalho `OrderProcessorWorkflow` quando um evento `NewOrder` é emitido.

Instruções:

1. Simular o Evento NewOrder:

- No Console do AWS EventBridge, vá para "Eventos".

- Selecione "Ônibus de eventos" e escolha o barramento de eventos `Serverlesspresso`.

- Clique em "Enviar eventos".

- Verifique se o barramento de eventos `Serverlesspresso` está selecionado.

- Origem do evento: Insira `awsserverlessda.serverlesspresso`.

- Tipo de Detalhe: Insira `Validator.NewOrder`.

- Detalhes do evento: Insira o seguinte JSON:

```
```json
```

```
{"userId":"1","orderId":"1"}
```

```
```
```

- Clique em "Enviar".
- Isso criará um ID de evento e exibirá um resumo de confirmação.

2. Verificar a Execução no Step Functions:

- No Console do AWS, vá para "Step Functions" e selecione `OrderProcessorWorkflow`.
- Verifique se há uma nova execução na coluna "Nome" com o status `Running`.
- Selecione essa execução para ver o caminho seguido no fluxo de trabalho, com estados verdes indicando o progresso e estados azuis indicando pontos onde a execução está suspensa aguardando retorno.

Conclusão

Se o fluxo de trabalho `OrderProcessorWorkflow` foi iniciado corretamente, a nova regra foi configurada com sucesso, e o evento `NewOrder` está sendo corretamente roteado para iniciar o processamento de pedidos automaticamente. Agora, você pode prosseguir para criar regras adicionais para outros eventos ou ações no sistema.

Fluxo de trabalho iniciado:

Para capturar o evento `WorkflowStarted`, gravar o token de tarefa no DynamoDB, e testar a função Lambda que gerencia isso, siga as etapas abaixo:

1. Criando a Regra "WorkflowStarted"

Objetivo: Configurar uma regra no EventBridge que roteie o evento `WorkflowStarted` para uma função Lambda que armazenará o token de tarefa no DynamoDB.

Instruções:

1. Acessar o Amazon EventBridge:

- No Console de Gerenciamento da AWS, vá para "Services" e selecione "EventBridge" em "Application Integration".
- Verifique se a região selecionada está correta.

2. Criar a Regra:

- Selecione "Regras" e clique em "Criar regra".
- Na Etapa 1 do assistente:
 - Nome: Insira `WorkflowStarted`.
 - Barramento de eventos: Insira `Serverlesspresso`.
 - Clique em "Avançar".
- Na Etapa 2:
 - Fonte do evento: Selecione "Outro".
 - Ignore o painel de eventos de amostra.
 - No painel "Padrão de evento", cole o seguinte JSON:

```

```json
{
 "detail-type": ["OrderProcessor.WorkflowStarted"],
 "source": ["awsserverlessda.serverlesspresso"]
}
```

```

- Clique em "Avançar".

- Na Etapa 3:

- No painel "Destino 1", escolha "serviço AWS".

- No menu suspenso "Selecionar um alvo", escolha "Lambda".

- No menu suspenso "Function", escolha a função Lambda associada ao nome 'WorkflowStarted'. Dica: você pode começar a digitar "WorkflowStarted" no campo para encontrar a função.

- Clique em "Avançar".

- Na Etapa 4, apenas revise as configurações e clique em "Avançar".

- Na Etapa 5, verifique se o "Event bus" está definido como `Serverlesspresso` e clique em "Criar regra".

2. Testando a Regra "WorkflowStarted"

Objetivo: Verificar se a nova regra invoca a função Lambda corretamente, armazena o token de tarefa no DynamoDB, e inspecionar os logs e a tabela DynamoDB.

Instruções:

1. Iniciar um Novo Fluxo de Trabalho:

- No Console do AWS EventBridge, vá para "Eventos".

- Selecione "Ônibus de eventos" e escolha o barramento de eventos `Serverlesspresso`.

- Clique em "Enviar eventos".

- Origem do evento: Insira `awsserverlessda.serverlesspresso`.

- Tipo de Detalhe: Insira `Validator.NewOrder`.

- Detalhes do evento: Insira o seguinte JSON:

```

```json

```

```
{"userId":"1","orderId":"1"}
```

```

```

```

- Clique em "Enviar".

- Isso deve acionar a regra `NewOrder`, que iniciará o fluxo de trabalho `OrderProcessorWorkflow` e emitirá um evento `WorkflowStarted`.

2. Verificar os Logs no CloudWatch:

- No Console da AWS, vá para "CloudWatch" em "Management & Governance".
- No menu esquerdo, escolha "Log groups".
- Selecione o grupo de logs chamado `/aws/events/serverlesspressoEventBus`.
- Verifique os fluxos de log mais recentes. Selecione o mais recente para expandir os detalhes e visualizar o evento `OrderProcessor.WorkflowStarted`, que inclui o `TaskToken`.

3. Verificar a Tabela no DynamoDB:

- No Console da AWS, vá para "DynamoDB" em "Database".
- No menu esquerdo, selecione "Explorar itens".
- Escolha a tabela `serverlesspresso-order-table`.
- No painel "Itens retornados", escolha o primeiro item para visualizar seu conteúdo.
- Verifique se o item contém o `TaskToken`, o `Order ID` (SK), e o `ORDERSTATE`.

Fluxo de trabalho do Order Manager:

O fluxo de trabalho do OrderManager gerencia as operações essenciais do processo de pedidos, interagindo diretamente com a tabela DynamoDB e o fluxo de trabalho do OrderProcessor. Vou te guiar pelos passos para entender, testar e interagir com esse fluxo de trabalho.

1. Entendendo o Fluxo de Trabalho do OrderManager

Funções Principais:

- Colocação do Cliente: Verifica se o pedido é válido e o insere na tabela `serverlesspresso-order-table`.
- Cancelar Pedido: Cancela o pedido, removendo ou marcando-o na tabela.
- Concluir Pedido: Marca o pedido como concluído quando o barista finaliza a preparação.
- Fazer/Desfazer: Move o pedido entre os estados de pendente e em preparação, conforme a interação do barista.

Estado Inicial:

- O fluxo começa com um estado `Choice`, que verifica o campo `\$.action` e direciona o fluxo de acordo com a ação solicitada (`placeOrder`, `cancelOrder`, `completeOrder`, `claimOrder`, etc.).

2. Testando o Fluxo de Trabalho “OrderManager”

Objetivo: Executar e testar o fluxo de trabalho do OrderManager para verificar se ele interage corretamente com o OrderProcessor e a tabela DynamoDB.

Instruções:

1. Acessar o Console do Step Functions:

- No Console de Gerenciamento da AWS, vá para "Services" e selecione "Step Functions" em "Application Integration".

- Verifique se a região está correta.

2. Executar o OrderManagerStateMachine:

- Selecione "State machines" no painel esquerdo.

- Escolha a máquina de estado `OrderManagerStateMachine`.

- Clique em "Start execution".

3. Enviar um Pedido Válido:

- Na área de texto de entrada, insira o seguinte JSON para simular a colocação de um pedido válido:

```
```json
{
 "action": "placeOrder",
 "body": {
 "userId": "1",
 "drink": "Cappuccino",
 "modifiers": [],
 "icon": "barista-icons_cappuccino-alternative"
 },
 "orderId": "1",
 "baristaUserId": "3"
}
```

- Clique em "Start execution" para iniciar o fluxo de trabalho.

**4. Verificar o Efeito no OrderProcessor:**

- No menu de navegação, selecione "State machines" e escolha o fluxo de trabalho `OrderProcessor`.

- Veja a execução mais recente e verifique se o fluxo de trabalho progrediu conforme esperado. O token de tarefa (`TaskToken`) deve ter sido usado para enviar uma chamada `TaskSuccess`, permitindo que o OrderProcessor continue.

**5. Testar um Pedido Inválido:**

- Retorne ao `OrderManagerStateMachine`.

- Clique em "Start execution" novamente.
- Insira o seguinte JSON para simular um pedido de uma bebida que não está no menu:

```
```json
{
  "action": "placeOrder",
  "body": {
    "userId": "1",
    "drink": "milkshake",
    "modifiers": [],
    "icon": "barista-icons_cappuccino-alternative"
  },
  "orderId": "1",
  "baristaUserId": "3"
}
```

```

- Clique em "Start execution".

- Resultado Esperado: O fluxo de trabalho deve seguir um caminho diferente, já que o item "milkshake" não está disponível no menu. O pedido não será inserido na `serverlesspresso-order-table`.

### 3. Verificação e Conclusão

Após executar os testes:

- Verifique se os pedidos válidos são inseridos corretamente na tabela DynamoDB.
- Verifique no CloudWatch se as execuções foram registradas corretamente.
- Certifique-se de que o fluxo de trabalho do OrderProcessor está interagindo como esperado, progredindo após a execução do OrderManager.

Isso completa a configuração e o teste do fluxo de trabalho do OrderManager. Ao seguir essas etapas, você garantirá que o sistema de gerenciamento de pedidos funcione corretamente, permitindo que os pedidos sejam processados, atualizados e concluídos conforme necessário.

Aguardando a conclusão:

Criando a Regra "WaitingCompletion"

A seguir estão os passos detalhados para criar a regra "WaitingCompletion" no Amazon EventBridge, que irá rotear o evento emitido pelo fluxo de trabalho `OrderProcessor` para uma função Lambda. Essa função atualizará a tabela `serverlesspresso-order-table` com o novo `TaskToken`, número do pedido e estado do pedido.

## 1. Acesse o Console do EventBridge

- No AWS Management Console, selecione Services e, em seguida, vá para EventBridge em Application Integration.

- Verifique se a região está correta para o ambiente de implantação.

## 2. Crie uma Nova Regra

- Selecione Rules no painel de navegação à esquerda.
- Clique em Create rule para iniciar o assistente de criação de regras.

## 3. Configuração da Regra

Na Etapa 1:

- Para o Nome, insira `WaitingCompletion`.
- Para Event bus, insira `Serverlesspresso`.
- Clique em Next para prosseguir.

## 4. Definição do Padrão de Evento

Na Etapa 2:

- Em Event source, selecione Other.
- Ignore o painel de eventos de amostra.
- No painel Event pattern, cole o seguinte código:

```
```json
{
    "detail-type": ["OrderProcessor.WaitingCompletion"],
    "source": ["awsserverlessda.serverlesspresso"]
}
```

- Clique em Next.

5. Configuração do Destino

Na Etapa 3:

- No painel Target 1, selecione AWS service.
- No menu suspenso Select a target, escolha Lambda function.
- No menu suspenso Function, selecione a função Lambda que contém o nome `WaitingCompletion`. (Você pode começar a digitar “WaitingCompletion” para encontrar a função rapidamente).
- Clique em Next.

6. Revisão e Criação

Na Etapa 4:

- Revise as configurações. Certifique-se de que o Event bus seja `Serverlesspresso`.
- Clique em Next para prosseguir.

Na Etapa 5:

- Verifique se todos os detalhes estão corretos.
- Clique em Create rule para finalizar a criação.

7. Revisão das Regras Criadas

Após criar a regra, siga estas etapas para revisar e garantir que todas as regras foram criadas corretamente:

1. Vá para a Página de Regras

- No painel de navegação do EventBridge, selecione Rules.

2. Selecione o Event Bus:

- Altere o menu suspenso Event bus para `Serverlesspresso`.

3. Verifique as Regras:

- Verifique se as 4 novas regras estão listadas, além das 4 regras criadas anteriormente durante o processo de configuração.

Teste de ponta a ponta:

Teste de Ponta a Ponta: Finalizando um Pedido

Aqui está um passo a passo para testar os fluxos de trabalho `OrderProcessor` e `OrderManager` em um cenário completo de criação, atualização e conclusão de um pedido de bebida.

1. Criando um Novo Pedido de Bebida

Este passo simula um cliente escaneando um código QR e iniciando um novo pedido.

Instruções:

1. Vá para o Console do EventBridge:

- Acesse a aba do EventBridge no seu navegador.
- Selecione Event bus e escolha `Serverlesspresso`.

2. Enviar Evento:

- Selecione Send events.
- Verifique se o `Serverlesspresso` está selecionado.
- Insira os detalhes do evento:

- Event source: `awsserverlessda.serverlesspresso`

- Event detail type: `Validator.NewOrder`

- Event detail:

```
```json
```

```
{"userId":"1","orderId":"2"}
```

```
```
```

- Clique em Send.

3. Verificar no OrderProcessor:

- Vá para a aba do `OrderProcessorWorkflow` no console do Step Functions.

- No painel Executions, abra a execução mais recente que estará em estado `Running`.

- O workflow estará pausado no estado `WorkflowStarted`.

4. Verificar no DynamoDB:

- Vá para a aba do DynamoDB e acesse a tabela `serverlesspresso-order-table`.

- Encontre a entrada onde `SK` é `2` e selecione Orders na coluna `PK`.

- Verifique se o `TaskToken` foi salvo junto com o ID do pedido.

2. Adicionando Detalhes ao Pedido

Simule a personalização do pedido pelo cliente através do `Customer App`.

Instruções:

1. Executar o Fluxo de Trabalho do OrderManager:

- Na aba do `OrderManagerStateMachine`, selecione Start execution.

- Insira a seguinte carga útil:

```
```json
```

```
{"action": "", "body": {"userId": "1", "drink": "Cappuccino", "modifiers": [], "icon": "barista-icons_cappuccino-alternative"}, "orderId": "2", "baristaUserId": "3"}
```

```
```
```

- Clique em Start execution.

2. Verificar a Retomada do OrderProcessor:

- Volte para a aba do `OrderProcessorWorkflow` e observe que o fluxo de trabalho foi retomado.

3. Verificar no DynamoDB:

- Verifique a tabela `serverlesspresso-order-table` para confirmar que o número do pedido e o `TaskToken` foram atualizados corretamente.

- A execução do `OrderProcessor` estará agora pausada, aguardando a conclusão do pedido.

3. Reivindicando o Pedido

Simule o barista reivindicando o pedido.

Instruções:

1. Executar o Fluxo de Trabalho do OrderManager:

- Na aba do `OrderManagerStateMachine`, selecione Start execution.

- Insira a seguinte carga útil:

```
```json
{
 "action": "make",
 "body": {},
 "orderId": "2",
 "baristaUserId": "3"
}
```

- Clique em start execution.

#### 2. Verificar Atualização no DynamoDB:

- O `OrderManager` atualizará a tabela com o ID do barista.

### 4. Concluindo o Pedido

Simule o barista finalizando o pedido no sistema.

Instruções:

#### 1. Executar o Fluxo de Trabalho do OrderManager:

- Na aba do `OrderManagerStateMachine`, selecione \*\*Start execution\*\*.

- Insira a seguinte carga útil:

```
```json
```

```
{"action": "complete", "body": {"userId": "1", "drink": "Cappuccino", "modifiers": [], "icon": "barista-icons_cappuccino-alternative"}, "orderId": "2", "baristaUserId": "3"}
```

```
```
```

- Clique em Start execution.

#### 2. Verificar a Conclusão do Pedido:

- Volte para a aba do `OrderProcessorWorkflow` e veja que a execução foi concluída.

- 
- No DynamoDB, confirme que o status do pedido foi atualizado para `Concluído`.

### 3. Configurando o Front End:

Visão Geral:

Configuração dos Aplicativos Frontend para o Serverlesspresso

Visão Geral

Os frontends do Serverlesspresso permitem que clientes e baristas interajam com o backend do aplicativo. Os três aplicativos frontend disponíveis são:

1. Aplicativo de Exibição: Mostra o código QR e os pedidos de bebidas futuros/concluídos.
2. Aplicativo Barista: Permite que os baristas concluam e cancelm pedidos.
3. Aplicativo do Cliente: Permite que os clientes façam e cancelm pedidos de bebidas.

Estes aplicativos já foram implantados utilizando o AWS Amplify e se comunicam com o backend por meio do Amazon Cognito e Amazon API Gateway. Além disso, utilizam o AWS IoT Core para comunicação em tempo real via WebSockets.

Como Funciona

- AWS Cognito: Garante autenticação e identificação de usuários.
- Amazon API Gateway: Permite a comunicação entre os frontends e os serviços principais do backend.
- AWS IoT Core: Gerencia a conexão WebSocket, permitindo que os frontends recebam mensagens em tempo real, categorizadas por tópicos.

Os aplicativos frontend assinam tópicos via AWS SDK e aguardam notificações de alterações no status dos pedidos, usando o padrão publicar-assinar.

Configurando os Aplicativos Frontend

Para conectar os frontends ao backend, você precisa configurar cada um dos aplicativos usando a AWS CLI no AWS CloudShell.

Passo a Passo para Configuração

#### 1. Inicie o AWS CloudShell:

- No AWS Management Console, digite "CloudShell" na barra de pesquisa e selecione CloudShell nas opções.

#### 2. Recupere o `poolId`:

- No terminal do CloudShell, execute o seguinte comando:

```
```bash
```

```
aws cognito-identity list-identity-pools --max-results 10
```

```
```
```

- Isso retornará o `poolId`, que identifica o pool de identidades do Cognito usado pelos frontends.

### 3. Recupere o `host`:

- Execute o seguinte comando no CloudShell:

```
```bash
```

```
aws iot describe-endpoint --endpoint-type iot:Data-ATS
```

```
```
```

- Isso retornará o `host`, que é o endpoint do AWS IoT Core necessário para configurar a conexão WebSocket.

### 4. Anote os Valores:

- Copie os valores de `poolId` e `host` para um bloco de notas ou documento. Eles serão utilizados na configuração dos aplicativos frontend.

## Configuração dos Aplicativos Frontend

A configuração para todos os três aplicativos (Display App, Barista App, e Customer App) segue o mesmo processo, mas cada aplicativo deve ser configurado individualmente. Certifique-se de que os valores de `poolId` e `host` estão corretos ao configurar cada aplicativo.

Essa configuração conecta os frontends ao backend que você implantou, permitindo que os aplicativos funcionem corretamente no ambiente do Serverlesspresso.

Q cloudshell

Search results for 'cloudsh'

Services (42)

Features (24)

Blogs (4,675)

Tutorials (24)

Events (259)

## Services



CloudShell

A browser-based shell



CloudHSM

Managed Hardware Se



Cloud9

A Cloud IDE for Writin

# Welcome to AWS CloudShell

AWS CloudShell is a browser-based shell that gives you complete access to your AWS resources. AWS CloudShell comes pre-installed with popular tools for working with AWS services.

You have the same credentials as you used to log in to the AWS Management Console.



## Pre-installed tools

AWS CLI, Python, Node.js and more

## Storage

1 GB of storage



Do not show again

```
aws cognito-identity list-identity-pools --max-results 10
```

```
FullAccess:~/environment $ aws cognito-
{
 "IdentityPools": [
 {
 "IdentityPoolId": "us-east-1:
 "IdentityPoolName": "Serverlesspresso"
 }
]
}
FullAccess:~/environment $ █
```

4. Execute este comando para recuperar o host valor:

```
aws iot describe-endpoint --endpoint-type iot:Data-ATS
```

```
FullAccess:~/environment $ aws iot d
{
 "endpointAddress": "a2ty1m17b5zn
}
```

Publicação de eventos nos frontends:

Publicação de Eventos nos Frontends do Serverlesspresso

Visão Geral

No Serverlesspresso, a comunicação entre o backend e os frontends é feita através de eventos publicados em tópicos do AWS IoT Core. O microsserviço Publisher desempenha o papel central de receber eventos e publicá-los nos tópicos apropriados, permitindo que os frontends (Aplicativo Barista, Aplicativo de Exibição e Aplicativo do Cliente) recebam notificações e atualizações em tempo real.

Como Funciona

1. Microsserviço Publisher:

- Recebe eventos por meio de regras configuradas no AWS EventBridge.
- Publica esses eventos em tópicos específicos no AWS IoT Core.

## 2. Tópicos no AWS IoT Core:

- Admin: Este tópico é destinado a eventos relacionados aos aplicativos de administração, como o Aplicativo Barista e o Aplicativo de Exibição.
- User: O Aplicativo do Cliente se inscreve neste tópico para receber informações específicas do usuário atualmente conectado.
- Config: Todos os aplicativos (Barista, Cliente, Exibição) se inscrevem neste tópico para receber atualizações de configuração do sistema, como mudanças no menu ou no estado da loja.

## 3. Lambdas e Regras do EventBridge:

- Existem três funções Lambda e três regras do EventBridge configuradas, cada uma responsável por publicar em um dos tópicos mencionados acima.

### Arquitetura

- EventBridge: As regras do EventBridge capturam eventos específicos do sistema e encaminham esses eventos para as funções Lambda apropriadas.
- Funções Lambda: Cada função Lambda corresponde a um dos tópicos (Admin, User, Config) e publica o evento recebido nesse tópico específico.
- AWS IoT Core: Os tópicos criados no AWS IoT Core recebem os eventos publicados pelas Lambdas, e os frontends inscritos nesses tópicos recebem as notificações em tempo real.

### Funcionamento dos Tópicos

- Admin: Notifica os frontends sobre eventos administrativos, como quando um barista atualiza o status de um pedido ou quando uma nova configuração é aplicada ao sistema.
- User: Envia notificações relacionadas ao usuário, como atualizações de pedidos ou alertas personalizados.
- Config: Distribui informações sobre mudanças de configuração que impactam todos os aplicativos, garantindo que todos estejam sincronizados com o estado atual do sistema.

Essa estrutura garante que todas as partes do sistema Serverlesspresso estejam coordenadas e atualizadas em tempo real, proporcionando uma experiência fluida tanto para os clientes quanto para os baristas.

### O aplicativo de exibição:

#### Configurando o Aplicativo de Exibição do Serverlesspresso

##### Visão Geral

O Aplicativo de Exibição do Serverlesspresso é uma interface visual que roda em um monitor acima do balcão de café. Ele serve para mostrar uma lista de bebidas futuras e concluídas e exibir um código QR que os clientes podem escanear para iniciar seus pedidos. Você vai configurá-lo para conectar-se ao backend e depois criar uma conta de administrador.

## Passos para Configuração

### 1. Acessando o Aplicativo de Exibição:

- No AWS Management Console, pesquise por CloudFormation e selecione-o na lista.
- Na Lista de Pilhas selecione a pilha que deve ter um nome semelhante a workshop.
- Vá para a aba Saídas.
- Encontre a saída chamada DisplayAppURI e abra a URL fornecida em uma nova aba do navegador. Esta é a interface do Aplicativo de Exibição.

### 2. Configurando o Aplicativo:

- A interface do aplicativo abrirá com a maioria das configurações já preenchidas.
- PoolId: Insira o valor de `poolId` que você obteve anteriormente.
- Host: Insira o valor de `host` que você obteve anteriormente.
- Selecione Salvar e recarregar para aplicar as configurações.

### 3. Criando uma Conta de Usuário Administrador:

- Na interface do Aplicativo de Exibição, selecione a aba Create Account.
- Insira um e-mail válido e crie uma senha. Selecione Create Account.
- Verifique o e-mail que você usou para obter o código de verificação e digite-o no aplicativo. Selecione Confirmar.

### 4. Configurando o Usuário como Administrador no Cognito:

- Abra o console do Cognito em uma nova aba do navegador e escolha ServerlesspressoUserPool.
- Vá para a aba Grupos e selecione Criar grupo.
- Nomeie o grupo como `admin` e escolha Criar grupo.
- Na aba Usuários, selecione o usuário que você criou.
- Em Associações de grupo, escolha Adicionar usuário ao grupo, selecione `admin` e escolha Adicionar.

### 5. Logando no Aplicativo de Exibição:

- Volte para a aba onde está o Aplicativo de Exibição e faça login com o usuário administrador que você criou.
- O aplicativo agora estará configurado e você verá a interface principal com os seguintes botões de administração:
  - Configurar aplicativo Barista: Para transferir configurações para o aplicativo Barista.
  - Configurar aplicativo de pedidos: Para transferir configurações para o aplicativo de pedidos.
  - Limpar configurações: Para limpar o cache de configurações locais.

- Sair: Para desconectar o usuário do Cognito.

#### 6. Observações:

- O código QR exibido mudará a cada cinco minutos e controlará o número total de pedidos mostrados na tela (10 por padrão).

The screenshot shows the AWS CloudFormation search results for the query 'cloud formation'. The search bar at the top contains the text 'cloud formation' with a magnifying glass icon. To the right, it says 'Search results for 'cloud form''.

**Services** (100)

- Features (169)
- Blogs (18,412)
- Documentation (3,383)
- Knowledge Articles (30)
- Tutorials (89)
- Events (490)
- Marketplace (423)

**Services**

**CloudFormation** Create and Manage Reso

**Top features**

StackSets   Resource import

**AWS Well-Architect** Use AWS Well-Architecte

**Global Accelerator** Improve your application

**MediaStore** Store and deliver video a

**CloudFormation** X

- Stacks **Stack details** A
- Drifts
- StackSets
- Exports

---

- Application Composer [New](#)
- IaC generator

---

▼ Registry

- Public extensions
- Activated extensions
- Publisher

---

Spotlight

CloudFormation > [Stacks](#) > workshop

**Stacks (1)** C

Filter by stack name

Filter status Active View nested < 1 >

| Stacks                                                      |
|-------------------------------------------------------------|
| workshop<br>2024-04-21 16:32:01 UTC+0200<br>CREATE_COMPLETE |

**Outputs (1)**

Search outputs

| Key           |
|---------------|
| DisplayAppURI |

[https://workshop-display.serverlesscoffee.com/?region=us-east-1&userPoolId=us-east-1\\_uOKOowviD&useCustomHost=true](https://workshop-display.serverlesscoffee.com/?region=us-east-1&userPoolId=us-east-1_uOKOowviD&useCustomHost=true)

#### ADD YOUR BACKEND SETTINGS

This hosted UI will connect to the backend stack you deploy in the workshop. Enter the parameters from your backend stack.

REGION (E.G. US-WEST-2)

us-east-1

USERPOOLID (E.G. US-EAST-1\_ABCDEFGH2)

us-east-1\_uOKOowviD

USERPOOLWEBCLIENTID (E.G. 123A456BCDE789FGHI012JKL)

isu35pffi925kg1vur02hsdk3

POOLID (E.G. US-EAST-1:ABCD1234-ABCD-ABCD-A123-ABC123ABC12)

HOST (E.G. A1BC2C45D6FGH7-ATS.IOT.US-EAST-1.AMAZONAWS.COM)

ORDERMANAGERENDPOINT

<https://tla7w33hth.execute-api.us-east-1.amazonaws.com/Prod/>

APIGWENDPOINTVALIDATORSERVICE

<https://krjgjht9c3.execute-api.us-east-1.amazonaws.com/Prod/>

APIGWENDPOINTCONFIGSERVICE

<https://whuw12v727.execute-api.us-east-1.amazonaws.com/Prod/>

**Save and reload**

Serverlesspresso Workshop - Disp X +

← → C 🔒 workshop-display.serverlesscoffee.com

**Sign In**

Email

Password

**Sign In**

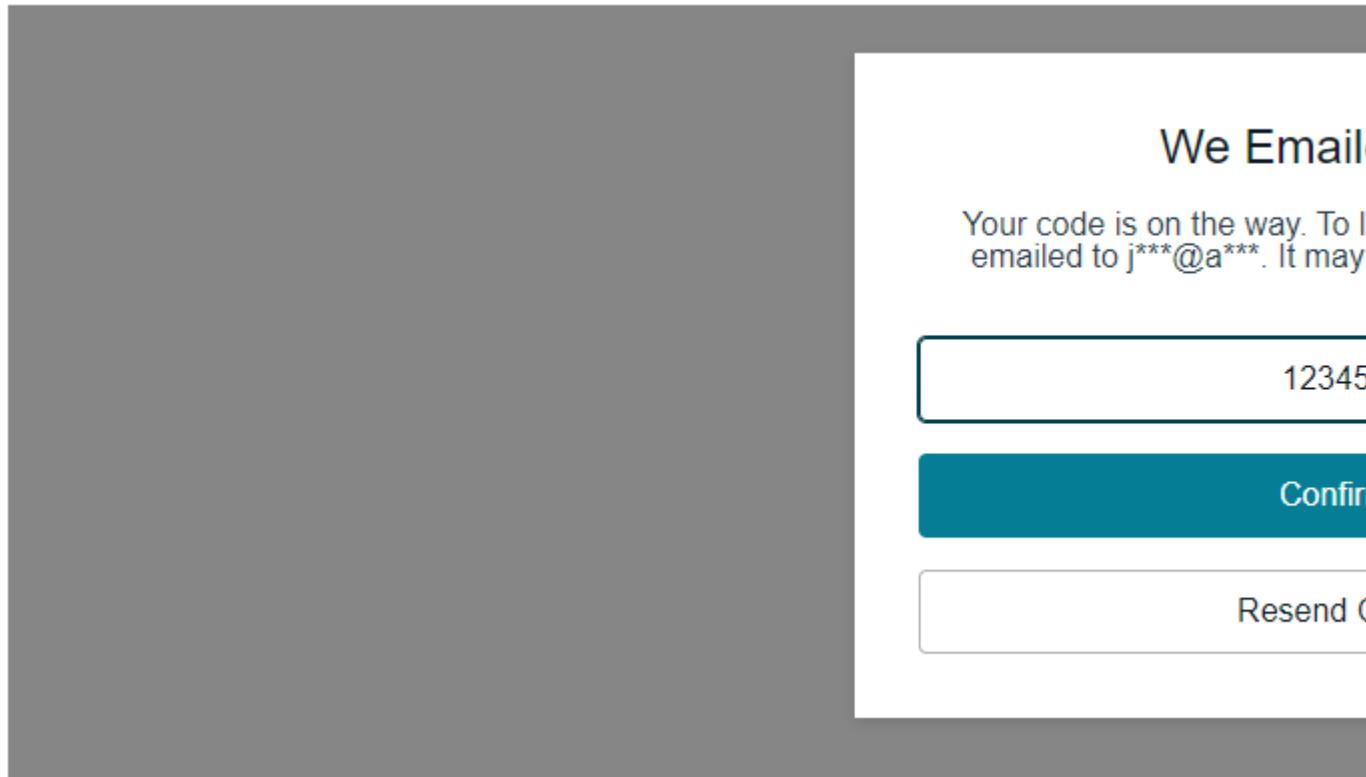
[Forgot your password?](#)

**Sign In**

jbeswick@americaneagle.com

.....  
.....

**Create Account**



Amazon Cognito X

User pools

Federated identities

Amazon Cognito > User pools

User pools (1) Info

View and configure your user pools. User pools are directories for managing your users.

C Delete Create user pool

Search user pools by name or ID

| User pool name           | User pool ID |
|--------------------------|--------------|
| ServerlesspressoUserPool | us-east-2... |

[User pools](#)

Federated identities

ServerlesspressoUserPool [Info](#)

## User pool overview

User pool name

ServerlesspressoUserPool

ARN

[arn:aws:cognito-idp:us-east-2:26237104669:userpool/us-east-2\\_94AyvA8gs](#)

User pool ID

[us-east-2\\_94AyvA8gs](#)

Estimated number of users

1

## ▶ Getting started

&lt; | Users |

[Groups](#)

Sign-in experience

Sign-up experience

Groups (0) [Info](#)

Configure groups and add users. Groups can be used to add permissions to the access token for m

[Delete](#)[Create group](#) Filter groups by name and description

Group name

Description

Precedence

No groups found

[Create group](#)

## Amazon Cognito

User pools

Federated identities

✓ Group "admin" has been created successfully.

Amazon Cognito > User pools > ServerlesspressoUserPool

## ServerlesspressoUserPool Info

### User pool overview

User pool name

ServerlesspressoUserPool

ARN

arn:aws:cognito-idp:us-east-2:262371046699:pool-94AyvA8gs

User pool ID

us-east-2\_94AyvA8gs

Estimated number of users

1

### ▶ Getting started



Users

Groups

Sign-in experience

### Groups (1) Info

Configure groups and add users. Groups can be used to add permissions to specific users.

[Delete](#)

[Create group](#)

Filter groups by name and description

Group name

Description

admin

## Amazon Cognito

### User pools

Federated identities

Amazon Cognito > User pools > ServerlesspressoUserPool

## ServerlesspressoUserPool Info

### User pool overview

User pool name

ServerlesspressoUserPool

ARN

arn:aws:cognito-user-pools:us-east-2:\_94AyvA8gs

User pool ID

us-east-2\_94AyvA8gs

Estimated number of users

1

### ▶ Getting started



[Users](#)

[Groups](#)

[Sign-in experience](#)

### Users (1) Info

View, edit, and create users in your user pool. Users that are enabled.



Delete user

Create user

User name



User name

Email address



james!

## Amazon Cognito

### User pools

Federated identities

X

Amazon Cognito > User pools > ServerlesspressoUser  
User: [REDACTED] > Add user to a group

## Add user to a group Info

### Groups (1) Info

Configure groups and add users. Groups can be used to add permissions.



Create group



Filter groups by property or value

Group name



admin

Scan the QR code to start!



**serverless**presso



10 drink(s) remaining.

**PREPARING**

O aplicativo do Barista:

Configurando o Aplicativo Barista do Serverlesspresso

Visão Geral

O Aplicativo Barista é utilizado pelos baristas para gerenciar pedidos em um tablet ao lado do balcão de café. Ele permite que os baristas visualizem pedidos futuros e marquem pedidos como concluídos ou cancelados.

Passos para Configuração

## 1. Acessando o Aplicativo Barista:

- Abra o link do Aplicativo Barista em seu navegador: [<https://workshop-barista.serverlesscoffee.com/>](<https://workshop-barista.serverlesscoffee.com/>).

- Ao acessar, a interface apresentará uma página de configuração.

## 2. Transferindo Configurações do Aplicativo de Exibição:

- Se você já configurou o Aplicativo de Exibição, alterne para a aba onde ele está aberto ou vá para [<https://workshop-display.serverlesscoffee.com/>](<https://workshop-display.serverlesscoffee.com/>).

- Clique no botão Configure Barista app na barra de ferramentas.

- Isso abrirá a página d configuração do Aplicativo Barista em uma nova janela, com as configurações de backend já preenchidas na URL.

## 3. Salvando as Configurações:

- Após abrir a página de configuração do Barista, clique em Salvar e recarregar para aplicar as configurações.

## 4. Login no Aplicativo Barista:

- Selecione a aba Sign In.

- Insira o e-mail e a senha da conta que você criou na seção anterior e clique em Sign In.

## 5. Interface do Aplicativo Barista:

- Após o login, o Aplicativo Barista será exibido. Na barra de ferramentas, você encontrará os seguintes botões de administração:

- Loja aberta: Alterna o estado da loja entre Open (Aberta) e Closed (Fechada). Quando a loja está fechada, o Aplicativo de Pedidos não pode fazer novos pedidos.

- Limpar configurações: Esvazia o cache de configurações locais e limpa as configurações de backend. Isso fará com que a página de configurações seja exibida na próxima vez que a página for recarregada.

- Sair: Desconecta o usuário do Cognito do frontend e retorna à página de login.

Mantenha o Aplicativo Barista aberto em uma aba do navegador para os próximos passos no fluxo de trabalho do seu sistema.

Scan the QR code to start!



**serverless**presso

**PREPARING**



10 drink(s) remaining.

#### ADD YOUR BACKEND SETTINGS

This hosted UI will connect to the backend stack you deploy in the workshop. Enter the environment parameters from your backend.

**REGION (E.G. US-WEST-2)**

us-east-1

**USERPOOLID (E.G. US-EAST-1\_ABCDEFGH2)**

us-east-1\_uOKOowviD

**USERPOOLWEBCLIENTID (E.G. 123A456BCDE789FGHI012JKL)**

isu35pffi925kg1vur02hsdk3

**POOLID (E.G. US-EAST-1:ABCD1234-ABCD-ABCD-A123-ABC123ABC12)**

us-east-1:79e8896c-33a8-4b4b-9e4e-1ec3316ebcc9

**HOST (E.G. A1BC2C45D6FGH7-ATS.IOT.US-EAST-1.AMAZONAWS.COM)**

ale6dkr395kalw-ats.iot.us-east-1.amazonaws.com

**ORDERMANAGERENDPOINT**

<https://tla7w33hth.execute-api.us-east-1.amazonaws.com/Prod/>

**APIGWENDPOINTVALIDATORSERVICE**

<https://krjgjht9c3.execute-api.us-east-1.amazonaws.com/Prod/>

**APIGWENDPOINTCONFIGSERVICE**

<https://whuw12v727.execute-api.us-east-1.amazonaws.com/Prod/>

**Save and reload**

V Serverlesspresso Workshop - Barista +

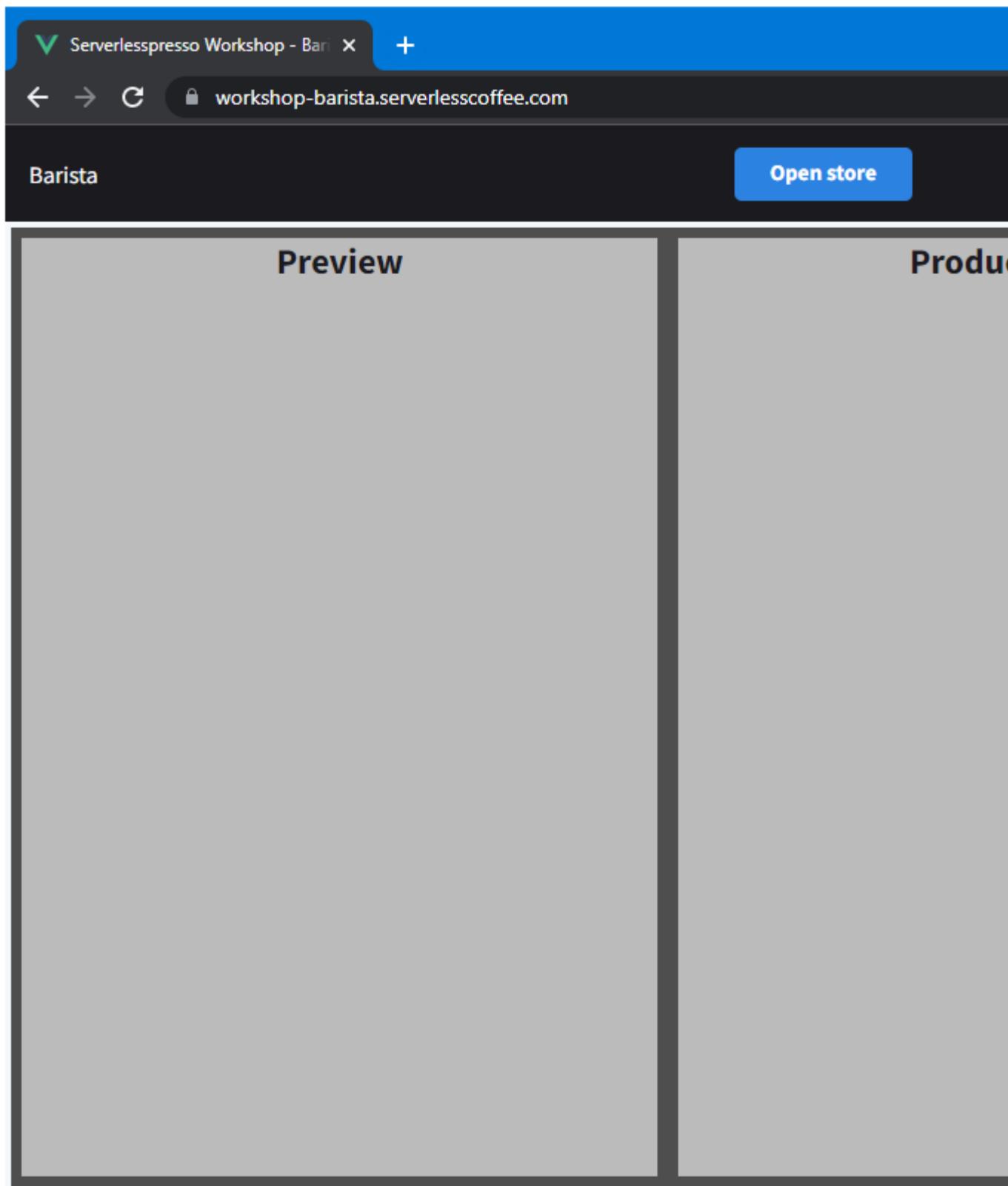
← → C 🔒 workshop-barista.serverlesscoffee.com

---

**Sign In** [Create Account](#)

**Sign in**

[Forgot your password?](#)



O aplicativo do Cliente:

Configurando o Aplicativo do Cliente do Serverlesspresso

Visão Geral

O Aplicativo do Cliente é utilizado pelos clientes em seus smartphones para fazer pedidos. Ao escanear o código QR pela primeira vez, eles são redirecionados para este aplicativo web.

#### Passos para Configuração

##### 1. Acessando o Aplicativo do Cliente:

- Abra o link do Aplicativo do Cliente em seu navegador: [https://workshop-order.serverlesscoffee.com/](https://workshop-order.serverlesscoffee.com/).
- Ao acessar, a interface apresentará uma página de configuração.

##### 2. Transferindo Configurações do Aplicativo de Exibição:

- Se você já configurou o Aplicativo de Exibição, alterne para a aba onde ele está aberto ou vá para [https://workshop-display.serverlesscoffee.com/](https://workshop-display.serverlesscoffee.com/).

- Clique no botão Configure order app na barra de ferramentas.
- Isso abrirá um pop-up contendo um código QR, que incorpora as configurações de backend em uma sequência de consulta.

##### 3. Escaneando o Código QR:

- No seu smartphone, use um aplicativo de scanner de código de barras para escanear o código QR exibido.
- Isso abrirá a página de configurações e preencherá os campos com as configurações de backend automaticamente.

##### 4. Salvando as Configurações:

- Após verificar se as configurações estão corretas, selecione Salvar e recarregar.

##### 5. Fechando a Aba:

- Você pode fechar esta aba do navegador por enquanto.

Scan the QR code to start!

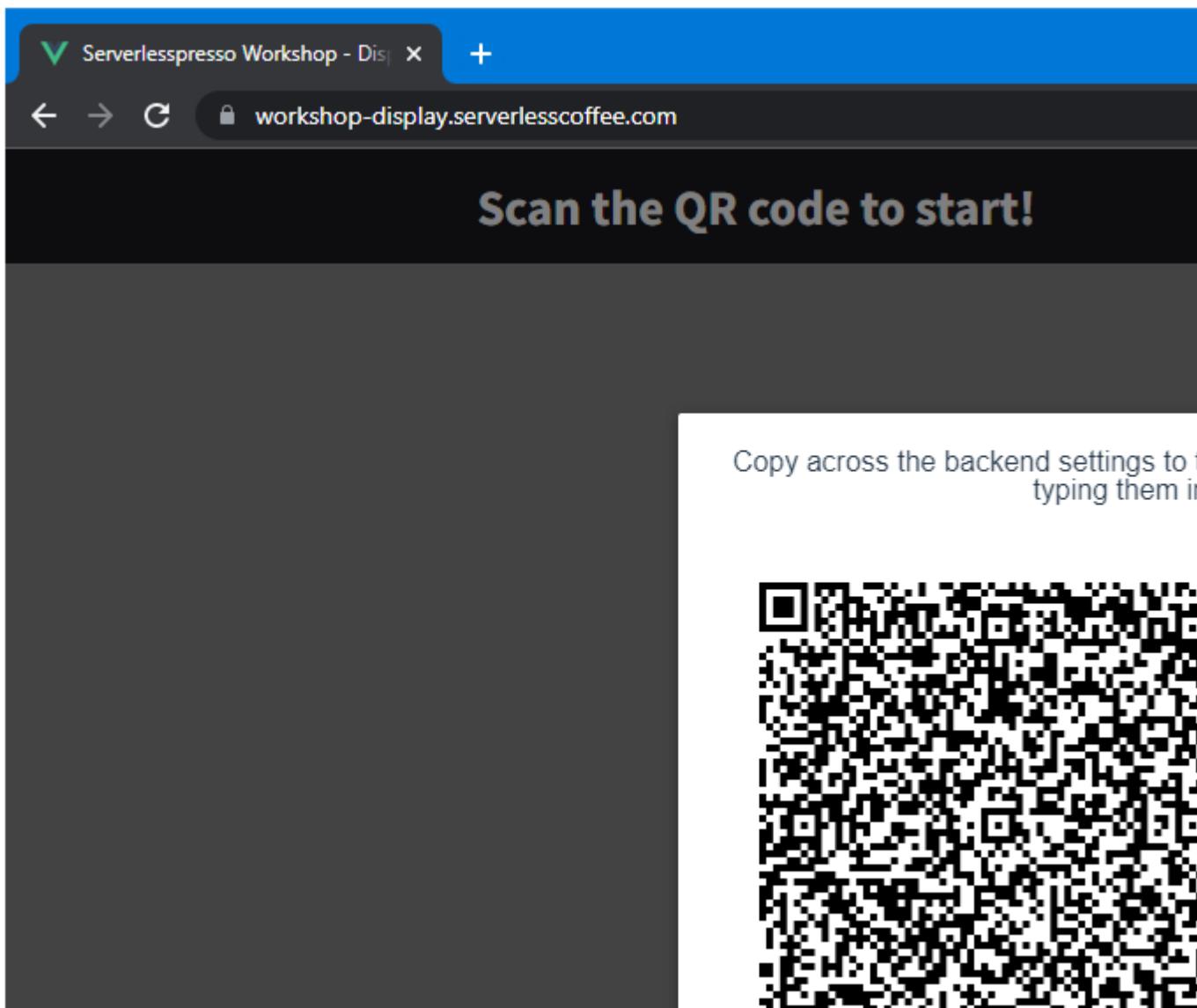


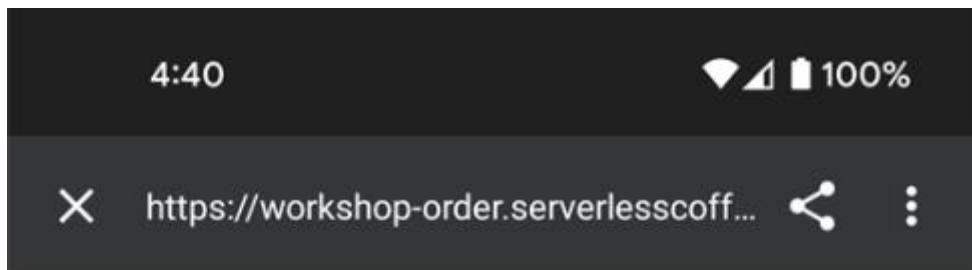
**serverless**presso



**PREPARING**

10 drink(s) remaining.





## ADD YOUR BACKEND SETTINGS

This hosted UI will connect to the backend stack you deploy in the workshop. Enter the environment variables and stack parameters from your backend.

REGION (E.G. US-WEST-2)

us-east-1

USERPOOLID (E.G. US-EAST-1\_ABCDEFGH2)

us-east-1\_SLyQEJgN4

USERPOOLWEBCLIENTID (E.G. 123A456BCDE789FGHI012JKL)

364o841trpu6semtpp47p4qtgbo

POOLID (E.G. US-EAST-1:ABCD1234-ABCD-ABCD-A123-ABC123A...)

us-east-1:a6da19f2-1bc3-4bc4-a776-c2d2cd4d5c

HOST (E.G. A1BC2C45D6FGH7-ATS.IOT.US-EAST-1.AMAZONAWS....)

a2ty1m17b5znw2-ats.iot.us-east-1.amazonaws.c

ORDERMANAGERENDPOINT

<https://41tzy6ap5a.execute-api.us-east-1.amazo>

APIGWENDPOINTVALIDATORSERVICE

<https://1rfne2v5e8.execute-api.us-east-1.amazo>

APIGWENDPOINTCONFIGSERVICE

<https://llo0k59n5g.execute-api.us-east-2.amazo>

Teste de ponta a ponta:

## Teste de Ponta a Ponta com o Aplicativo do Cliente

### Objetivo

Concluir um teste completo de ponta a ponta do sistema usando o Aplicativo do Cliente em um smartphone, interagindo com os aplicativos Display e Barista.

### Instruções Passo a Passo

#### 1. Preparação dos Aplicativos:

- Certifique-se de que o Display App e o Barista App estejam abertos em duas abas separadas do navegador.
- O Display App deve estar visível em uma tela acima do balcão de café, mostrando o código QR.
- O Barista App deve estar acessível em um tablet, pronto para processar os pedidos.

#### 2. Escaneando o Código QR:

- Abra o scanner de código de barras no seu smartphone. Se o seu smartphone não tiver um scanner de QR, baixe um aplicativo gratuito de scanner QR.
- Escaneie o código QR exibido no Display App.
- Nota: Se o código QR não estiver visível porque a tela entrou em um período de tempo limite, aguarde até que o código reapareça.

#### 3. Login no Aplicativo do Cliente:

- Após escanear o código QR, faça login no aplicativo usando a conta que você criou anteriormente.

#### 4. Fazendo um Pedido:

- Após validar o token, selecione uma bebida do menu.
- Escolha a opção Pedir agora.
- Importante: Certifique-se de que o seu navegador não esteja em modo "navegação privada" ou "incógnito", pois isso pode causar problemas com a configuração do aplicativo.

#### 5. Verificando os Pedidos:

- Após fazer o pedido, observe os aplicativos Display e Barista para ver o novo pedido chegar.
- No Display App, você verá o novo pedido listado.
- No Barista App, o pedido aparecerá na lista de pedidos futuros.

#### 6. Interagindo com o Barista App:

- No Barista App, você pode marcar o pedido como concluído ou cancelá-lo.
- Observe como as atualizações nos aplicativos Display e Cliente refletem essas mudanças em tempo real.

#### 7. Repetindo o Processo:

- Repita as etapas de 2 a 6 para fazer pedidos adicionais e experimentar diferentes funcionalidades do aplicativo.
- Para monitorar os fluxos de trabalho de cada bebida de café, você também pode navegar até o console Step Functions.

## Conclusão

Este teste de ponta a ponta permite que você verifique a interação entre os diferentes aplicativos e assegure que o sistema esteja funcionando conforme o esperado. Certifique-se de testar várias interações para garantir a robustez do sistema.

---

## 4. Avançado – Métricas de negócios com SQS e DynamoDB:

### Métricas de Negócios com SQS e DynamoDB

#### Visão Geral

Esta seção demonstra como a arquitetura orientada a eventos pode ser estendida para atender a novos requisitos funcionais sem impactar a pilha de aplicativos existente. Usando Amazon EventBridge, Amazon SQS e AWS Lambda, você poderá coletar e processar métricas de negócios relacionadas aos pedidos concluídos.

#### Objetivo

Roteamento de eventos de pedidos concluídos do fluxo de trabalho OrderManager para uma fila SQS. Uma função Lambda processará os eventos em lotes e atualizará uma tabela de métricas no DynamoDB.

#### Arquitetura de Fluxo de Trabalho

1. Coleta de Métricas: Utilizando uma nova regra no EventBus, você coletará métricas agregadas dos pedidos realizados. O evento OrderManager.WaitingCompletion será usado para atualizar a tabela serverlesspresso-metrics-table.

2. Métricas de Interesse: As métricas que podem ser coletadas incluem:

- Pedidos por tipo de item
- Totais de pedidos diários
- Contagem de pedidos concluídos ou cancelados

#### Passos para Configuração

##### 1. Iniciar o Modelo AWS CloudFormation

- Esta seção tem um modelo \*\*CloudFormation separado da pilha principal. Siga os passos abaixo para implantar os recursos necessários.

#### Passo a Passo:

##### 1. Selecione sua Região:

- Escolha a região onde você deseja implantar o modelo. Para este exemplo, use Leste dos EUA (Norte da Virgínia) us-east-1.

##### 2. Criar a Pilha:

- Clique no link Launch para criar a pilha em sua conta.

### 3. Configurações da Pilha:

- Insira um nome de pilha (ou mantenha o nome padrão).
- Marque as caixas na seção Capacidades para permitir que o CloudFormation crie os recursos necessários.

### 4. Criar a Pilha:

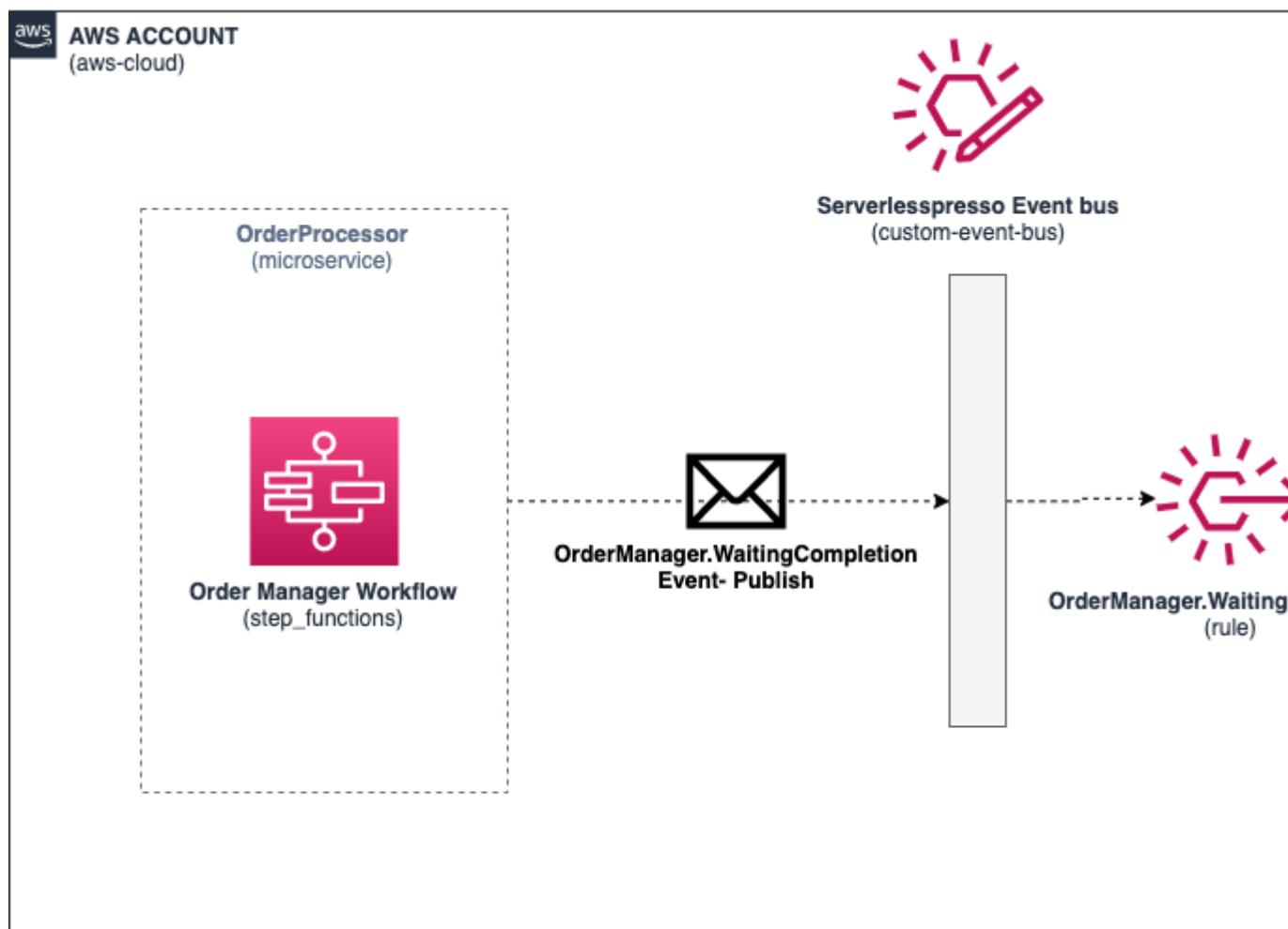
- Clique em Criar pilha e aguarde a conclusão da criação dos recursos.

#### Considerações Finais

- **Responsabilidade:** Ao executar esses modelos, você assume a responsabilidade pelo ciclo de vida e pelos custos associados ao provisionamento deles.

- **Desmontagem:** Após a conclusão do workshop, siga as instruções de desmontagem para remover todos os recursos da sua conta AWS e evitar custos inesperados.

Essa arquitetura não só ajudará na coleta de métricas, mas também proporcionará uma base sólida para futuros desenvolvimentos e melhorias no seu aplicativo.



Criando as regras:

## Criando a Regra “WaitingCompletion” no EventBridge

### Instruções Passo a Passo

#### 1. Acessar o Console do EventBridge:

- No AWS Management Console, selecione Services e, em seguida, EventBridge na seção Application Integration.
- Verifique se a região selecionada está correta.

#### 2. Criar uma Nova Regra:

- Selecione Regras no menu à esquerda.
- Clique em Criar regra.

#### 3. Etapa 1: Nome e Barramento de Eventos:

- Para Nome, insira OrderMetrics-DynamoDB.
- Para Barramento de eventos, digite Serverlesspresso.
- Selecione Avançar.

#### 4. Etapa 2: Configuração da Origem do Evento:

- Em Origem do evento, selecione Outro.
- Ignore o painel de eventos de amostra.
- No painel Padrão de evento, cole o seguinte código:

```
```json
{
    "source": ["awsserverlessda.serverlesspresso"],
    "detail-type": ["OrderManager.WaitingCompletion"]
}
```

- Selecione Avançar.

5. Etapa 3: Configuração do Destino:

- No painel Destino 1, escolha serviço AWS.
- No menu suspenso Selecionar um destino, escolha Fila SQS.
- No menu suspenso Fila, escolha Fila SQS MetricsQueue. (Você pode começar a digitar MetricsQueue no campo para encontrá-la facilmente).
- Selecione Avançar.

6. Etapa 4: Revisão da Configuração:

- Na Etapa 4 do assistente, escolha Avançar sem fazer alterações.

7. Etapa 5: Definir Regra:

- Verifique se o painel de detalhes Definir regra mostra que o Event bus é Serverlesspresso.
- Escolha Criar regra.

Conclusão

Após concluir essas etapas, a regra OrderMetrics-DynamoDB será criada e estará configurada para rotear os eventos OrderManager.WaitingCompletion para a fila MetricsQueue no SQS, permitindo que você colete métricas de negócios com eficiência.

Define rule detail [Info](#)

Rule detail

Name

Maximum of 64 characters consisting of numbers, lower/upper case letters, .,-,_.

Description - optional

Event bus | [Info](#)
Select the event bus this rule applies to, either the default event bus or a custom or partner event bus.

ⓘ Schedule rule is not supported when custom or partner event bus is selected.

Enable the rule on the selected event bus

Rule type | [Info](#)

Rule with an event pattern
A rule that runs when an event matches the defined event pattern. EventBridge sends the event to the specified target.

Schedule
A rule that runs on a schedule

```
{  
    "source": ["awsserverlessda.serverlesspresso"],  
    "detail-type": ["OrderManager.WaitingCompletion"]  
}
```

Select target(s)



Permissions

Note: When using the EventBridge console, EventBridge will automatically configure the proper permissions for the selected targets. If you're using the AWS CLI, SDK, or CloudFormation, you'll need to configure permissions.

Target 1

Target types

Select an EventBridge event bus, EventBridge API destination (SaaS partner), or another AWS service as a target.

- EventBridge event bus
- EventBridge API destination
- AWS service

Select a target | [Info](#)

Select target(s) to invoke when an event matches your event pattern or when schedule is triggered (limit of 5 targets per rule).

SQS queue

Queue

MetricsQueue

► Additional settings

Add another target

Cancel

Prev

Testando:

Testando a Regra “OrderManager.WaitingCompletion”

Instruções Passo a Passo

1. Teste de Ordem Única

- Crie um novo pedido no aplicativo de pedidos.
- Acesse a tabela serverlesspresso-metrics-table no DynamoDB.
- Você deverá ver 2 itens:
 - `2022-XX-XX#Total de Vendas`
 - `2022-XX-XX#NomeDoItem`
- Esses valores devem ser incrementados sempre que houver um evento OrderManager.WaitingCompletion no Event Bus.

2. Teste de Carga

- Vamos usar a função Lambda para simular um grande número de pedidos e ver como o SQS atua como um buffer durante picos de tráfego.
- Esses eventos serão roteados para MetricsQueue através da regra WaitingCompletion.

3. Configuração da Função Lambda:

- Cada invocação da função Lambda processará eventos em lotes de até 10 registros ou dentro de janelas de 30 segundos.
- A Concorrência Reservada está configurada para 1 execução simultânea da função Lambda, mesmo que o padrão permita até 1000 execuções simultâneas.
- Você pode personalizar ReservedConcurrentExecutions, BatchSize e MaximumBatchingWindowInSeconds conforme necessário.

4. Verificando a Tabela de Métricas:

- Navegue até o console do Lambda e procure pela função EventsLoadTest.
- Invoque a função EventsLoadTest clicando no botão Testar.
- Quando solicitado, crie um evento de teste:
 - Nome do Evento: Teste
 - JSON do Evento: Use o valor padrão.
- Clique em Salvar e, em seguida, clique no botão Testar novamente para executar o teste de carga.

5. Visualizando Resultados do Teste de Carga:

- Abra a função Lambda PublishMetrics e vá até a aba Monitoring.
- Expanda o gráfico de Invocações da função e selecione “Máximo” para visualizar a execução simultânea.
- Você deve ver apenas 1 execução simultânea durante o teste de carga.

6. Verificando a Tabela de Métricas:

- Acesse a tabela serverlesspresso-metrics-table no DynamoDB.
- Verifique os itens e você deve observar que cerca de 500 pedidos totais foram simulados durante o teste de carga.

7. Análise do Gráfico de Uso:

- Abra a aba Monitoramento na tabela serverlesspresso-metrics-table.
- Expanda o gráfico de uso de gravação e selecione “Máximo” para observar o consumo de capacidade.
- A linha vermelha representará a capacidade provisionada, enquanto a linha azul mostrará o consumo da função PublishMetrics.

Takeaways

- Implementamos métricas de negócios para os investidores ao criar uma nova regra e microsserviço sem modificar a pilha de aplicativos existente.
- O uso de SQS como buffer entre o Event Bus e o Lambda evita a limitação do banco de dados ao atualizar o DynamoDB.
- O monitoramento com CloudWatch é essencial para avaliar o desempenho da arquitetura orientada a eventos e identificar possíveis pontos fracos.

Items returned (8)			
	PK	SK	val
<input type="checkbox"/>	Aggregate	2022-10-10#Cappuccino	12
<input type="checkbox"/>	Aggregate	2022-10-10#Espresso	64
<input type="checkbox"/>	Aggregate	2022-10-10#Latte	30
<input type="checkbox"/>	Aggregate	2022-10-10#TotalSales	106

Functions (20)

 Filter by tags and attributes or search by keyword

eventsloadtest



Clear filters



Function name



sqs-module-EventsLoadTest-
Xxy8DRle5xg5

Code source [Info](#)

File Edit Find View Go Tools Window Test Deploy

Go to Anything (⌘ P)

Environment

EventsLoadTest.js

```
// Import required AWS SDK clients and command
const AWS = require("aws-sdk");
var eventbridge = new AWS.EventBridge({apiVersion: "2018-11-01"});

exports.handler = async (event) => {
    // Get drinks
    const drinks = ["Latte", "Cappuccino", "Espresso"];
    let batchSize = 10;

    for (let i = 1000; i < 1500; i+=batchSize)
        var entries = []
    
```

Test event action

Create new event

Edit save

Event name

Test

Maximum of 25 characters consisting of letters, numbers, dots, hyphens and underscores.

Event sharing settings

Private

This event is only available in the Lambda console and to the event creator. You can change this later.

Shareable

This event is available to IAM users within the same account who have permissions to access it.

Template - *optional*

hello-world

Event JSON

```
1  {  
2    "key1": "value1",  
3    "key2": "value2",  
4    "key3": "value3"  
5 }
```

Invocations

1 minute ▾

Maximum ▾

Reset zoom ⌂

1h

Count

2.0

1.5

1.0

0.5

0

13:47

13:48

13:49

13:50

13:51

13:52

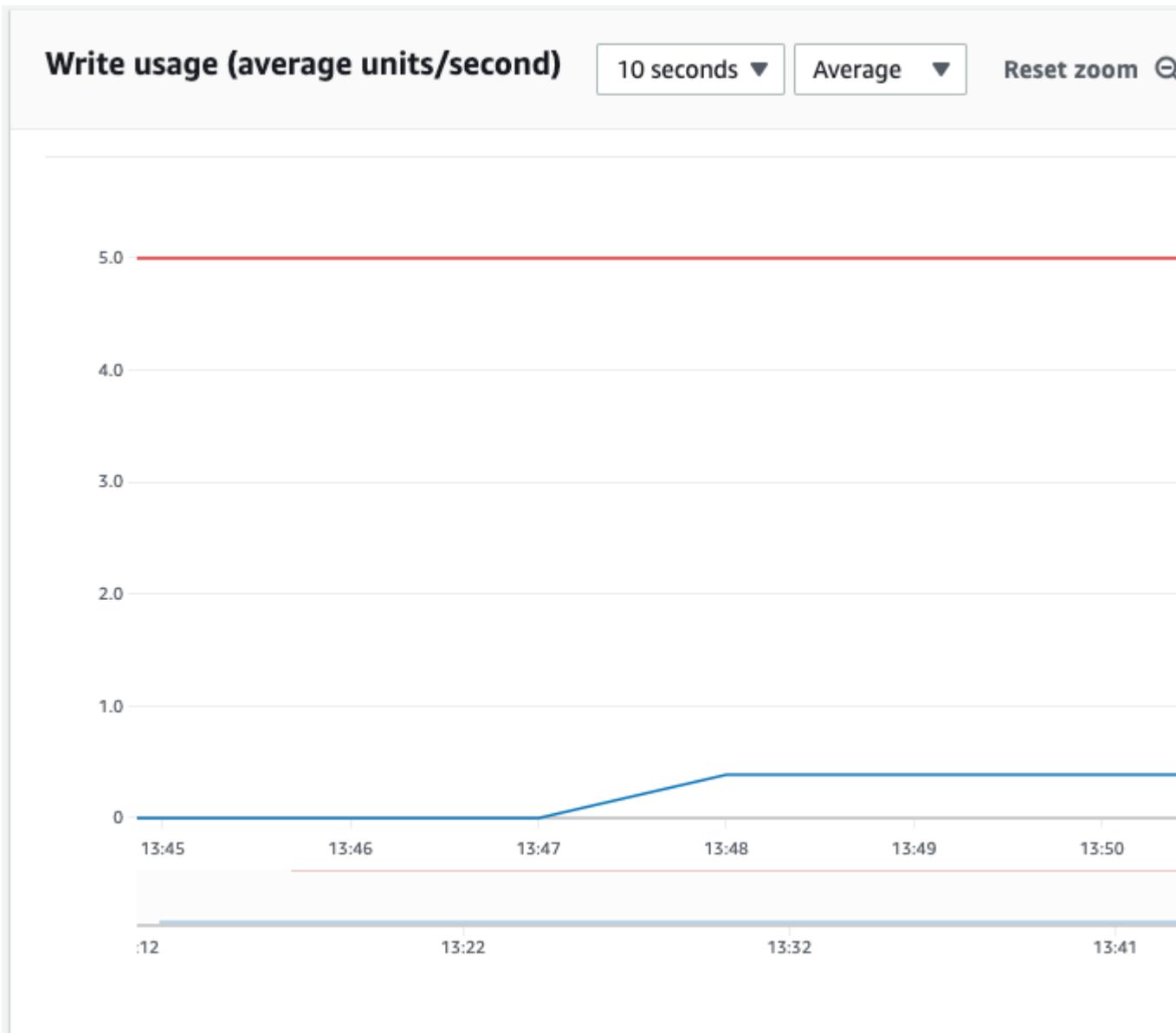
:01

13:10

13:20

13:30

■ Invocations



Métricas de negócios com SQS e Cloudwatch:

Visão Geral

Nesta seção, abordaremos a extensibilidade de arquiteturas orientadas a eventos. Com a frequência de novos requisitos funcionais em aplicativos de produção, é essencial adaptar-se sem impactar a pilha existente. Vamos demonstrar como rotearmos eventos de pedidos concluídos do fluxo de trabalho do OrderManager para uma fila SQS e como uma função Lambda processará esses eventos, inserindo métricas de negócios no CloudWatch.

Usar SQS como destino para eventos do EventBridge ajuda a aliviar a pressão nos sistemas downstream durante picos de tráfego.

Arquitetura de Fluxo de Trabalho

Os investidores da Serverlesspresso estão interessados em entender detalhes sobre as vendas diárias de bebidas. Embora possamos consultar a tabela de pedidos, é mais eficiente criar uma nova regra no Event Bus e um microsserviço para fornecer métricas agregadas.

Durante o fluxo de trabalho do pedido, a função WaitingCompletion emite um evento OrderManager.WaitingCompletion para o Event Bus. Esse evento pode ser usado para atualizar a tabela serverlesspresso-metrics-table. Podemos coletar métricas como:

- Pedidos por tipo de item.
- Totais de pedidos diários.

Além disso, a coleta de métricas pode ser estendida para incluir eventos OrderManager.OrderCancelled para rastrear pedidos concluídos ou cancelados ao longo do tempo.

Configuração da Regra EventBridge

Neste módulo, vamos configurar uma nova regra EventBridge para capturar todos os eventos OrderManager.WaitingCompletion e roteá-los para uma fila SQS. A função Lambda processará os eventos em lotes, atualizando as métricas no CloudWatch.

1. Inicie o Modelo AWS CloudFormation

Essa seção tem seu próprio modelo CloudFormation, separado da pilha principal. Siga os passos abaixo para implantar os recursos necessários:

1. Selecione sua Região:

- Escolha a região preferida (ex.: Leste dos EUA (Norte da Virgínia) us-east-1).

2. Clique no Link de Lançamento:

- Clique em Launch para criar a pilha na sua conta.

3. Configurações da Pilha:

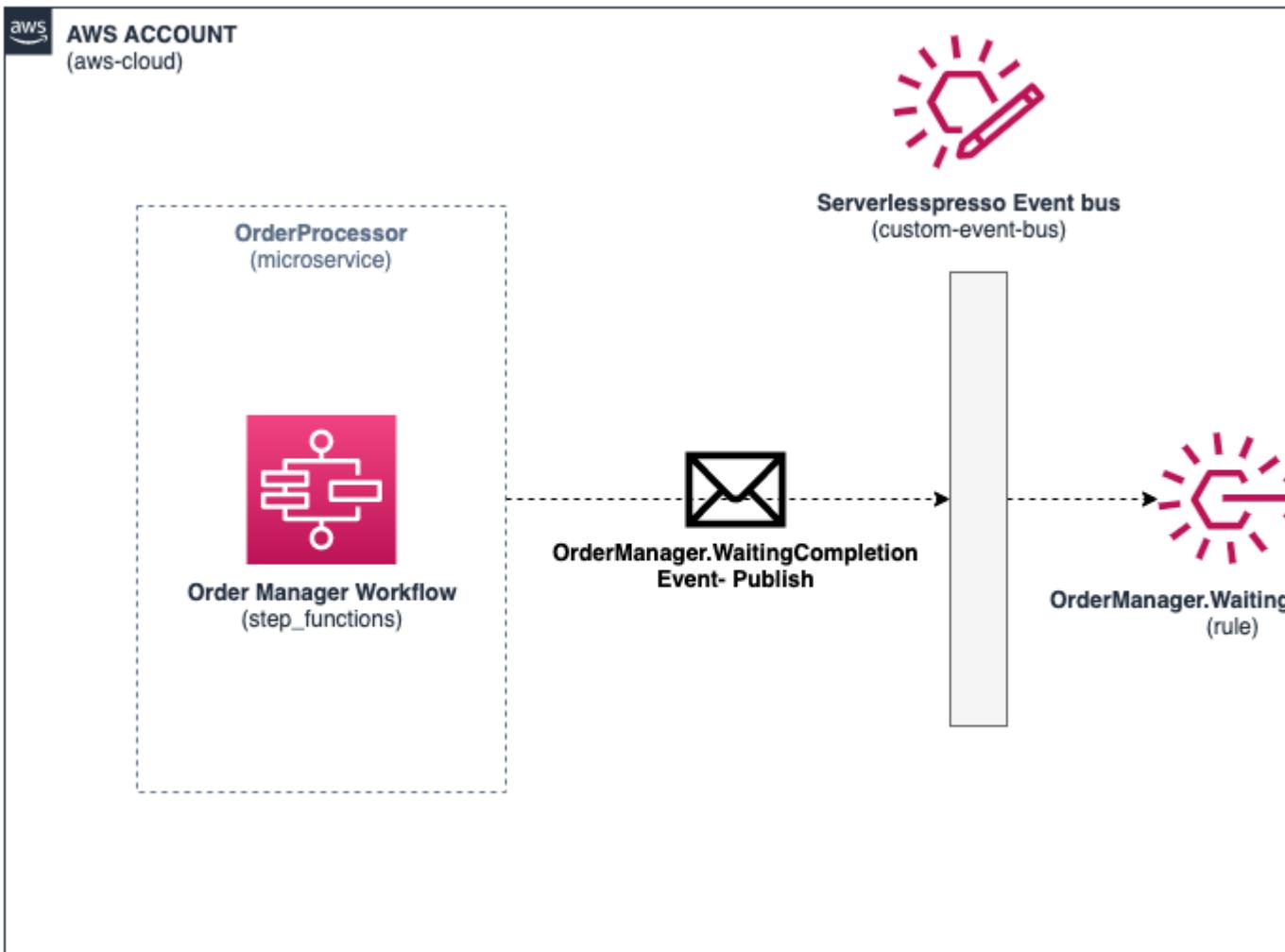
- Insira um nome para a pilha (ou mantenha o nome padrão).
- Marque as caixas na seção Capacidades.

4. Clique em Criar Pilha:

- Inicie o processo de criação da pilha.

Próximos Passos

Após a criação da pilha, você poderá configurar a regra no EventBridge e o processamento de métricas com CloudWatch. Esse fluxo permitirá monitorar efetivamente os dados de pedidos e oferecer insights valiosos aos investidores da Serverlesspresso.



Criando as regras:

Criando a Regra “WaitingCompletion” no EventBridge

Instruções Passo a Passo

1. Acesse o Console do EventBridge:

- No AWS Management Console, selecione Services e, em seguida, EventBridge sob Application Integration.

- Certifique-se de que sua região esteja correta.

2. Selecione Regras:

- Clique em Regras no painel lateral.

- Selecione Criar regra.

3. Etapa 1: Definindo a Regra:

- Nome: Insira 'OrderManagerWaitingCompletion'.

- Barramento de eventos: Digite 'Serverlesspresso'.

- Selecione Avançar.

4. Etapa 2: Origem do Evento:

- Origem do evento: Selecione Outro.
- Ignore o painel de eventos de amostra.
- No painel Padrão de evento, cole o seguinte código:

```
```json
{
 "source": ["awsserverlessda.serverlesspresso"],
 "detail-type": ["OrderManager.WaitingCompletion"]
}
```

- Escolha Avançar.

5. \*\*Etapa 3: Configuração do Destino:

- No painel Destino 1, escolha serviço AWS.
- No menu suspenso Selecionar um destino, escolha Fila SQS.
- No menu suspenso Fila, escolha MetricsQueue-Cloudwatch. (Dica: você pode começar a digitar `MetricsQueue-Cloudwatch` no campo para encontrá-la.)
- Selecione Avançar.

6. Etapa 4: Revisar Configuração:

- Escolha Avançar.

7. Etapa 5: Detalhes da Regra:

- Verifique se o painel de detalhes da regra indica que o Event bus é `Serverlesspresso`.
- Escolha Criar regra.

Conclusão

A regra `OrderManagerWaitingCompletion` agora está configurada para capturar eventos do EventBridge e enviá-los para a fila MetricsQueue-Cloudwatch. Isso permitirá coletar métricas de negócios relevantes no CloudWatch.

# Define rule detail Info

## Rule detail

Name

OrderCompleted

Maximum of 64 characters consisting of numbers, lower/upper case letters, .,-,\_.

Description - *optional*

*Enter description*

Event bus Info

Select the event bus this rule applies to, either the default event bus or a custom or partner event bus.

Serverlesspresso

i Schedule rule is not supported when custom or partner event bus is selected.

Enable the rule on the selected event bus

Rule type Info

Rule with an event pattern

A rule that runs when an event matches the defined event pattern. EventBridge sends the event to the specified target.

Schedule

A rule that runs on a schedule

```
{
 "source": ["awsserverlessda.serverlesspresso"],
 "detail-type": ["OrderManager.WaitingCompletion"]
}
```

# Select target(s)



## Permissions

Note: When using the EventBridge console, EventBridge will automatically configure the proper permissions for the selected targets. If you're using the AWS CLI, SDK, or CloudFormation, you'll need to configure permissions.

### Target 1

#### Target types

Select an EventBridge event bus, EventBridge API destination (SaaS partner), or another AWS service as a target.

- EventBridge event bus
- EventBridge API destination
- AWS service

#### Select a target | [Info](#)

Select target(s) to invoke when an event matches your event pattern or when schedule is triggered (limit of 5 targets per rule).

SQS queue

#### Queue

MetricsQueue

#### ► Additional settings

[Add another target](#)

[Cancel](#)

[Pre](#)

Testando:

Testando a Regra “OrderManager.WaitingCompletion”

Instruções Passo a Passo

Teste de Ordem Única

1. Crie um Novo Pedido:

- Realize um pedido usando o aplicativo do cliente.

## 2. Verifique o Painel do CloudWatch:

- Navegue até o CloudWatch no console da AWS e selecione Dashboards.
- Você deve ver 2 métricas no painel com valores preenchidos:
  - Total de pedidos: 1
  - Bebida Vendida: 1
  - Esses valores são incrementados sempre que há um evento `OrderManager.WaitingCompletion` no Event Bus.

## Teste de Carga

### 3. Simulando um Grande Número de Pedidos:

- Usaremos uma função Lambda para simular um grande número de pedidos e observar como o SQS atua como um buffer durante picos de tráfego.

### 4. Verifique como o Lambda Processa Mensagens:

- Cada invocação da função Lambda processará eventos em lotes de até 10 registros ou janelas de 30 segundos.
  - A Concorrência Reservada define quantas execuções simultâneas do Lambda podem ler mensagens da fila; o padrão é até 1000 execuções simultâneas, mas você irá configurar para apenas 1 execução simultânea.

### 5. Ajuste as Configurações se Necessário:

- Os valores ReservedConcurrentExecutions, BatchSize e MaximumBatchingWindowInSeconds podem ser personalizados para ajustar o processamento.

### 6. Verifique o Painel do CloudWatch:

- Navegue até o painel Serverlesspresso\_Metrics para ver as métricas enquanto elas são atualizadas durante o teste de carga.
- Você deve ver quatro métricas: contagens para cada tipo de item e uma contagem total de pedidos.

### 7. Acesse o Console do Lambda:

- Procure pela função `EventsLoadTest` no console do Lambda.

### 8. Invocar a Função Lambda:

- Clique no botão Testar para simular eventos `OrderManager.WaitingCompletion` no Event Bus.
- Você será solicitado a criar um evento de teste:
  - Nome do Evento: Insira 'Teste'.
  - JSON do Evento: Use o valor padrão.
- Clique em Salvar e depois clique novamente no botão Testar para executar o teste de carga.

## Visualizando Resultados do Teste de Carga

### 9. Acesse a Função `PublishMetrics`:

- Navegue até a função `PublishMetrics` e vá até a aba Monitoring.
- Expanda o gráfico Invocações da função e selecione “Máximo” em vez de “Soma”.
- Você verá que apenas 1 execução simultânea ocorreu durante o teste de carga, devido à configuração de Concorrência Reservada.

### 10. Verifique o Painel `Serverlesspresso\_Metrics`:

- Você verá que cerca de 500 pedidos totais foram simulados durante o teste de carga.

## Takeaways

- Conseguimos fornecer métricas de negócios para os investidores configurando uma nova regra e microsserviço sem precisar modificar a pilha de aplicativos existente.
- Use as métricas do CloudWatch para métricas de negócios personalizadas e habilitar a observabilidade no desempenho do seu aplicativo orientado a eventos.

CloudWatch > Dashboards > Serverlesspress

## Serverlesspresso\_Metrics ▾ ☆

Drink ⌂

308

■ Cappuccino

387

■ Latte

■ E

■ Tot

## Functions (20)

 Filter by tags and attributes or search by keyword

eventsloadtest 

**Clear filters**



Function name 



sqs-module-EventsLoadTest-  
Xxy8DRle5xg5

Code source [Info](#)

File Edit Find View Go Tools Window Test Deploy

Go to Anything (⌘ P)

Environment

sqS-module-Events1

- EventsLoadTest.js
- PublishMetrics.js

EventsLoadTest.js

```
// Import required AWS SDK clients and command
const AWS = require("aws-sdk");
var eventbridge = new AWS.EventBridge({apiVersion: "2018-11-01"});

exports.handler = async (event) => {
 // Get drinks
 const drinks = ["Latte", "Cappuccino", "Espresso"];
 let batchSize = 10;
 let entries = [];

 for (let i = 1000; i < 1500; i+=batchSize) {
 var entry = {
 "Source": "LatteMachine" + i,
 "DetailType": "LatteOrder"
 };
 entries.push(entry);
 }

 await eventbridge.putEvents({entries}).promise();
}
```

## Test event action

Create new event

Edit saved event

### Event name

Test

Maximum of 25 characters consisting of letters, numbers, dots, hyphens and underscores.

### Event sharing settings

Private

This event is only available in the Lambda console and to the event creator. You can configure a trigger for this event to invoke a Lambda function.

Shareable

This event is available to IAM users within the same account who have permissions to access and invoke the Lambda function.

### Template - *optional*

hello-world

## Event JSON

```
1 - [{}
2 "key1": "value1",
3 "key2": "value2",
4 "key3": "value3"
5]
```

Fluxo de trabalho do processador de pedidos – TESTE:

1. Testando o Fluxo de Trabalho com Pedidos em Excesso

Embora o Step Functions possa escalar para dezenas de milhares de execuções, a cafeteria é configurada para lidar apenas com até 20 pedidos simultâneos. O fluxo de trabalho rejeita novos pedidos até que haja menos de 20 pedidos. Você testará isso adicionando 21 pedidos.

Instruções Passo a Passo

1. Acesse o Console do Step Functions:

- No AWS Management Console, selecione Services e depois Step Functions em Application Integration.

- Certifique-se de que sua região esteja correta.

## 2. Selecione a Máquina de Estado:

- No menu à esquerda, selecione State machines e escolha `OrderProcessorWorkflow` na lista.

- Copie o valor ARN para um local temporário, pois você precisará dele mais tarde.

- Clique em Edit.

## 3. Inicie o Fluxo de Trabalho:

- Use a API StartExecution para iniciar um novo fluxo de trabalho. No painel do terminal do Cloud9, insira o seguinte comando, substituindo `YOUR\_STATE\_MACHINE\_ARN` pelo ARN copiado na etapa anterior:

```
```bash
aws stepfunctions start-execution --state-machine-arn YOUR_STATE_MACHINE_ARN --input
"\"detail\":{\"orderId\":\"1\",\"userId\":\"testuser\"}"
```
```

```

4. Repita o Comando:

- Execute o mesmo comando mais 25 vezes. A capacidade configurada para a loja é 20, portanto, após a 20^a execução, a verificação de capacidade no fluxo de trabalho falhará para solicitações subsequentes.

5. Verifique o Caminho do Fluxo de Trabalho:

- Na exibição da máquina de estado no console, o inspetor de gráfico mostrará o caminho do fluxo de trabalho tomado, indicando que a capacidade estava indisponível.

2. Testando Pedidos com Tempo Limite Esgotado

Ao criar o fluxo de trabalho, foram adicionadas duas transições que aguardam retornos de chamada: uma para o cliente enviar os detalhes do pedido e outra para o barista preparar as bebidas. O cliente tem 5 minutos para concluir sua etapa, enquanto o barista tem 15 minutos.

Instruções Passo a Passo

1. Acesse o Console do Step Functions:

- Novamente, vá para o AWS Management Console e selecione Step Functions em Application Integration.

2. Selecione a Máquina de Estado:

- No menu à esquerda, selecione State machines e escolha `OrderProcessorWorkflow` na lista.

3. Aguarde o Tempo Limite:

- Após 5 minutos desde que você iniciou a lista de execução, as execuções que estavam em andamento agora estarão em um estado de Failed.

4. Verifique as Execuções com Falha:

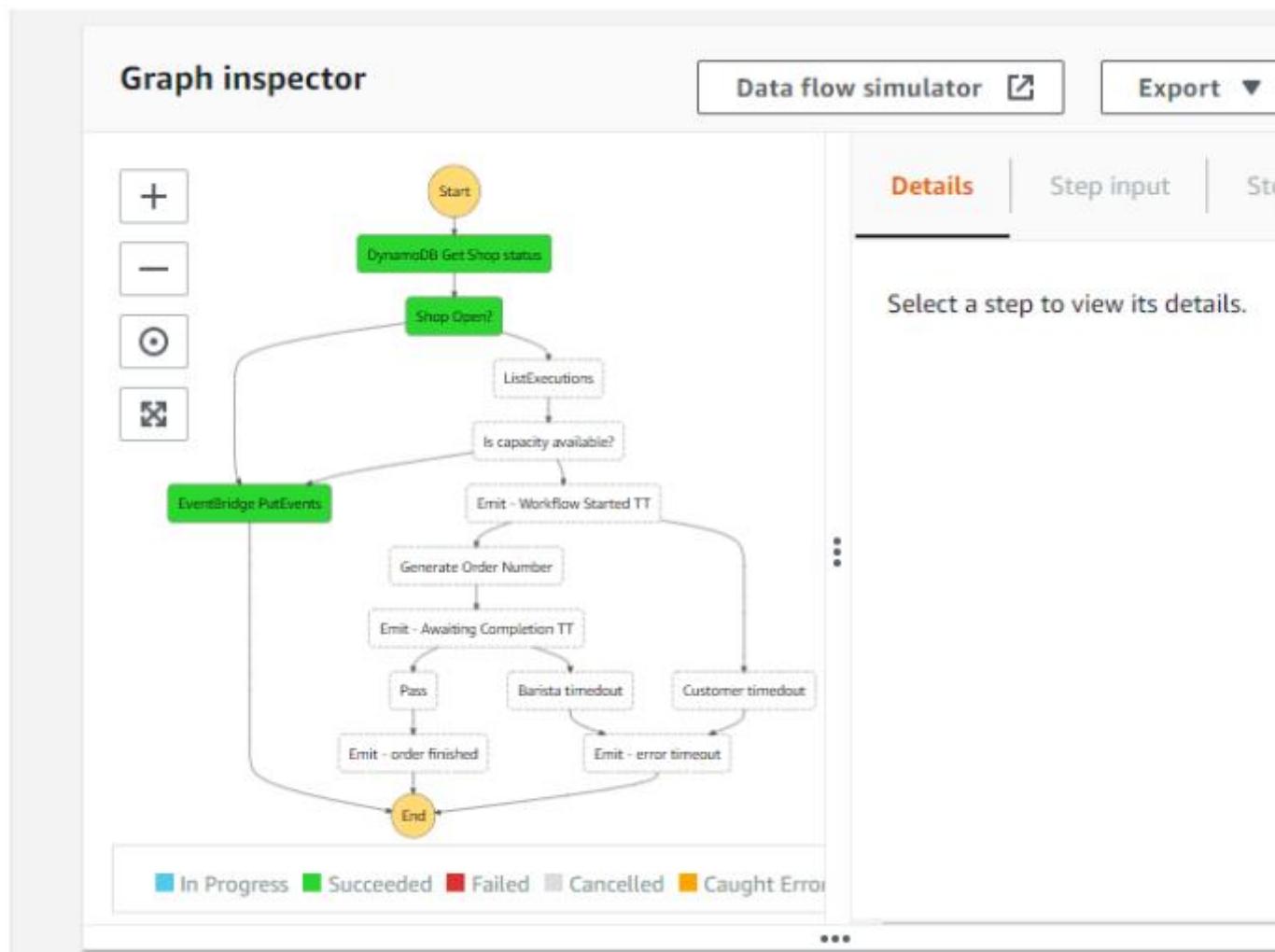
- Selecione a primeira execução com falha na lista. O Graph inspector mostrará a transição de estado `Emit - Workflow Started TT` em laranja, direcionando para o estado `Customer timeout`.

- Selecione a aba Step output para ver o erro e a causa da falha.

Observações Finais

- A simulação de pedidos em excesso ajuda a verificar como o fluxo de trabalho lida com a capacidade limitada.
- O teste de tempo limite permite avaliar a robustez do sistema e como ele responde a condições adversas, como a falta de resposta do cliente ou do barista.

```
aws stepfunctions start-execution --state-machine-arn YOUR_STATE_MACHINE_ARN --input '{"orderId": "1", "userId": "testuser"}'
```



OrderProcessorWorkflow

[Edit](#)[Start execution](#)[Delete](#)

Details

ARN

arn:aws:states:us-east-1:XXXXXXXXXX:stateMachine:OrderProcessorWorkflow

Type

Standard

IAM role ARN

arn:aws:iam::XXXXXXXXXX:role/serverlesspresso-backend-test-OrderProcessorRole-1B3OGY79A6RSI 

Creation date

Mar 7, 2022 08:03:18.270 AM

[Executions](#)

[Logging](#)

[Definition](#)

[Tags](#)

Executions (61)



[View details](#)

[Stop execution](#)

[St](#)

[Filter by status](#)



<

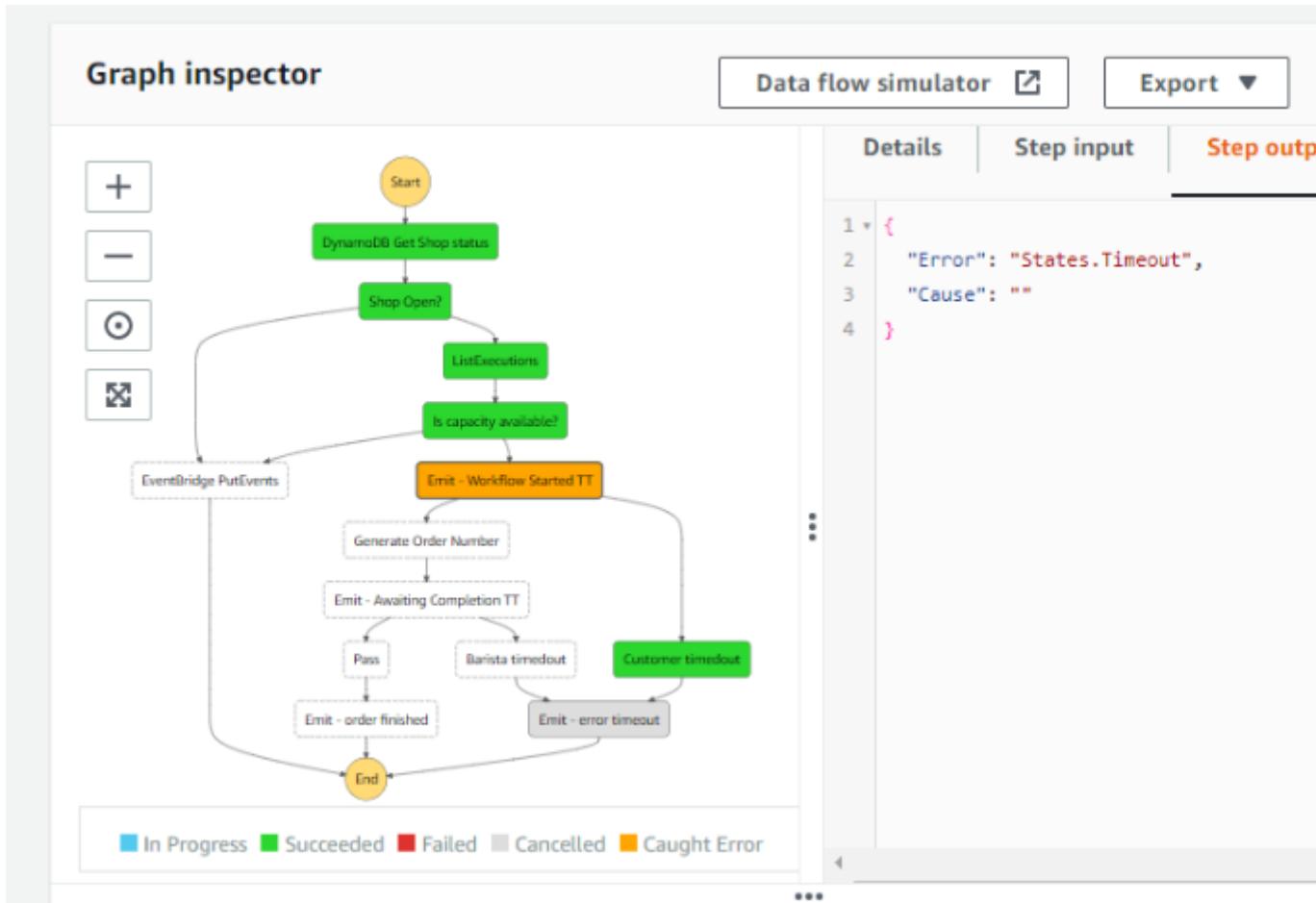
Name

Status

Started

End Time

	615789a4-9af3-4b85-b957-f8634a08bd31	Failed	Mar 7, 2022 10:47:52.720 AM	Mar 7, 2022 10:52:53
	b4c72136-0516-4a72-85b2-4fdb2cfcd459	Failed	Mar 7, 2022 10:47:51.302 AM	Mar 7, 2022 10:52:52
	b9b05482-5055-4d14-9cb9-677a0318174a	Failed	Mar 7, 2022 10:47:49.872 AM	Mar 7, 2022 10:52:50
	0b304021-5a93-4136-b8ee-6cbaf80101d8	Failed	Mar 7, 2022 10:47:48.359 AM	Mar 7, 2022 10:52:49
	3222ad37-cfe6-4633-a7ab-f8d5201861f7	Failed	Mar 7, 2022 10:47:46.895 AM	Mar 7, 2022 10:52:47



Limpar:

Instruções para Limpar Recursos do Workshop

Limpando Recursos

1. Buckets S3

- No Cloud9, para listar os buckets usados neste workshop, insira o seguinte comando:

```
```bash
```

```
aws s3 ls | grep serverlesspresso
```

```
...
```

- Para excluir cada bucket e seu conteúdo, substitua `your-bucket-name` pelo nome de cada bucket:

```
```bash
```

```
aws s3 rb --force s3://your-bucket-name
```

```
...
```

2. Recursos no CloudFormation

- No Cloud9, obtenha uma lista de pilhas usadas neste workshop:

```
```bash
aws cloudformation list-stacks | grep serverlesspresso
```
```

```

- Para excluir cada pilha que começa com `serverlesspresso`, substitua `your-stack-name` pelo nome de cada pilha:

```
```bash
aws cloudformation delete-stack --stack-name your-stack-name
```
```

```

3. Regras do EventBridge

- No Cloud9, obtenha uma lista de regras do EventBridge usadas neste workshop:

```
```bash
aws events list-rules --event-bus-name Serverlesspresso
```
```

```

- Para excluir cada regra, substitua `your-rule-name` pelo nome da regra:

```
```bash
aws events delete-rule --name 'your-rule-name'
```
```

```

4. Excluir a Instância do Cloud9

- No console do Cloud9, selecione sua instância e escolha Excluir . Isso excluirá todos os dados do workshop da instância e interromperá o faturamento.

Considerações Finais

Após seguir esses passos, todos os recursos criados durante o workshop serão removidos, ajudando a evitar cobranças futuras indesejadas.

```
aws s3 ls | grep serverlesspresso
```

2. Exclua cada bucket e seu conteúdo, substituindo `your-bucket-name` pelo nome de cada bucket:

```
aws s3 rb --force s3://your-bucket-name
```

2. Recursos no CloudFormation

1. No Cloud9, obtenha uma lista de pilhas usadas neste workshop:

```
aws cloudformation list-stacks | grep serverlesspresso
```

2. Exclua cada pilha que começa com `serverlesspresso`, substituindo `your-stack-name` pelo nome de cada pilha:

```
aws cloudformation delete-stack --stack-name your-stack-name
```

3. Regras do EventBridge

1. No Cloud9, obtenha uma lista de regras do EventBridge usadas neste workshop:

```
aws events list-rules --event-bus-name Serverlesspresso
```

2. Exclua cada regra, substituindo `your-rule-name` pelo nome da regra:

```
aws events delete-rule --name 'your-rule-name'
```