
AWS
re:Invent
NOV. 28 – DEC. 2, 2022 | LAS VEGAS, NV

SVS001

Inside Serverlesspresso: Creating a production workload for pennies a day

James Beswick

Sr. Manager, Developer Advocacy
AWS Serverless

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.



serverlesspresso

An event-driven coffee-ordering app
built with serverless architecture

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

The frontend applications

VUE.JS APPS HOSTED WITH AWS AMPLIFY CONSOLE

1. Ordering app:

- Loads menu
- Scans barcode to get order token
- Sends order
- Displays updates

2. TV display app:

- Shows current barcode
- Receives order status updates
- Listens for store open/close events

3. Barista app:

- Shows incoming orders
- Enables order completion or cancellation
- Enables store open/close events

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

What is Serverlesspresso?



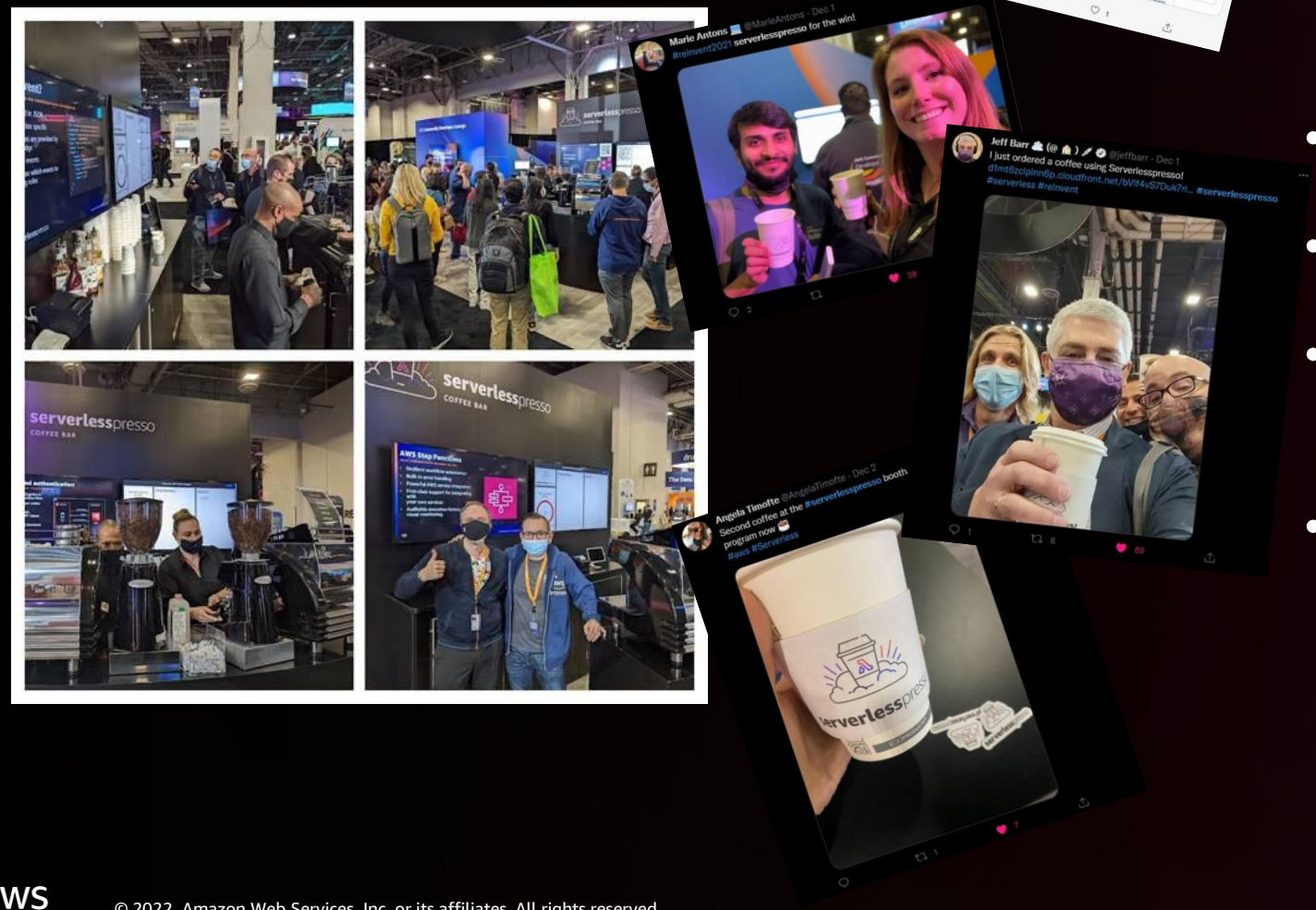
Serverlesspresso is a coffee-ordering app built with serverless architecture.

1. Place an order on your mobile device.
2. The order appears on the monitor and the barista's tablet app.
3. You get a notification when the drink is ready.

serverlesspresso

Learn more at:
https://s12d.com/espresso

serverlesspresso



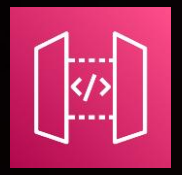
- 1,920 drinks in re:Invent 2021
- 71 drinks per hour
- Appeared at AWS Summits, EDA Day, GOTO, and others
- Averages 1,000 drinks per day

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS Serverless services used



AWS Amplify Console



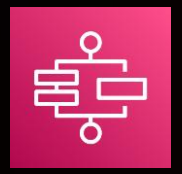
Amazon API Gateway



Amazon DynamoDB



Amazon EventBridge



AWS Step Functions



AWS IoT Core



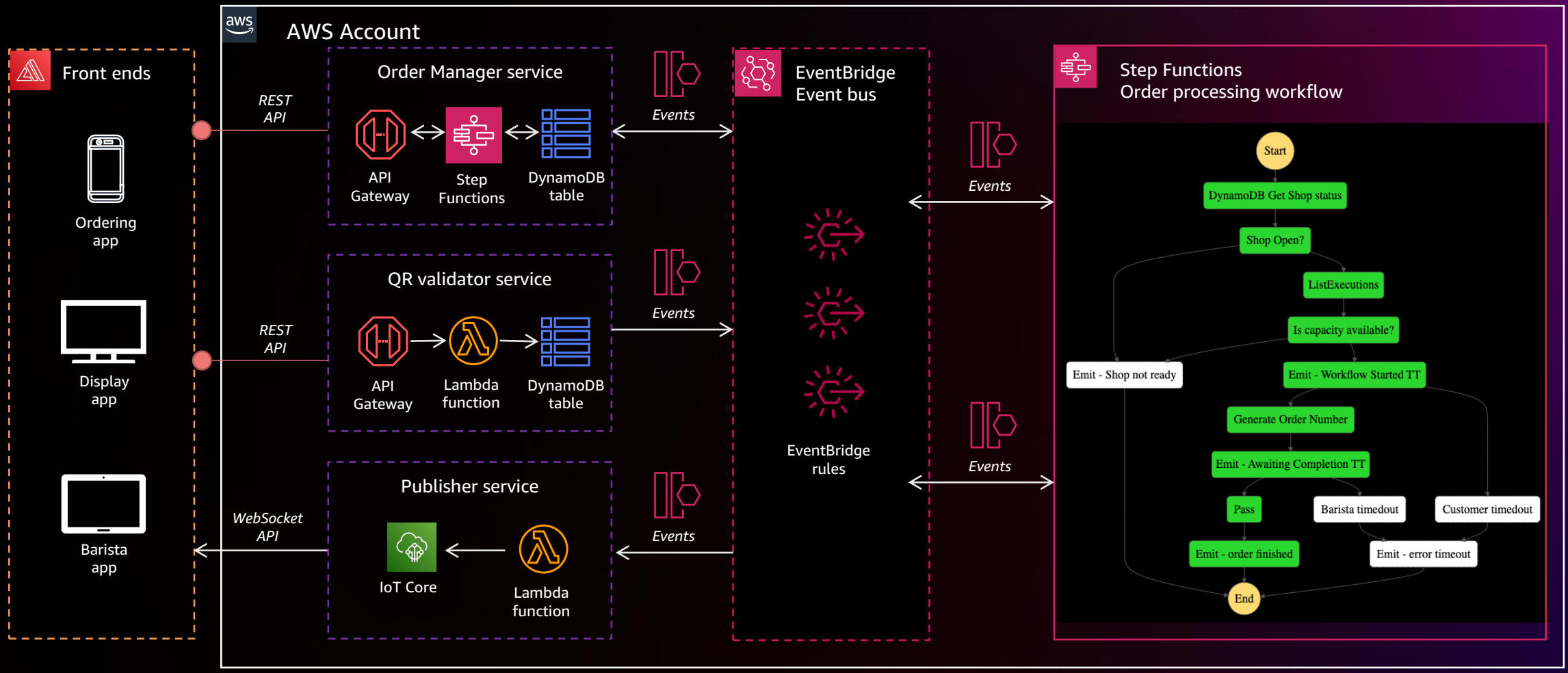
AWS Lambda

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

High-level architecture

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

How it works



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

QR validator service

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

QR code generation

- A new QR code every 5 minutes
- Valid for 10 scans
- Hidden once all scans are “used”



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

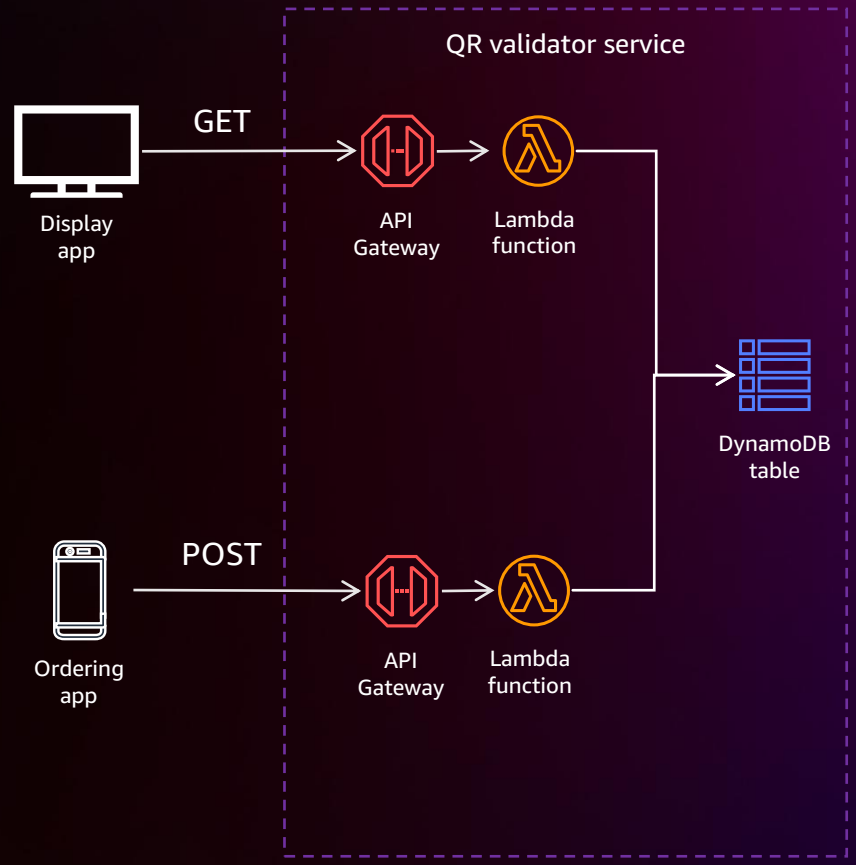
QR code generation

Generate

- The display app makes a request to generate the QR code and store the ID in DynamoDB

Validate

- The ordering app makes a POST request to validate the QR code and ID



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Token bucket system

When a valid QR is scanned, it decrements the *Available tokens* number and emits an event that triggers the order processor workflow.

PK	Available tokens	End_ts	Last_code	Last_id	Start_ts
1234423	10	1645200599999	41g_-KJHGT	1234423	1645200300000
1234123	8	1645628399999	6KJH_-FJ5Lh	1234123	1645628100000
5412322	1	1645449599999	91HHFFJHF	5412322	1645449300000
3435657	2	1645435199999	OCZomT756	3435657	1645434900000

DynamoDB validator table

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

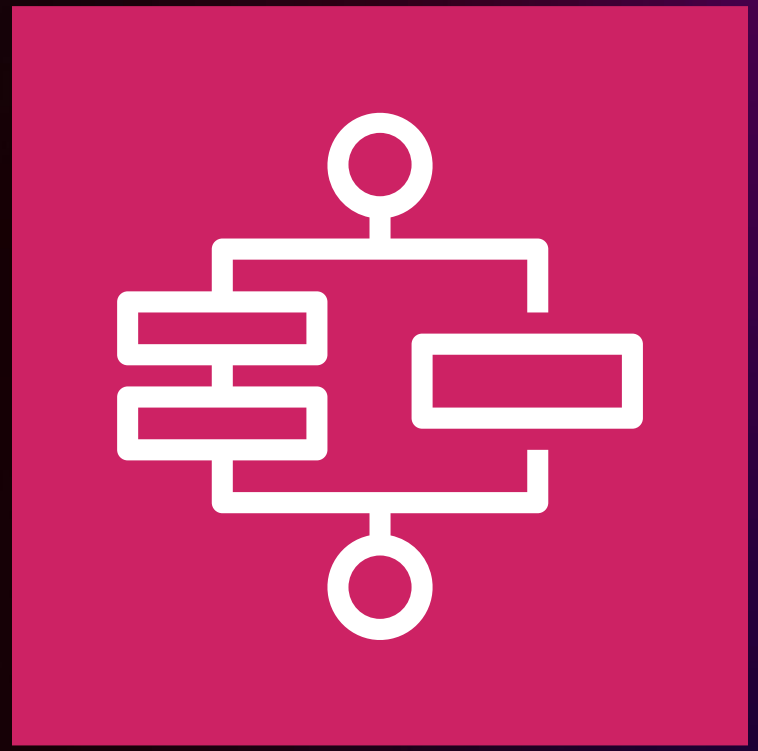
Order processor service

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS Step Functions

FULLY MANAGED STATE MACHINES ON AWS

- Resilient workflow automation
- Built-in error handling
- Powerful AWS service integration
- First-class support for integrating with your own services
- Auditable execution history and visual monitoring



aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Define the workflow...

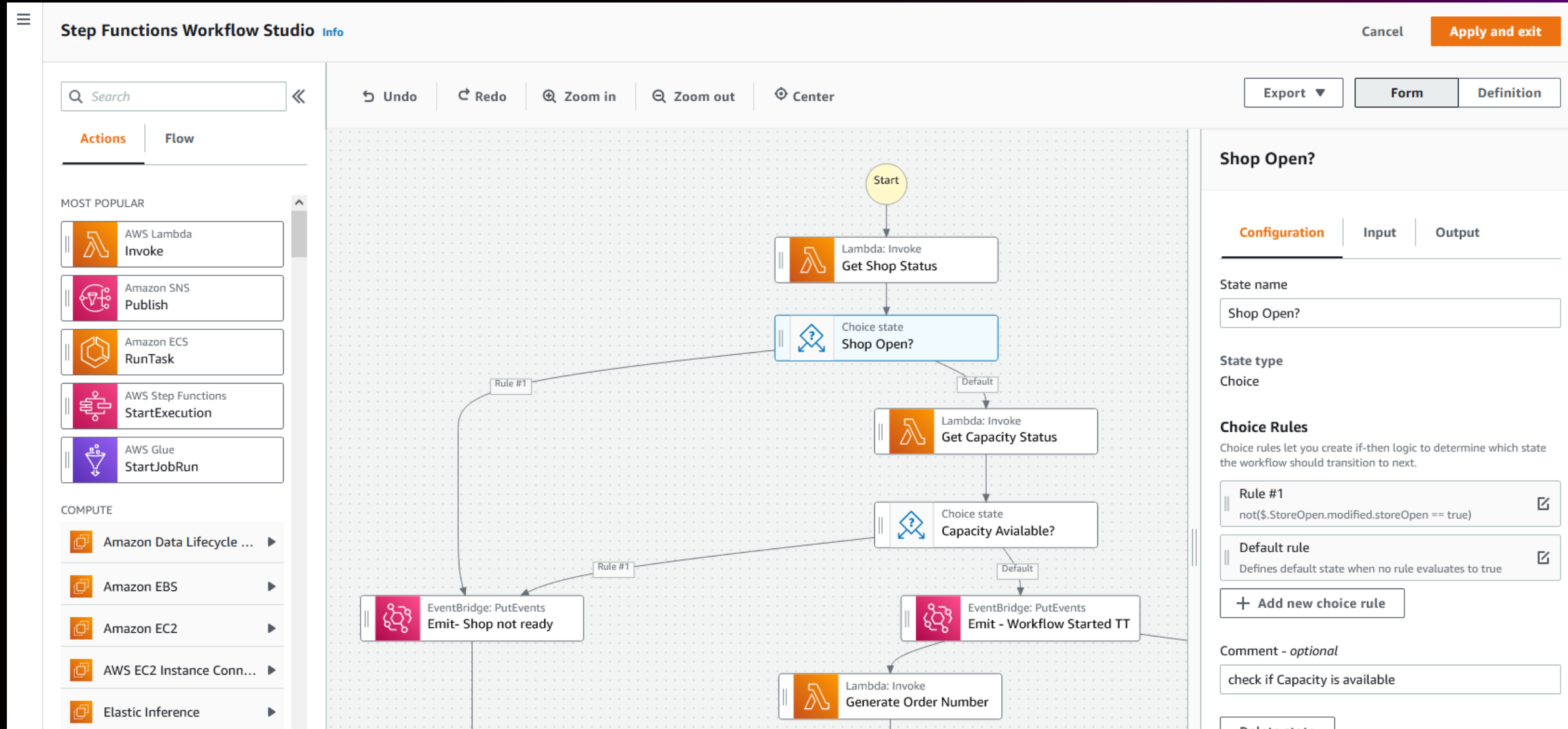
Process for making a drink

1) Check the store is open
2) Get barista capacity
3) Wait for the customer order; cancel if > 5 minutes
4) Generate an order number
5) Wait for barista to make drink; cancel if > 15 minutes
6) Also handle cancelation by customer or barista



… then design visually with Workflow Studio

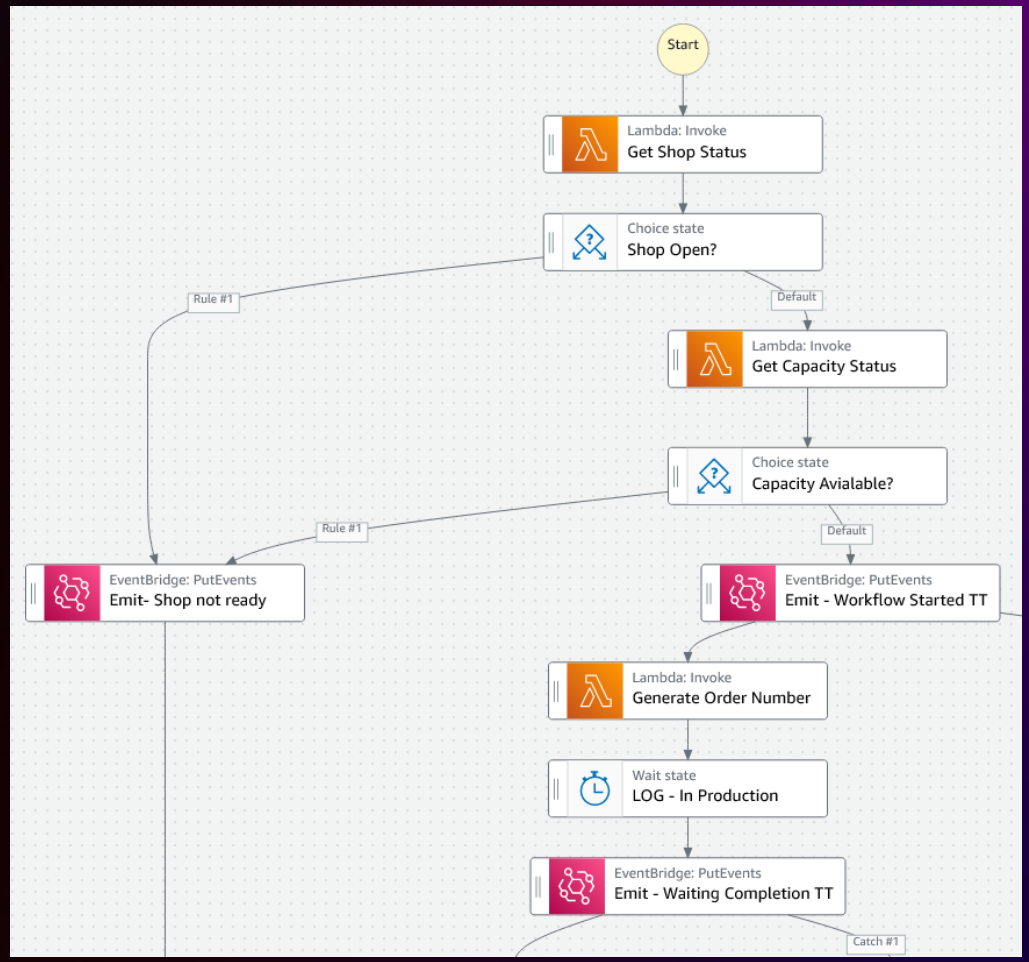
REPLACE SPAGHETTI CODE WITH STATE MACHINES



Managing each coffee's journey

USING AWS STEP FUNCTIONS TO MANAGE WORKFLOWS

- Workflow ensures store is open and baristas have capacity
- Allows customer 5 minutes to order before timing out or cancelling
- Allows barista 15 minutes to make before timing out or cancelling
- Uses Lambda functions for custom logic; uses direct service integrations otherwise



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Order Manager service

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Order table

- Each order is persisted to a DynamoDB table.
- The entry is updated at various stages of the order lifecycle.

PK	SK	TS	UID	OrderNo	TaskToken	OrderState	DrinkOrder
Orders	2	1645726346199	1	10	AAAAKgAAAAIAAAAAAAAAAAlp4um0gPw/FO3rqpCDvlEAL+l/h+	COMPLETED	{"userId":"1","drink":"Cappuccino","modifiers":[],"icon":"barista-icons_cappuccino-alternative"}

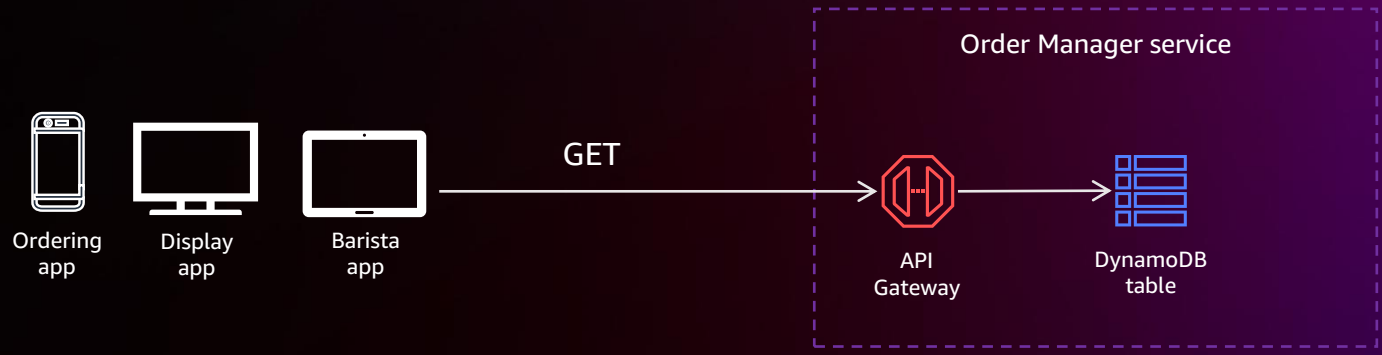
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

API Gateway to DynamoDB

/myOrders - GET
/orders - GET
/orders/{id} - GET

Frontend applications query *Order* table directly from API Gateway.

Velocity mapping templates modify the incoming request.




```
#set($subFromJWT = $context.authorizer.claims.sub)

{
    "TableName": "serverlesspresso-order-table",
    "IndexName": "GSI-userId",
    "KeyConditionExpression": "#USERID = :USERID",
    "ExpressionAttributeNames": {
    "#USERID": "USERID"
    },
    "ExpressionAttributeValues": {
    ":USERID": {
      "S": "$subFromJWT"
    }
    },
    "ScanIndexForward": true,
    "ProjectionExpression": "PK, SK, orderNumber, robot, drinkOrder, ORDERSTATE, TS"
}
```

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

CRUD operations

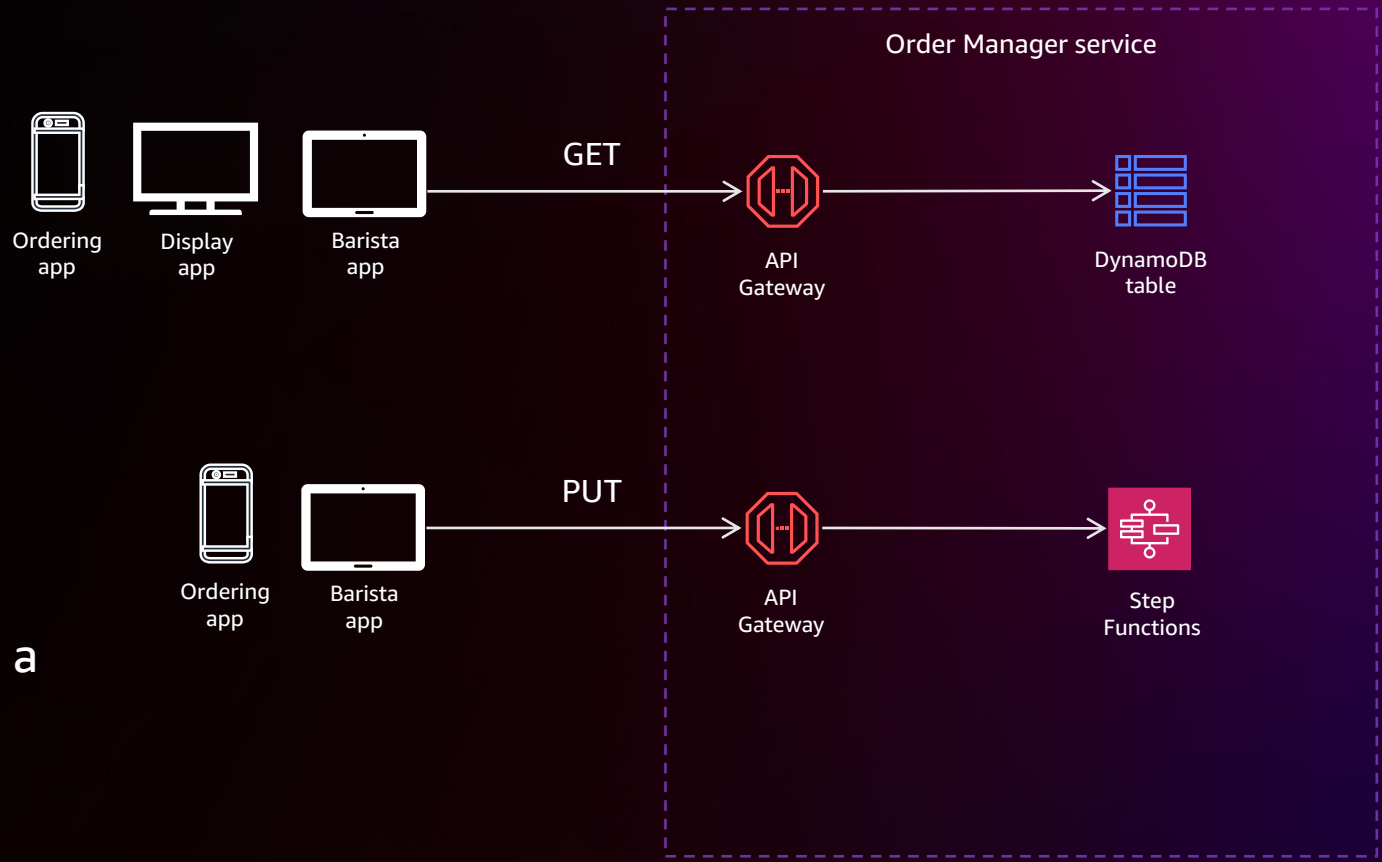
/myOrders - GET

/orders - GET

/orders/{id} - GET

/orders/{id} - PUT

Updates are made to the *Order* table via a PUT request.

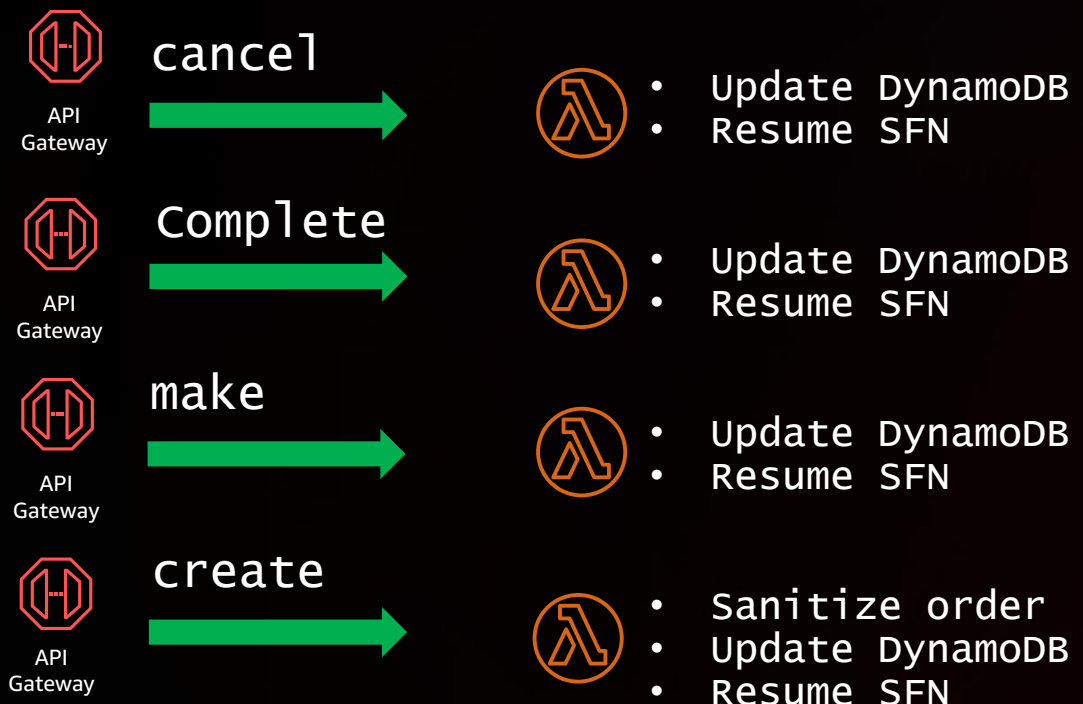


© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Order Manager service as a Lambda function

Version 1

Each operation invokes a Lambda function



The application grew more complex over time, performing multiple tasks to handle increasingly complex business logic.

This leads to

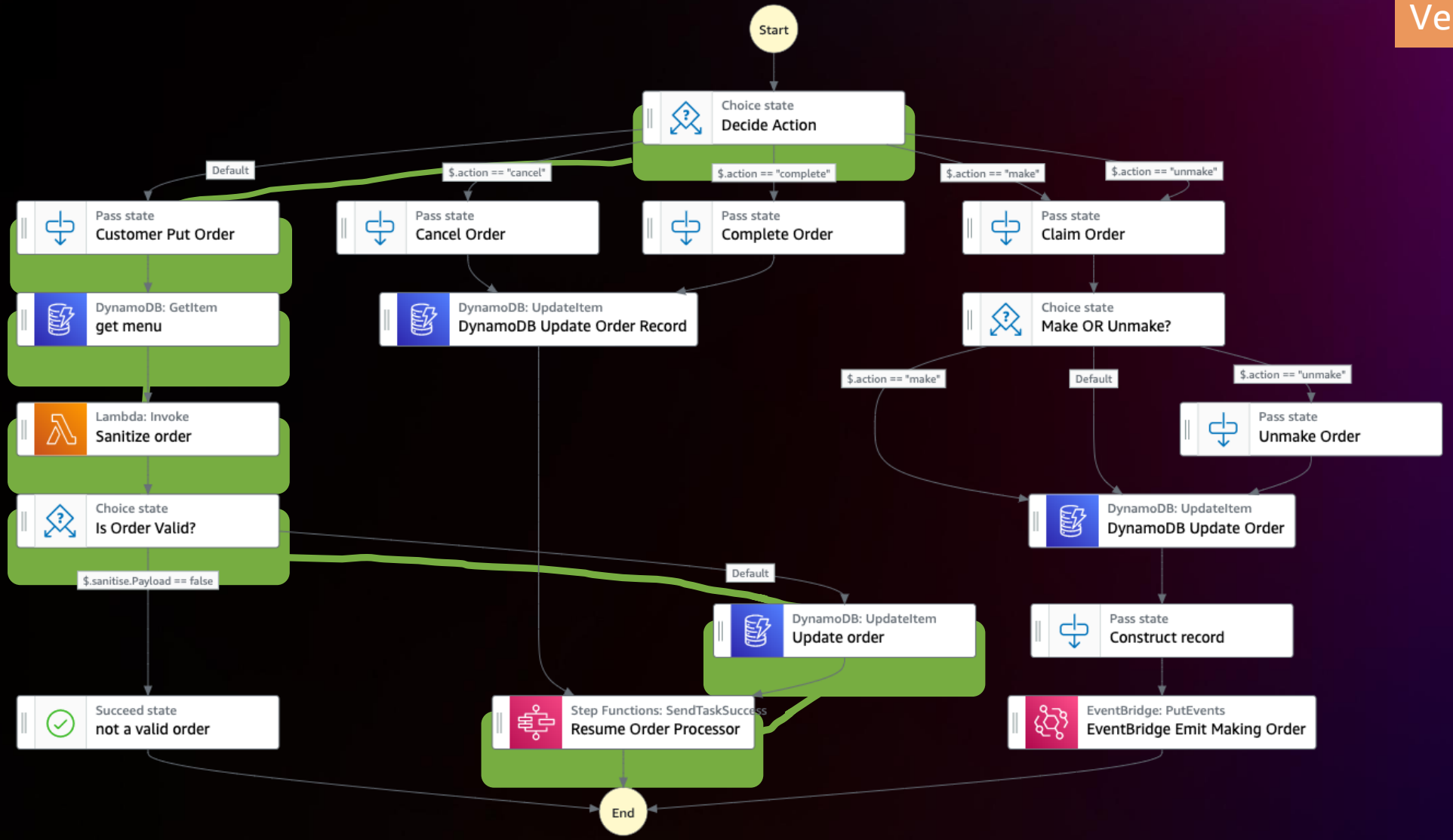
- Tightly coupled code base
- Slower release cadence
- Poor discoverability
- Additional complexity

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

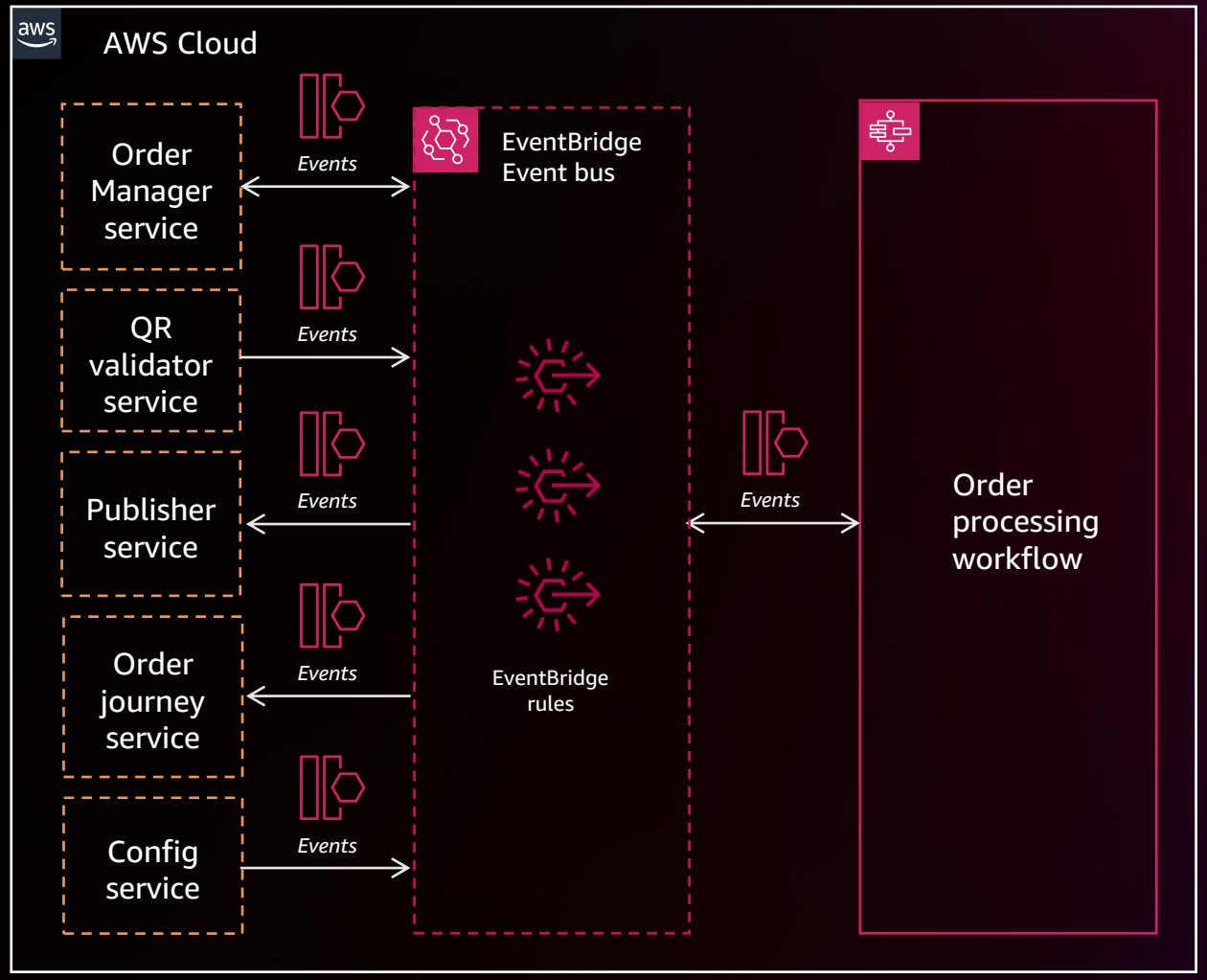
Order Manager service as a workflow

Version 2



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Communicate between microservices using events



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

What is an event?

- An event is defined in JSON
- “Detail” is application specific
- Envelope attributes are provided by Amazon EventBridge
- Producers create events
- Consumers choose which events to listen to by using rules


```
{
  "version": "0",
  "id": "6ac4e27b-1234-1234-1234-5fb02c880319",
  "detail-type": "OrderProcessor.OrderStarted",
  "source": "awsserverlessda.serverlesspresso",
  "account": "123456789012",
  "time": "2021-11-28T13:12:30Z",
  "region": "us-west-2",
  "detail": {
      "userId": "jbesw",
      "orderId": "eYmAfqLD67vlbdUViLe_D",
      "drinkOrder": {
        "icon": "barista-icons_cafe-latte",
        "modifiers": [],
        "drink": "Latte"
    }
  }
}
```


© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.



Amazon EventBridge

A serverless event bus service for AWS services, your own applications, and SaaS providers

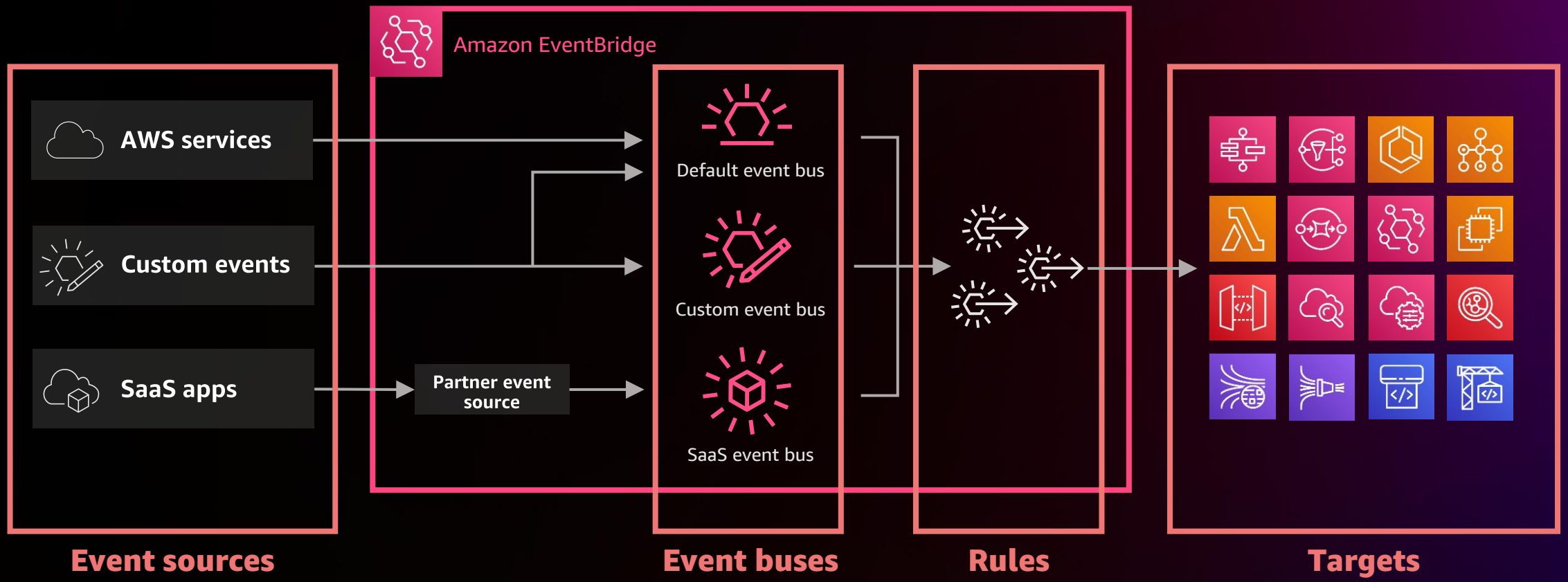
- Serverless—pay only for the events you process
- Simplified scaling avoids increasing costs to sustain and manage resources
- No upfront investments, ongoing licensing, or maintenance costs
- No specialist knowledge needed

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Amazon EventBridge architecture

ROUTING EVENTS BETWEEN MICROSERVICES

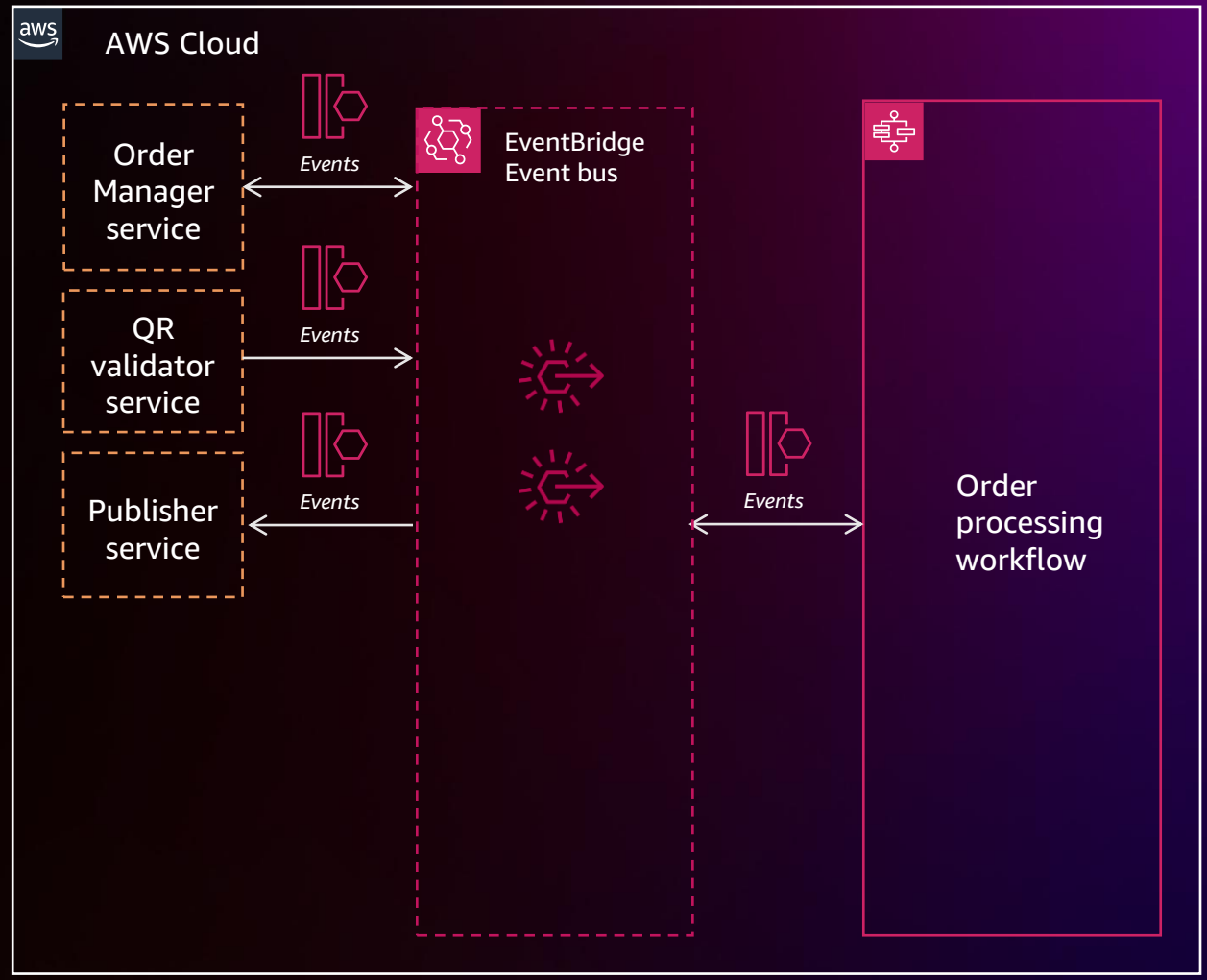


© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Event-driven architectures

CHOREOGRAPHING MICROSERVICES

- Event flow drives the application
- Events choreograph the services, while Step Functions orchestrates the transactions

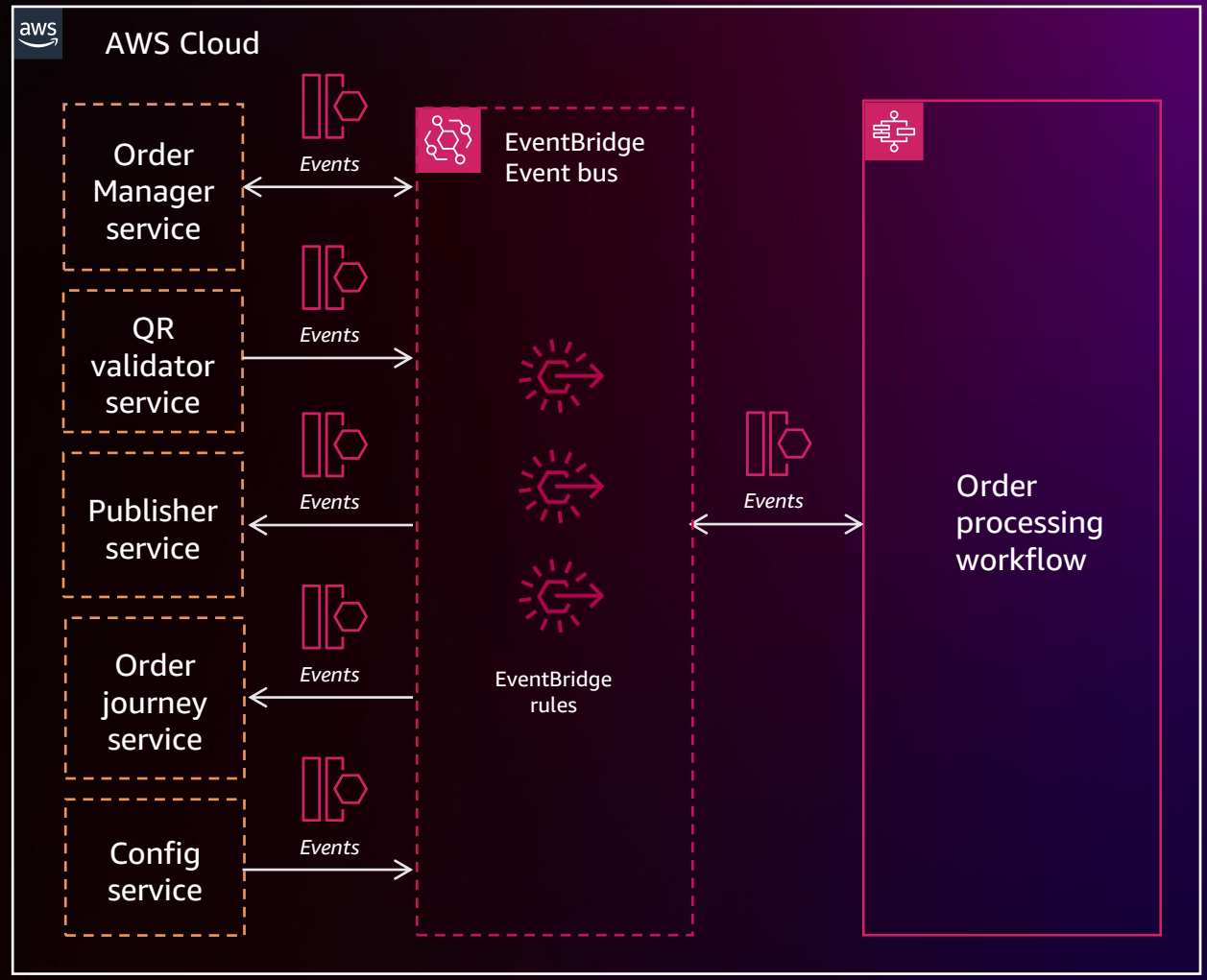


© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Event-driven architectures

CHOREOGRAPHING MICROSERVICES

- Event flow drives the application
- Events choreograph the services, while Step Functions orchestrates the transactions
- Add new microservices as event consumers without changing existing code
- Microservices emit events independently of consumers



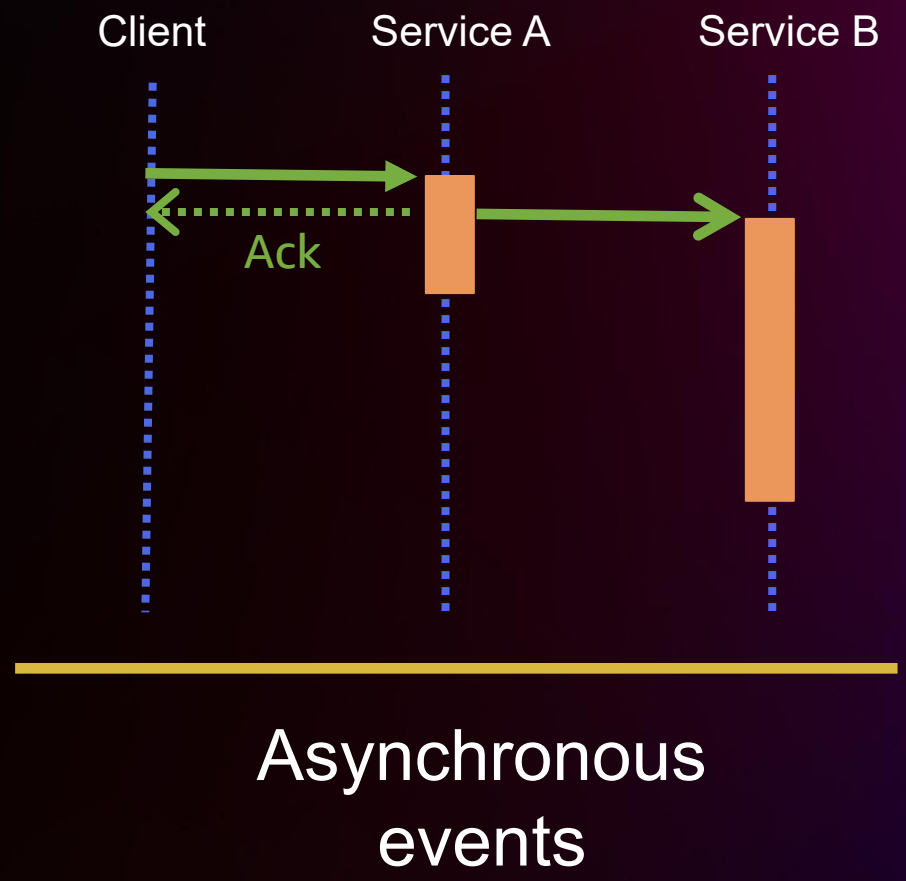
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Handing async responses with real-time updates

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

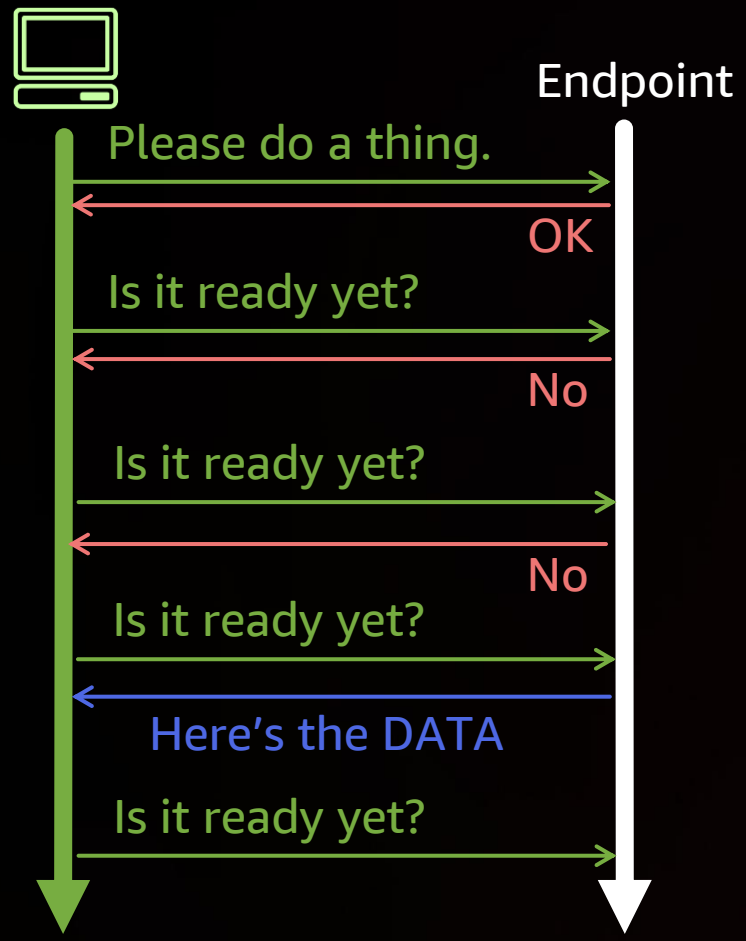
Handling response values and state for asynchronous requests

No return path to provide further information beyond the initial acknowledgment



aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

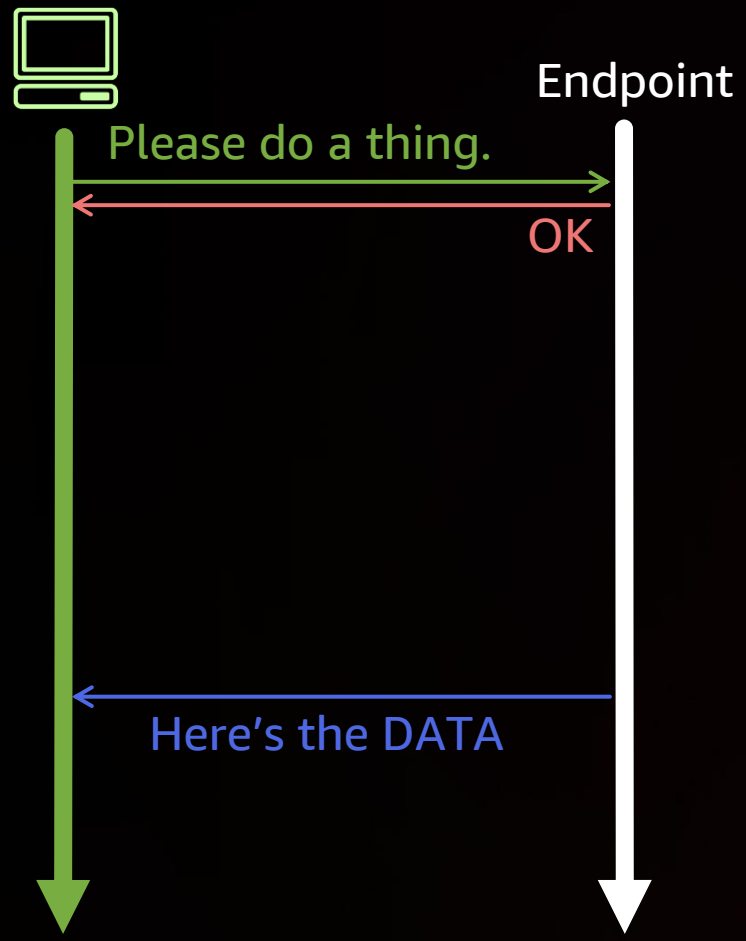
Tracking an inflight request: Polling



- Initial request returns a tracking identifier
- Create a second API endpoint for the frontend to check the status of the request, referencing the tracking ID
- Use DynamoDB or another data store to track the state of the request
- Simple mechanism to implement
- Can create many empty calls
- Delay between availability and frontend notification

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Tracking an inflight request: WebSocket

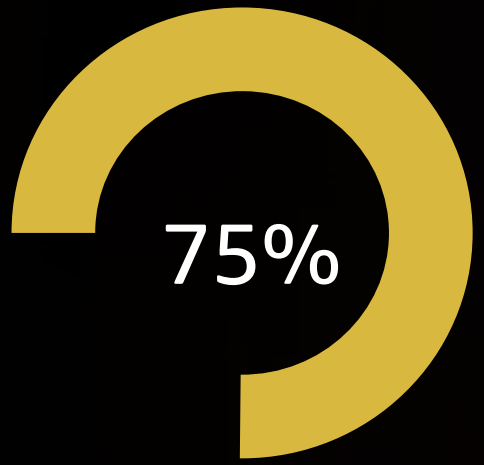


- A bidirectional connection between the frontend client and the backend service
- Your backend services can continue to send data back to the client by using a WebSocket connection
- Closer to real time
- Reduces the number of messages between the client and backend system
- Often more complex to implement

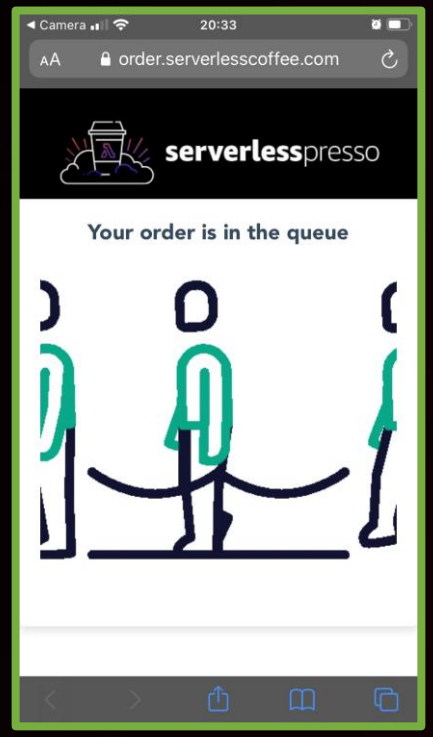
aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Using AWS IoT Core for real-time messaging

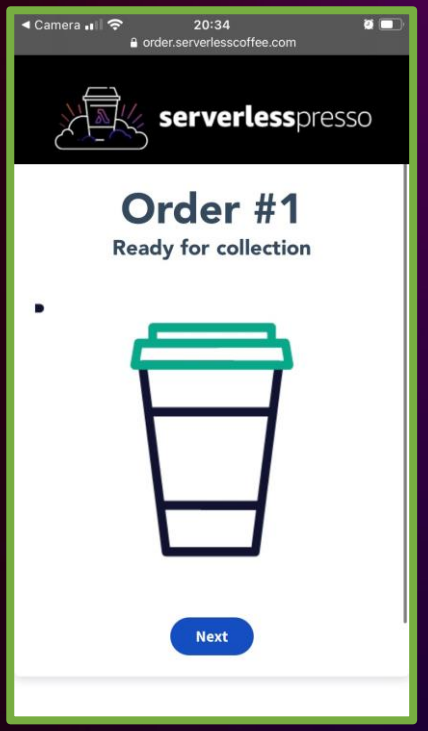
Web applications often require partial information:



Completion percentages





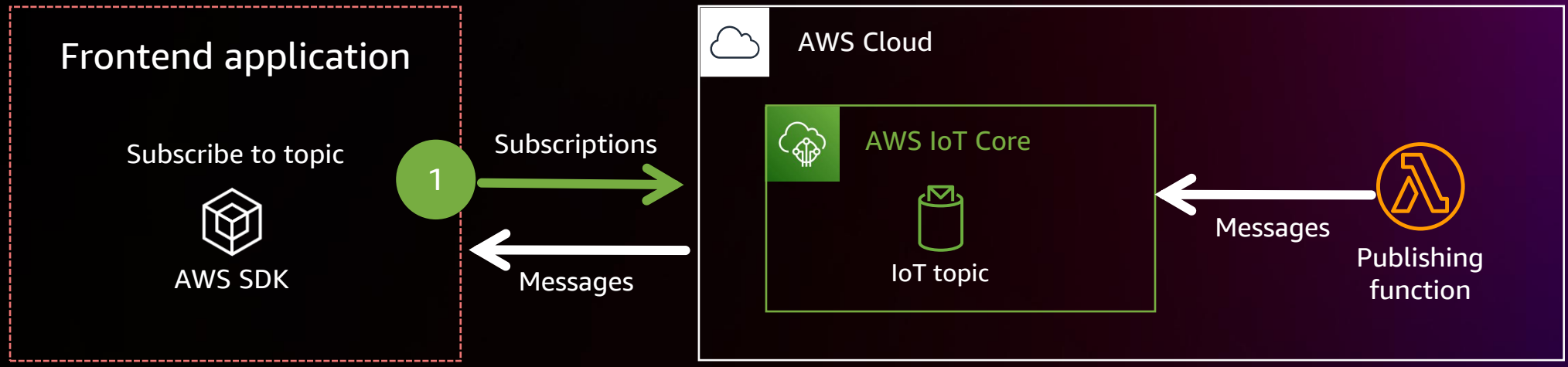


Continuous data changes

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Using AWS IoT Core for real-time messaging

Frontend application uses pub-sub to “listen” for event updates

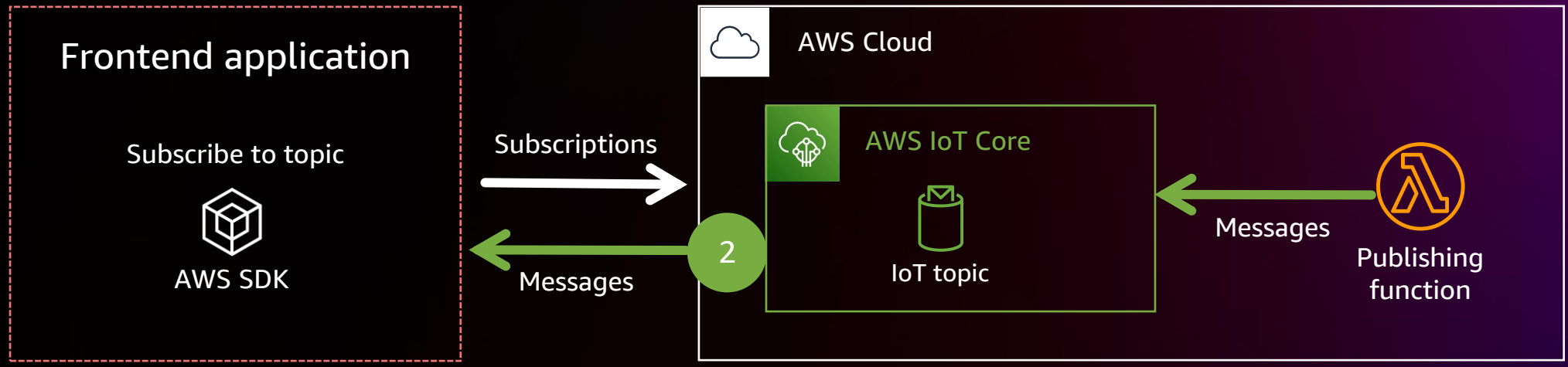


Frontend uses AWS SDK to subscribe to a topic based on user’s unique user ID.

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Using AWS IoT Core for real-time messaging

Frontend application uses pub-sub to “listen” for event updates

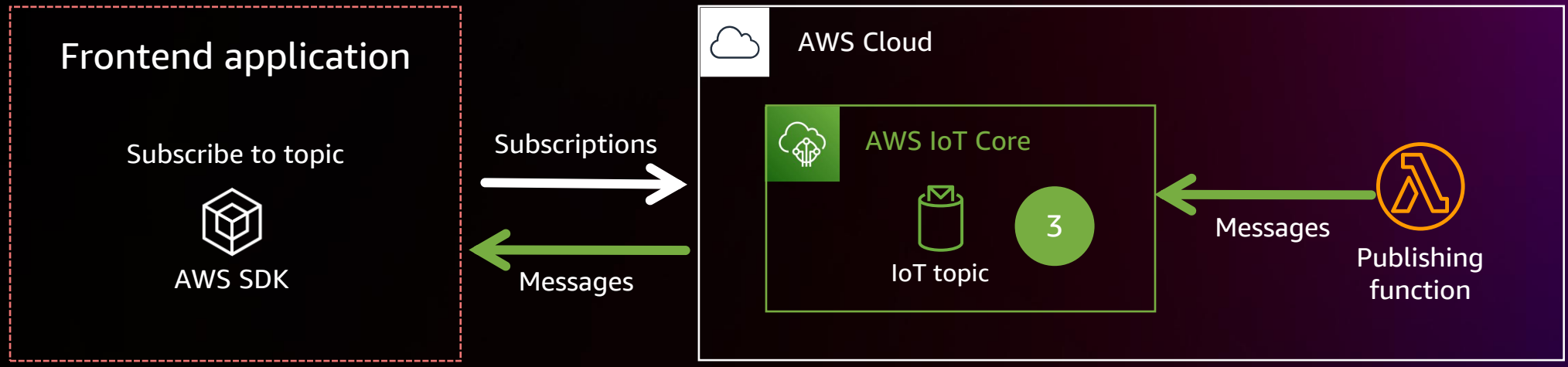


Receives messages published by the backend to this topic.

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Using AWS IoT Core for real-time messaging

Frontend application uses pub-sub to “listen” for event updates

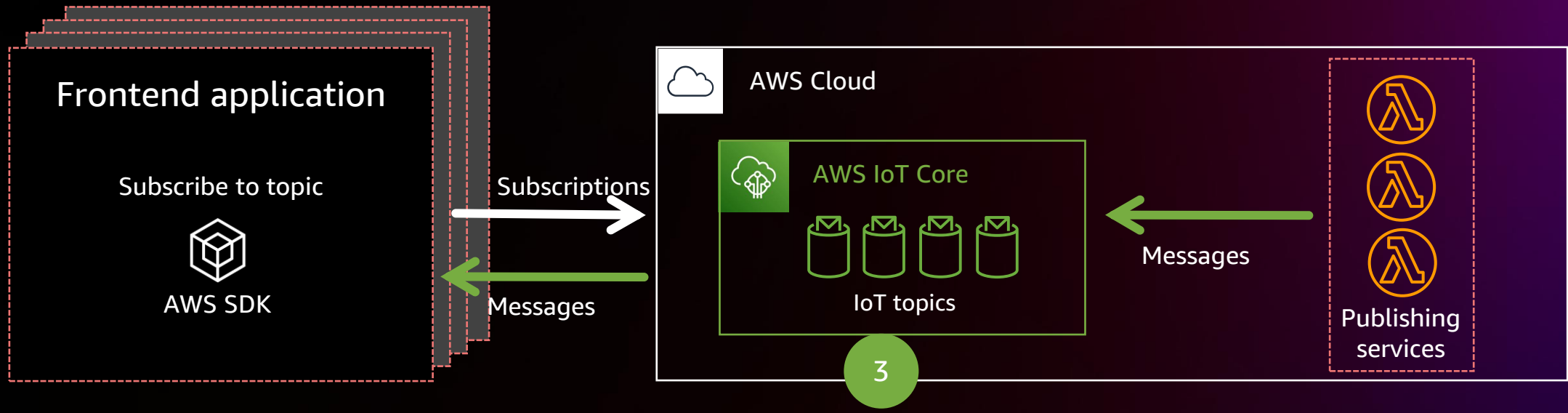


Messages are categorized using topics. Topic names are UTF 8 encoded strings.

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Using AWS IoT Core for real-time messaging

Frontend application uses pub-sub to “listen” for event updates



The IoT core service manages the WebSocket connection between backend publishers and frontend subscribers.

This enables fanout functionality to thousands of frontend devices.

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Using AWS IoT Core for real-time messaging

The *IotData* class in the AWS SDK returns a client that uses the MQTT protocol

Front End app

```
mqttClient.on('connect', function () {
     console.log('mqttClient connected’)
})

mqttClient.on('error', function (err) {
     console.log('mqttClient error: ', err)
})

mqttClient.on('message', function (topic, payload) {
     const msg = JSON.parse(payload.toString())
     console.log('IoT msg: ', topic, msg)
})
```

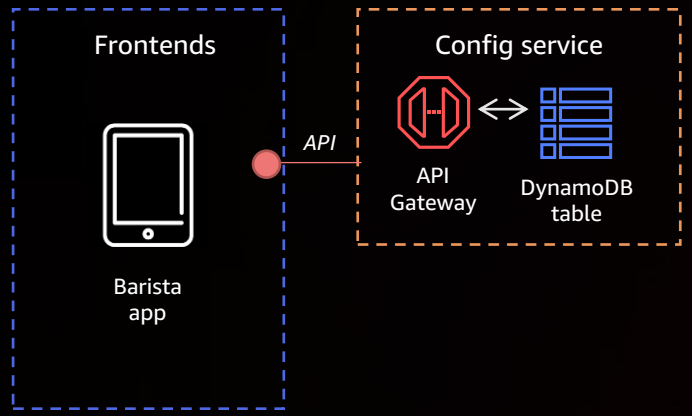
Once the frontend application establishes the connection, it returns messages, errors, and connection status via callbacks.

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Combining multiple approaches for your frontend application

Many frontend applications can combine synchronous and asynchronous response models.

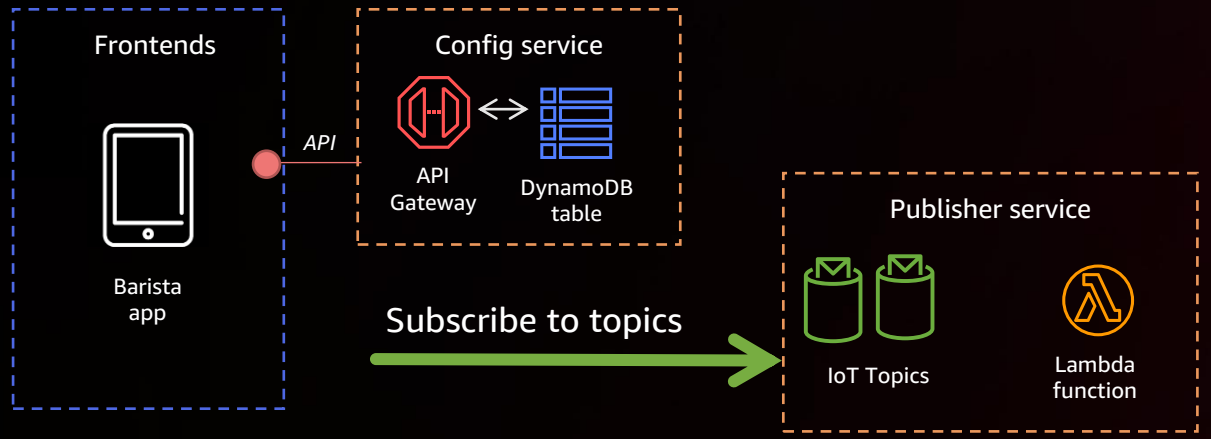


Serverlesspresso sends an initial synchronous request to retrieve the current “state of things.”

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Combining multiple approaches for your frontend application

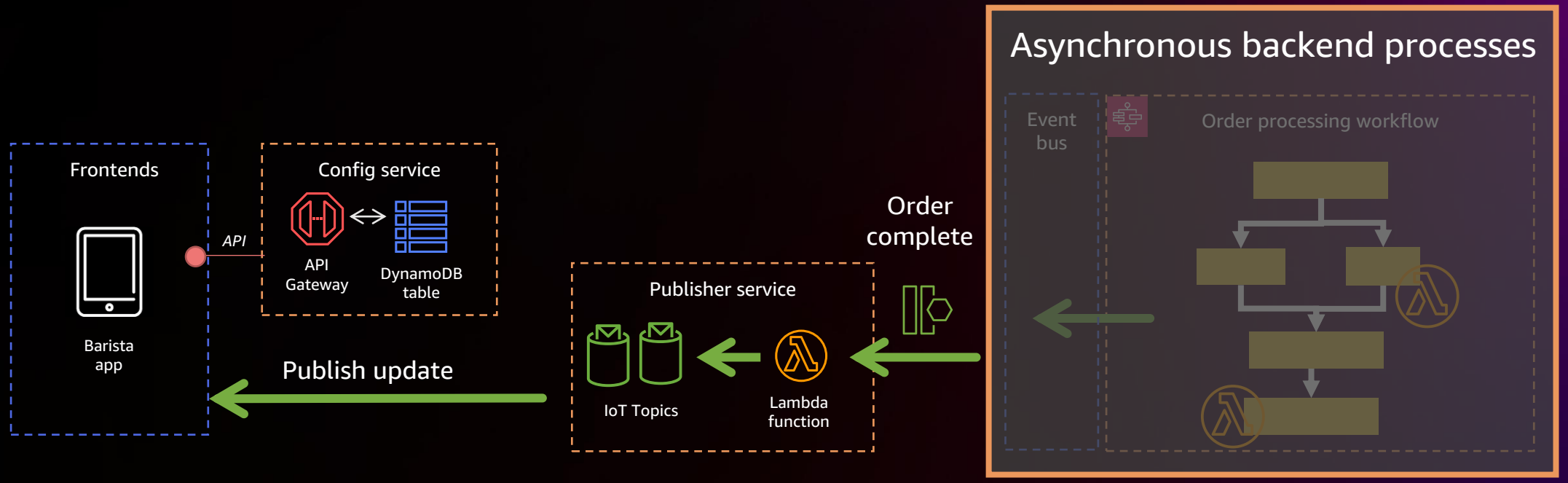
Many frontend applications can combine synchronous and asynchronous response models.



Serverlesspresso sends an initial synchronous request to retrieve the current “state of things.”

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Combining multiple approaches for your frontend application



New orders are posted to the application and processed asynchronously, with updates published to the frontend via the IoT topic.

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

What does it cost to run this workload?

WE SERVE UP TO 960 DRINKS TO 960 CUSTOMERS EVERY DAY

Service	Daily cost	With Free Tier
AWS Amplify Console	$0.28	Free
Amazon API Gateway	$0.01	Free
Amazon Cognito	$0.00	Free
Amazon DynamoDB	$0.01	Free
Amazon EventBridge	$0.01	Free
AWS IoT Core	$0.01	Free
AWS Lambda	$0.01	Free
Amazon SNS	$7.98	$7.98
AWS Step Functions	$0.29	Free

Learn about the AWS Free Tier:
https://aws.amazon.com/free

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Summary



serverlesspresso

An event-driven coffee-ordering app
built with serverless architecture

Uses Step Functions to orchestrate resources within a microservice

Uses events to communicate between microservices

Uses IoT Core to maintain open connection for asynchronous responses

aws

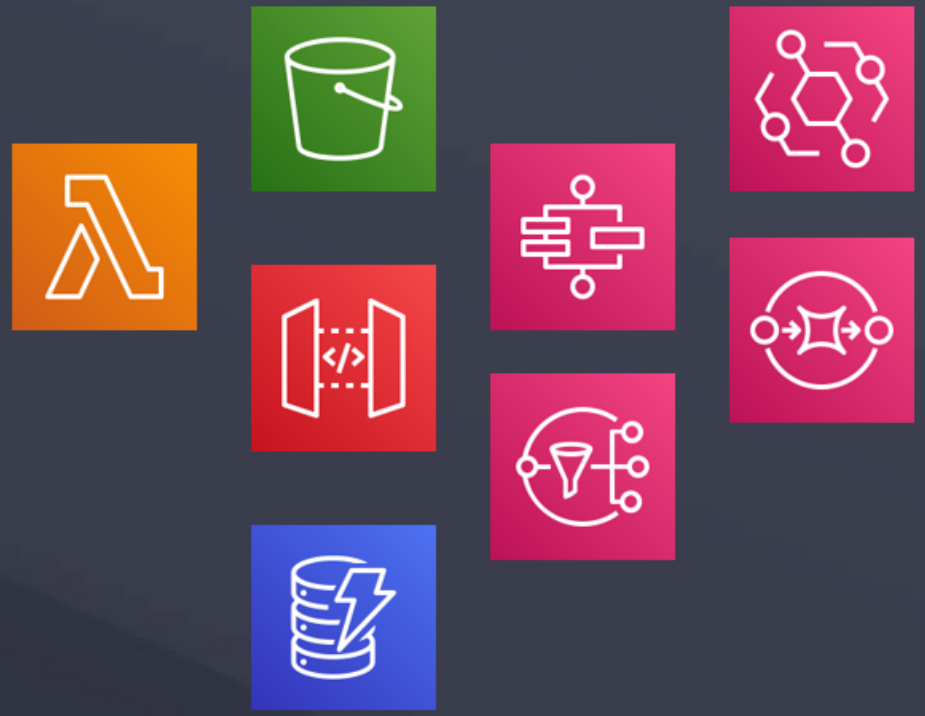
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

New Blogs Videos Learn Events Patterns About Search Search

Welcome to **Serverless** Land

This site brings together all the latest blogs, videos, and training for AWS Serverless. Learn to use and build apps that scale automatically on low-cost, fully-managed serverless architecture.

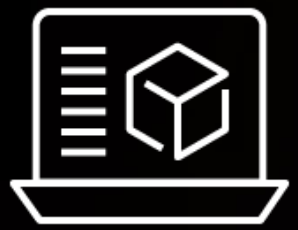
Learn More



For more serverless learning resources, visit
https://serverlessland.com

Continue your AWS Serverless learning

Learn at your own pace



Expand your serverless skills with our Learning Plan on **AWS Skill Builder**

Increase your knowledge



Use our **Ramp-Up Guides** to build your serverless knowledge

Earn AWS Serverless badges



Demonstrate your knowledge by earning **digital badges**

https://s12d.com/serverless-learning

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Thank you!

James Beswick
@jbesw



Please complete the session survey in the **mobile app**

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.