AWS re:Invent

NOV. 29 – DEC. 3, 2021 | LAS VEGAS, NV

SVS317

Best practices for building interactive applications with AWS Lambda

Benjamin Smith
Senior Developer Advocate
AWS Serverless

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.



About me

- Ben Smith
  - Email: benjasl@amazon.com
  - Twitter: @benjamin_l_s
- Most interested in
  - Serverless developer workflow
  - Serverless and PHP
- Random fact
  - I once taught David Hasselhoff how to ski

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Agenda

Interactive AWS Lambda applications

Business logic

Synchronous and asynchronous

Production-ready web apps

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

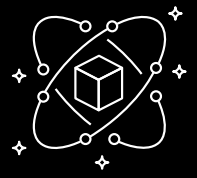
The end goal



Customer value



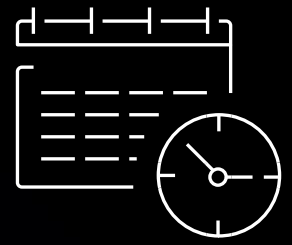
Reliability



Resilience



Scalability



And do it faster

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Today’s focus

Lambda-based applications

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Today’s focus

Interactive



Lambda-based applications

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Today’s focus

Interactive

HTTP request



AWS
Lambda

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Today’s focus

Interactive

HTTP request  HTTP response



AWS
Lambda

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Interactive AWS Lambda applications

Connecting applications

Extending applications

Asynchronous workloads

ML inferencing

Web applications

Backend applications

Data processing

Chatbots

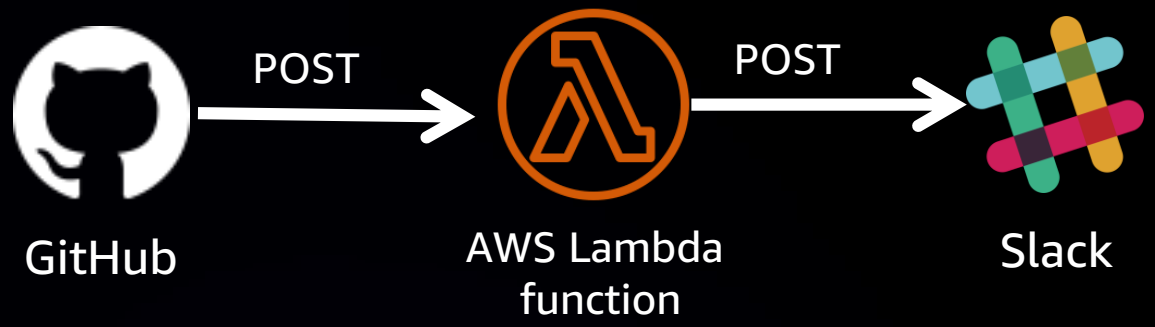
Amazon Alexa skill

IT automation

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Interactive AWS Lambda applications

- Connecting applications
- Extending applications
- Asynchronous workloads
- ML inferencing
- Web applications
- Backend applications
- Data processing
- Chatbots
- Amazon Alexa skill
- IT automation





Learn more

http://s12d.com/**getting-started**

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Interactive AWS Lambda applications

Connecting applications

Extending applications

Asynchronous workloads

ML inferencing

Web applications

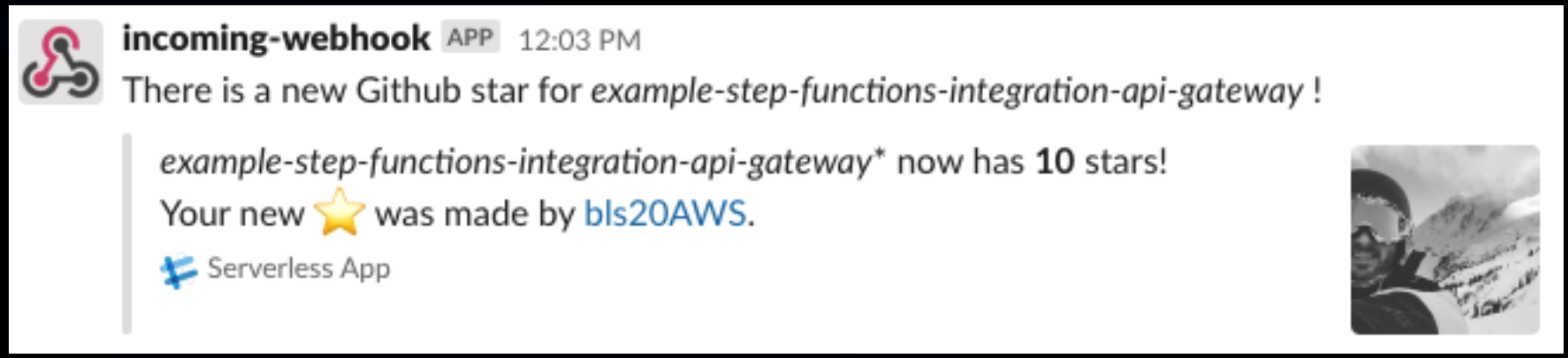
Backend applications

Data processing

Chatbots

Amazon Alexa skill

IT automation





Learn more

http://s12d.com/**getting-started**

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.



Learn more

http://s12d.com/**getting-started**

The business logic

Do “something” to some data
to produce valuable output

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS Lambda: Input, process, output



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS Lambda: Input, process, output



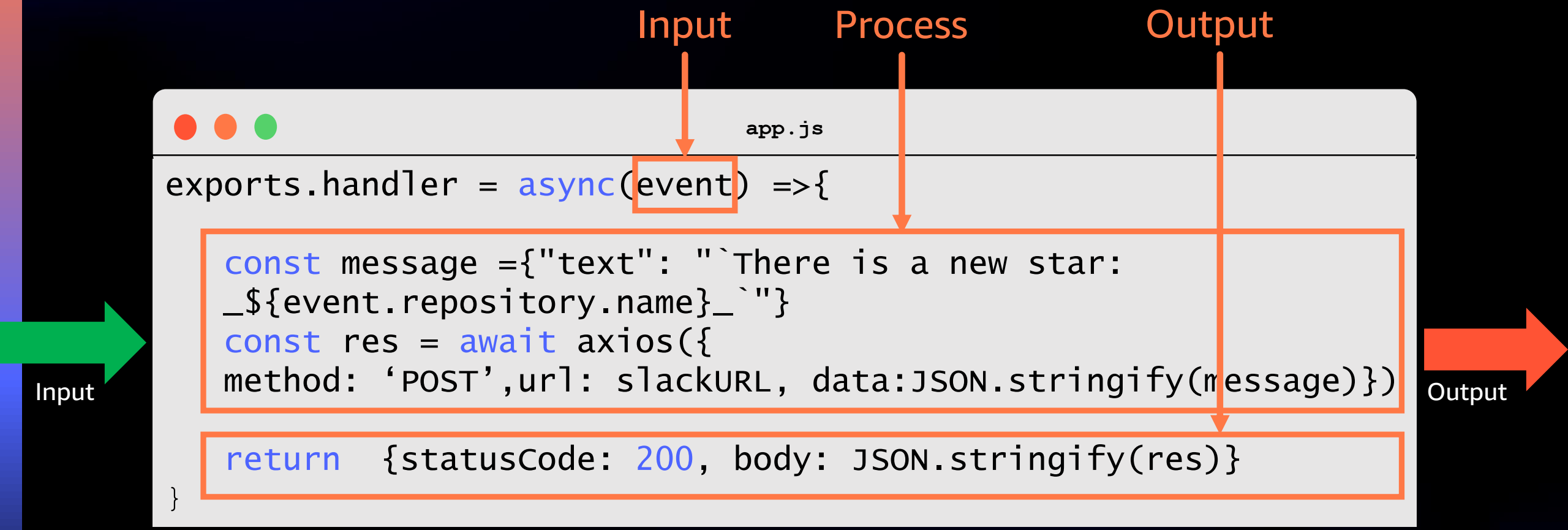
Event Payload

```
{
   "repository": {
      "name": "getting-started-with-serverless",
      "stargazers_count": 38
   },
   "sender": {
      "login": ”bls20AWS”
   }
}
```

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS Lambda function

RUN BUSINESS LOGIC IN RESPONSE TO EVENTS

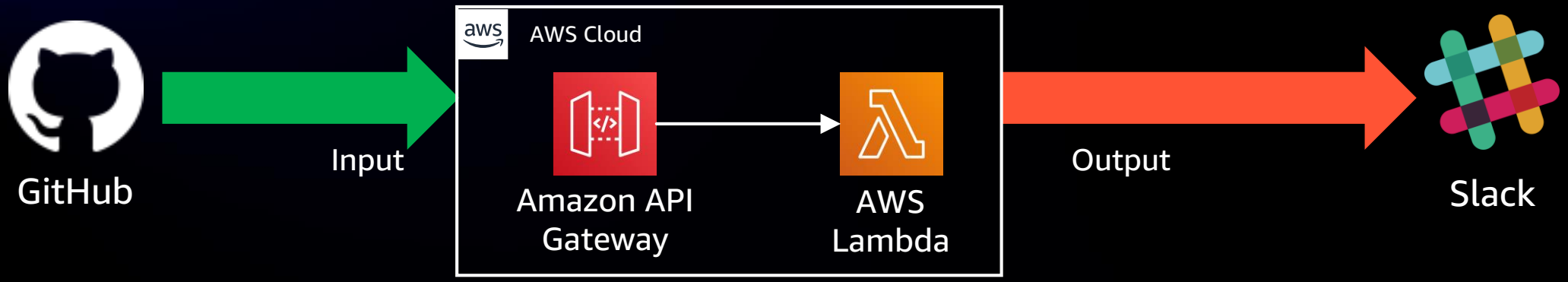


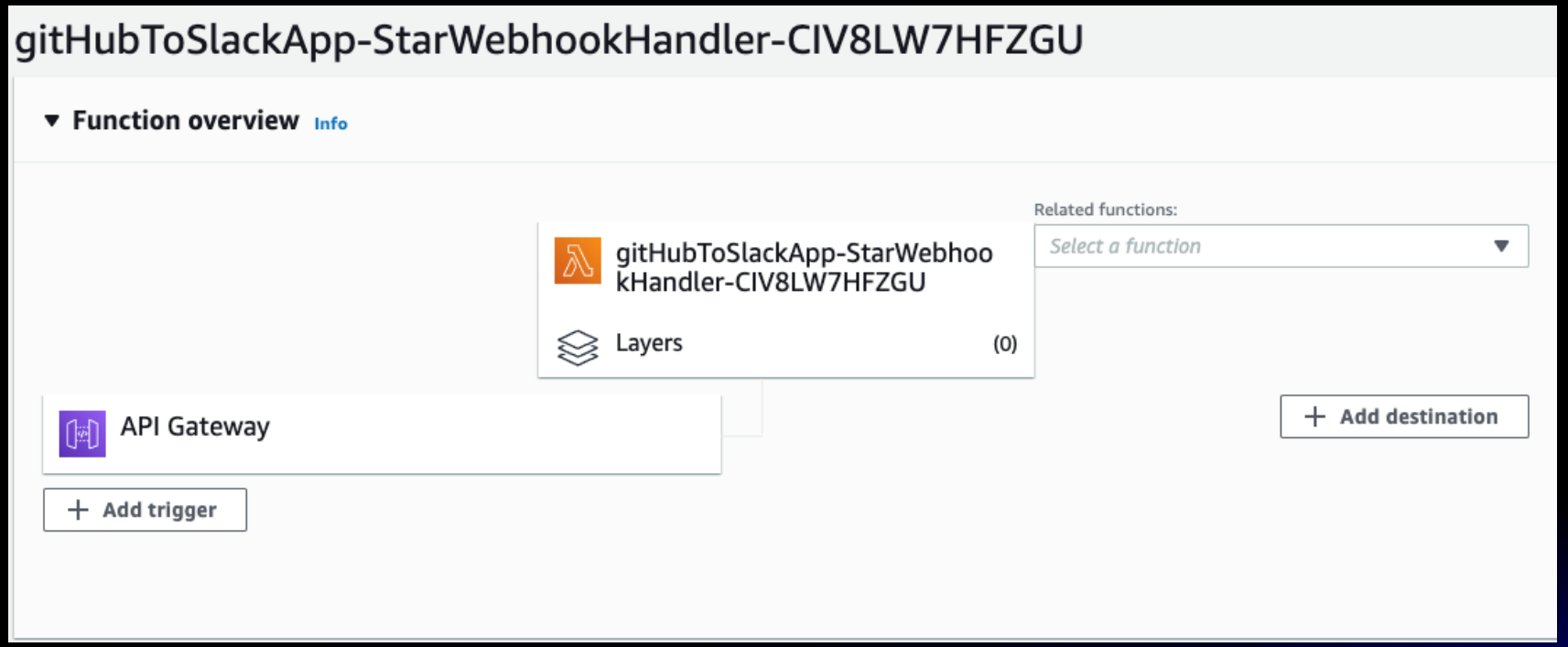
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

How did the event payload get from GitHub to AWS Lambda?

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS Lambda and Amazon API Gateway





© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Why do developers need serverless technologies?

I only want to write business logic.

I don’t want to pay when my workload is idle.



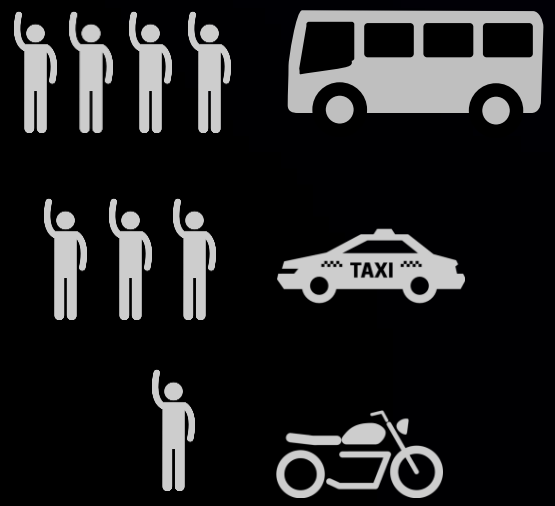
I want my application to scale on its own.

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Vertical scaling and horizontal scaling

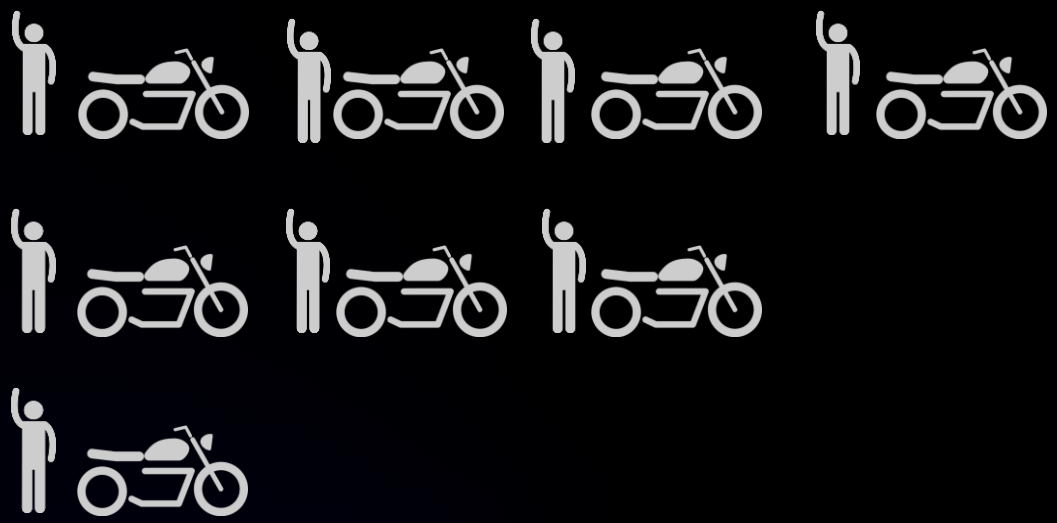
Vertical scaling

Each increase in load requires the purchase of a larger transport mode to bear the load



Horizontal scaling

Add an additional capacity unit in line with demand

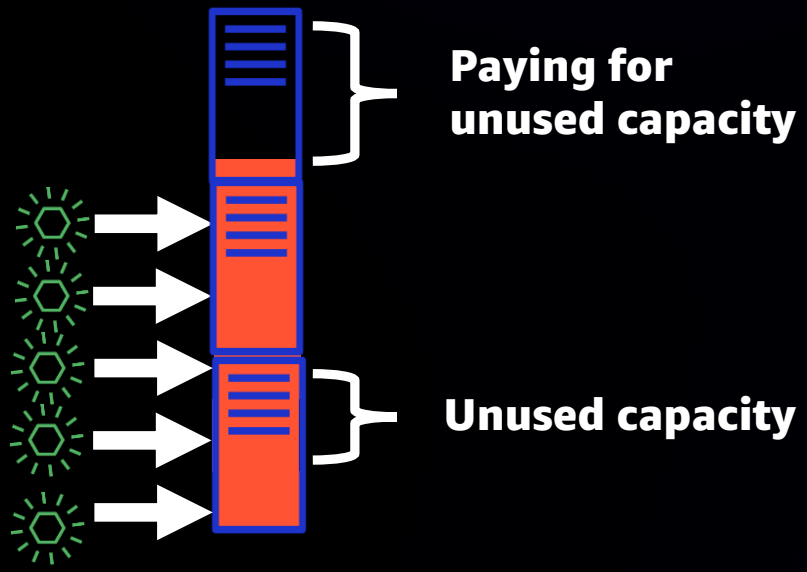


© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Serverless scaling

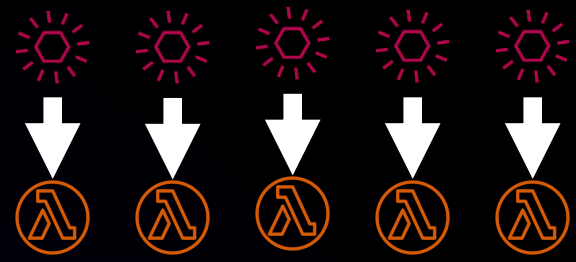
Vertical scaling

Each increase in load requires the purchase of a larger transport mode to bear the load



Horizontal scaling

Add an additional capacity unit in line with demand



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Why do developers need serverless technologies?

Traditional app dev

Before you start coding, there are a number of tasks to think about

Provision server(s)

Configure network access

Raising a ticket

Configure backups

Maintain server

Secure server

Patch and update

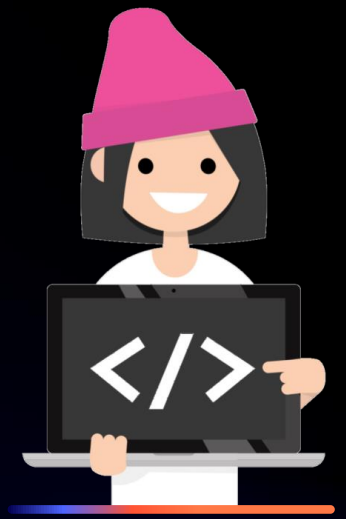
Configure load balancing



Serverless app dev

Before you start coding . . .

What runtime shall I choose?



aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

How much

LOGIC

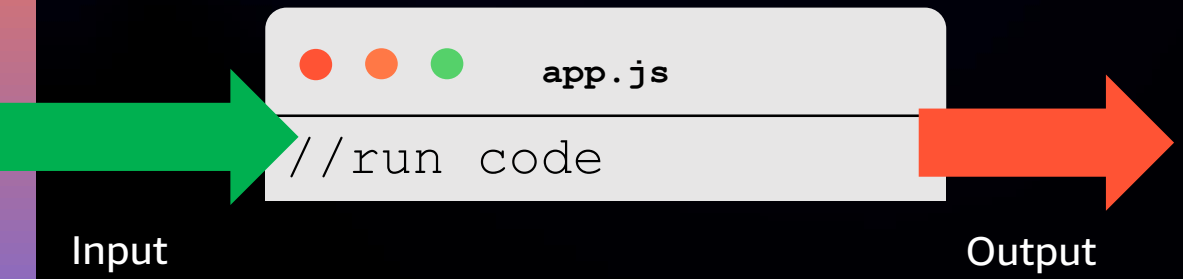


are you squeezing in to your code?

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS Lambda function

RUN BUSINESS LOGIC IN RESPONSE TO EVENTS



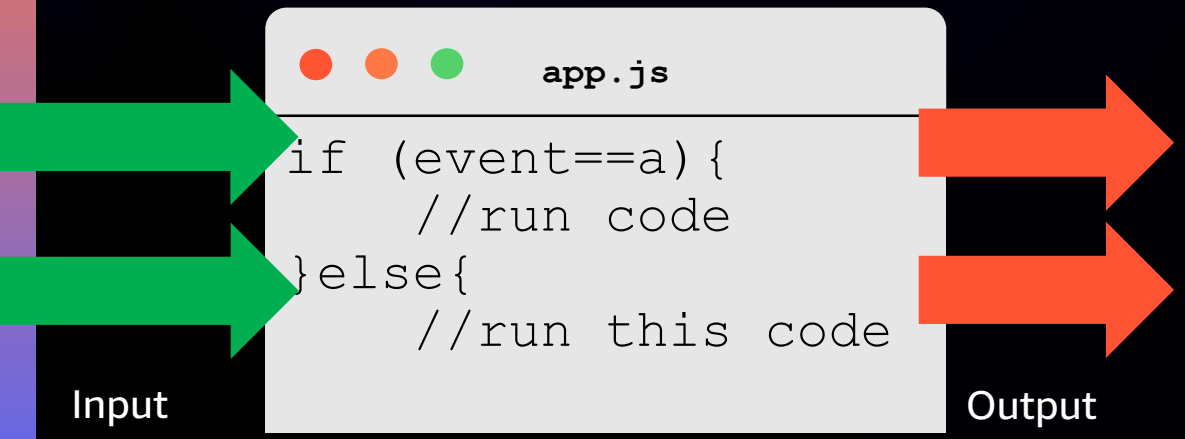
Complex business logic leads to:

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

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS Lambda function

RUN BUSINESS LOGIC IN RESPONSE TO EVENTS



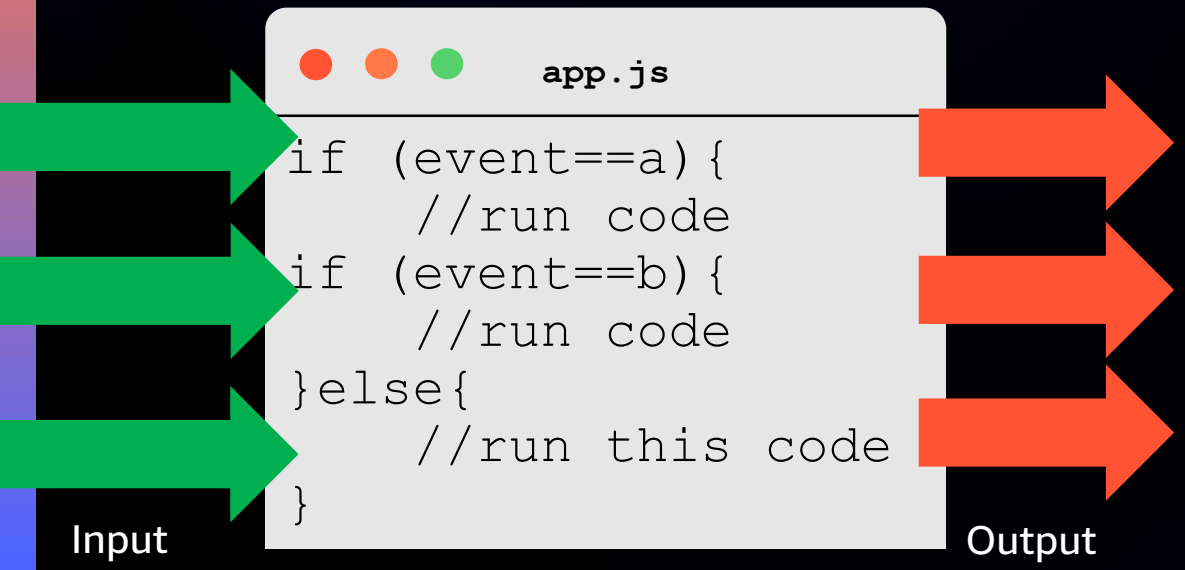
Complex business logic leads to:

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

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS Lambda function

RUN BUSINESS LOGIC IN RESPONSE TO EVENTS



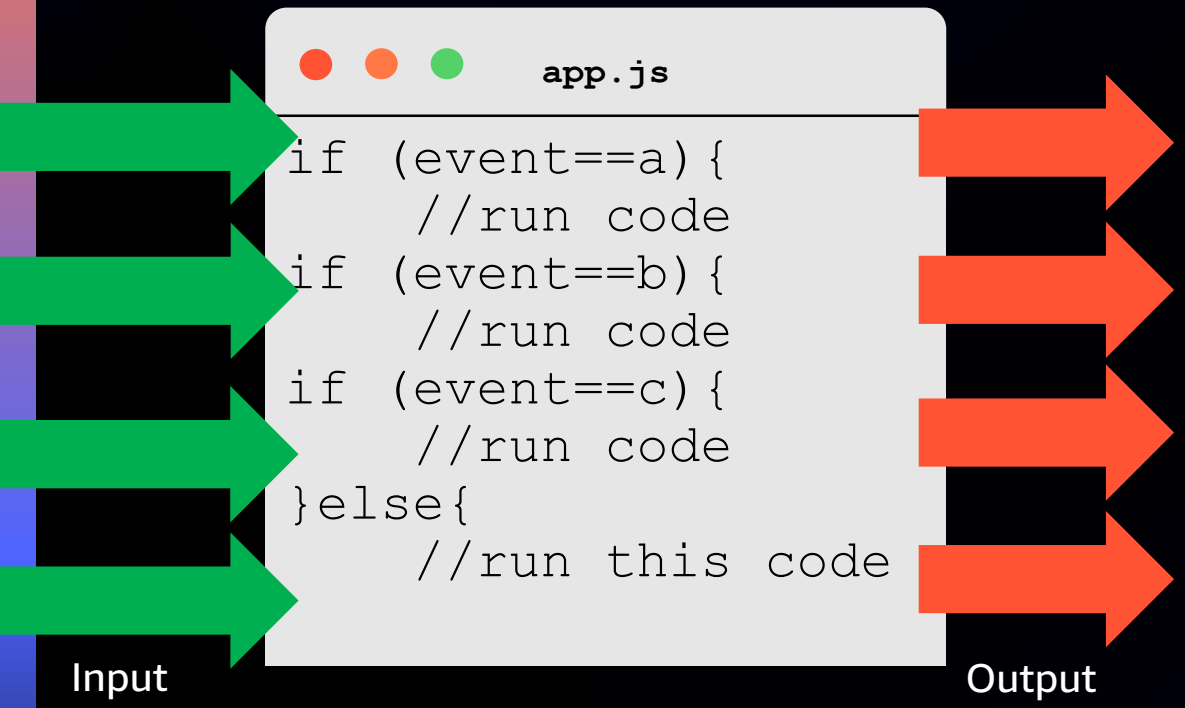
Complex business logic leads to:

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

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS Lambda function

RUN BUSINESS LOGIC IN RESPONSE TO EVENTS



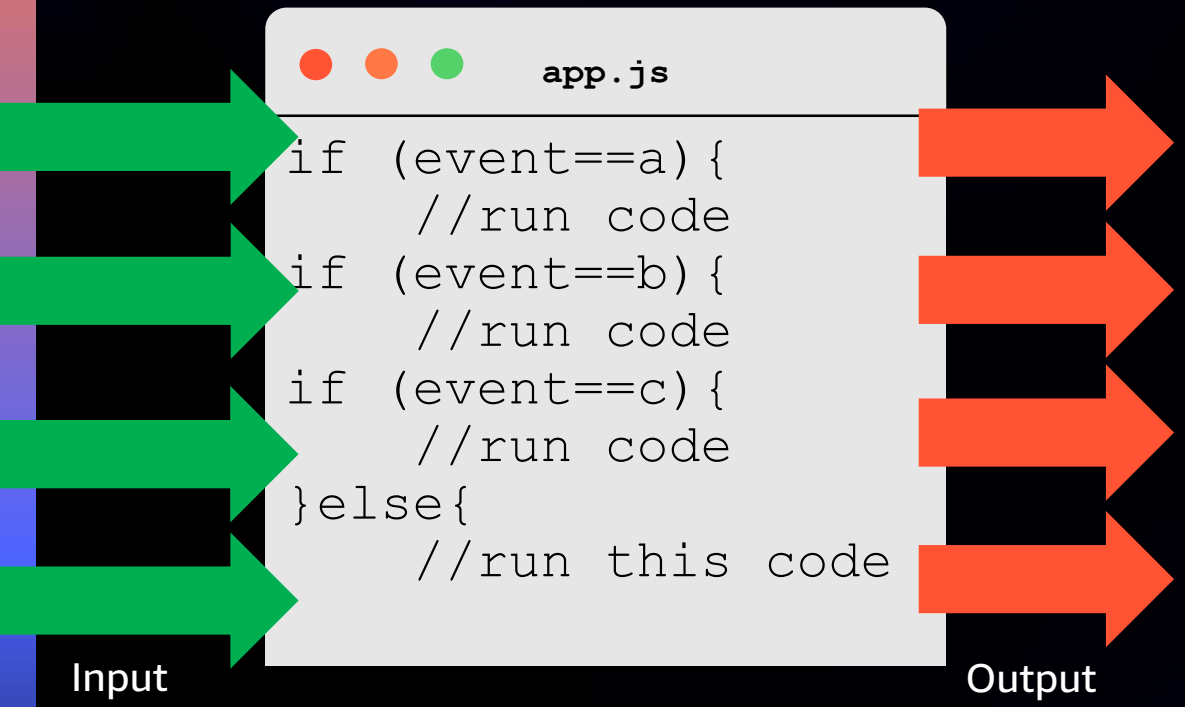
Complex business logic leads to:

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

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS Lambda function

RUN BUSINESS LOGIC IN RESPONSE TO EVENTS



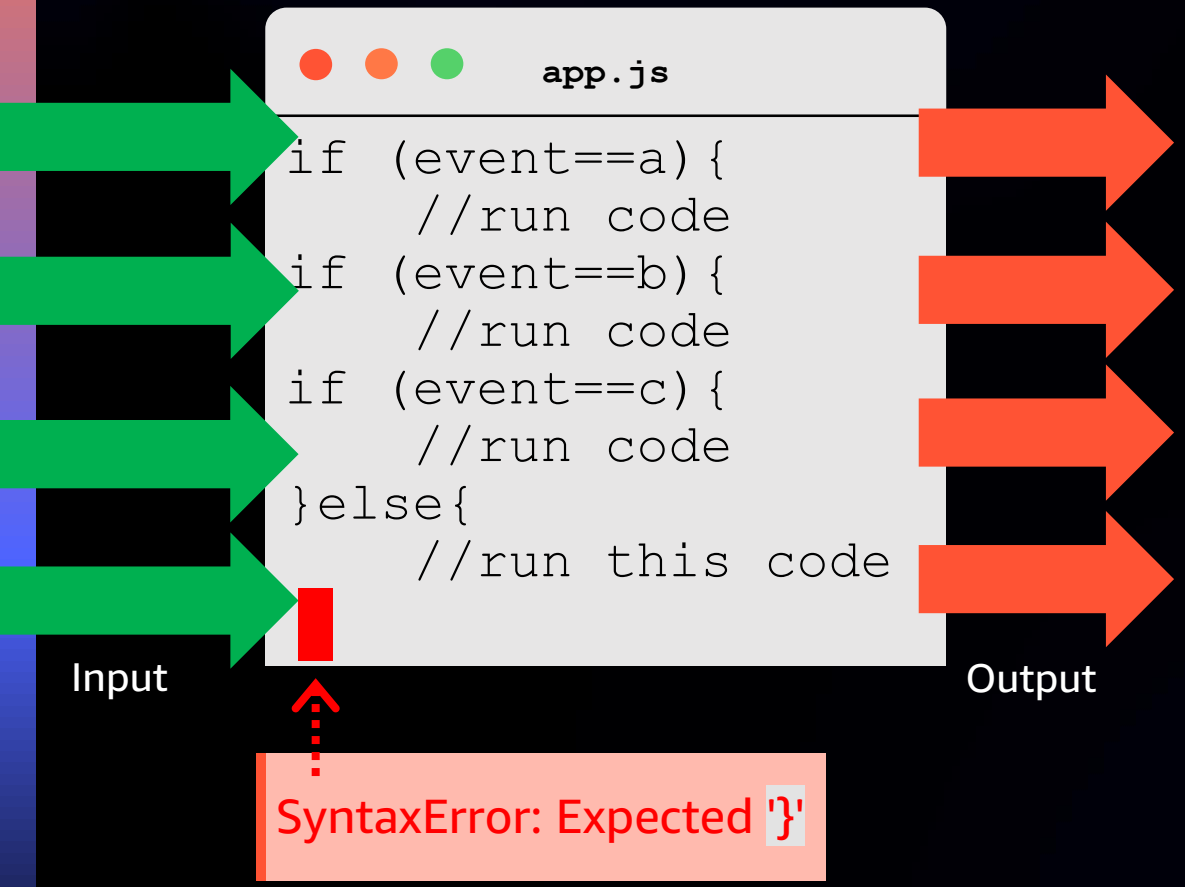
If something is going wrong, my code is the most probable place.

Why?

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS Lambda function

RUN BUSINESS LOGIC IN RESPONSE TO EVENTS



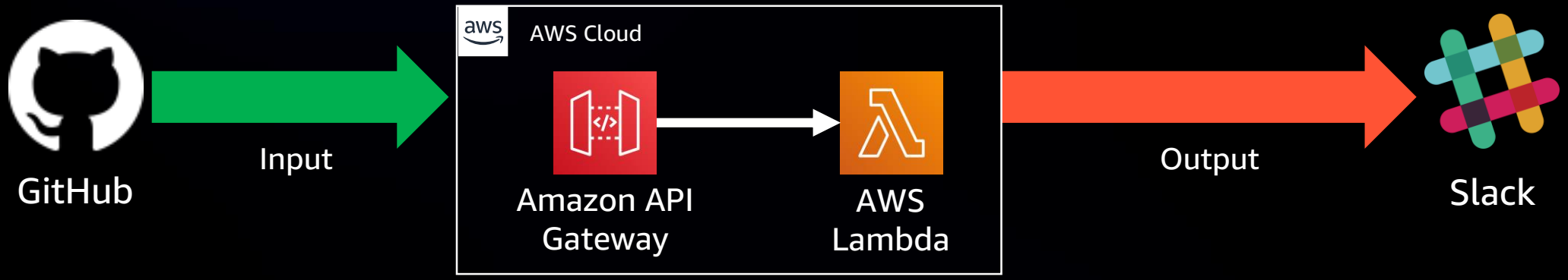
If something is going wrong, my code is the most probable place.

Why?

Because it's MY code.

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

How can we improve this?

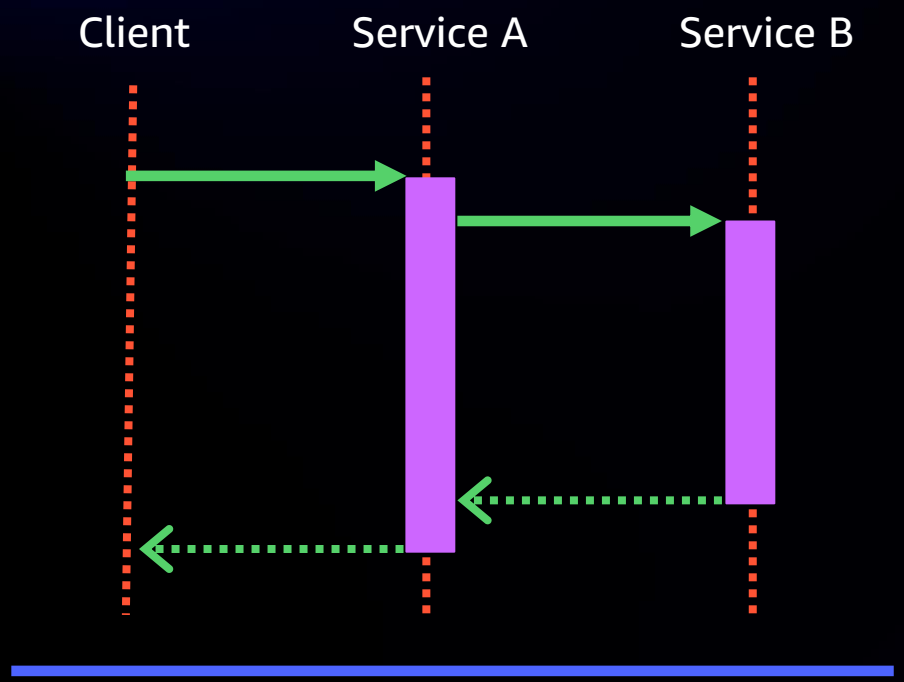


© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Thinking asynchronously

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

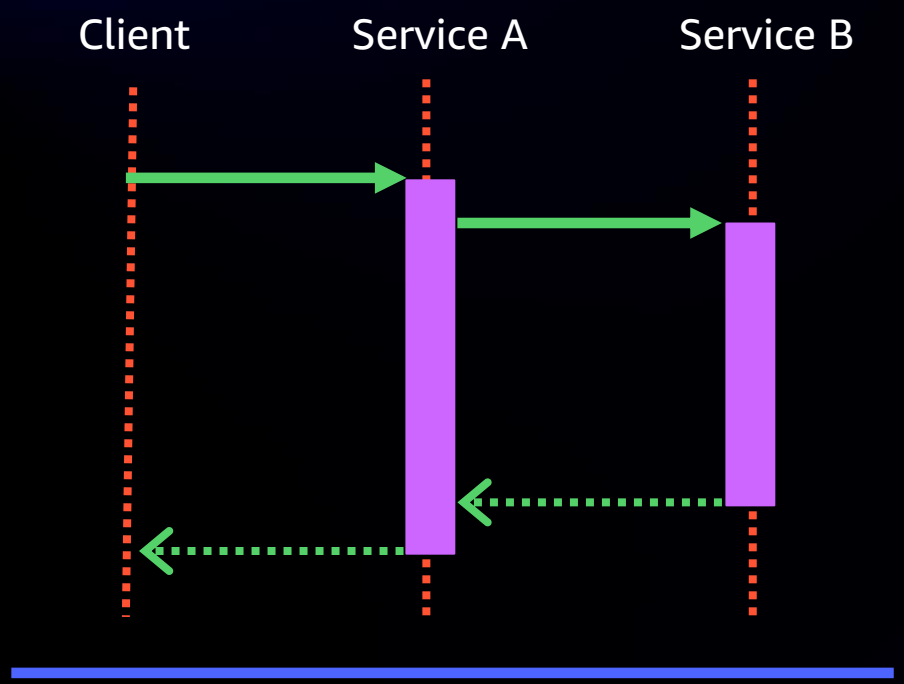
Synchronous and asynchronous



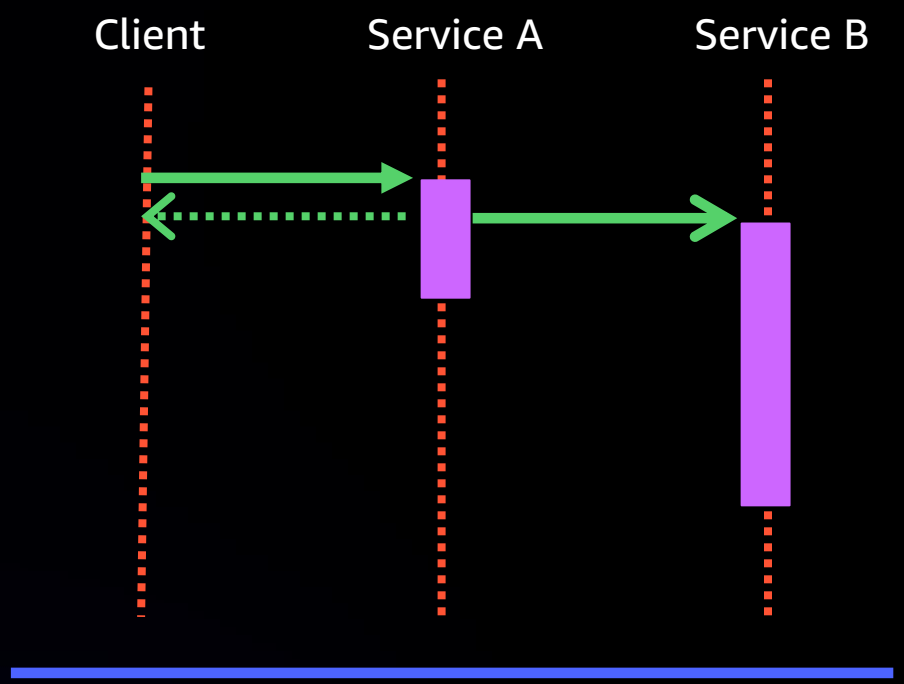
Synchronous commands

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Synchronous and asynchronous



Synchronous commands

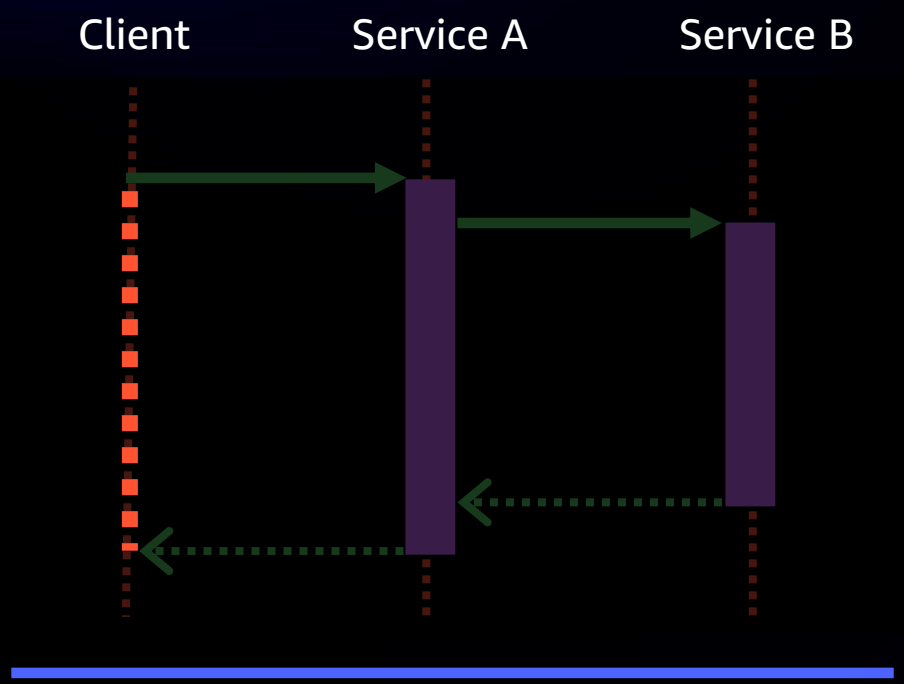


Asynchronous events

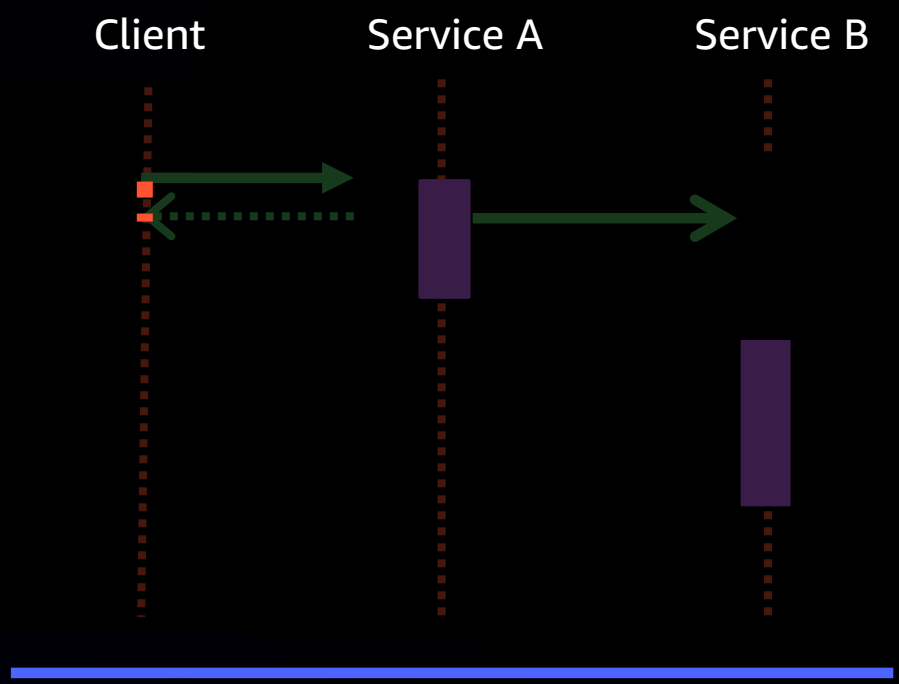
aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Synchronous and asynchronous



Synchronous commands

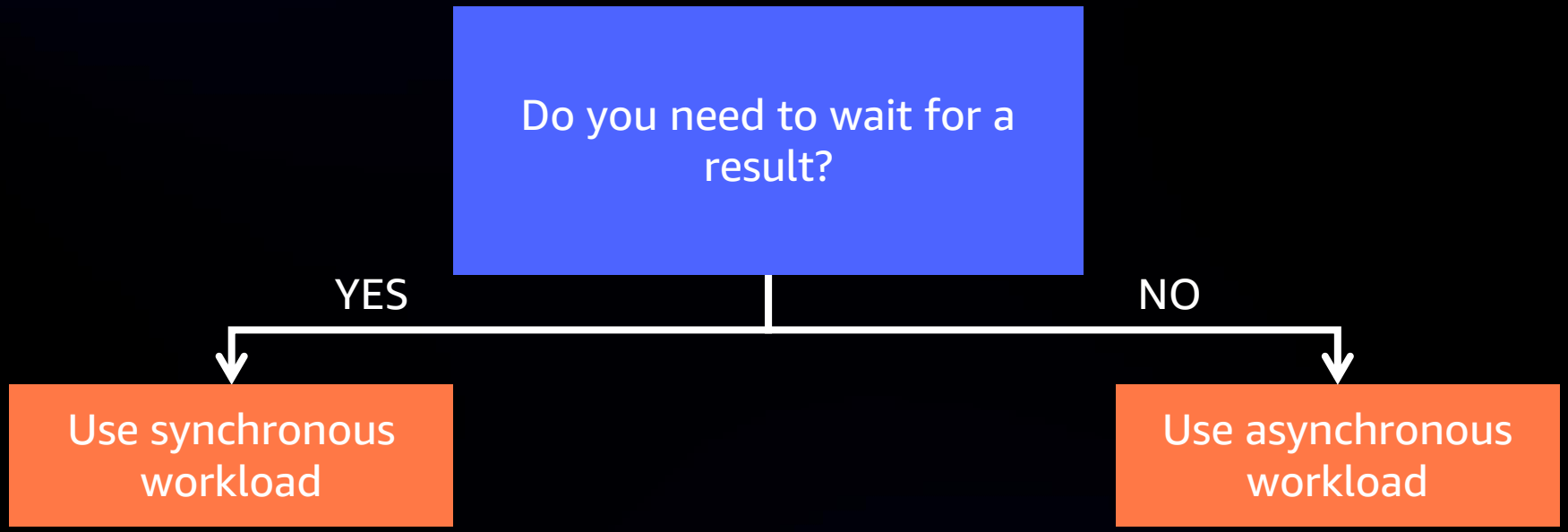


Asynchronous events

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Choosing synchronous or asynchronous

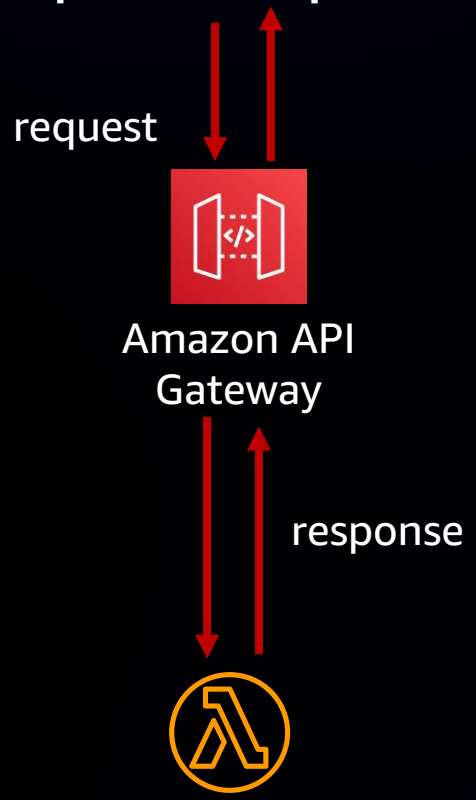


© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

The AWS Lambda invocation model

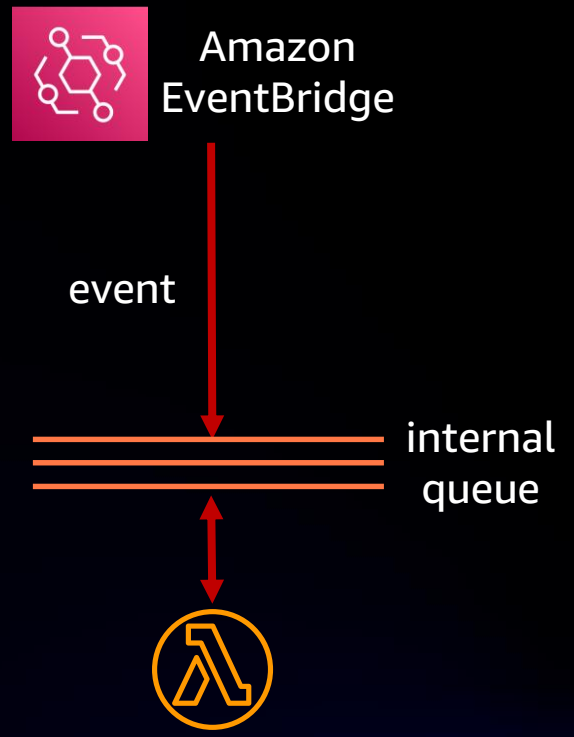
Synchronous

request/response



Asynchronous

event

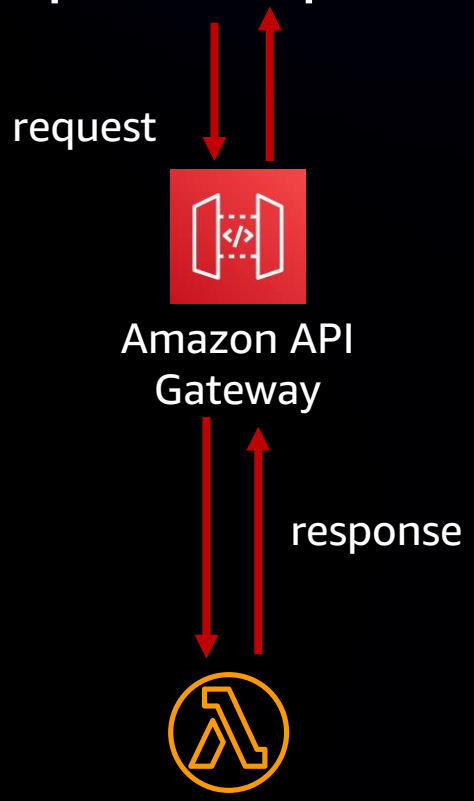


© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Synchronous and asynchronous invocations

Synchronous

request/response



Response syntax

HTTP/1.1 StatusCode (200 success)
X-Amz-Function-Error: FunctionError
X-Amz-Log-Result: LogResult
X-Amz-Executed-Version: ExecutedVersion

Response payload:

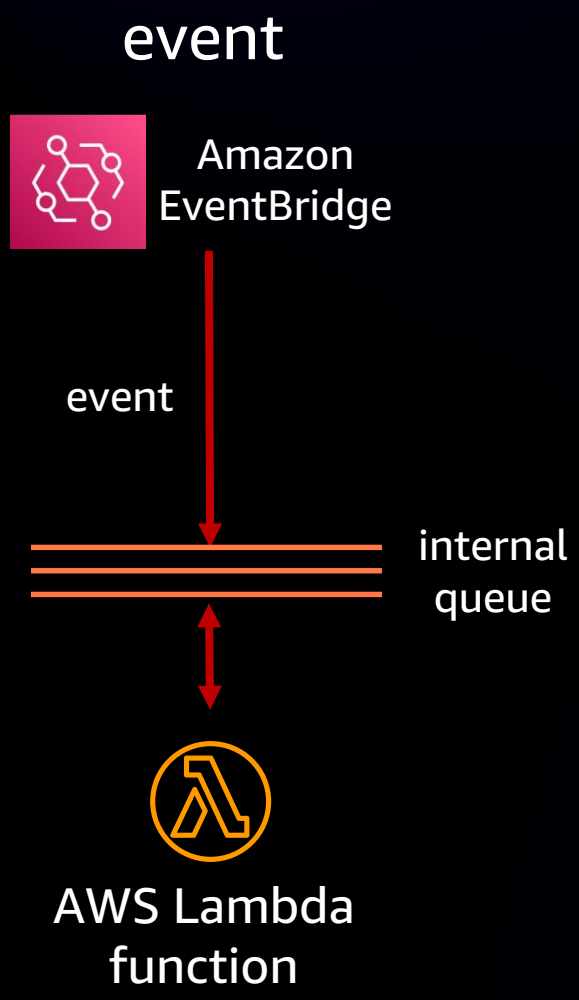
response.JSON

```
{
    "statusCode":200,
    "body":"[{\"id\":\"1\",\"name\":\"ben\"}]"
}
```

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Synchronous and asynchronous invocations

Asynchronous



Response syntax

HTTP/1.1 StatusCode (202 success)

Response payload:
x

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Sync and async invocation responses

ASYNCHRONOUS

Function invocation has been received

202

SYNCHRONOUS

Function has finished Running
and here is the result

200

```
{
    "body":"[{\"id\":\"1\",\"name\":\"ben\"}]”
}
```

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Sync and async invocation responses

Asynchronous

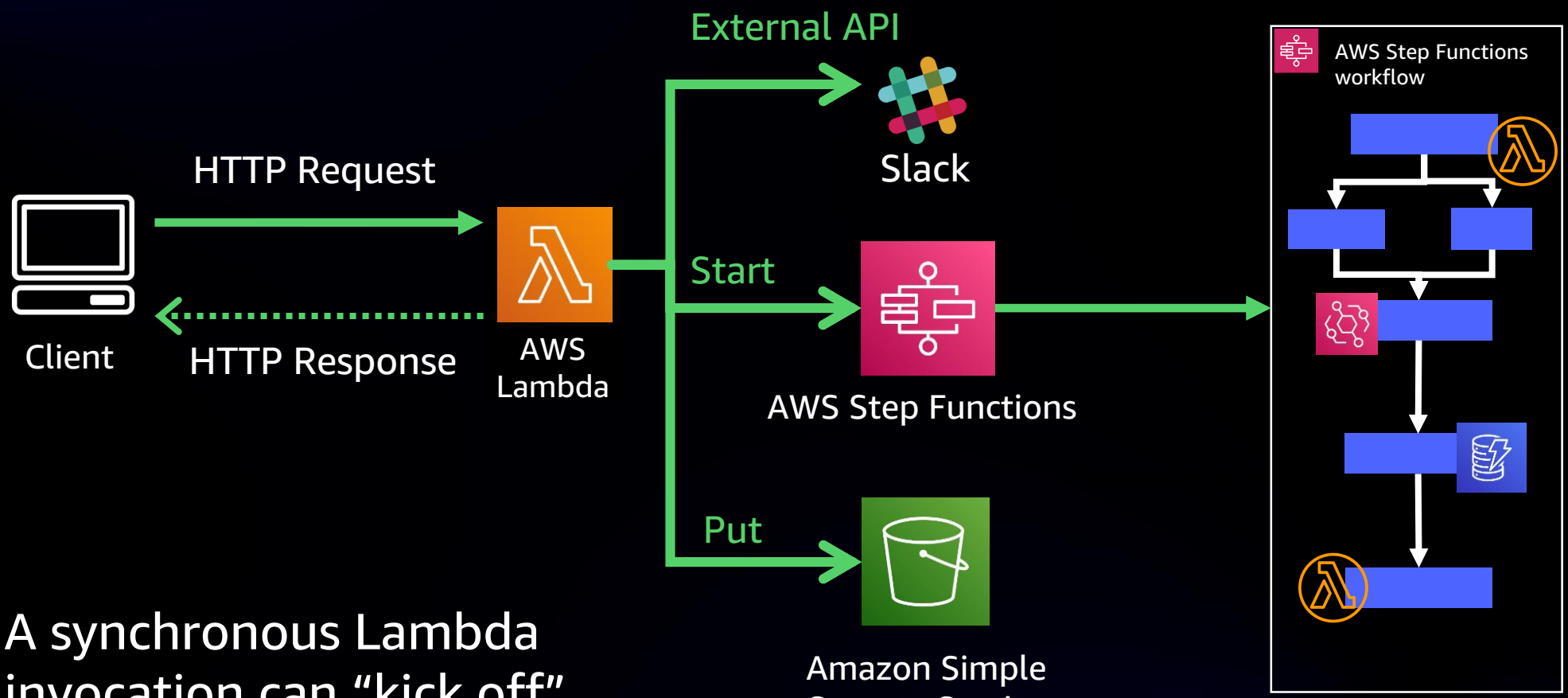
Returns an acknowledgement

Synchronous

Returns a response

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

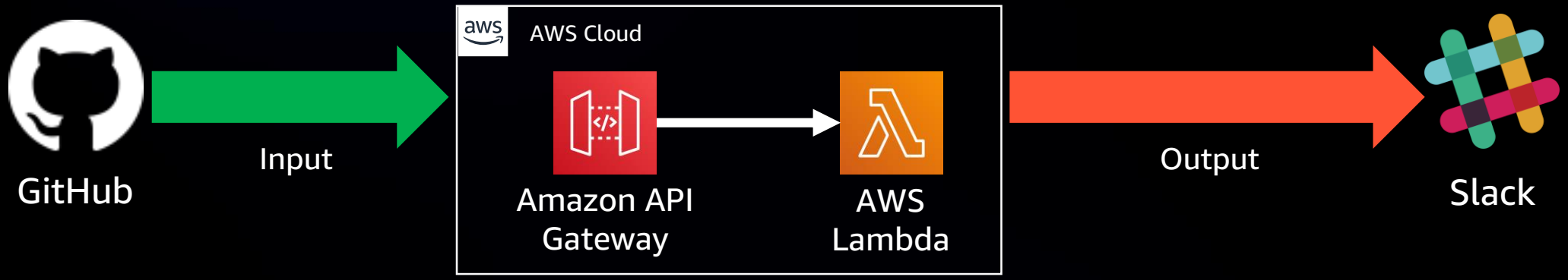
Thinking asynchronously



A synchronous Lambda invocation can “kick off” an asynchronous workload

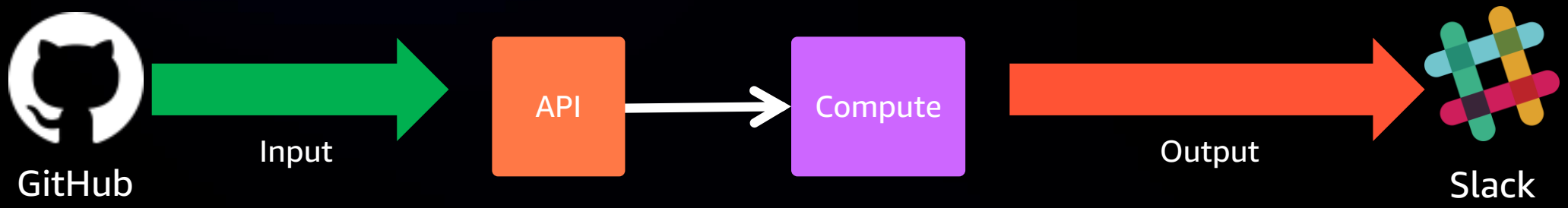
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

How can we improve this?



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

How can we improve this?



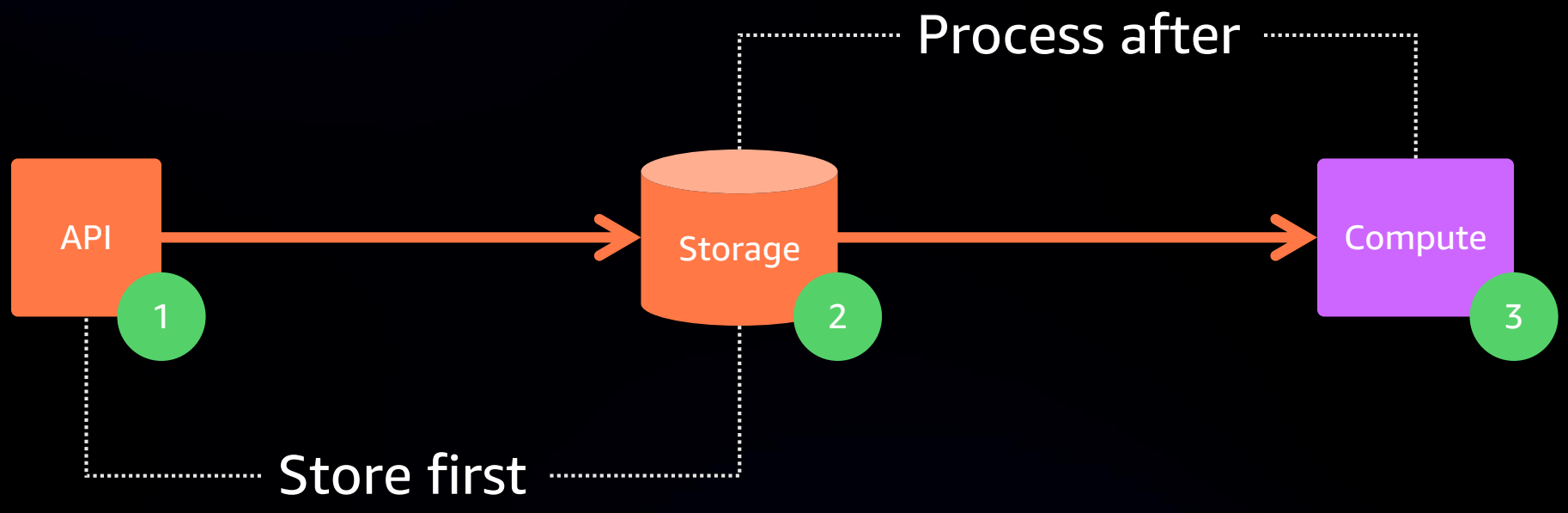
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Thinking asynchronously



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

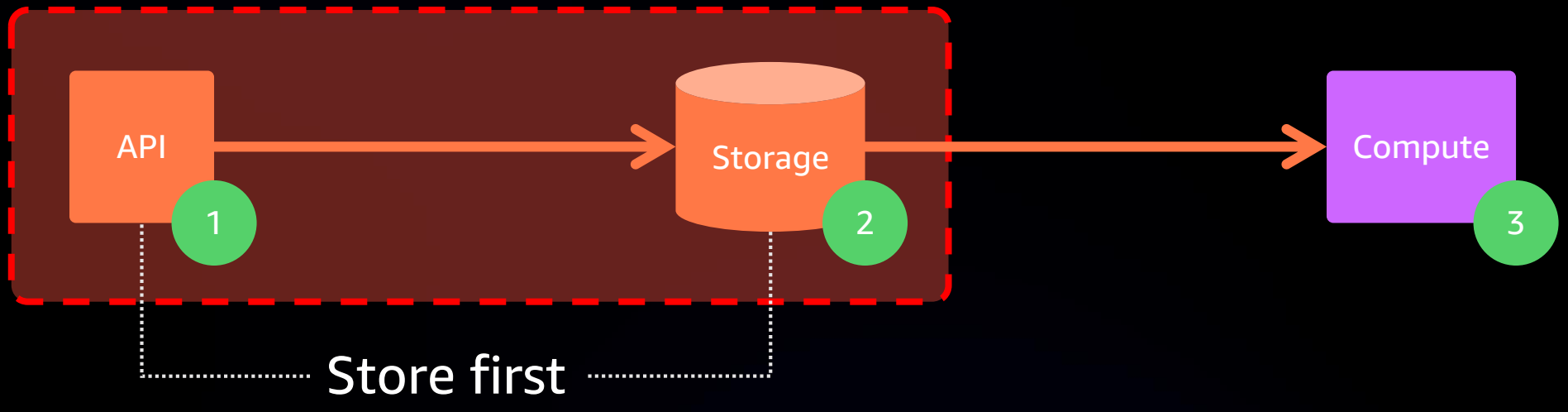
Thinking asynchronously



aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Thinking asynchronously

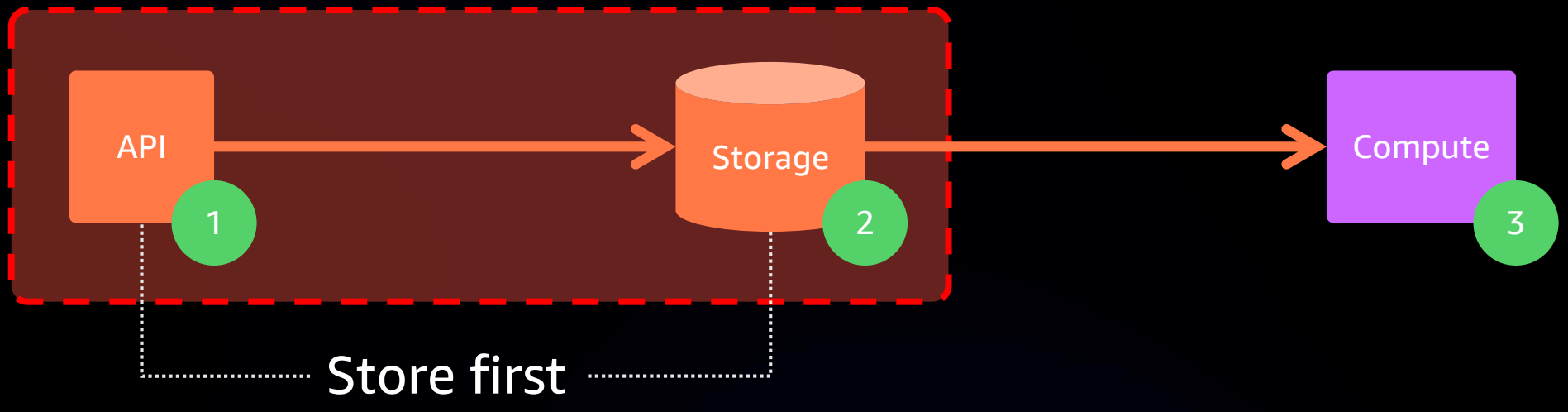


Greater reliability

Data is stored before my code gets a hold of it

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Thinking asynchronously



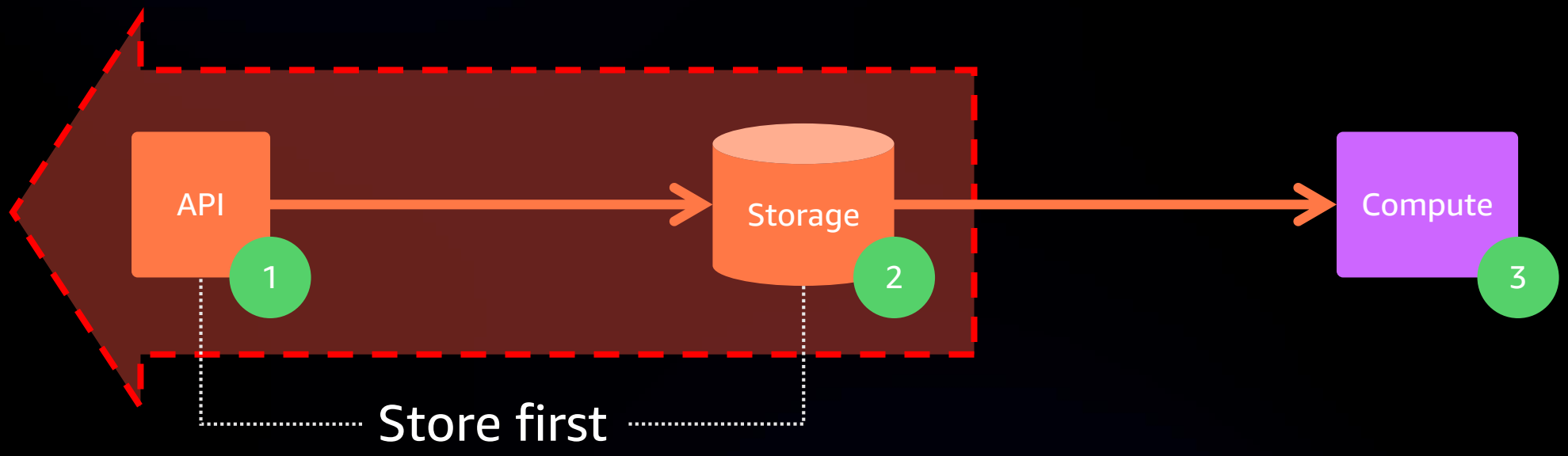
Less code

Reduce code through direct service integrations

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Thinking asynchronously



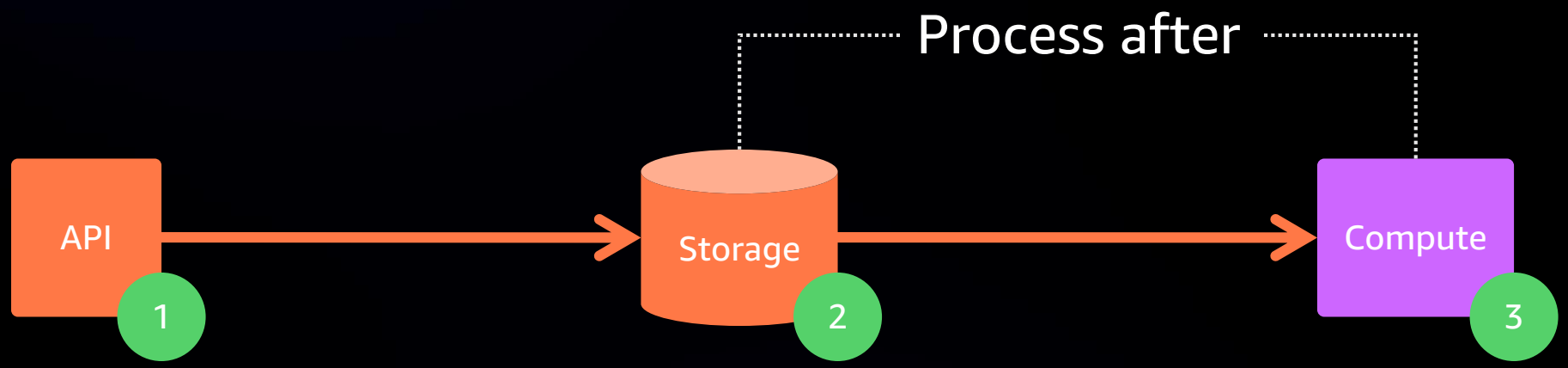
Faster response time

Client receives acknowledgement and can poll for further data if needed

aws

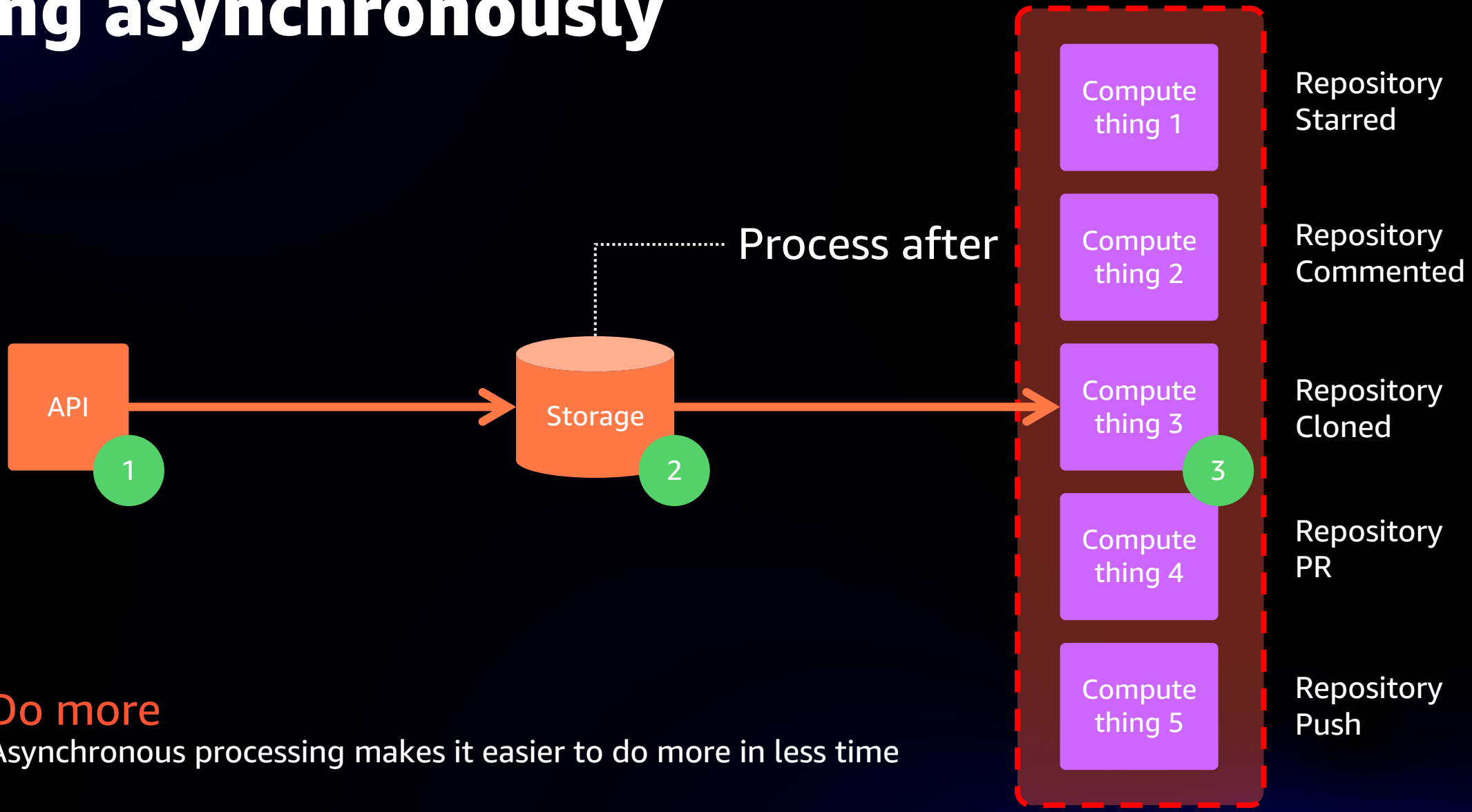
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Thinking asynchronously



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Thinking asynchronously



Do more

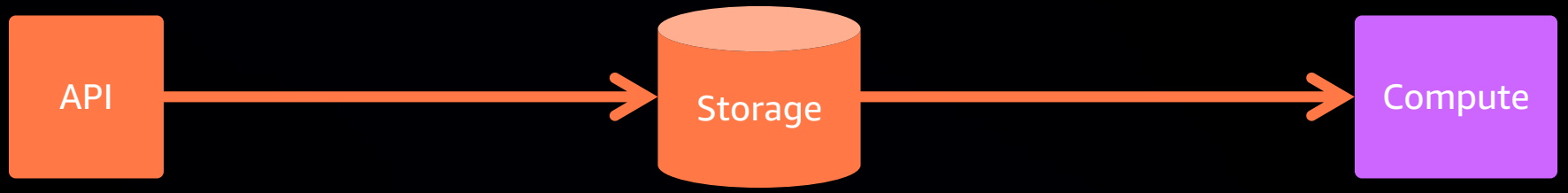
Asynchronous processing makes it easier to do more in less time

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Making it work

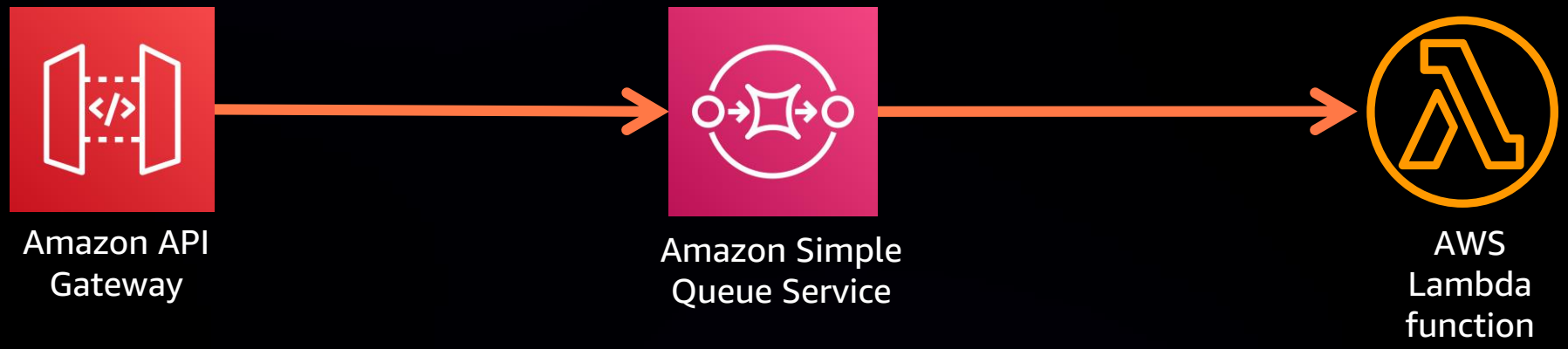
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Making it work



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Making it work on AWS

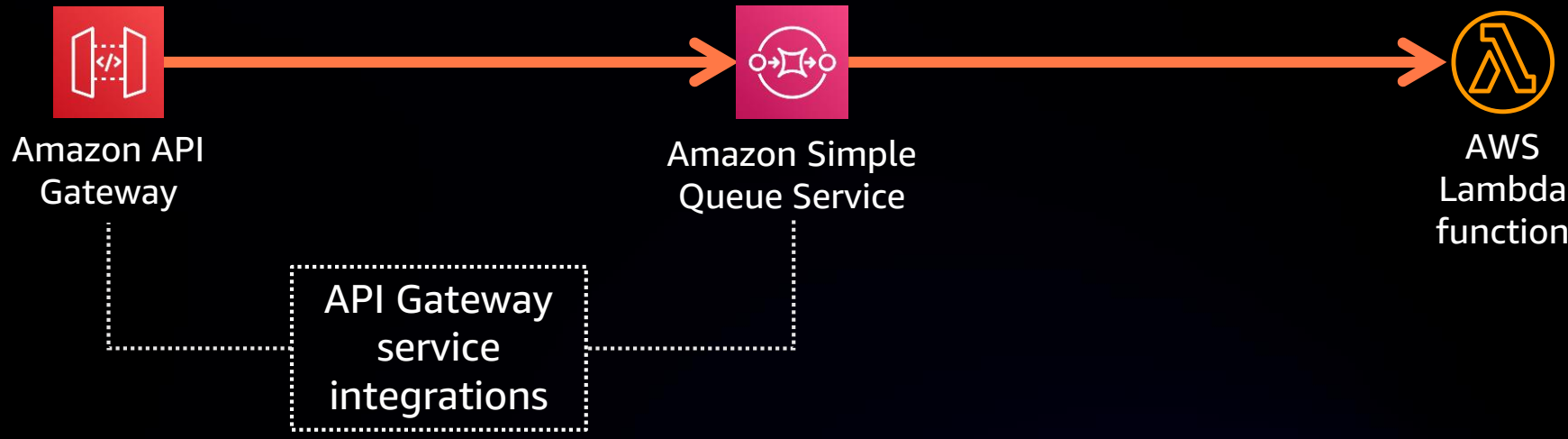


© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Making it work on AWS

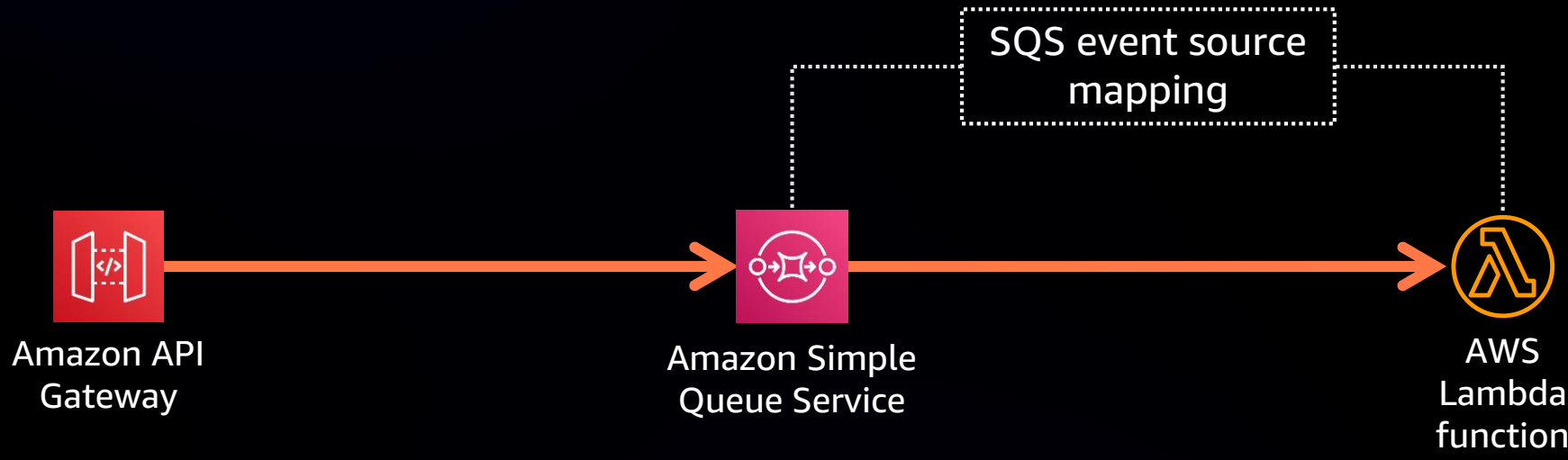
Service integrations

- Communicate directly with 100+ AWS services
- Modify requests and response payloads with Apache Velocity Templating Language (VTL)



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Making it work on AWS



Event source mapping

AWS Lambda polls the queue and invokes your function with an event containing a batch of messages

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Introducing new filters for AWS Lambda Event source mappings

New

FILTER MESSAGES BEFORE AWS LAMBDA INVOCATIONS

template.yaml

```
Events:
    Type: SQS
    Properties:
     Filters:
      - Pattern: "{\"data\": ”\sender\": {
                          “\login\": “\bls20AWS\”}
```

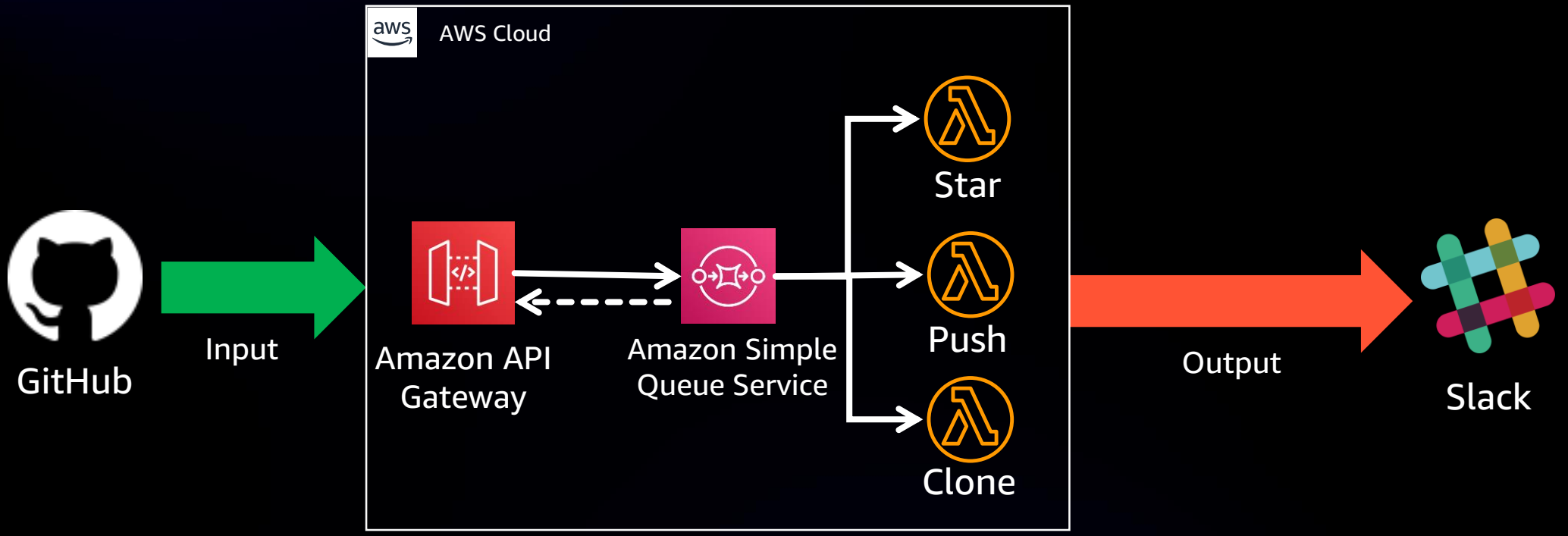
Event source mapping filtering

- Reduce the number of messages that your functions process
- Simplify code
- Reduce cost

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

How can we improve this?

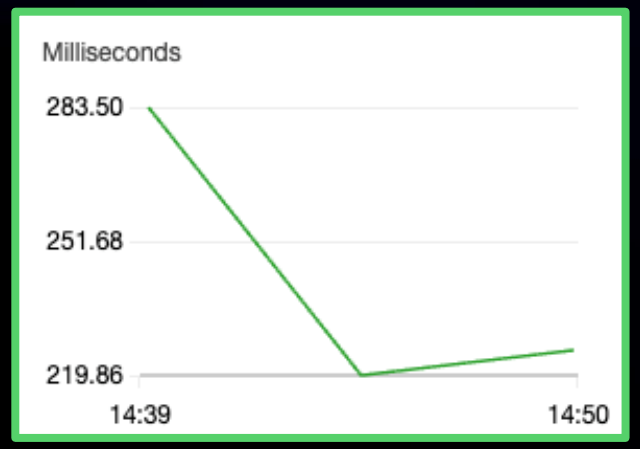


- Store input payload in Amazon SQS
- Invoke AWS Lambda functions based on event type

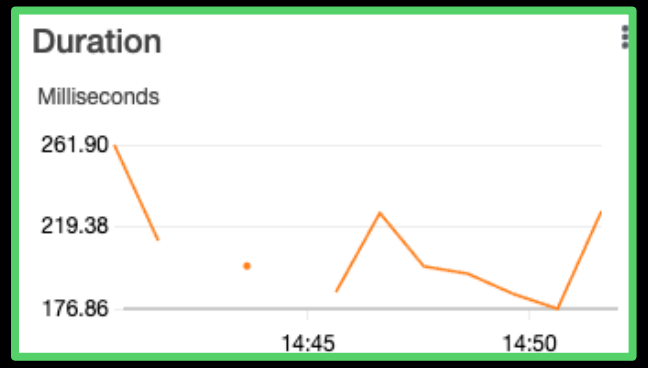
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

How did this affect the app?

Synchronous

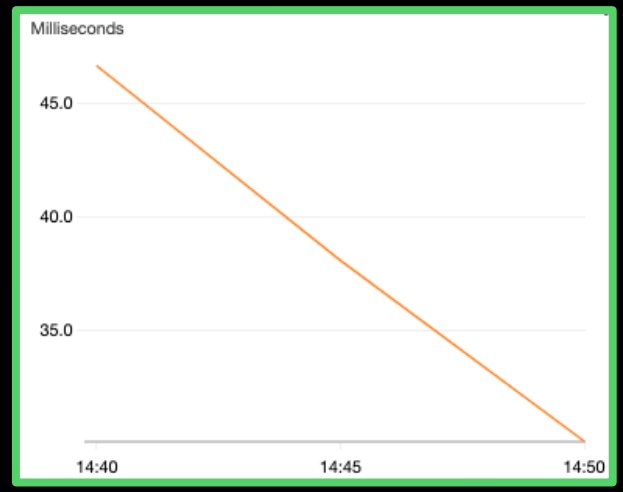


Average
API latency
252 milliseconds

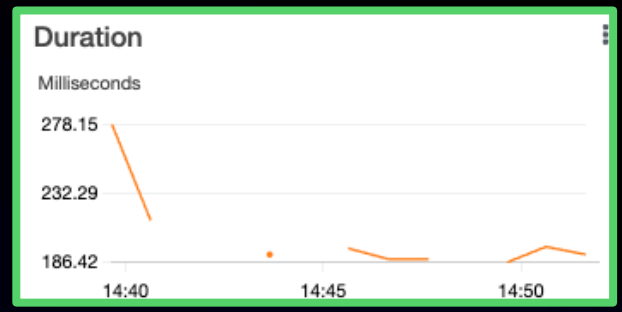


Average
AWS Lambda
function duration
206 milliseconds

Asynchronous



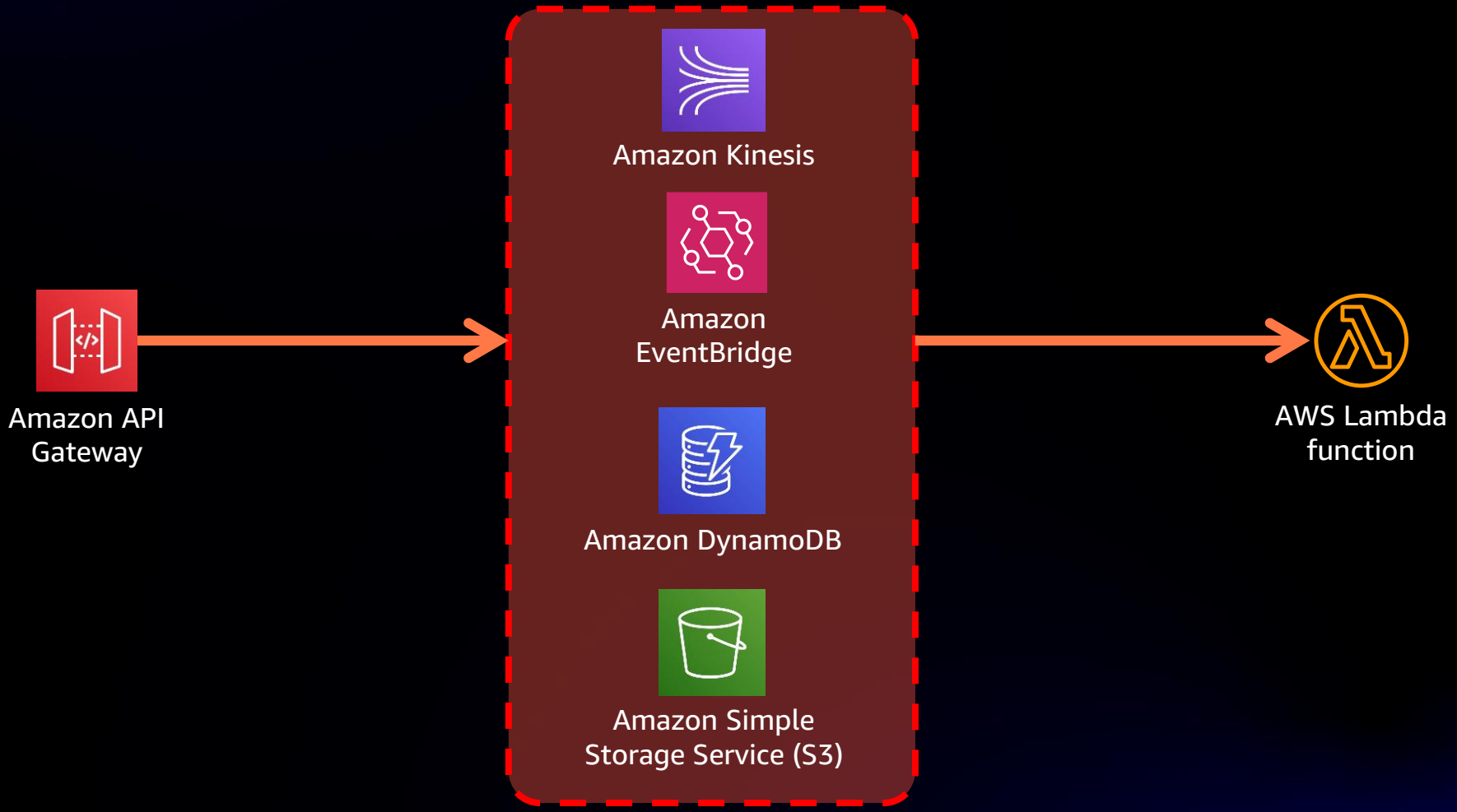
Average
API latency
39 milliseconds



Average
AWS Lambda
function duration
205 milliseconds

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Other “storage first” options



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Translation web application

Synchronous example

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

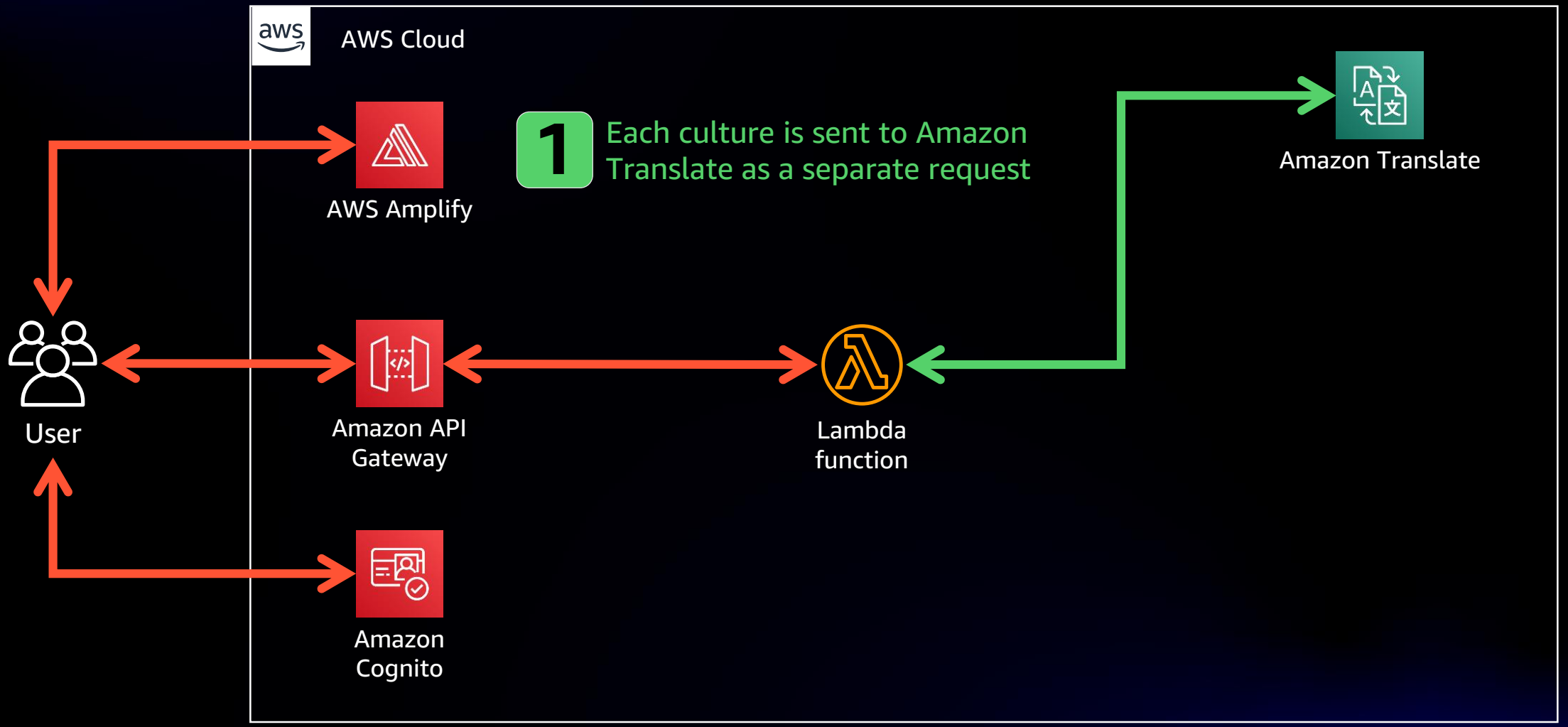
Event data

event.JSON

```
{
      "data":”Hello, welcome to my translation demo.",
      "culture":[”fr-FR","tr-TR”,“de-DE”,”fr-CA”]
}
```

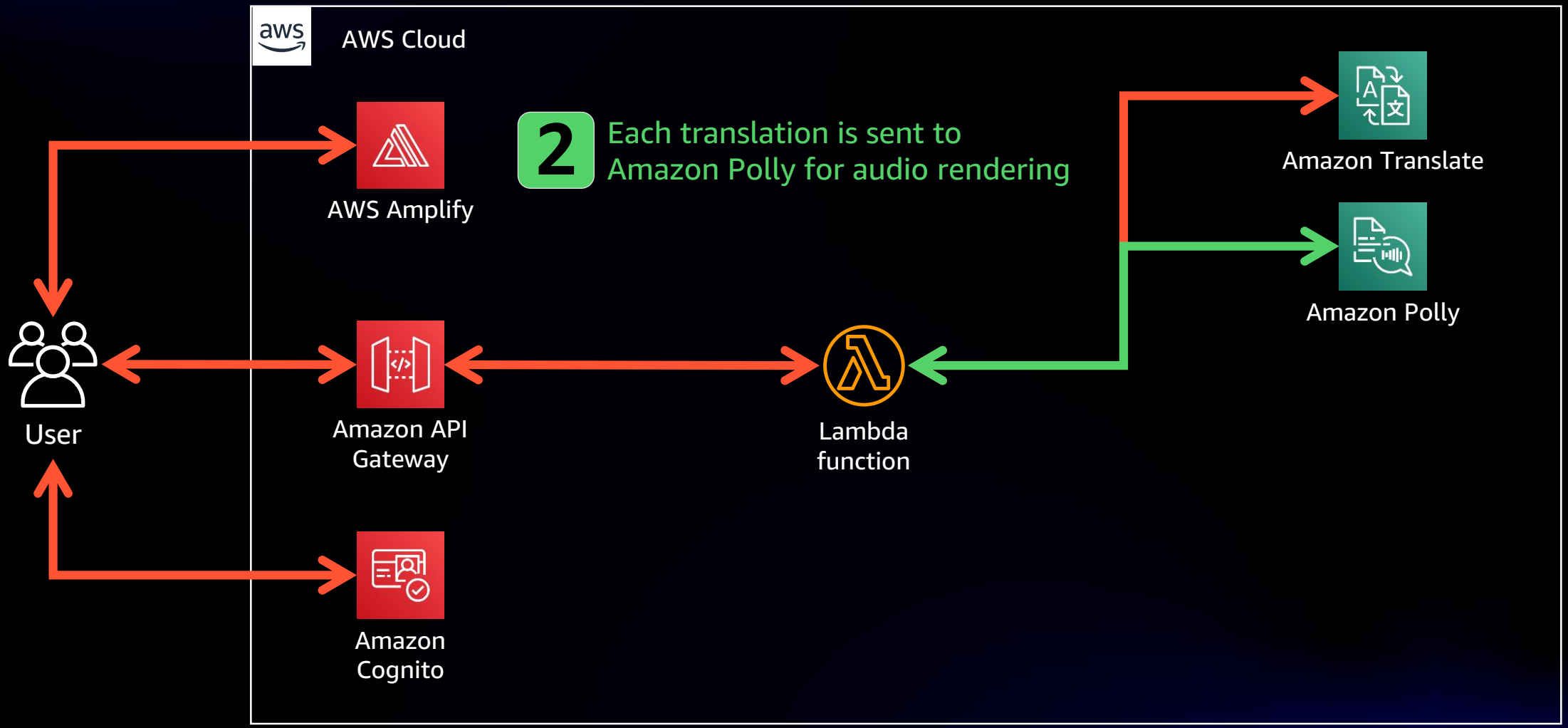
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Synchronous demo



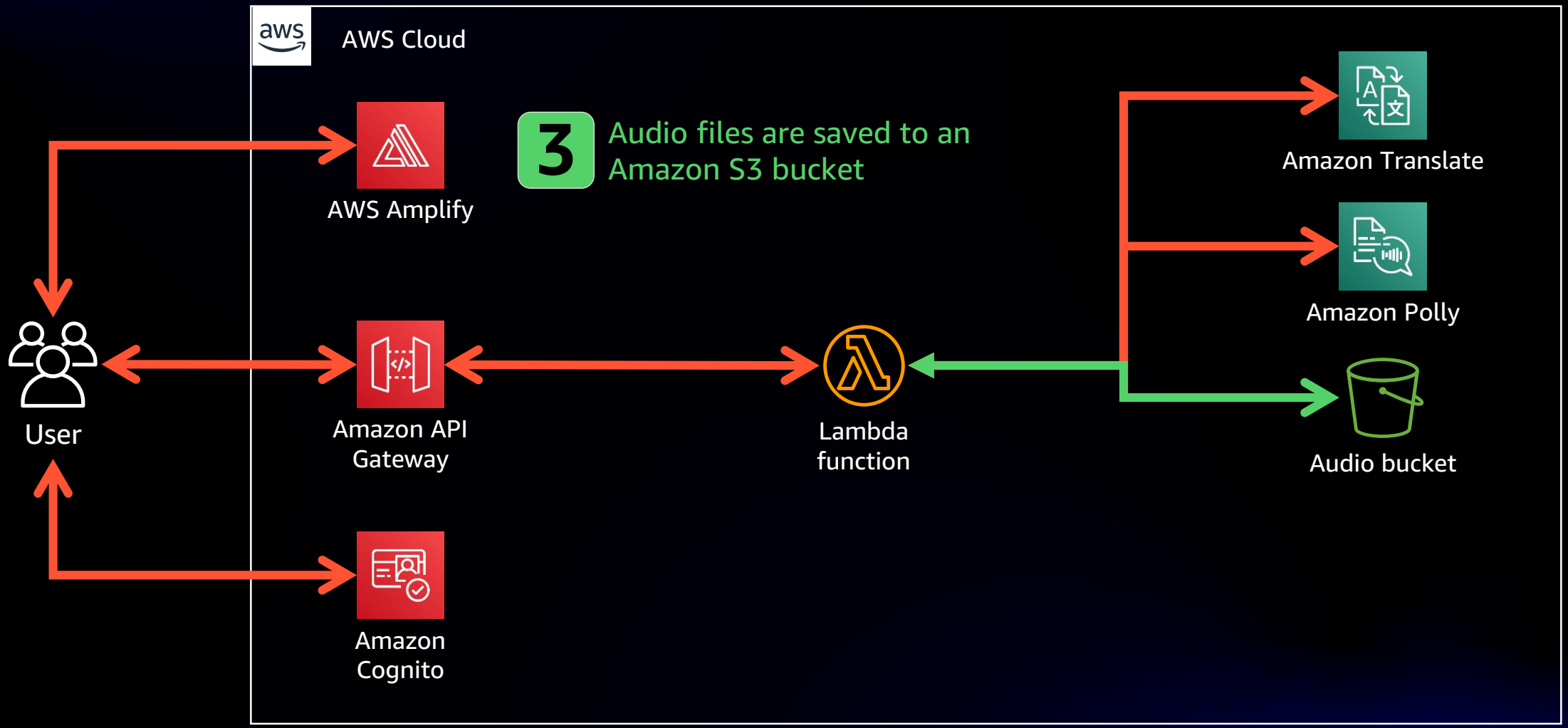
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Synchronous demo



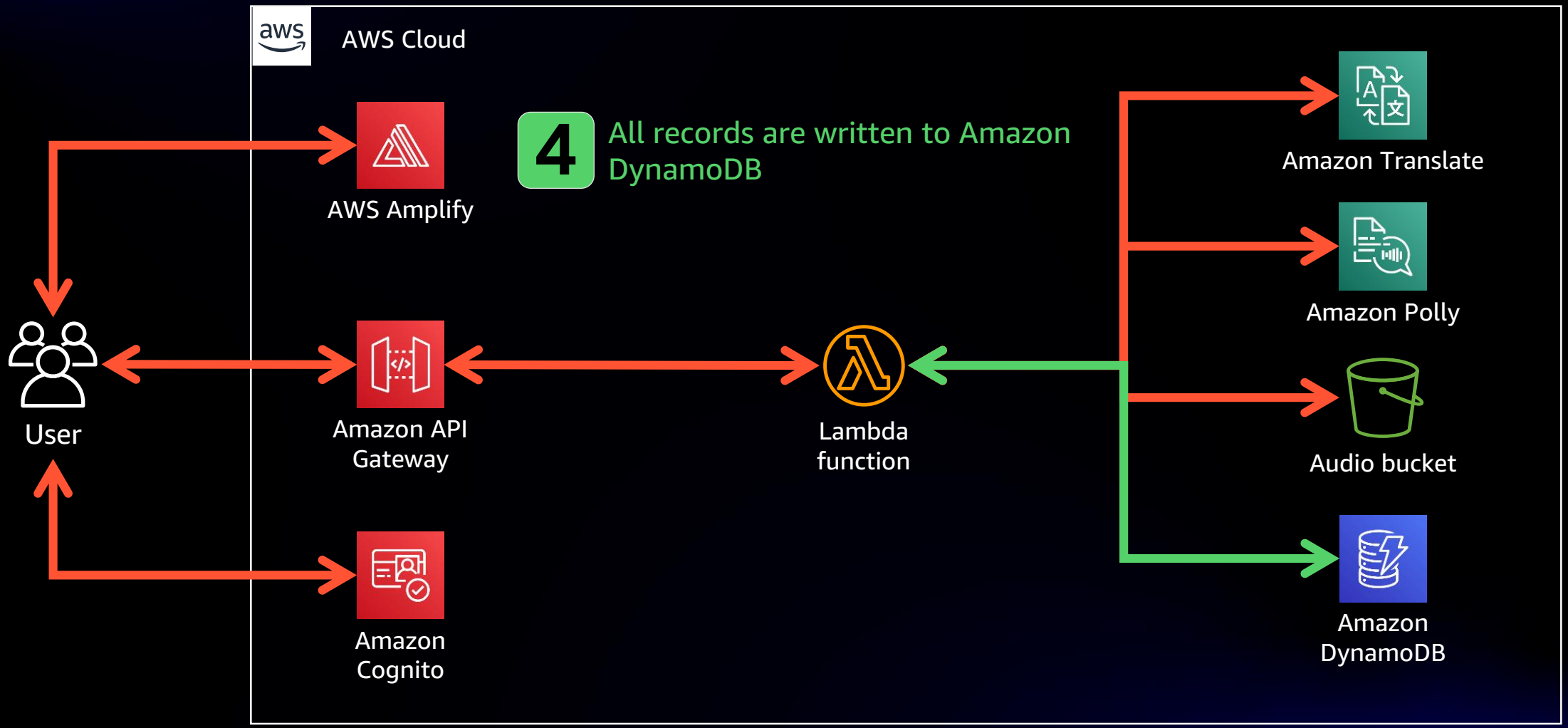
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Synchronous demo



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Synchronous demo



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Resulting data

Amazon DynamoDB

id	type	culture	data	timestamp	originalId
55de7a40-d4ba-11e9-b999-e9c6d1f6a327	audio-de-DE	de-DE	https://translator-synch-audiobucket-li8k1wox4fxm.s3-…	1568223241096	
55de7a40-d4ba-11e9-b999-e9c6d1f6a327	audio-fr-CA	fr-CA	https://translator-synch-audiobucket-li8k1wox4fxm.s3-…	1568223241096	
55de7a40-d4ba-11e9-b999-e9c6d1f6a327	audio-fr-FR	fr-FR	https://translator-synch-audiobucket-li8k1wox4fxm.s3-…	1568223241096	
55de7a40-d4ba-11e9-b999-e9c6d1f6a327	audio-tr-TR	tr-TR	https://translator-synch-audiobucket-li8k1wox4fxm.s3-…	1568223241096	
55de7a40-d4ba-11e9-b999-e9c6d1f6a327	original	[{ "S" : "fr-FR" }, { "S" : "tr-…	Hello, welcome to my translation demo.	1568223238116	original-55de7a40-d4ba-11e9-b999-e9c6d1f6a
55de7a40-d4ba-11e9-b999-e9c6d1f6a327	translation-de-DE	de-DE	Hallo, willkommen zu meiner Übersetzungsdemo.	1568223239938	
55de7a40-d4ba-11e9-b999-e9c6d1f6a327	translation-fr-CA	fr-CA	Bonjour, bienvenue à ma démo de traduction.	1568223239938	
55de7a40-d4ba-11e9-b999-e9c6d1f6a327	translation-fr-FR	fr-FR	Bonjour, bienvenue à ma démo de traduction.	1568223239938	
55de7a40-d4ba-11e9-b999-e9c6d1f6a327	translation-tr-TR	tr-TR	Merhaba, çeviri demoma hoş geldiniz.	1568223239938	

Amazon S3

Name	Last modified	Size	Storage class
55de7a40-d4ba-11e9-b999-e9c6d1f6a327_de-DE_Marlene.mp3	Sep 11, 2019 11:34:01 AM GMT-0600	17.3 KB	Standard
55de7a40-d4ba-11e9-b999-e9c6d1f6a327_fr-CA_Chantal.mp3	Sep 11, 2019 11:34:02 AM GMT-0600	16.4 KB	Standard
55de7a40-d4ba-11e9-b999-e9c6d1f6a327_fr-FR_Celine.mp3	Sep 11, 2019 11:34:02 AM GMT-0600	16.6 KB	Standard
55de7a40-d4ba-11e9-b999-e9c6d1f6a327_tr-TR_Filiz.mp3	Sep 11, 2019 11:34:02 AM GMT-0600	15.8 KB	Standard

Viewing 1 to 4

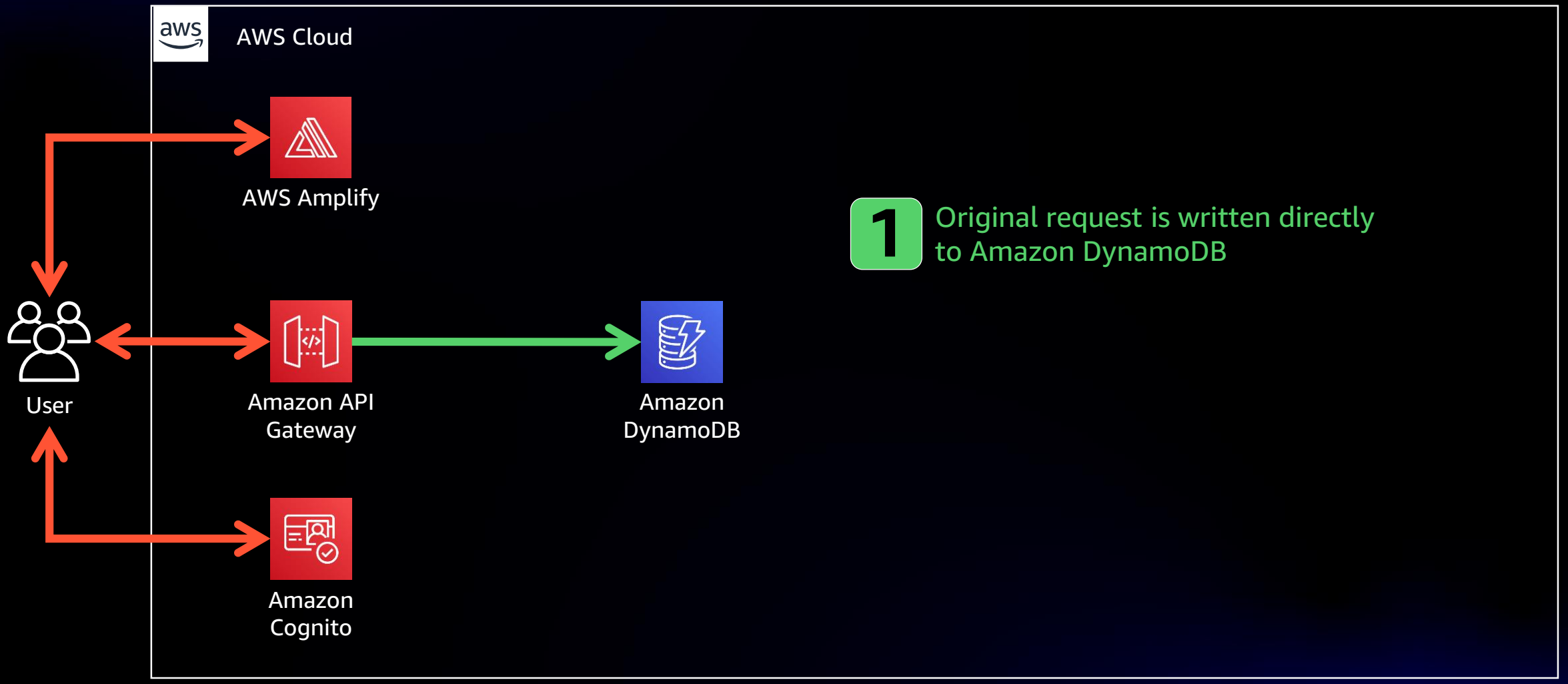
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Translation web application

Asynchronous example

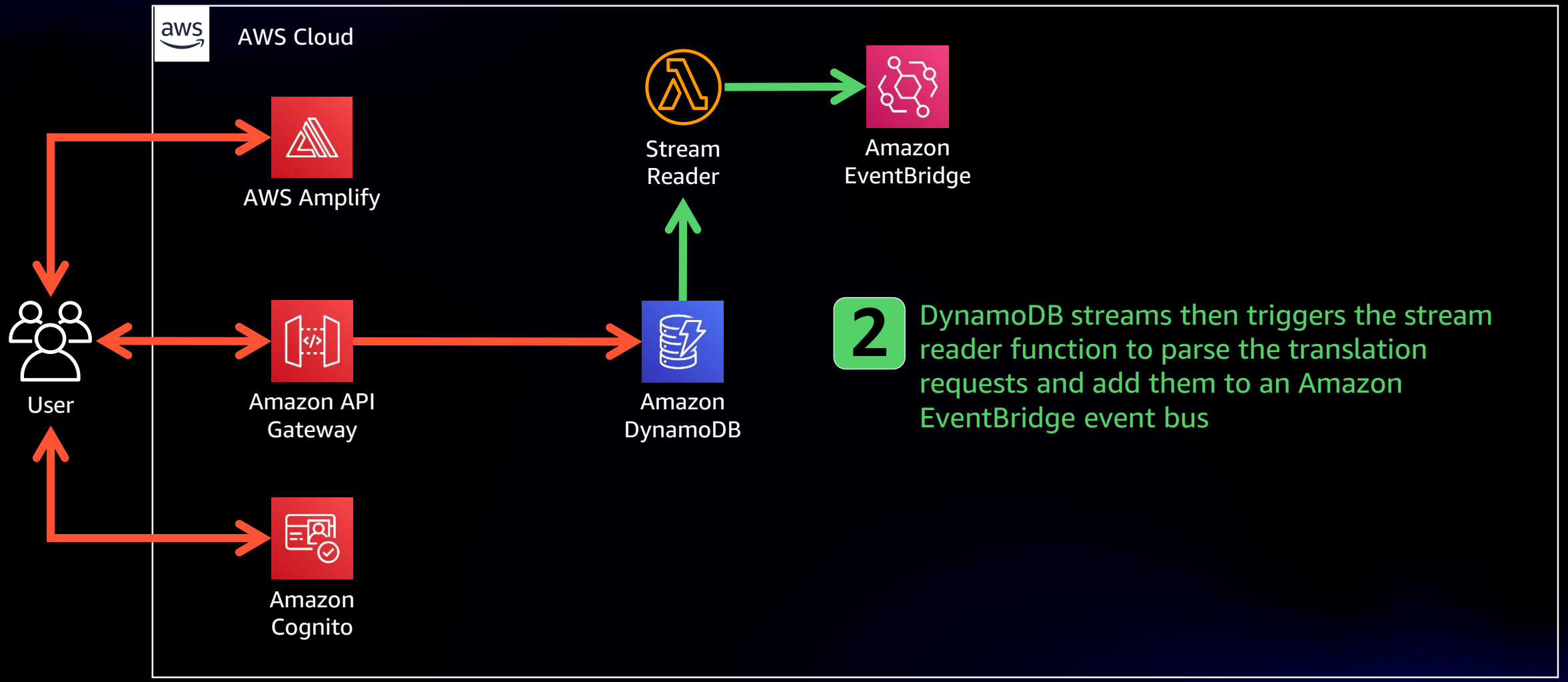
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Asynchronous demo



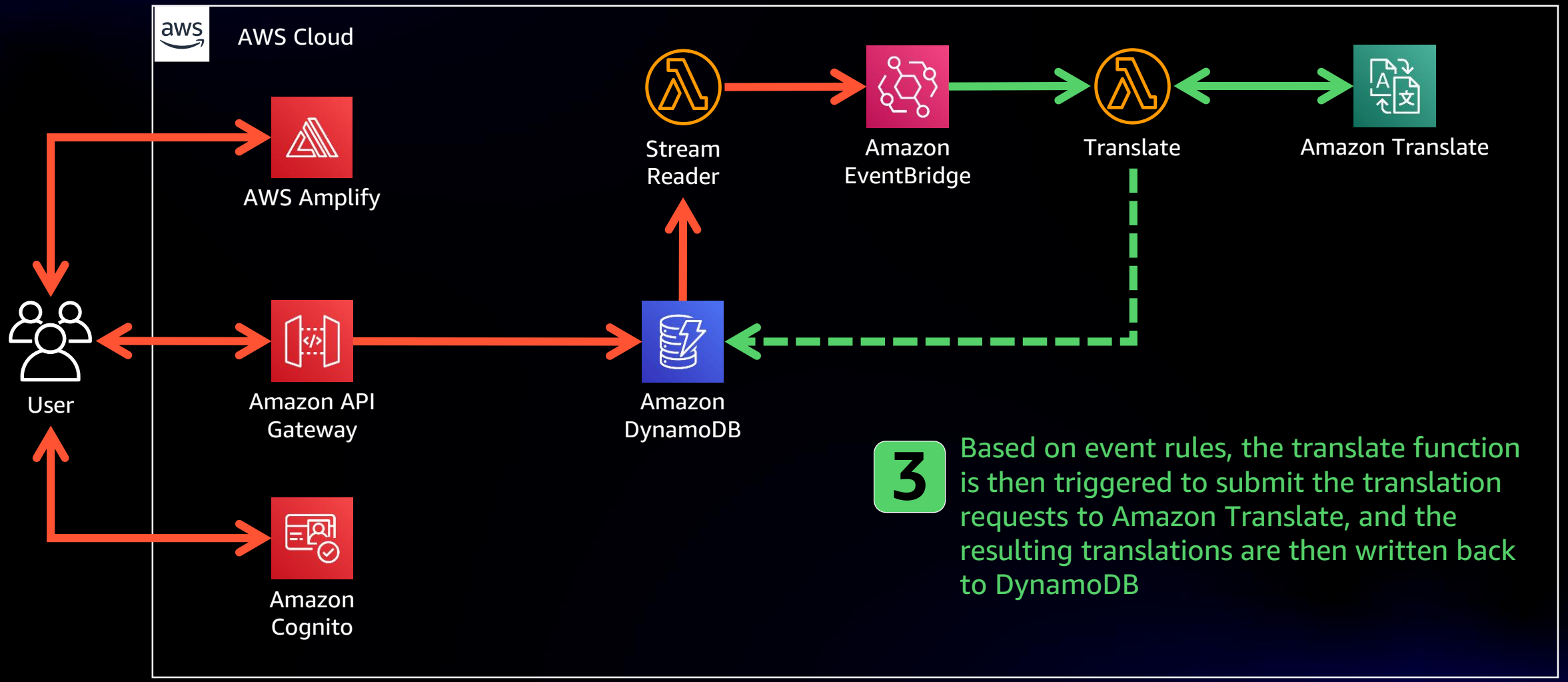
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Asynchronous demo



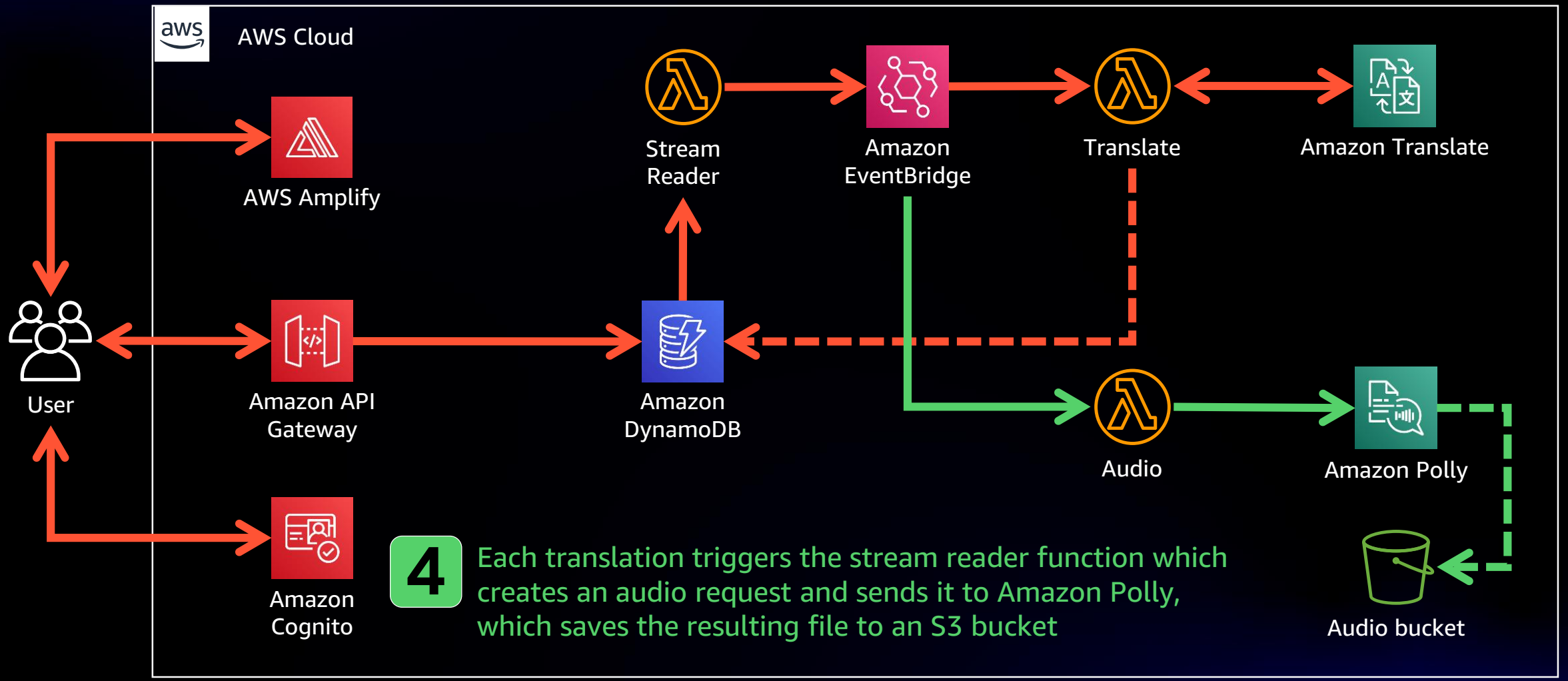
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Asynchronous demo



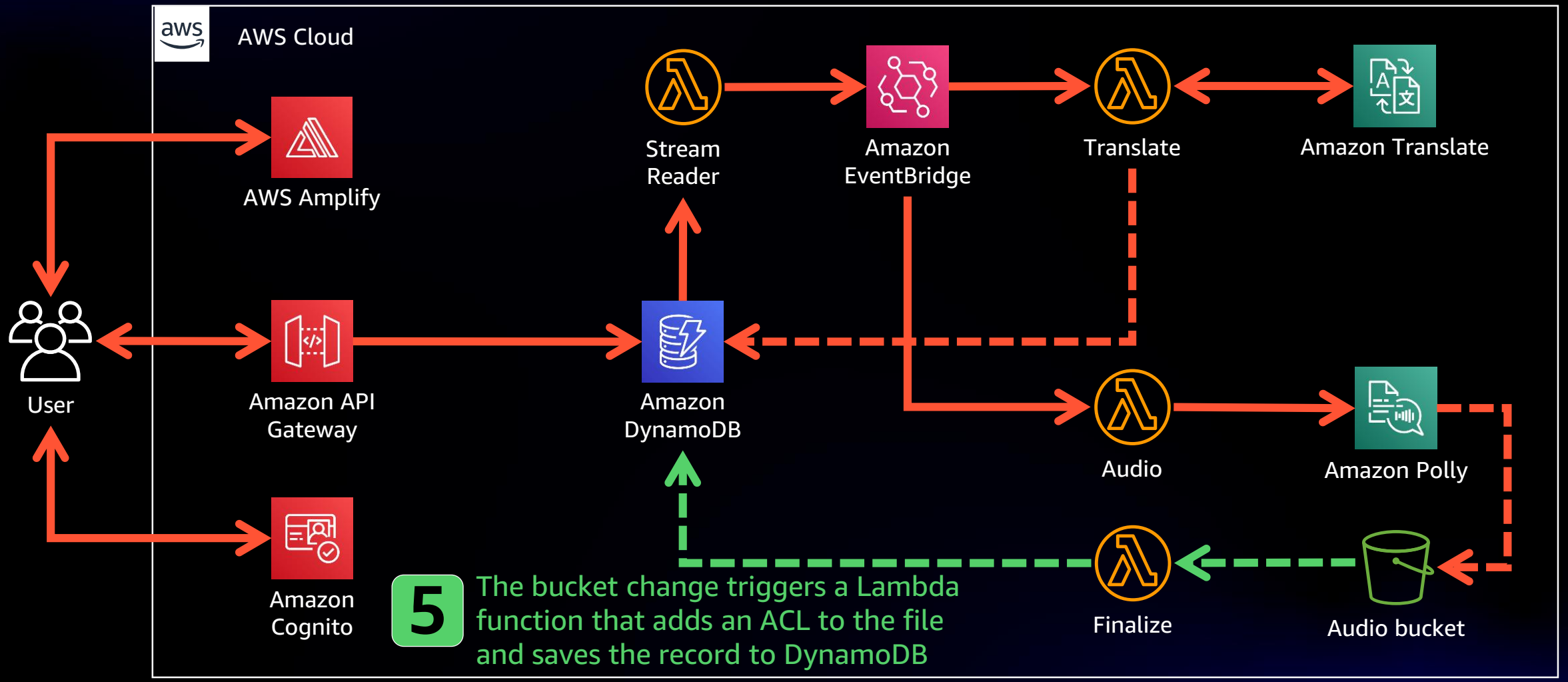
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Asynchronous demo



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Asynchronous demo



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Synchronous and Asynchronous request behaviors

	Synchronous requests	Asynchronous requests
Blocking time	Caller waits until the end of processing for a response	
Costs	Waiting may incur cost	
Latency	Downstream slowness or outages affects the overall request	
Payload transport	Passes payloads between steps	
Durability	Failure affects entire request	
Implementation	Easy to implement	

Why use Asynchronous?

Easier to extend.
Faster response times.
Less code.
Increased reliability.

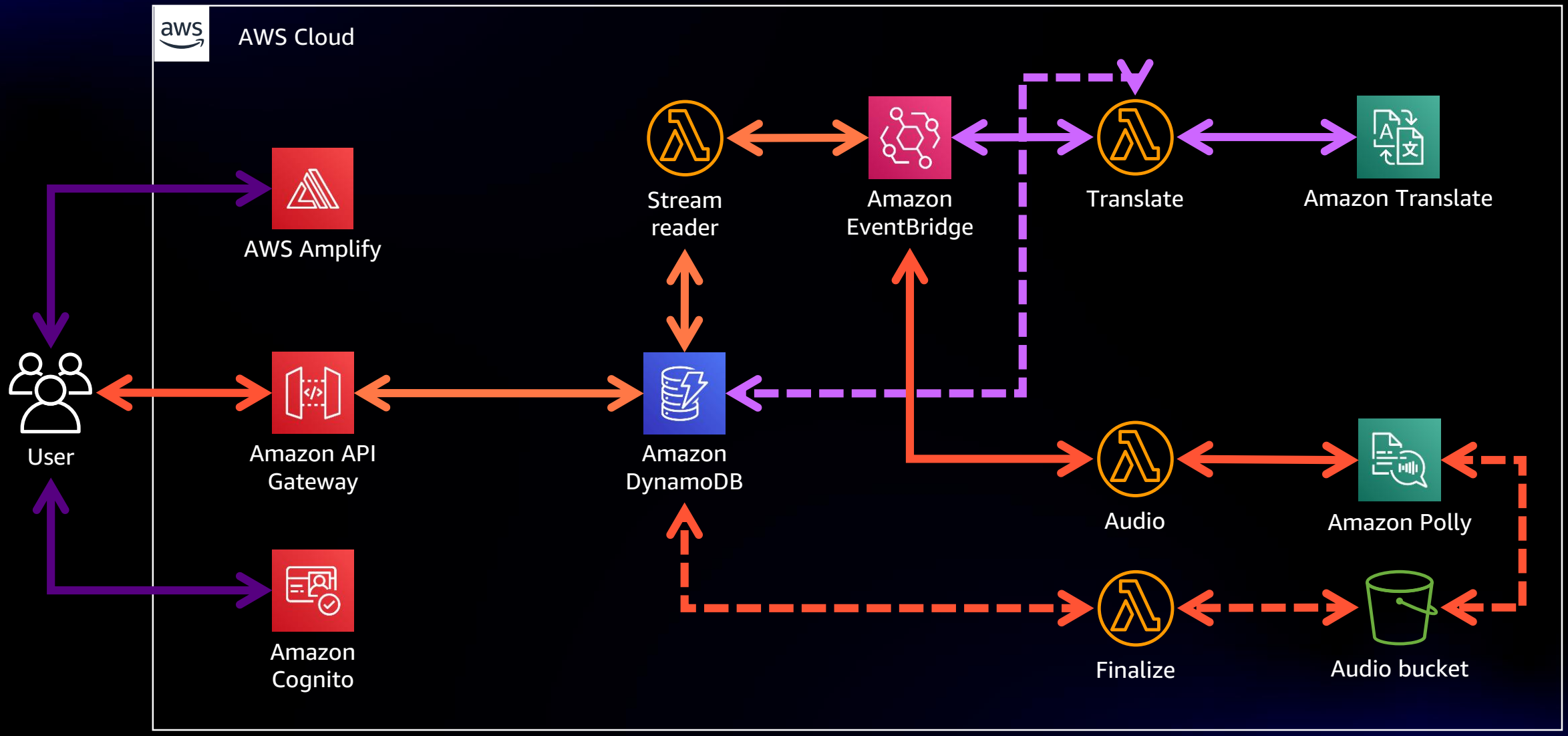
aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Adding more reliability

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

More reliability



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

More reliability



Stream reader



Translate



Audio

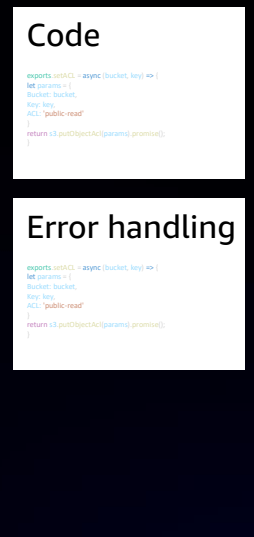


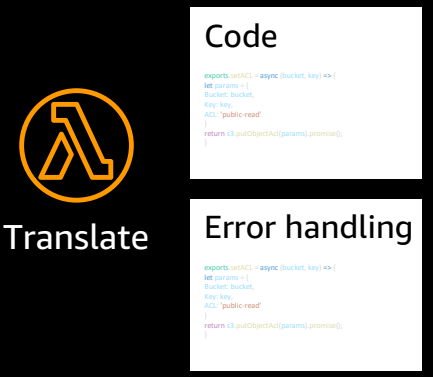
Finalize

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

More reliability









© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

More reliability





Stream reader



Translate







Audio







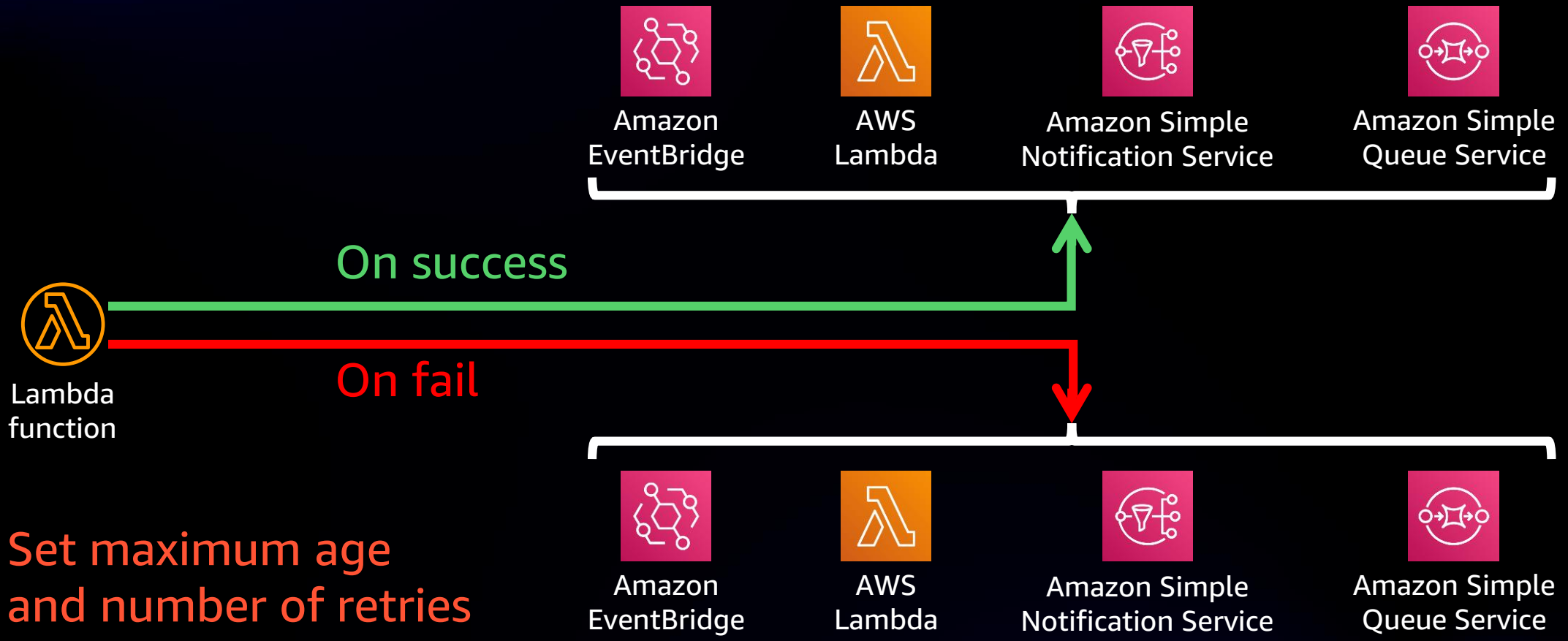
Errors



Finalize

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

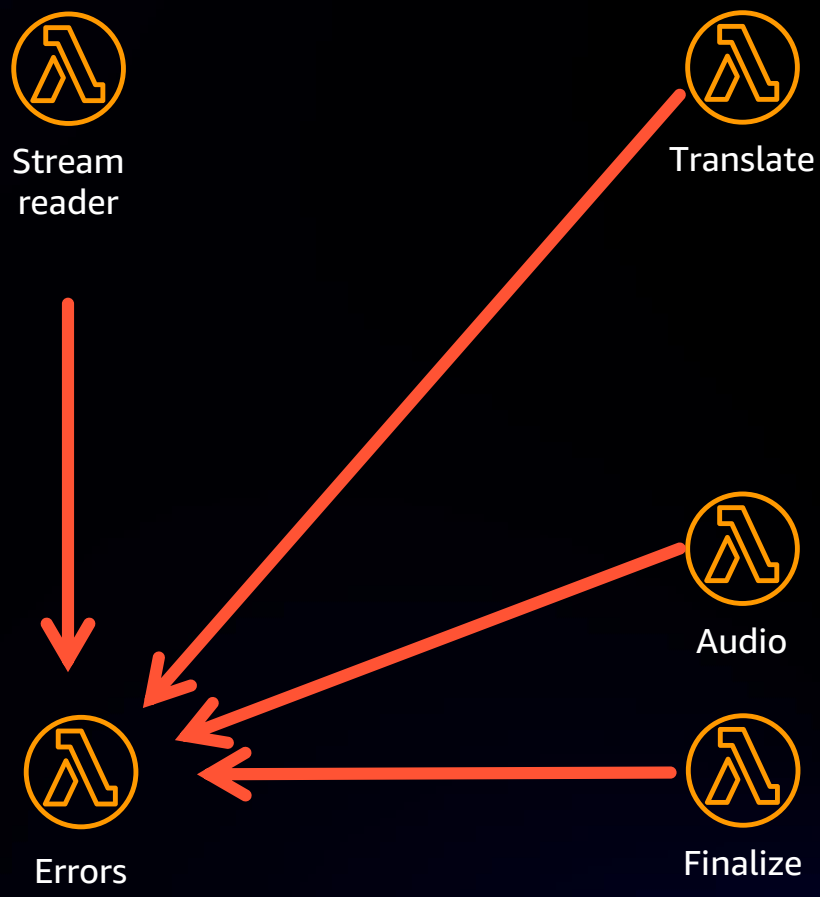
More reliability: Lambda destinations



Set maximum age and number of retries

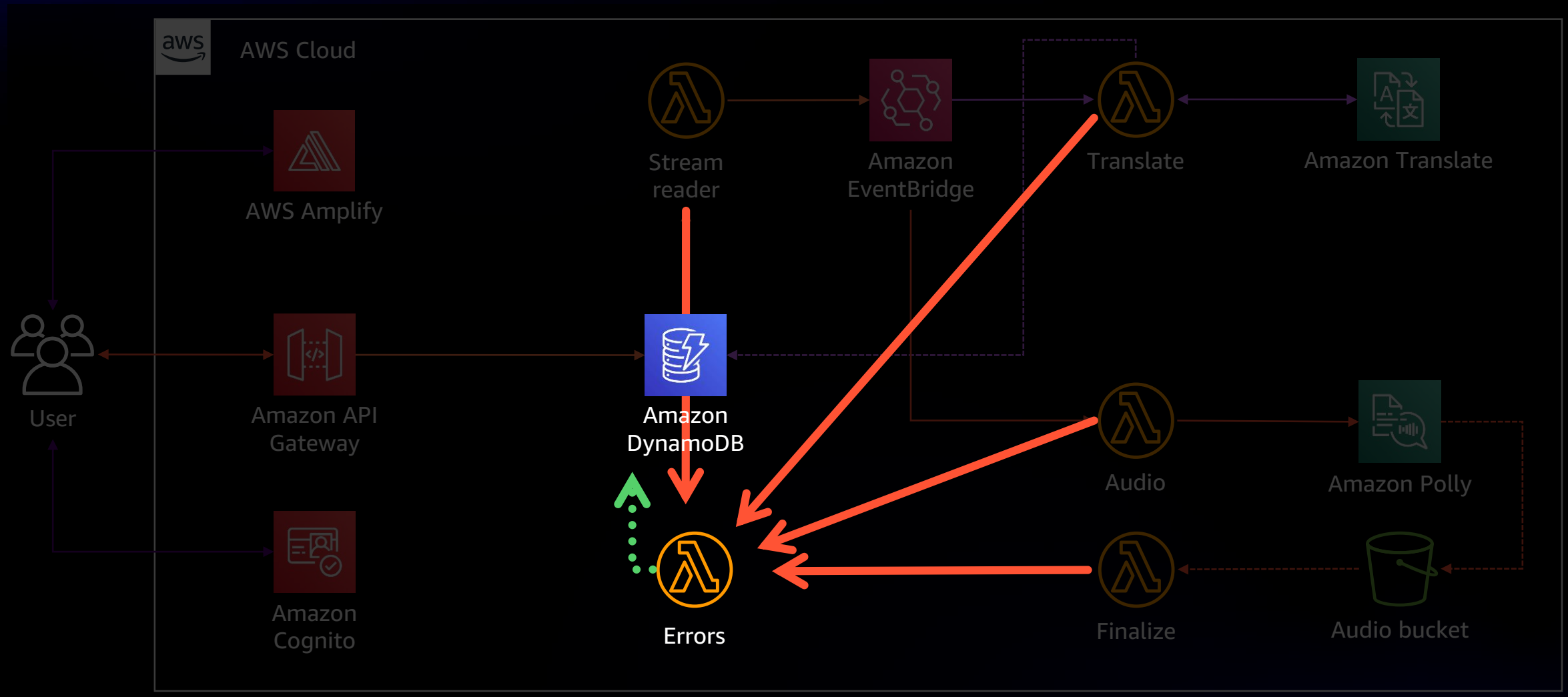
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

More reliability



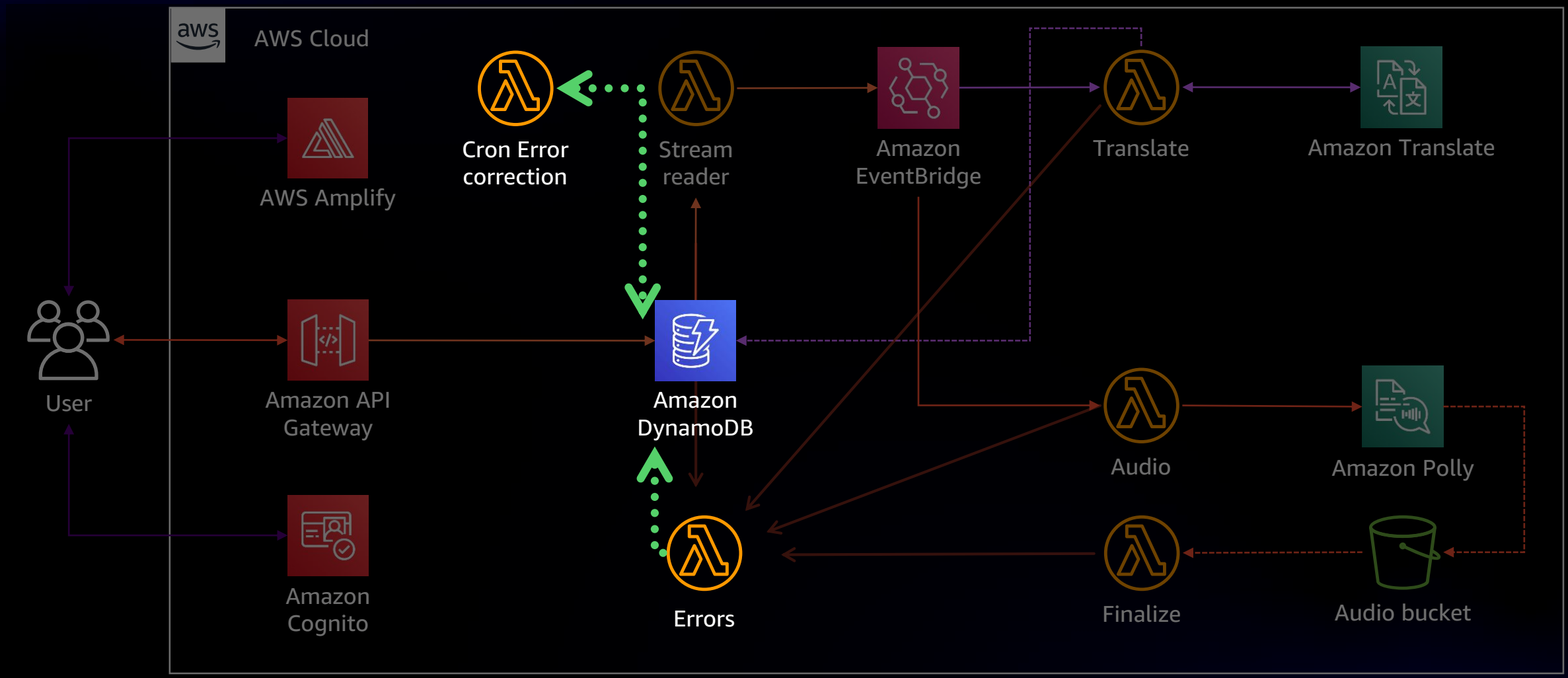
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Asynchronous demo



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Asynchronous demo



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

The lift and shift

Auth
Caching logic
Throttling logic
Routing



Backend
Error handling
Retry logic
Front end

Storage



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

The migration

Offload processes to managed services



Front end

Auth
Caching logic



Throttling logic
Routing

Backend

Error handling



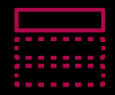
Bus



Queue



Workflow



Pub/sub

Retry logic











Storage



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Castle and moat vs. Zero Trust

From

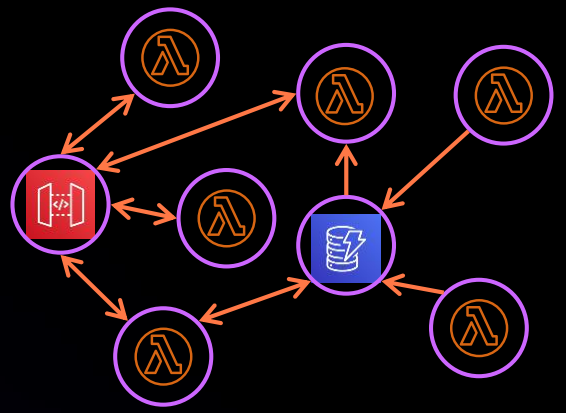
Castle and moat



Protecting network perimeters with firewalls, proxy servers, honeypots, and other intrusion prevention tools

to

Zero Trust



Trust “no one” by default

Build micro-perimeters around each resource to enforce strict verification of every person or service

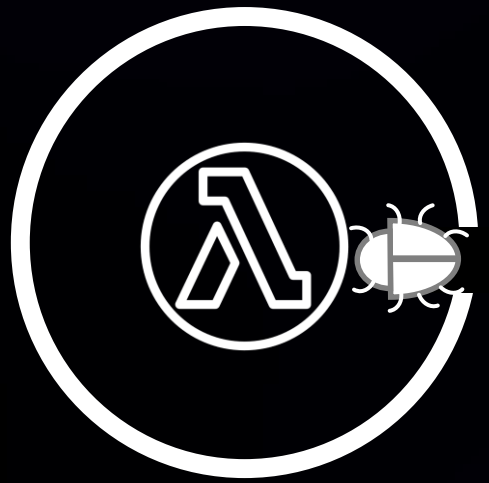
aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Castle and moat vs. Zero Trust

From

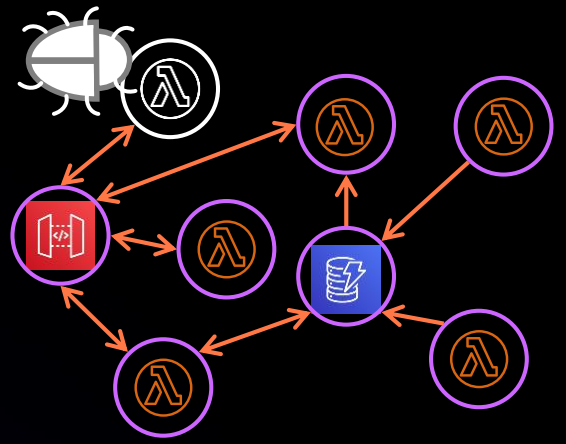
Castle and moat



If an unauthorized user gains access to the network, they have access to everything inside

to

Zero Trust



If an unauthorized user gains entry, they are limited to only the service within that micro-perimeter

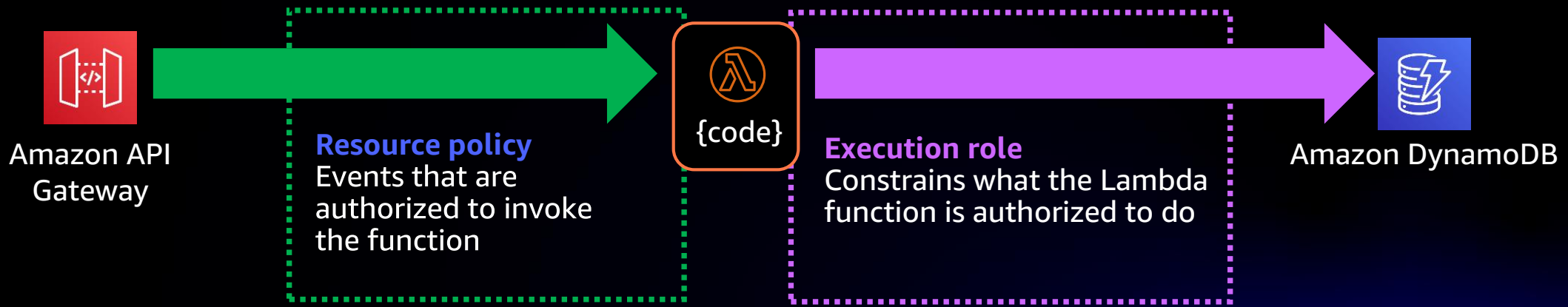
aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Implementing Zero Trust

DECOUPLE BUSINESS LOGIC FROM SECURITY POSTURE WITH ROLES AND PERMISSIONS

- Use AWS IAM resource policies and execution roles
- Default to deny – never use * permissions



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Implementing Zero Trust

DECOUPLE BUSINESS LOGIC FROM SECURITY POSTURE WITH ROLES AND PERMISSIONS

- Use AWS IAM resource policies and execution roles
- Default to deny – never use * permissions

```
"Action": [
  "lambda:InvokeFunction"
]
```

Resource policy
Events that are authorized to invoke the function




```
"Effect": "Allow",
"Action": "dynamodb:PutItem",
"Resource": "arn:aws:dynamodb
```

Execution role
Constrains what the Lambda function is authorized to do

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.



serverlesspresso

A re:Invent coffee bar built on serverless technologies
by the serverless Developer Advocate team

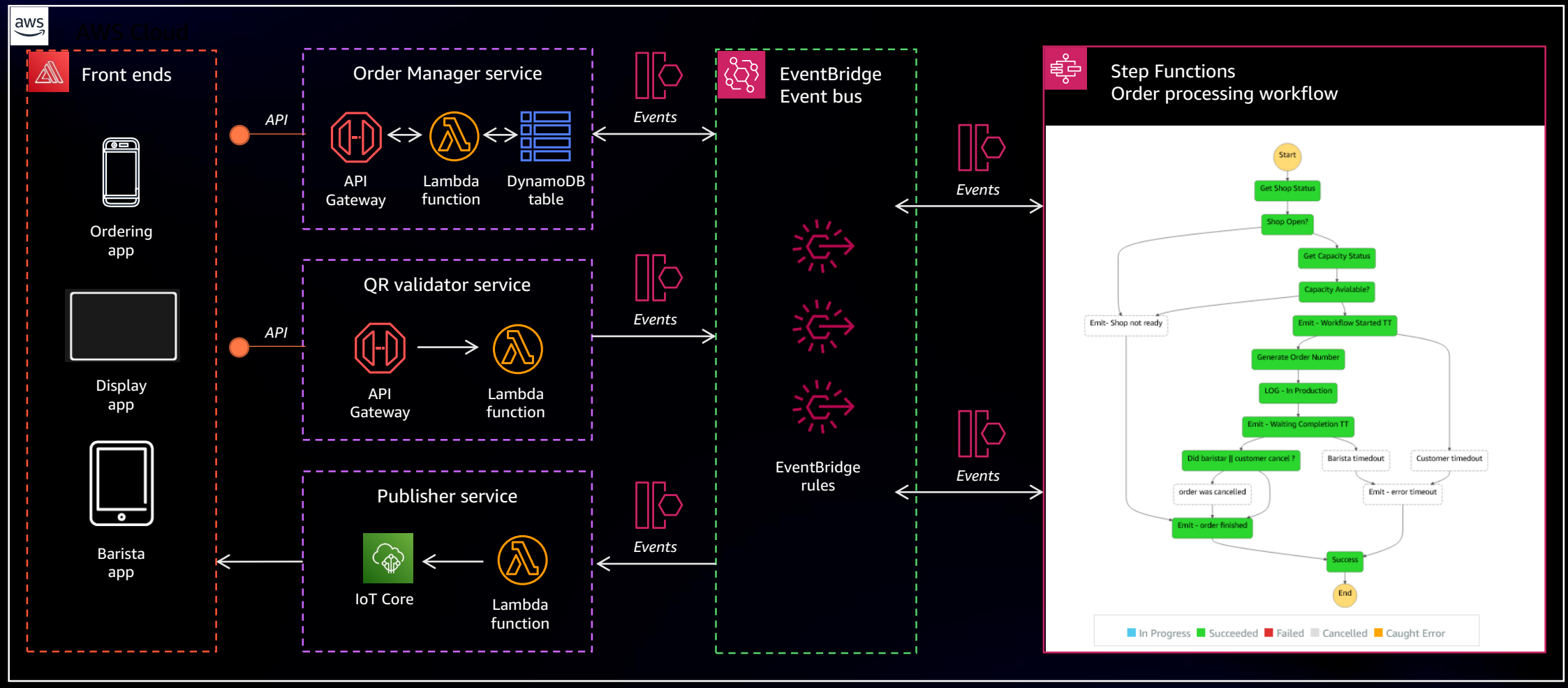
Developer
Lounge at
The Venetian

Monday **4pm – 7pm**
Tuesday **10am – 6pm**
Wednesday **10am – 6pm**
Thursday **10am – 4pm**

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

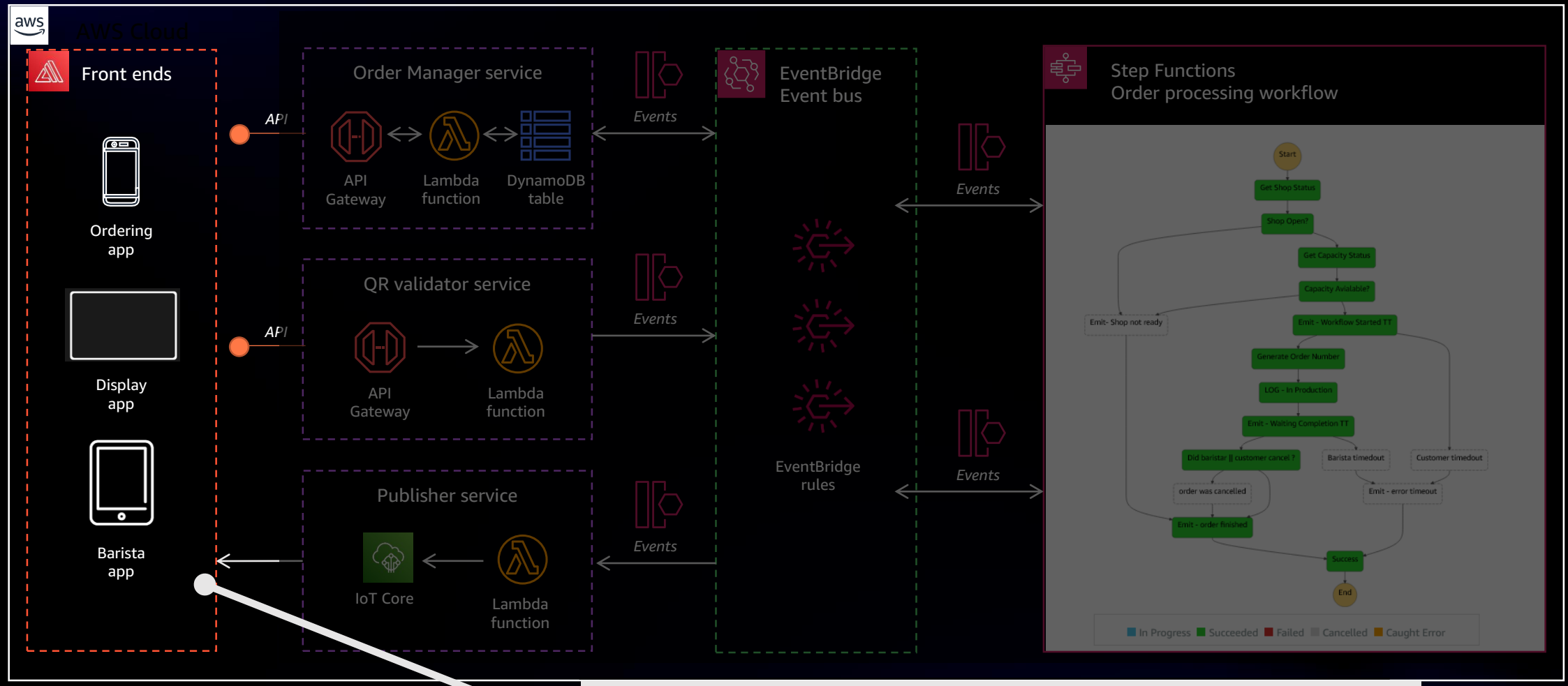
How it works



serverlesspresso

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

How it works

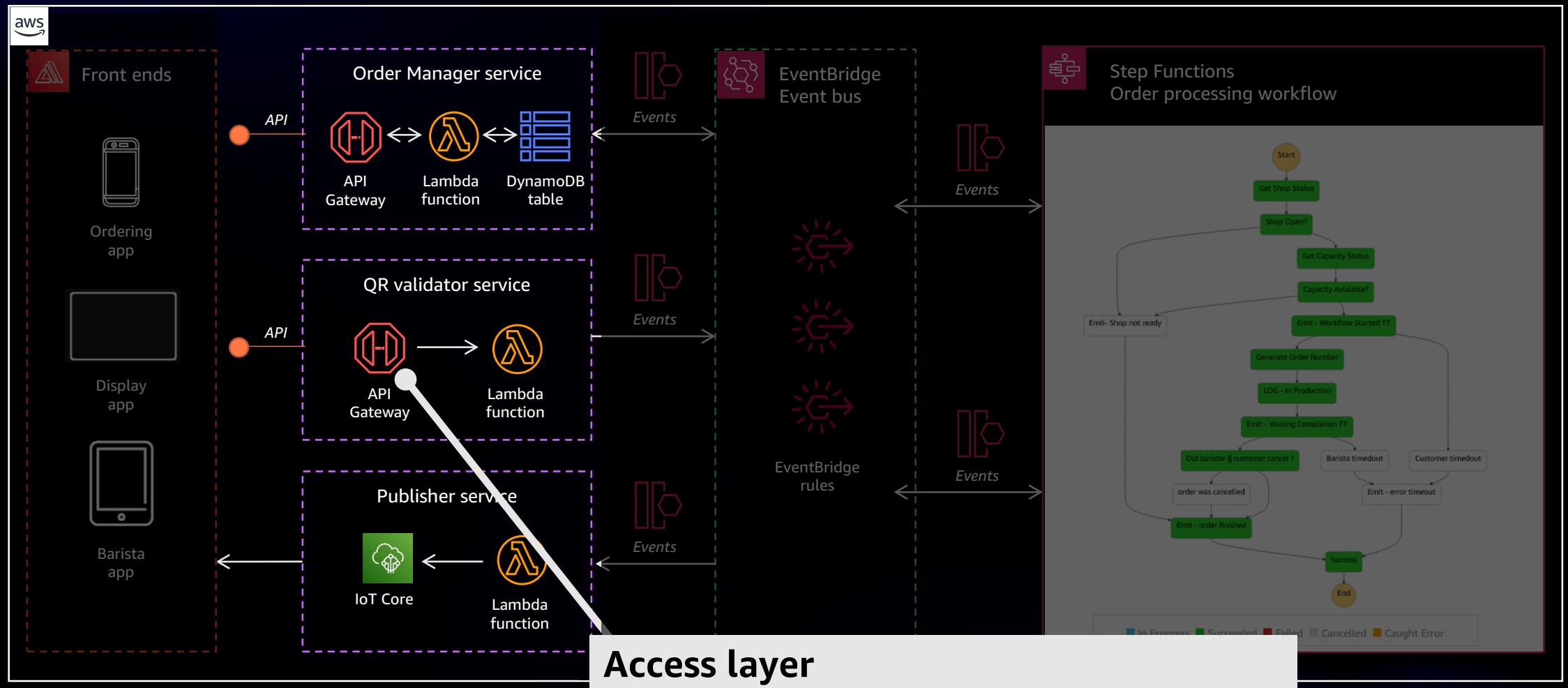


Front ends
3 web apps hosted on AWS Amplify Console

serverlesspresso

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

How it works

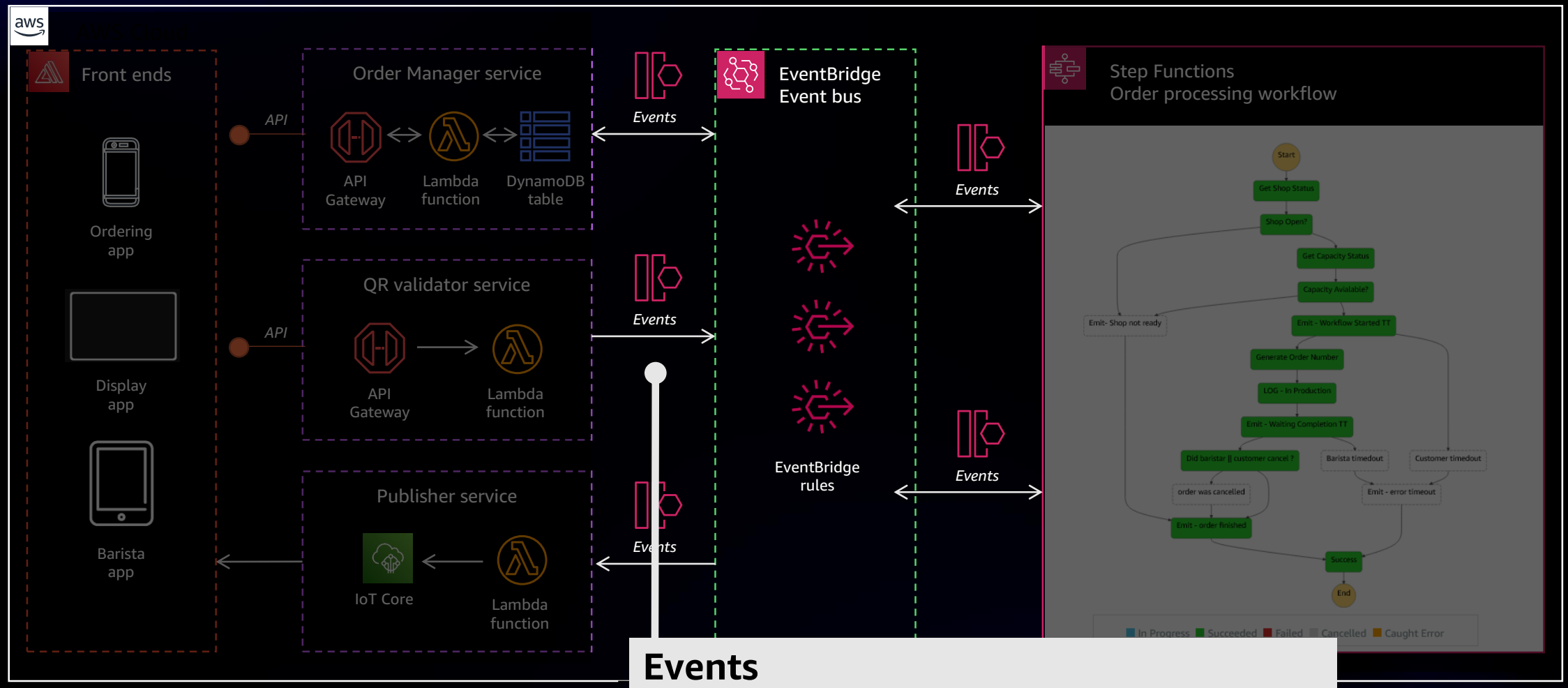


Access layer
APIs provide access to business logic

serverlesspresso

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

How it works

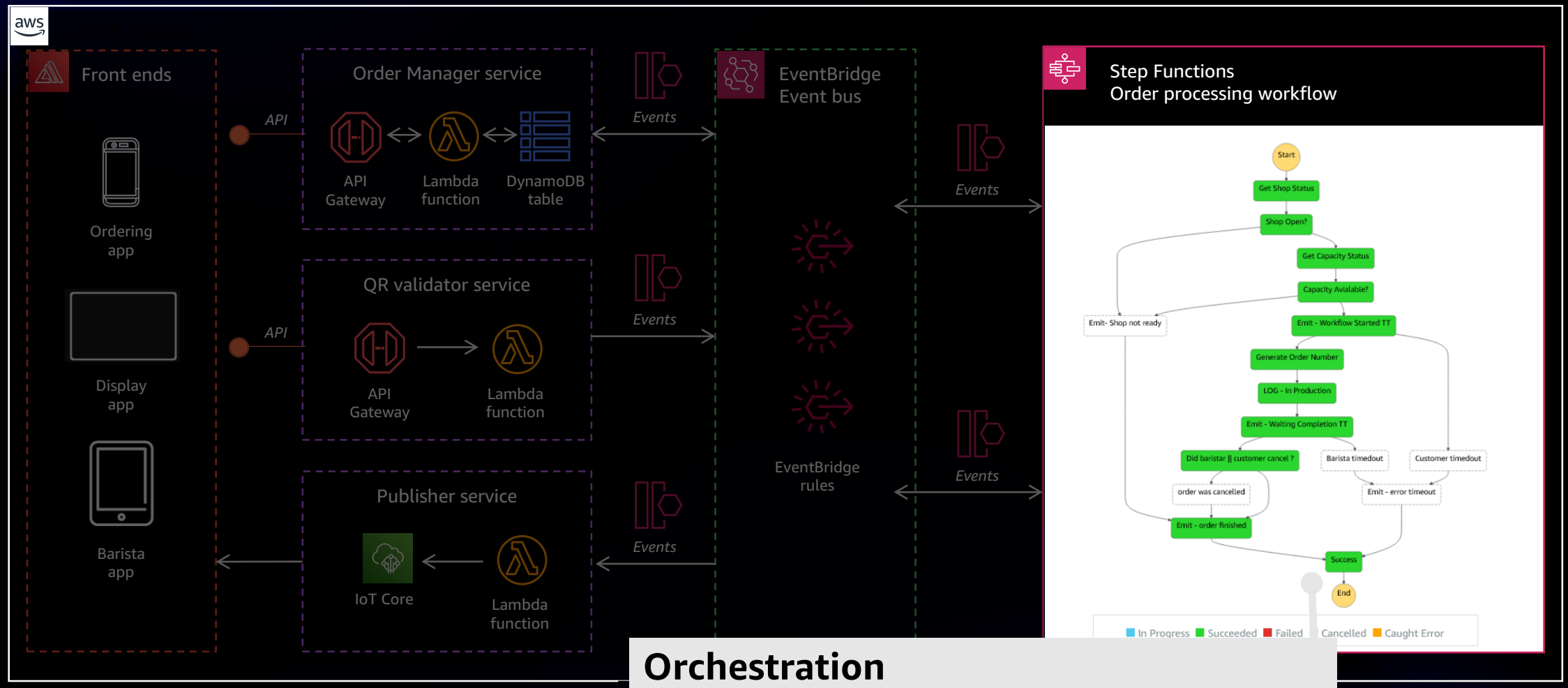


Events
Serverless event bus delivers messages about the state of each order

serverlesspresso

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

How it works

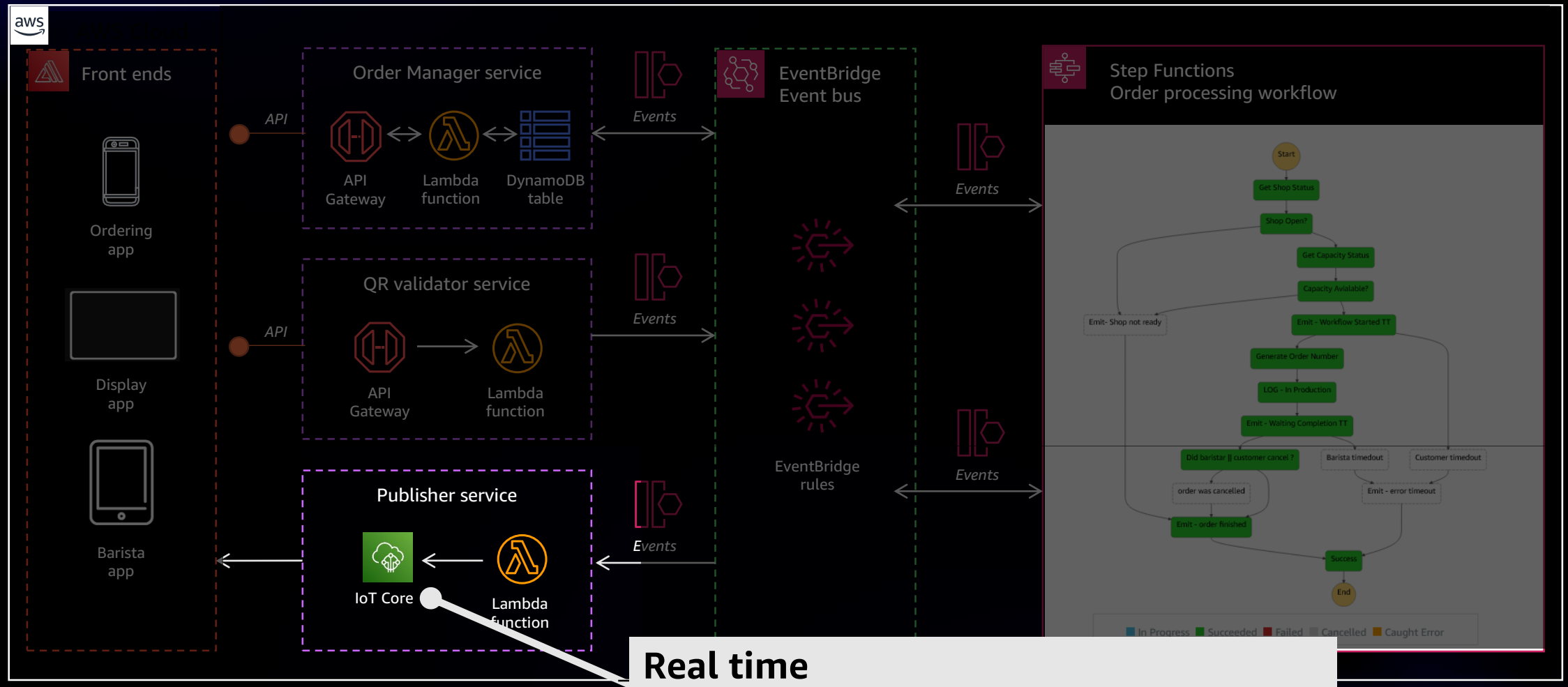


Orchestration
AWS Step Functions orchestrates each order from start to completion

serverlesspresso

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

How it works



Real time
IoT topic updates web front ends

serverlesspresso

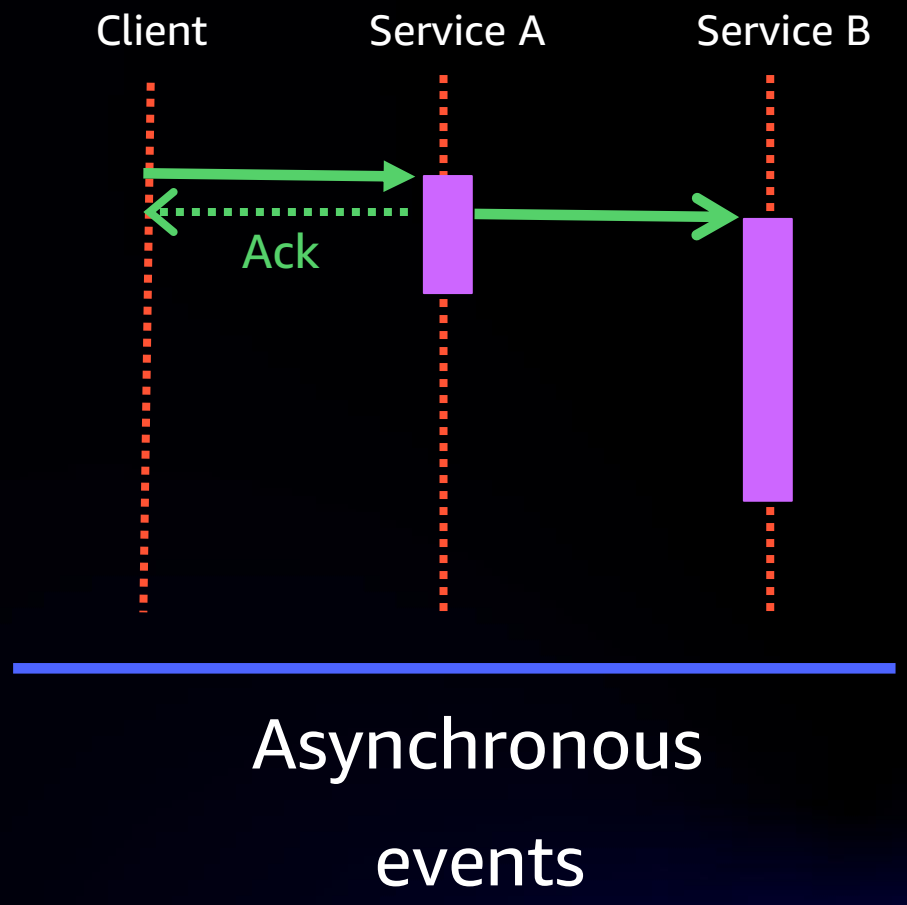
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Handling response values and state for asynchronous requests

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Handling response values and state for asynchronous requests

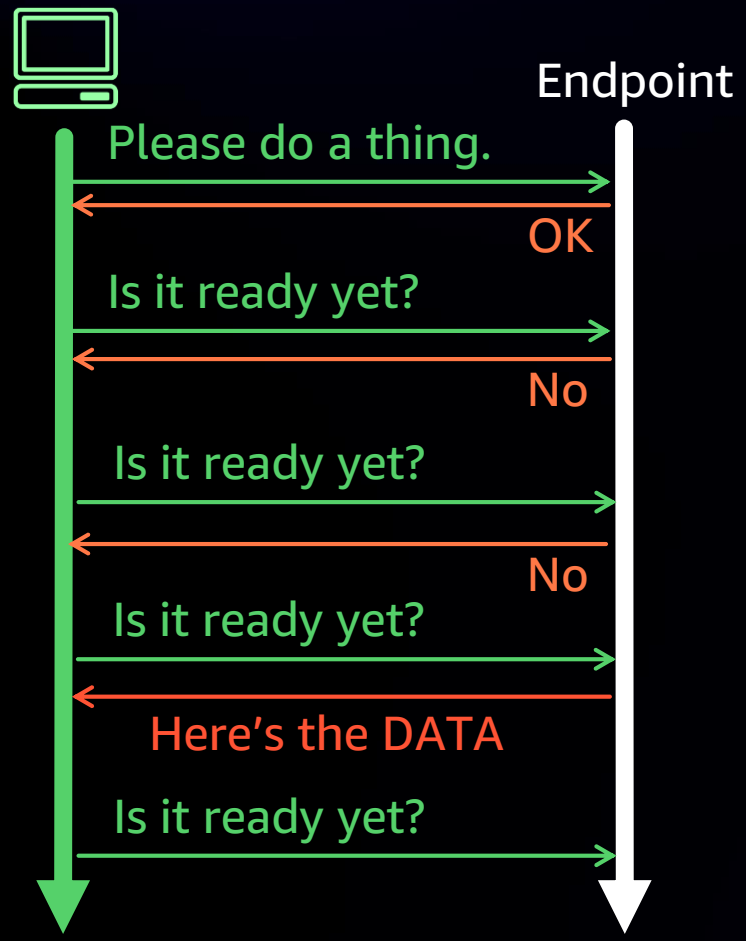
No return path to provide further information beyond the initial acknowledgement



aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

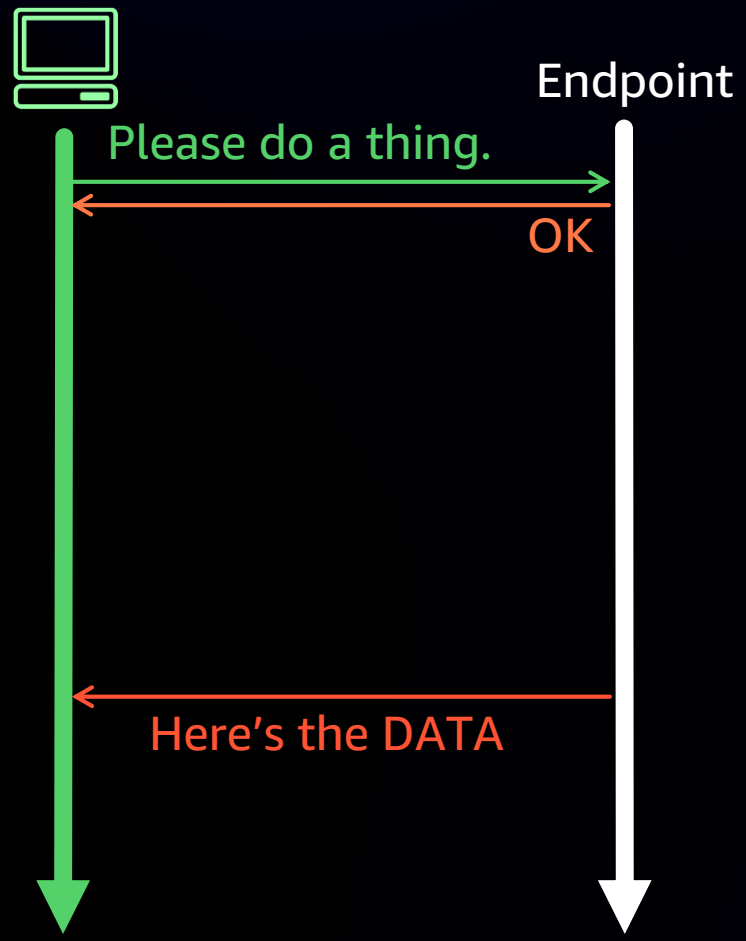
Tracking an inflight request: Polling



- Initial request returns a tracking identifier
- Create a second API endpoint for the front end 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 front-end notification

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Tracking an inflight request: WebSocket

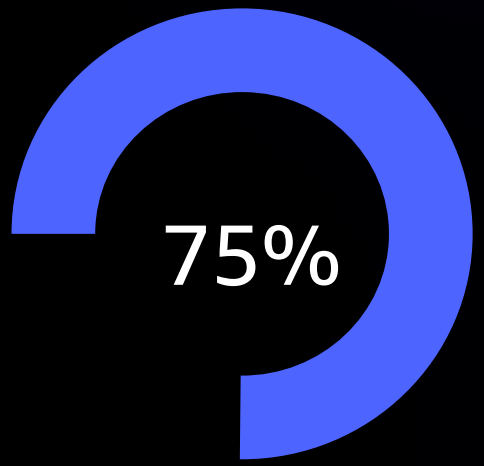


- A bidirectional connection between the front end client and the backend service
- Your backend services can continue to send data back to the client by using a WebSocket connection
- Closer to real time
- Reduces the number of messages between the client and backend system
- Often more complex to implement

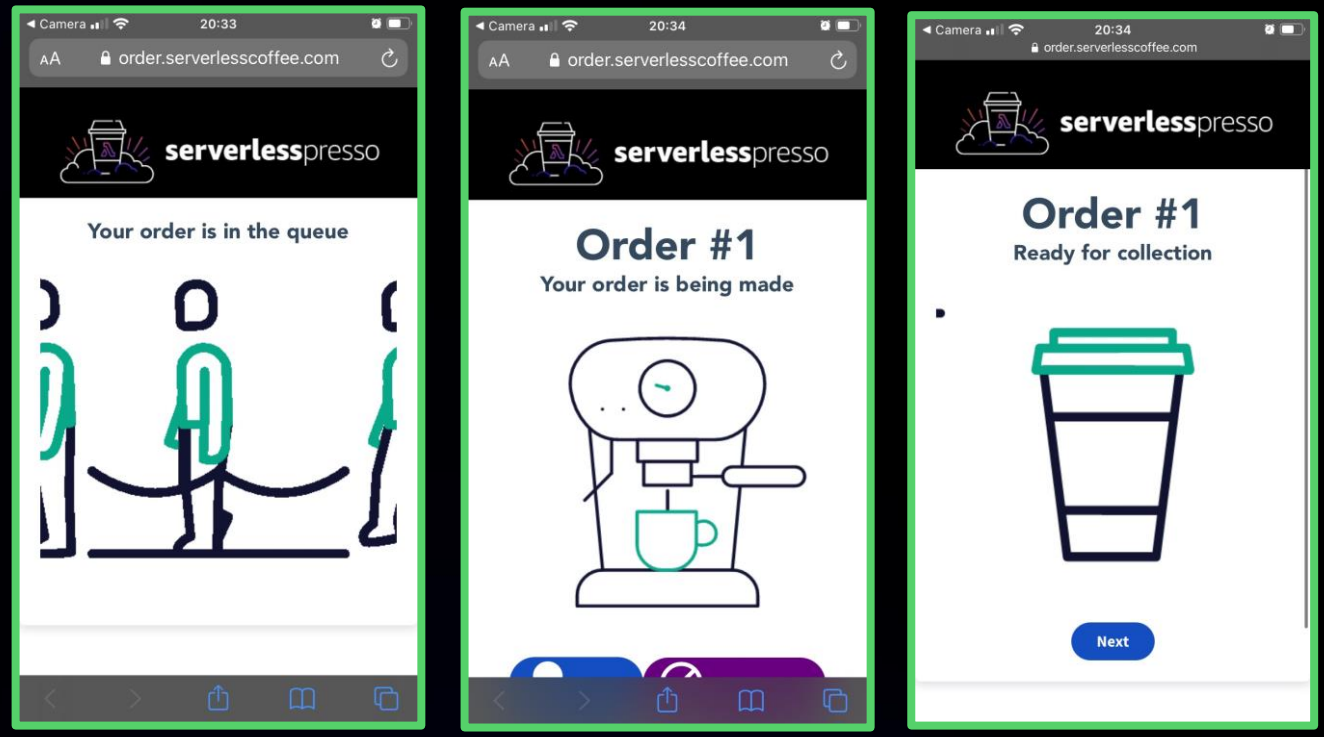
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Using AWS IoT Core for real-time messaging

Web applications often require partial information:



Completion percentages

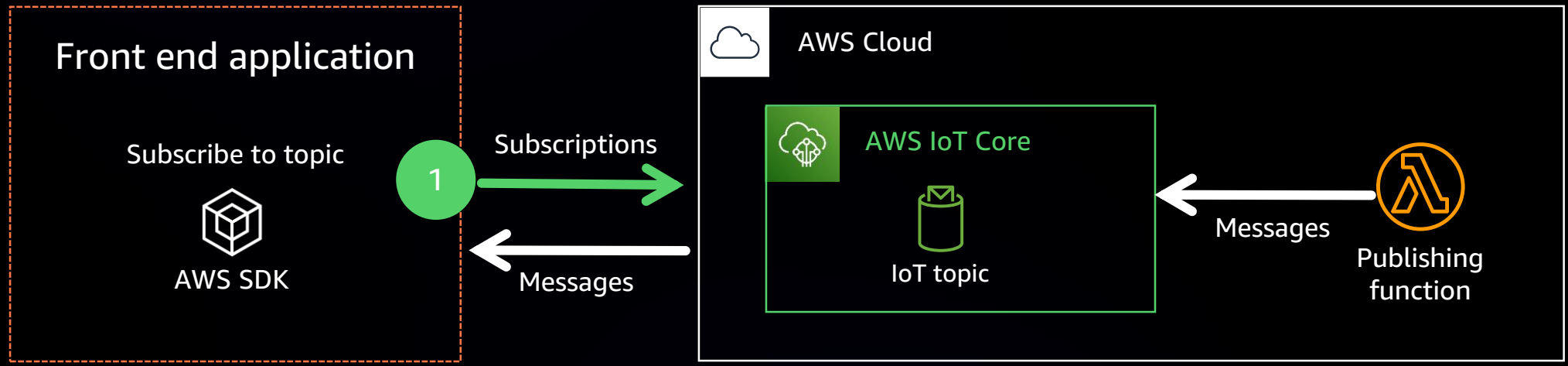


Continuous data changes

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Using AWS IoT Core for real-time messaging

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

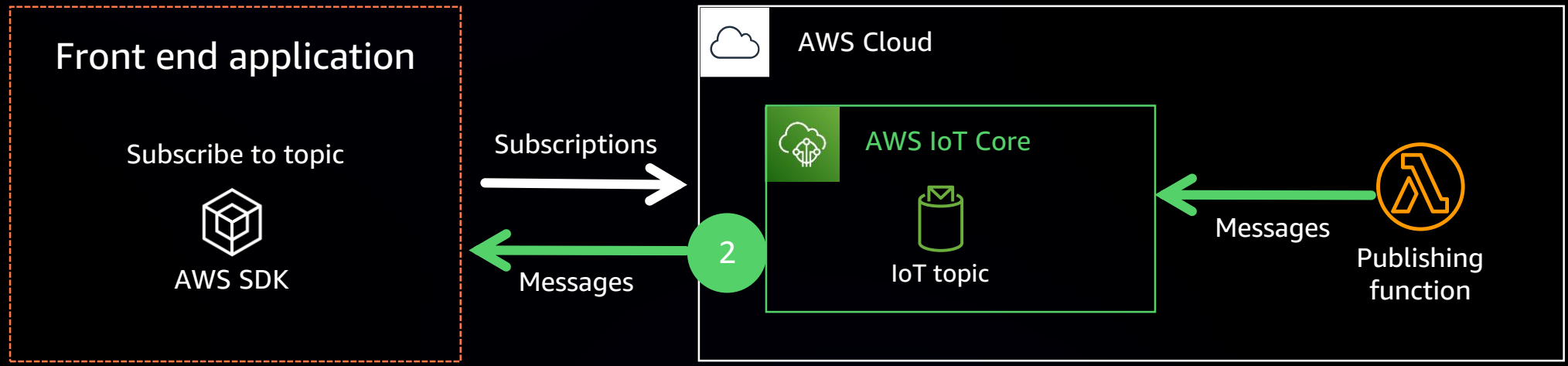


Front end uses AWS SDK to subscribe to a topic based on user’s unique user ID.

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Using AWS IoT Core for real-time messaging

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

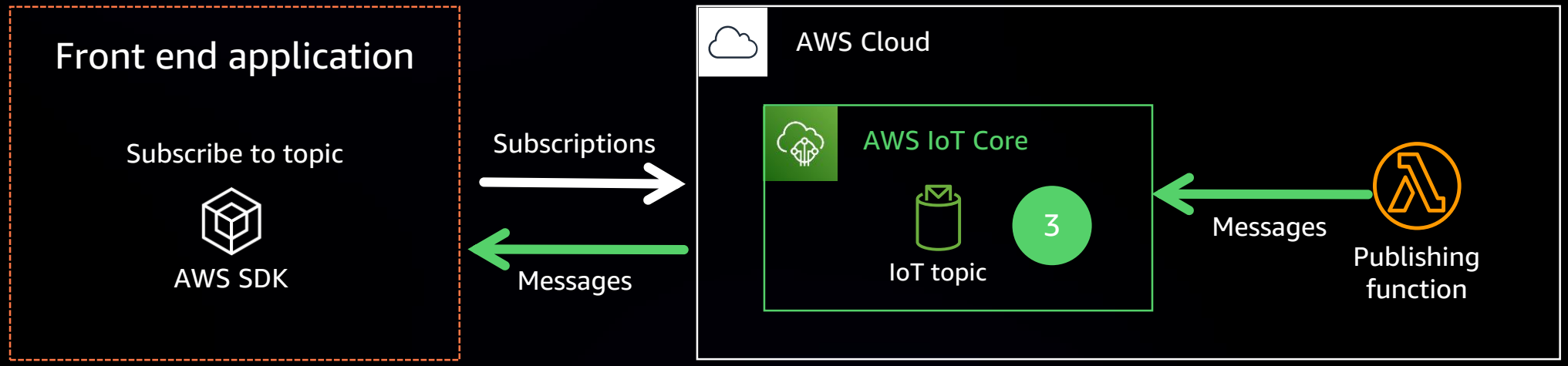


Receives messages published by the backend to this topic.

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Using AWS IoT Core for real-time messaging

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

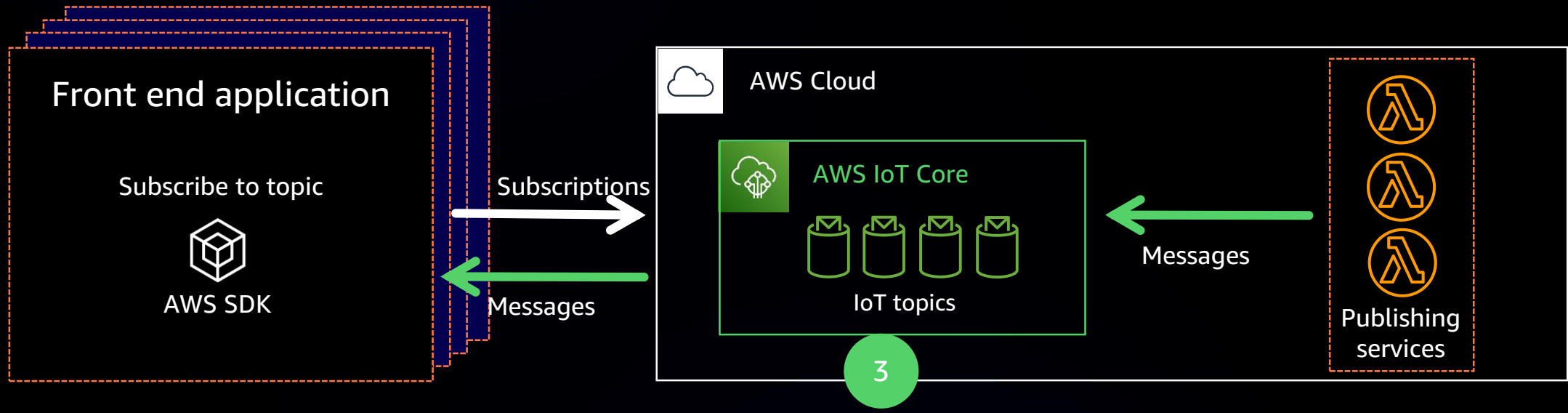


Messages are categorized using topics. Topic names are UTF 8 encoded strings.

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Using AWS IoT Core for real-time messaging

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



The IoT core service manages the WebSocket connection between backend publishers and front-end subscribers.

This enables fanout functionality to thousands front-end devices.

© 2021, 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)
})

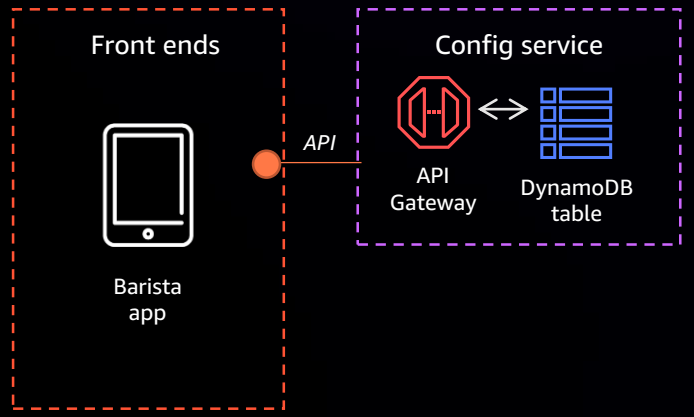
mqttClient.on('message', function (topic, payload) {
     const msg = JSON.parse(payload.toString())
     console.log('IoT msg: ', topic, msg)
})
```

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

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Combining multiple approaches for your front-end application

Many front-end applications can combine synchronous and asynchronous response models.



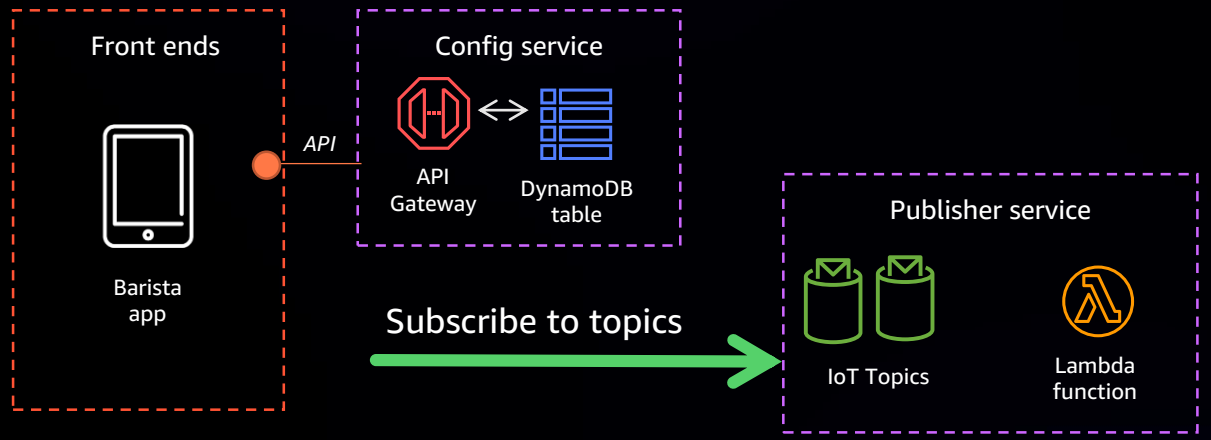
Serverlesspresso sends an initial synchronous request to retrieve the current “state of things.”

serverlesspresso

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Combining multiple approaches for your front-end application

Many front-end applications can combine synchronous and asynchronous response models.



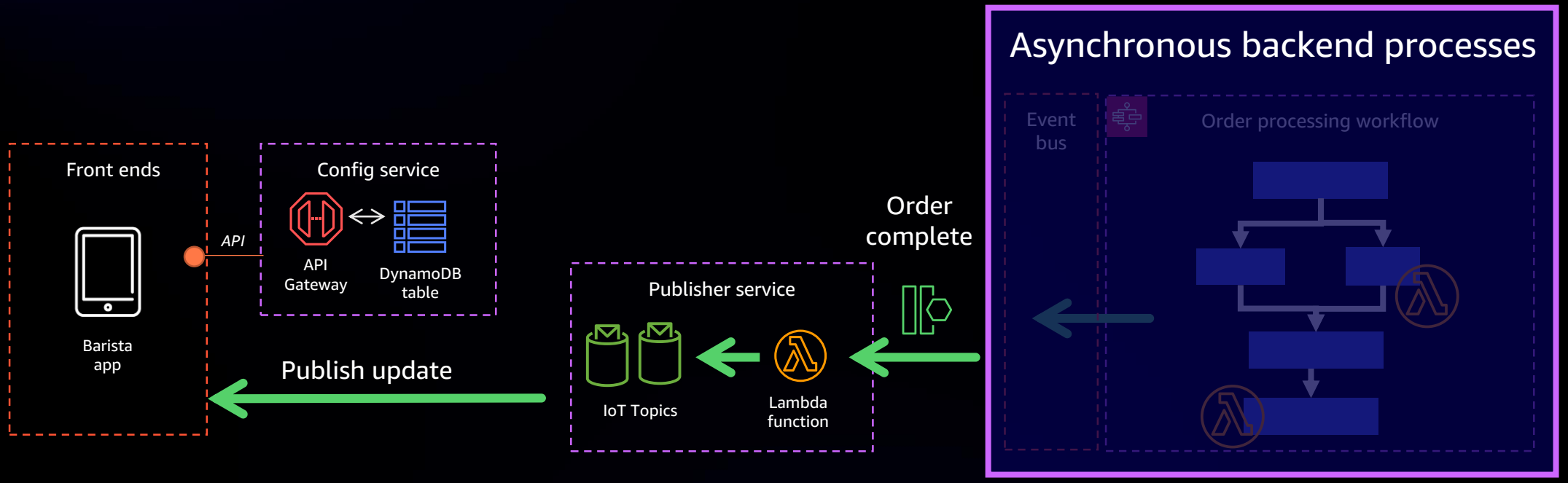
Simultaneously the front end subscribes to global and user-based IoT topics.



serverlesspresso

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Combining multiple approaches for your front-end application



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

serverlesspresso

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Recap

BEST PRACTICES FOR BUILDING INTERACTIVE APPLICATIONS WITH AWS LAMBDA

- Use asynchronous workloads (where possible)
- Reduce code – use the services
- Put storage first (when possible)

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Thank you!

Benjamin Smith

@Benjamin_l_s

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.