

Serverless Workshop

Let's have a cup of coffee ...

Lars Röwekamp | @mobileLarson | #WISSENTEILEN



Das bin ich >



Lars Röwekamp



@mobileLarson

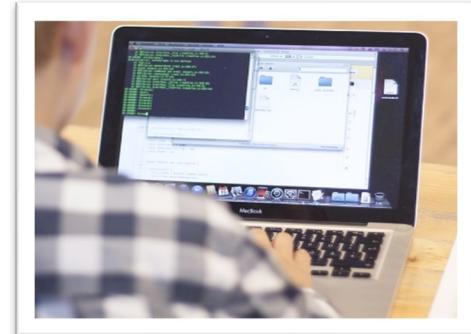
CIO New Technologies
OPEN KNOWLEDGE

(Architecture, Microservices, Cloud, AI & ML)

Und da arbeite ich >



OPEN KNOWLEDGE
@_openknowledge



A photograph of a coffee shop interior. In the foreground, two women are seated at a wooden counter, facing away from the camera. Behind them, a man in a white t-shirt is working at the counter. The background features a chalkboard wall covered in various coffee-related drawings and text, including "coffee MENU", "TASTE YOUR COFFEE", "Choux", "French Press", "Aero Press", "Siphon", "Americano", "Cafe Latte", "Milk foam", "Espresso", "Mochiato", "HIERARCHY", and "LA VIỆT COFFEE". The lighting is warm, with several green pendant lights hanging above the counter.

let's start

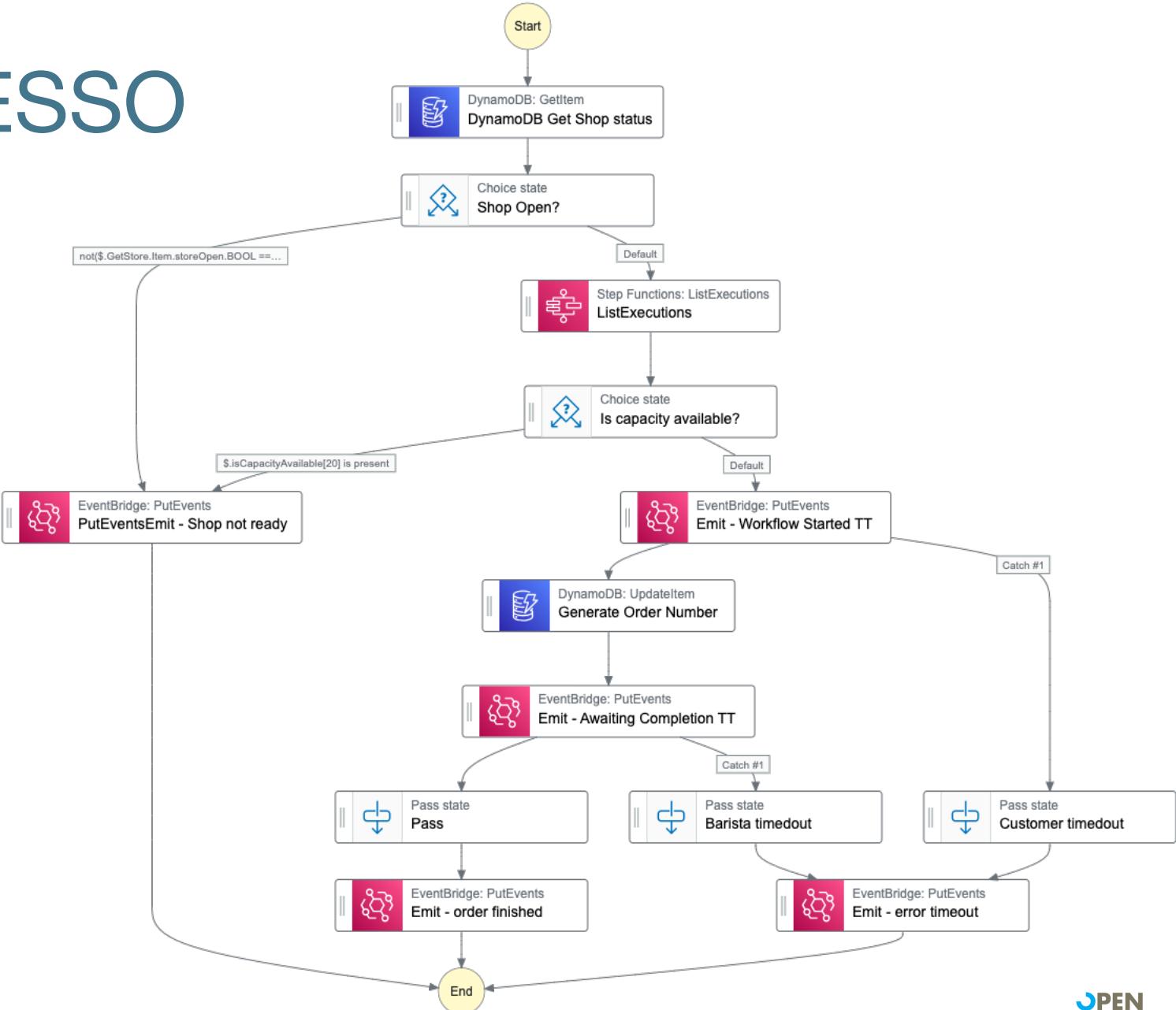


Serverless Workshop Show-Case



SERVERLESSPRESSO

Order Workflow



SERVERLESSPRESSO

Order Workflow

↓ Kunde startet Order via Code Scan.

SERVERLESSPRESSO

Order Workflow

- | **Kunde** startet Order via Code Scan.
- ↓ Coffee Shop geöffnet? Freie Kapazitäten?

SERVERLESSPRESSO

Order Workflow

- | **Kunde** startet Order via Code Scan.
- | Coffee Shop geöffnet? Freie Kapazitäten?
- ↓ **Kunde** hat 5 min seine Order zu platzieren.

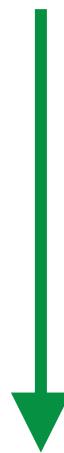
SERVERLESSPRESSO

Order Workflow

- | **Kunde** startet Order via Code Scan.
Coffee Shop geöffnet? Freie Kapazitäten?
- | **Kunde** hat 5 min seine Order zu platzieren.
- ↓ **Barista** hat 15 min die Order zu bearbeiten.

SERVERLESSPRESSO

Order Workflow

- 
- Kunde** startet Order via Code Scan.
 - Coffee Shop geöffnet? Freie Kapazitäten?
 - Kunde** hat 5 min seine Order zu platzieren.
 - Barista** hat 15 min die Order zu bearbeiten.

Bestellung **zubereitet**

SERVERLESSPRESSO

Order Workflow

Kunde startet Order via Code Scan.

Coffee Shop geöffnet? Freie Kapazitäten?

Kunde hat 5 min seine Order zu platzieren.

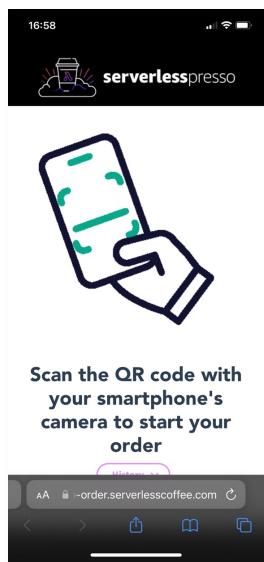
Barista hat 15 min die Order zu bearbeiten.

Bestellung **zubereitet**

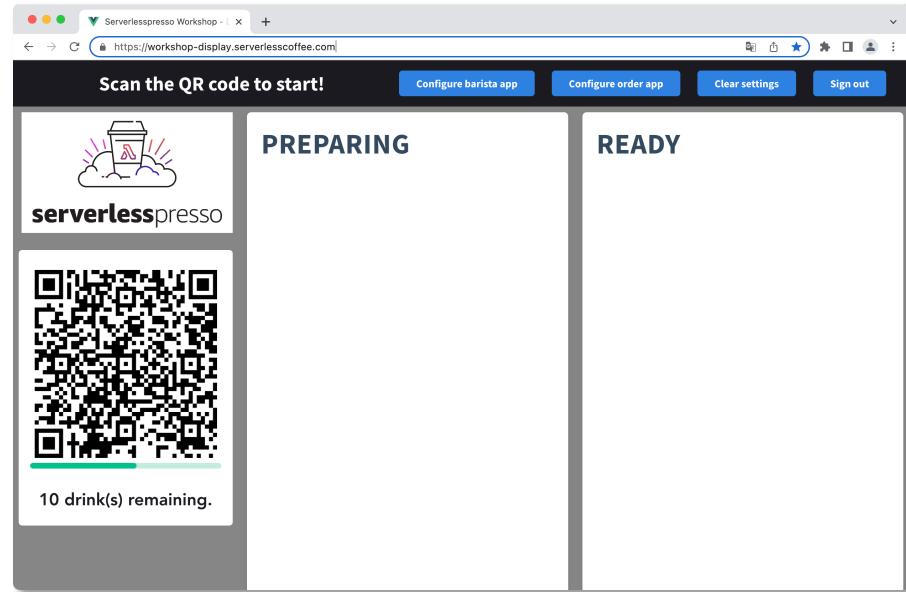
Bestellung **canceled**



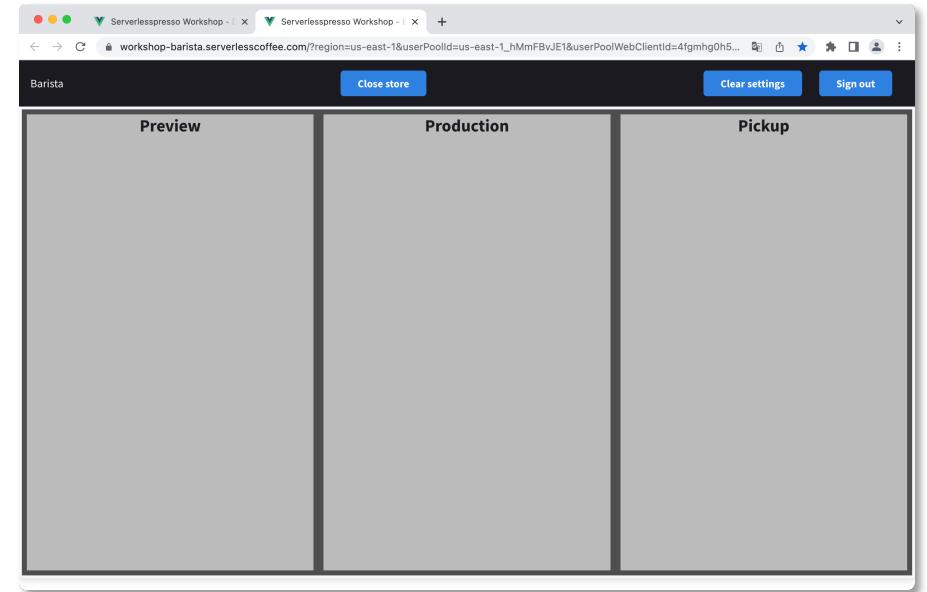
Order App



Coffeeshop Display App

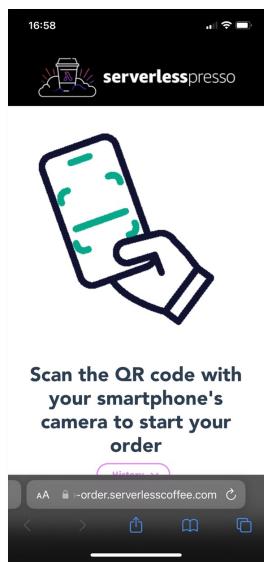


Barista App



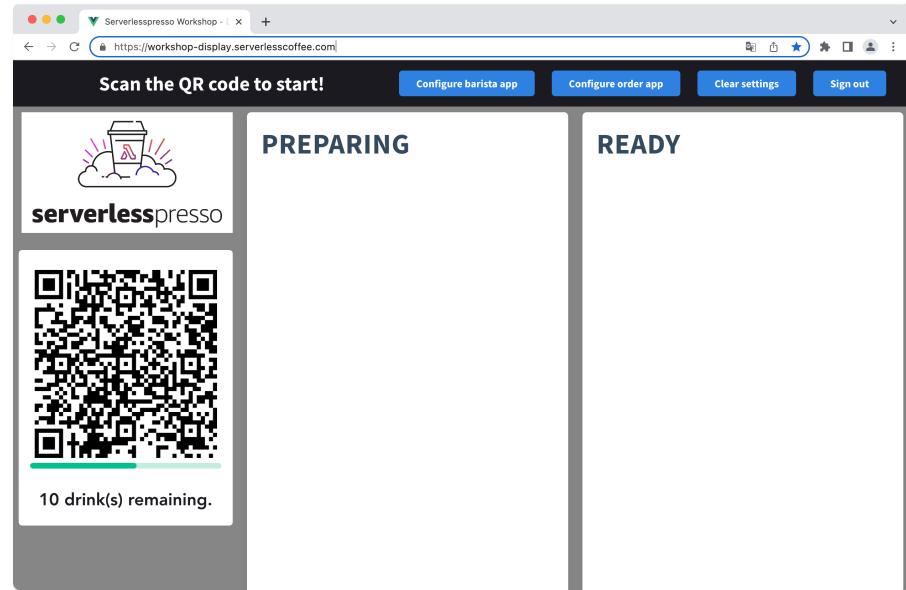


Order App



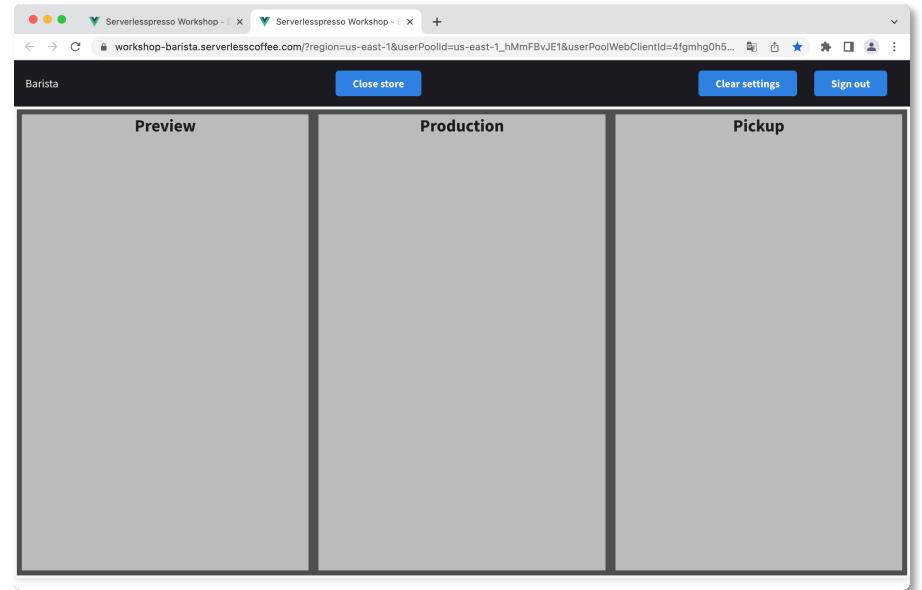
App is ready
to scan code.

Coffeeshop Display App



No orders in preparation.
Remaining capacity is 10.

Barista App



No orders in queue
or in preparation.

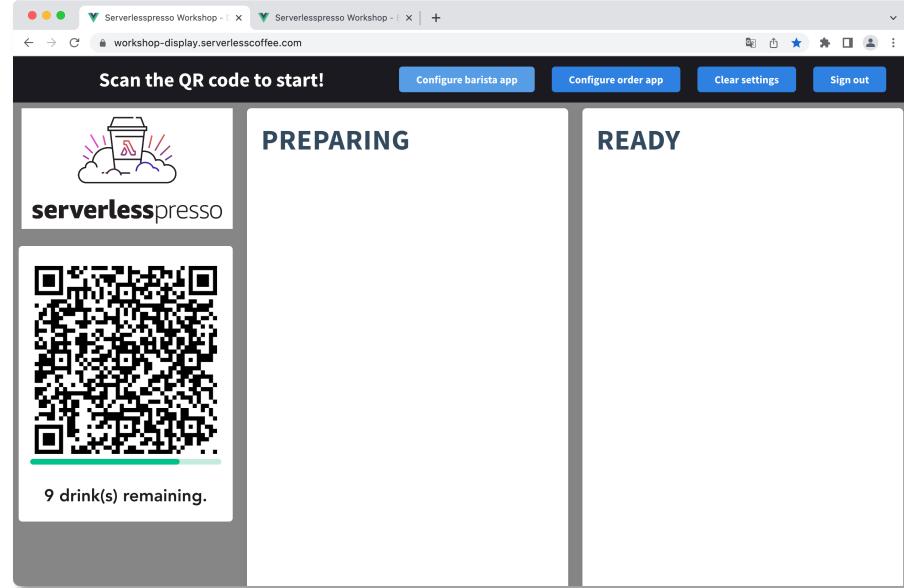


Order App



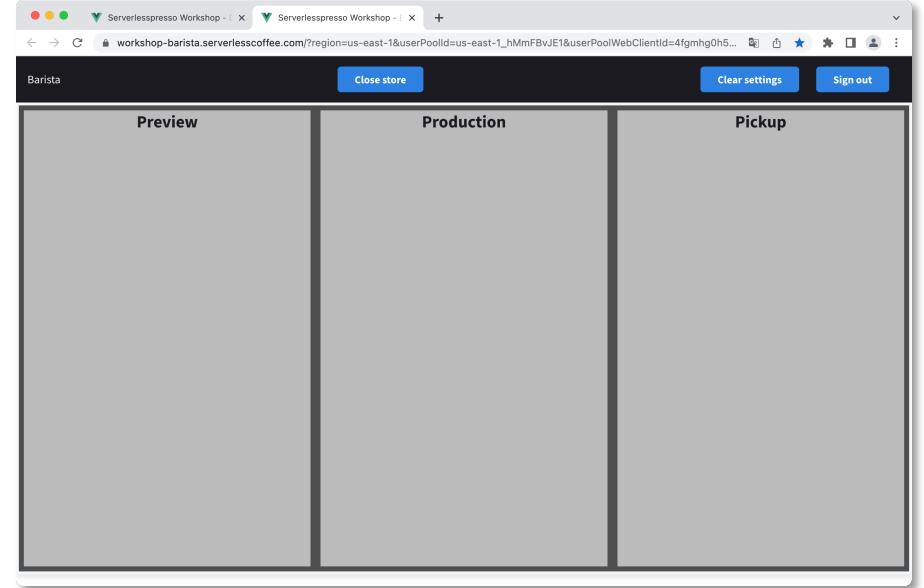
Code
scanned.

Coffeeshop Display App



5 minute counter started.
Remaining capacity is 9.

Barista App



No orders in queue
or in preparation.

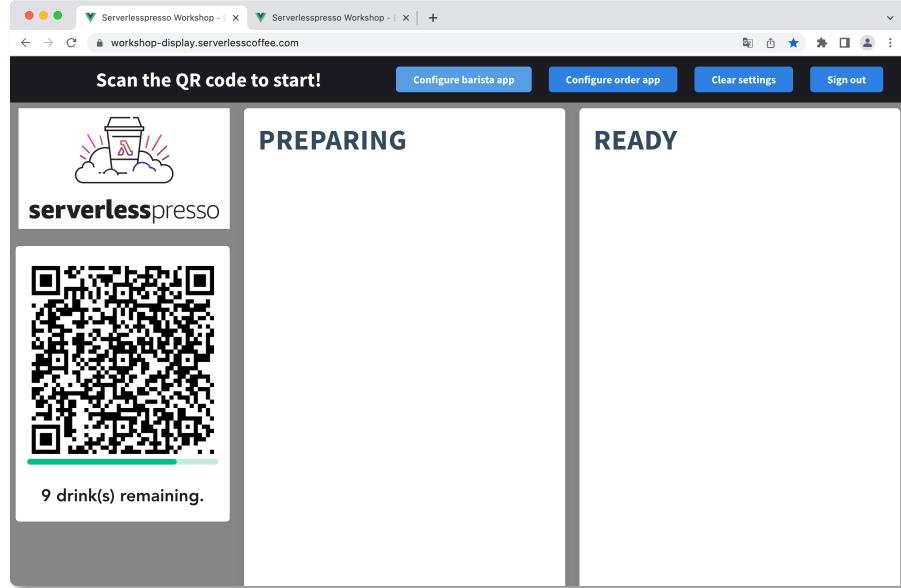


Order App



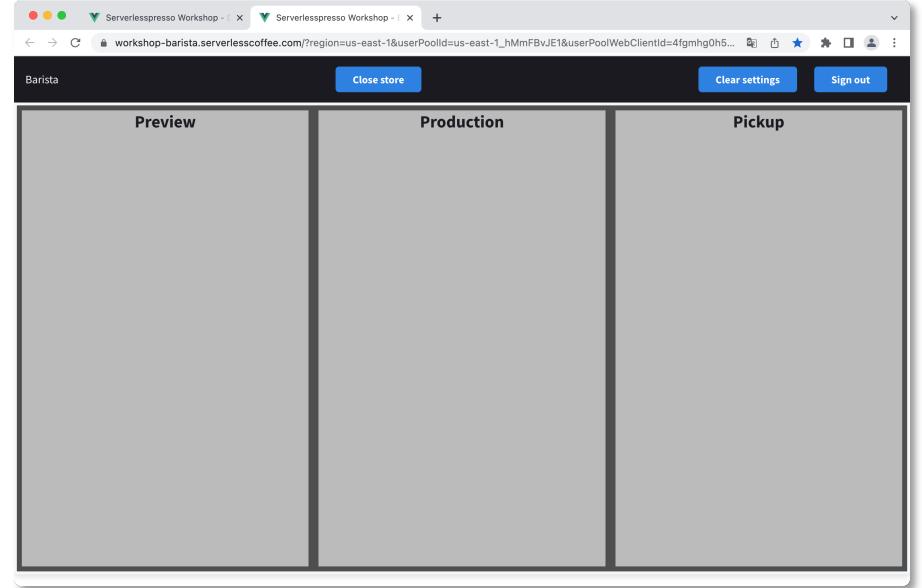
Drink to order selected.

Coffeeshop Display App



Waiting for outstanding order and for other customers.

Barista App



No orders in queue or in preparation.

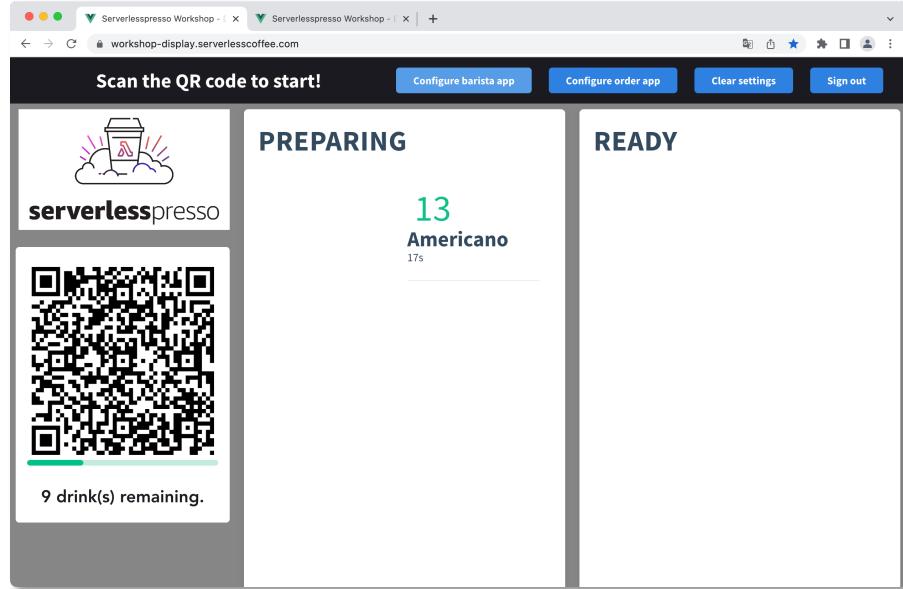


Order App



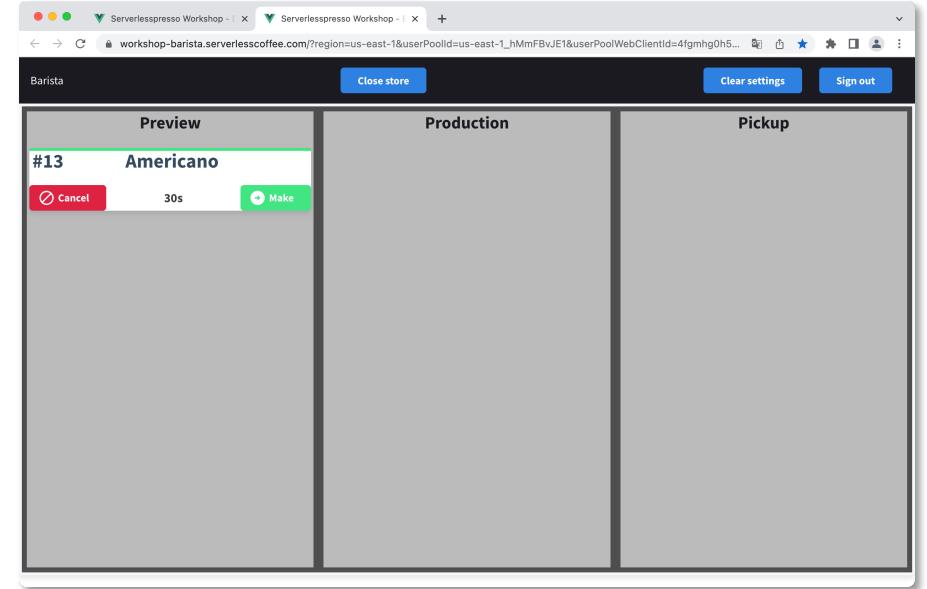
Selected drink ordered.

Coffeeshop Display App



Order is in preparation (or at least queued).

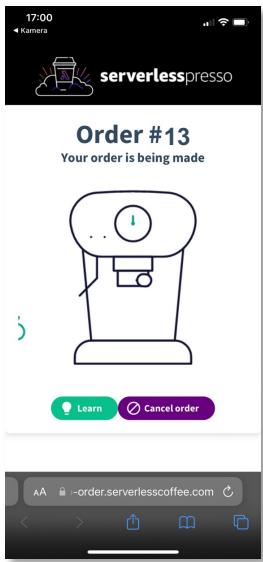
Barista App



Some work to do. Lets hurry, only 15 minutes left.

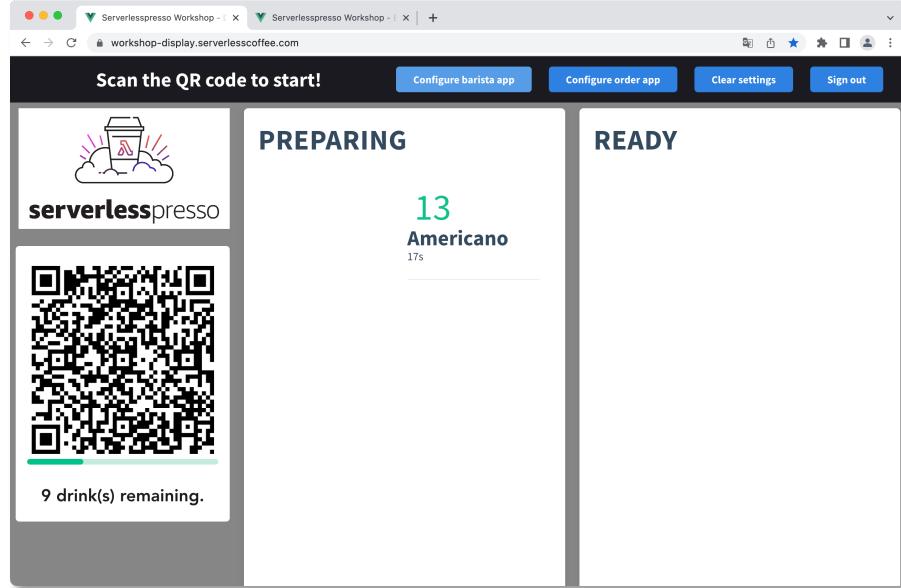


Order App



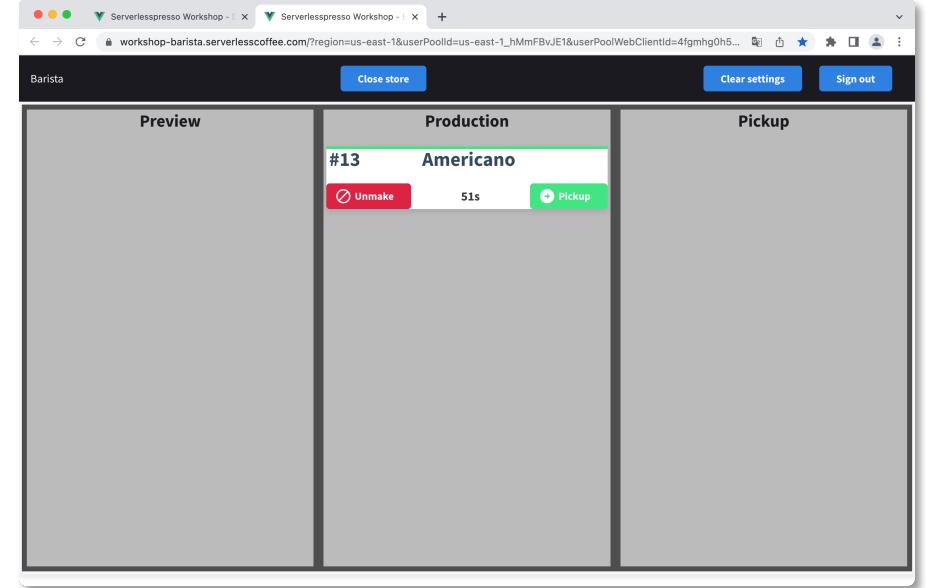
Waiting
for order

Coffeeshop Display App



Order is in preparation.
Waiting for other orders.

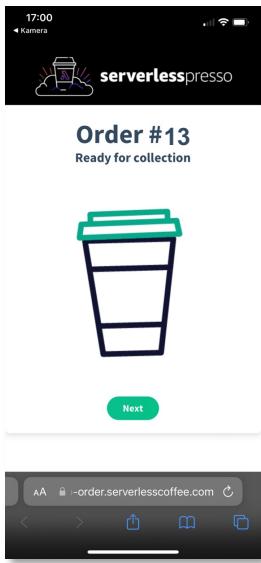
Barista App



Order is in production.
Waiting for other orders.

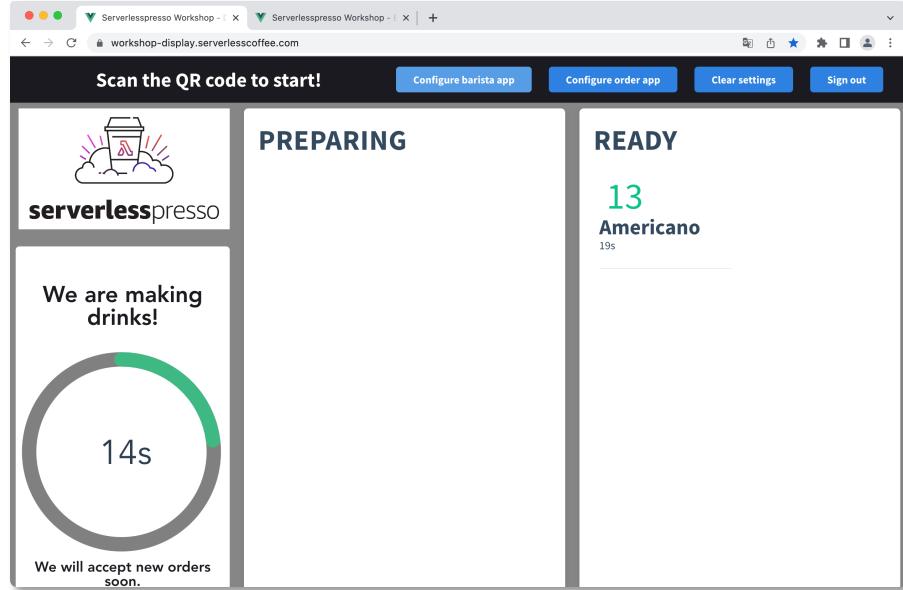


Order App



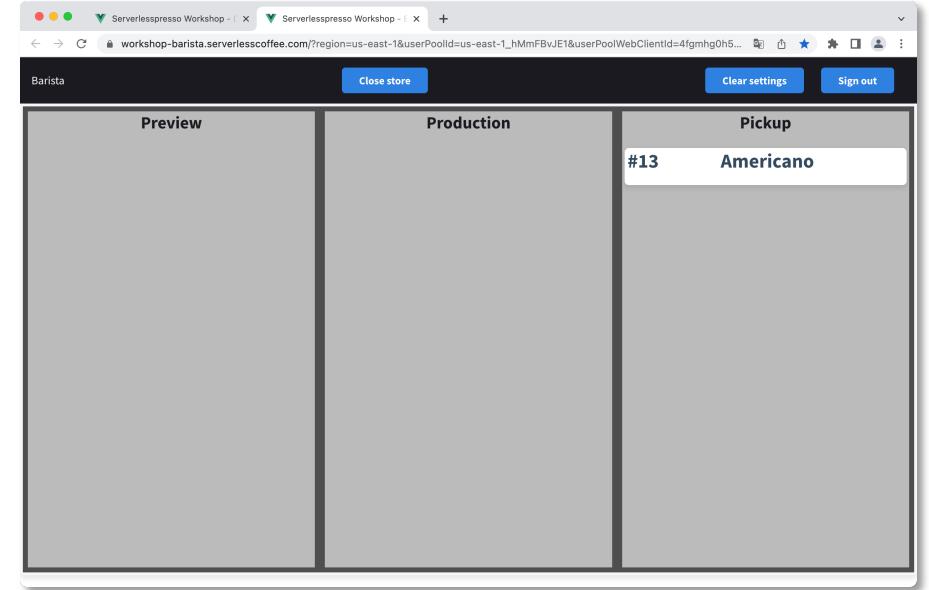
Order is ready
for pick up

Coffeeshop Display App



Order is ready.
Waiting for other orders.

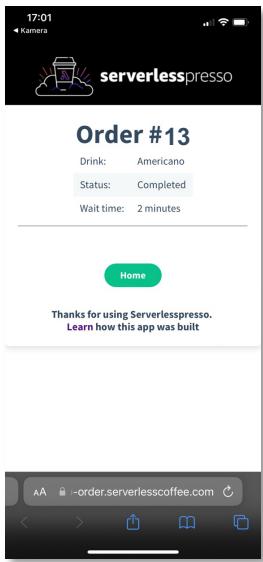
Barista App



Order production is done.
Waiting for other orders.

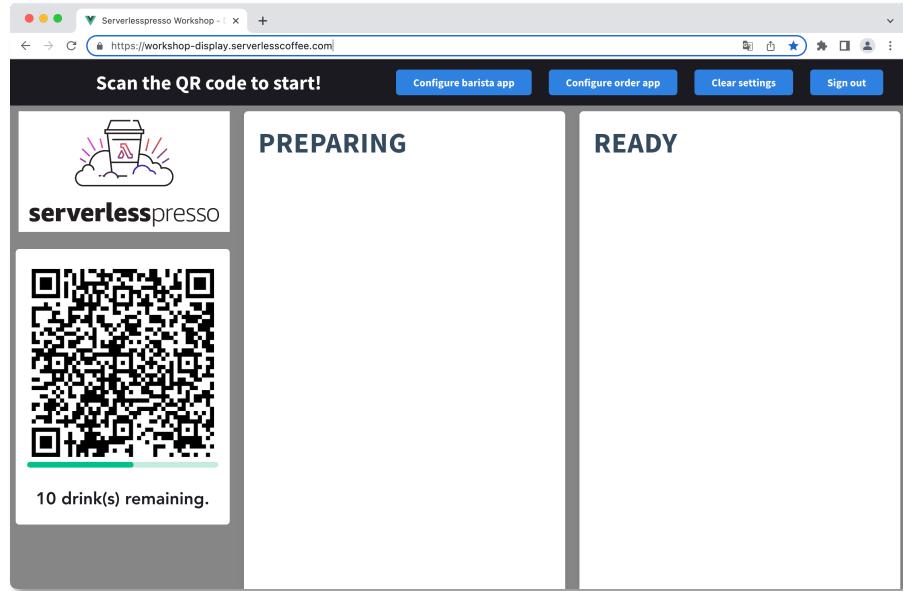


Order App



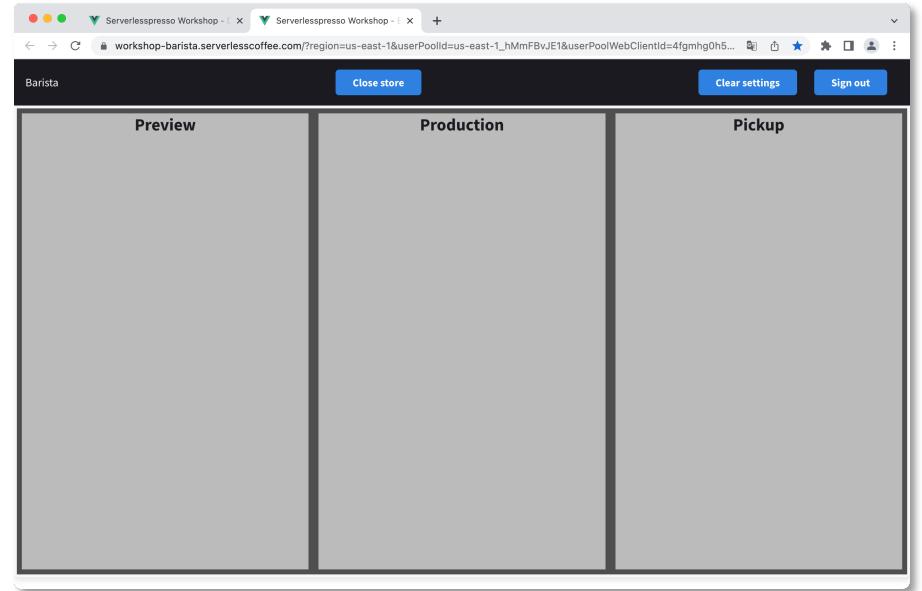
Order
picked up

Coffeeshop Display App



No orders in preparation.
Remaining capacity is 10.

Barista App



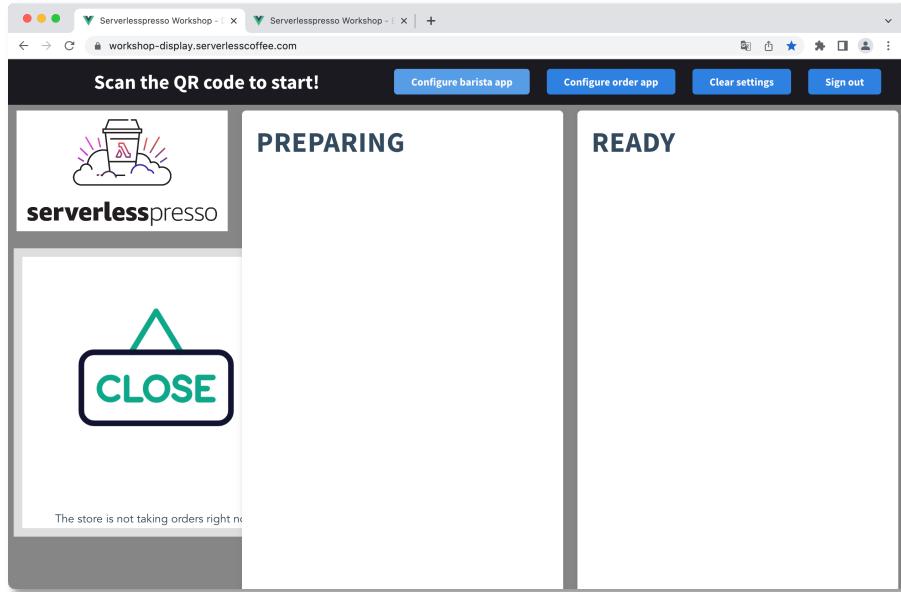
No orders in queue
or in preparation.



Order App

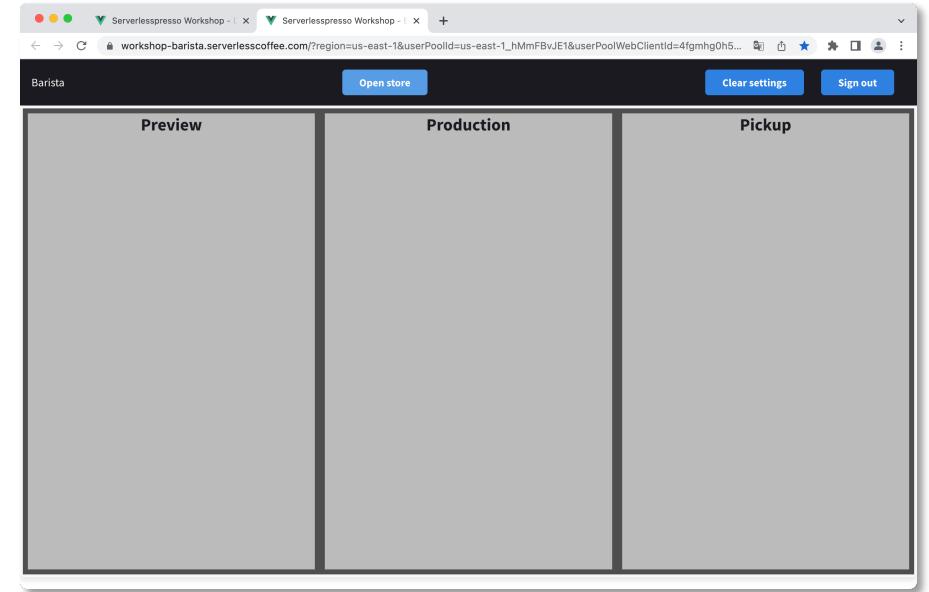


Coffeeshop Display App



Store **closed!**

Barista App



Store is **closed!**

OFFEN KUNDIG GUT

Welcome to the Serverlesspresso

workshop.serverlesscoffee.com

aws

Introduction

Setup

1a. Building the workflow - Part 1

1b. Building the workflow - Part 2

2. Routing events

3. Configuring the frontends

4. Advanced

Cleanup

After the workshop

Bug/feature request

Get the code from GitHub

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

WELCOME TO THE SERVERLESSPRESSO WORKSHOP!

>



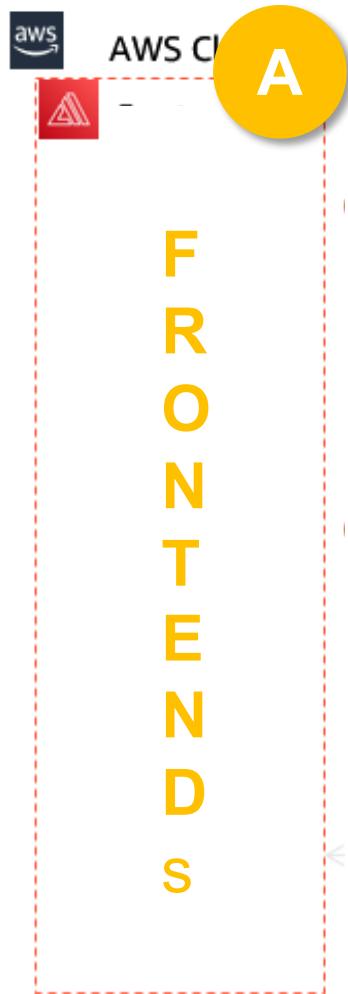
serverlesspresso

In this workshop, you will deploy a [serverless](#) backend that supports a pop-up coffee shop. You will then test your application using 3 front-end applications that are provided.

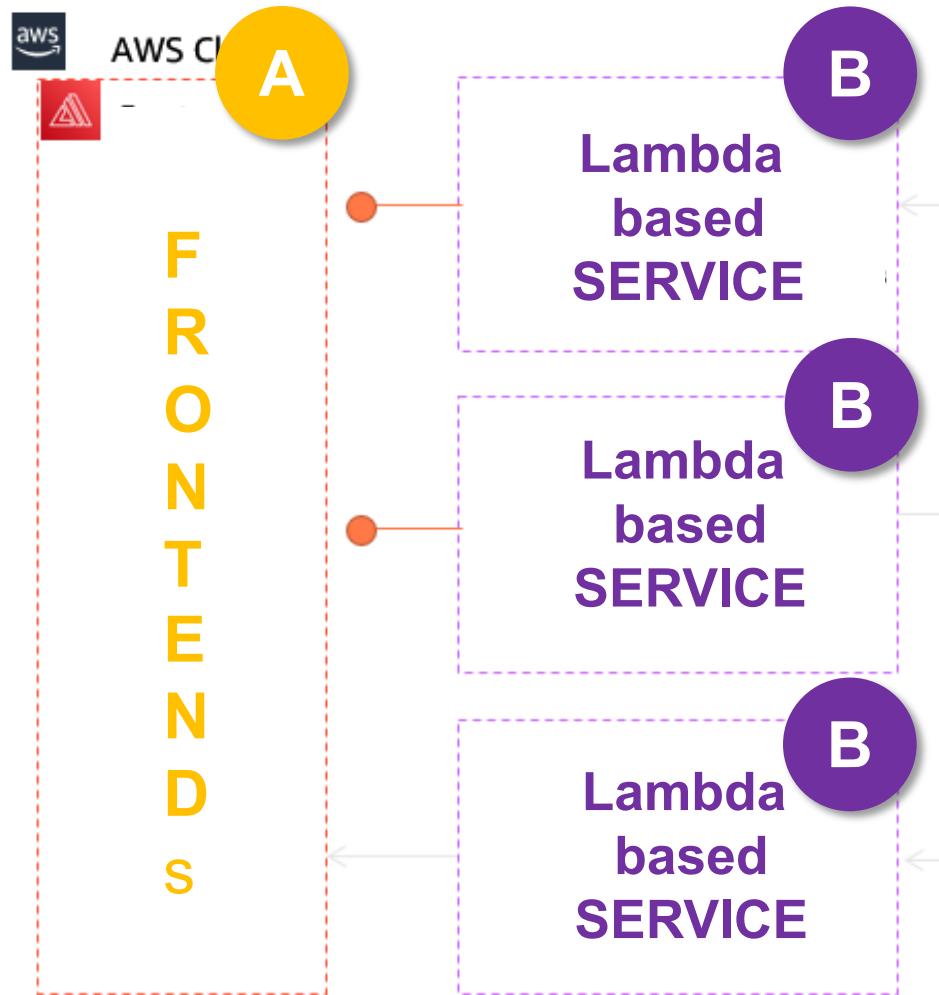
At this event, your instructor will provide an overview of the steps involved. There are AWS employees available to help with any of your questions - don't be afraid to ask if you get

AWS Serverlesspresso Tutorial: <https://workshop.serverlesscoffee.com/>

Serverless Coffee Architecture

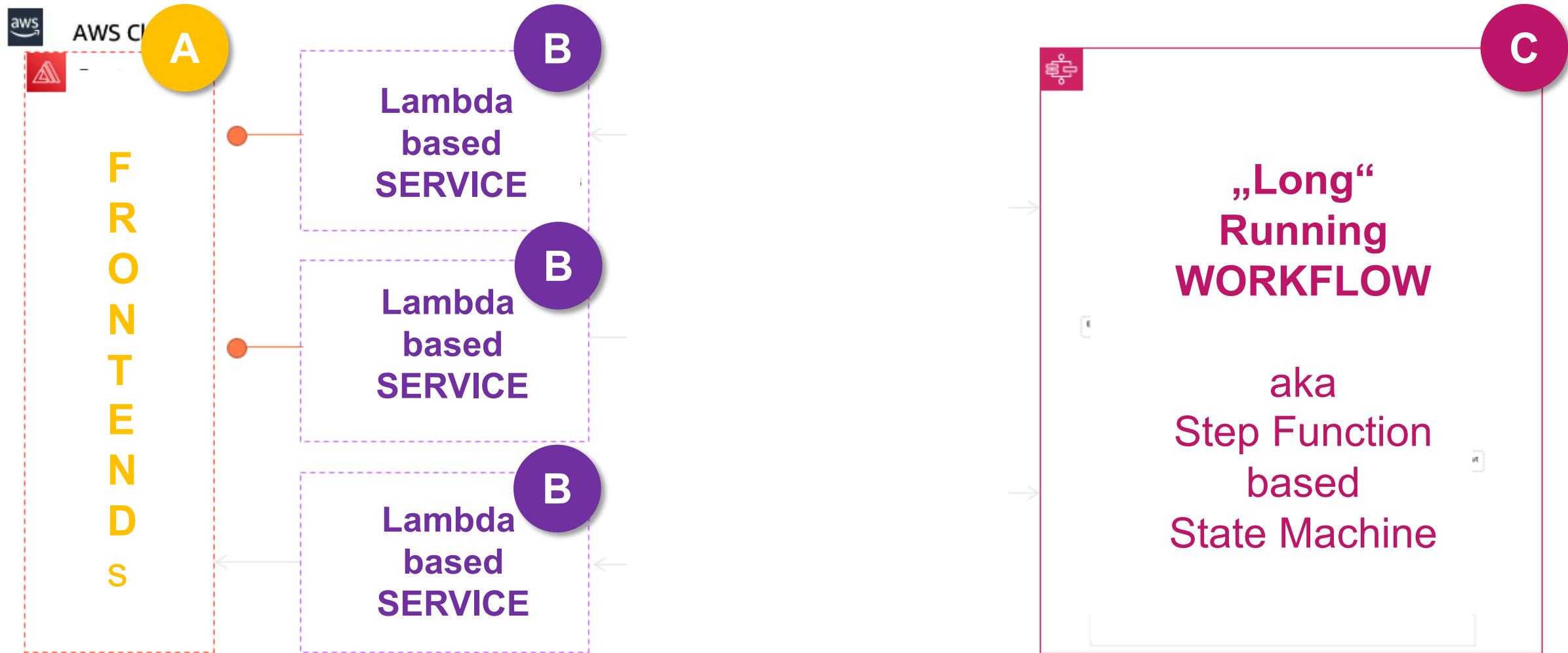


Serverless Coffee Architecture



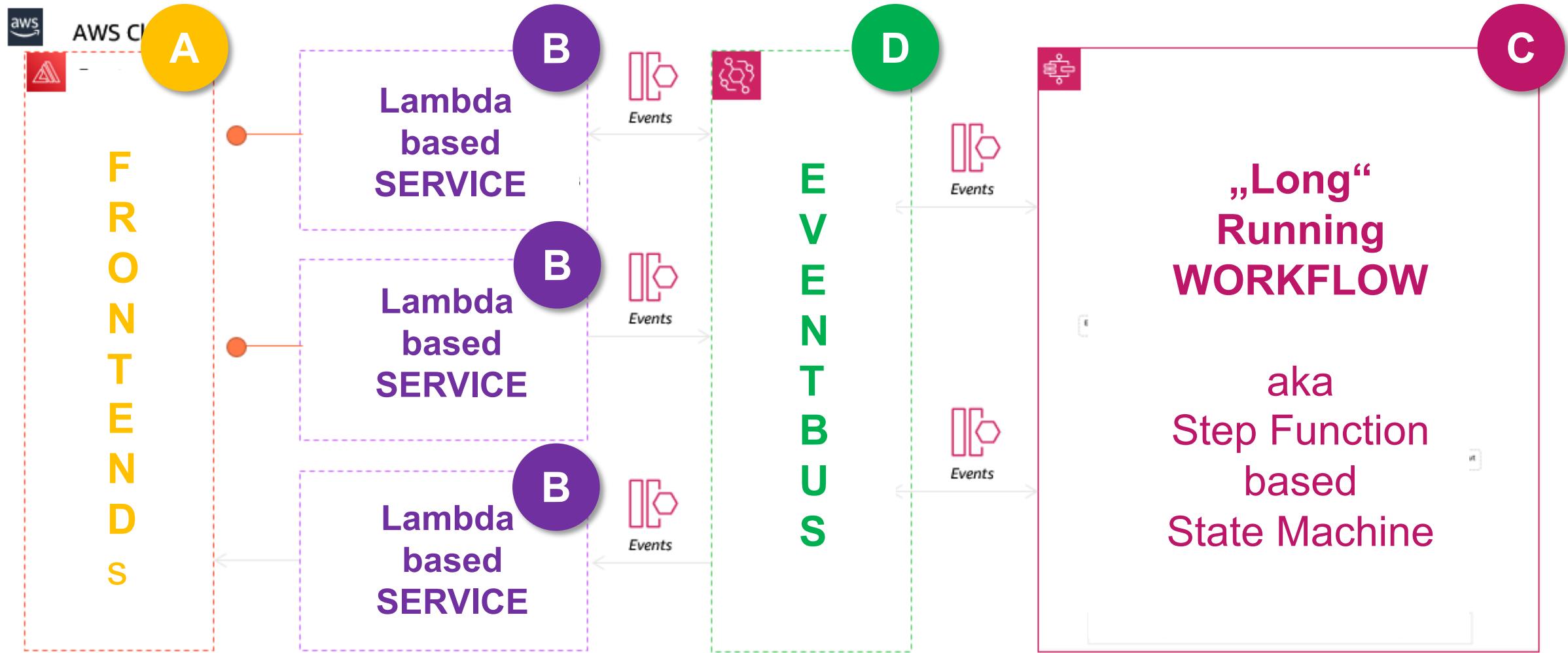
AWS Serverlesspresso Architecture: <https://workshop.serverlesscoffee.com/images/se-0-architecture.png>

Serverless Coffee Architecture



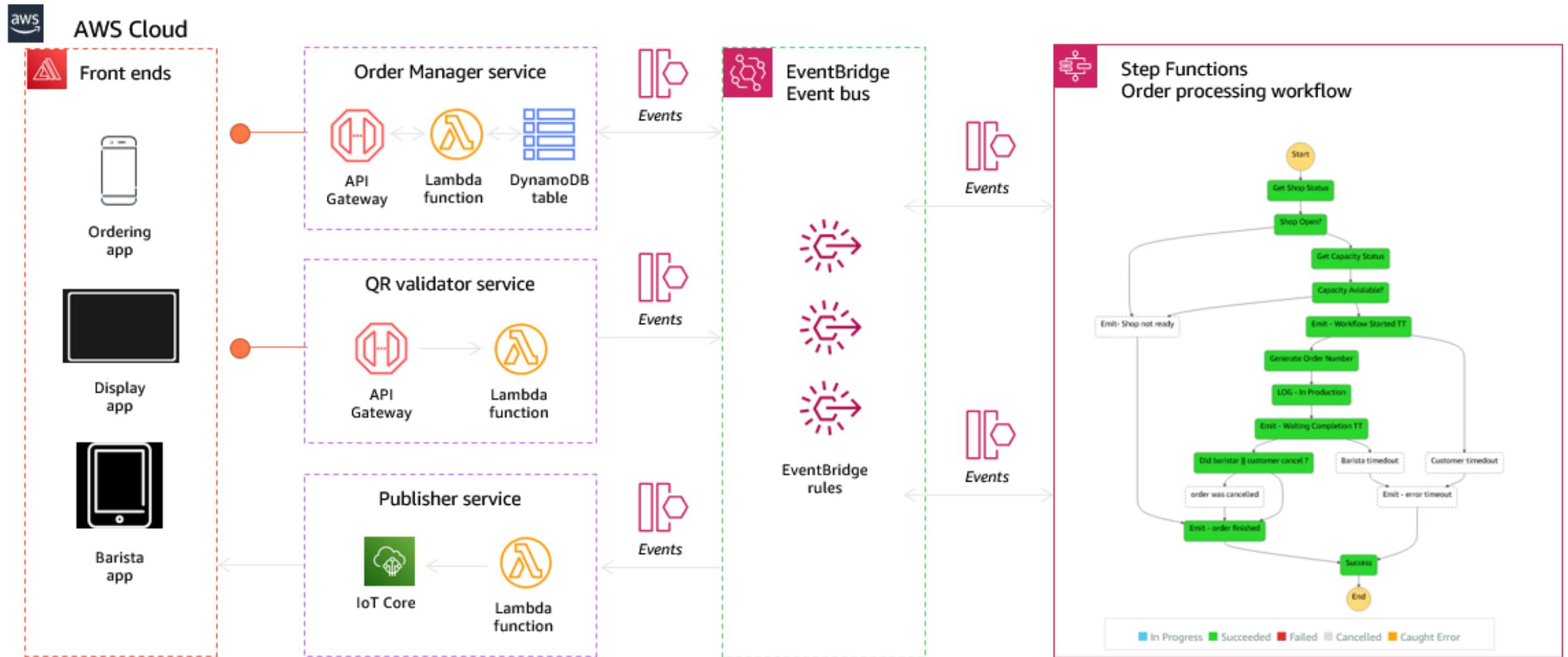
AWS Serverlesspresso Architecture: <https://workshop.serverlesscoffee.com/images/se-0-architecture.png>

Serverless Coffee Architecture



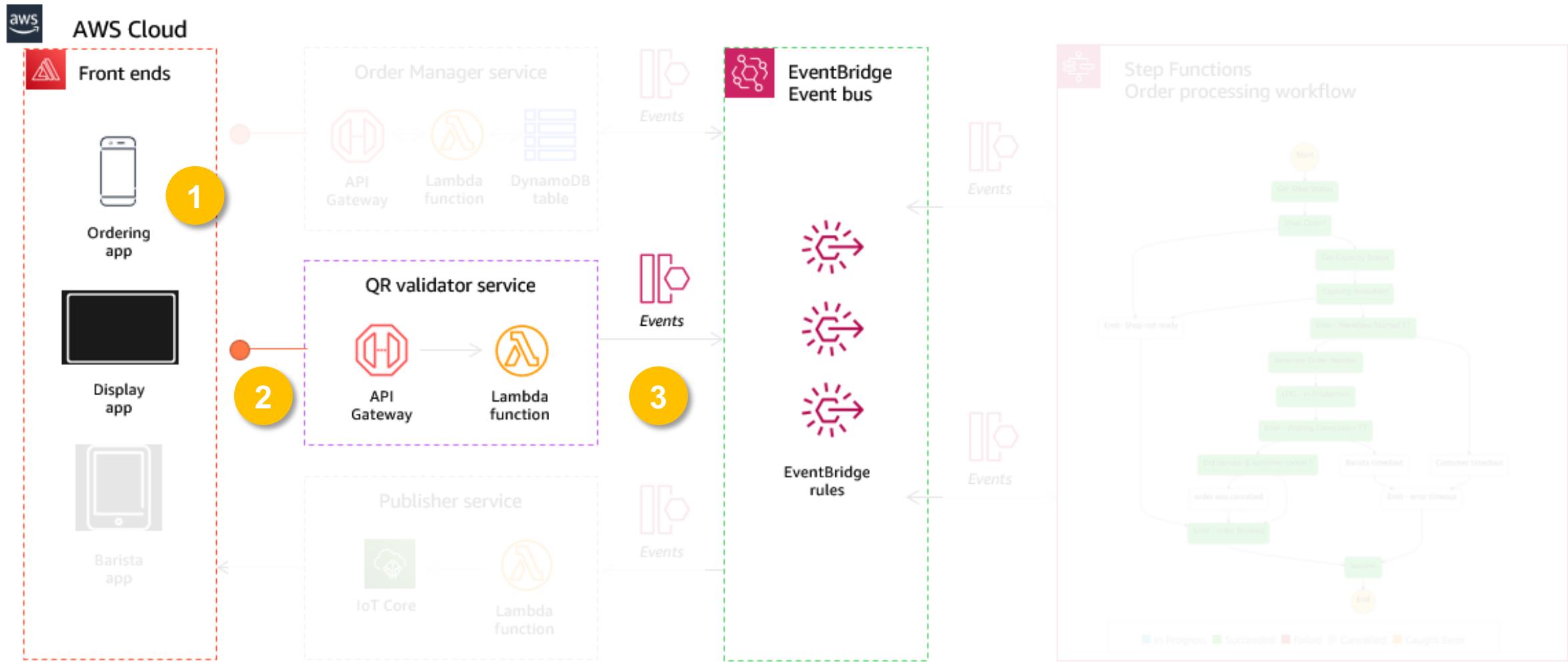
AWS Serverlesspresso Architecture: <https://workshop.serverlesscoffee.com/images/se-0-architecture.png>

Serverless Coffee Architecture



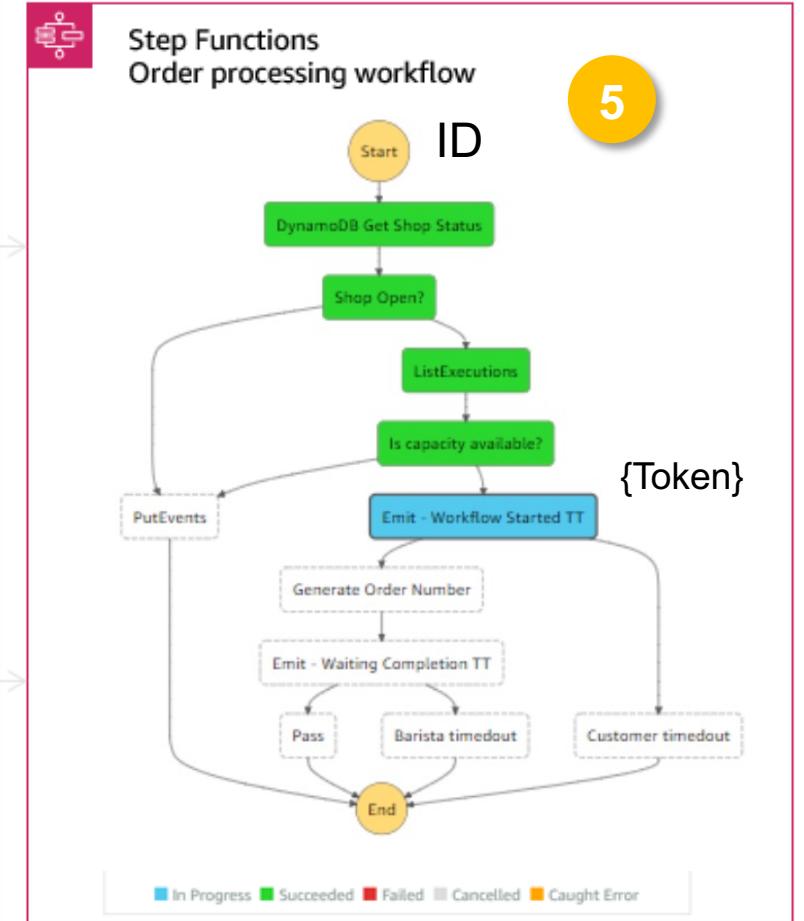
AWS Serverlesspresso Architecture: <https://workshop.serverlesscoffee.com/images/se-0-architecture.png>

Serverless Coffee „Code scanned“

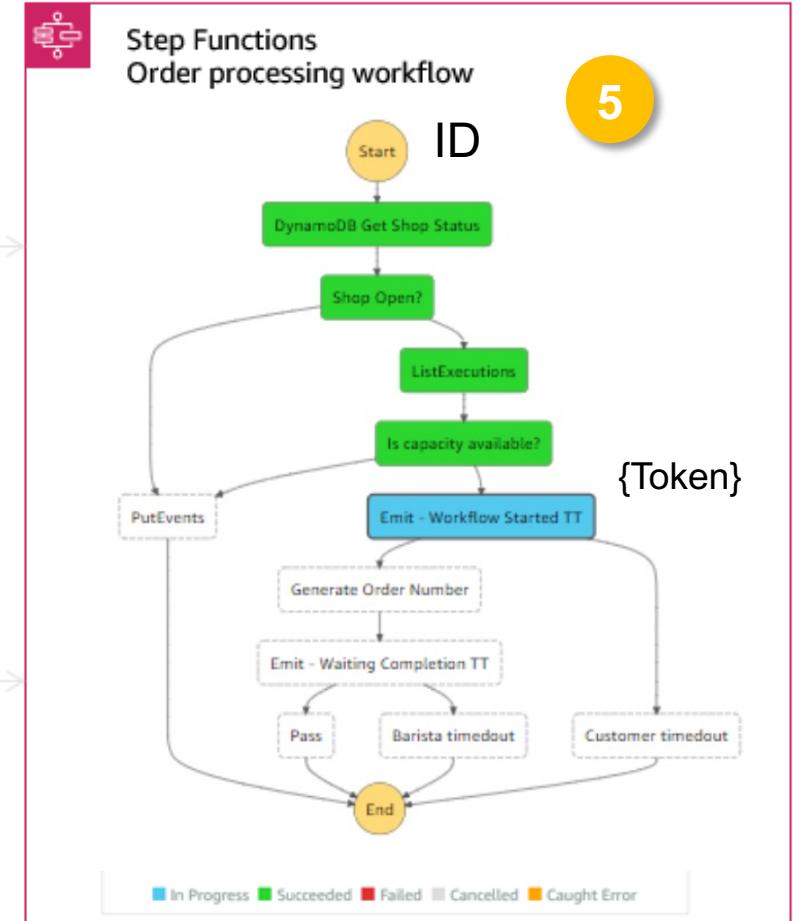
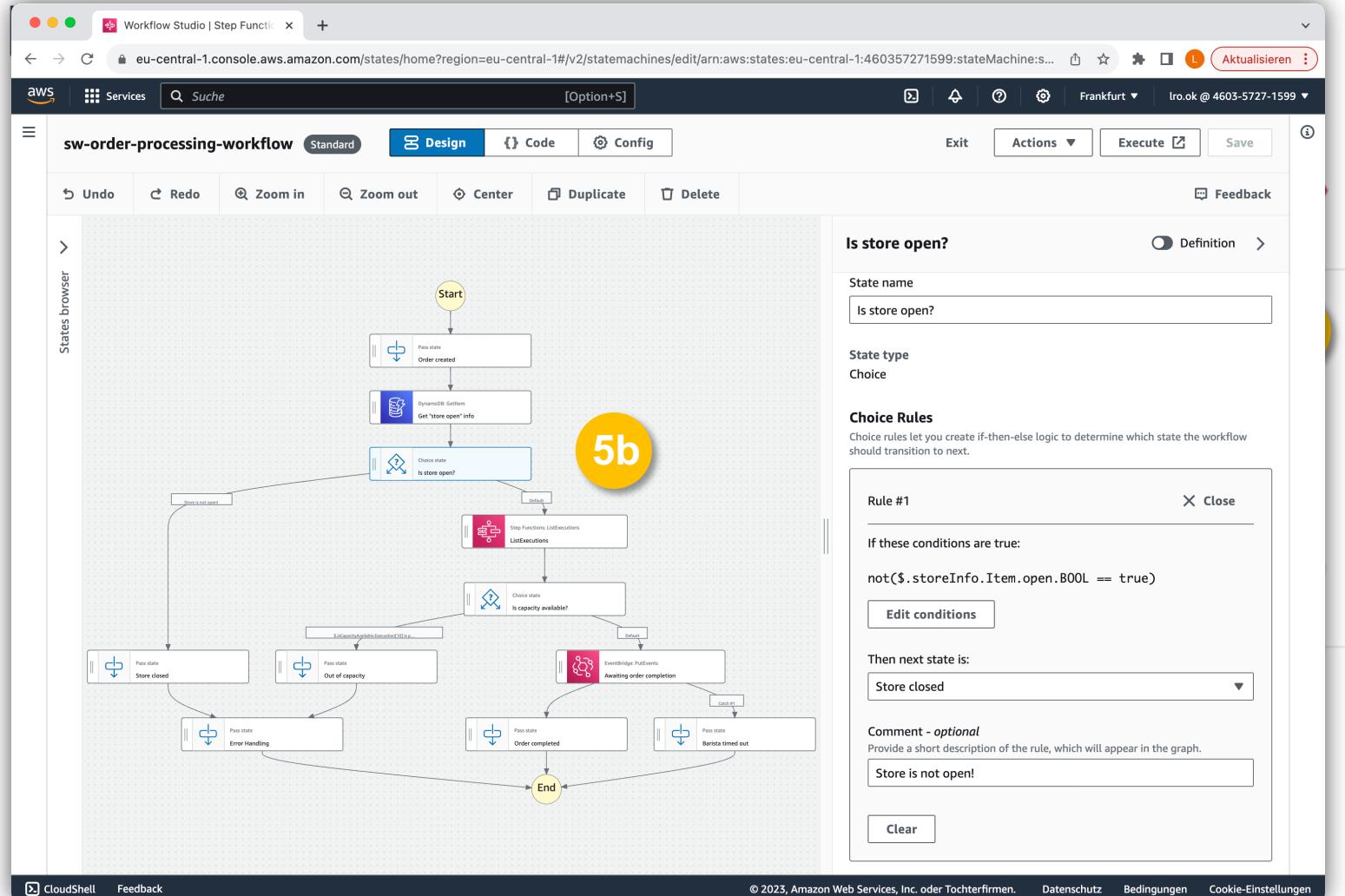


Serverless Coffee „Code scanned“

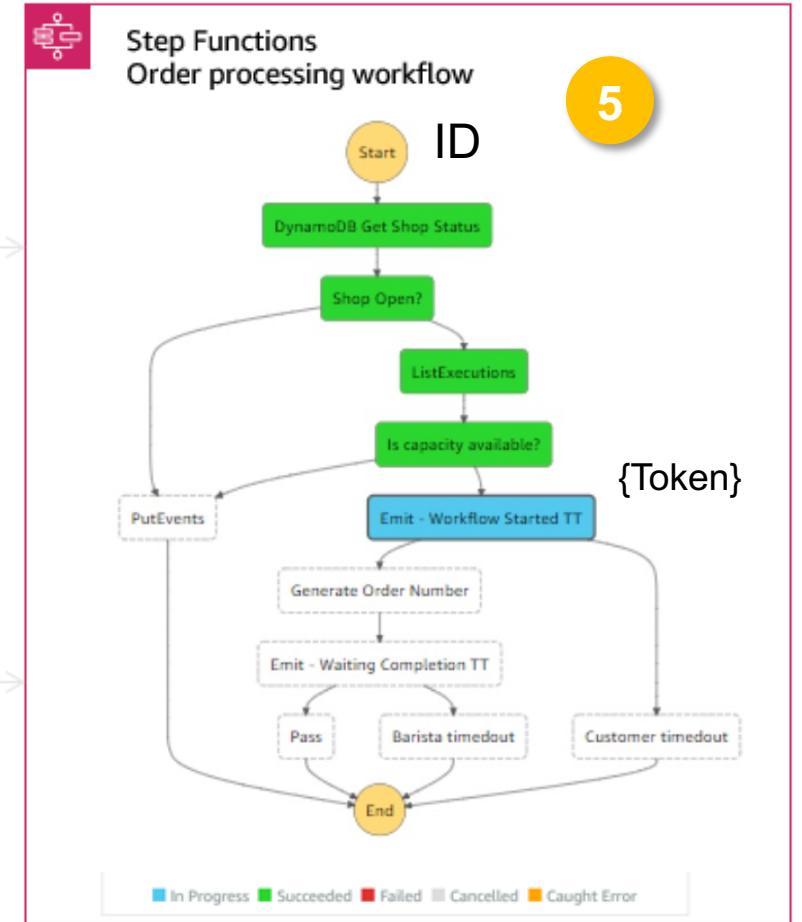
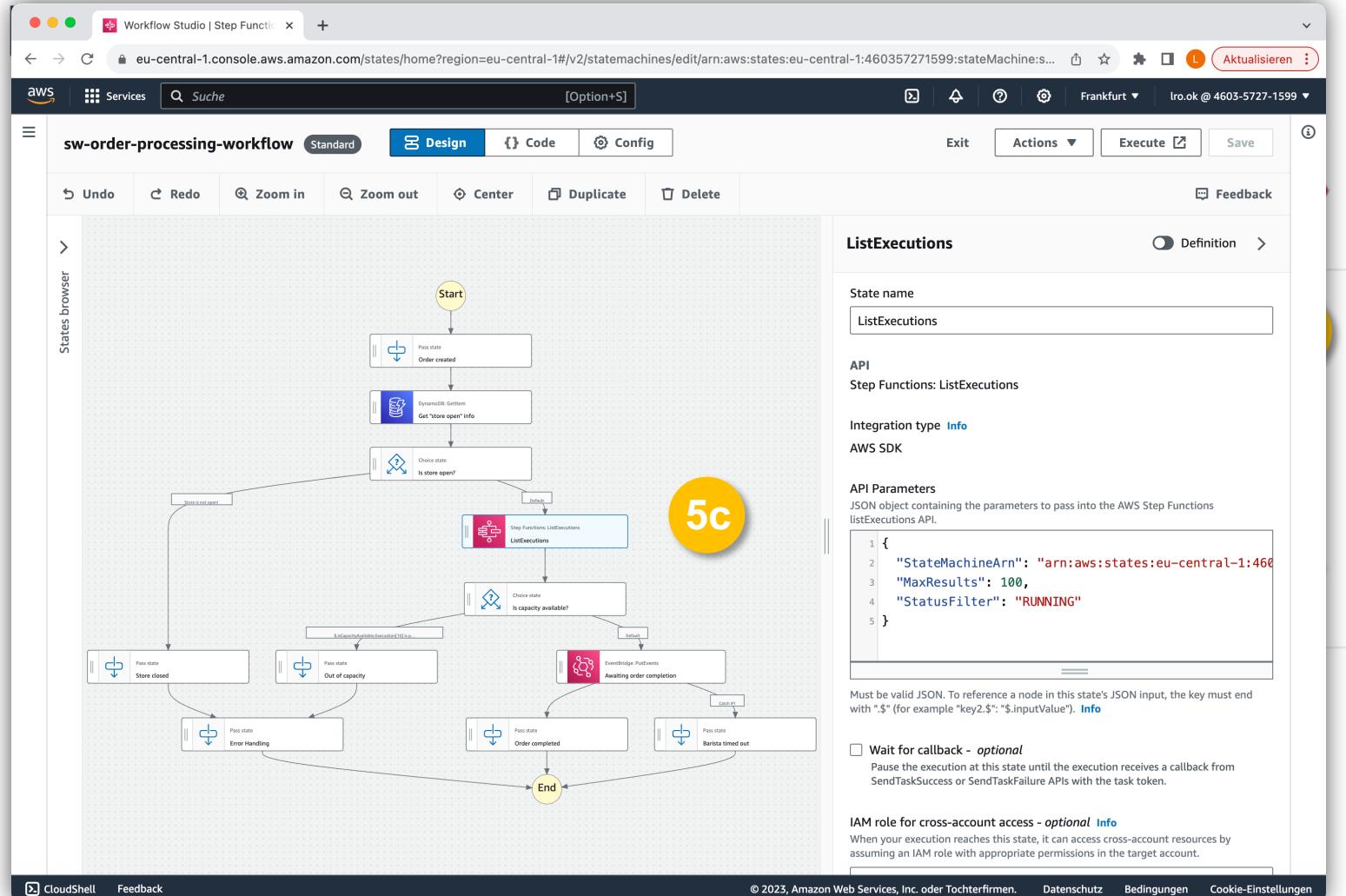
The screenshot shows the AWS Step Functions Workflow Studio interface. On the left, the 'Design' tab is selected, displaying a state machine diagram for 'sw-order-processing-workflow'. The diagram starts with a 'Start' state, followed by a 'DynamoDB GetItem' state ('Get "store open" info'). This leads to a 'Choice state' ('Is store open?'). If 'Store is not open', it goes to 'Pass state' ('Store closed') and then 'End'. If 'Store is open', it continues to a 'Step Functions ListExecutions' state, then another 'Choice state' ('Is capacity available?'). If 'Capacity is available', it leads to 'Pass state' ('Order completed') and then 'End'. If 'Capacity is not available', it leads to 'EventBridge PutEvents' ('Awaiting order completion'), then 'Pass state' ('Out of capacity'), and finally 'Error Handling' before reaching 'End'. A yellow circle labeled '5a' is overlaid on the diagram. On the right, a detailed view of the 'Get "store open" info' step is shown, including its API (DynamoDB: GetItem), integration type (Optimized), API parameters (JSON code), IAM role for cross-account access, and next state ('Is store open?').



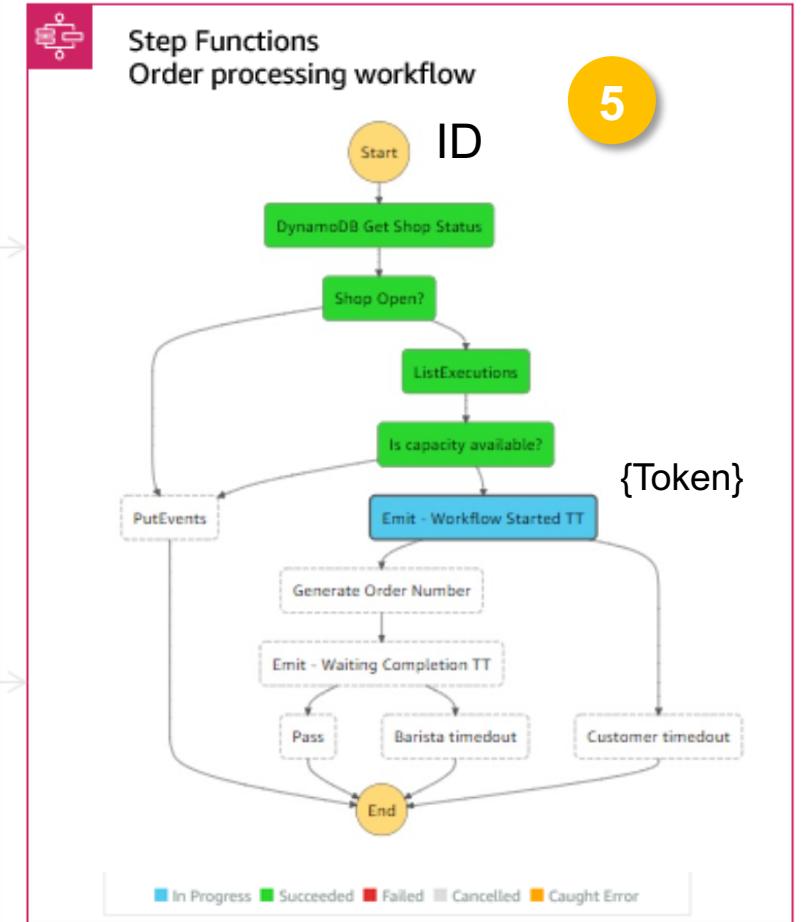
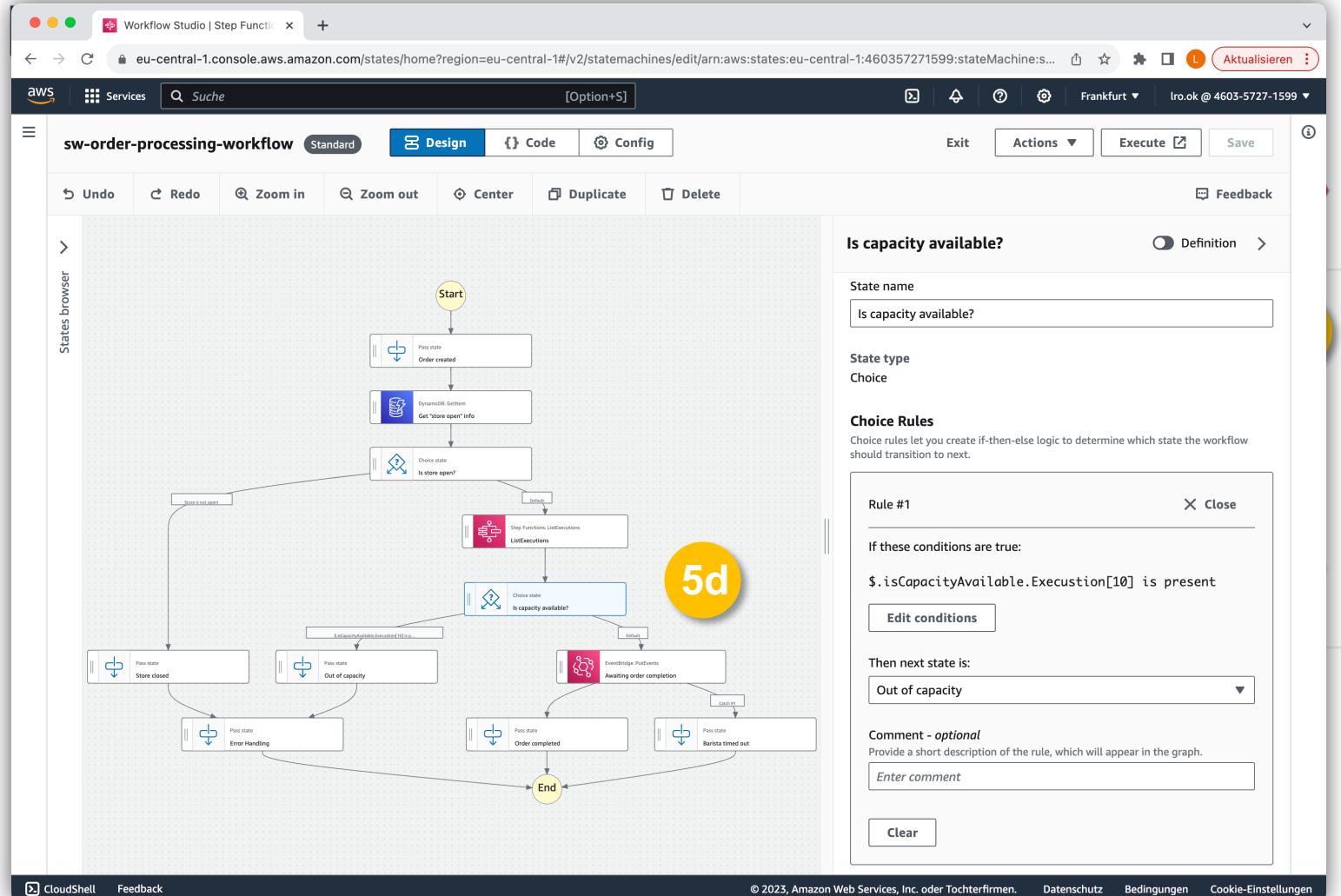
Serverless Coffee „Code scanned“



Serverless Coffee „Code scanned“



Serverless Coffee „Code scanned“



Serverless Coffee „Code scanned“

Workflow Studio | Step Functions

eu-central-1.console.aws.amazon.com/states/home?region=eu-central-1#/v2/statemachines/edit/arn:aws:states:eu-central-1:460357271599:stateMachine:sw-order-processing-workflow

Standard [Option+S]

Design Code Config

Undo Redo Zoom in Zoom out Center Duplicate Delete Feedback

Exit Actions Execute Save

sw-order-processing-workflow

Awaiting order completion

Definition API EventBridge: PutEvents Integration type Info Optimized

API Parameters JSON object containing the parameters to pass into this API. Contains sample values. Update the JSON with your own parameter values. Note: parameter names must be in PascalCase.

```
3 {
  "Detail": {
    "Message": "Thanks for 'submitting an order'",
    "TaskToken.$": "$$.Task.Token",
    "orderId.$": "$.orderToCreate.orderId",
    "userId.$": "$.orderToCreate.userId",
    "drink.$": "$.orderToCreate.drink"
  },
  "DetailType": "OrderProcessor.WaitingCompletion",
  "EventBusName": "sw-event-bus",
  "Source": "ok.serverlessworkshop"
}
```

Must be valid JSON. To reference a node in this state's JSON input, the key must end with ".\$" (for example "key2.\$": "\$.inputValue"). [Info](#)

Wait for callback - optional

Pause the execution at this state until the execution receives a callback from SendTaskSuccess or SendTaskFailure APIs with the task token.

Start → Pass state Order created → DynamoDB GetItem Get "store open" info → Choice state Is store open? → Default: Step Functions ListExecutions ListExecutions → Choice state Is capacity available? → Default: EventBridge PutEvents Awaiting order completion → Catch #1: Pass state Order completed → Pass state Barista timed out → End

Shop is not open: Pass state Store closed → Error Handling → End

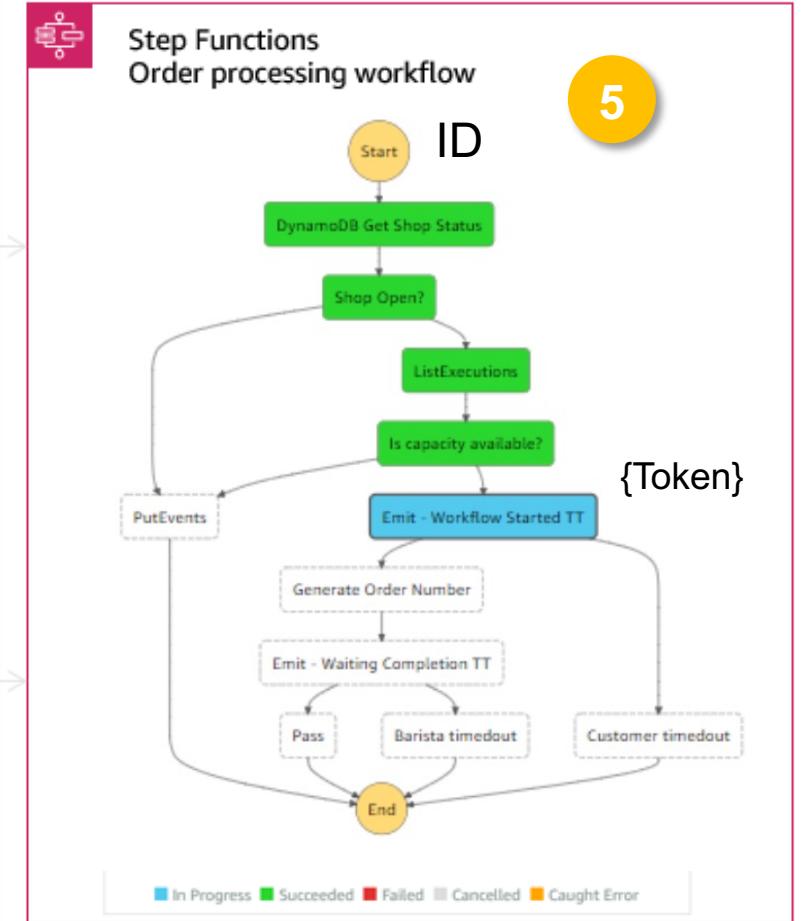
Out of capacity: Pass state Out of capacity → End

States browser

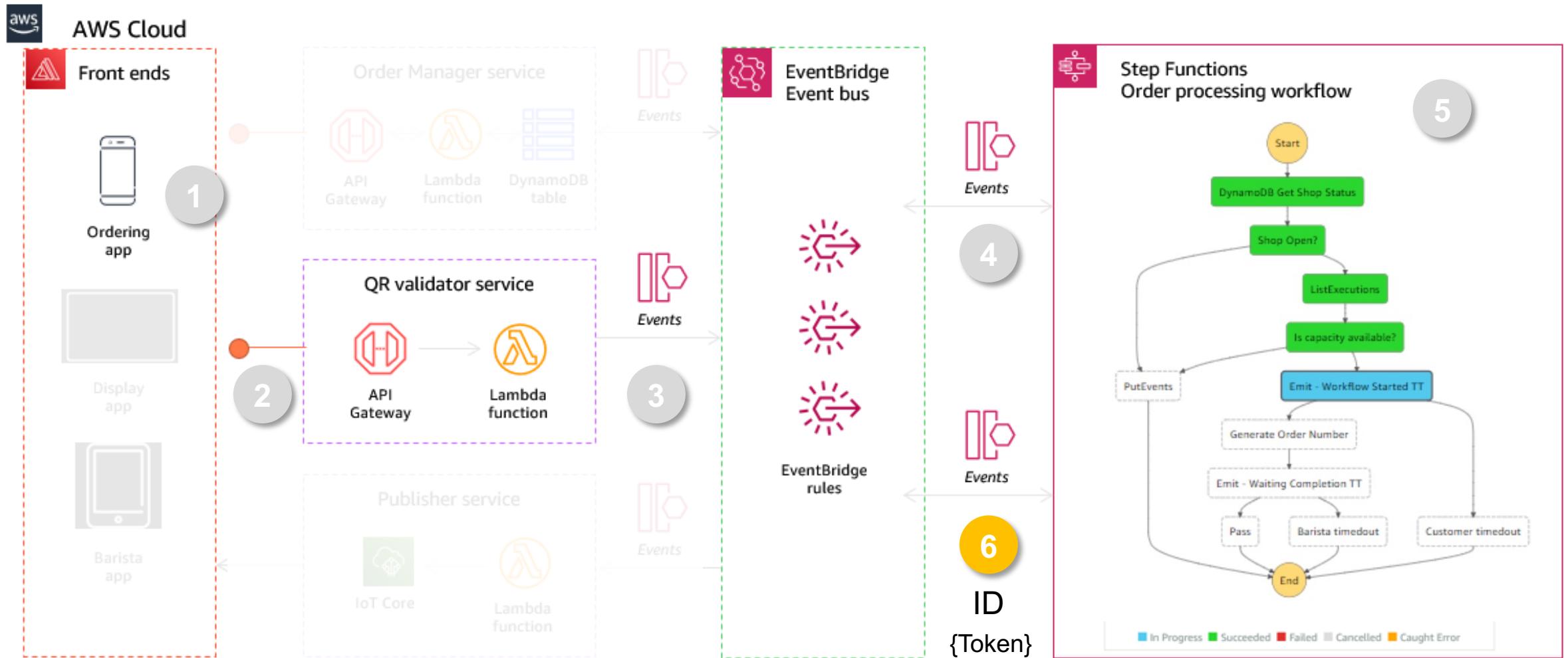
CloudShell Feedback

© 2023, Amazon Web Services, Inc. oder Tochterfirmen. Datenschutz Bedingungen Cookie-Einstellungen

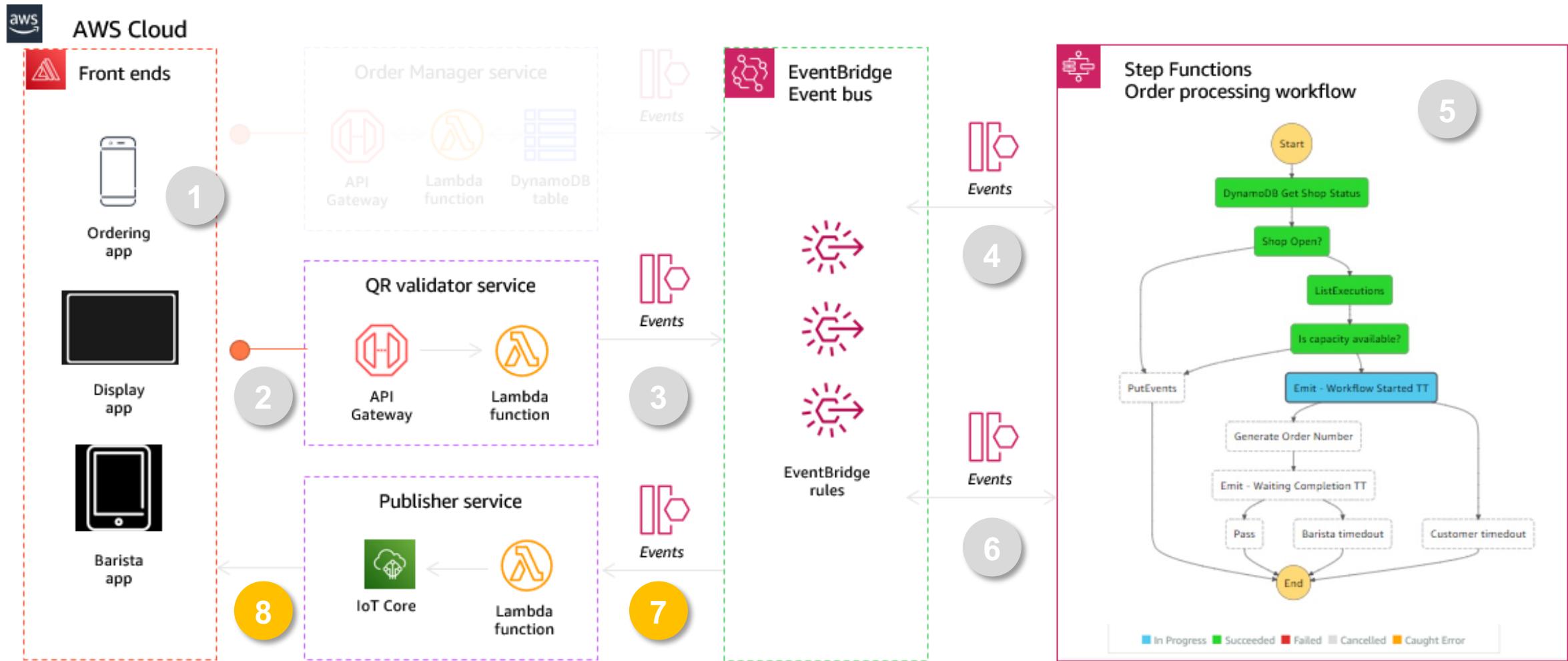
5e



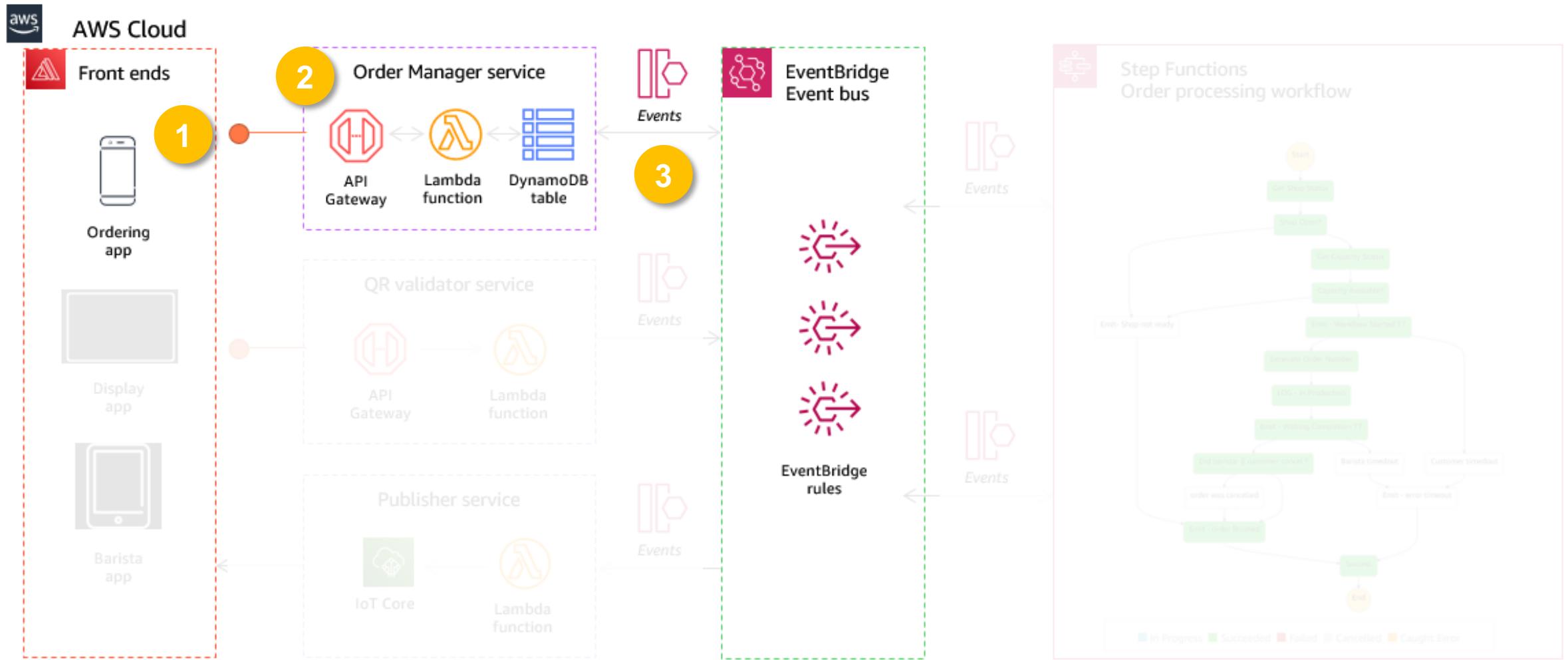
Serverless Coffee „Code scanned“



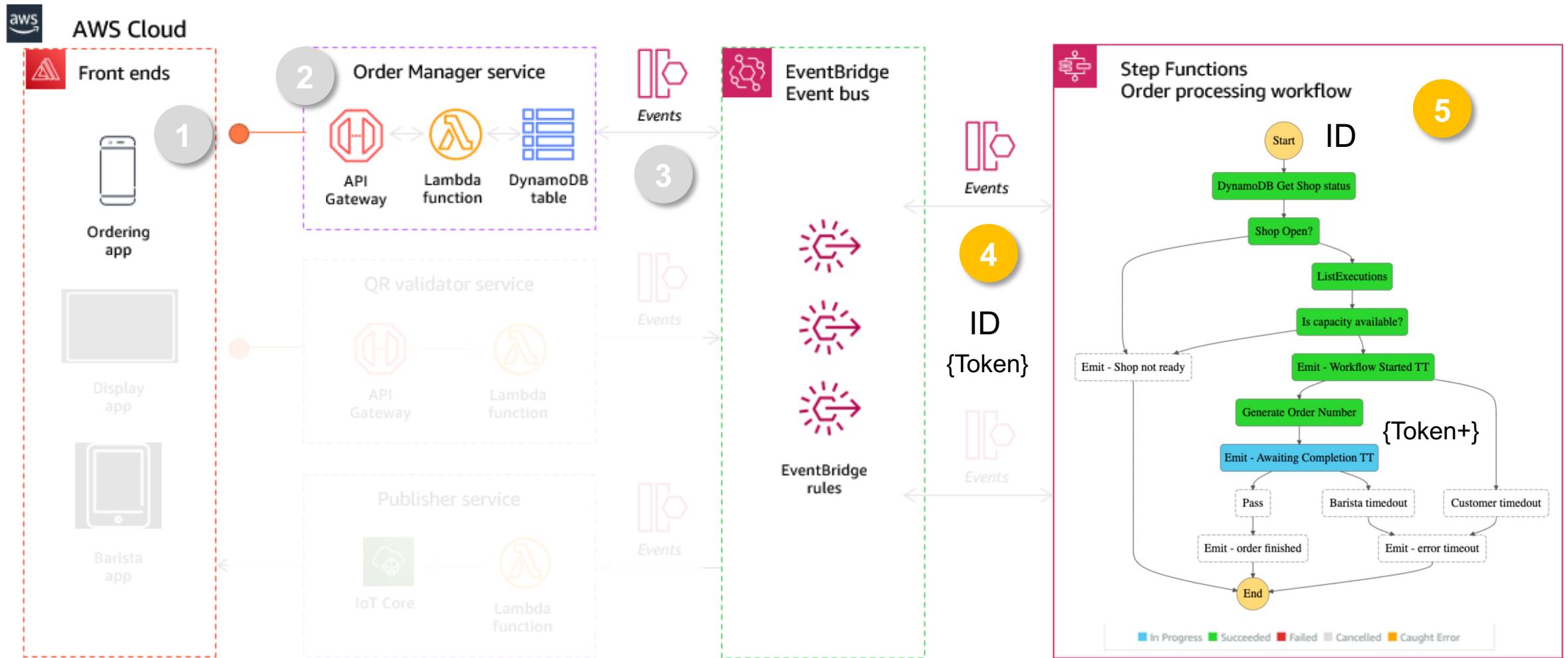
Serverless Coffee „Code scanned“



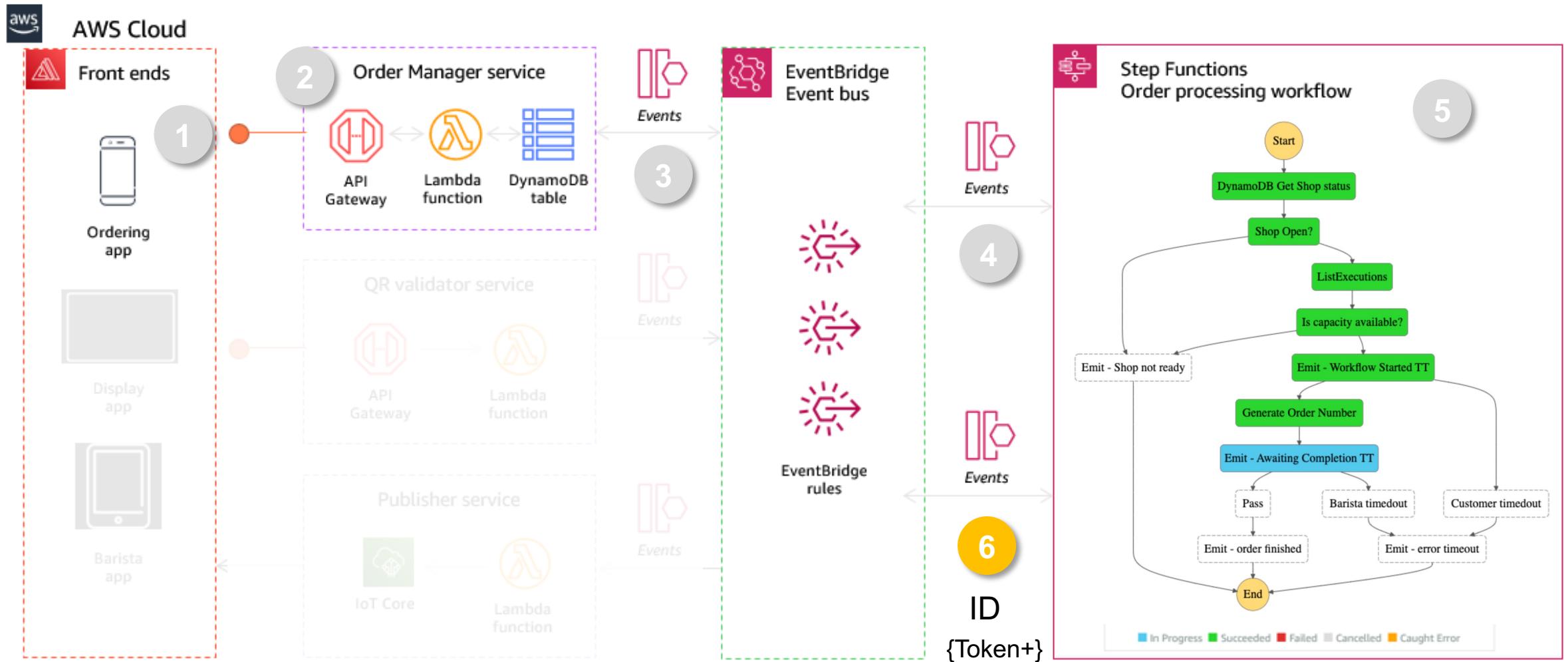
Serverless Coffee „Place Order“



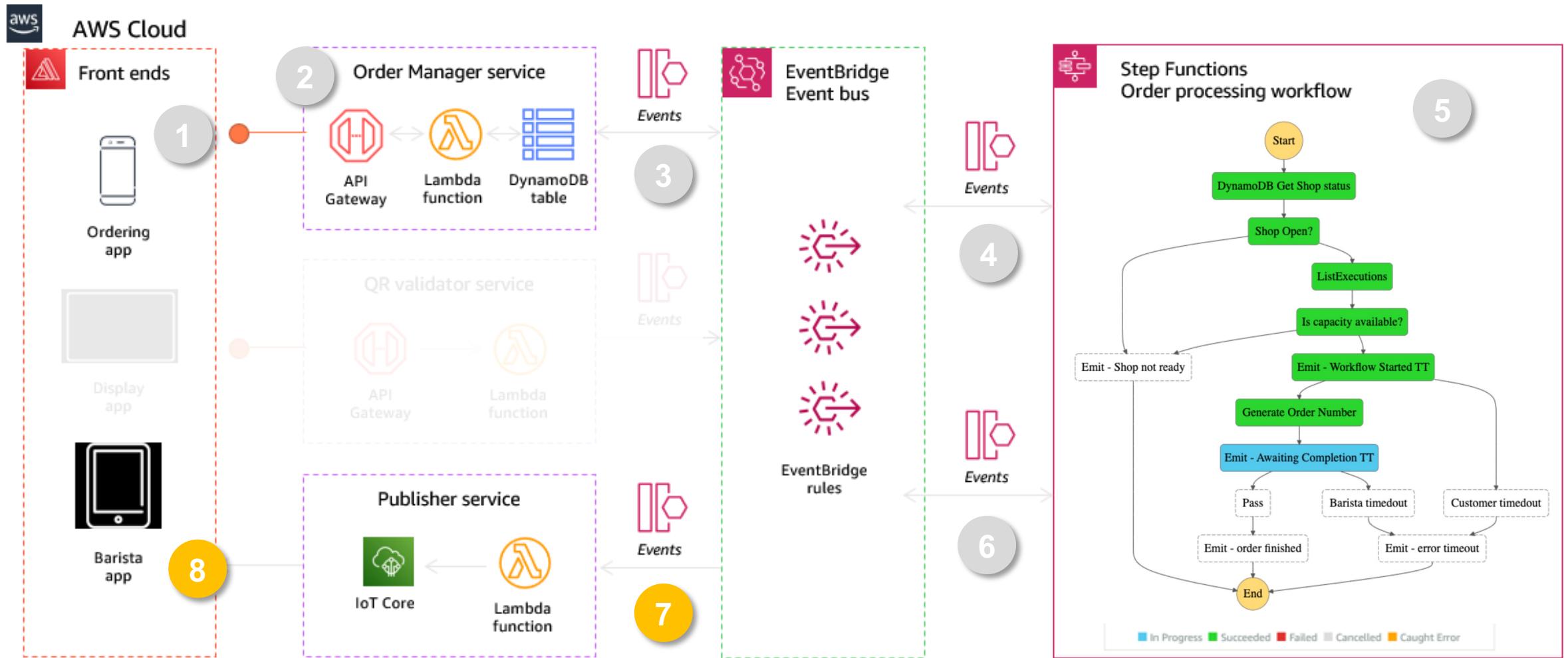
Serverless Coffee „Place Order“



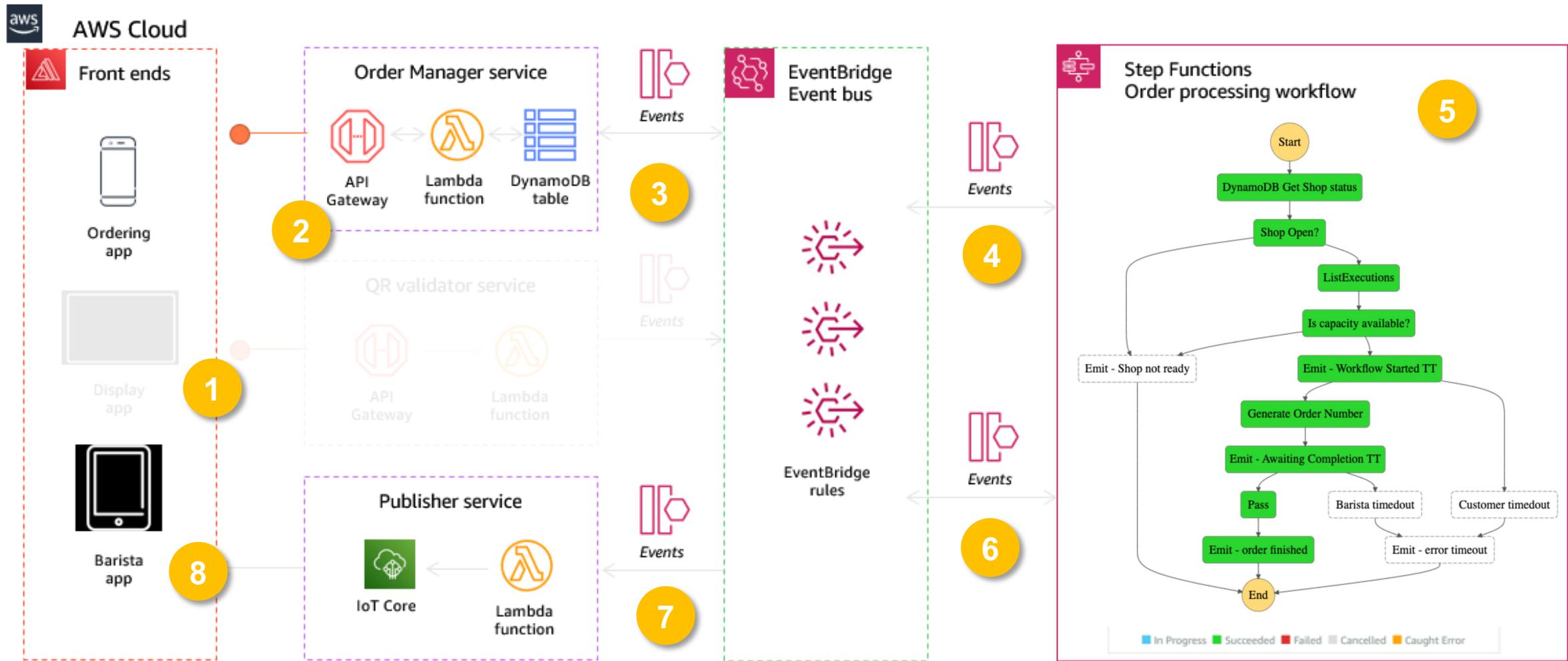
Serverless Coffee „Place Order“



Serverless Coffee „Place Order“



Serverless Coffee „Coffee in Making“



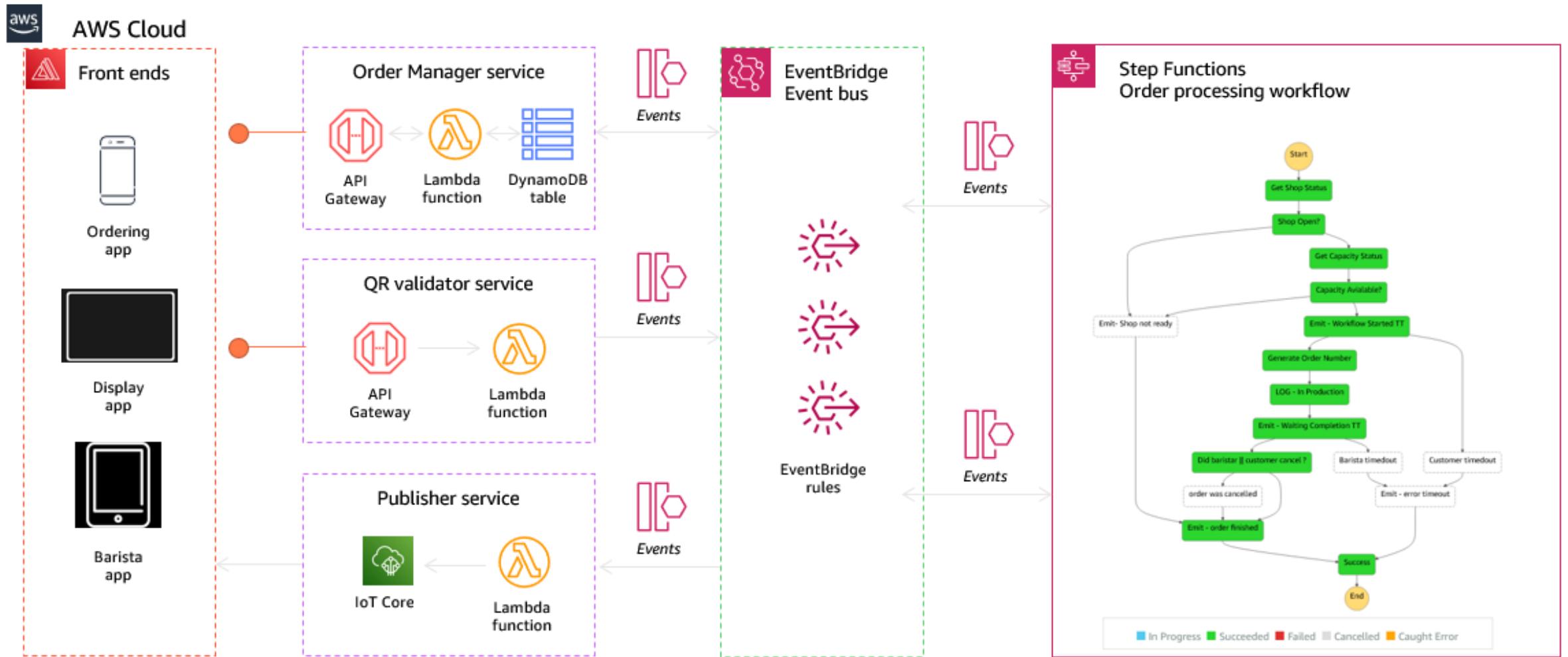
OMG!?



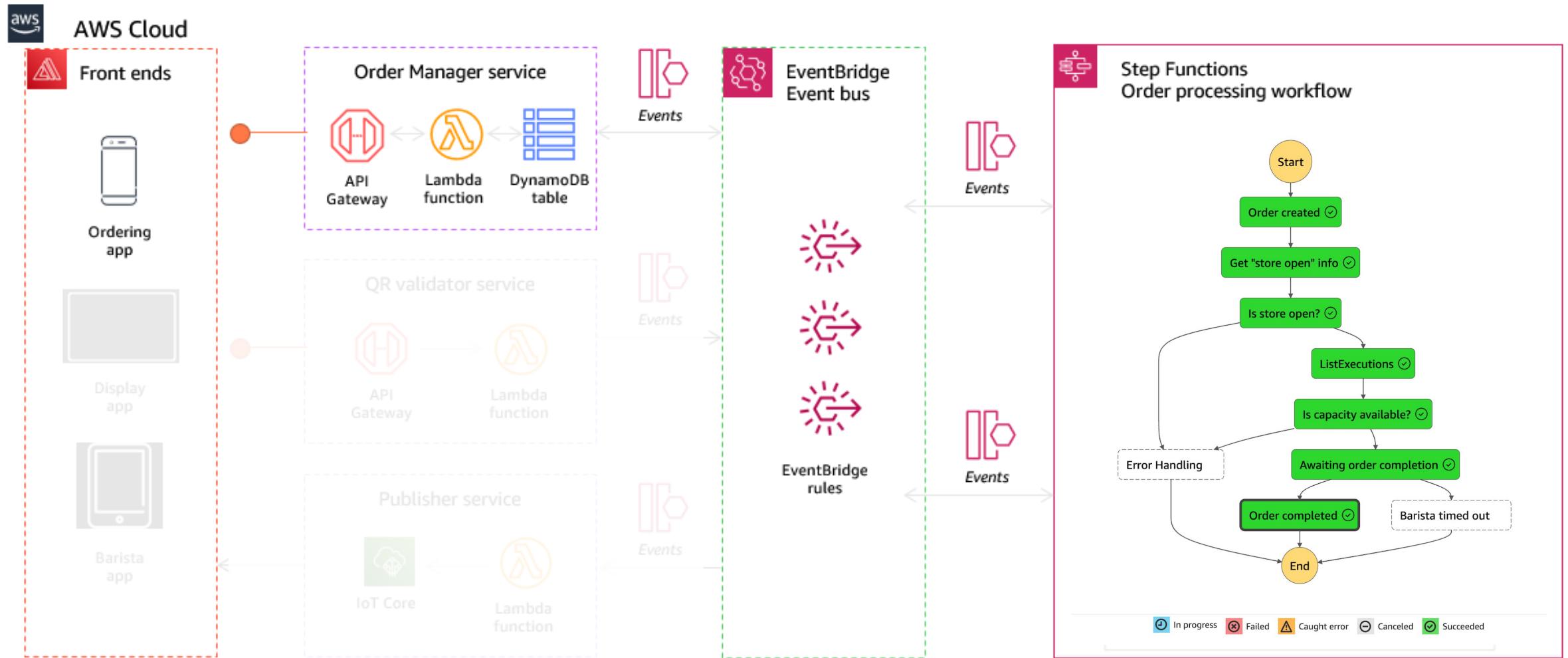


*„Aber eigentlich
möchte ich doch nur
einen Kaffee trinken.“*

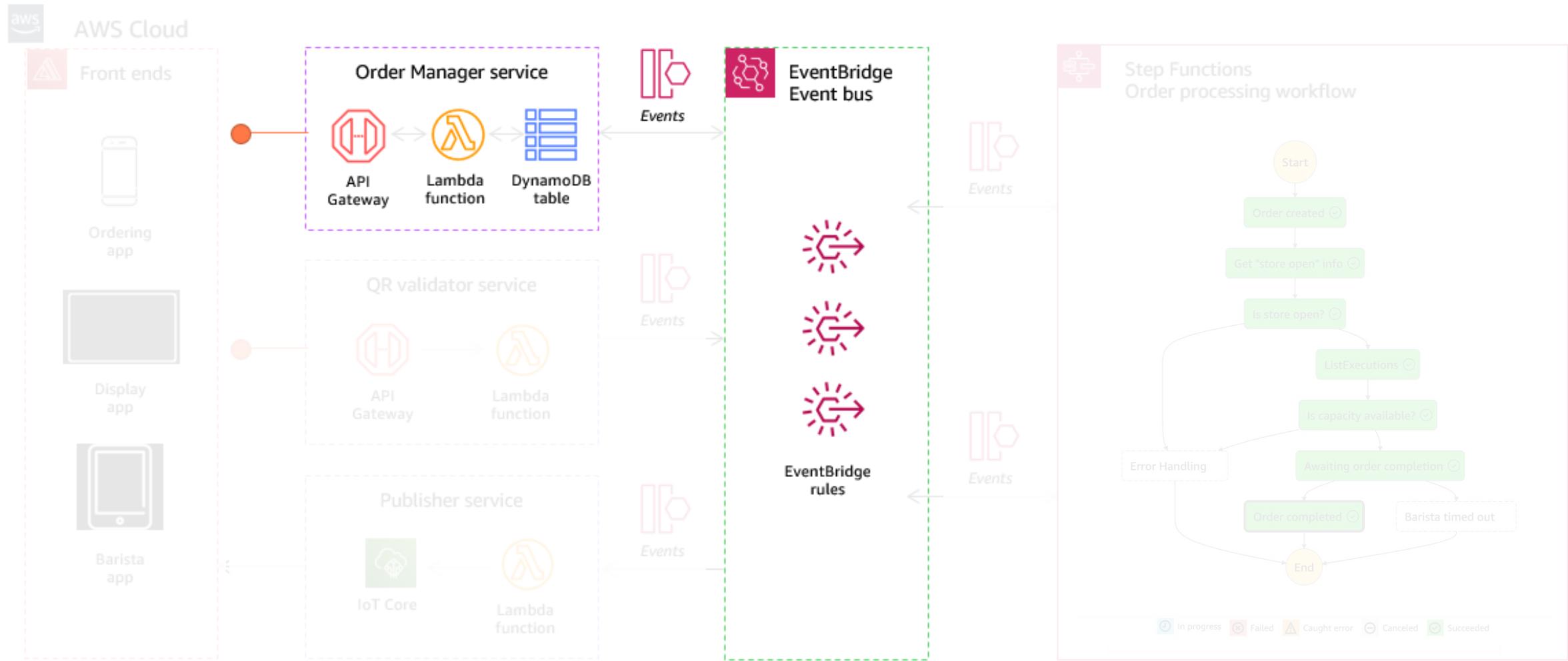
Serverless Coffee „Full Edition“



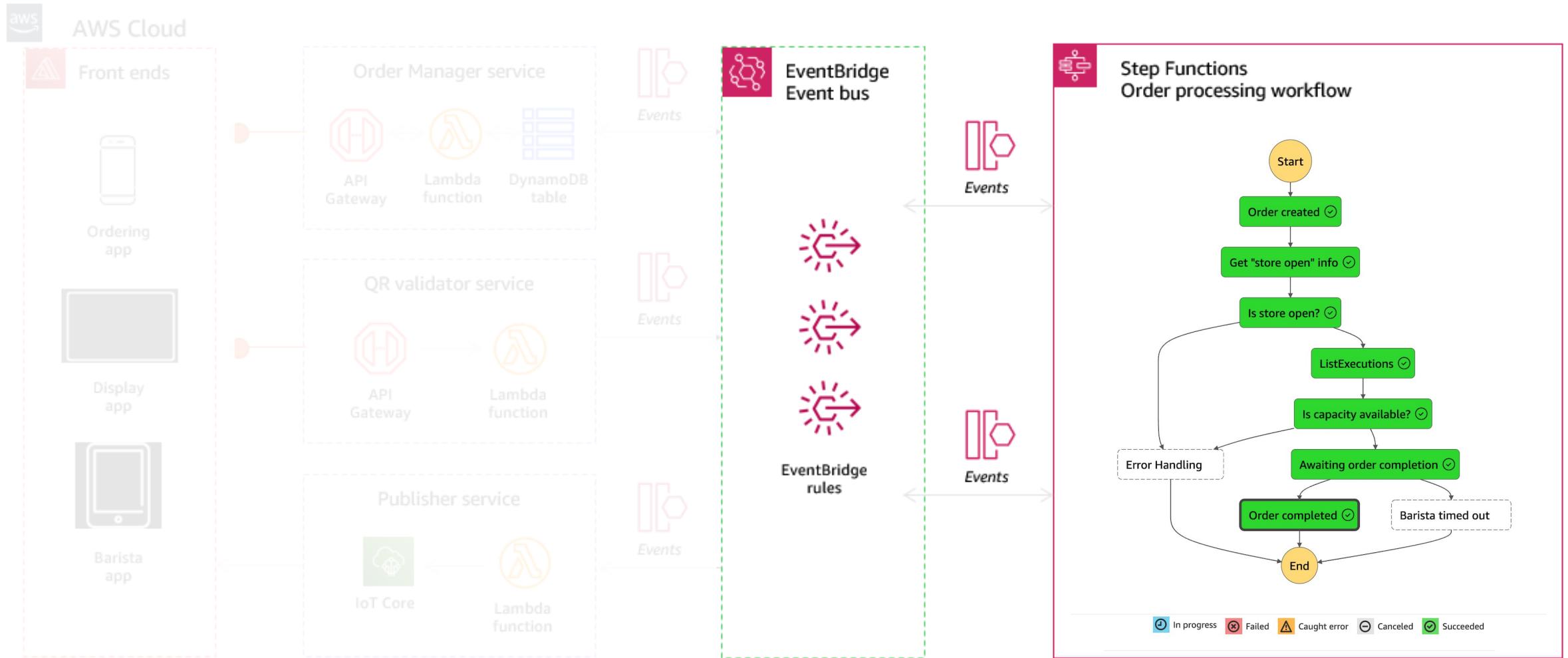
Serverless Coffee „Light Edition“



Serverless Coffee „Order Management“



Serverless Coffee „Order Processing“



Serverless Coffee „API Gateway“





Agenda

Hello World & Set-up

Order Management

Order Processing

API Gateway

Testing



Serverless Coffee

Hello Coffee & Set-up



Hello
Coffee

**“Run your business code
highly-available
in the cloud in response
to events and scale without
any servers to manage.”***

* AWS Lambda Advertising

“ *Function as self-contained application*

Serverless Function: Entwickler schreibt eine Business-Funktion in einer der unterstützten Programmiersprachen, „bundled“ diese mit den entsprechenden Abhängigkeiten (LIBs) und lädt sie in die Cloud.

Serverless Environment: Führt die Funktion bei „Aufruf“ in der passenden Runtime effizient, flexibel und hoch skalierbar aus.

“

Serverless *no need to maintain*

Entwickler: Fokussiert sich ausschließlich auf die Umsetzung der Business-Logik und das Erstellen des Function-Bundle.

Cloud Provider: liefert und maintains rundum-sorglos Umgebung für die Serverless Functions, inklusive etwaiger Cloud Services (z.B. Storage, DB, Streaming, AI).

■■■ Telekom.de LTE 12:45 ⚡ *

< Twitter +f

 **Kate Pearce**
@secvalve

The next time you try and use the word
"serverless" just remember it's like
calling takeout "kitchenless".

[Original \(Englisch\) übersetzen](#)

30.05.17, 23:32

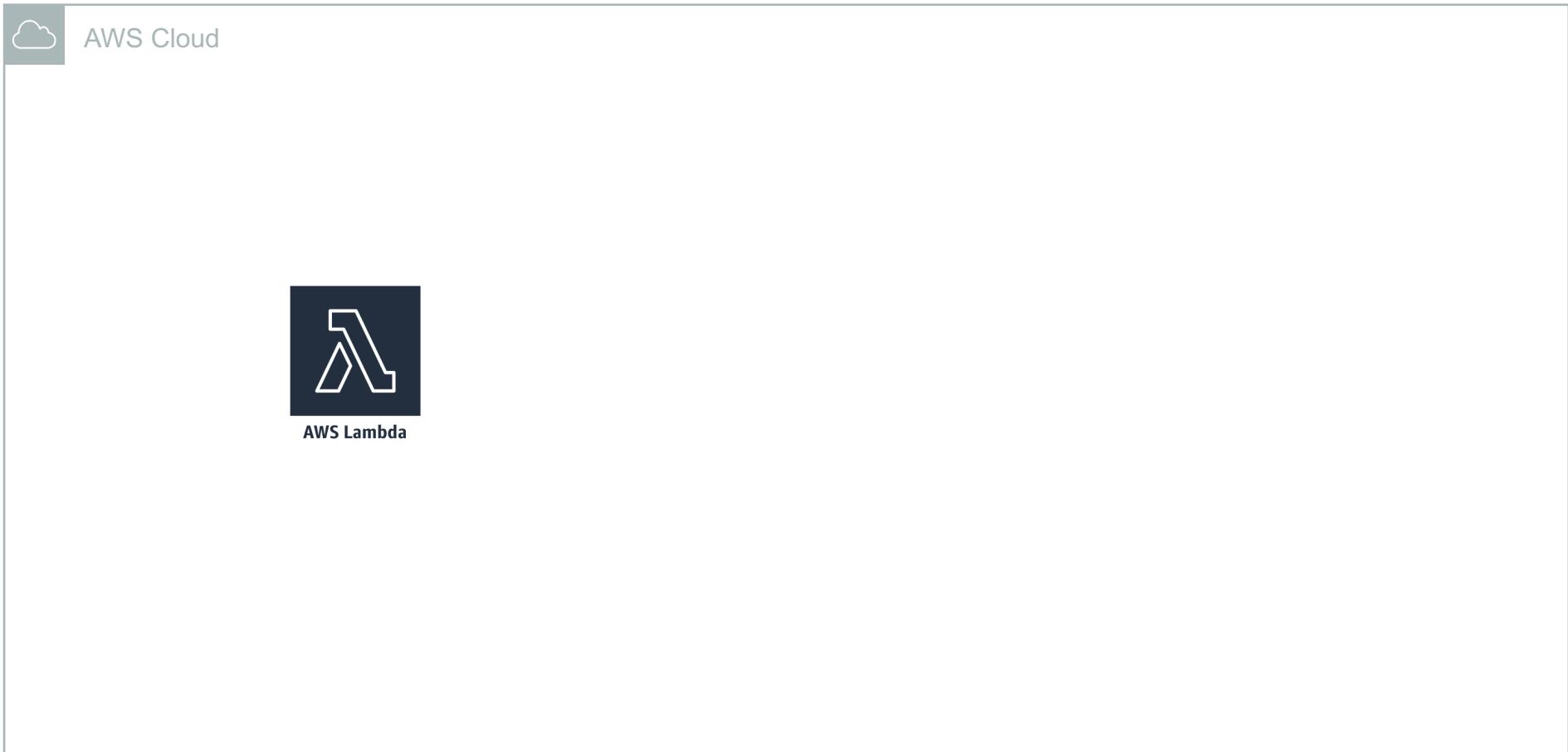
HANDS-ON

Hello Serverless Coffee

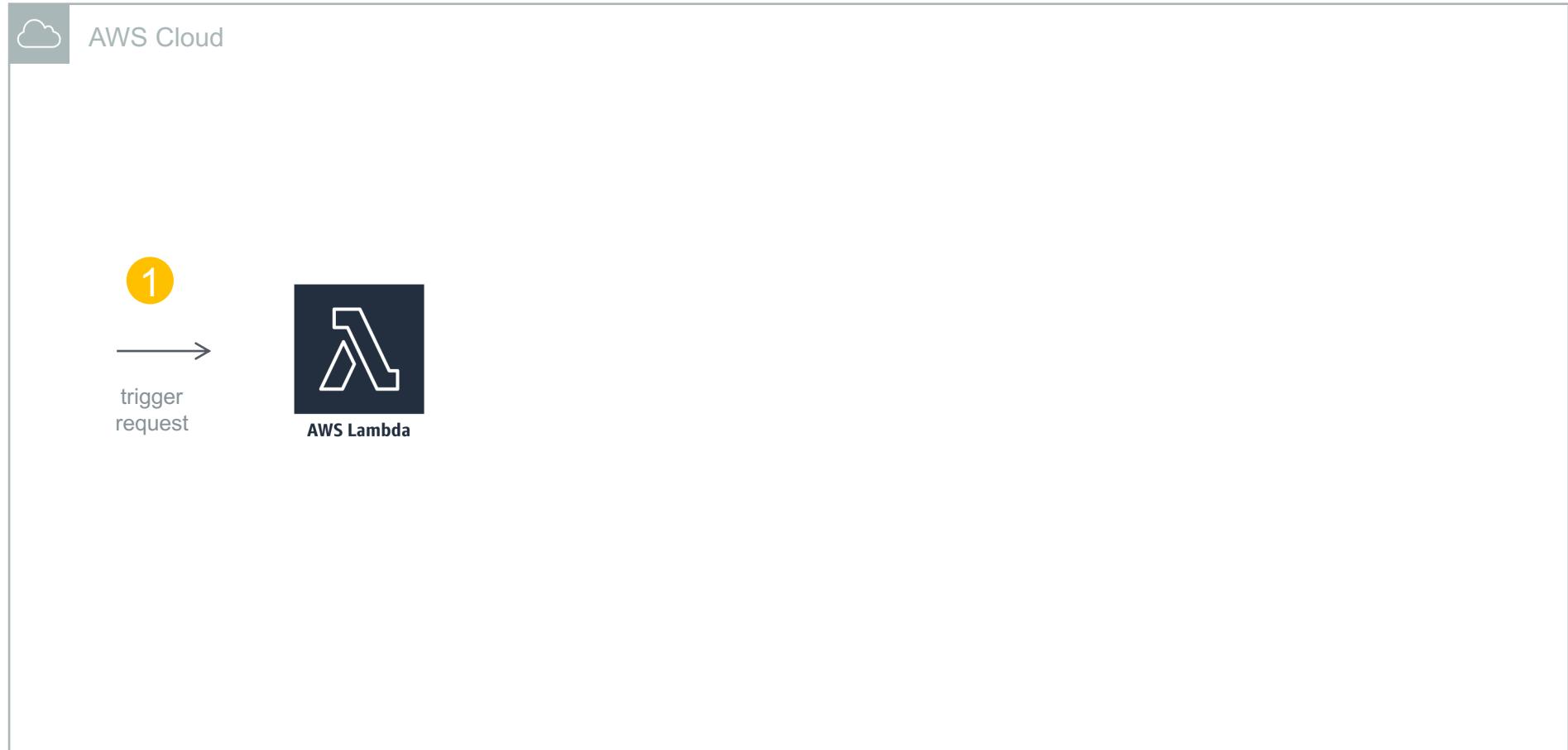
- Implementation
- Upload
- Testing



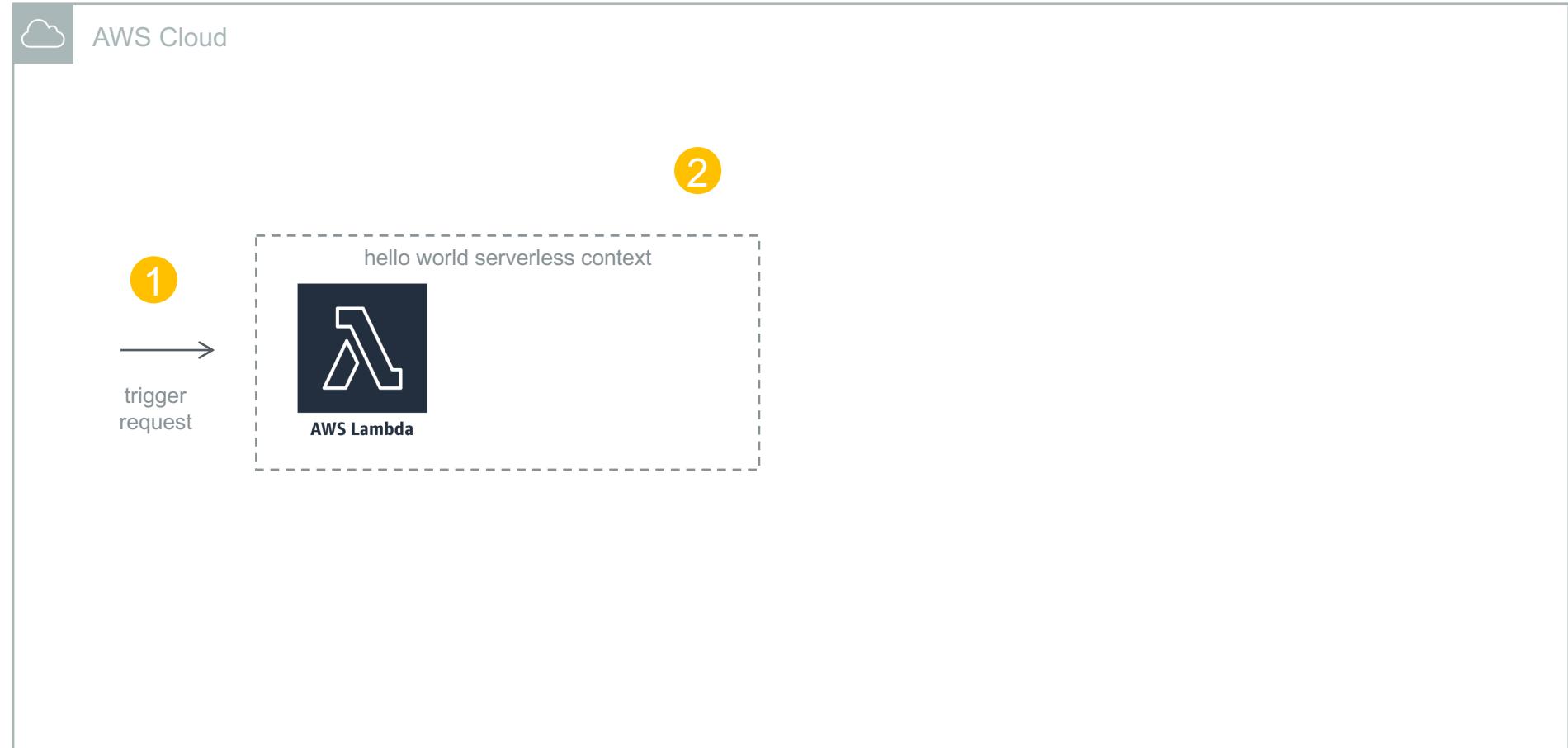
Hands-on: Hello Serverless Coffee



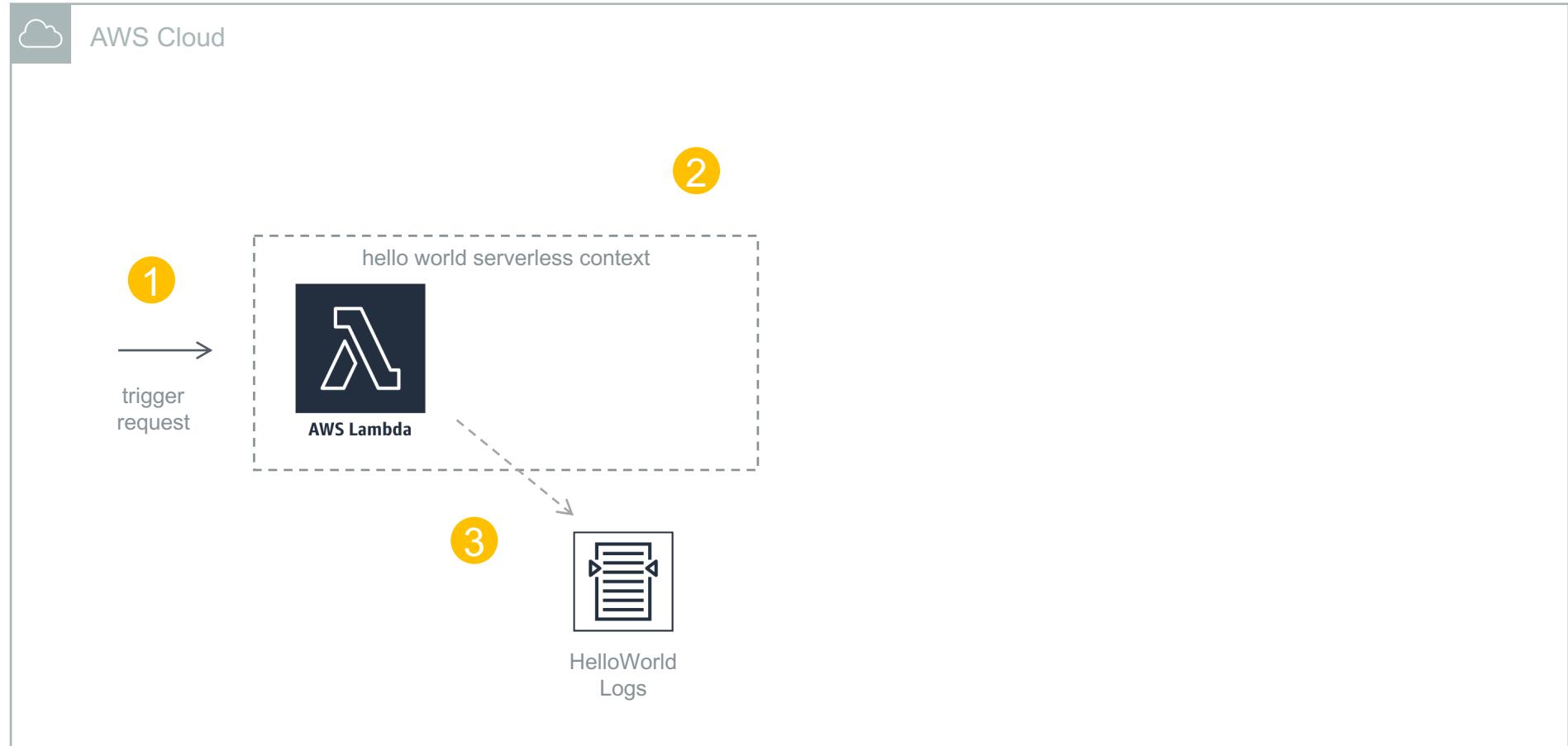
Hands-on: Hello Serverless Coffee



Hands-on: Hello Serverless Coffee



Hands-on: Hello Serverless Coffee



The screenshot shows the AWS Toolkit for IntelliJ IDEA interface. The project navigation bar at the top indicates the current file is `serverless-workshop > 01_hello_serverless > target > hello-serverless-world.jar`. The main window displays the Java code for `HelloWorld.java`:

```
serverless-workshop – HelloWorld.java [hello-serverless]
...
1  ...
16 package de.openknowledge.workshop.cloud.serverless.lambda;
17
18 import ...
19
20 /**
21  * Simplest aws lambda function class possible.
22  */
23
24 // de.openknowledge.workshop.cloud.serverless.lambda.HelloWorld::greet
25 public class HelloWorld {
26
27     /**
28      * Converts a name (first name, last name) into a greeting
29      *
30      * @param name Request object of type <code>HelloWorldRequest</code> with first name and last name
31      * @param context Lambda function context object of type <code>Context</code>
32      * @return Greeting string wrapped inside a response object of type <code>HelloWorldResponse</code>
33
34 }
35
36 }
```

The code editor has syntax highlighting for Java and annotations. The terminal below shows Maven validation warnings:

```
[WARNING] Plugin validation issues were detected in 2 plugin(s)
[WARNING]
[WARNING] * org.apache.maven.plugins:maven-shade-plugin:3.4.1
[WARNING] * org.apache.maven.plugins:maven-resources-plugin:3.3.0
[WARNING]
[WARNING] For more or less details, use 'maven.plugin.validation' property with one of the values (case insensitive): [BRIEF, DEFAULT, VERBOSE]
[WARNING]
(base) lars@MacBook-Pro-von-Lars-7 01_hello_serverless %
```

The bottom status bar shows the time as 29:14, encoding as LF UTF-8, 4 spaces, and 6 connections to AWS.

Hello Serverless Coffee: Source Code

The screenshot shows the AWS Lambda console interface for a function named "sw-hello-serverless-world".

Function Overview:

- Name:** sw-hello-serverless-world
- Layers:** (0)
- Triggers:** + Auslöser hinzufügen
- Targets:** + Ziel hinzufügen

Description: -

Last Change: vor 6 Minuten

Function ARN: arn:aws:lambda:eu-central-1:460357271599:function:sw-hello-serverless-world

Function URL: Info -

Code Tab: The "Code" tab is selected.

Code Source: Codequelle Info

Note: Der Code-Editor unterstützt die Java 11 Laufzeit nicht.

Bottom Navigation: CloudShell, Feedback, Sprache, © 2023, Amazon Web Services, Inc. oder Tochterfirmen., Datenschutz, Bedingungen, Cookie-Einstellungen

Um Ihre Funktion aufzurufen, ohne ein Ereignis zu speichern, ändern Sie das Ereignis und wählen Sie dann „Test“ aus. Lambda verwendet das geänderte Ereignis, um Ihre Funktion aufzurufen, überschreibt das ursprüngliche Ereignis jedoch erst, wenn Sie „Save changes“ (Änderungen speichern) auswählen.

Testereignisaktion

Ereignis-Name

greetings-lars

Ereignis-JSON

```
1+ {  
2 "firstName": "Lars", |  
3 "lastName": "Roekekamp"  
4 }
```

JSON formatieren

The screenshot shows the AWS Lambda console interface for a function named "sw-hello-serverless-world". The "Test" tab is selected. A green checkmark indicates the function executed successfully. The output log shows the response: {"greeting": "Hello, Lars Roewekamp! i am pleased to meet you."}. Below the log, there's an "Übersicht" (Overview) section with performance metrics and resource configuration. At the bottom, there's a "Protokollausgabe" (Log Output) section with CloudWatch logs for the request.

Ausführen der Funktion: erfolgreich (Protokolle [\[Details\]](#))

▼ Details

Der Bereich unten zeigt die letzten 4 KB des Ausführungsprotokolls.

```
{  
  "greeting": "Hello, Lars Roewekamp! i am pleased to meet you."  
}
```

Übersicht

| | |
|---|--------------------------------------|
| Code SHA-256 | Anfrage-ID |
| XgxbUxSP4o/8IOysTSIxsl5MZGjAEApavJzGCjLQ= | 0116cd0d-9990-4e93-8930-cec76b1b64e6 |
| Init-Dauer | Dauer |
| 494.65 ms | 14.96 ms |
| Abgerechnete Dauer | Ressourcen konfiguriert |
| 15 ms | 1024 MB |
| Maximal genutzter Speicher | |
| 77 MB | |

Protokollausgabe

Der folgende Abschnitt zeigt die Protokollierungsaufrufe in Ihrem Code. [Klicken Sie hier](#), um die entsprechende CloudWatch-Protokollgruppe anzuzeigen.

```
START RequestId: 0116cd0d-9990-4e93-8930-cec76b1b64e6 Version: $LATEST  
END RequestId: 0116cd0d-9990-4e93-8930-cec76b1b64e6  
REPORT RequestId: 0116cd0d-9990-4e93-8930-cec76b1b64e6 Duration: 14.96 ms Billed Duration: 15 ms Memory Size: 1024 MB Max Memory Used: 77 MB Init Duration: 494.65 ms
```

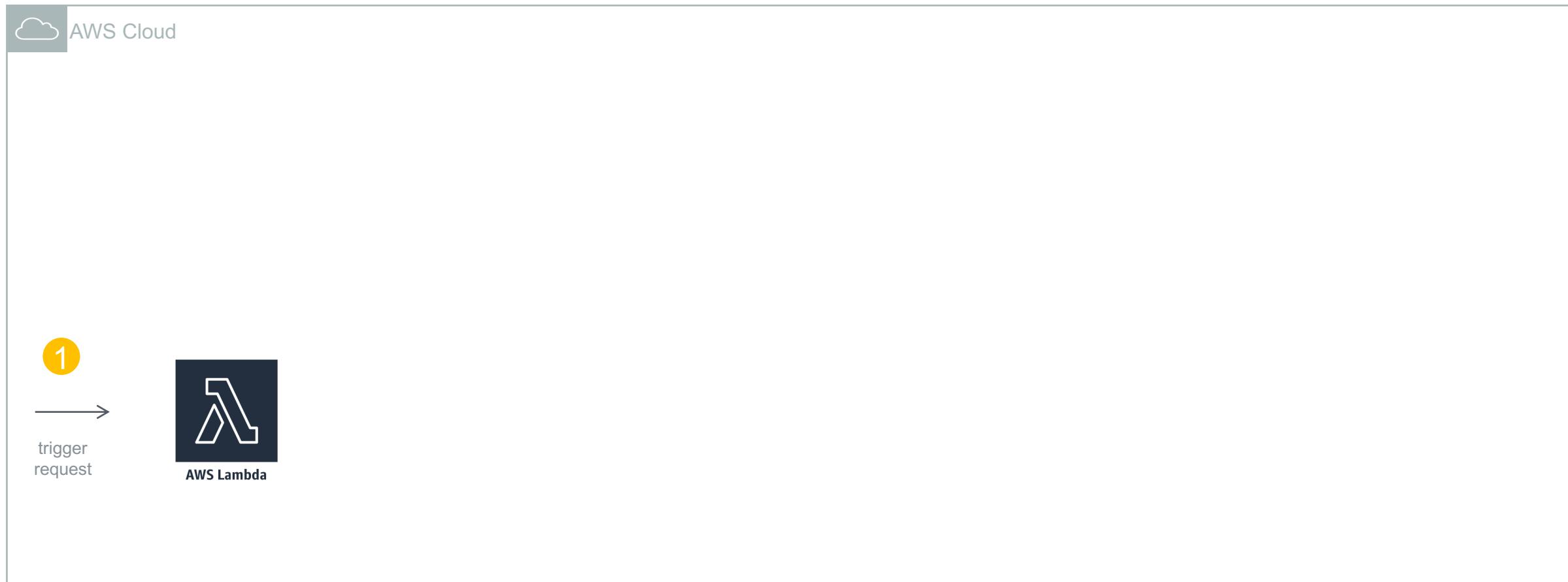
[CloudShell](#) [Feedback](#) [Sprache](#) © 2023, Amazon Web Services, Inc. oder Tochterfirmen. [Datenschutz](#) [Bedingungen](#) [Cookie-Einstellungen](#)

Hello Serverless Coffee: Successful Test

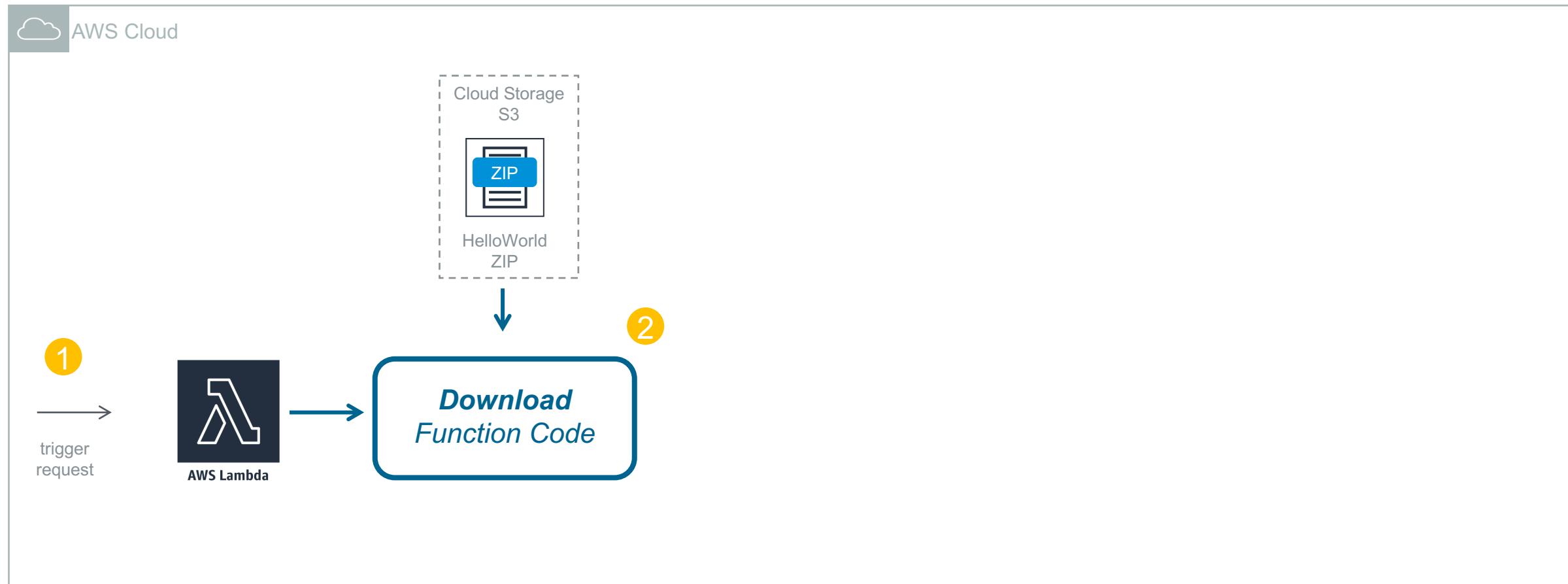
A magnifying glass with a black handle and silver frame is positioned on a light blue background with fine horizontal lines. The lens is focused on the center of the image, creating a circular magnification effect.

Serverless unter der Haube

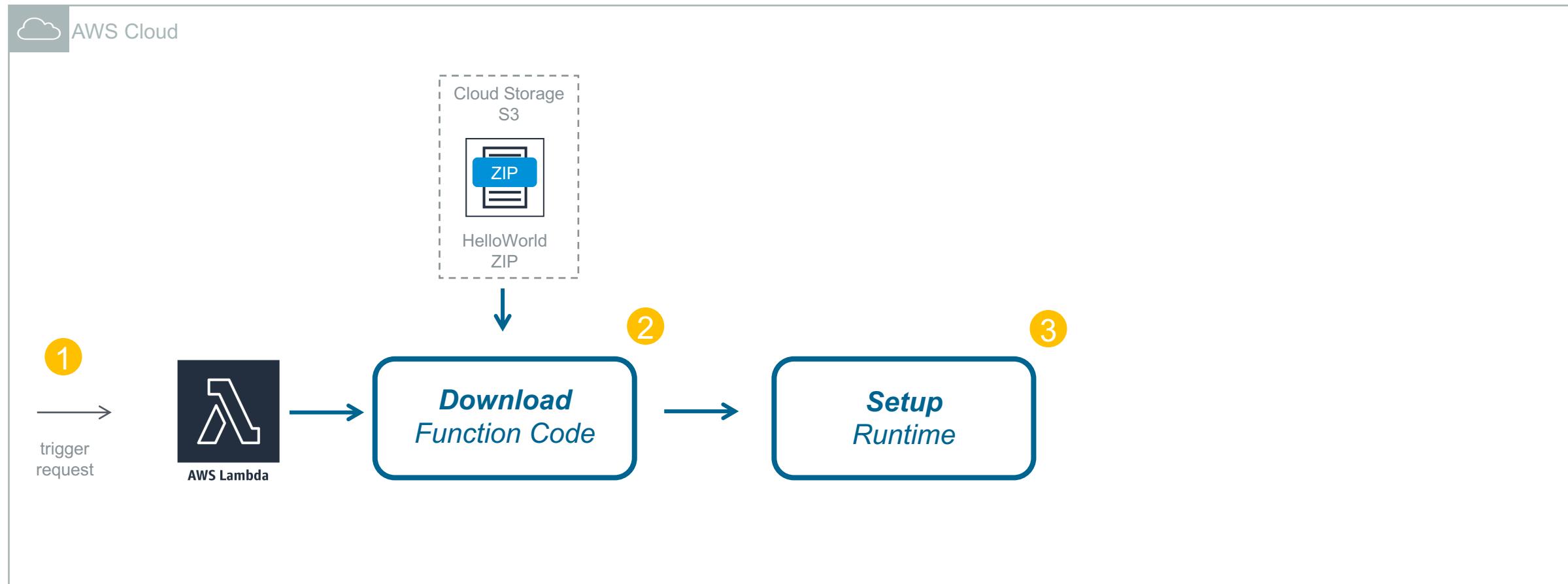
Hello Serverless „under the Hood“



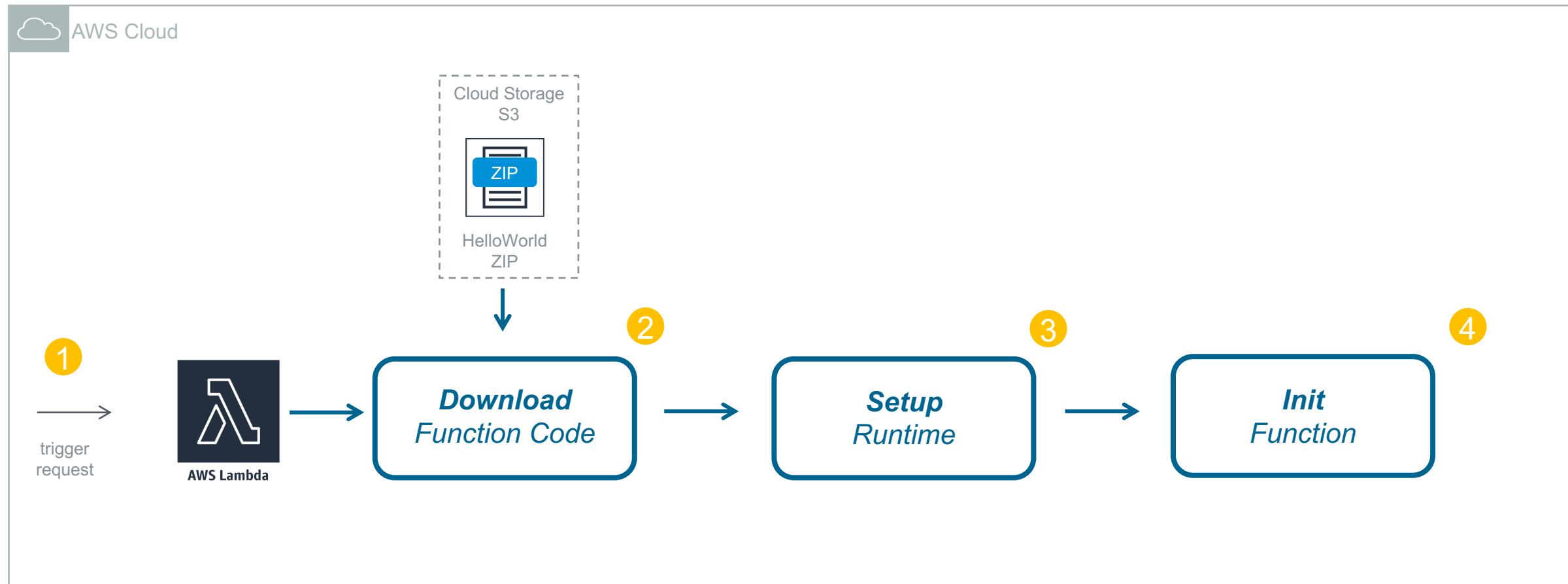
Hello Serverless „under the Hood“



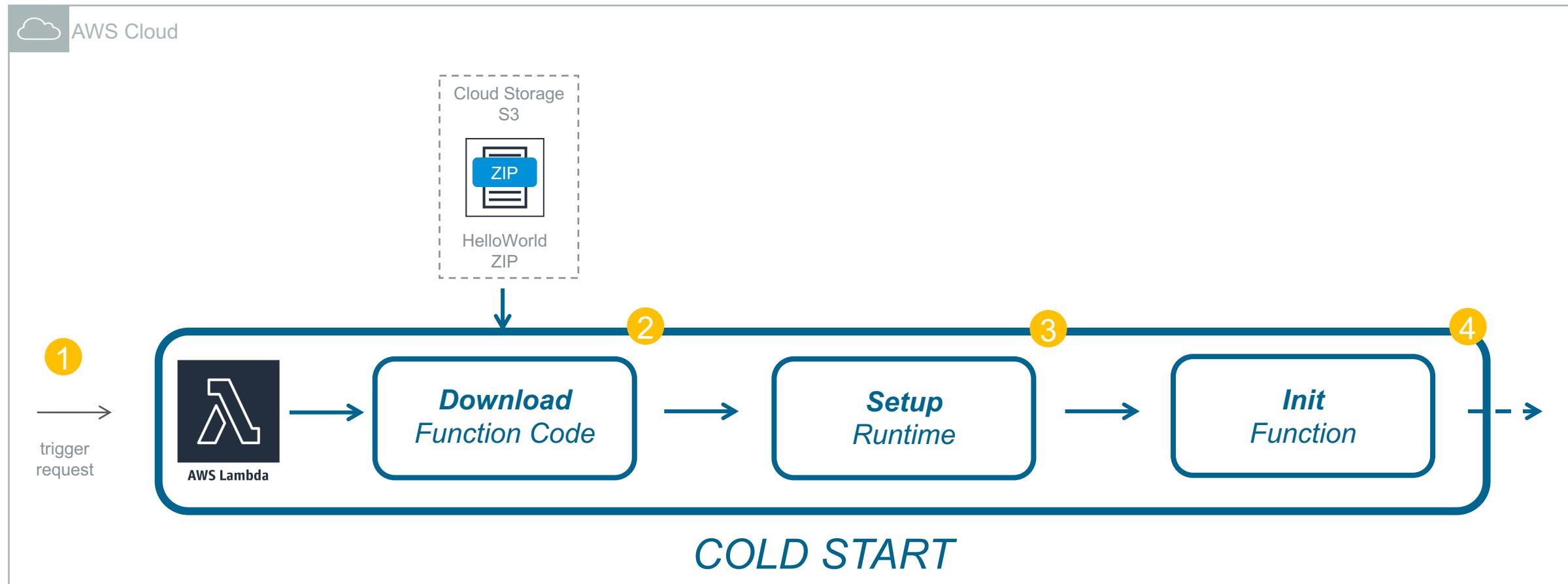
Hello Serverless „under the Hood“



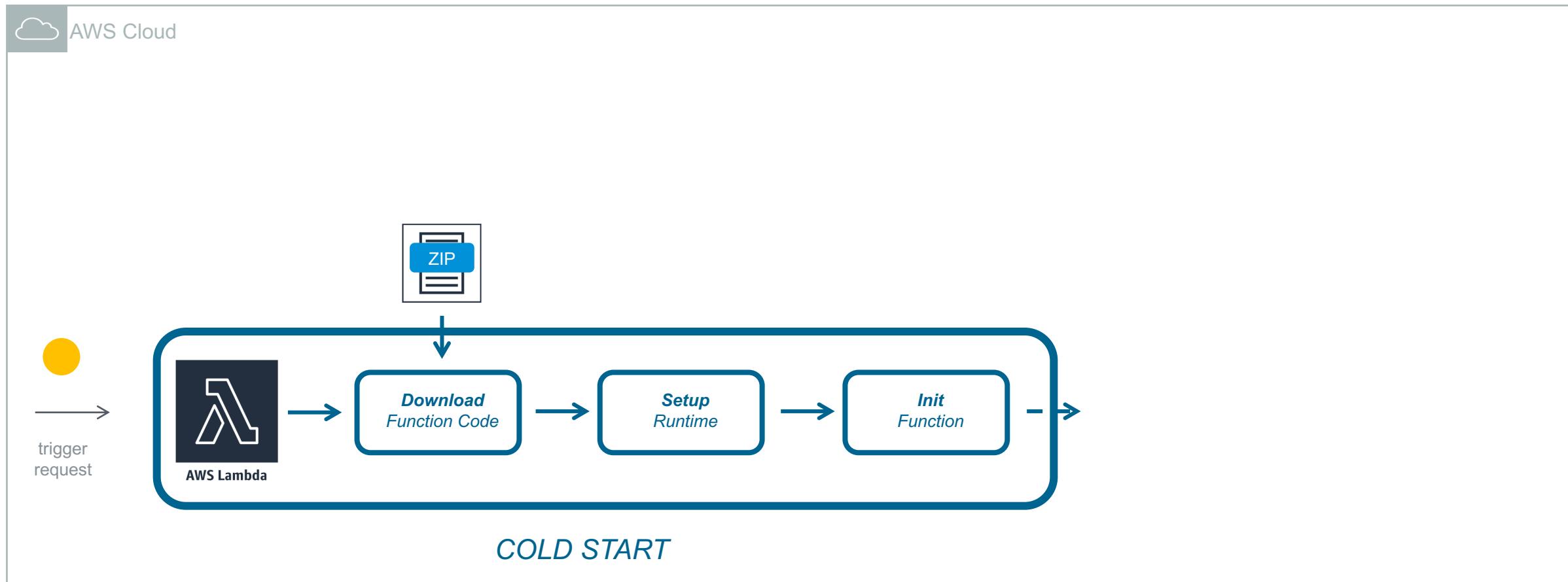
Hello Serverless „under the Hood“



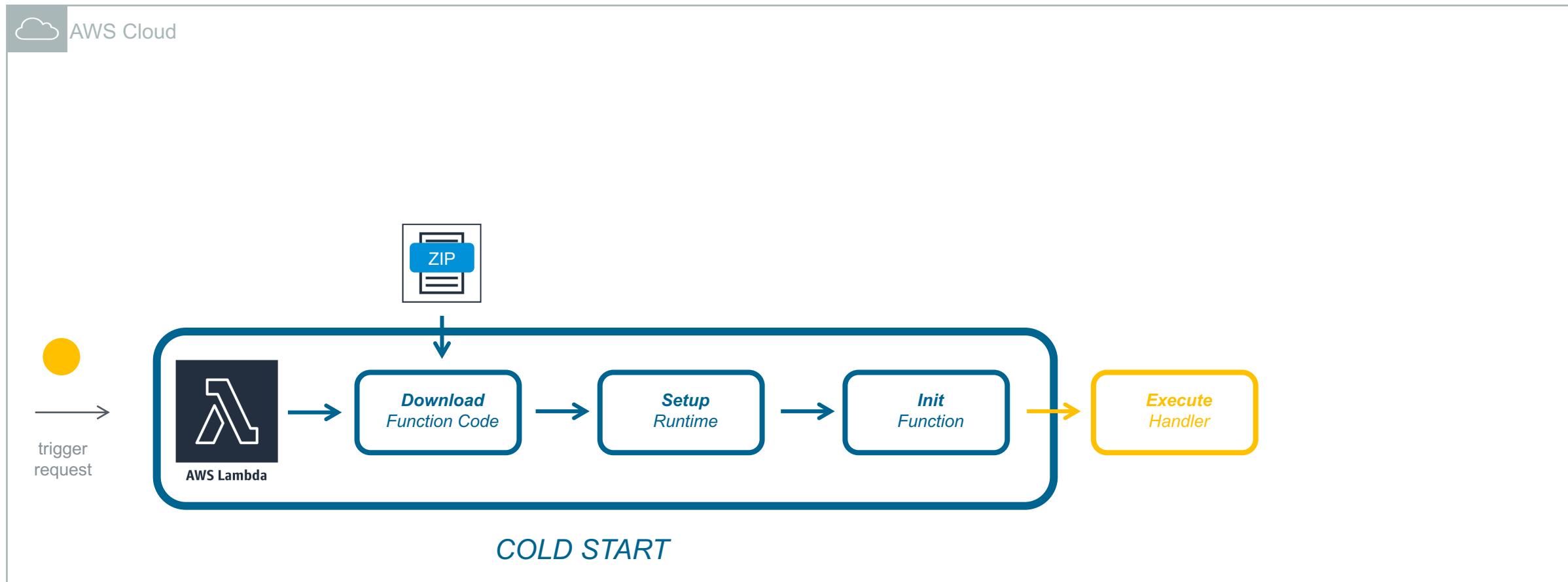
Hello Serverless „under the Hood“



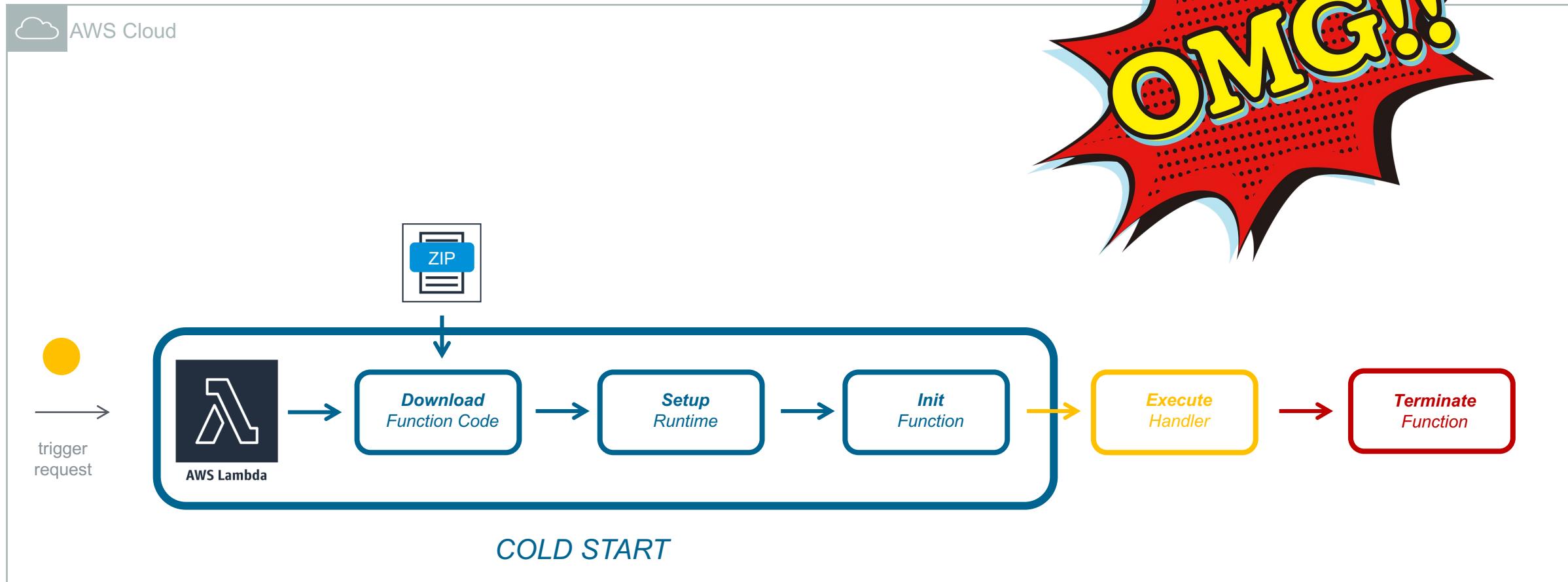
Hello Serverless „under the Hood“



Hello Serverless „under the Hood“



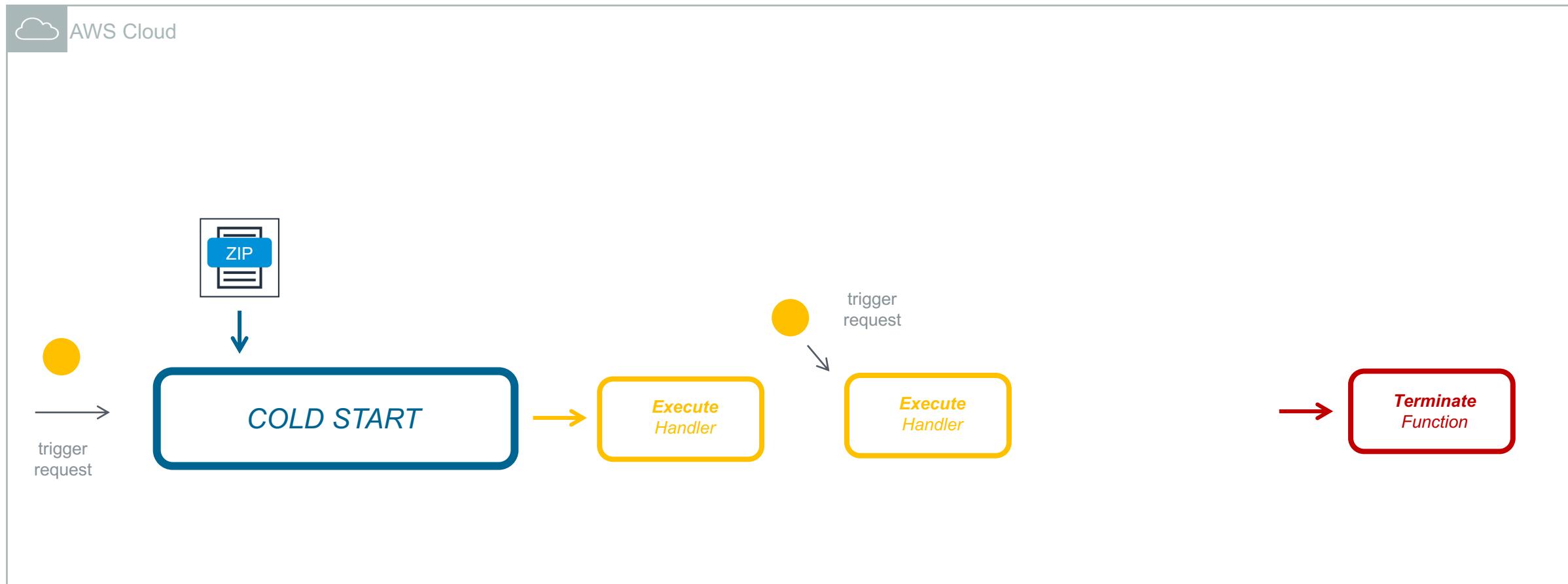
Hello Serverless „under the Hood“



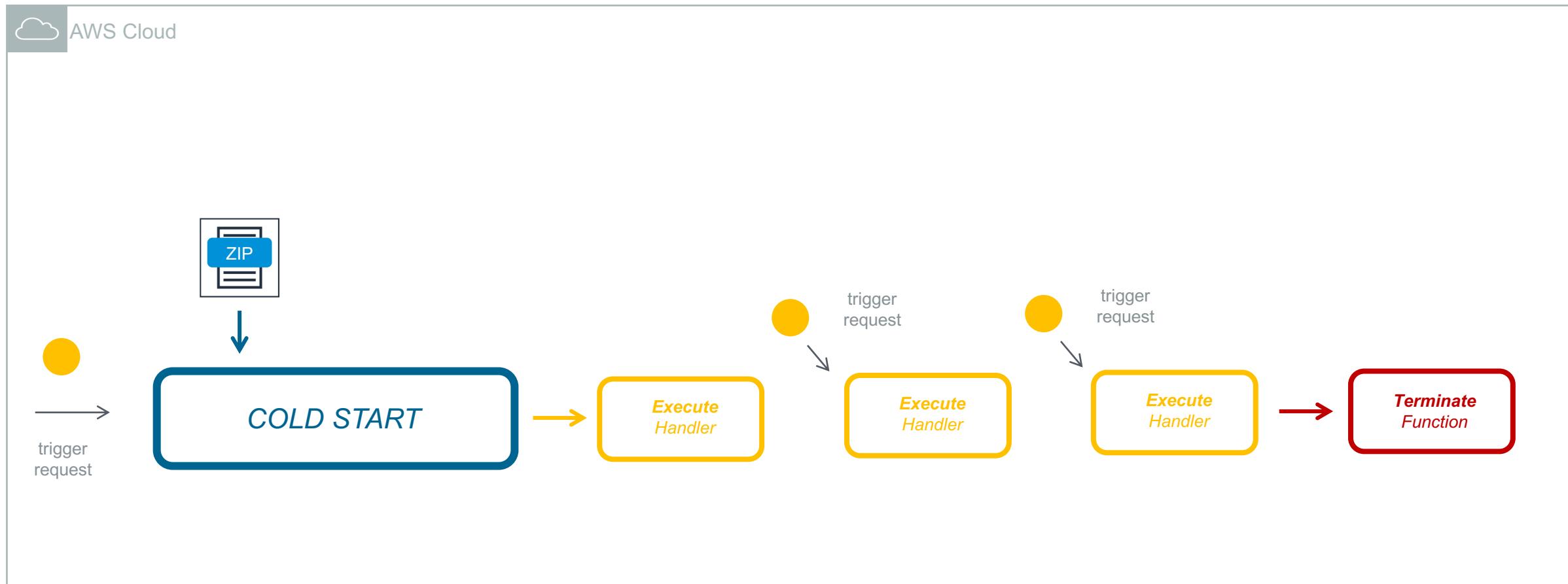
Hello Serverless „under the Hood“



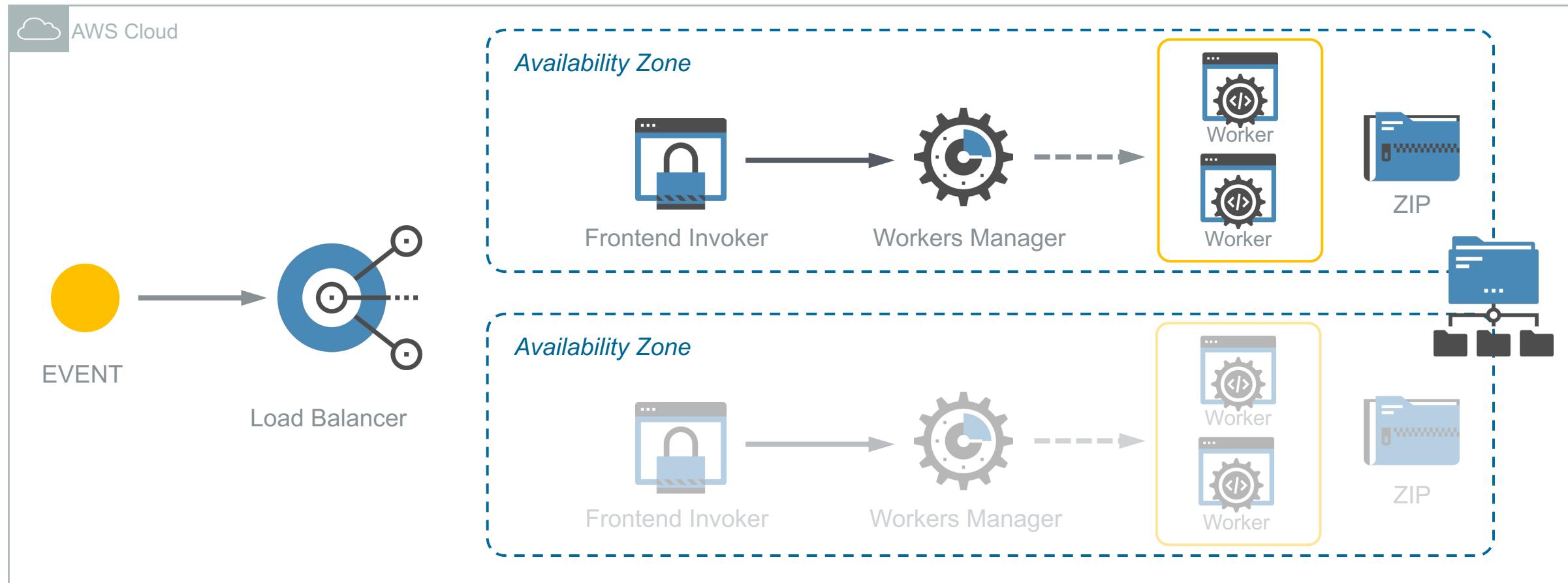
Hello Serverless „under the Hood“



Hello Serverless „under the Hood“



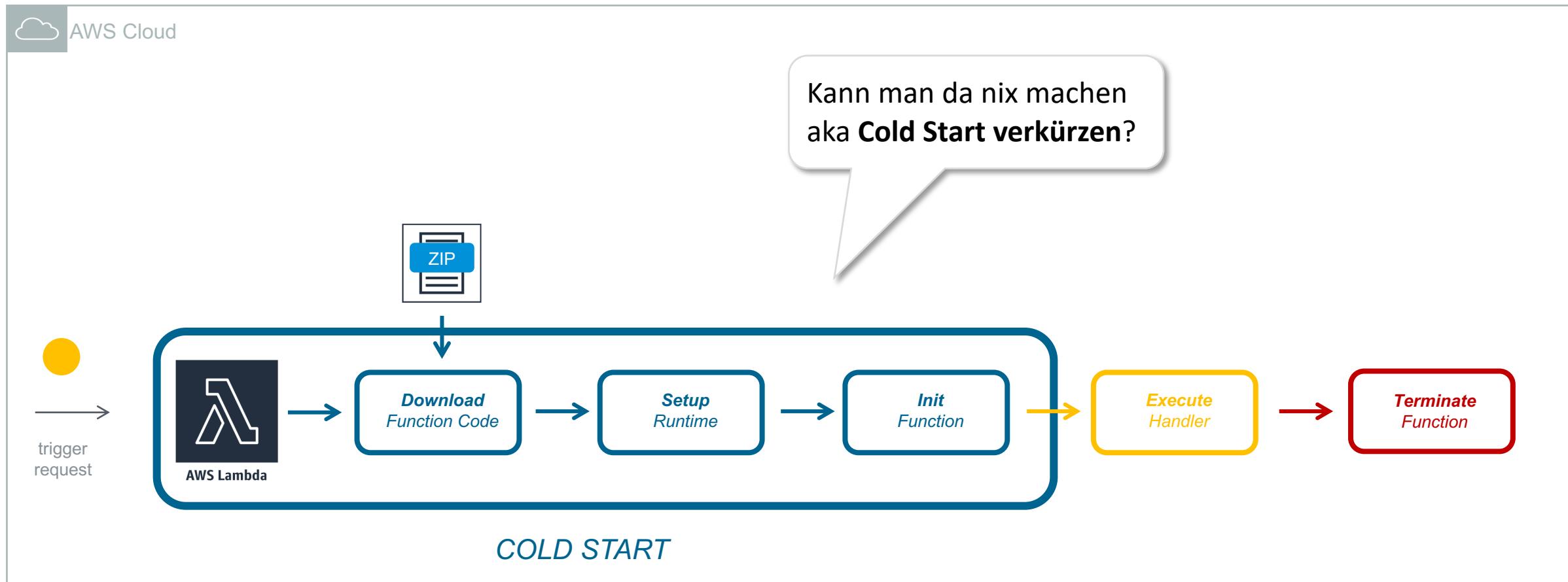
Hello Serverless **REALLY** „under the Hood“





Spaß mit
Cold Start

Spaß mit Cold Start



Spaß mit Cold Start: SnapStart

The screenshot shows the AWS Lambda console interface. At the top, there's a navigation bar with tabs: 'Funktionsübersicht' (selected), 'Auslöser hinzufügen', 'Ziel hinzufügen', 'Code', 'Test', 'Überwachen', and 'Konfiguration'. Below this, a card displays the function name 'sw-hello-serverless-world:1', its description 'v0.1', and the last update time 'vor 6 Minuten'. A button '+ Auslöser hinzufügen' is visible on the left, and '+ Ziel hinzufügen' is on the right. The 'Konfiguration' tab is selected, showing two panels: 'Allgemeine Konfiguration' and 'Allgemeine Konfiguration'. The 'Allgemeine Konfiguration' panel contains fields for 'Beschreibung' (v0.1), 'Timeout' (0 Min. 15 Sek.), 'Arbeitsspeicher' (512 MB), and 'Flüchtiger Speicher' (512 MB). The 'SnapStart-Optimierungsstatus' field is highlighted with a yellow box and set to 'On'. A callout bubble points from this highlighted area to the right, containing the text: 'Lambda SnapStart ist eine Leistungsoptimierung, mit der die Startlatenz ohne zusätzliche Kosten reduziert werden kann. BINGO! Oder?'.

▼ Funktionsübersicht [Info](#)

sw-hello-serverless-world:1

Layers (0)

+ Auslöser hinzufügen + Ziel hinzufügen

Beschreibung
v0.1

Letzte Änderung
vor 6 Minuten

Code Test Überwachen Konfiguration

Allgemeine Konfiguration

Auslöser

Berechtigungen

Ziele

Funktion-URL

Allgemeine Konfiguration [Info](#)

Beschreibung
v0.1

Timeout
0 Min. 15 Sek.

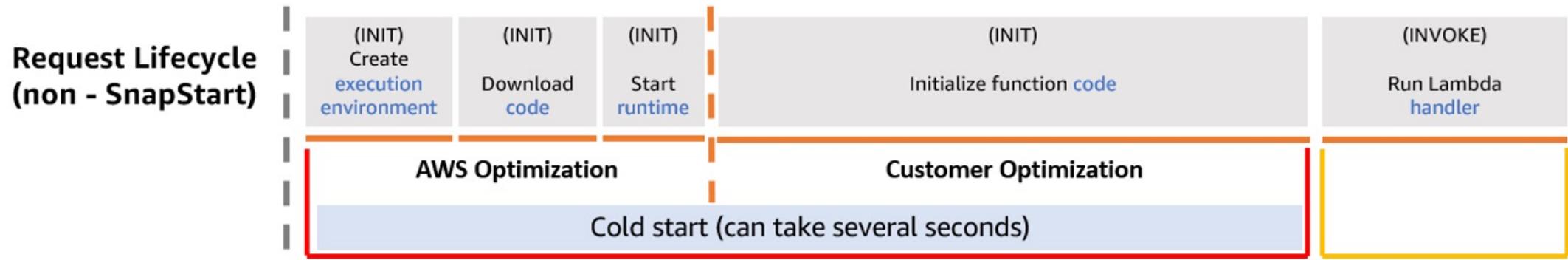
Arbeitsspeicher
512 MB

Flüchtiger Speicher
512 MB

SnapStart-Optimierungsstatus [Info](#)
On

Lambda SnapStart ist eine Leistungsoptimierung, mit der die Startlatenz ohne zusätzliche Kosten reduziert werden kann. BINGO! Oder?

Spaß mit Cold Start: SnapStart



<https://aws.amazon.com/de/blogs/compute/startng-up-faster-with-aws-lambda-snapstart/>

Spaß mit Cold Start: SnapStart

Request Lifecycle (non - SnapStart)

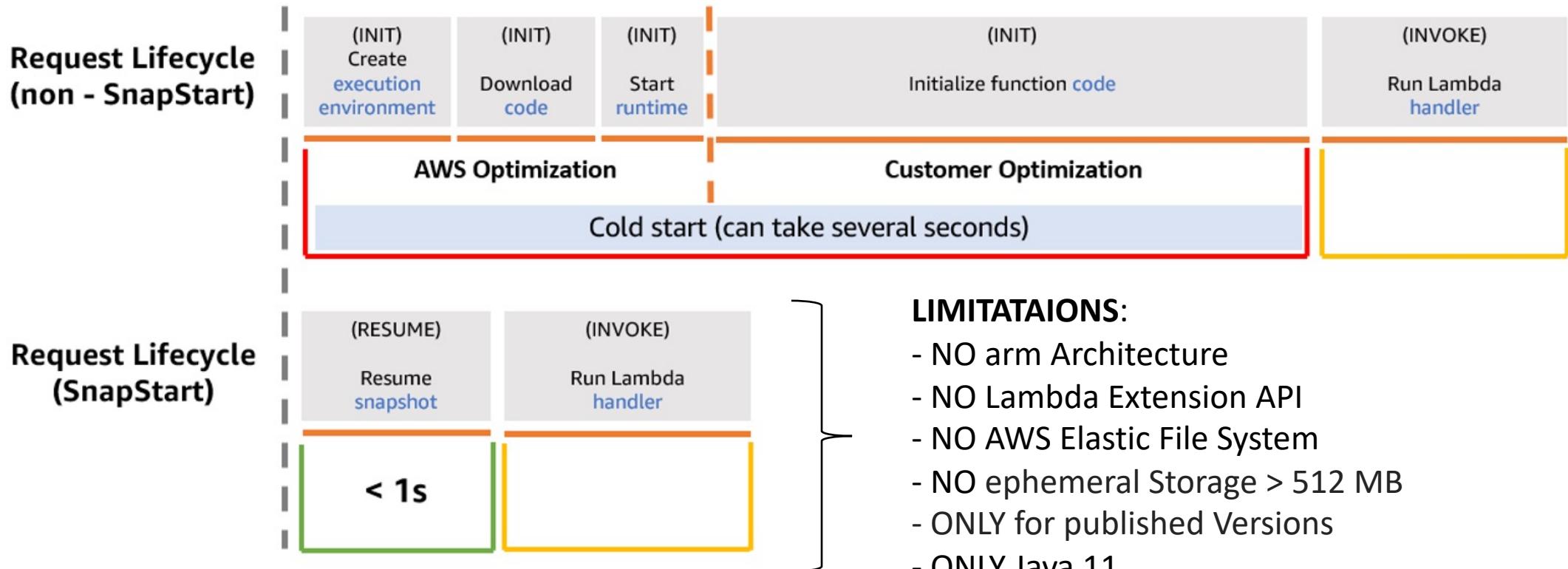
```
REPORT RequestId: 4f0e385d-1083-4d0c-a0e1-3c749169b600 Duration: 493.25
ms Billed Duration: 494 ms Memory Size: 512 MB Max Memory Used: 160 MB
Init Duration: 511.52 ms XRAY TraceId: 1-638fa4d3-
52b791875abf058d12ca200f SegmentId: 5c4362c041371b98 Sampled: true
```

Request Lifecycle (SnapStart)

```
REPORT RequestId: d1c283c7-cf6c-4ff6-a141-484ac460ae90 Duration: 470.25
ms Billed Duration: 803 ms Memory Size: 512 MB Max Memory Used: 124 MB
Restore Duration: 469.23 ms
```

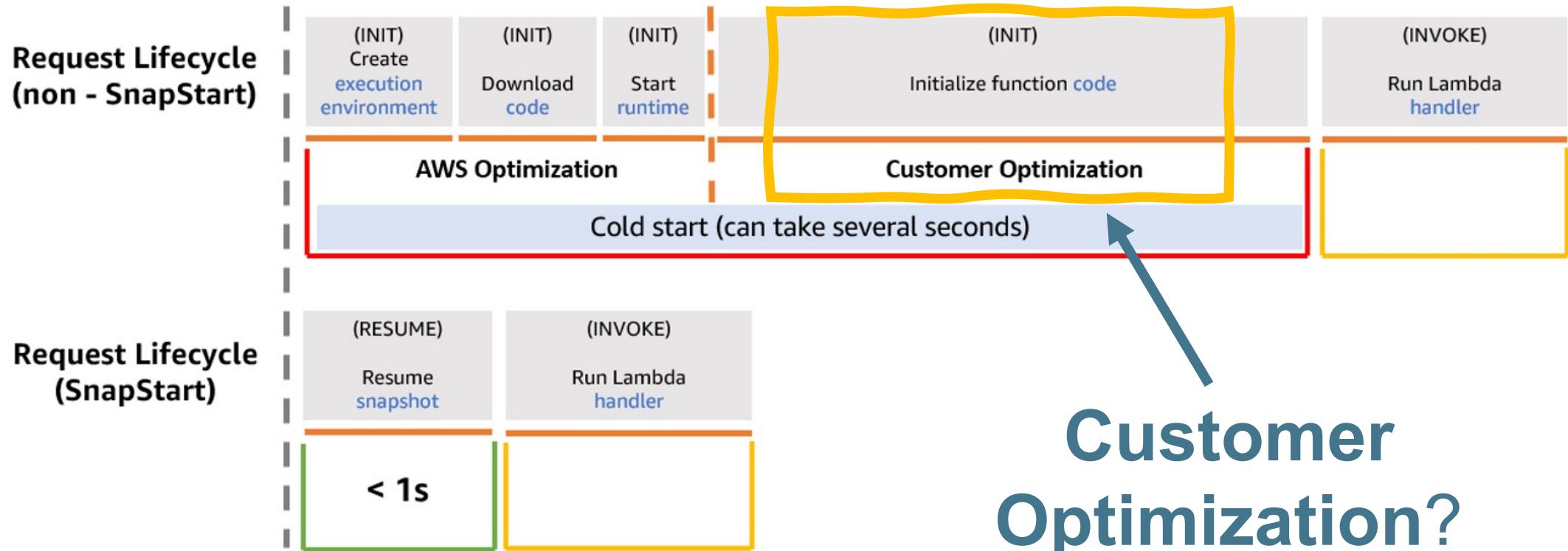
<https://aws.amazon.com/de/blogs/compute/startng-up-faster-with-aws-lambda-snapstart/>

Spaß mit Cold Start: SnapStart



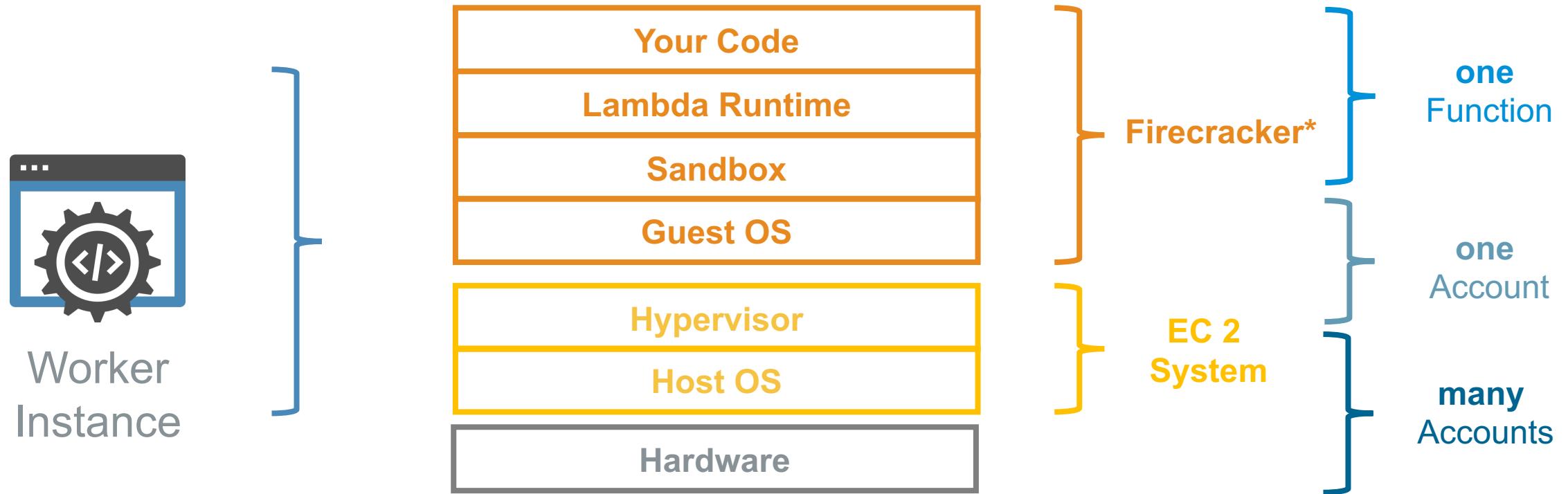
<https://aws.amazon.com/de/blogs/compute/startup-faster-with-aws-lambda-snapstart/>

Spaß mit Cold Start: SnapStart



<https://aws.amazon.com/de/blogs/compute/startng-up-faster-with-aws-lambda-snapstart/>

Spaß mit Cold Start: Initialization



Spaß mit Cold Start

Initialization vs. Handler



```
1 public class ReusableHandler implements RequestHandler<Input, Output> {
2
3     // dynamic code running each time
4     public Output handleRequest(Input input, Context context) {
5         Connection expensiveConnection = ConnectionFactory.getConnection(EXPENSIVE);
6
7         DataFetcher dataFetcher = expensiveConnection.initDataFetcher();
8         Data data = dataFetcher.getData();
9
10        return new Output(data);
11    }
12
13
14 }
```

Spaß mit Cold Start

Initialization vs. Handler

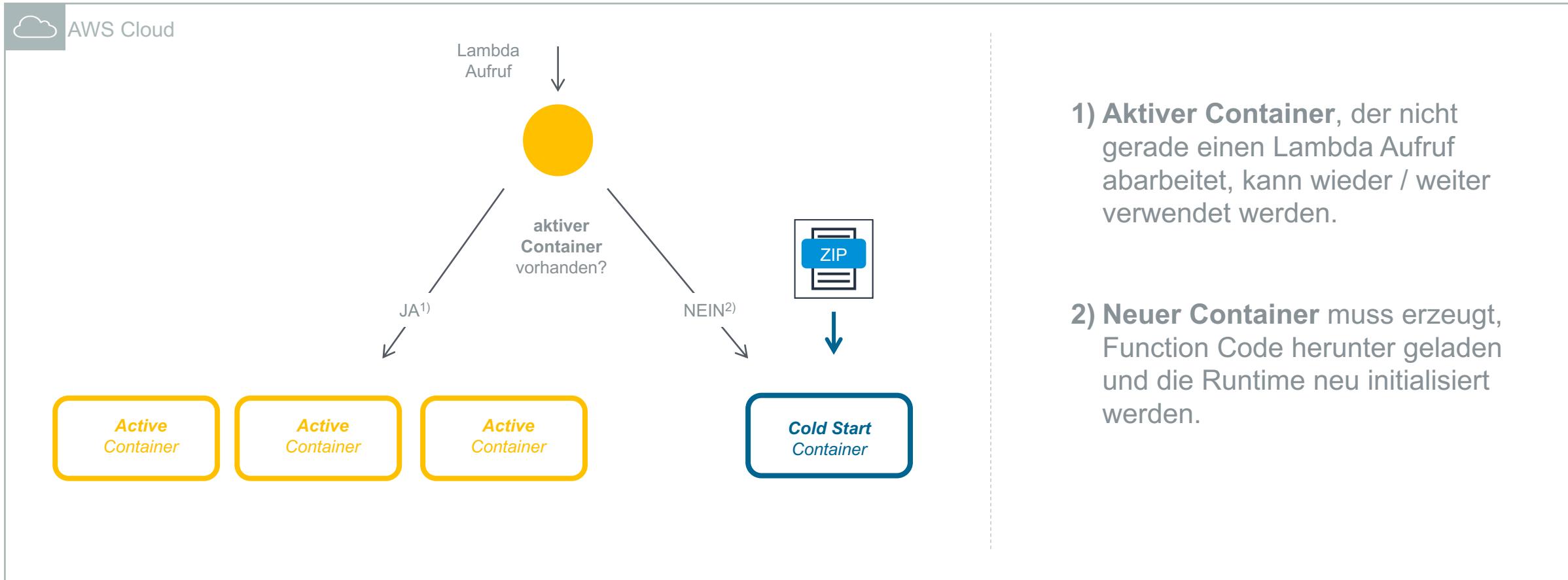


```
1 public class ReusableHandler implements RequestHandler<Input, Output> {
2
3     // static code running only once (per lambda container instance)
4     private static Connection EXPENSIVE_CONNECTION
5         = ConnectionFactory.getConnection(EXPENSIVE);
6
7     // dynamic code running each time
8     public Output handleRequest(Input input, Context context) {
9
10        DataFetcher dataFetcher = EXPENSIVE_CONNECTION.initDataFetcher();
11        Data data = dataFetcher.getData();
12
13        return new Output(data);
14    }
15
16 }
```

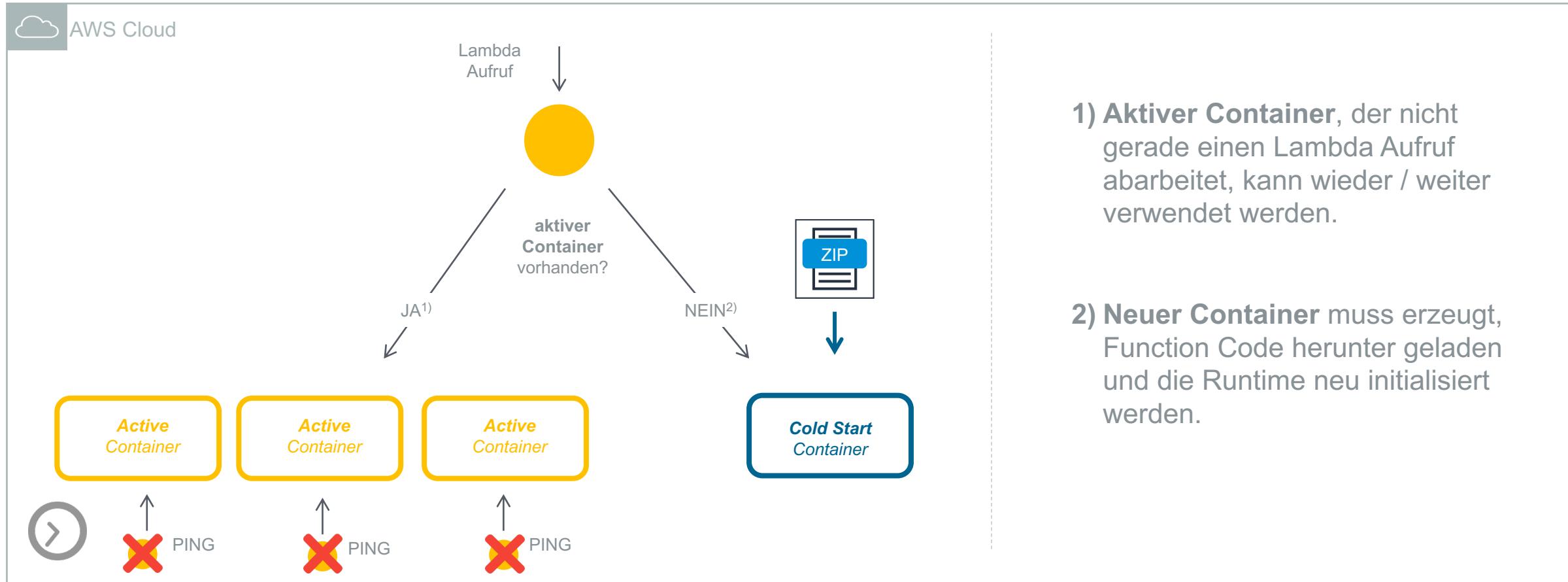


Spaß mit
Concurrency

Spaß mit Concurrency

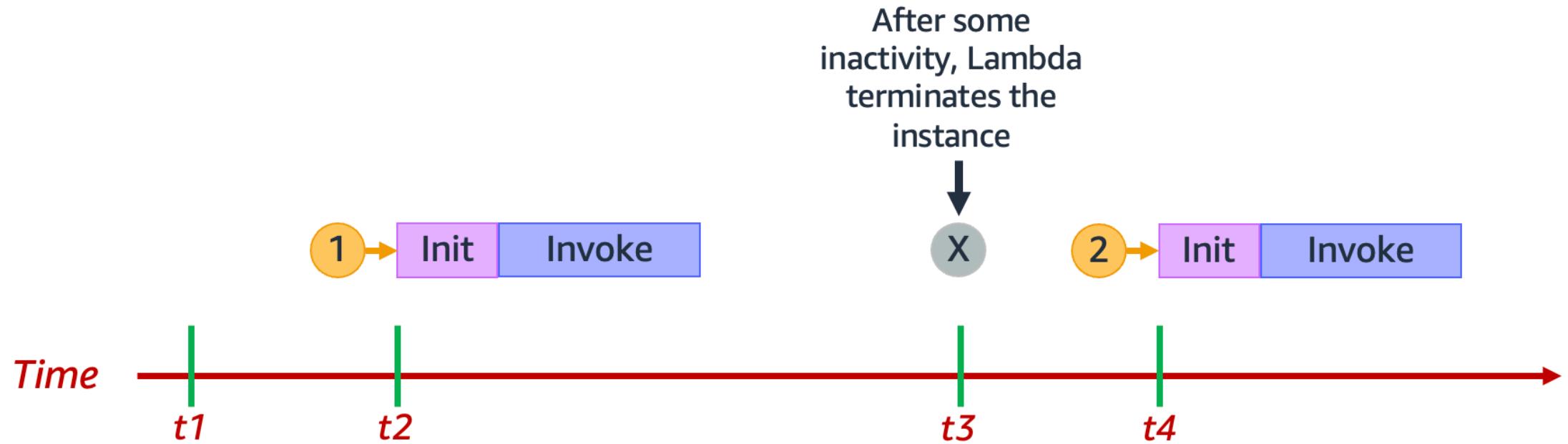


Spaß mit Concurrency



„Provisioned Concurrency“ aka \$\$\$, d.h. so viel wie nötig, so wenig wie möglich!

Spaß mit Concurrency



<https://docs.aws.amazon.com/lambda/latest/dg/lambda-concurrency.html>

Spaß mit Concurrency

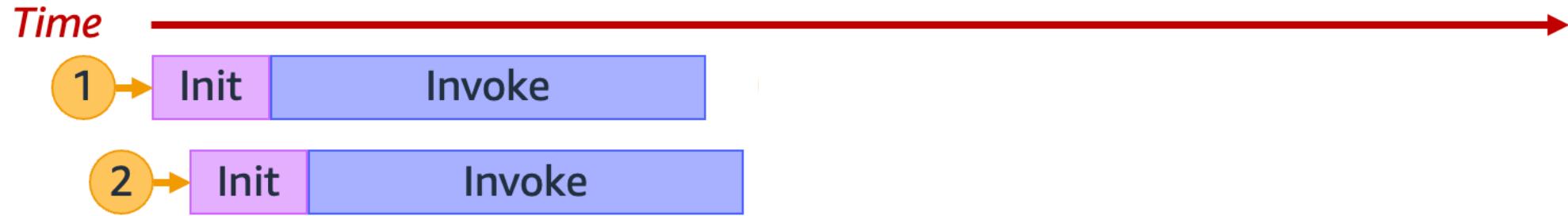
Theorie vs Praxis



<https://docs.aws.amazon.com/lambda/latest/dg/lambda-concurrency.html>

Spaß mit Concurrency

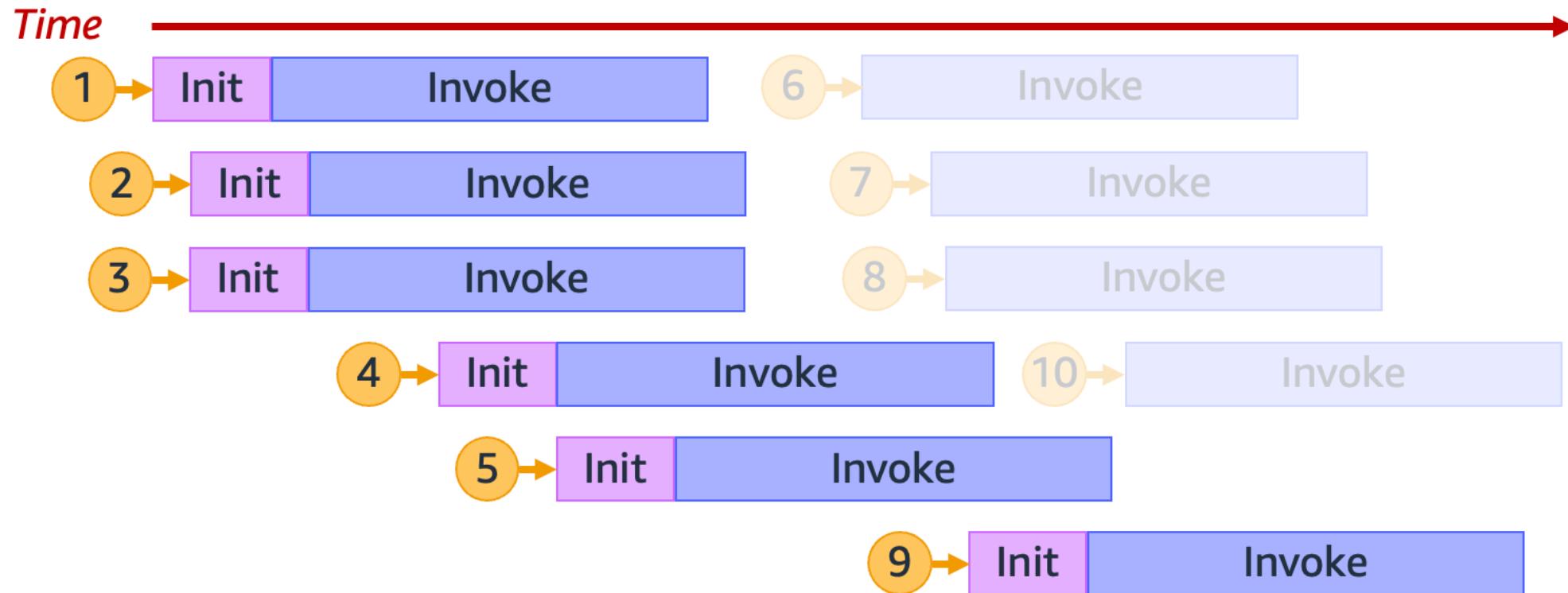
Theorie vs Praxis



<https://docs.aws.amazon.com/lambda/latest/dg/lambda-concurrency.html>

Spaß mit Concurrency

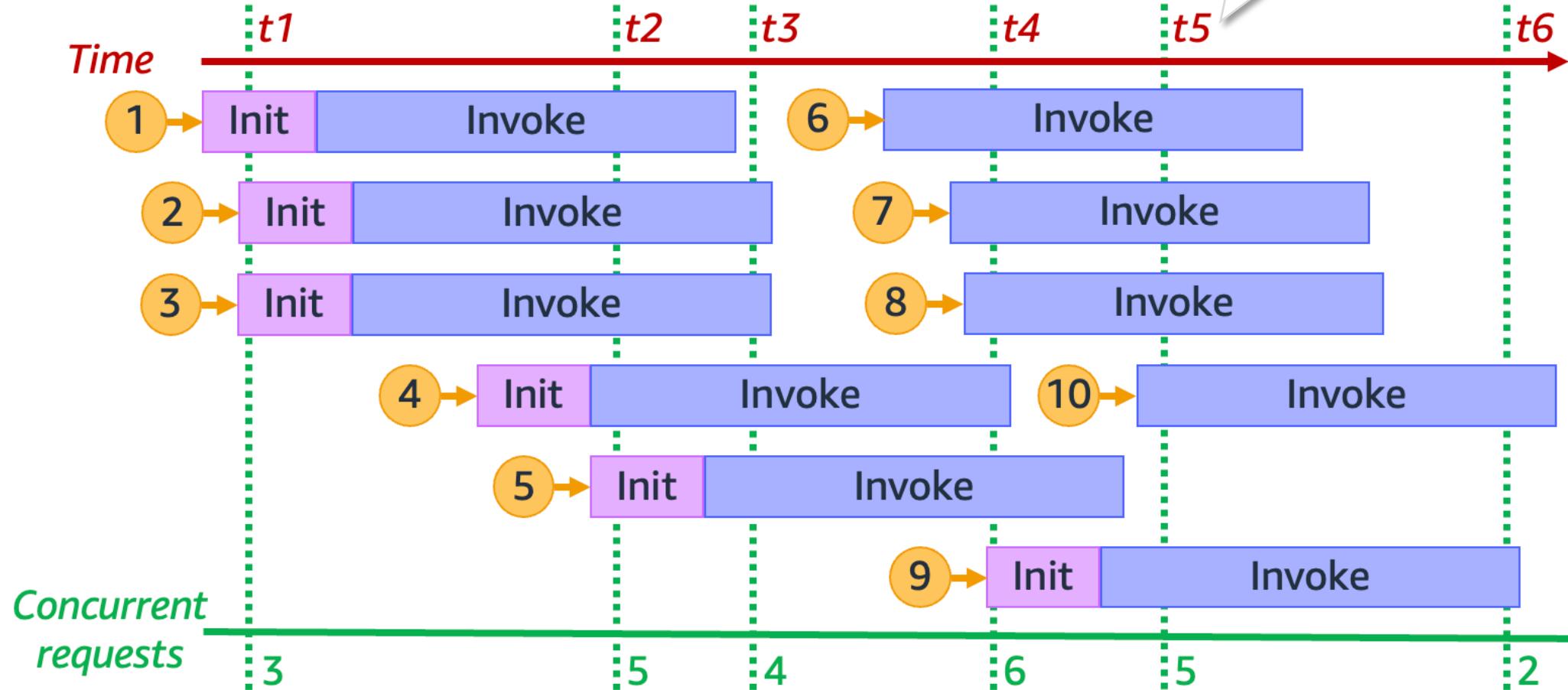
Theorie vs Praxis



<https://docs.aws.amazon.com/lambda/latest/dg/lambda-concurrency.html>

Spaß mit Concurrency

Theorie vs Praxis



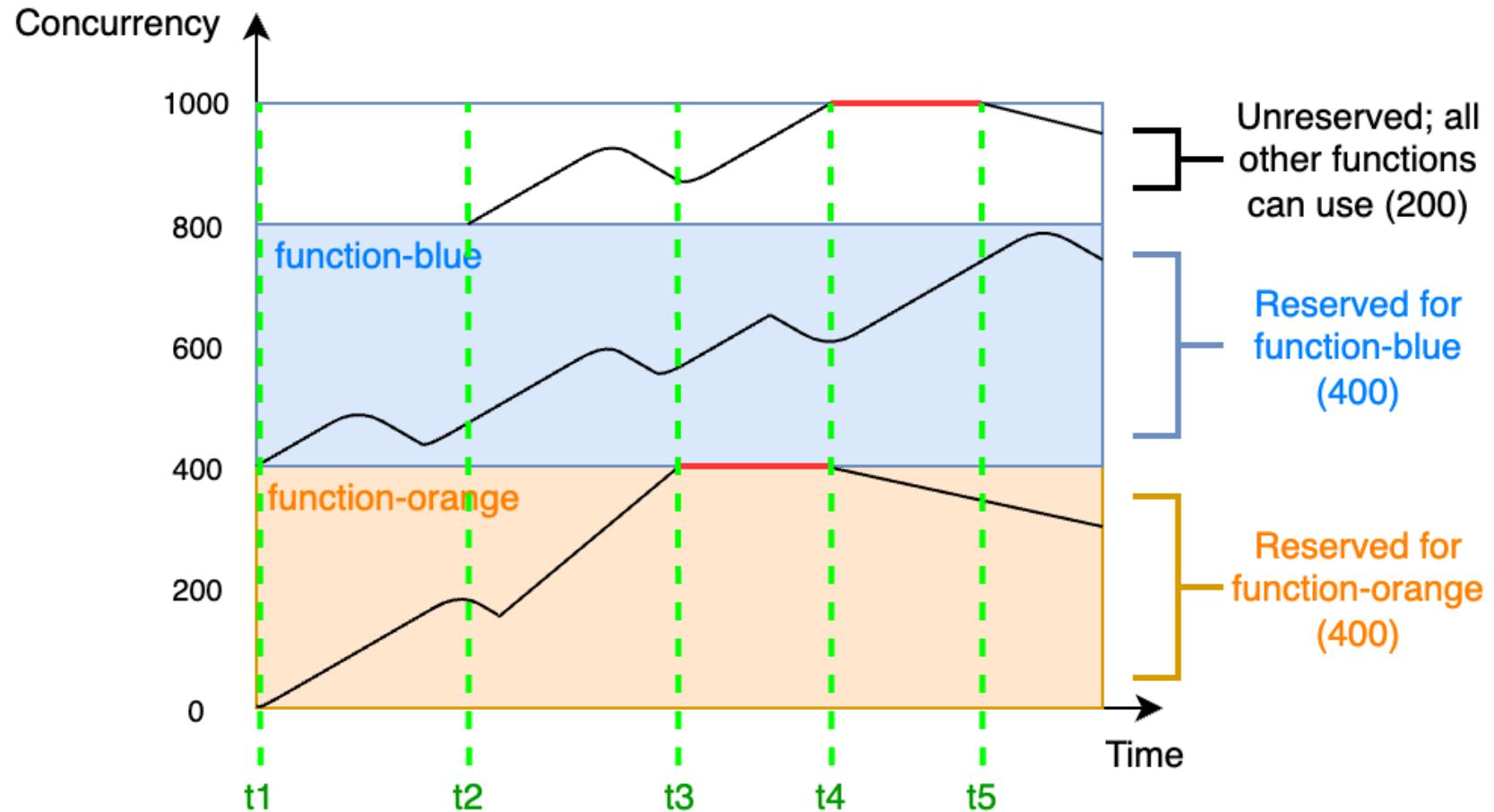
<https://docs.aws.amazon.com/lambda/latest/dg/lambda-concurrency.html>

Spaß mit Concurrency

Concurrency = (**average** req per **sec**)
 * (**average** req **duration** in sec)
 + some buffer

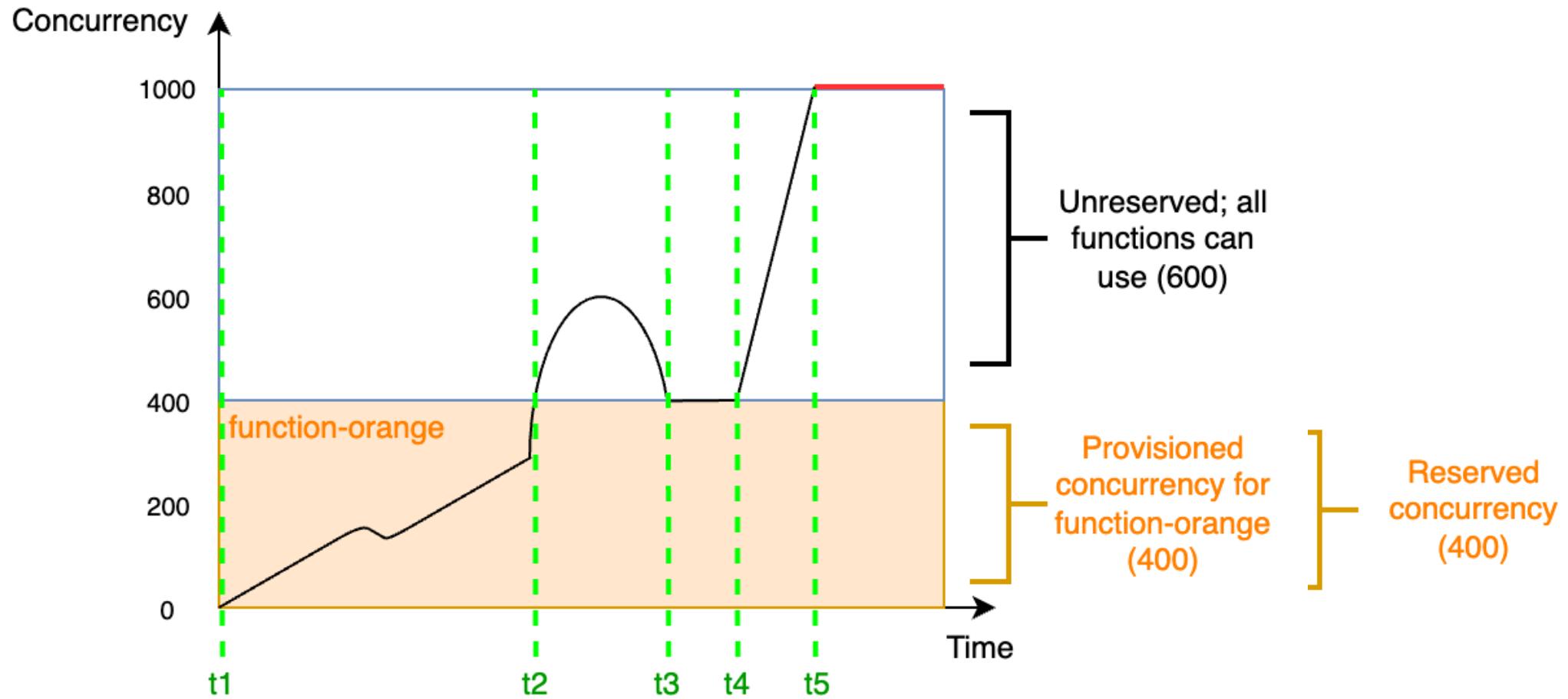
(max 1000 concurrent invocations: <https://docs.aws.amazon.com/lambda/latest/dg/gettingstarted-limits.html>)

Spaß mit Concurrency



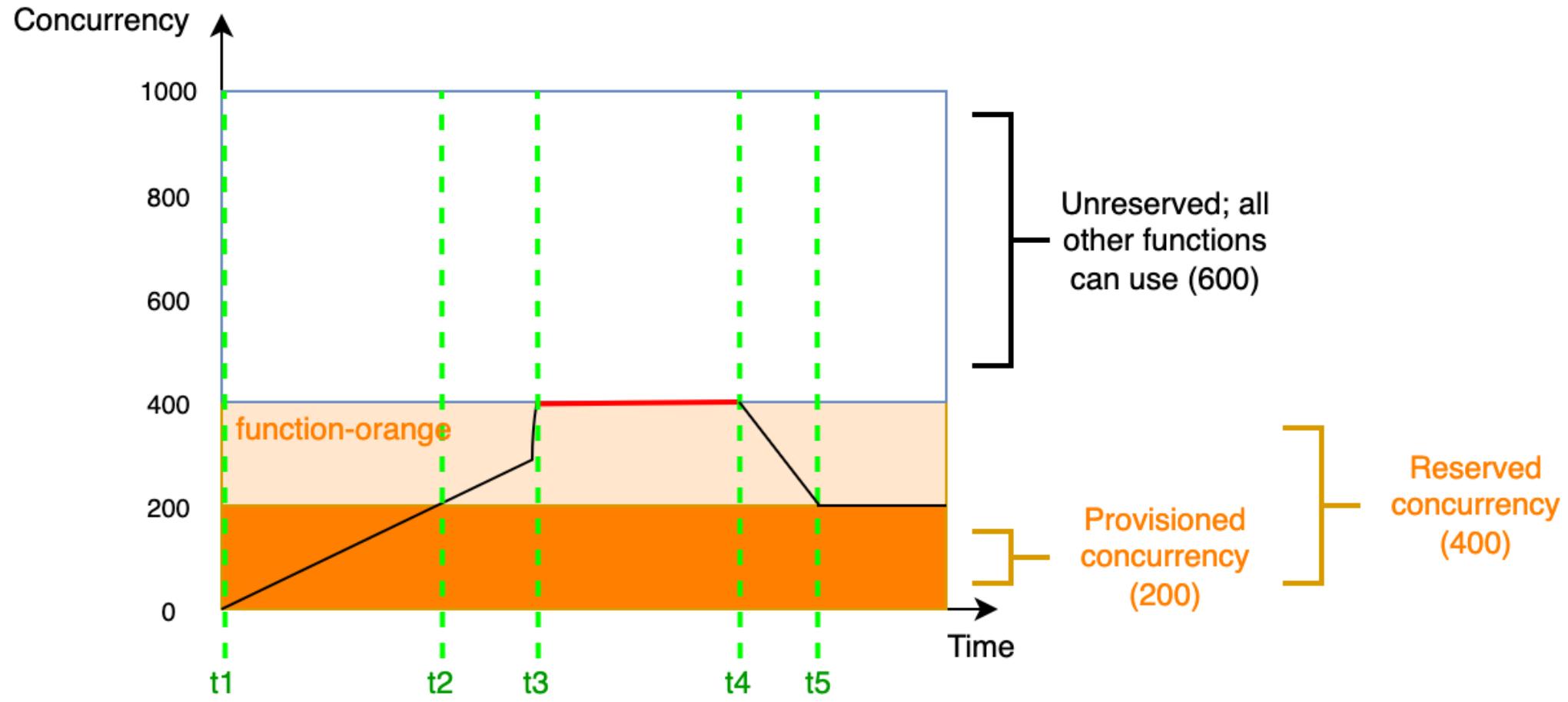
Provisioned vs. Reserved Concurrency

Spaß mit Concurrency



Provisioned vs. Reserved Concurrency

Spaß mit Concurrency

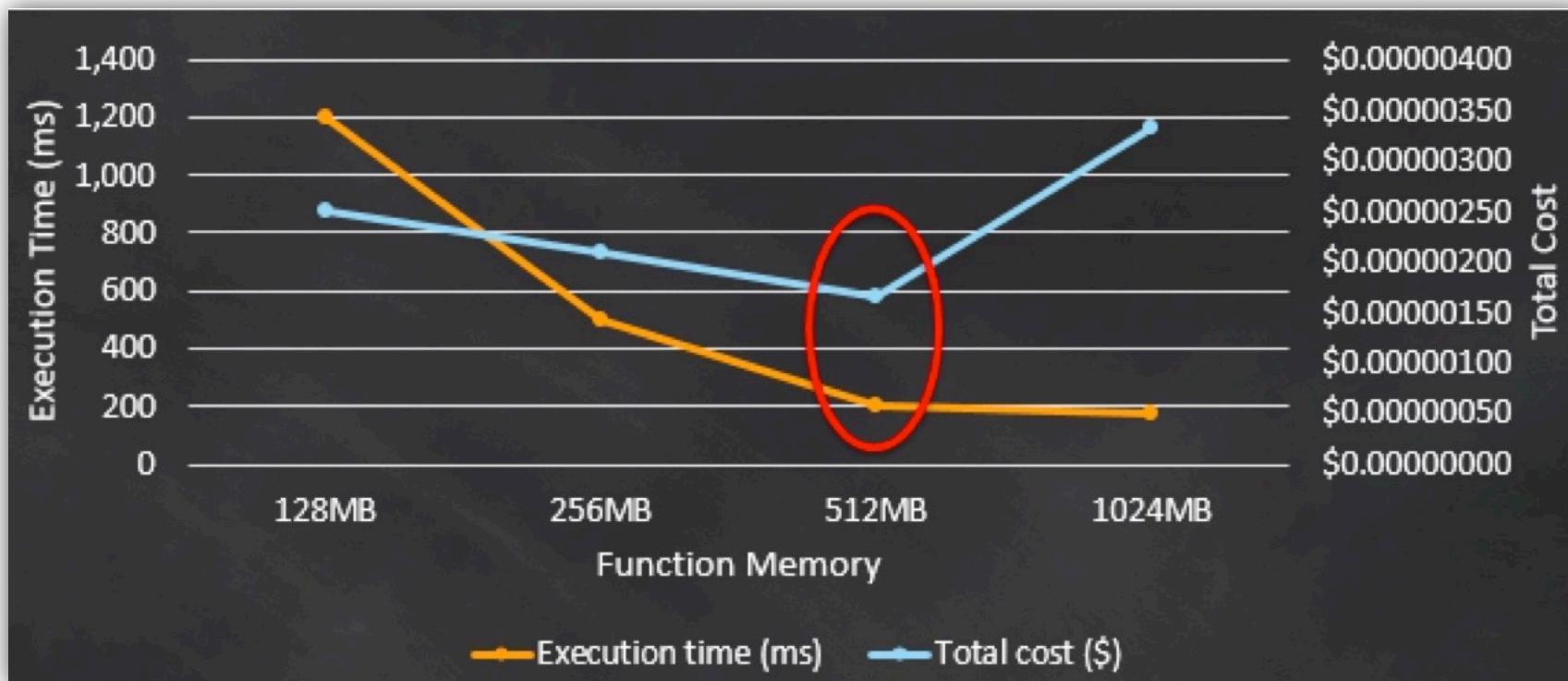


Provisioned vs. Reserved Concurrency

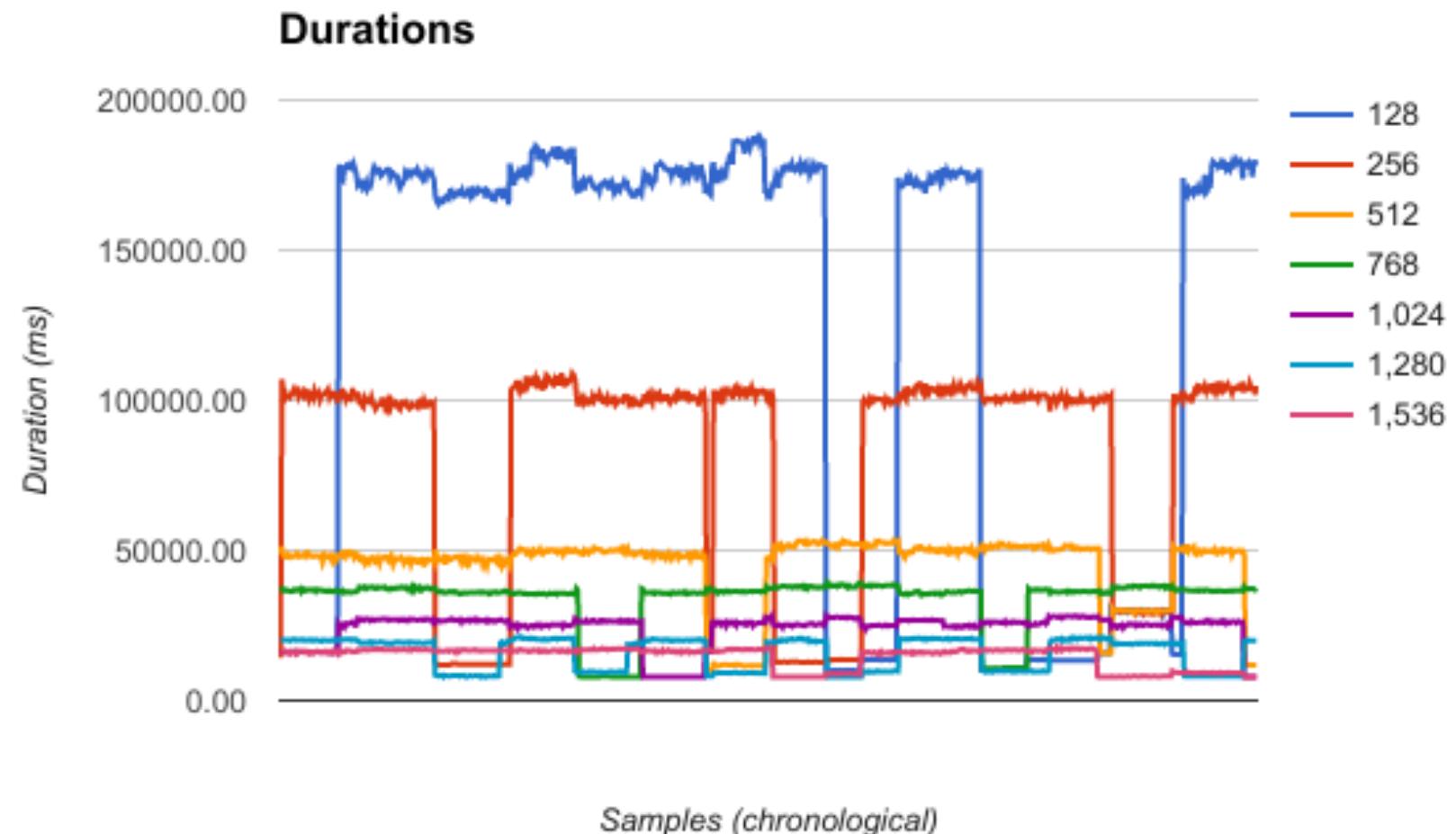


Spaß mit
Konfiguration

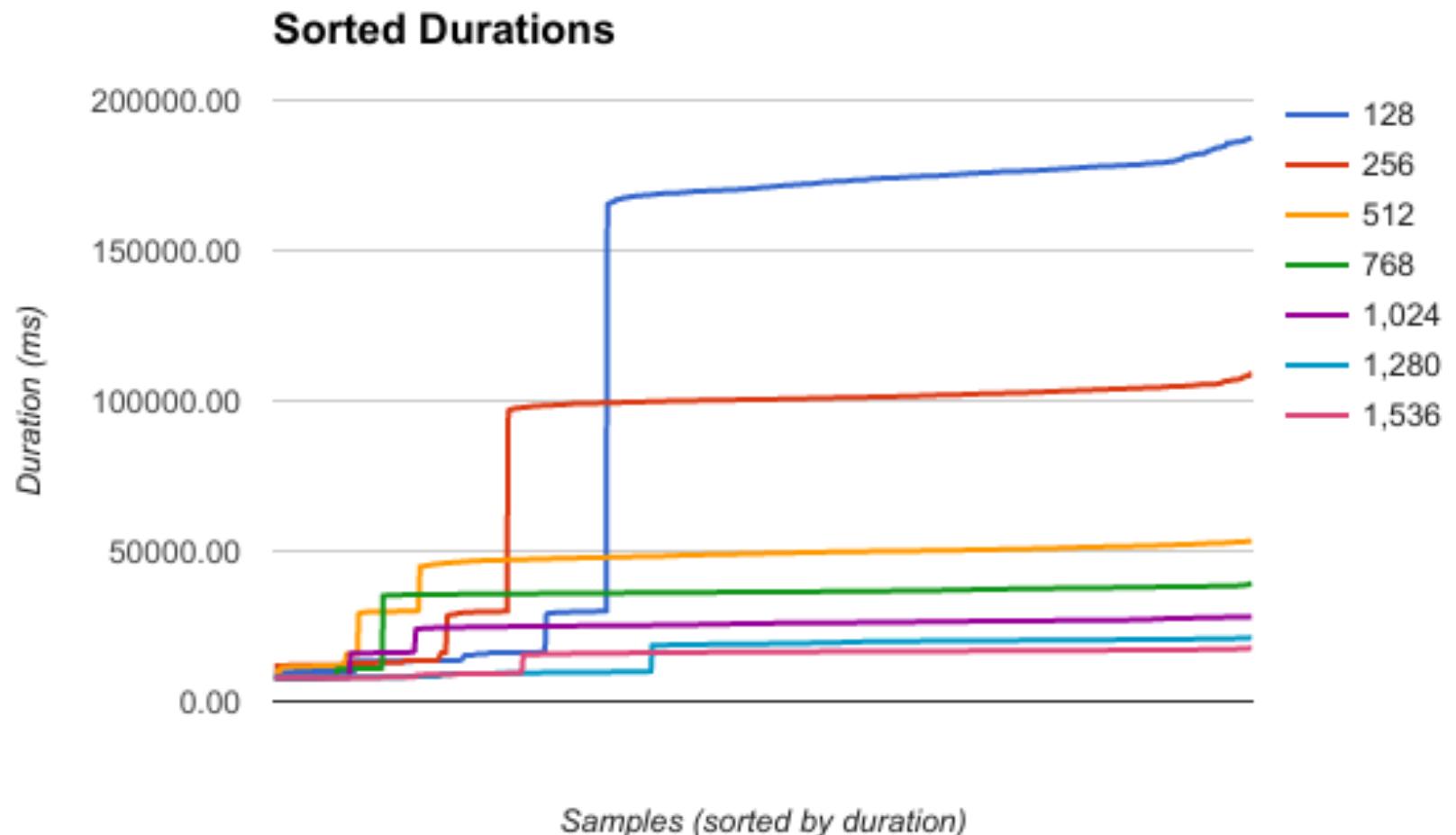
Spaß mit Konfiguration



Spaß mit Konfiguration



Spaß mit Konfiguration



HANDS-ON

Hello Serverless Coffee V2

- Implementation
- Upload
- Testing



SnapStart in Action

The screenshot shows the AWS Lambda function configuration page for the function 'sw-hello-serverless-world'. The function has no triggers or targets currently assigned.

Funktionsübersicht (Function Overview) details:

- Icon: sw-hello-serverless-world
- Layers: (0)
- + Auslöser hinzufügen (+ Add trigger)
- + Ziel hinzufügen (+ Add target)

Beschreibung (Description): -

Letzte Änderung (Last update): vor 1 Minute (about 1 minute ago)

Funktions-ARN (Function ARN): arn:aws:lambda:eu-central-1:460357271599:function:sw-hello-serverless-world

Funktion-URL (Function URL): -

Konfiguration (Configuration) tab is selected. The 'Allgemeine Konfiguration' (General Configuration) section is shown, with the 'Arbeitsspeicher' (Working memory) and 'Flüchtiger Speicher' (Transient memory) fields both set to 512 MB. The 'SnapStart' field is set to None.

CloudShell and **Feedback** buttons are at the bottom left. Copyright notice: © 2023, Amazon Web Services, Inc. oder Tochterfirmen. Datenschutz, Bedingungen, Cookie-Einstellungen links at the bottom right.

SnapStart in Action

The screenshot shows the AWS Lambda function configuration page for 'sw-hello-serverless-world' version 2. The 'Konfiguration' tab is selected. A yellow box highlights the 'Allgemeine Konfiguration' section, specifically the memory settings and the 'SnapStart-Optimierungsstatus' field.

Allgemeine Konfiguration

- Auslöser
- Berechtigungen
- Ziele
- Funktion-URL
- Umgebungsvariablen
- VPC

Allgemeine Konfiguration

| Arbeitsspeicher | Flüchtiger Speicher |
|-----------------|---------------------|
| 512 MB | 512 MB |

SnapStart-Optimierungsstatus: **On**

Aliase

Provisioned Concurrency in Action

The screenshot shows the AWS Lambda configuration interface for a function named "sw-hello-serverless-world". The "Konfiguration" tab is selected. A yellow box highlights the "Nebenläufigkeit" (Concurrency) section. This section contains two fields: "Funktionsnebenläufigkeit" (Function concurrency) set to "Reservierte Nebenläufigkeit verwenden" (Use reserved concurrency) and "Reservierte Nebenläufigkeit" (Reserved concurrency) set to 100. Below this, the "Bereitgestellte Nebenläufigkeitskonfigurationen" (Allocated concurrency configurations) section is shown, which is currently empty.

Allgemeine Konfiguration

Auslöser

Berechtigungen

Ziele

Funktion-URL

Umgebungsvariablen

Tags

VPC

Überwachungs- und Betriebstools

Nebenläufigkeit

Asynchroner Aufruf

Code Signing

Nebenläufigkeit

Bearbeiten

Funktionsnebenläufigkeit

Reservierte Nebenläufigkeit verwenden

Reservierte Nebenläufigkeit

100

Bereitgestellte Nebenläufigkeitskonfigurationen

Bearbeiten

Entfernen

Hinzufügen

Konfiguration suchen

Keine Konfigurationen

Konfiguration hinzufügen

© 2023, Amazon Web Services, Inc. oder Tochterfirmen.

Datenschutz

Bedingungen

Cookie-Einstellungen

CloudShell

Feedback

Provisioned Concurrency in Action

The screenshot shows the AWS Lambda console interface for configuring provisioned concurrency. A yellow oval highlights the central configuration area. The URL in the browser is eu-central-1.console.aws.amazon.com/lambda/home?region=eu-central-1#/functions/sw-hello-serverless-world/configure/concurrent-execution.

Bereitgestellte Nebenläufigkeit konfigurieren

Bereitgestellte Nebenläufigkeit

Qualifier-Typ
Sie können die bereitgestellte Nebenläufigkeit für einen Alias oder eine Version konfigurieren.

Alias
 Version

Version
Stellen Sie Nebenläufigkeit für eine Version bereit.

1
Aliasse: -
Beschreibung: v0.1

Bereitgestellte Nebenläufigkeit
Damit Ihre Funktion ohne Schwankungen der Latenz skaliert werden kann, verwenden Sie die bereitgestellte Nebenläufigkeit. Mit Auto Scaling von Anwendungen können Sie die bereitgestellte Nebenläufigkeit automatisch anpassen, um eine konfigurierte Zielauslastung beizubehalten. Die bereitgestellte Nebenläufigkeit wird kontinuierlich ausgeführt und verfügt über separate Preise für Nebenläufigkeit und Ausführungszeit. [Weitere Informationen](#)

67,16 \$ pro Monat zusätzlich zu den Preisen für Dauer und Anforderungen. [Preise](#)

10
100 verfügbar

CloudShell Feedback © 2023, Amazon Web Services, Inc. oder Tochterfirmen. Datenschutz Bedingungen Cookie-Einstellungen

Lessons Learned:

Hello Serverless



HELLO SERVERLESS

Lessons Learned (so far)

- **Business-Code** vs Infrastruktur-Code
- **Least Privilege Execution Role**
- **Cold Start** beim ersten Call (langsam/teuer)*
- **CloudWatch** ist dein Freund
- **Konfiguration** ist kontextsensitiv

(*SnapStart ODER Provisioned Concurrency als mögliche Lösung)



Initial Set-up

SETUP – das WAS

Was gilt es vorab einzurichten?

- Tabellen
- Event-Bus
- Rollen & Rechte

SETUP – das WIE

Wie gilt es vorab einzurichten?

1. CloudFormation Template einspielen
2. Initiale Tabellenwerte setzen

HANDS-ON

Workshop Set-up

- CF Template einspielen
- Tabellen initialisieren
- Set-Up checken



serverless-workshop – template.yaml [infrastructure]

Project Bookmarks AWS Toolkit Structure

serverless-workshop > 00_setup > infrastructureAsCode > template.yaml

1 AWSTemplateFormatVersion: 2010-09-09
2 Transform: AWS::Serverless-2016-10-31
3 Description: serverless workshop - core stack.
4
5 Parameters:
6 AppName:
7 Type: String
8 Description: Application name (eg. serverless workshop)
9 Default: 'serverlessWorkshop'
10 Service:
11 Type: String
12 Description: Service name (eg. core)
13 Default: 'core'
14 LogRetentionInDays:
15 Type: Number
16 Default: 14
17 Description: CloudWatch Logs retention period
18 TimeInterval:
19 Type: Number

Document 1/1 > AWSTemplateFormatVersion:

Terminal: Local Local (2) +
(base) lars@MacBook-Pro-von-Lars-7 serverless-workshop % aws cloudformation deploy --template-file serverless-workshop.yaml --stack-name serverless-workshop --capabilities CAPABILITY_IAM

Version Control TODO Problems Terminal Services Profiler Build Dependencies

1:1 LF UTF-8 2 spaces Schema: schema.json AWS: 6 Connections

serverless-workshop – template.yaml [infrastructure]

Project Bookmarks AWS Toolkit Structure

serverless-workshop > 00_setup > infrastructureAsCode > template.yaml

1 AWSTemplateFormatVersion: 2010-09-09
2 Transform: AWS::Serverless-2016-10-31
3 Description: serverless workshop - core stack.
4
5 Parameters:
6 AppName:
7 Type: String
8 Description: Application name (eg. serverless workshop)
9 Default: 'serverlessWorkshop'
10 Service:
11 Type: String
12 Description: Service name (eg. core)
13 Default: 'core'
14 LogRetentionInDays:
15 Type: Number
16 Default: 14
17 Description: CloudWatch Logs retention period
18 TimeInterval:
19 Type: Number

Document 1/1 > AWSTemplateFormatVersion:

Terminal: Local Local (2) +
(base) lars@MacBook-Pro-von-Lars-7 serverless-workshop % aws cloudformation deploy --template-file serverless-workshop.yaml --stack-name serverless-workshop --capabilities CAPABILITY_IAM

Version Control TODO Problems Terminal Services Profiler Build Dependencies

1:1 LF UTF-8 2 spaces Schema: schema.json AWS: 6 Connections

CloudFormation Template: serverless-workshop.yaml

CloudFormation – Stack

eu-central-1.console.aws.amazon.com/cloudformation/home?region=eu-central-1#/stacks?filteringText=&filteringStatus=active&viewNested=true

CloudFormation Services Suche [Option+S]

CloudFormation > Stacks

Stacks (1)

Nach Stack-Namen filtern

Aktiv

Verschachtelte anzeigen

Stack-Name Status Erstellungszeit Beschreibung

ok-serverless-workshop CREATE_COMPLETE 2023-06-12 12:25:21 UTC+0100 serverless workshop - core stack.

StackSets Exporte

Designer

Registry

Öffentliche Erweiterungen Aktivierte Erweiterungen Herausgeber

Spotlight Neu

Feedback

CloudShell Feedback Sprache

© 2023, Amazon Web Services, Inc. oder Tochterfirmen. Datenschutz Bedingungen Cookie-Einstellungen

The screenshot shows the AWS CloudFormation service interface. On the left, there's a sidebar with navigation links for 'Stacks', 'StackSets', 'Exports', 'Designer', 'Registry' (with sub-links for 'Öffentliche Erweiterungen', 'Aktivierte Erweiterungen', and 'Herausgeber'), 'Spotlight' (marked as 'Neu'), and 'Feedback'. The main content area is titled 'CloudFormation > Stacks' and shows a table for 'Stacks (1)'. The table has columns for 'Stack-Name', 'Status', 'Erstellungszeit', and 'Beschreibung'. A single row is listed: 'ok-serverless-workshop' with status 'CREATE_COMPLETE', created on '2023-06-12 12:25:21 UTC+0100', and the description 'serverless workshop - core stack.'. Above the table are filters for 'Nach Stack-Namen filtern' and 'Aktiv', and a button to 'Verschachtelte anzeigen'. At the bottom of the page, there are links for 'CloudShell', 'Feedback', 'Sprache', and a footer with copyright information and links for 'Datenschutz', 'Bedingungen', and 'Cookie-Einstellungen'.

CloudFormation – Stack ok-ser

eu-central-1.console.aws.amazon.com/cloudformation/home?region=eu-central-1#/stacks/resources?filteringText=&filteringStatus=active&viewNeste...

Services Suche [Option+S]

CloudFormation X

CloudFormation > Stacks > ok-serverless-workshop

Stacks (1)

Nach Stack-Namen filtern

Aktiv ▾ Verschachtelte anzeigen

Stacks

ok-serverless-workshop

2023-06-12 12:25:21 UTC+0100

CREATE_COMPLETE

ok-serverless-workshop

Löschen Aktualisieren Stack-Aktionen ▾ Stack erstellen ▾

Stack-Info Ereignisse Ressourcen Ausgaben Parameter Vorlage Änderungssätze

Ressourcen (8)

Ressourcen suchen

| Logische ID | Physische ID | Typ | Status | Modul |
|----------------------------|--|-----------------------|-----------------|-------|
| ConfigTable | sw-config-table | AWS::DynamoDB::Table | CREATE_COMPLETE | - |
| CoreEventBusARNParameter | /serverlessWorkshop/core/eventbusarn | AWS::SSM::Parameter | CREATE_COMPLETE | - |
| CoreEventBusNameParameter | /serverlessWorkshop/core/eventbusname | AWS::SSM::Parameter | CREATE_COMPLETE | - |
| CountingTable | sw-order-counter-table | AWS::DynamoDB::Table | CREATE_COMPLETE | - |
| OrderTable | sw-order-table | AWS::DynamoDB::Table | CREATE_COMPLETE | - |
| ServerlessWorkshopEventBus | sw-event-bus | AWS::Events::EventBus | CREATE_COMPLETE | - |
| swOrderManagementRole | ok-serverless-workshop-swOrderManagementRole-e-1SRM28WI3X7L4 | AWS::IAM::Role | CREATE_COMPLETE | - |

CloudShell Feedback Sprache

© 2023, Amazon Web Services, Inc. oder Tochterfirmen.

Datenschutz Bedingungen Cookie-Einstellungen

The screenshot shows the AWS CloudFormation service interface. On the left, the navigation pane includes 'Stacks', 'StackSets', 'Exports', 'Designer', 'Registry' (with 'Öffentliche Erweiterungen' and 'Aktivierte Erweiterungen'), 'Herausgeber', 'Spotlight', and 'Feedback'. The main content area displays a stack named 'ok-serverless-workshop' with one resource, 'ConfigTable', which is in 'CREATE_COMPLETE' status. The 'Ressourcen' tab is selected, showing a table of resources with columns for Logische ID, Physische ID, Typ, Status, and Modul. Other resources listed include CoreEventBusARNParameter, CoreEventBusNameParameter, CountingTable, OrderTable, ServerlessWorkshopEventBus, and swOrderManagementRole.

CloudFormation Service: serverless-workshop.yaml

The screenshot shows the AWS DynamoDB console interface. On the left, a sidebar menu includes options like Dashboard, Tabellen (selected), PartiQL-Editor, Backups, Exporte nach S3, Importe aus S3, Reservierte Kapazität, Einstellungen, and DAX. The main area displays a table titled "Tabellen (3) Info" with three rows:

| <input type="checkbox"/> | Name | Status | Partitionsschlüssel | Sortierschlüssel | Indi... | Löschschutz | Lesekapazitätsmod... | Schreibkapazitätsmodus |
|--------------------------|------------------------|--------|---------------------|------------------|---------|-------------|----------------------|------------------------|
| <input type="checkbox"/> | sw-config-table | Aktiv | PK (S) | - | 0 | Aus | On-Demand | On-Demand |
| <input type="checkbox"/> | sw-order-counter-table | Aktiv | PK (S) | - | 0 | Aus | On-Demand | On-Demand |
| <input type="checkbox"/> | sw-order-table | Aktiv | PK (S) | - | 0 | Aus | On-Demand | On-Demand |

At the bottom of the page, there are links for CloudShell, Feedback, Sprache, © 2023, Amazon Web Services, Inc. oder Tochterfirmen., Datenschutz, Bedingungen, and Cookie-Einstellungen.

DynamoDB: Tabellenübersicht

The screenshot shows the AWS DynamoDB console interface. The left sidebar contains navigation links for Dashboard, Tabellen, PartiQL-Editor, Backups, Exporte nach S3, Importe aus S3, Reservierte Kapazität, Einstellungen, DAX (Cluster, Subnetzgruppen, Parametergruppen, Ereignisse), CloudShell, Feedback, and Sprache. The main content area is titled "DynamoDB > Elemente erkunden > sw-config-table". A sub-menu titled "Tabellen (3)" is open, showing three tables: "sw-config-table" (selected), "sw-order-counter-table", and "sw-order-table". The main panel is titled "SW-config-table" and contains a section "Elemente scannen oder abfragen" with a radio button selected for "Scannen". It also includes dropdowns for selecting the table ("Tabelle - sw-config-table") and attribute projection ("Alle Attribute"). Below this is a "Filter" section and a large orange "Ausführen" button. The bottom section is titled "Zurückgegebene Elemente (2)" and lists two items: "PK" with value "entries" and "open" (with a dropdown arrow); and "menu" with a JSON value: [{"M": {"available": {"BOOL": true}}, "i..."}]. There are "Aktionen" and "Element erstellen" buttons at the top of this list. The footer includes links for CloudShell, Feedback, Sprache, © 2023, Amazon Web Services, Inc. oder Tochterfirmen., Datenschutz, Bedingungen, and Cookie-Einstellungen.

DynamoDB: sw-config-table

The screenshot shows the Amazon DynamoDB Item Explorer interface for the 'sw-order-counter-table'. The left sidebar contains navigation links for Dashboard, Tabellen, PartiQL-Editor, Backups, Exporte nach S3, Importe aus S3, Reservierte Kapazität, Einstellungen, DAX, Cluster, Subnetzgruppen, Parametergruppen, and Ereignisse. The main area displays the 'Tabellen (3)' panel with three tables: 'sw-config-table', 'sw-order-counter-table' (selected), and 'sw-order-table'. Below this, the 'sw-order-counter-table' details are shown with the title 'sw-order-counter-table'. A section titled 'Elemente scannen oder abfragen' has 'Scannen' selected. It includes dropdowns for 'Tabelle - sw-order-counter-table' and 'Alle Attribute', and a 'Filter' button. At the bottom, there's a 'Zurückgegebene Elemente (1)' section showing one item with PK 'order-counter' and currentValue '10'. Navigation icons like CloudShell, Feedback, Sprache, and footer links for © 2023, Datenschutz, Bedingungen, and Cookie-Einstellungen are at the bottom.

DynamoDB: sw-order-counter-table

Screenshot of the AWS IAM Management Console showing the Roles page.

The left sidebar shows the navigation menu under "Identity and Access Management (IAM)".

- Dashboard
- Zugriffsverwaltung
 - Benutzergruppen
 - Benutzer
 - Rollen**
 - Richtlinien
 - Identitätsanbieter
 - Kontoeinstellungen
- Zugriffsberichte
 - Zugriffsanalysator
 - Archivregeln
 - Analysatoren
 - Einstellungen
 - Bericht zu Anmeldeinformationen
 - Organisationsaktivität
 - Service-Kontrollrichtlinien (SCPs)

The main content area displays the "Rollen (38) Informationen" section. A search bar shows "sw". The table lists three roles:

| <input type="checkbox"/> | Rollenname | Vertrauenswürdige Entitäten | Letzte ... |
|--------------------------|--|-----------------------------|------------|
| <input type="checkbox"/> | ok-serverless-workshop-swOrderManagementRole-1SRM28WI3X7L4 | AWS-Service: lambda | Gestern |
| <input type="checkbox"/> | ok-serverless-workshop-swOrderProcessorRole-8H54FOVLO8M9 | AWS-Service: states | Gestern |
| <input type="checkbox"/> | sw-lambda-dynamodb-execution | AWS-Service: lambda | Gestern |

The "Roles Anywhere" section provides information on how to authenticate AWS-fremden Workloads:

- Auf AWS über Ihre AWS-fremden Workloads zugreifen**: Betreiben Sie Ihre AWS-fremden Workloads mit derselben Authentifizierungs- und Autorisierungsstrategie, die Sie auch in AWS verwenden.
- X.509-Standard**: Verwenden Sie Ihre eigene vorhandene PKI-Infrastruktur oder verwenden Sie [AWS Certificate Manager Private Certificate Authority](#) zur Authentifizierung von Identitäten.
- Temporäre Anmeldeinformationen**: Verwenden Sie problemlos temporäre Anmeldeinformation und profitieren Sie von der erhöhten Sicherheit, die sie bieten.

Identity & Access Management: Serverless Workshop Rollen



Agenda

Hello World & Set-up



Order Management

Order Processing

API Gateway

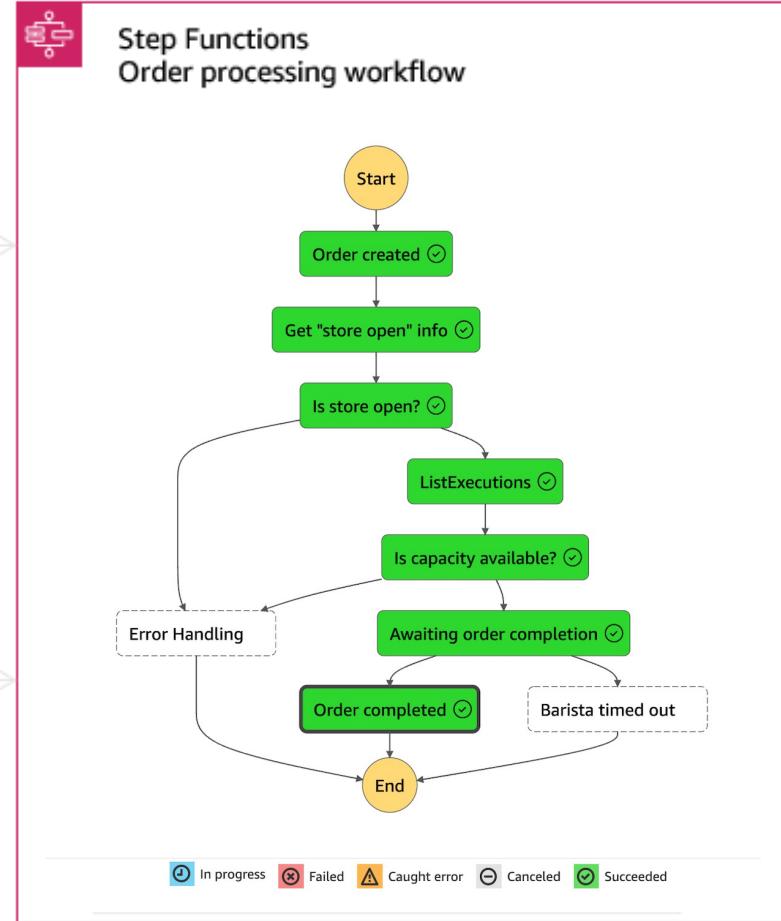
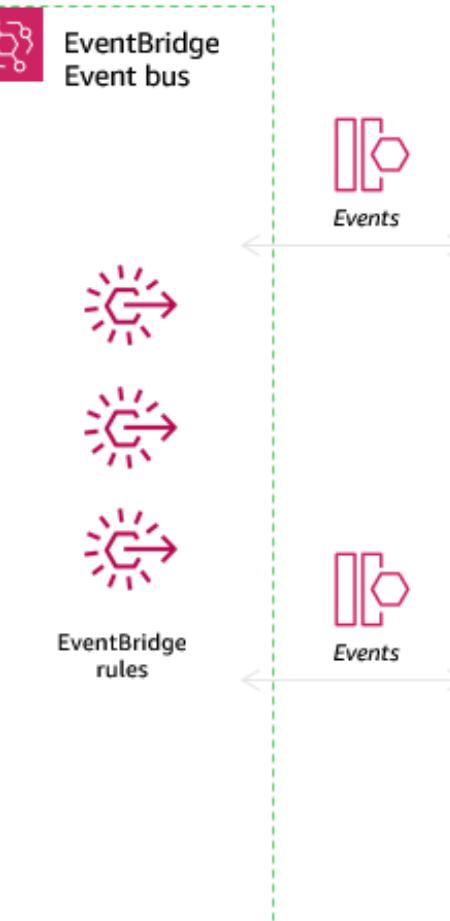
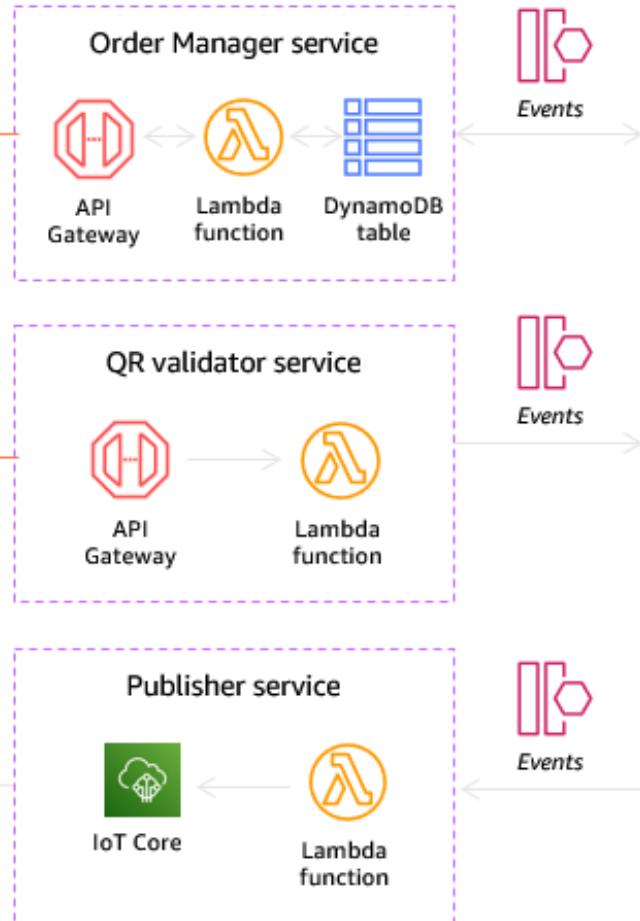
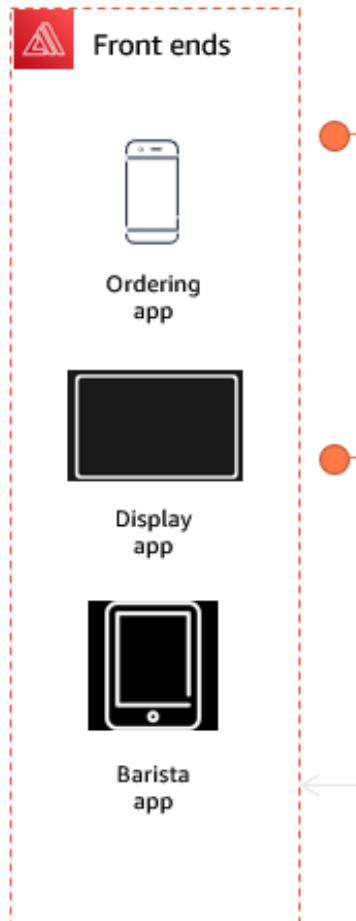
Testing



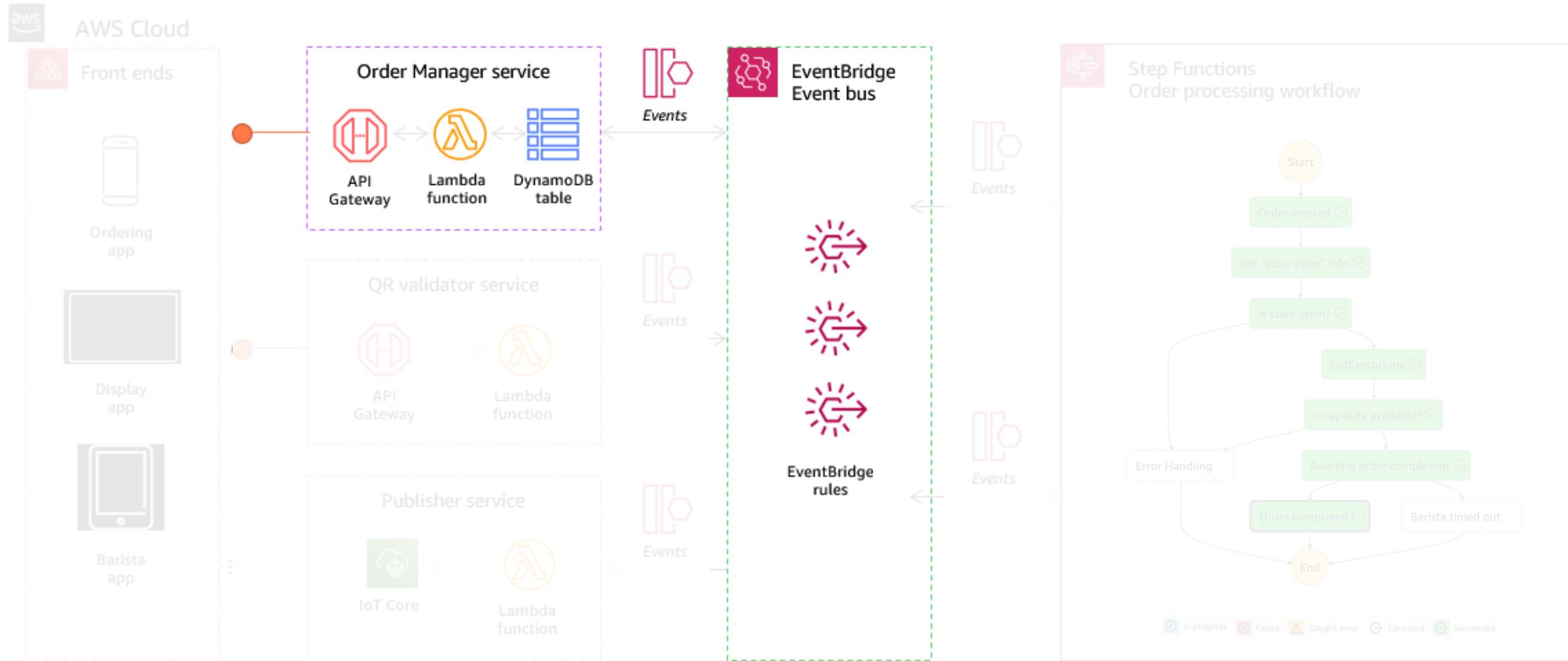
Serverless Coffee Order Management



AWS Cloud



⌚ In progress ⚡ Failed ⚠️ Caught error ⏹ Canceled ✅ Succeeded



HANDS-ON

Order Management V1

- Simple Order Handler
- dynamoDB Access



The screenshot shows an IDE interface with the following details:

- Project View:** Shows the project structure under "serverless-workshop". The "src/main/java/de/openknowledge/workshop/serverless/lambda" package contains the "OrderHandler.java" file.
- Code Editor:** The "OrderHandler.java" file is open, displaying Java code for an AWS Lambda function. The class implements `RequestHandler<OrderRequest, OrderResponse>`. The code includes imports for `java.util.logging.Logger` and `de.openknowledge.workshop.serverless.lambda.OrderRequest`, and defines constants for audit logging.
- AWS Toolkit:** A sidebar on the left provides AWS integration. It shows the region as "eu-central-1" and lists tables in the "DynamoDB" section, including "sw-config-table", "sw-order-counter-table", and "sw-order-table".
- Status Bar:** At the bottom, it shows the full path as "serverless-workshop > 02_order-manager > 01_simple-order-manager > src > main > java > de > openknowledge > workshop > serverless > lambda > OrderHandler", along with connection information: "AWS: 6 Connections 32:14 LF UTF-8 4 spaces".

Order Management Service V1

The screenshot shows the AWS Toolkit for IntelliJ IDEA interface. The left sidebar displays the project structure under the 'serverless-workshop' root, including subfolders like '.idea', '.mvn', '00_setup [infrastructure]', '01_hello_serverless [hello]', '02_order-manager [order]', '01_simple-order-manager', 'src', 'main', 'java', 'de', 'openknowledge', 'workshop', 'serverless', 'lambda', and 'OrderHandler.java'. The right side features the 'OrderHandler.java' code editor with the following content:

```
OrderHandler.java
```

```
package com.amazonaws.lambda.function.ordermanagement;

import com.amazonaws.services.dynamodbv2.model.*;
import com.amazonaws.services.dynamodbv2.document.*;

import java.util.List;
import java.util.Map;
import java.util.stream.Collectors;

import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.ScanRequest;
import software.amazon.awssdk.services.dynamodb.model.ScanResponse;

public class OrderHandler {
    private final DynamoDbClient dynamoDbClient = DynamoDbClient.create();

    public void handleEvent(Record record) {
        ScanRequest scanRequest = ScanRequest.builder().table("sw-order-table").build();
        ScanResponse scanResponse = dynamoDbClient.scan(scanRequest);
        List<Map<String, AttributeValue>> items = scanResponse.items();
        Map<String, String> confirmedOrders = items.stream()
            .filter(item -> item.get("state").asS().equals("confirmed"))
            .collect(Collectors.toMap(item -> item.get("orderNo").asS(), item -> item.get("userId").asS()));
        System.out.println("Confirmed orders: " + confirmedOrders);
    }
}
```

The central part of the interface shows the 'sw-order-table' in the AWS DynamoDB service, with a 'Scan' operation results table containing the following data:

| PK | drink | orderNo | state | userId |
|-----------------------|-----------------------|---------|-------------|------------------|
| "b942cac4-8717-4b8... | "Americano grande ... | 1 | "confirmed" | "lars.roewekamp" |
| "560bf13d-1b19-404... | "Americano grande" | 2 | "confirmed" | "lars.roewekamp" |

The bottom left corner of the interface shows the AWS Toolkit status bar with the following information:

- AWS: 6 Connections
- 32:14 LF
- UTF-8 4 spaces

Order Management Service V1

HANDS-ON

Order Management V2

- Advanced Order Handler
- CRUD Operations



The screenshot shows the IntelliJ IDEA interface with the project navigation bar at the top. The current project is 'serverless-workshop' and the active file is 'OrderHandler.java' located in the 'oms...' package. The code implements the `RequestHandler<OrderRequest, OrderResponse>` interface. The code handles an order request by extracting the action, delegating to a specific handler (CreateOrderHandler), and returning a response. A tooltip for the `OrderHandler` class is visible in the bottom right corner.

```
17
18 > import ...
19
20 /**
21 * Order Management Request Dispatcher:
22 * - extracts order action
23 * - delegates call to related order handler (create, read, update, delete)
24 * - collects and transforms result
25 */
26
27 // de.openknowledge.workshop.serverless.oms.lambda.OrderHandler::handleRequest
28 public class OrderHandler implements RequestHandler<OrderRequest, OrderResponse> {
29
30     private static final Logger logger = LoggerFactory.getLogger(OrderHandler.class);
31
32     @Override
33     public OrderResponse handleRequest(OrderRequest orderRequest, Context context) {
34
35         // check for valid order action
36         Order orderToHandle = orderRequest.getOrder();
37         OrderAction action = getOrderAction(orderRequest.getAction());
38
39         logger.info(String.format("Action = %s", action));
40         logger.info(String.format("Detail = %s", orderToHandle));
41
42         switch (action) {
43             case CREATE:
44                 CreateOrderRequest createOrderRequest = new CreateOrderRequest(orderRequest);
45                 CreateOrderHandler createOrderHandler = new CreateOrderHandler();
46                 return new OrderResponse(createOrderHandler.handleRequest(createOrderRequest, context));
47         }
48     }
49
50 }
```

Order Management Service V2

HANDS-ON

Order Management V3

- Enhanced Order Manager
- OR-Mapper Repository



The screenshot shows a Java code editor within an IDE. The left sidebar displays the project structure under 'serverless-workshop'. The 'src' folder contains 'main' and 'java' subfolders. In 'java', there is a package 'de.openknowledge.workshop.serverless.oms.model' which contains the 'Order.java' file currently being edited.

```
1 > /...
16 package de.openknowledge.workshop.serverless.oms.model;
17
18 > import ...
22
23 @DynamoDbBean
24 public class Order {
25
26     private String orderId;
27     private Long orderNo;
28     private String orderStatus;
29     private String userId;
30     private String baristaId;
31     private String drink;
32
33     public Order(String orderId, Long orderNo, String orderStatus, String userId, String baristaId, String drink) {
34         this.orderId = orderId;
35         this.orderNo = orderNo;
36         this.orderStatus = orderStatus;
37         this.userId = userId;
38         this.baristaId = baristaId;
39         this.drink = drink;
40     }
41
42     public Order() {
43         // default constructor for dynamoDB Schema creation
44     }
45
46     @DynamoDbPartitionKey
```

The status bar at the bottom shows the current path: 'serverless-workshop > 02_order-manager > 02_advanced-order-manager > src > main > java > de > openknowledge > workshop > serverless > oms > model > Order'. It also indicates 'AWS: 6 Connections' and the system status '24:14 LF UTF-8 4 spaces'.

Order Management Service V3

HANDS-ON

Order Management V4

- Single Responsibility
- Lambda Layer
- Domain Events



The screenshot shows an IDE interface with a project structure on the left and a code editor on the right. The project structure includes several modules: serverless-workshop, 00_setup [infrastructure], 01_hello_serverless [hello-serverless], 02_order-manager [order-manager] (with sub-modules 01_simple-order-manager [simple-order-manager] and 02_advanced-order-manager [advanced-order-manager]), and 03_layered-order-manager [layered-order-manager] (with sub-modules _distribution and oms-create-order). The oms-create-order module is currently selected. The code editor displays Java code for a CreateOrderHandler:

```
1 > /...
16 package de.openknowledge.workshop.serverless.oms.lambda;
17
18 > import ...
26
27 /**
28 * AWS lambda creating an order with the help of a given event
29 */
30 // de.openknowledge.workshop.serverless.oms.lambda.CreateOrderHandler::handleRequest
31 public class CreateOrderHandler implements RequestHandler<CreateOrderRequest, CreateOrderResponse> {
32
33     private static final Logger logger = LoggerFactory.getLogger(CreateOrderHandler.class);
34
35 /**
36 * Creates an order response with the help of a given <code>OrderRequest</code>
37 * representing the ordering user and the drink to create.
38 *
39 * @param createOrderRequest order event (userId, drink)
40 * @param context aws lambda context
41 * @return order response wrapping the created order
42 */
43
44     public CreateOrderResponse handleRequest(CreateOrderRequest createOrderRequest, Context context) {
45
46         CreateOrderResponse response;
47
48         if (createOrderRequest.isValid()) {
49
50             try {
51                 Order order = CreateOrderService.createOrder(createOrderRequest.getUserId(), createOrd...
```

The code editor shows syntax highlighting for Java and annotations. A tooltip for the `handleRequest` method is visible, providing information about its parameters and return type. The status bar at the bottom indicates the file path (.order-manager > 03_layered-order-manager > oms-create-order > src > main > java > de > openknowledge > workshop > serverless > oms > lambda > CreateOrderHandler), the number of connections (AWS: 6), and the file encoding (UTF-8).

Order Management Service V4

Lessons Learned:

Lambdas im Zusammenspiel



LAMBDAS IM ZUSAMMENSPIEL

Lessons Learned

- **Separation of Concerns**
- **OR-Mapping** via DynamoDB Annotations
- **Lambda Layer** für die Wiederverwendung
- **Domain Events** via EventBus zur Kommunikation
- **Hobby-Testing** via AWS CLI



Agenda

Hello World & Set-up



Order Management



Order Processing

API Gateway

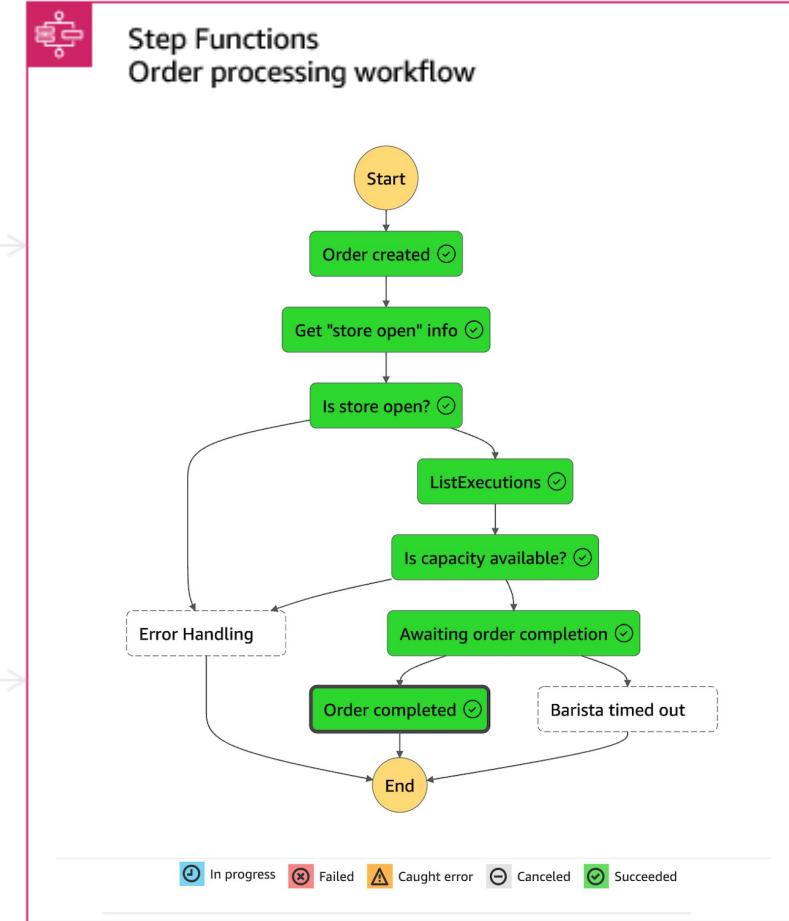
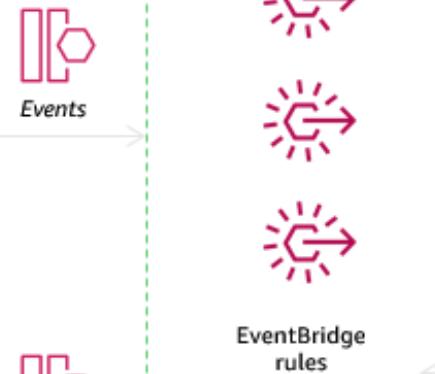
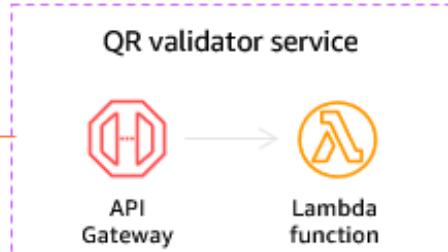
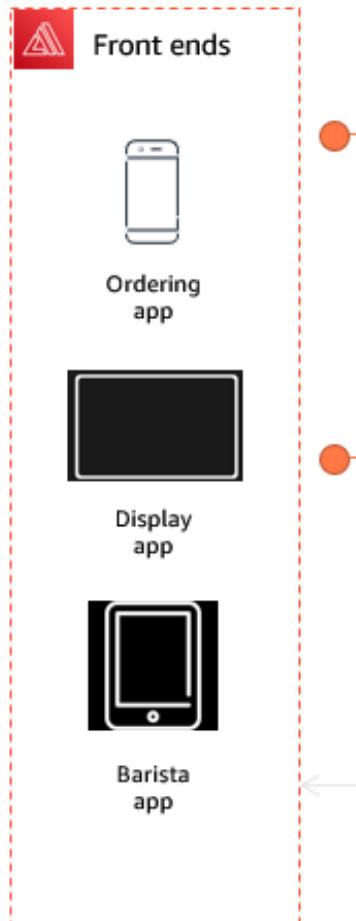
Testing

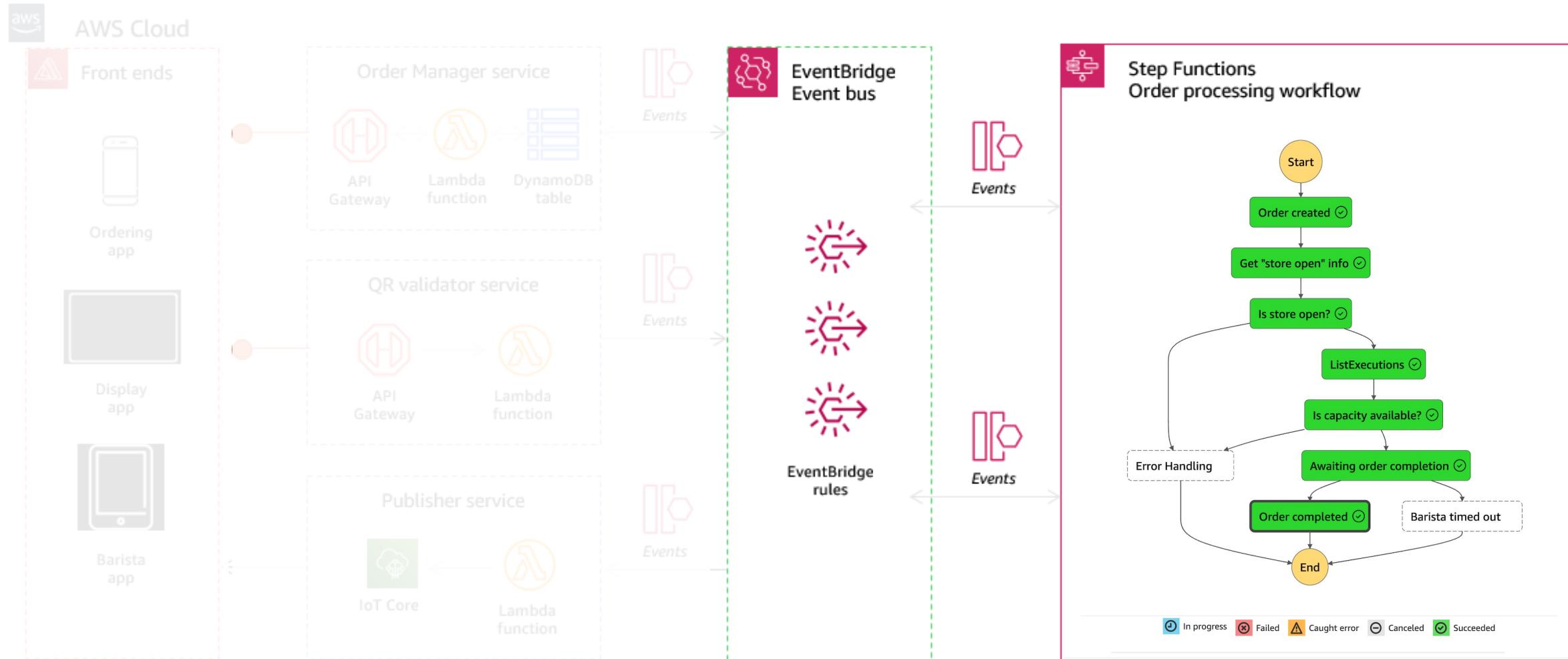


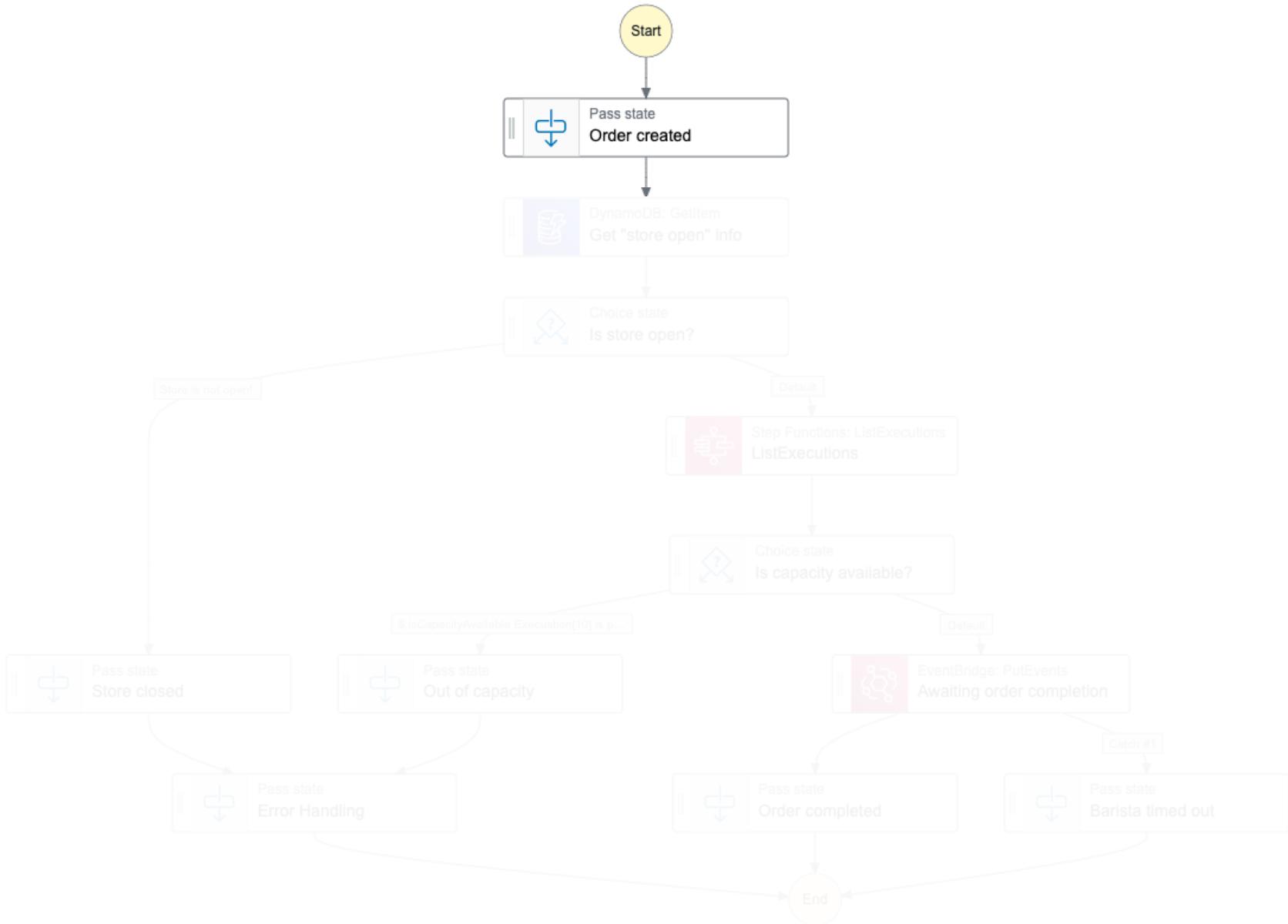
Serverless Coffee Order Processing



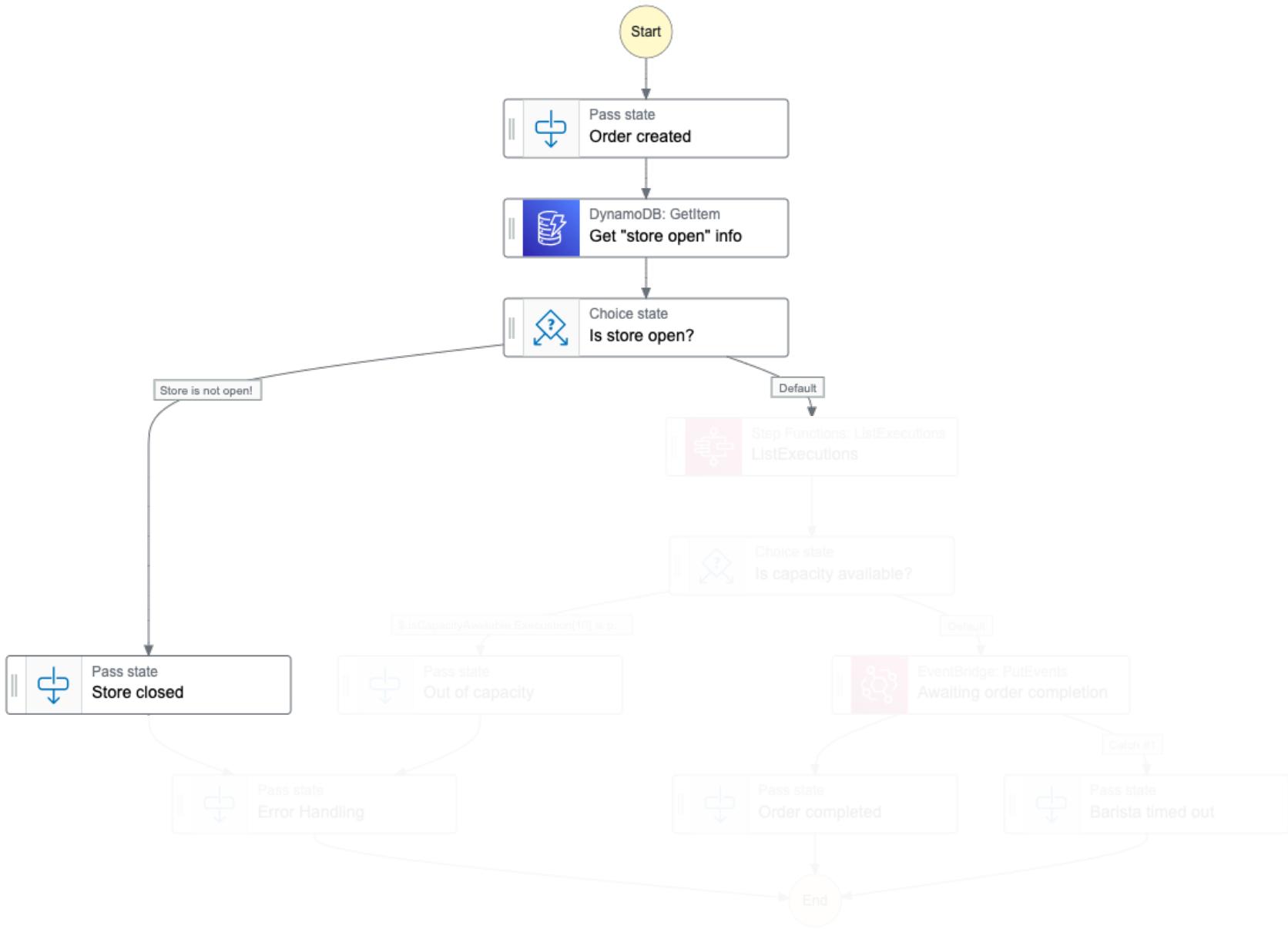
AWS Cloud



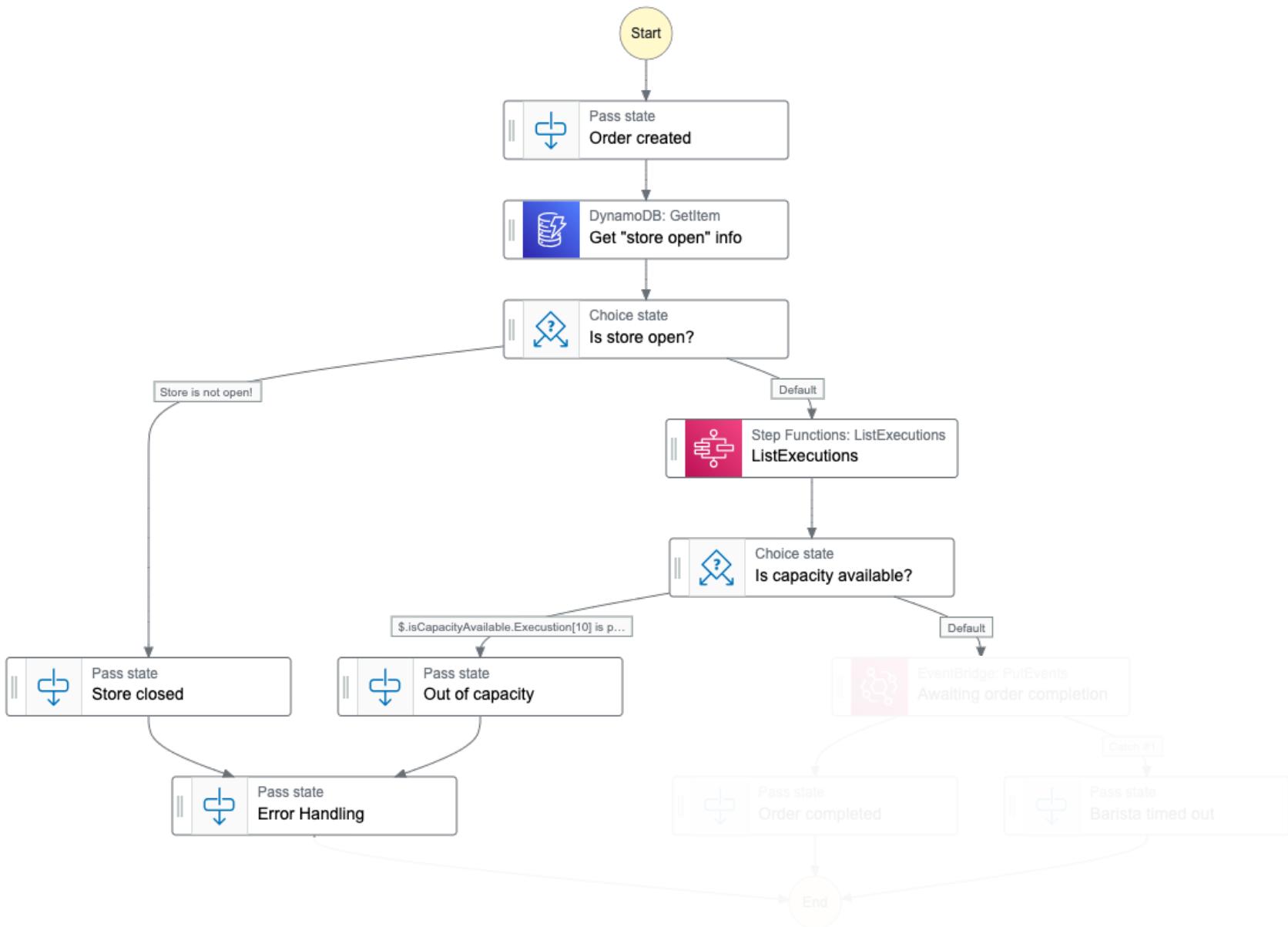




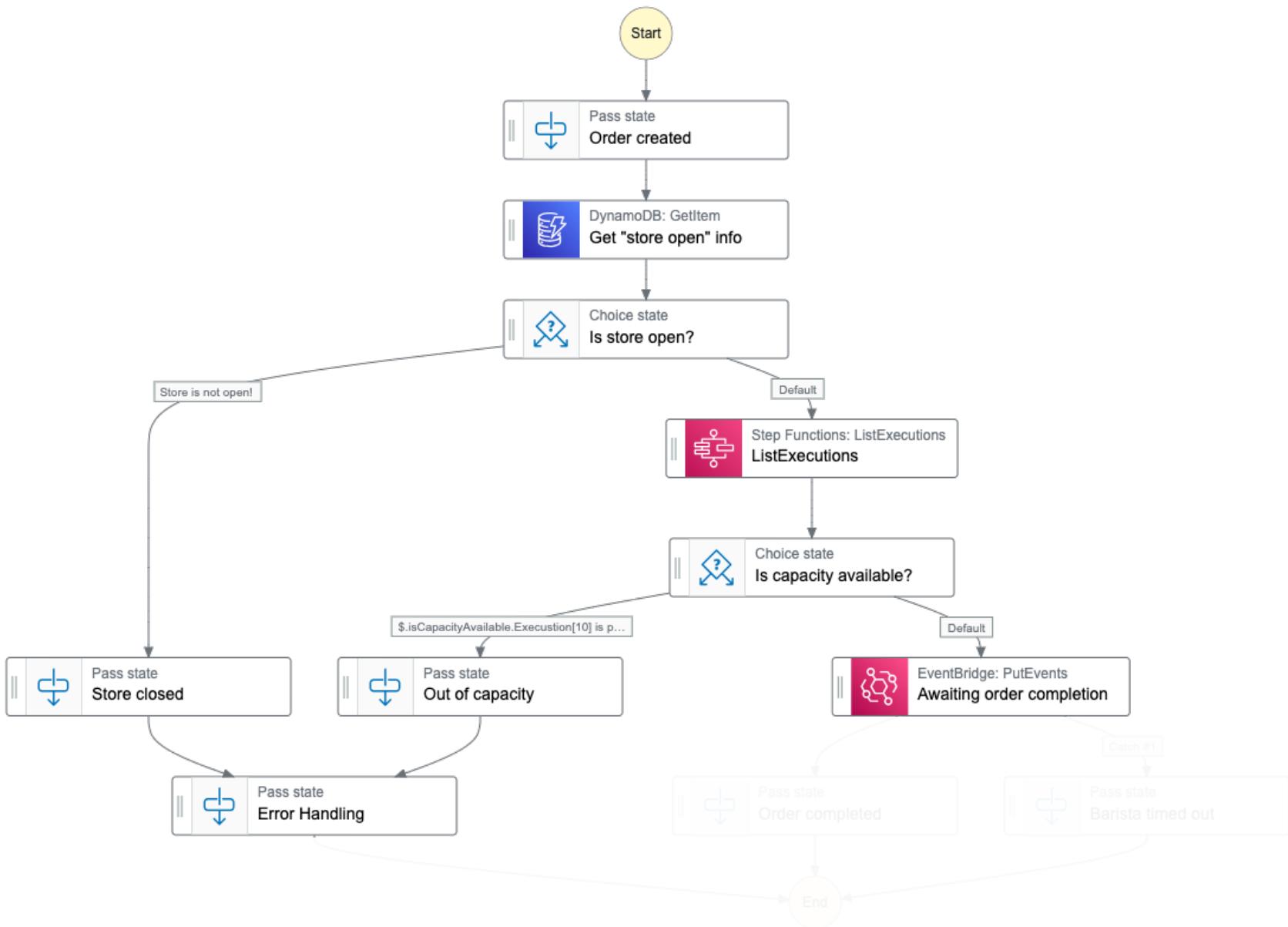
Order Processing: Initialization



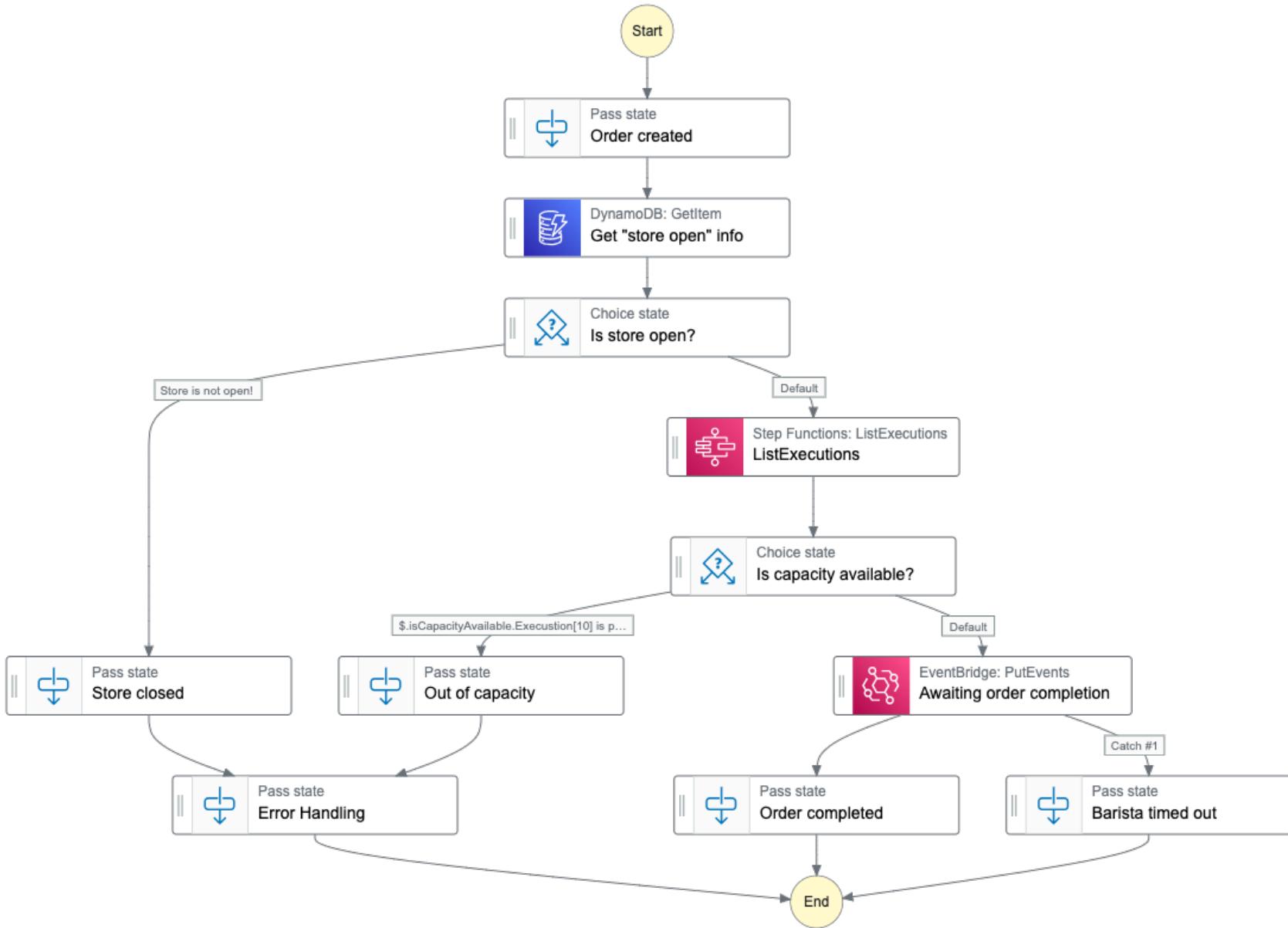
Order Processing: Is store open?



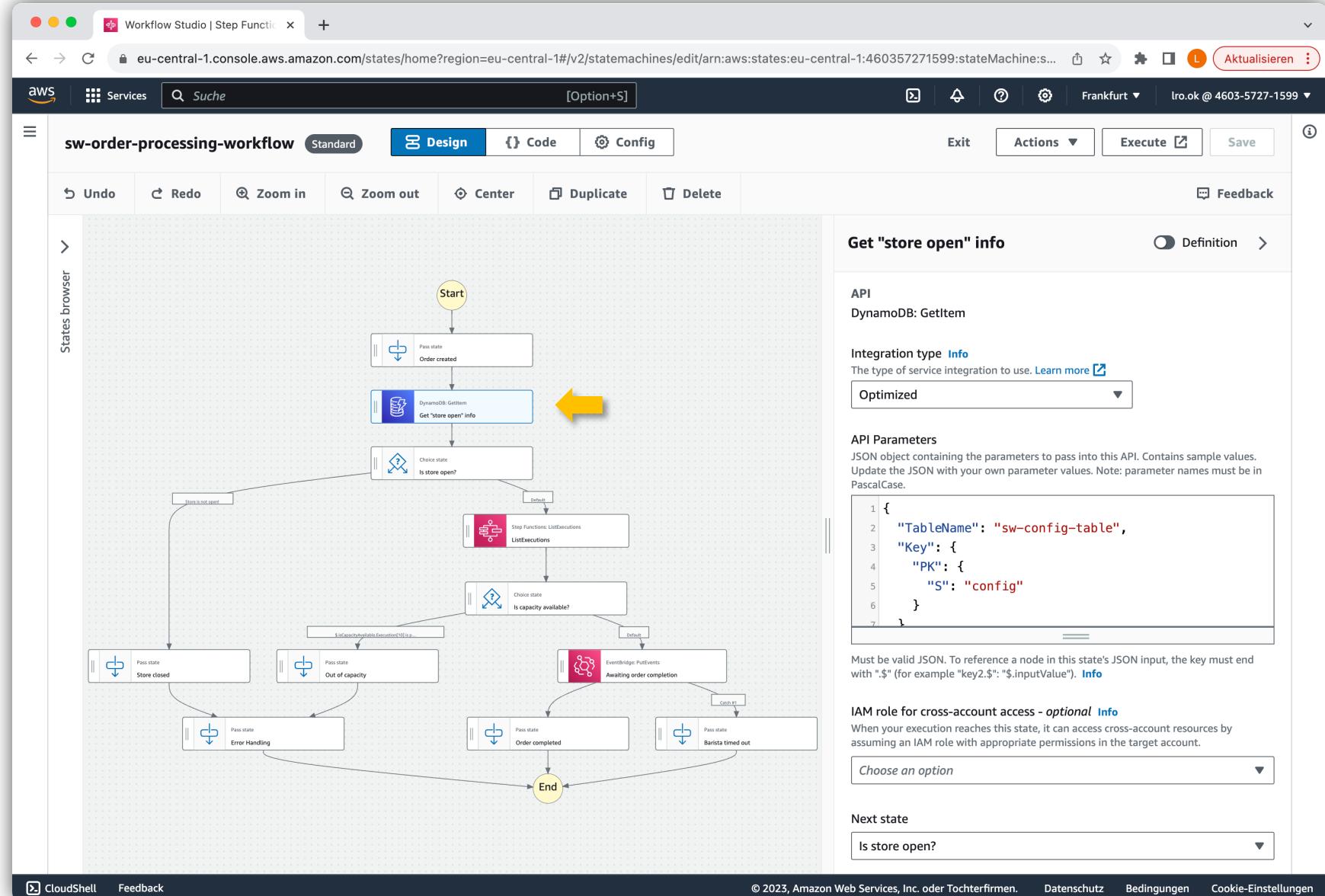
Order Processing: Is capacity available?

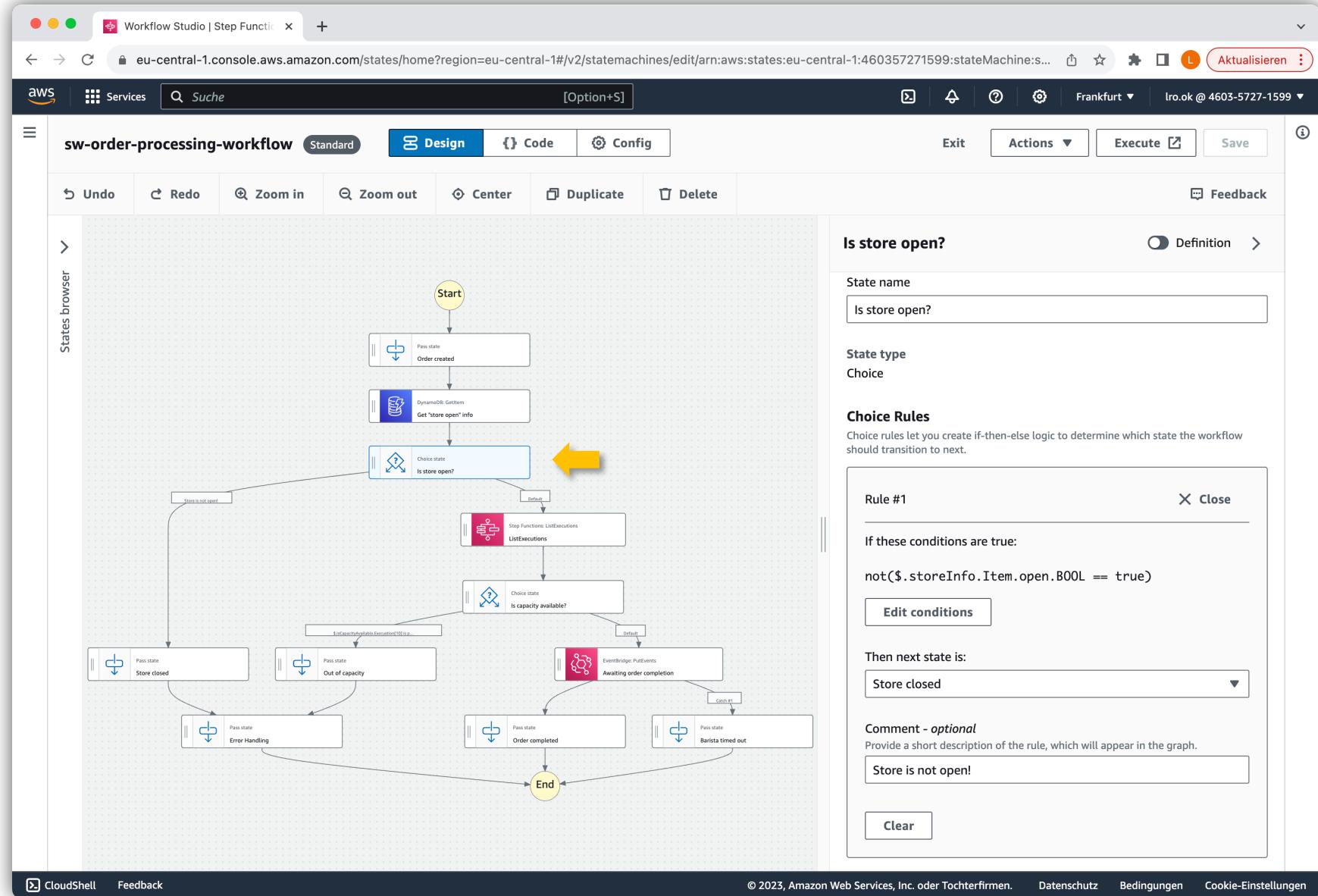


Order Processing: Awaiting completion!

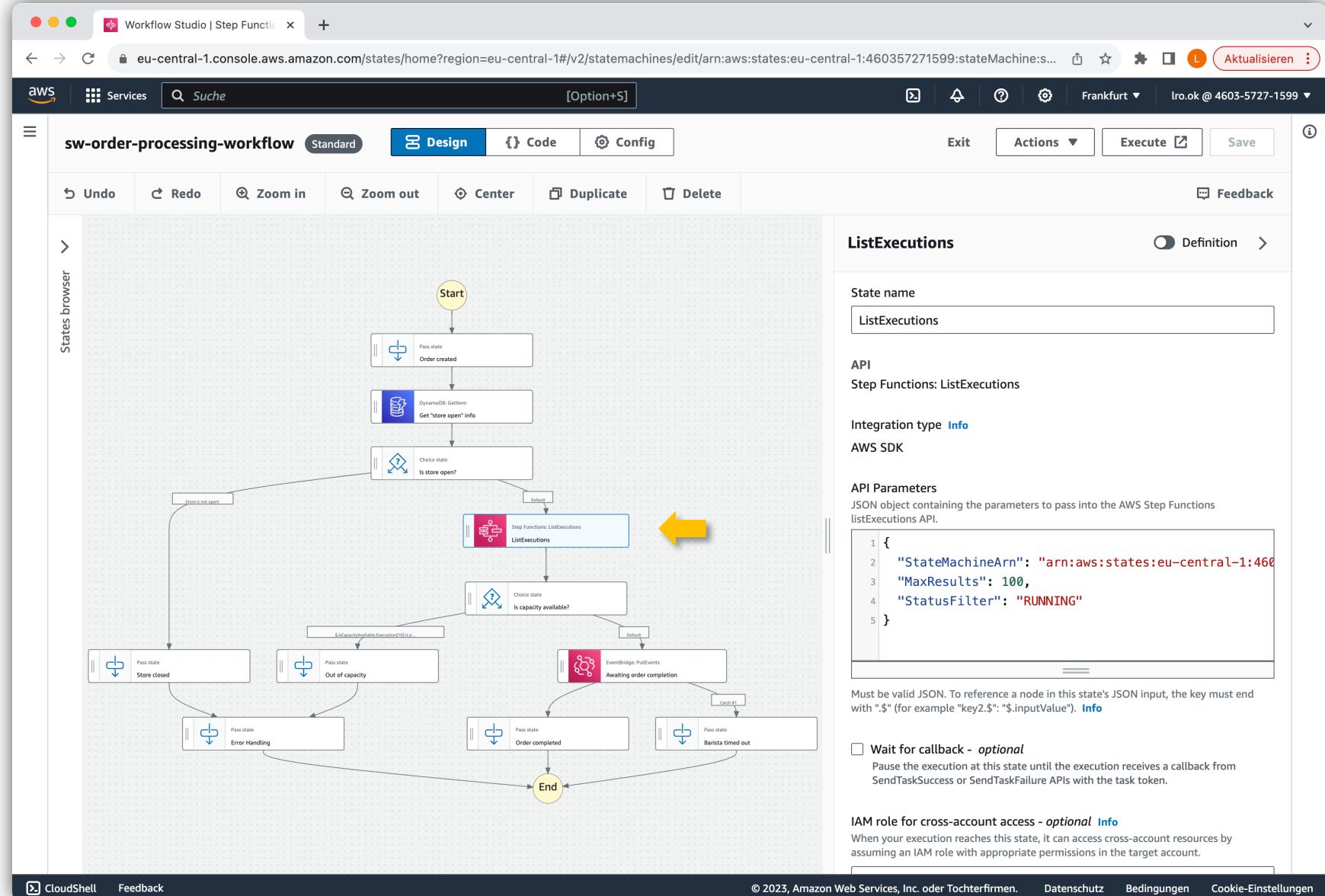


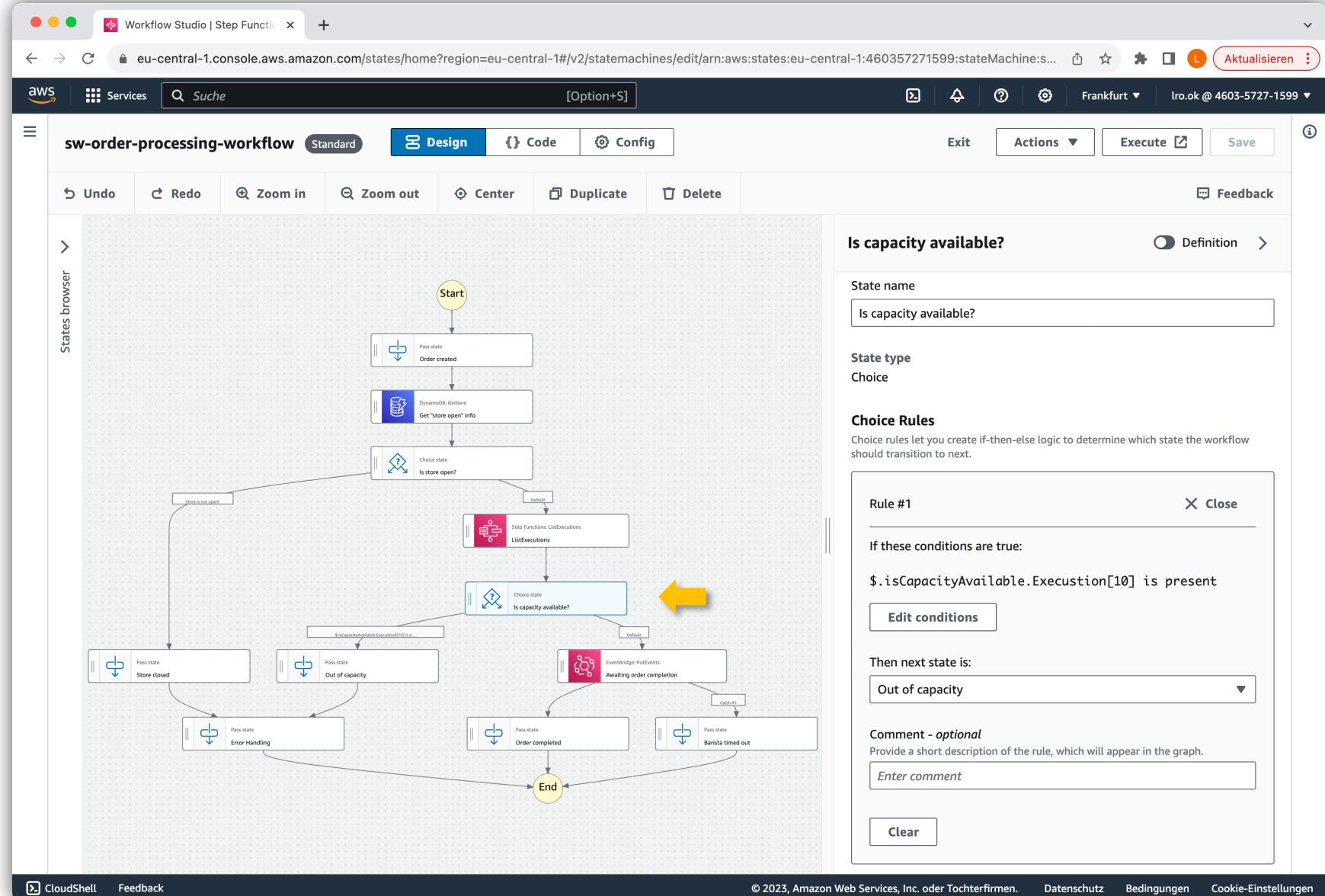
Order Processing: Final Workflow



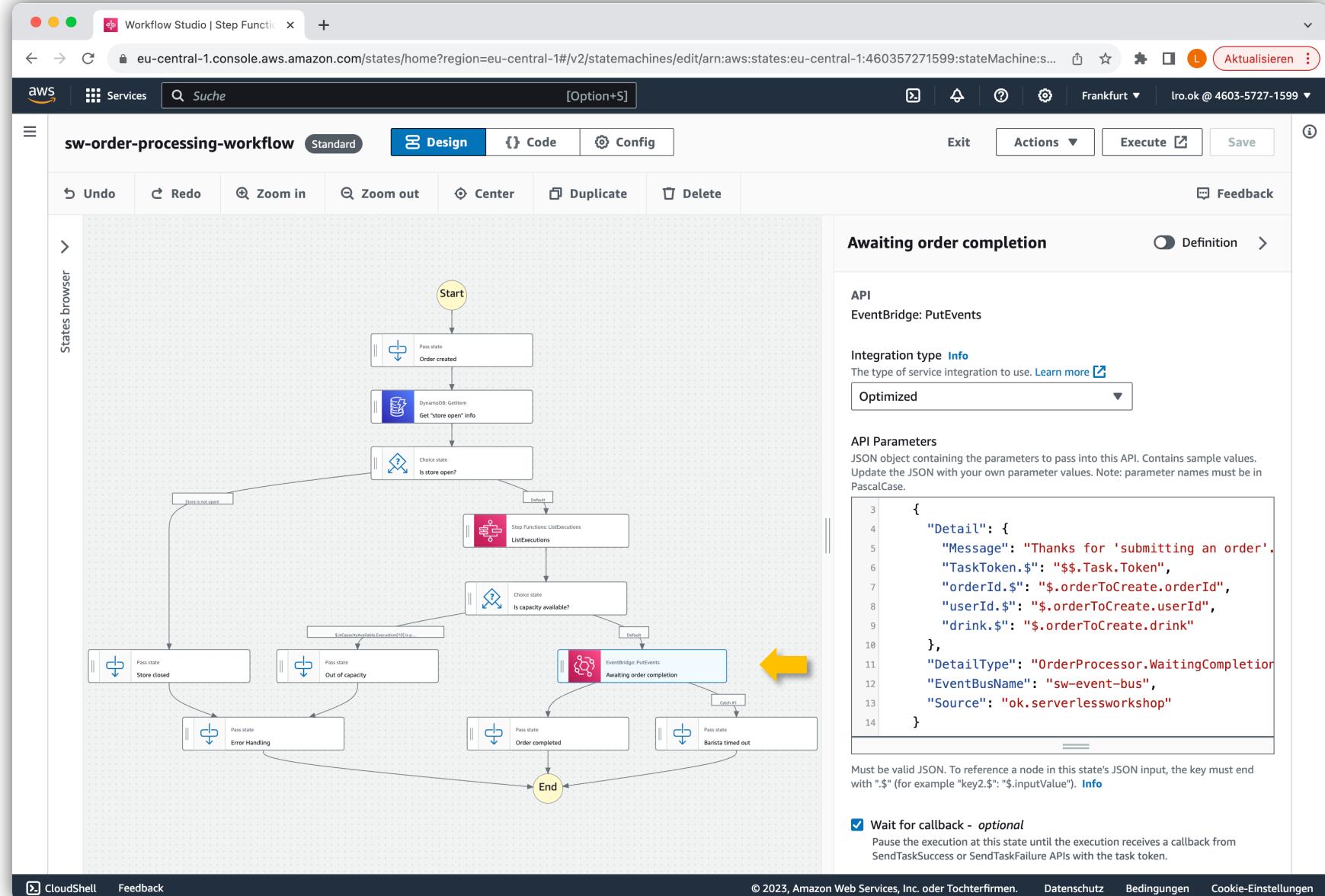


Order Processing: Check if Store Open





Order Processing: Check Capacity



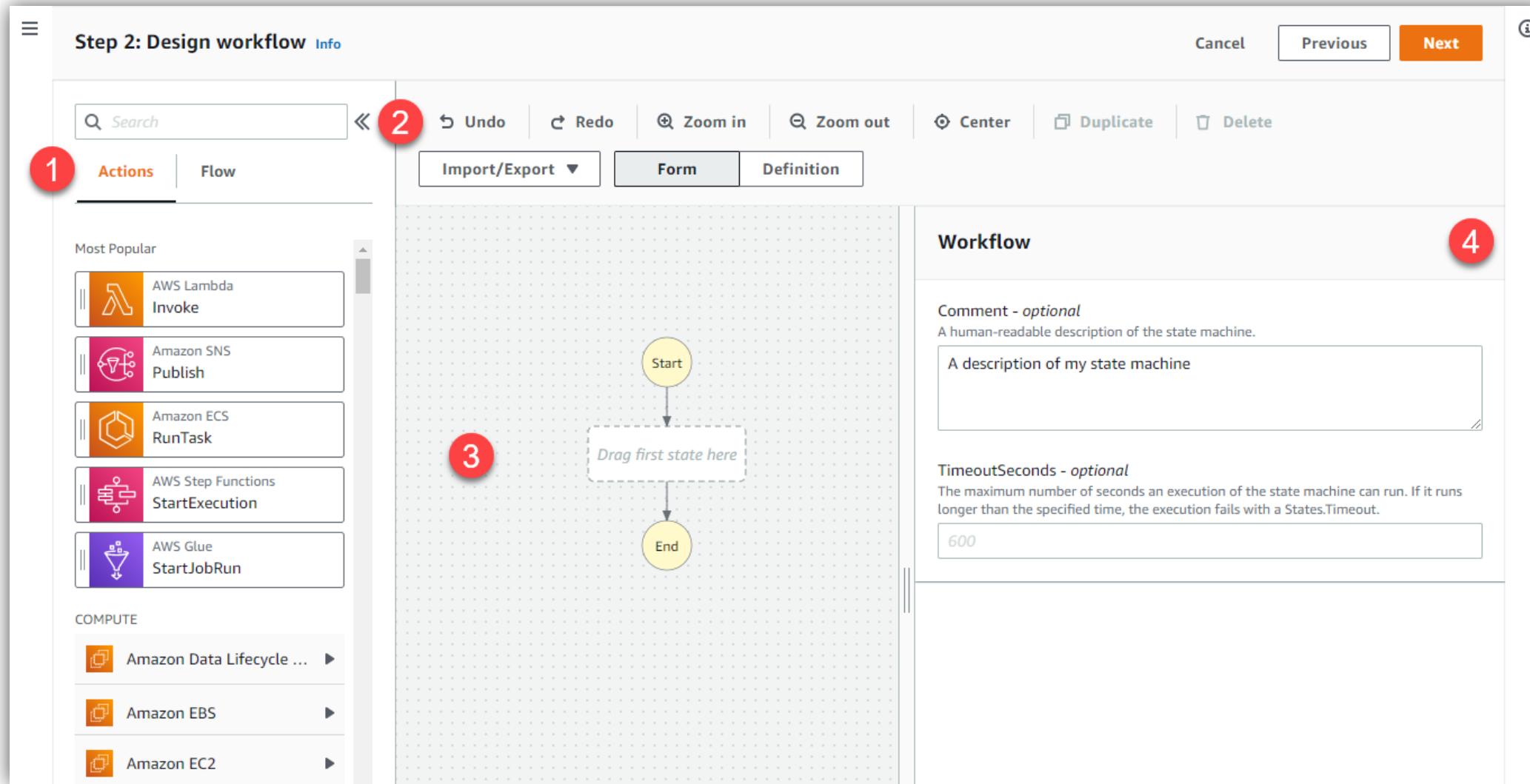
Order Processing: Send Event to EventBridge and ... WAIT

HANDS-ON

Order Processing

- Store open?
- Capacity available?
- Coffee done?
- Success Handling
- Error Handling





Lessons Learned:

Workflows via StepFunctions

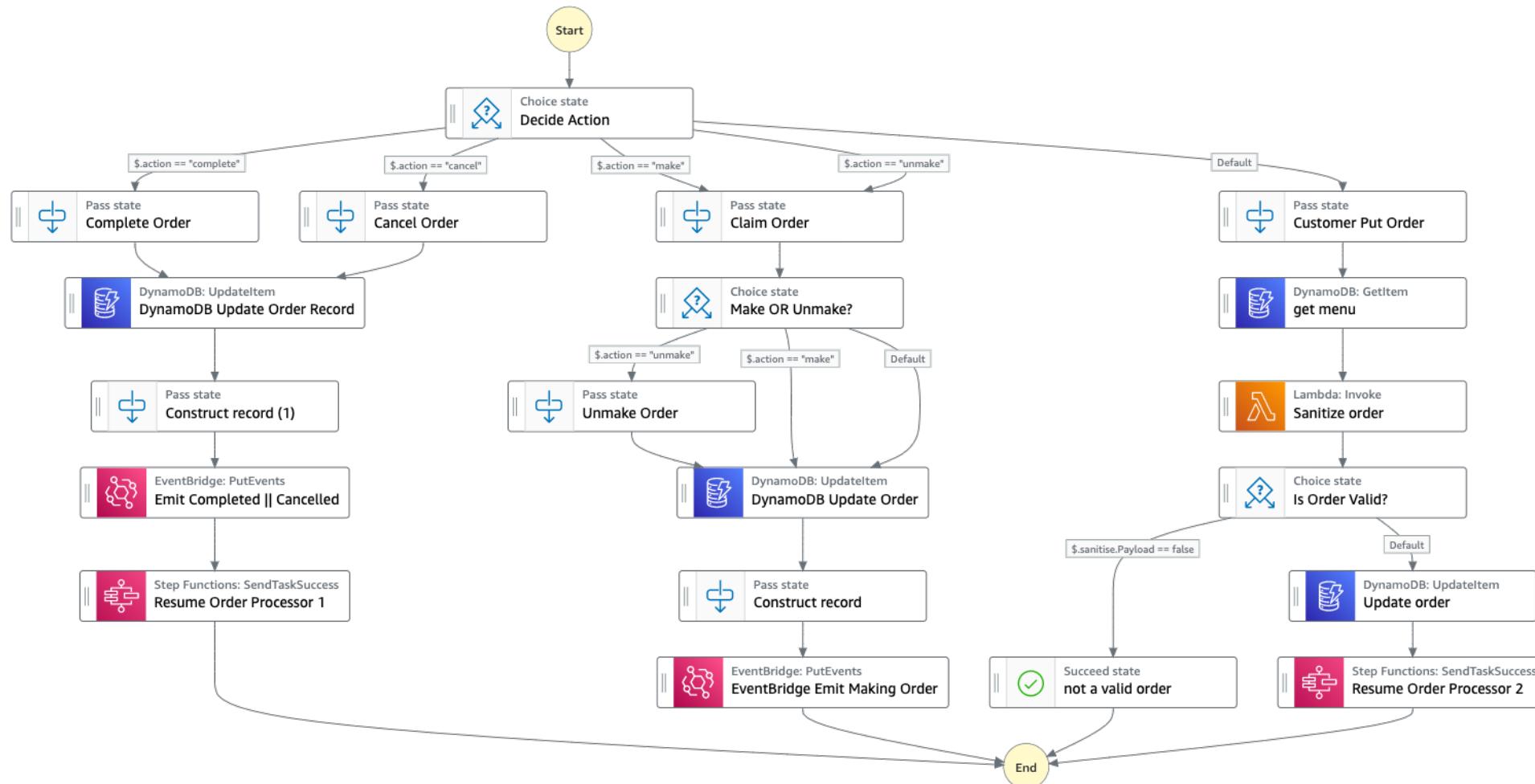


WORKFLOWS VIA STEP FUNCTIONS

Lessons Learned

- **Flow Elements**
- **Action Elements**
- **Callback** via EventBridge Events
- **Payload Manipulation** via JSONPath
- **Success Handling**
- **Error Handling**

WORKFLOWS VIA STEP FUNCTIONS



AWS StepFunctions: Order Management Service (alternative Version)



Agenda

Hello World & Set-up



Order Management



Order Processing



API Gateway

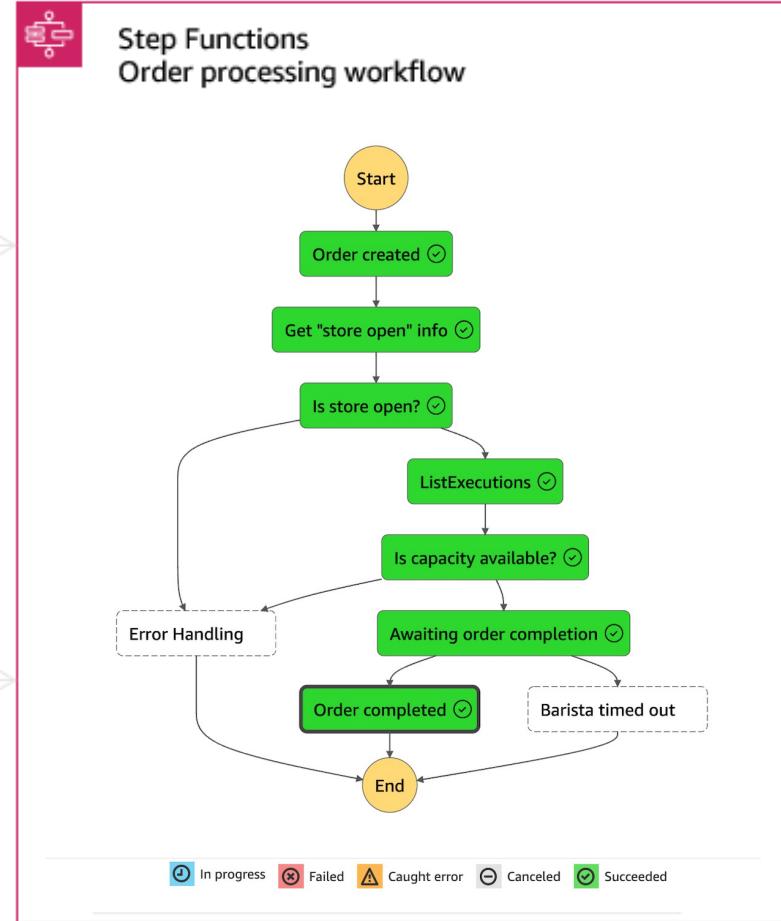
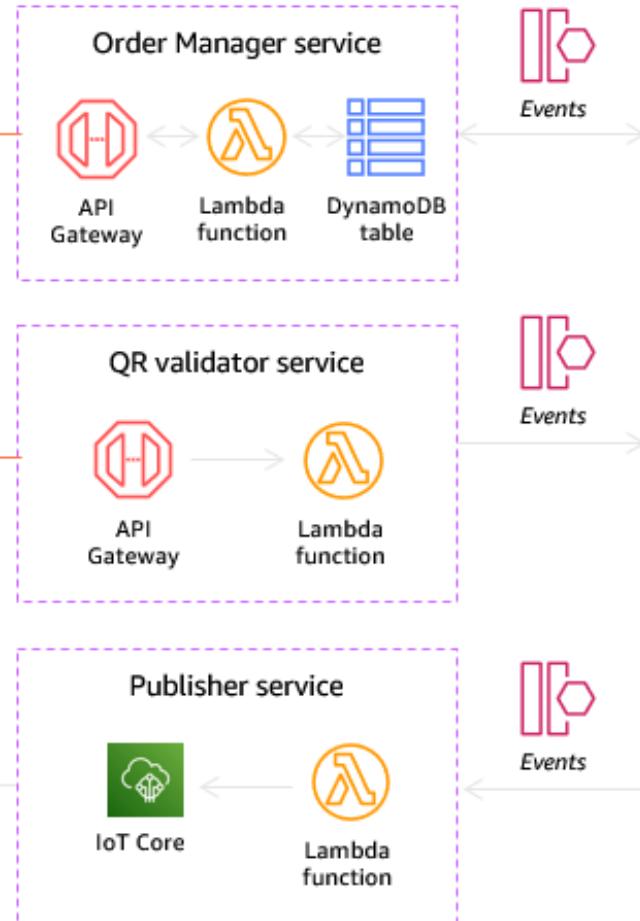
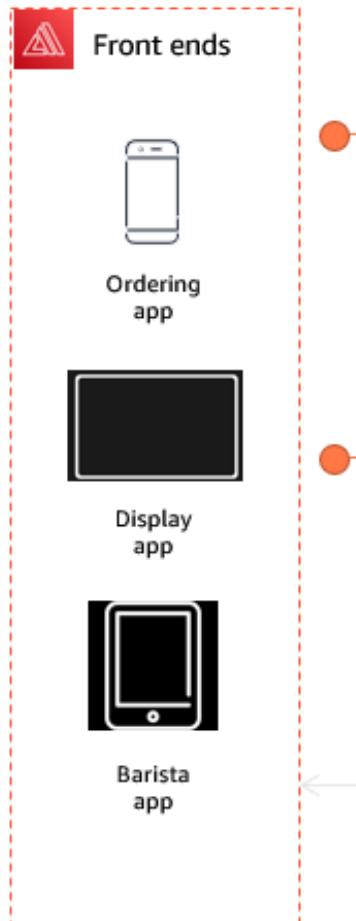
Testing



Serverless Coffee API Gateway

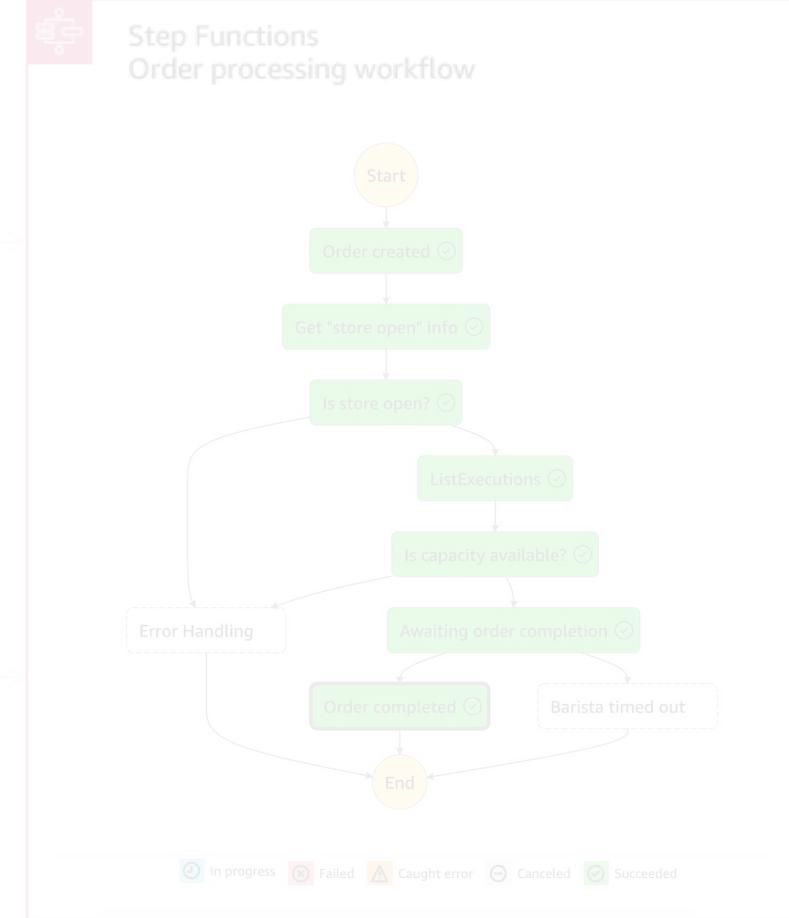
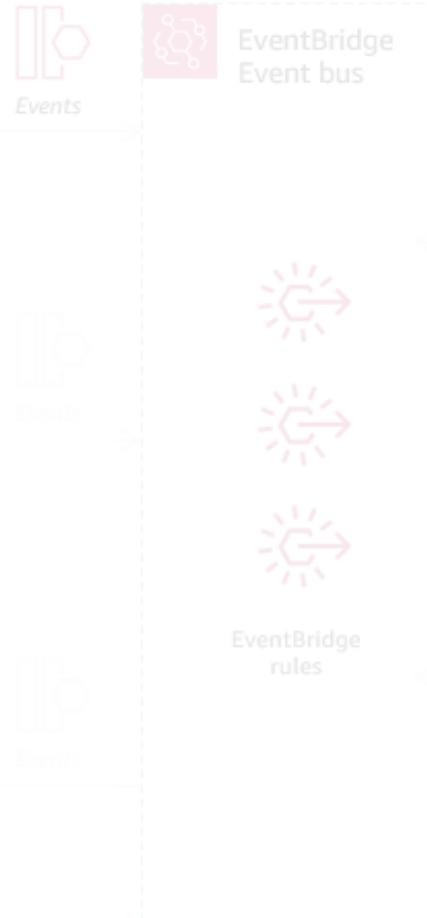


AWS Cloud

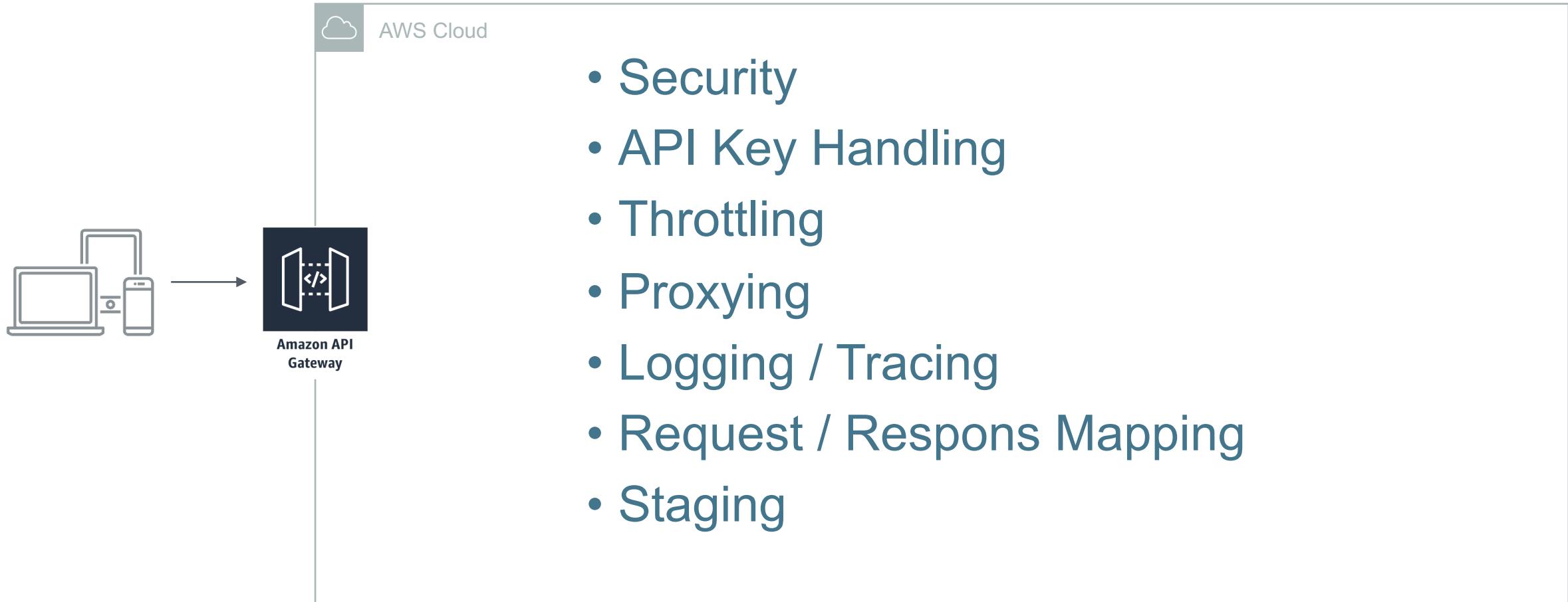




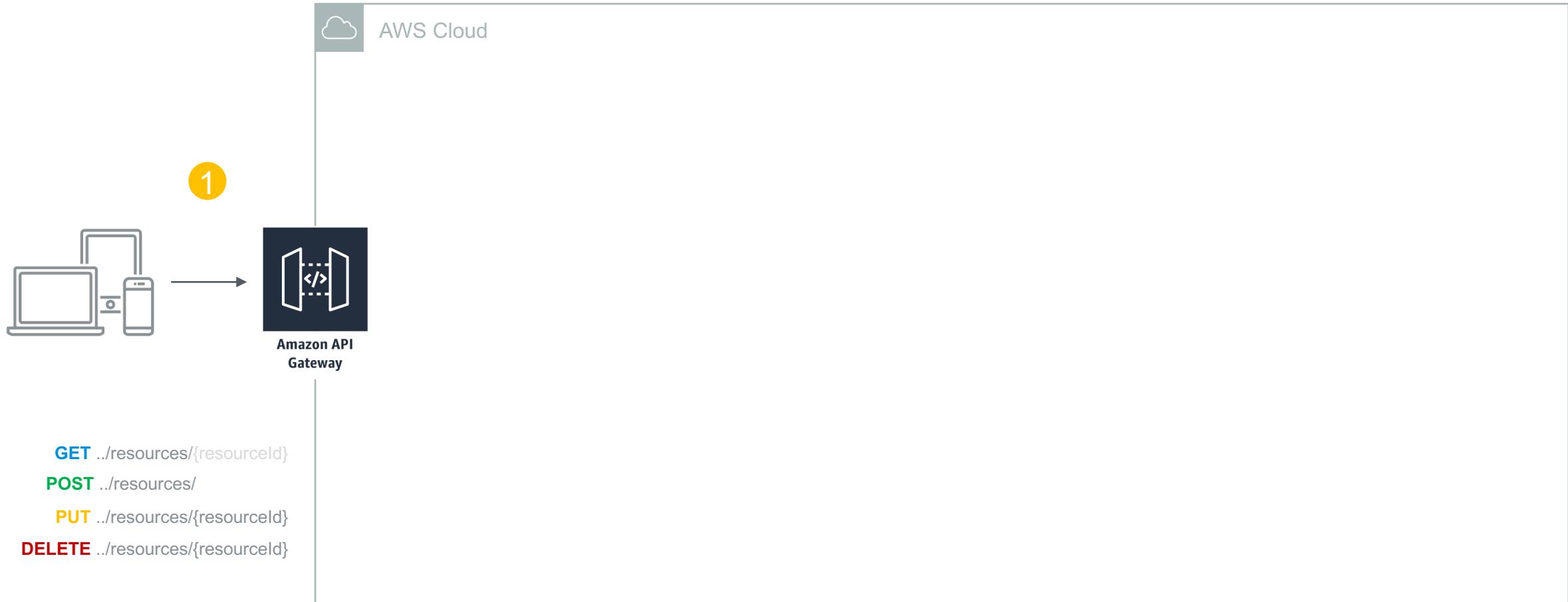
AWS Cloud



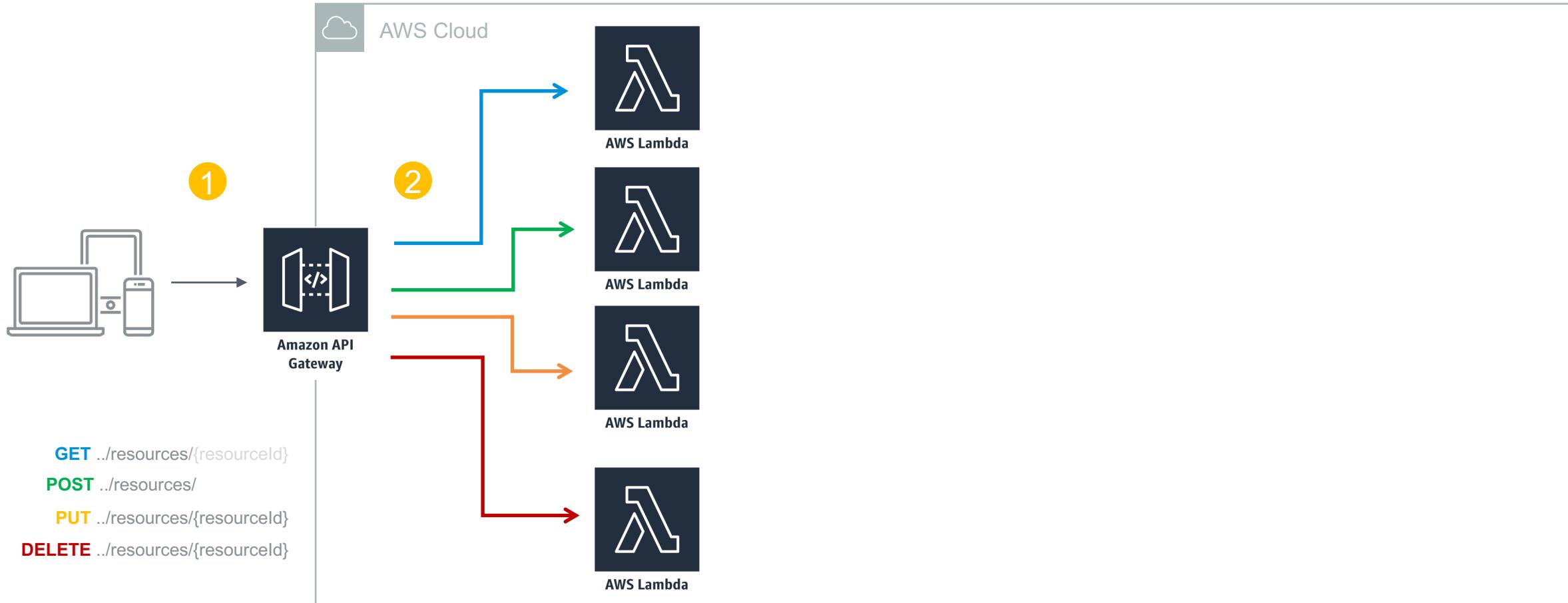
Wofür ist ein API Gateway gut?



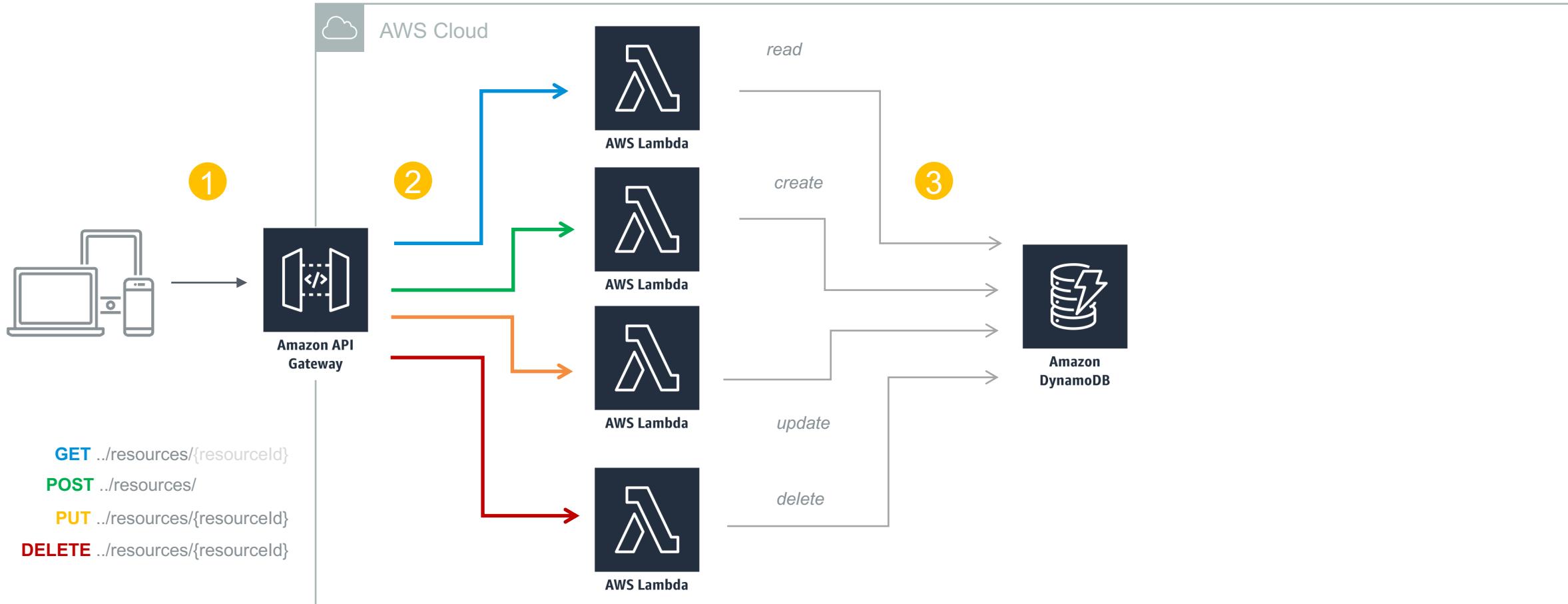
AWS API Gateway



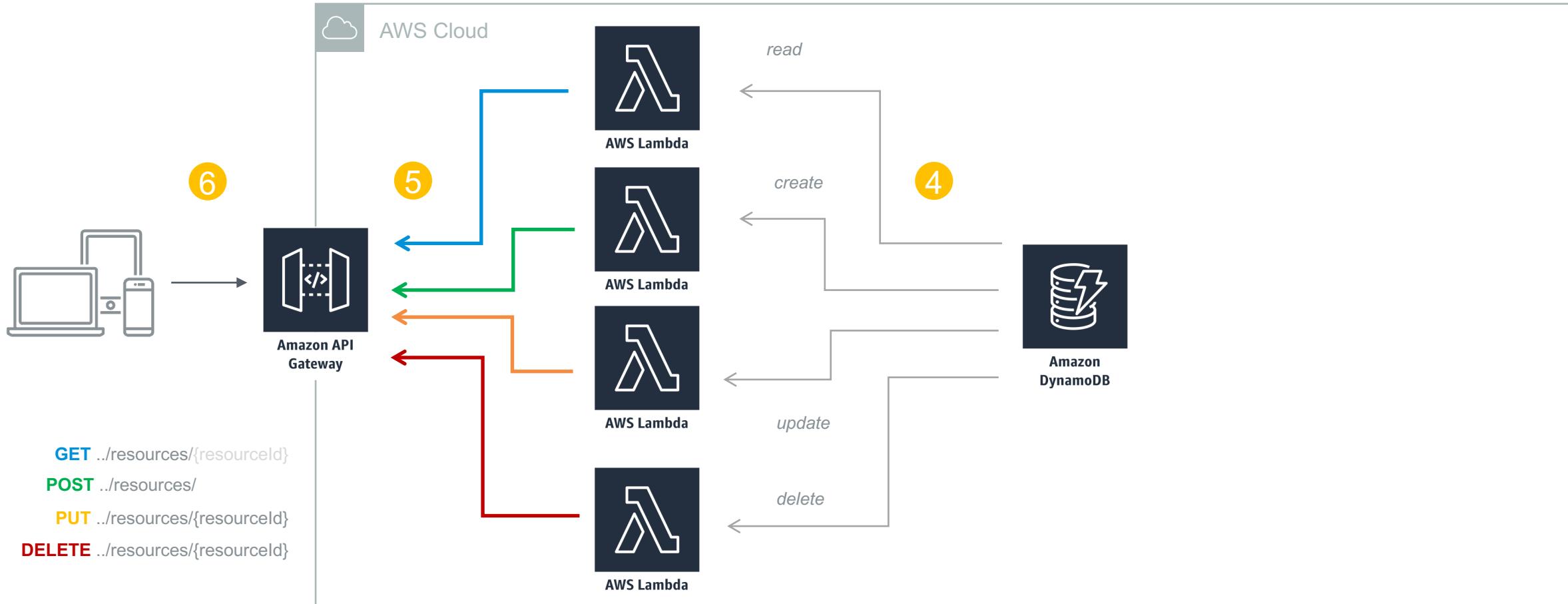
AWS API Gateway



AWS API Gateway



AWS API Gateway



HANDS-ON

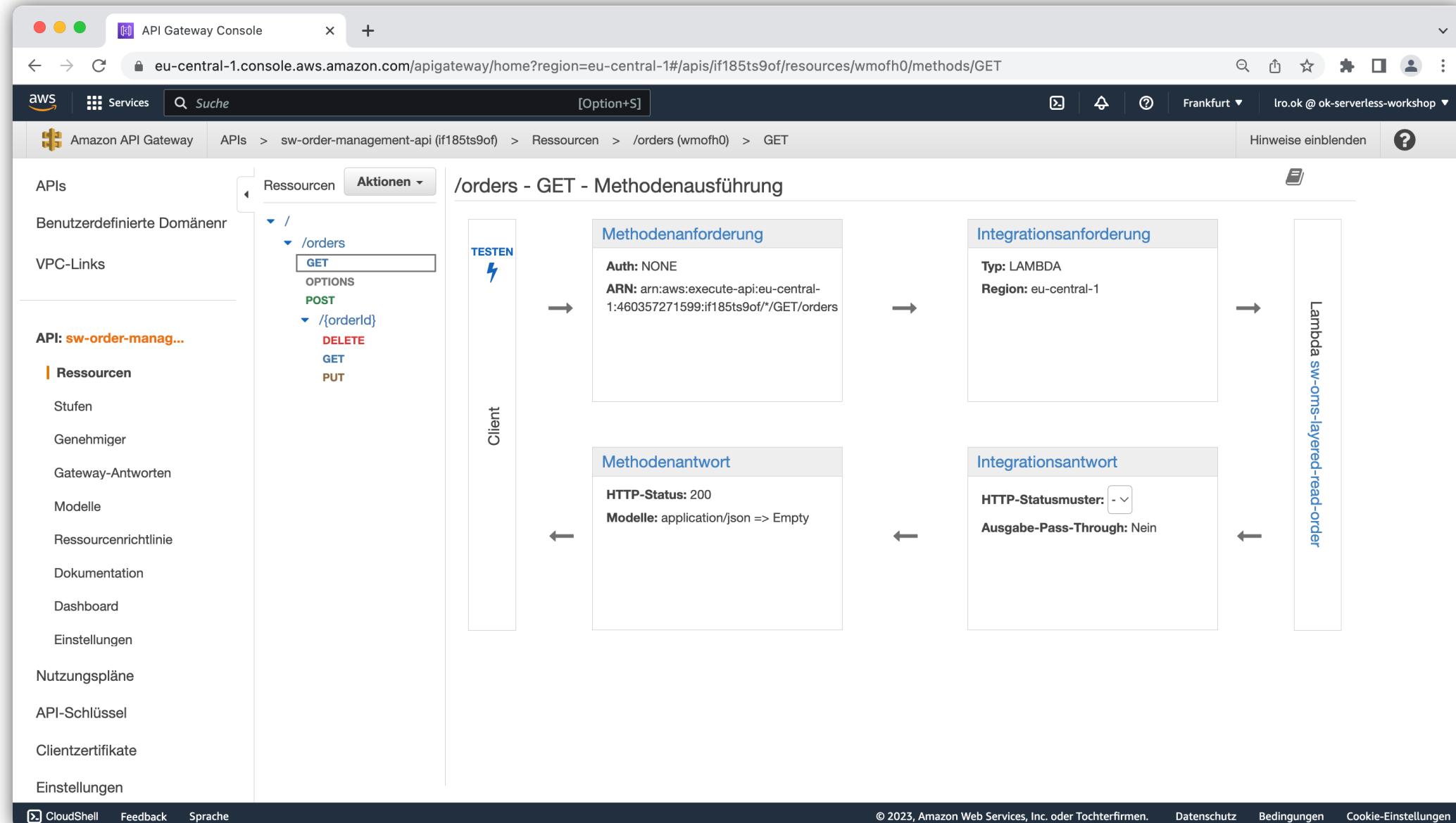
API Gateway

- Backend Integration
- Request Mapping
- Response Mapping
- Mocking
- Multiple Stages



The screenshot shows the AWS API Gateway console interface. The left sidebar has a 'API Gateway' title and a 'APIs' section selected. The main area displays a table titled 'APIs (1)'. The table has columns: Name, Beschreibung, ID, Protokoll, Endpunkttyp, and Erstellt. One row is shown, corresponding to the 'sw-order-management-api' listed in the sidebar. The table includes a search bar at the top and various buttons like 'C', 'Aktionen', and 'API erstellen'.

| Name | Beschreibung | ID | Protokoll | Endpunkttyp | Erstellt |
|-------------------------|----------------------------------|------------|-----------|-------------|------------|
| sw-order-management-api | RESTful API for Order Management | if185ts9of | REST | Regional | 2023-06-08 |



API Gateway: Ressourcen & Methoden

API Gateway Console x +

eu-central-1.console.aws.amazon.com/apigateway/home?region=eu-central-1#/apis/if185ts9of/resources/wmofh0/methods/GET

aws Services Suche [Option+S]

Amazon API Gateway APIs > sw-order-management-api (if185ts9of) > Ressourcen > /orders (wmofh0) > GET Frankfurt Iro.ok @ ok-serverless-workshop Hinweise einblenden ?

APIs Ressourcen Aktionen / /orders GET OPTIONS POST /{orderId} DELETE GET PUT

Methodenausführung /orders - GET - Methodentest

Machen Sie einen Testaufruf an Ihre Methode. Wenn Sie einen Testanruf tätigen, überspringt API Gateway die Autorisierung und ruft Ihre Methode direkt auf

Pfad Anforderungsdatum: /orders
Status: 200
Latenz: 6898 ms
Antworttext

Für diese Ressource sind keine Pfadparameter vorhanden. Sie können Pfadparameter mit der Syntax `{myPathParam}` im Ressourcenpfad definieren.

Abfragezeichenfolgen {orders}

Header {orders}

Stufenvariablen Für diese Methode sind keine Stufenvariablen vorhanden.

Anforderungstext Der Anforderungstext wird für GET-Methoden nicht unterstützt.

Antwort-Header

```
{ "status": "OK", "message": "Orders successfully read.", "orderCount": 2, "orders": [ { "orderId": "b942cac4-8717-4b8d-ac4c-024581ab69fb", "orderNo": 1, "orderStatus": "confirmed", "userId": "lars.roewekamp", "drink": "Americano grande SUPER MILKY" }, { "orderId": "560bf13d-1b19-404c-a7dd-28dd4182e2aa", "orderNo": 2, "orderStatus": "confirmed", "userId": "lars.roewekamp", "drink": "Americano grande" } ] }
```

CloudShell Feedback Sprache © 2023, Amazon Web Services, Inc. oder Tochterfirmen. Datenschutz Bedingungen Cookie-Einstellungen

API Gateway: GET all orders

API Gateway Console x +

eu-central-1.console.aws.amazon.com/apigateway/home?region=eu-central-1#/apis/if185ts9of/resources/wmofh0/methods/POST

Services Suche [Option+S]

Amazon API Gateway APIs > sw-order-management-api (if185ts9of) > Ressourcen > /orders (wmofh0) > POST

Hinweise einblenden ?

APIs Ressourcen Aktionen ▾

Benutzerdefinierte Domänenre... VPC-Links

API: sw-order-manag... Ressourcen

- Stufen
- Genehmiger
- Gateway-Antworten
- Modelle
- Ressourcenrichtlinie
- Dokumentation
- Dashboard
- Einstellungen
- Nutzungspläne
- API-Schlüssel
- Clientzertifikate
- Einstellungen

Methodenausführung /orders - POST - Methodentest

Machen Sie einen Testaufruf an Ihre Methode. Wenn Sie einen Testanruf tätigen, überspringt API Gateway die Autorisierung und ruft Ihre Methode direkt auf

Pfad Anforderungsdatum: /orders
Status: 422
Latenz: 1560 ms
Antworttext

Für diese Ressource sind keine Pfadparameter vorhanden. Sie können Pfadparameter mit der Syntax **{myPathParam}** im Ressourcenpfad definieren.

Abfragezeichenfolgen {orders}

Header {orders}

Stufenvariablen Für diese Methode sind keine Stufenvariablen vorhanden.

Anforderungstext 1

Antwort-Header {"Content-Type": ["application/json"], "X-Amzn-Trace-Id": ["Root=1-648c9d2a-8498dc97a6030e4f2a53fdd0; Sampled=0; lineage=e7f97465:0"]}

Protokolle Execution log for request de5ab12e-eb54-445c-90a8-8da9801ec16 Fri Jun 16 17:34:34 UTC 2023 : Starting execution for request de5ab12e-eb54-445c-90a8-8da9801ec16

CloudShell Feedback Sprache © 2023, Amazon Web Services, Inc. oder Tochterfirmen. Datenschutz Bedingungen Cookie-Einstellungen

API Gateway: POST new order (ERROR)

API Gateway Console x +

eu-central-1.console.aws.amazon.com/apigateway/home?region=eu-central-1#/apis/if185ts9of/stages/SW-STAGE

aws Services Suche [Option+S]

Amazon API Gateway APIs > sw-order-management-api (if185ts9of) > Stufen > SW-STAGE Frankfurt Hinweise einblenden ?

Erstellen Stufen SW-STAGE Stage-Editor Stufe löschen Tags konfigurieren

Aufruf-URL: <https://if185ts9of.execute-api.eu-central-1.amazonaws.com/SW-STAGE>

Einstellungen Protokolle/Ablaufverfolgung Stufenvariablen Generieren von SDKs Exportieren Bereitstellungsverlauf Verlauf der Dokumentation Canary

Cache-Einstellungen

API-Cache aktivieren

Standard-Methode für das Drosseln

Legen Sie die Standarddrosselungsebene für die Methoden in dieser Stufe fest. Diese Einstellungen für Rate und Steigerung werden von allen Methoden dieser Stufe eingehalten. Ihre aktuelle Drosselungsrate auf Kontoebene beträgt **10000** Anforderungen pro Sekunde mit einer Steigerung von **5000** Anforderungen. [Weitere Informationen zur Drosselung in API Gateway](#)

Drosselung aktivieren ⓘ

Tarif Anforderungen pro Sekunde

Steigerung Anforderungen

Webanwendungsfirewall (WAF) [Weitere Informationen](#).

Wählen Sie die Web-ACL aus, die auf diese Stufe angewendet werden soll.

Web-ACL

Clientzertifikat

CloudShell Feedback Sprache © 2023, Amazon Web Services, Inc. oder Tochterfirmen. Datenschutz Bedingungen Cookie-Einstellungen

The screenshot shows the AWS API Gateway Stage Editor for the 'SW-STAGE'. On the left, a sidebar lists various API management features. The main area displays the 'Einstellungen' tab of the stage editor. It includes a 'Cache-Einstellungen' section with a checkbox for 'API-Cache aktivieren'. Below it is a 'Standard-Methode für das Drosseln' section with a note about rate and burst limits. A 'Drosselung aktivieren' checkbox is checked. Underneath are fields for 'Tarif' (10000 requests per second) and 'Steigerung' (5000 requests per second). There's also a 'Webanwendungsfirewall (WAF)' section and a 'Web-ACL' dropdown set to 'Keine'. At the bottom, there's a 'Clientzertifikat' section. The top navigation bar shows the URL 'eu-central-1.console.aws.amazon.com/apigateway/home?region=eu-central-1#/apis/if185ts9of/stages/SW-STAGE' and the AWS logo.

API Gateway: Staging

The screenshot shows the Postman application interface. The top navigation bar includes Home, Workspaces, API Network, Explore, a search bar, and various user settings. The left sidebar, titled "My Workspace", lists collections like "Serverless Workshop" and "API Gateway", along with APIs, environments, mock servers, and history. The main workspace displays an API endpoint for reading orders:

HTTP Serverless Workshop / API Gateway / **read all orders**

GET https://{{gateway_url}}/orders

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

| | Key | Value | Description | ... | Bulk Edit |
|--|-----|-------|-------------|-----|-----------|
| | Key | Value | Description | | |

Body Cookies Headers (7) Test Results Status: 200 OK Time: 512 ms Size: 806 B Save as Example ...

Pretty Raw Preview Visualize JSON

```
1 {  
2   "status": "OK",  
3   "message": "Orders successfully read.",  
4   "orderCount": 2,  
5   "orders": [  
6     {  
7       "orderId": "b942cac4-8717-4b8d-ac4c-024581ab69fb",  
8       "orderNo": 1,  
9       "orderStatus": "confirmed",  
10      "userId": "lars.roewekamp",  
11      "drink": "Americano grande SUPER MILKY"  
12    }  
]
```

At the bottom, there are buttons for Online, Find and replace, Console, Runner, Capture requests, Cookies, Trash, and Help.

API Gateway: External Call (via POSTMAN)

Lessons learned:

API Gateway



API GATEWAY

Lessons Learned

- **Schutztor** zur der Aussenwelt
- **HTTP Request** werden zu Lambda Triggern
- **HTTP Payload** wird zu Event Payload
- **Lambda Result** wird zu HTTP Status Code
- **Lambda Result Object** wird zu HTTP Payload
- **Staging-Konzept** sorgt für Flexibilität



Agenda

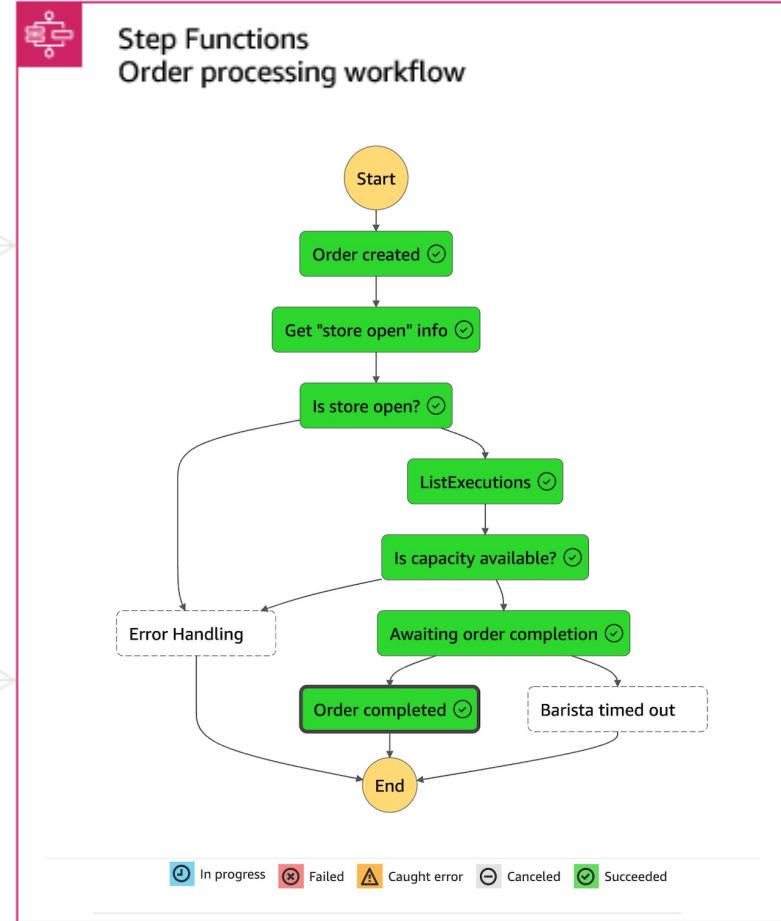
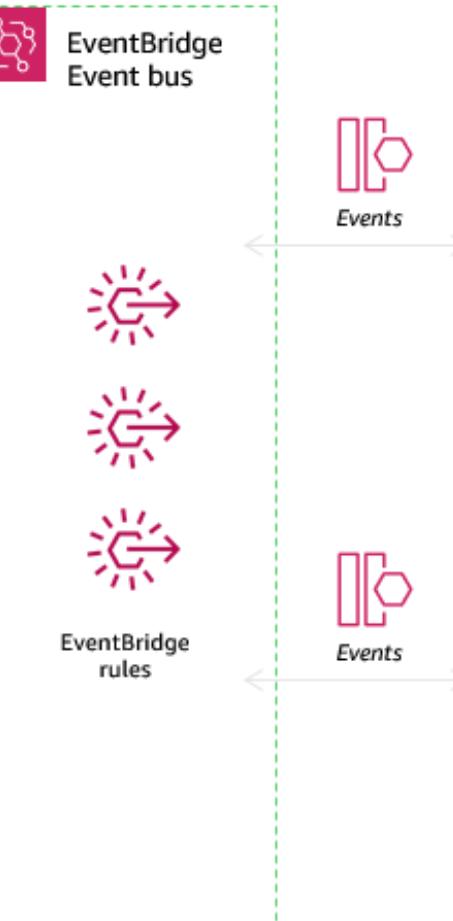
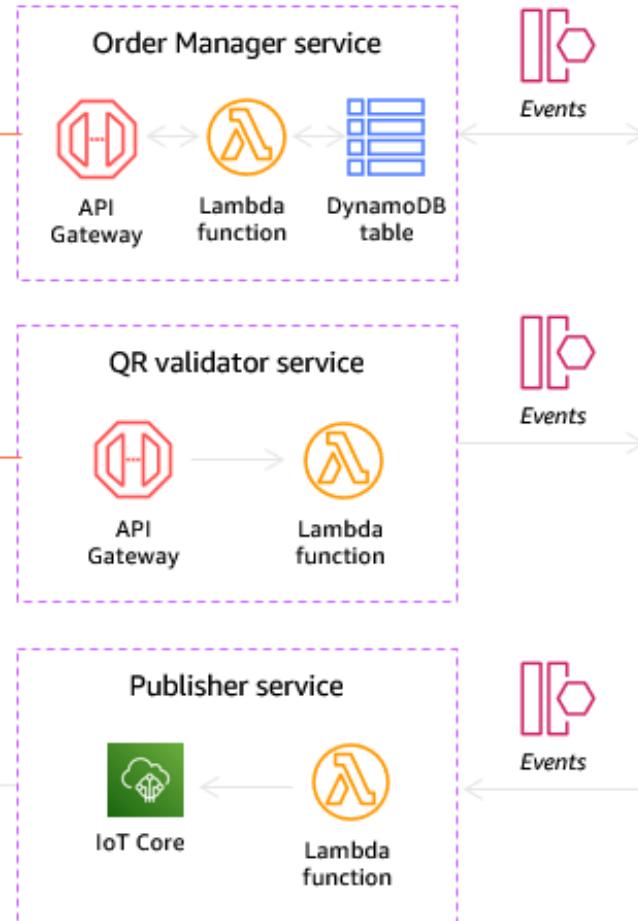
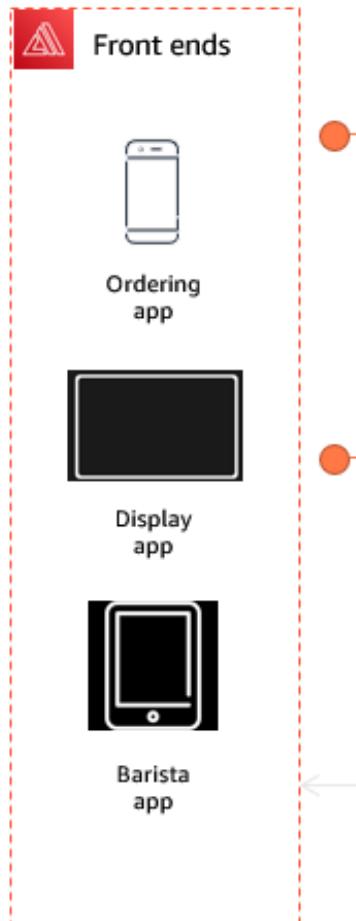
- Hello World & Set-up*
- Order Management*
- Order Processing*
- API Gateway*
- Testing*



Serverless Coffee *Testing U/I/A*



AWS Cloud



HANDS-ON

Testing

- Unit Testing
- Integration Testing
- Acceptance Testing





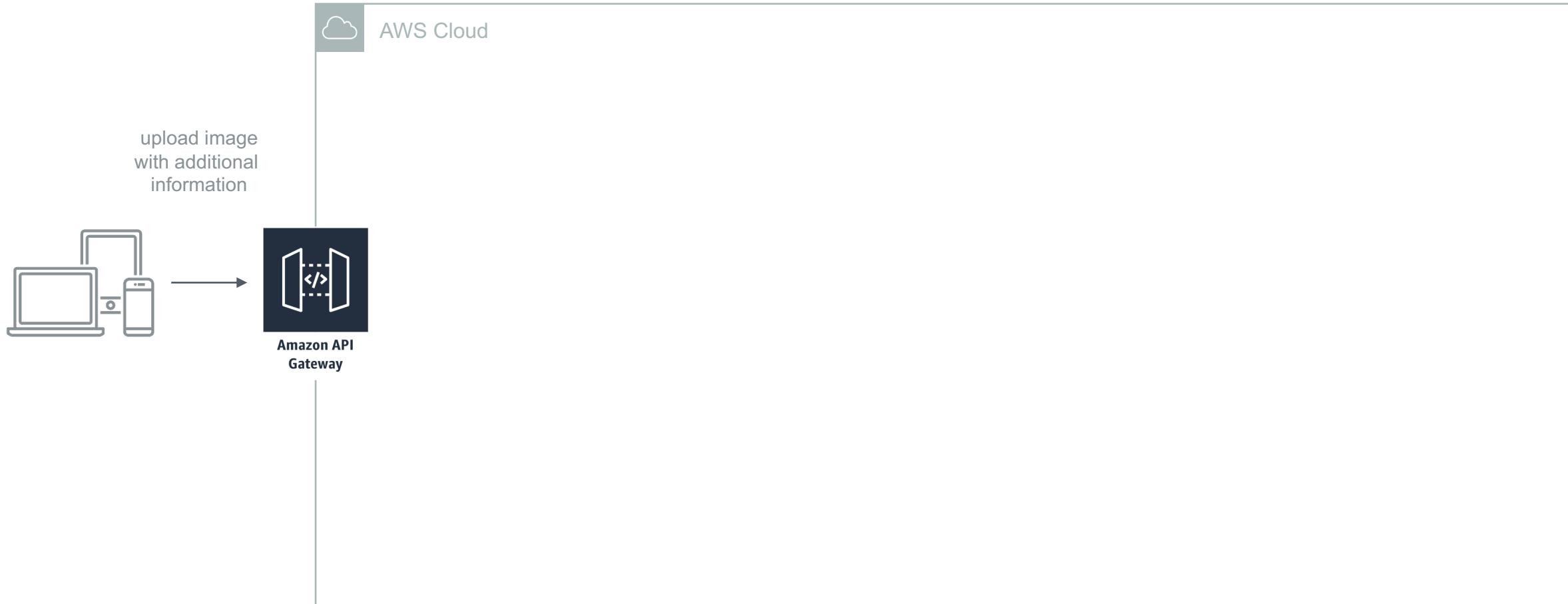
USE CASE

OrderApp

Admin erstellt neuen Menu-Eintrag via API Gateway

POST /drinks ...

Upload Data & Image



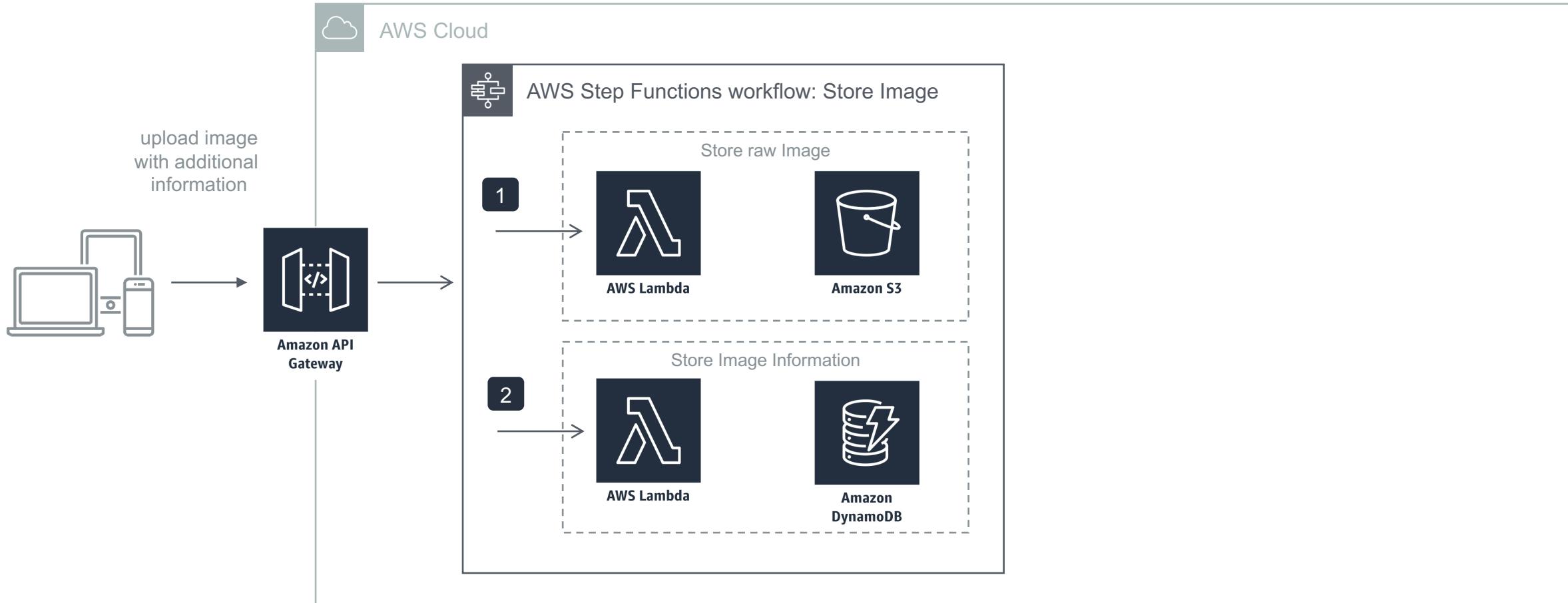
Upload Data & Image



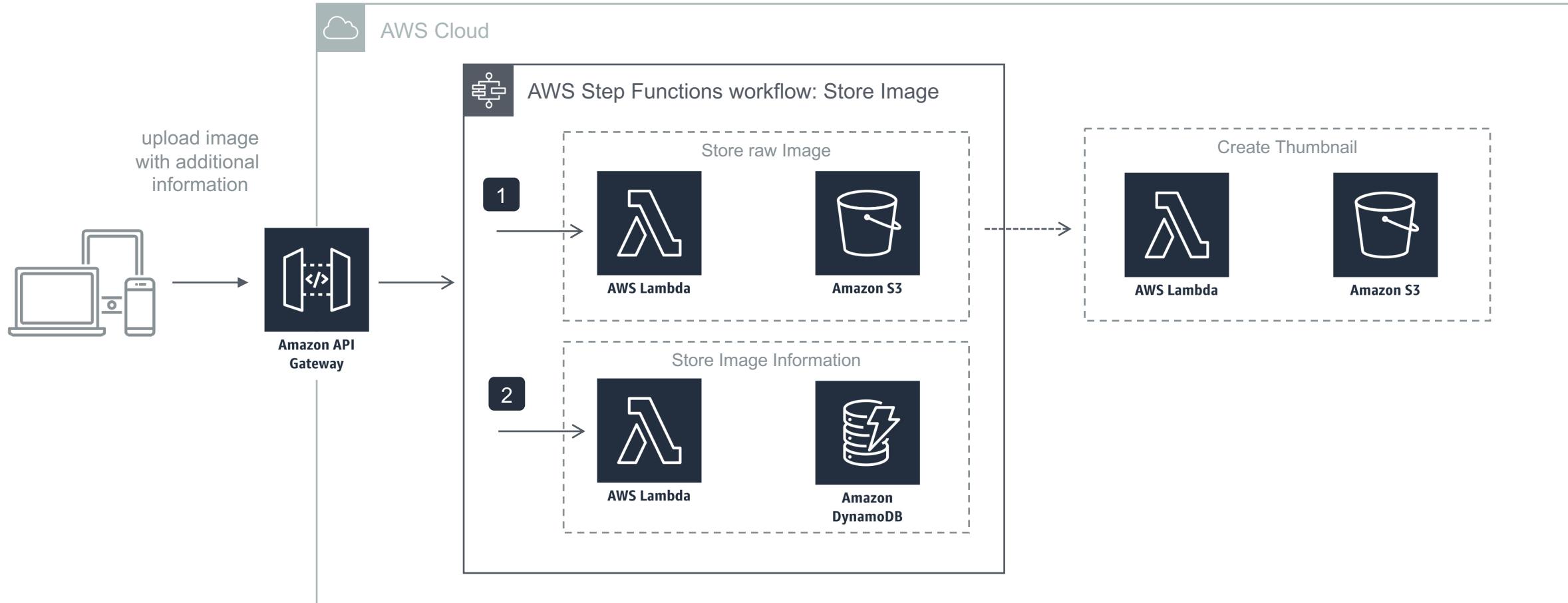
Upload Data & Image



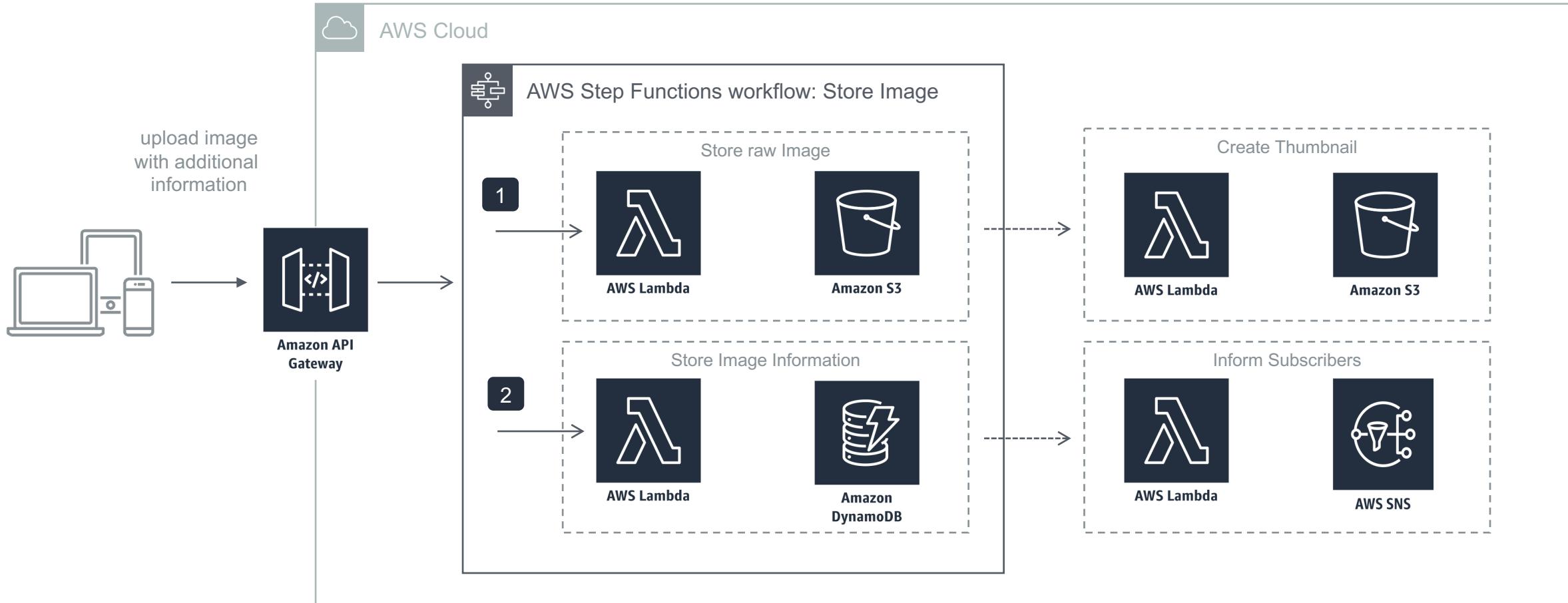
Upload Data & Image



Upload Data & Image



Upload Data & Image

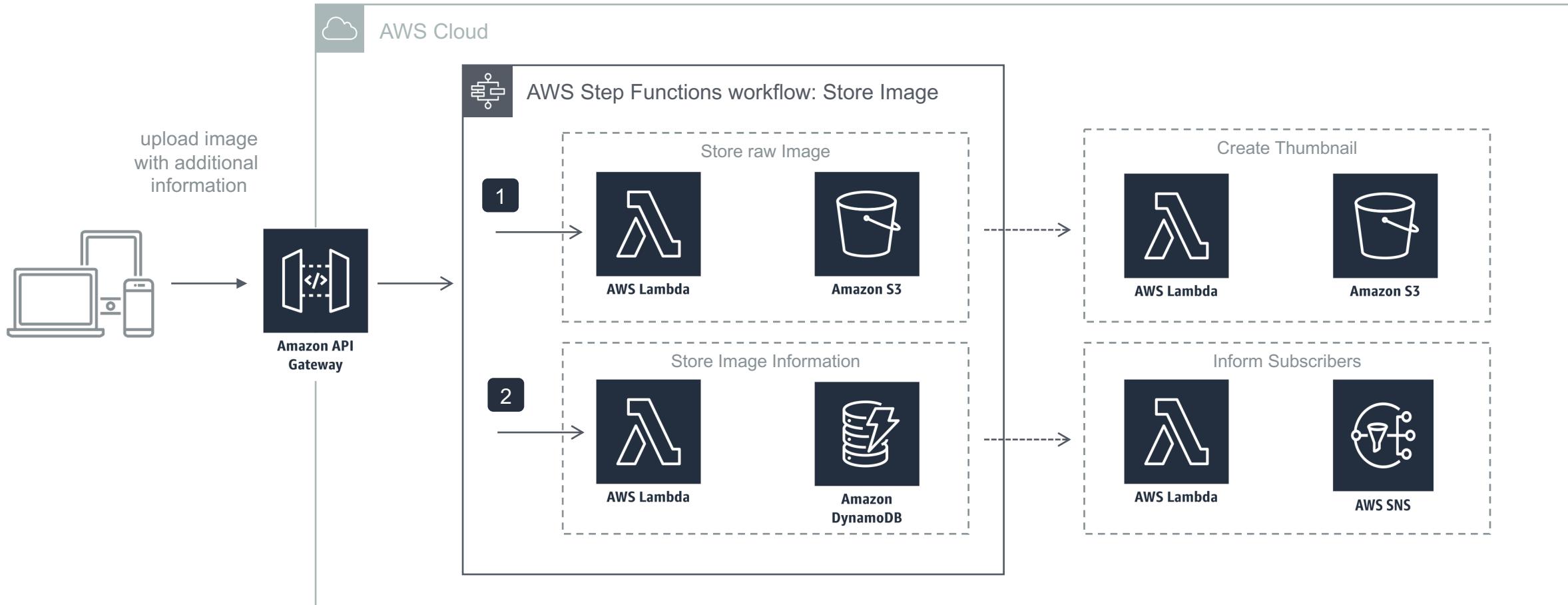


Upload Data & Image

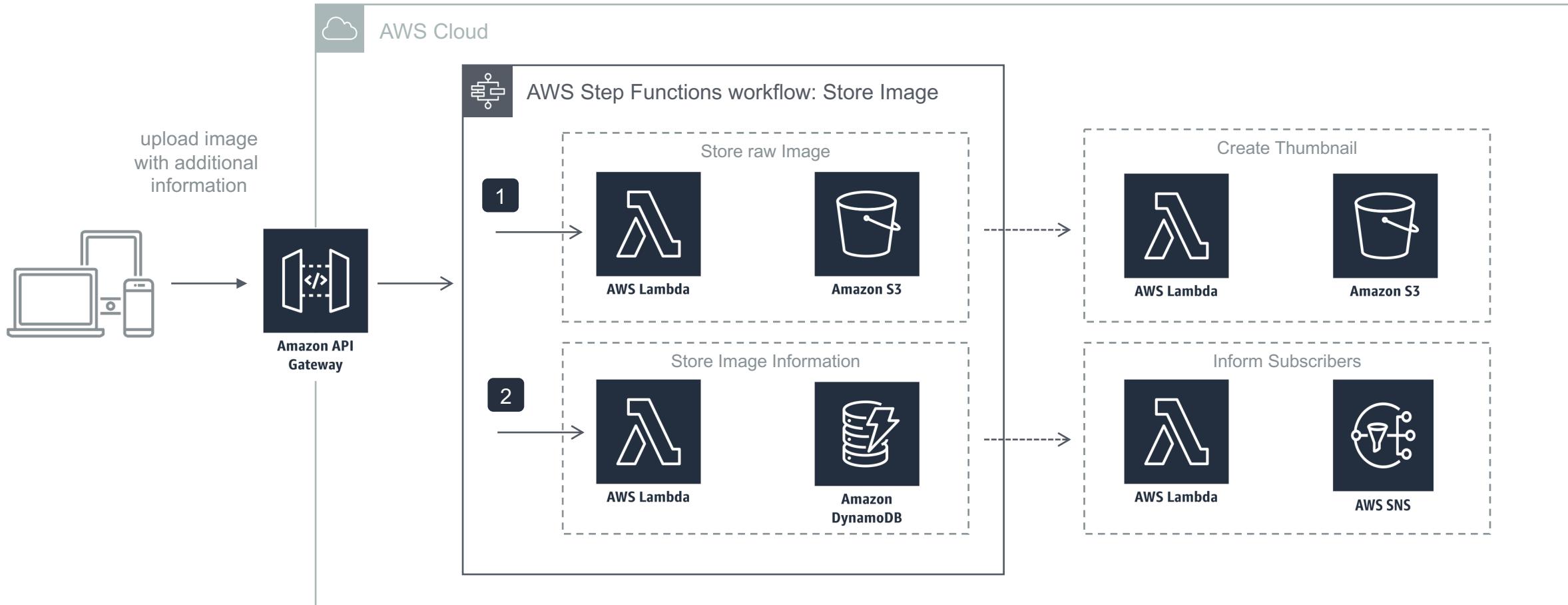


Amazon
Cognito

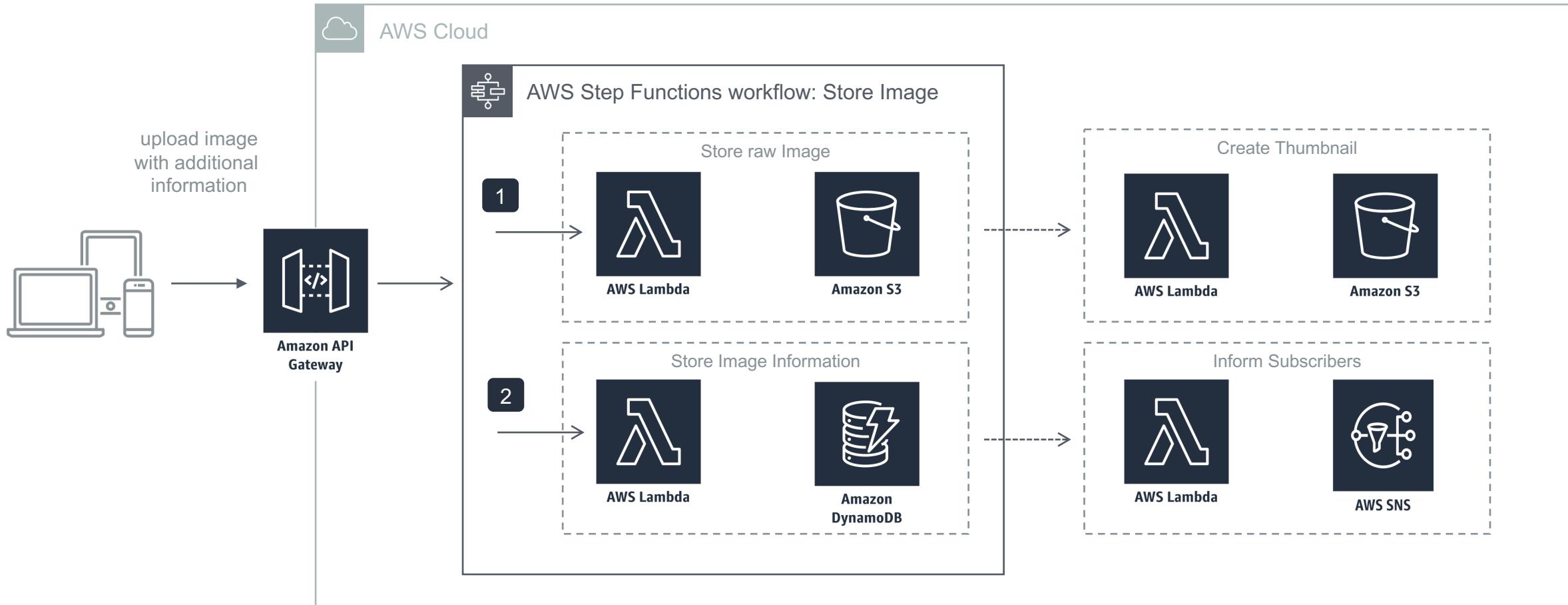
AWS Identity
and Access
Management



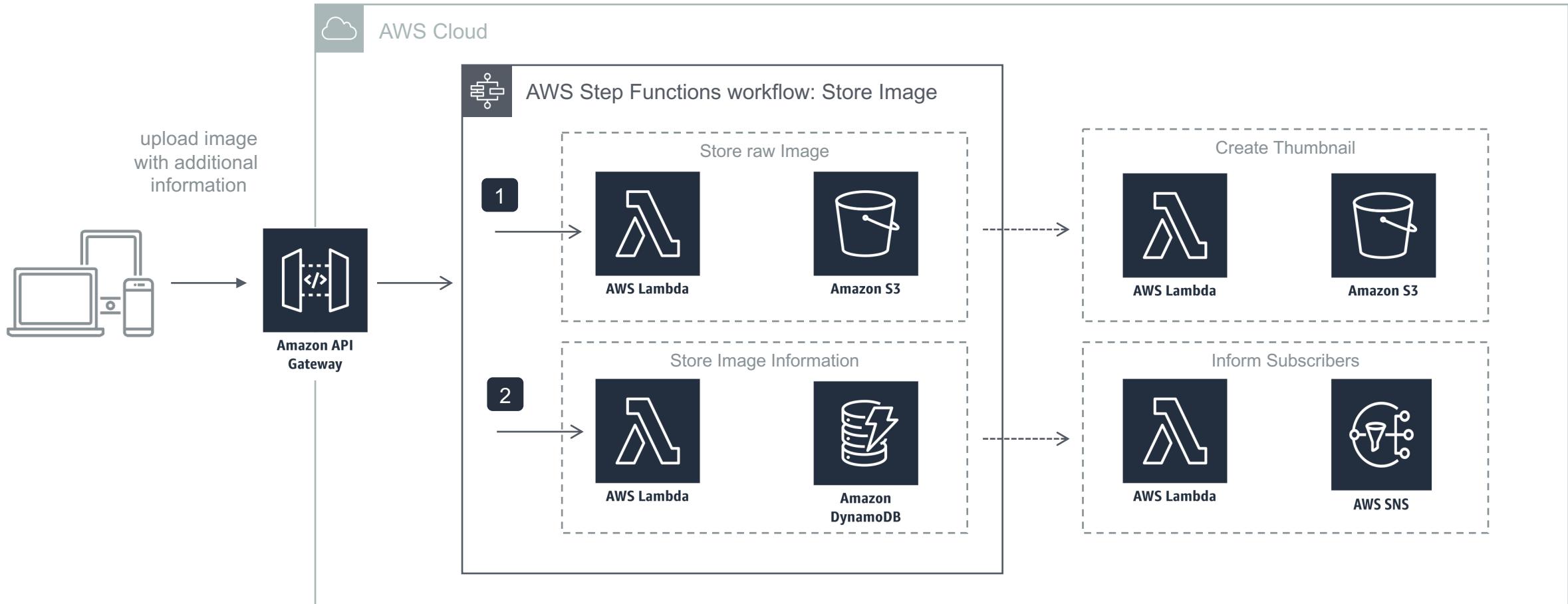
Upload Data & Image



Upload Data & Image



Upload Data & Image





NO SERVERS
NO STRESS

*„Was soll da schon
schiefgehen?“*



AWS
CloudFormation



Amazon
Route 53



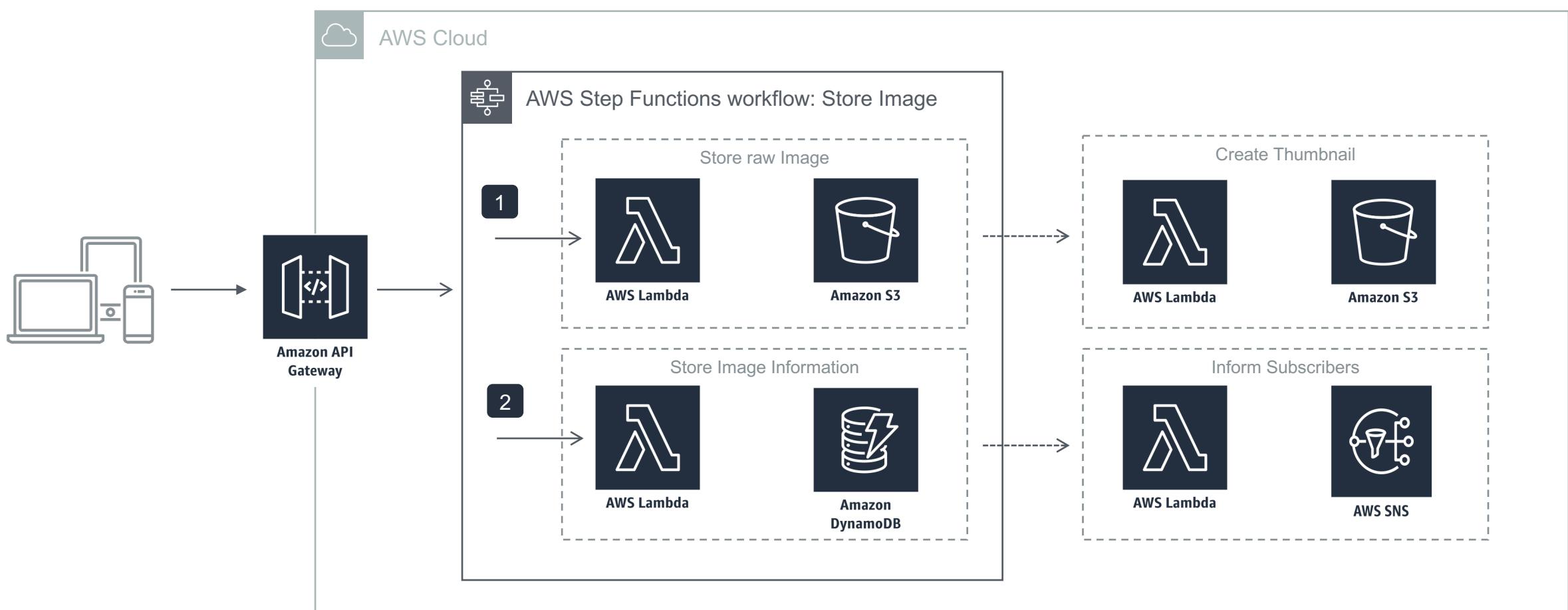
Amazon
CloudWatch



Amazon
Cognito



AWS Identity
and Access
Management





AWS
CloudFormation



Amazon
Route 53



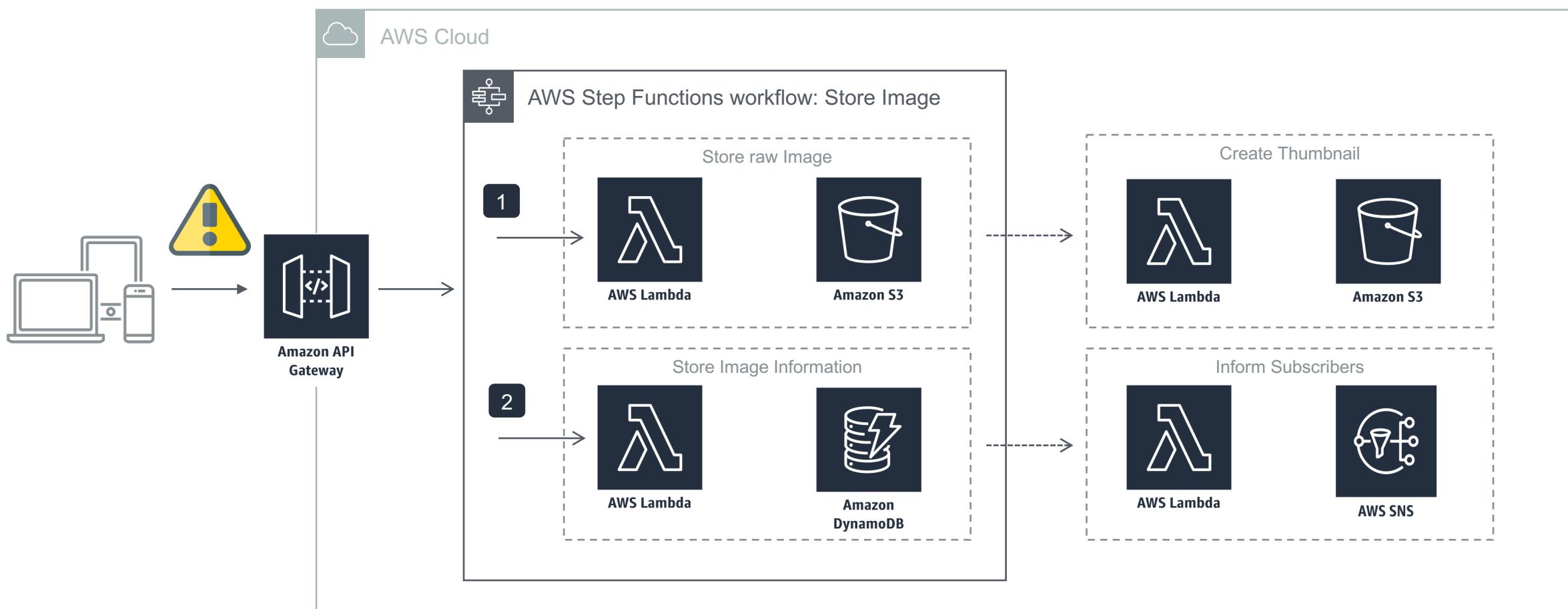
Amazon
CloudWatch



Amazon
Cognito



AWS Identity
and Access
Management





AWS
CloudFormation



Amazon
Route 53



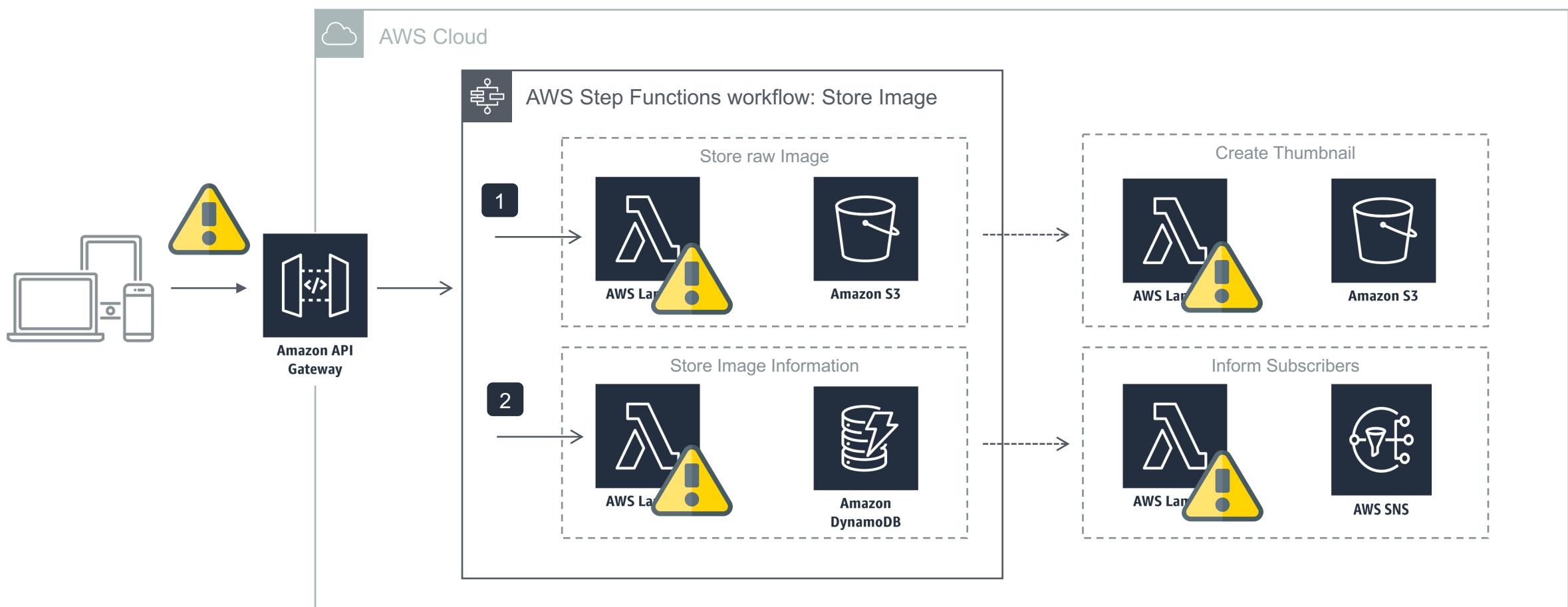
Amazon
CloudWatch



Amazon
Cognito



AWS Identity
and Access
Management





AWS
CloudFormation



Amazon
Route 53



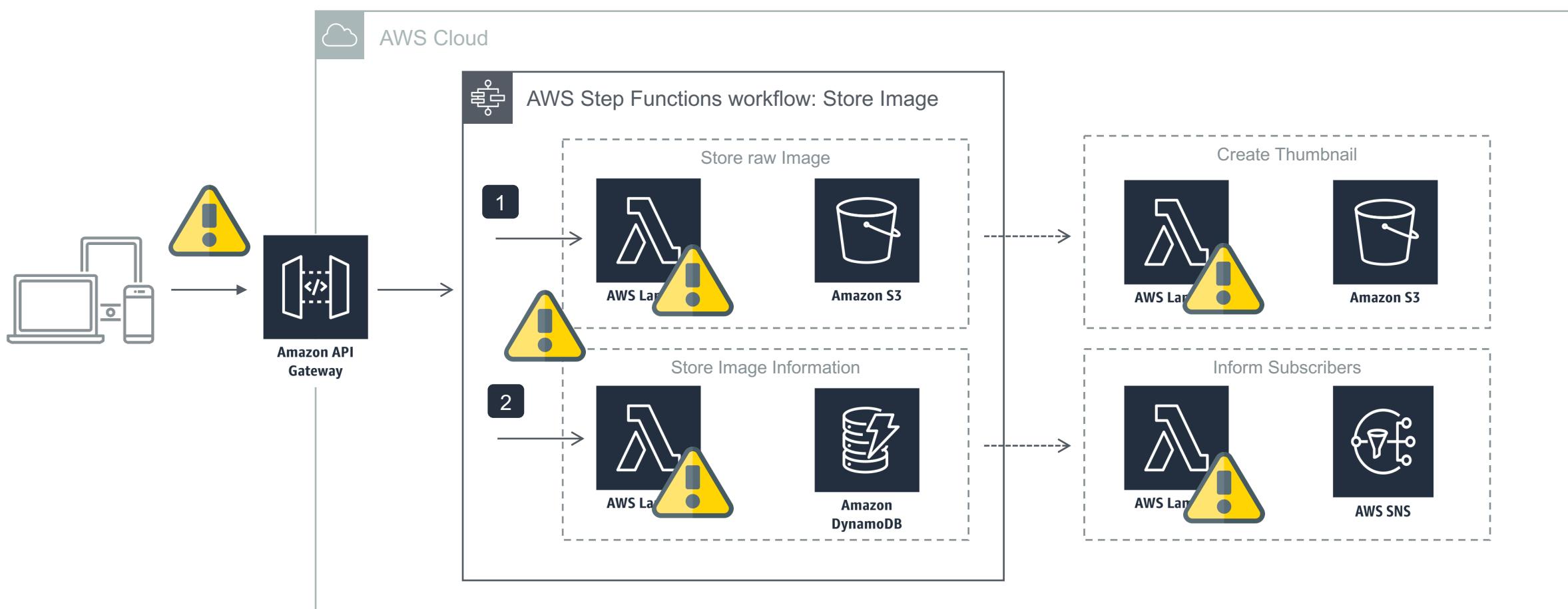
Amazon
CloudWatch



Amazon
Cognito



AWS Identity
and Access
Management





AWS
CloudFormation



Amazon
Route 53



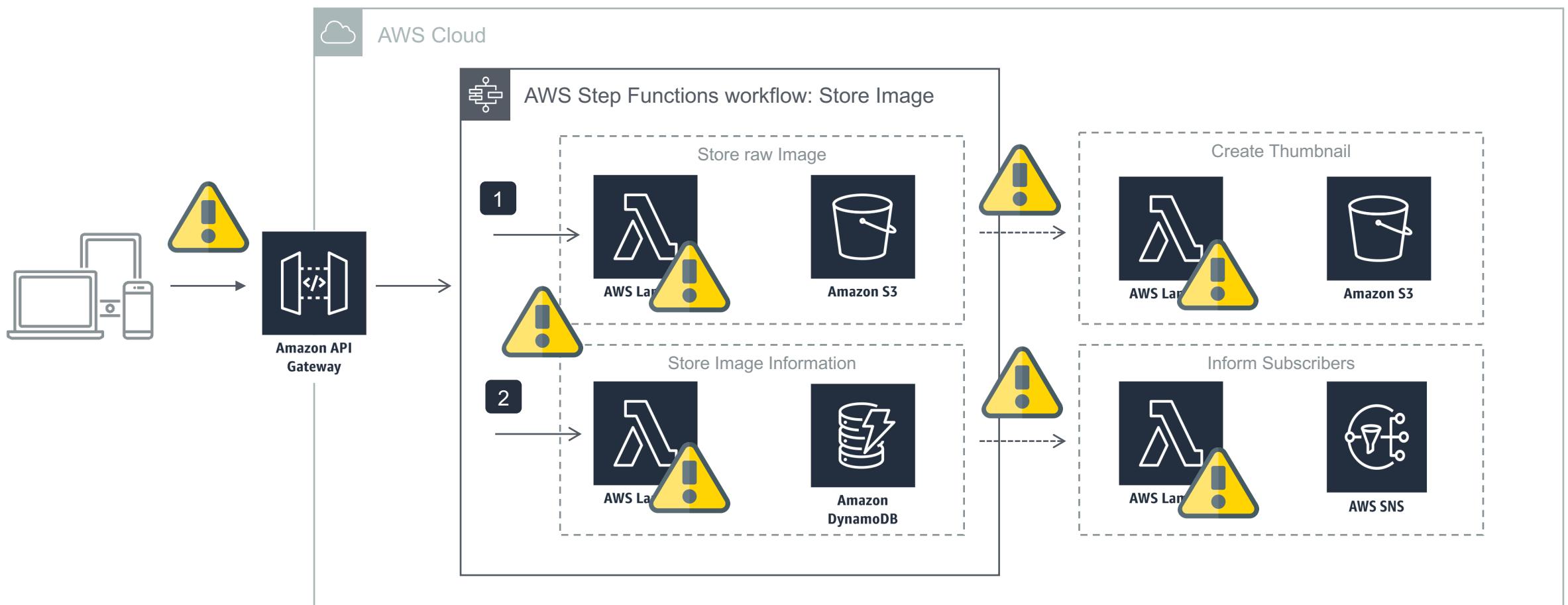
Amazon
CloudWatch



Amazon
Cognito



AWS Identity
and Access
Management





AWS
CloudFormation



Amazon
Route 53



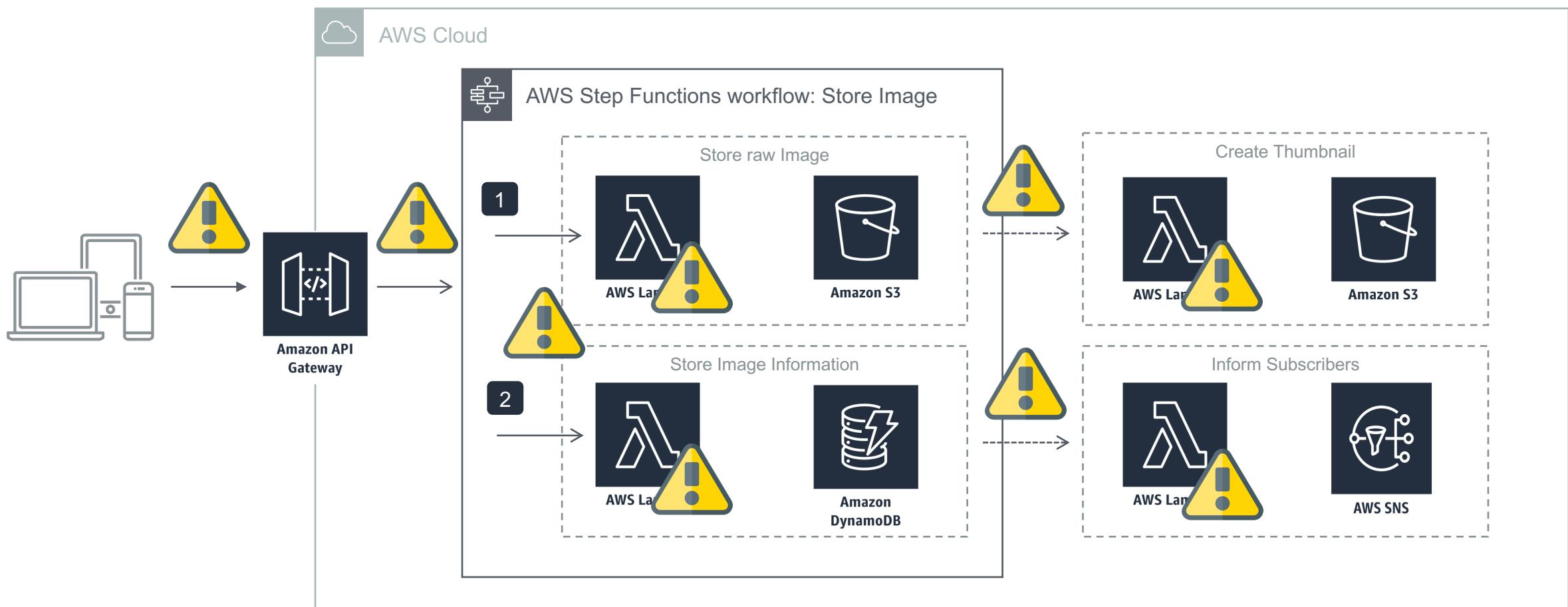
Amazon
CloudWatch



Amazon
Cognito



AWS Identity
and Access
Management





AWS
CloudFormation



Amazon
Route 53



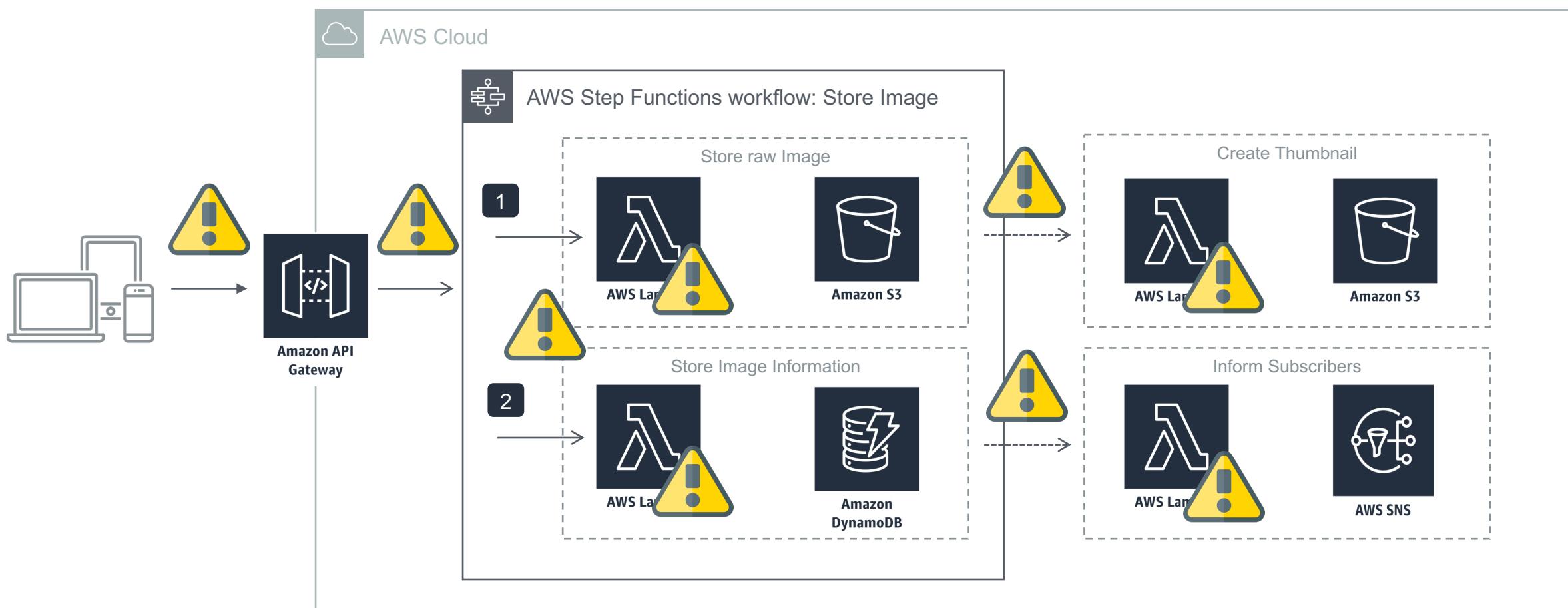
Amazon
CloudWatch



Amazon
Cognito



AWS
Identity
and Access
Management





AWS
CloudFormation



Amazon
Route 53



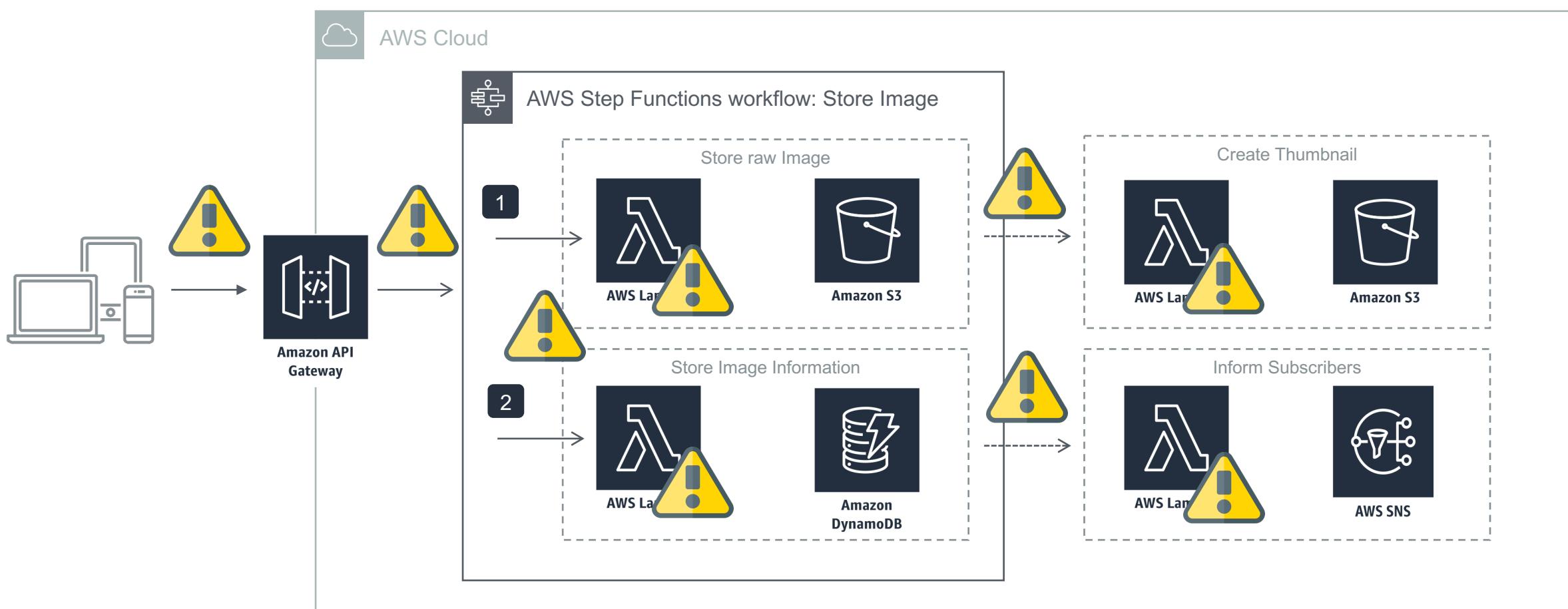
Amazon
CloudWatch



Amazon
Cognito



AWS Id
and
Access
Management





AWS
CloudFormation



Amazon
Route 53



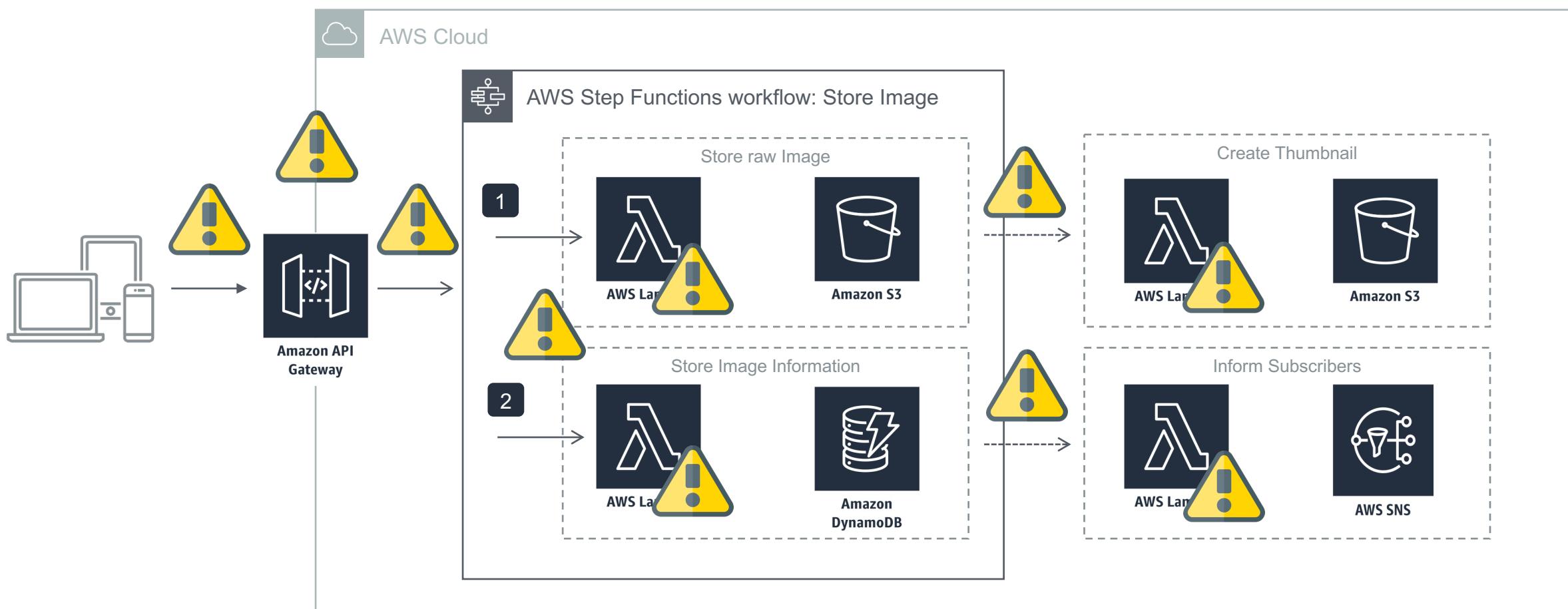
Amazon
CloudWatch

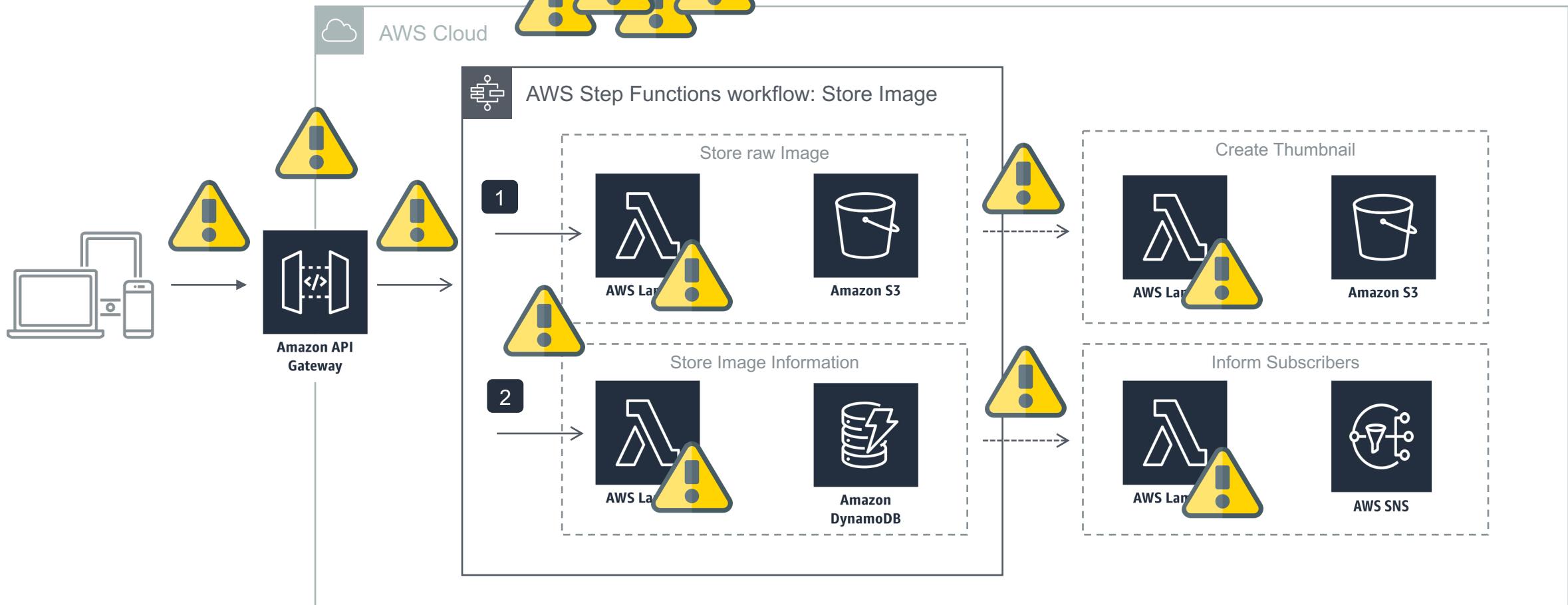


Amazon
Cognito



AWS Id
and
Access
Management





A photograph of a green ceramic mug filled with dark coffee. The mug is positioned in the upper left corner of the slide. A large amount of coffee has spilled onto the light-colored surface below it, creating a large puddle and several smaller droplets scattered around the base of the mug.

... schiefgehen?

- interne Programmierfehler
- Fehler in der Businesslogik
- unerwartete Calls / Payloads
- verlorene oder doppelte Events
- hohe Latenz & Timeouts
- Security Attacken (z.B. DoS/DDos)
- SLAs / Usage Plans
- Workload Peaks
- ...

„Run Code,
not Servers!“

**“Run your business code
highly-available
in the cloud in response
to events and scale
without any servers to
manage.”***

*(AWS Lambda product description)

“Run your business code
highly distributed
and **event driven** in a **non**
transparent environment
with **no single**
point of control.”*

*(my personal interpretation)



Wie teste ich? meine Serverless Application

Was, wann, wie und wo sollte ich testen, um ...

- **Vertrauen in meinen Code zu gewinnen**
- **das Risiko von Fehlern zu minimieren***

* vor allem in Produktion

Testen in der **Serverless** Welt

„The biggest complexity is not within the function itself, but in how it interacts with other functions and services (a.k.a. cloud components).“



Testing Best Practices

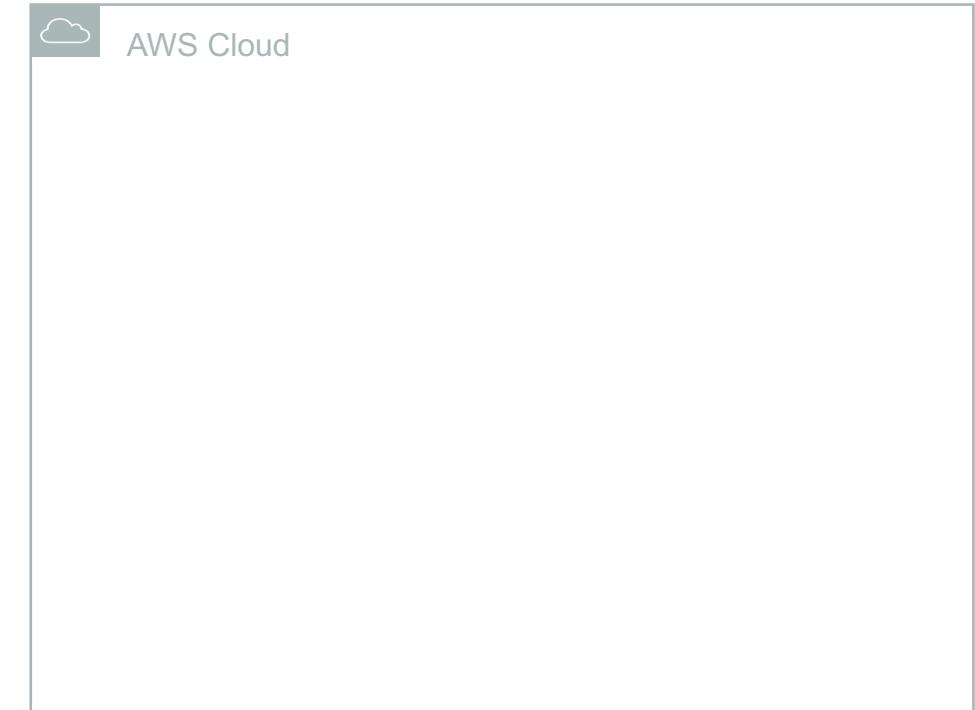
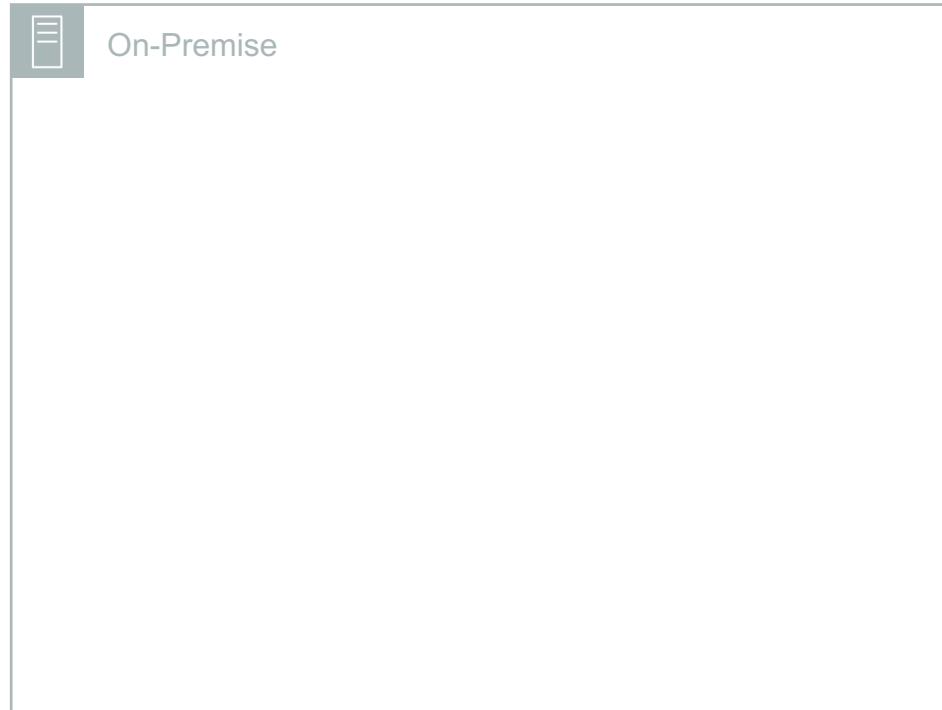
*„Welche Art von ‚Benchmarks‘ wollen wir für unser **Testing**?“*

- **funktionale Änderungen** schnell/kosteneffizient testen
- **integrative Änderungen** schnell/kosteneffizient testen
- **integrative Änderungen** so „real“ wie möglich testen
- **Use-Cases** und **User-Stories** so „real“ wie möglich testen

Testing Best Practices

- u Kandidat für Unit Tests
- i Kandidat für Integration Tests
- e Kandidat für End-to-End Tests

#1 Trennen von Businesslogik und Infrastruktur



Infrastructure

Business Logic

```
1 // AWS lambda handler generating a "greet" response
2 public GreetingResponse handleRequest(GreetingRequest name, Context context) {
3
4     LambdaLogger logger = context.getLogger();
5
6     // Print info from the context object
7     logger.log("Function name: " + context.getFunctionName());
8     logger.log(String.format("firstName: %s", name.getFirstName()));
9     logger.log(String.format("lastName: %s", name.getLastName()));
10
11    if (firstName != null && lastName != null) {
12        return new GreetingResponse(String.format("Hello, %s %s!",
13                                         firstName, lastName));
14    } else {
15        return new GreetingResponse(DEFAULT_GREET);
16    }
17 }
```

Infrastructure

Business Logic

```
1 // AWS lambda handler generating a "greet" response
2 public GreetingResponse handleRequest(GreetingRequest name, Context context) {
3
4     logContextInfo(context);
5
6     String greeting = GreetingService.greet(name.getFirstName(),
7                                              name.getLastName());
8     return new GreetingResponse(greeting);
9 }
10
11 // Print info from the context object
12 private void logContextInfo(Context context) {
13     LambdaLogger logger = context.getLogger();
14     logger.log("Function name: " + context.getFunctionName());
15 }
```

Business Logic Tests

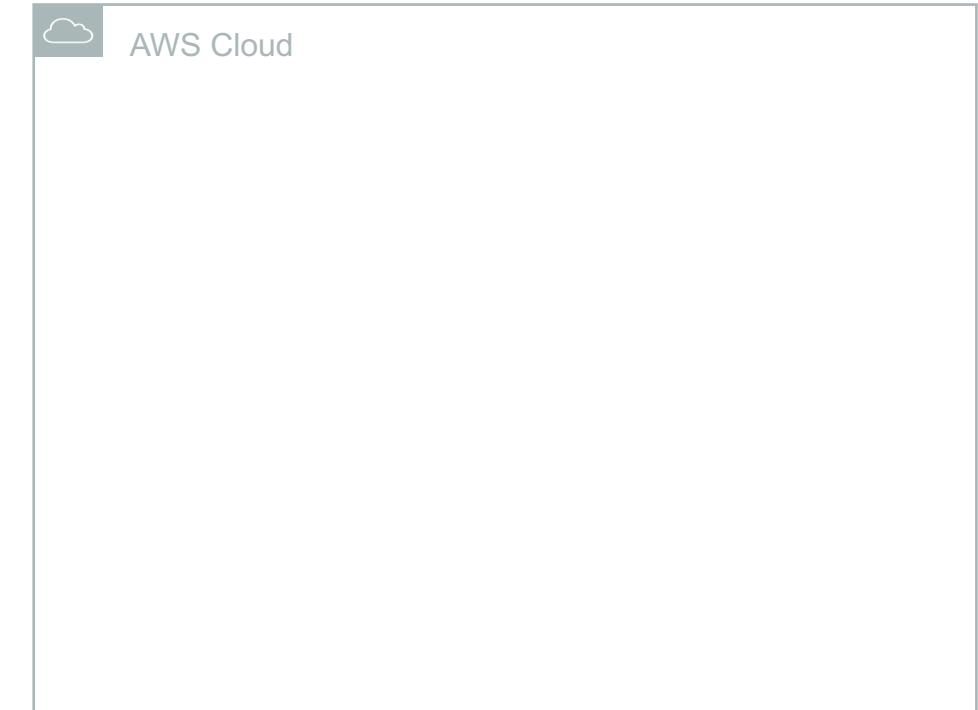
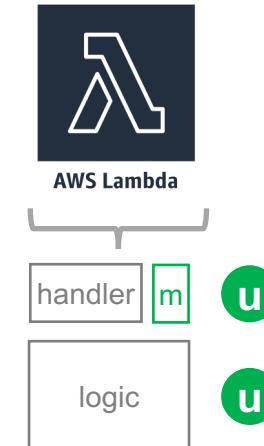
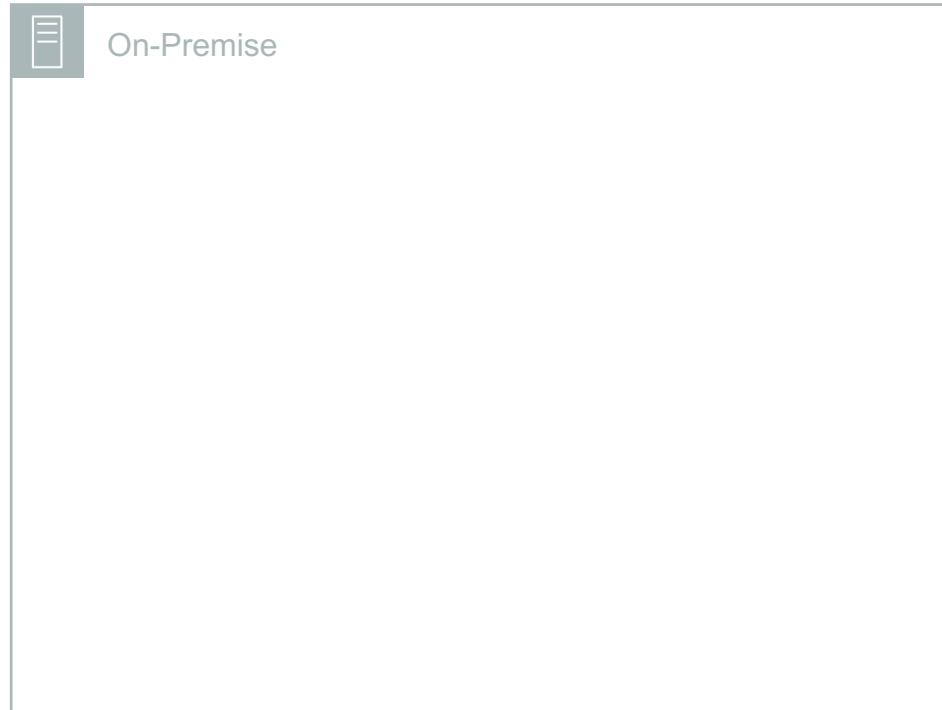


```
1 @Test
2 void whenGreetWithFirstNameAndLastName_thenCorrect() {
3
4     // given
5     String firstName = "Lars";
6     String lastName = "Roewekamp";
7
8     // when
9     String greetResult = GreetingService.greet(firstName, lastName);
10
11    // then
12    assertEquals("Hello, Lars Roewekamp! i am pleased to meet you.", greetResult);
13 }
14
15 @Test
16 void whenGreetWithoutFirstName_thenInCorrect() { ... }
17
18 @Test
19 void whenGreetWithoutLastName_thenInCorrect() { ... }
```

Testing Best Practices

- u Kandidat für Unit Tests
- i Kandidat für Integration Tests
- e Kandidat für End-to-End Tests

#2 Cloud-Infrastruktur Komponenten mocken



Infrastructure

```
1 // AWS lambda handler generating a "greet" response
2 public GreetingResponse handleRequest(GreetingRequest name, Context context) {
3
4     logContextInfo(context);
5
6     String greeting = GreetingService.greet(name.getFirstName(),
7                                              name.getLastName());
8     return new GreetingResponse(greeting);
9 }
10
11 // Print info from the context object
12 private void logContextInfo(Context context) {
13     LambdaLogger logger = context.getLogger();
14     logger.log("Function name: " + context.getFunctionName());
15 }
```

AWS agnostic logger (log4j)

```
1 // initialize the log4j logger
2 static final Logger logger = LogManager.getLogger(GreetingHandler.class)
3
4 /
5 p
6
7     1 // set log level depending on the environment variable "logLevel"
8     2 private void initLogger(Context context) {
9         3     String logLevel = LambdaEnvironment.getEnvVarAsString("logLevel");
10        4     Configurator.setAllLevels(LogManager.getRootLogger().getName(),
11          5                           Level.toLevel(logLevel));
12     6 }
13
14 /
15 private void logContextInfo(Context context) {
16     logger.debug("Function name: %s", context.getFunctionName());
17 }
```

fake infrastructure component (Context)

```
1 @Test
2 void whenHandleRequestWithFirstNameAndLastName_thenCorrect() {
3
4     // given
5     String firstName = "Lars";
6     String lastName = "Roewekamp";
7     GreetingRequest request = new GreetingRequest(firstName, lastName);
8     GreetingRequestHandler clazzUnderTest = new GreetingRequestHandler();
9     Context context = new TestContext() { ... };
10
11    // when
12    GreetingResponse response = clazzUnderTest.handleRequest(request, context);
13
14    // then
15    assertEquals("Hello, Lars Roewekamp! i am pleased to meet you.",
16                 response.getGreeting());
17 }
```

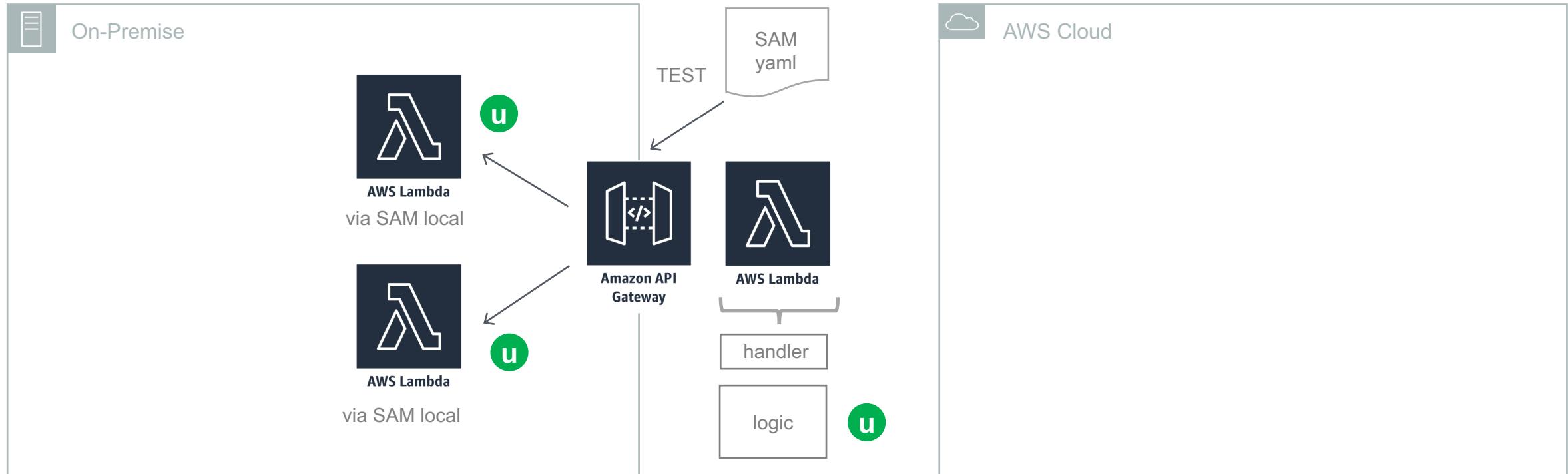
mock infrastructure component (Context)

```
1 @BeforeEach
2 void setUp() {
3     // mock AWS lambda context with Mockito
4     mockContext = mock(Context.class);
5
6     // define expectations when calling the mock (when ... then ...)
7     when(mockContext.getFunctionName()).thenReturn("handleRequest");
8     when(mockContext.getMemoryLimitInMB()).thenReturn(100);
9     when(mockContext.getMemoryLimitInMB()).thenReturn(50);
10 }
11 }
```

Testing Best Practices

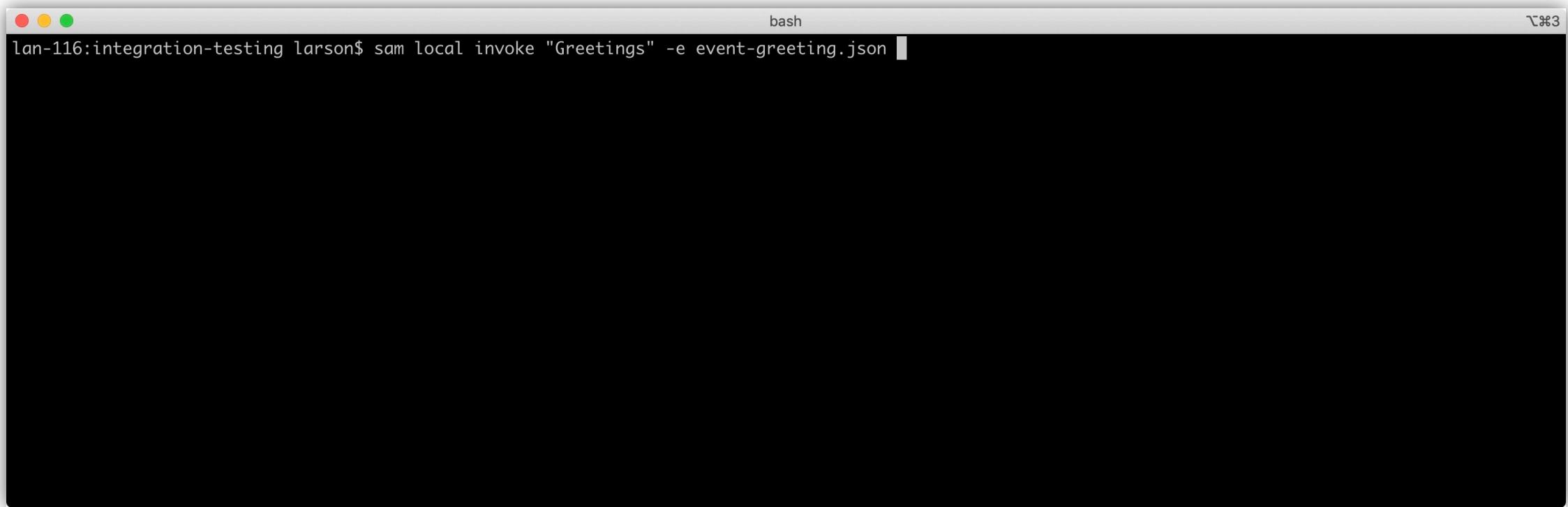
- u Kandidat für Unit Tests
- i Kandidat für Integration Tests
- e Kandidat für End-to-End Tests

#3 Lokale Umgebung für funktionale Tests verwenden (z.B. SAM local)





```
1 Description: AWS Lambda Greeting Service
2 Resources:
3   Greetings:
4     Type: AWS::Serverless::Function
5     Properties:
6       Handler: de.openknowledge.workshop.cloud.serverless.lambda.GreetingHandler
7       CodeUri: ./target/integration-testing-1.0-SNAPSHOT.jar
8       Runtime: java8
9       Timeout: 300
10      Environment:
11        Variables:
12          ENVIRONMENT: "test"
13      Events:
14        CreateResource:
15          Type: Api
16          Properties:
17            Path: /greetings/
18            Method: post
```



A screenshot of a terminal window titled 'bash'. The command 'sam local invoke "Greetings" -e event-greeting.json' is entered at the prompt. The terminal has a dark background and light-colored text.

\$ **sam local invoke "Greetings" -e event-greeting.json --env-vars env.json**



function name



payload for function

```
lan-116:integration-testing larson$ sam local invoke "Greetings" -e event-greeting.json
2019-10-21 11:33:52 Found credentials in shared credentials file: ~/.aws/credentials
2019-10-21 11:33:52 Invoking de.openknowledge.workshop.cloud.serverless.lambda.GreetingHandler (java8)
2019-10-21 11:33:52 Decompressing /Volumes/work/conferences/2019/serverless/se_serverless_testing/code/integration-testing/target/integration-testing-1.0-SNAPSHOT.jar

Fetching lambci/lambda:java8 Docker container image.....
2019-10-21 11:33:54 Mounting /private/var/folders/dx/06pnkls14jx15lyyzllgxy340000gn/T/tmp9r_mff97 as /var/task:ro inside runtime container
START RequestId: 7fc54ea3-9279-4947-81be-599461ec88b0 Version: $LATEST
END RequestId: 7fc54ea3-9279-4947-81be-599461ec88b0
REPORT RequestId: 7fc54ea3-9279-4947-81be-599461ec88b0 Duration: 85.26 ms Billed Duration: 100 ms Memory Size: 128 MB Max Memory Used: 5 MB

{"greeting":"Hello, Lars Roewekamp! i am pleased to meet you."}
lan-116:integration-testing larson$
```

\$ **sam local invoke "Greetings" -e event-greeting.json --env-vars env.json**



function name

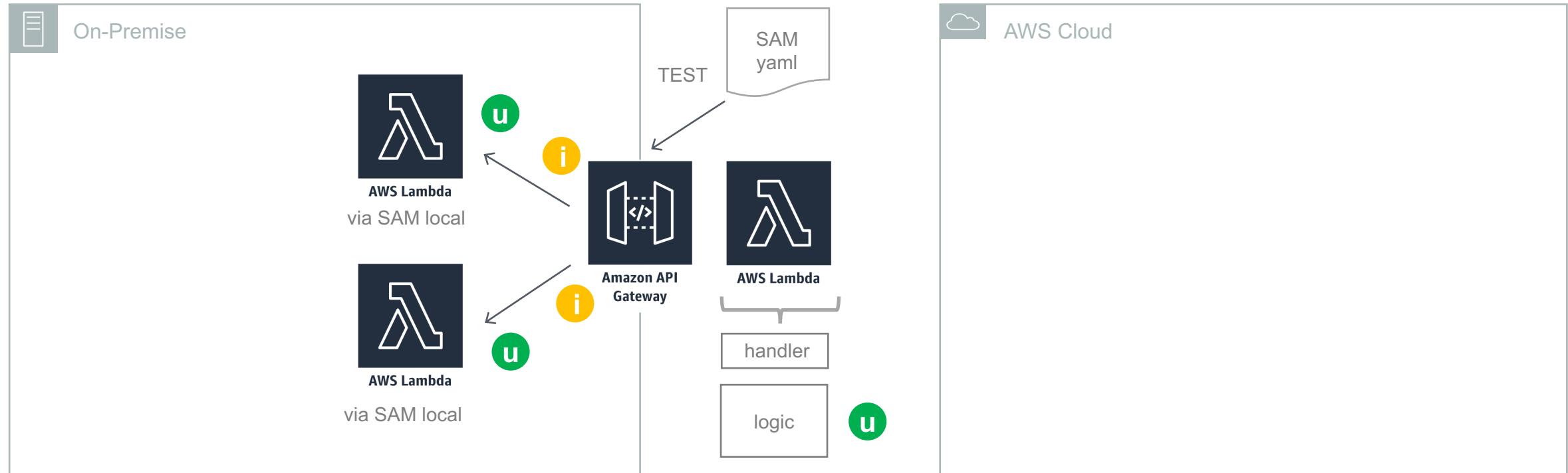


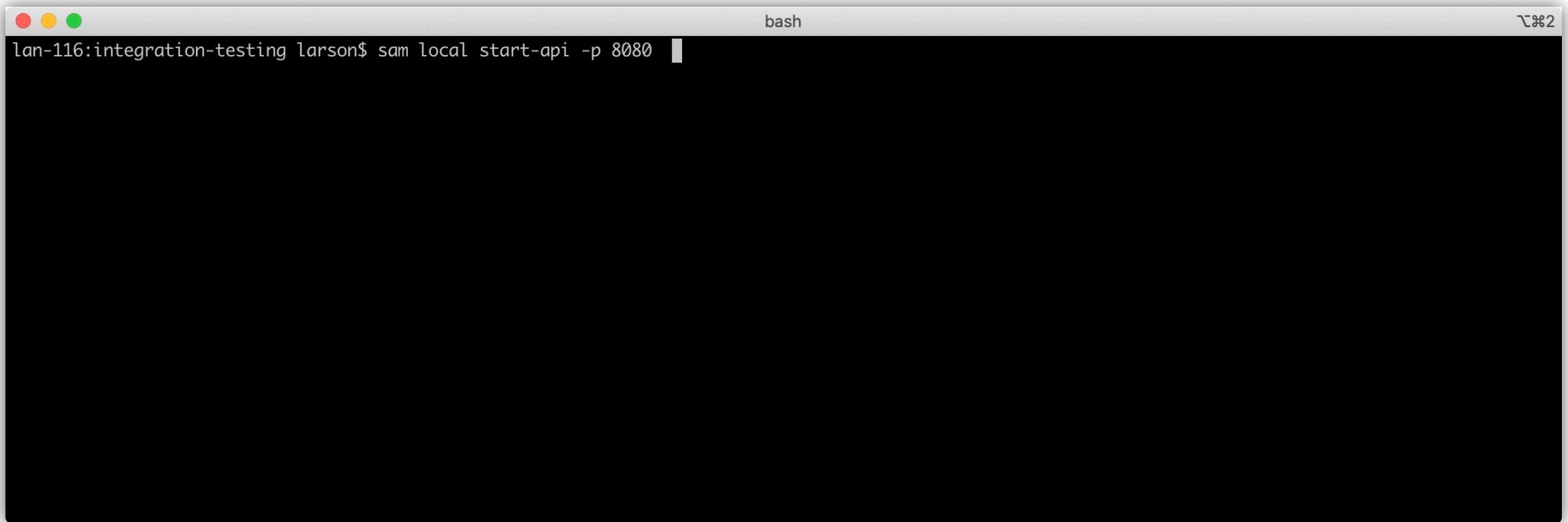
payload for function

Testing Best Practices

- u Kandidat für Unit Tests
- i Kandidat für Integration Tests
- e Kandidat für End-to-End Tests

#4 Lokale Umgebung zum Triggern von Integration Tests verwenden





```
lan-116:integration-testing larson$ sam local start-api -p 8080
```

\$ sam local start-api –p 8080

lan-116:integration-t
Mounting Greetings at
You can now browse to
will be reflected inst
2019-10-21 16:03:14

POST http://127.0.0.1:8080/greetings

Untitled Request

Comments (0)

POST http://127.0.0.1:8080/greetings Send Save

Params Authorization Headers (10) Body **JSON** BETA Beautify

none form-data x-www-form-urlencoded raw binary GraphQL BETA

```
1 {  
2   "firstName" : "Lars",  
3   "lastName" : "Roewekamp"  
4 }
```

Body Cookies Headers (4) Test Results Status: 200 OK Time: 9.34s Size: 209 B Save Response

Pretty Raw Preview Visualize BETA JSON

```
1 {  
2   "greeting": "Hello, Lars Roewekamp! i am pleased to meet you."  
3 }
```

Bootcamp

The screenshot shows the Postman application interface. A modal window is open in the foreground, displaying a terminal log entry:

```
lan-116:integration-t
Mounting Greetings at
You can now browse to
will be reflected inst
2019-10-21 16:03:14
```

Below the modal, the main Postman window is visible. It shows an 'Untitled Request' for a POST method to the URL `http://127.0.0.1:8080/greetings`. The 'Body' tab is selected, containing the following JSON payload:

```
1 {  
2   "firstName" : "Lars",  
3   "lastName" : "Roewekamp"  
4 }
```

The response status is 200 OK, and the response body is:

```
1 {  
2   "greeting": "Hello, Lars Roewekamp! i am pleased to meet you."  
3 }
```

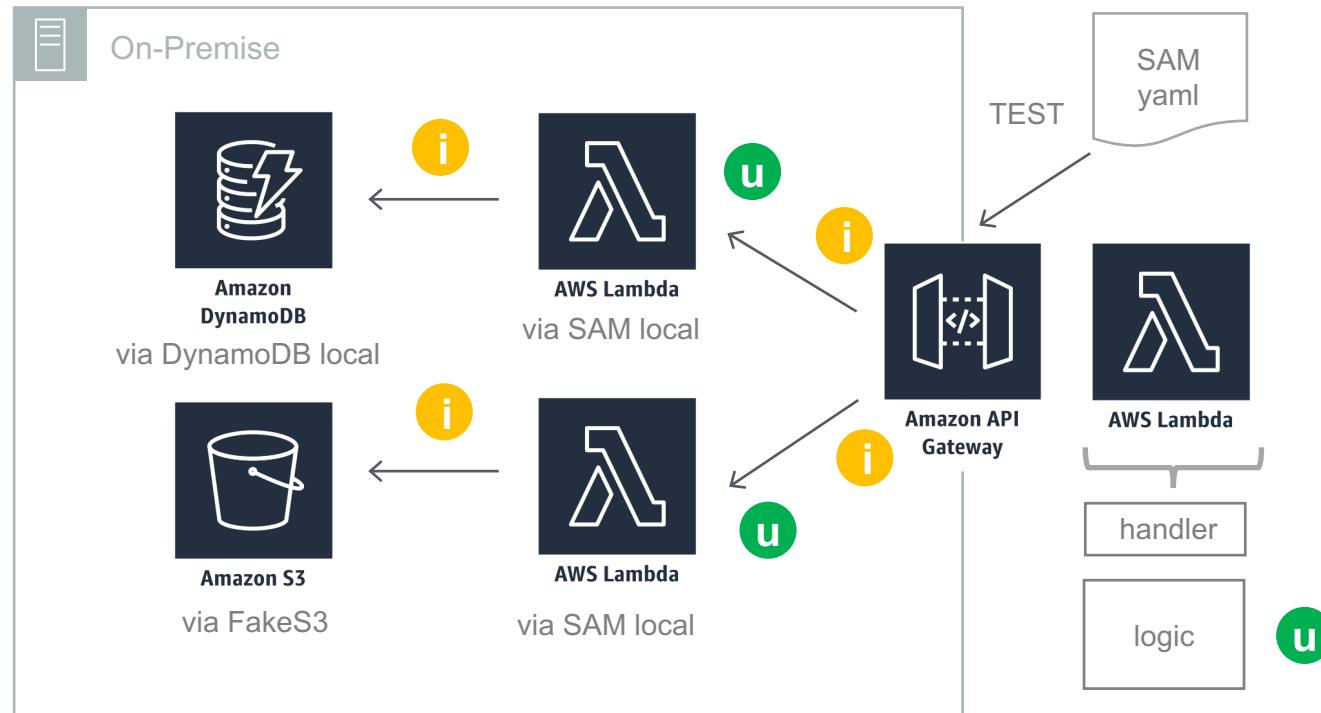
\$ sam lo

```
lan-116:integrati python3.7
Mounting Greetin
You can now brow
ill be reflected
2019-10-21 16:03
2019-10-21 16:03
Invoking de.open
2019-10-21 16:03
Decompressing /V
Fetching lambci/
Mounting /privat
START RequestId:
END RequestId: 2
REPORT RequestId
No Content-Type
2019-10-21 16:03
$ sam local start-api -p 3000
1 {
2   "body": "{ \"firstName\": \"Lars\", \"lastName\": \"Roewekamp\"}",
3   "resource": "/{proxy+}",
4   "path": "/greetings",
5   "httpMethod": "POST",
6   "isBase64Encoded": true,
7   "queryStringParameters": {
8     "foo": "bar"
9   },
10 ...
11 }
12
```

Testing Best Practices

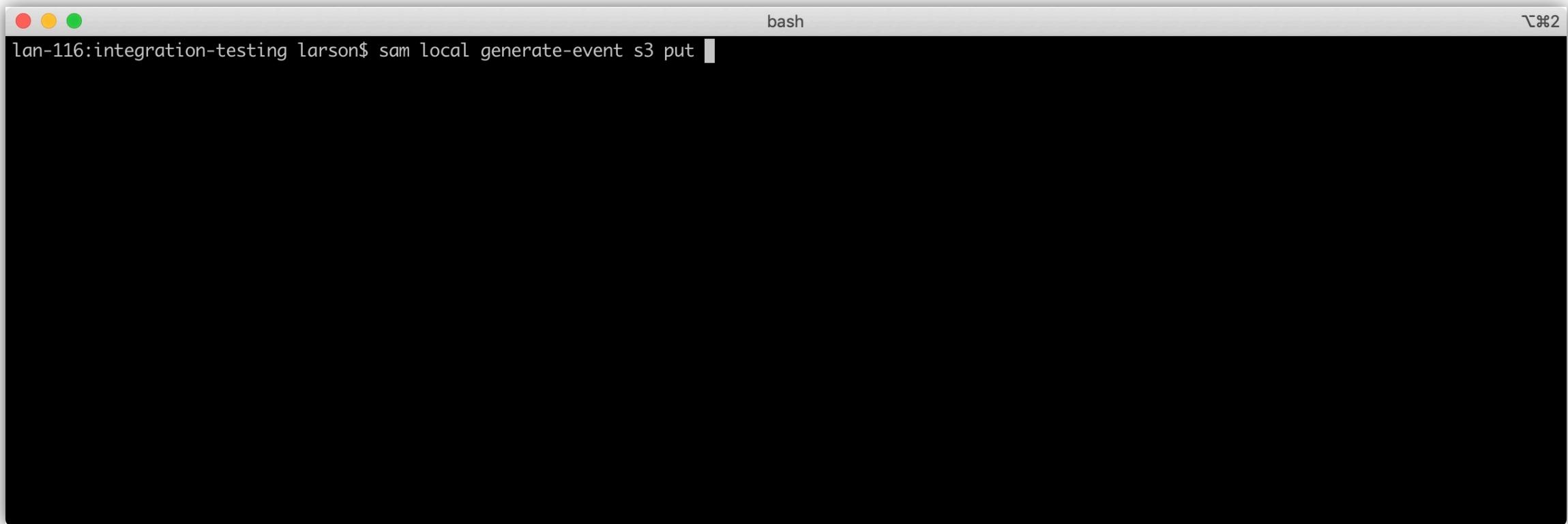
- u Kandidat für Unit Tests
- i Kandidat für Integration Tests
- e Kandidat für End-to-End Tests

#5 Lokale Cloud-Komponenten für Integration Tests*



WARNING: lokale Cloud Komponenten können lediglich funktionale Korrektheit sicherstellen, nicht aber infrastrukturelle, wie z.B. DLQs, Timeouts, Throttling, SLAs, ...

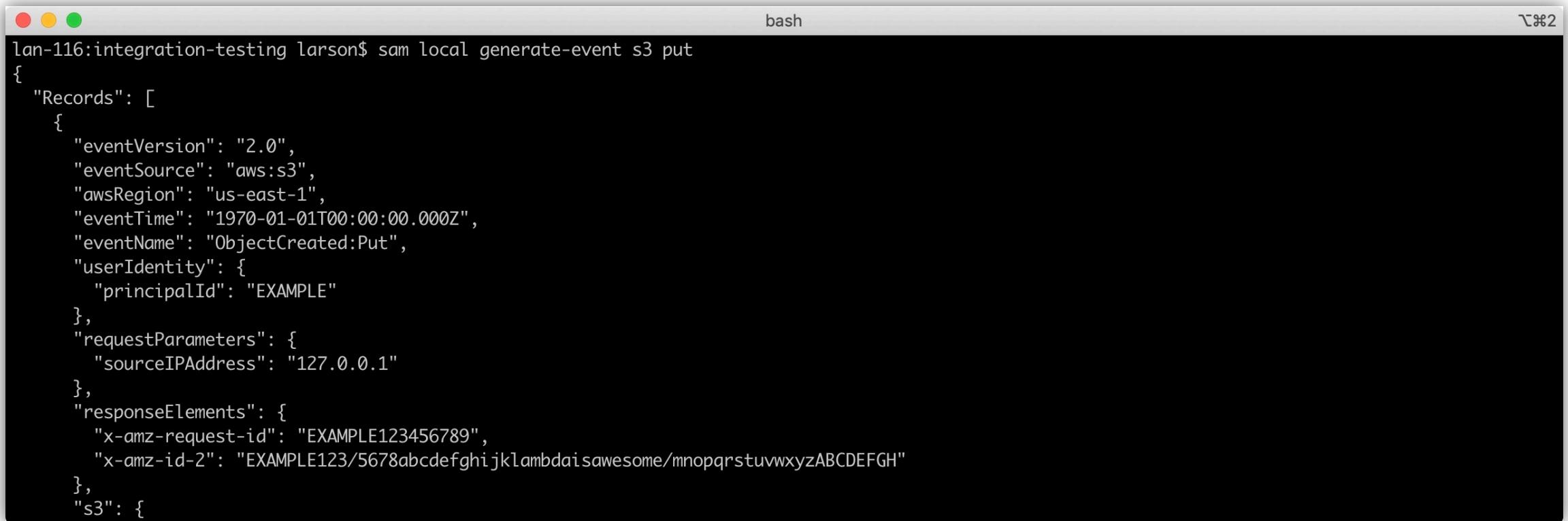
Simulate Component Event to trigger Lambda

A screenshot of a macOS terminal window titled "bash". The window has red, yellow, and green close buttons in the top-left corner and a "⌘2" icon in the top-right corner. The terminal prompt is "lan-116:integration-testing larson\$". Below the prompt, the command "sam local generate-event s3 put" is typed and followed by a cursor. The rest of the terminal window is blank black space.

```
lan-116:integration-testing larson$ sam local generate-event s3 put
```

\$ **sam local generate-event [SERVICE] [OPTION]**

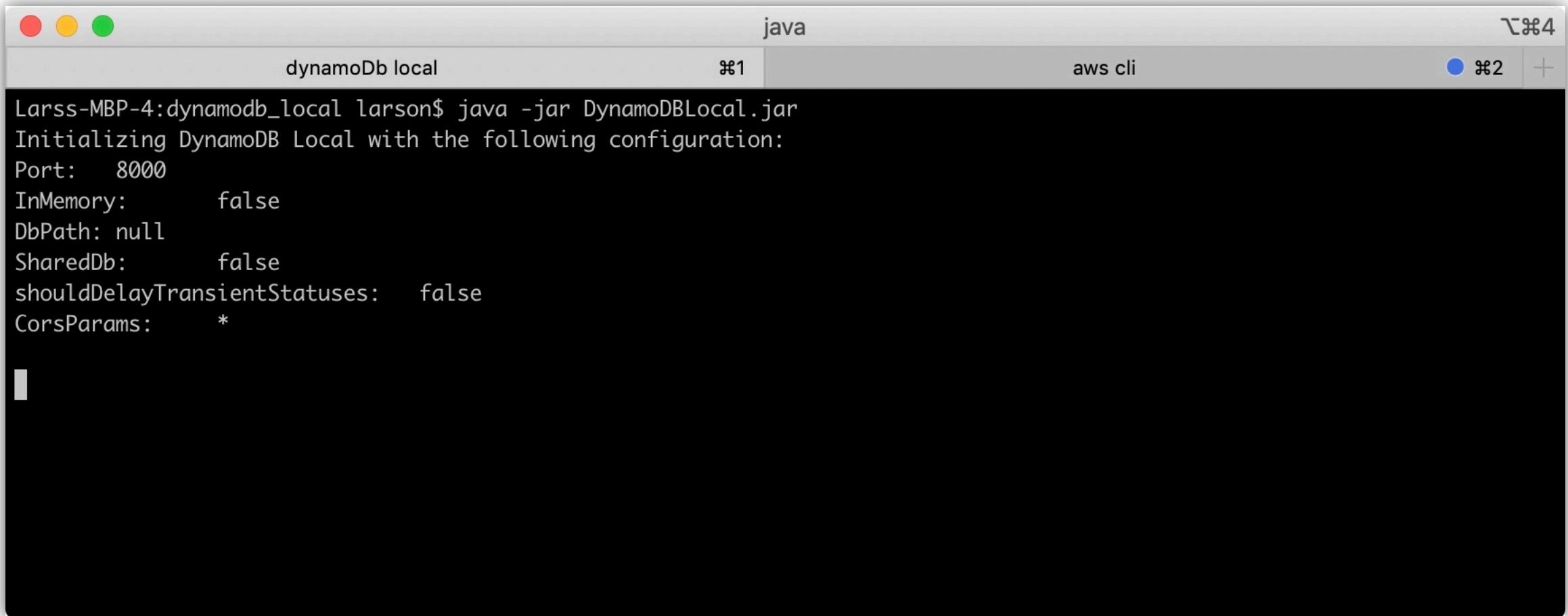
Simulate Component Event to trigger Lambda



```
lan-116:integration-testing larson$ sam local generate-event s3 put
{
  "Records": [
    {
      "eventVersion": "2.0",
      "eventSource": "aws:s3",
      "awsRegion": "us-east-1",
      "eventTime": "1970-01-01T00:00:00.000Z",
      "eventName": "ObjectCreated:Put",
      "userIdentity": {
        "principalId": "EXAMPLE"
      },
      "requestParameters": {
        "sourceIPAddress": "127.0.0.1"
      },
      "responseElements": {
        "x-amz-request-id": "EXAMPLE123456789",
        "x-amz-id-2": "EXAMPLE123/5678abcdefghijklambdaisawesome/mnopqrstuvwxyzABCDEFGH"
      },
      "s3": {
```

\$ **sam local generate-event [SERVICE] [OPTION]**

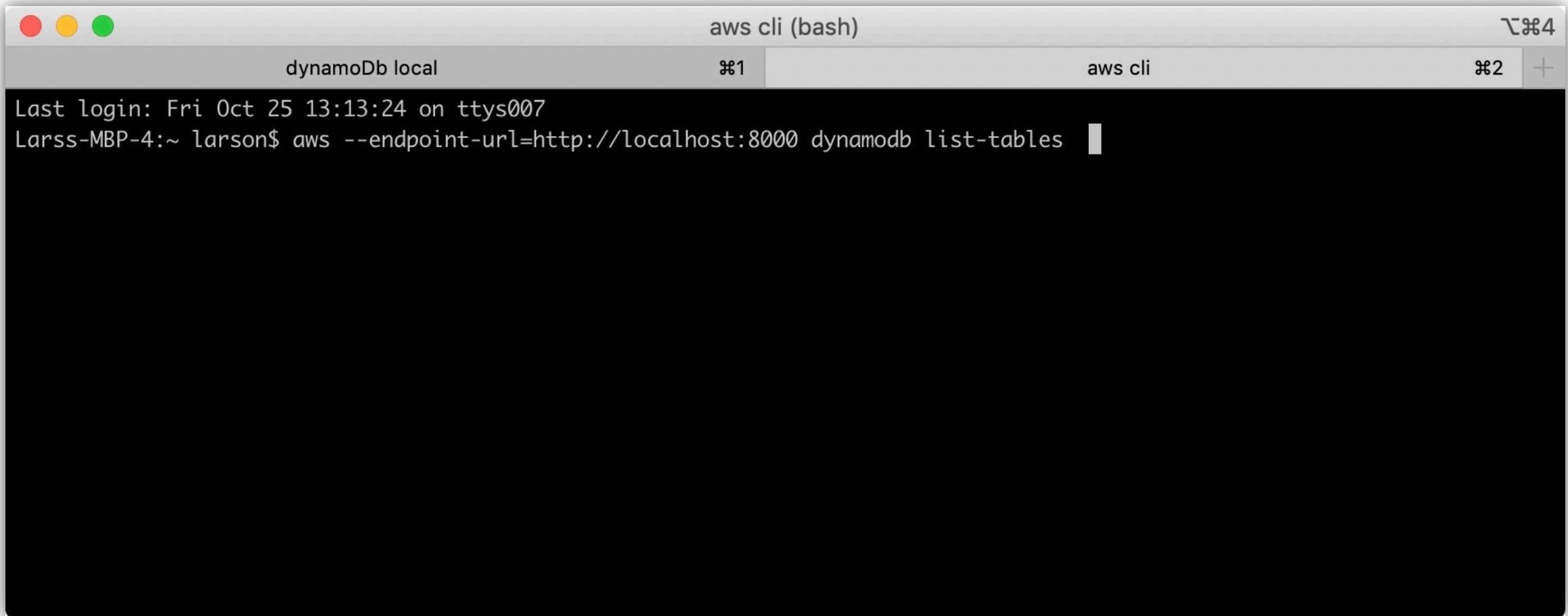
Simulate Component triggered by Lambda



Larss-MBP-4:dynamodb_local larson\$ java -jar DynamoDBLocal.jar
Initializing DynamoDB Local with the following configuration:
Port: 8000
InMemory: false
DbPath: null
SharedDb: false
shouldDelayTransientStatuses: false
CorsParams: *

\$ **java –jar DynamoDBLocal.jar**

Simulate Component triggered by Lambda

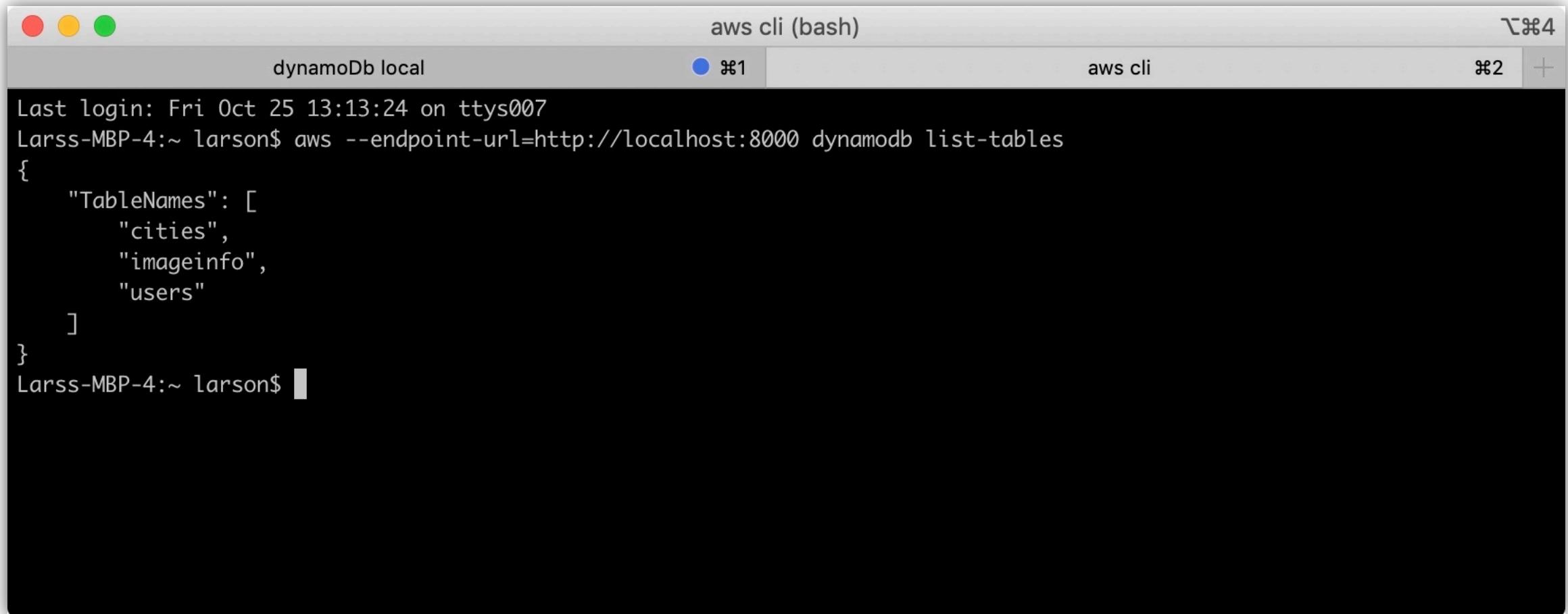


The screenshot shows a macOS terminal window with the title bar "aws cli (bash)". The window has three tabs: "dynamoDb local" (selected), "#1", and "aws cli". The terminal output shows:

```
Last login: Fri Oct 25 13:13:24 on ttys007
Larss-MBP-4:~ larson$ aws --endpoint-url=http://localhost:8000 dynamodb list-tables
```

\$ **aws --endpoint-url=http://localhost:8000 dynamodb list-tables**

Simulate Component triggered by Lambda



The screenshot shows a macOS terminal window with the title bar "aws cli (bash)". The window contains the following text:

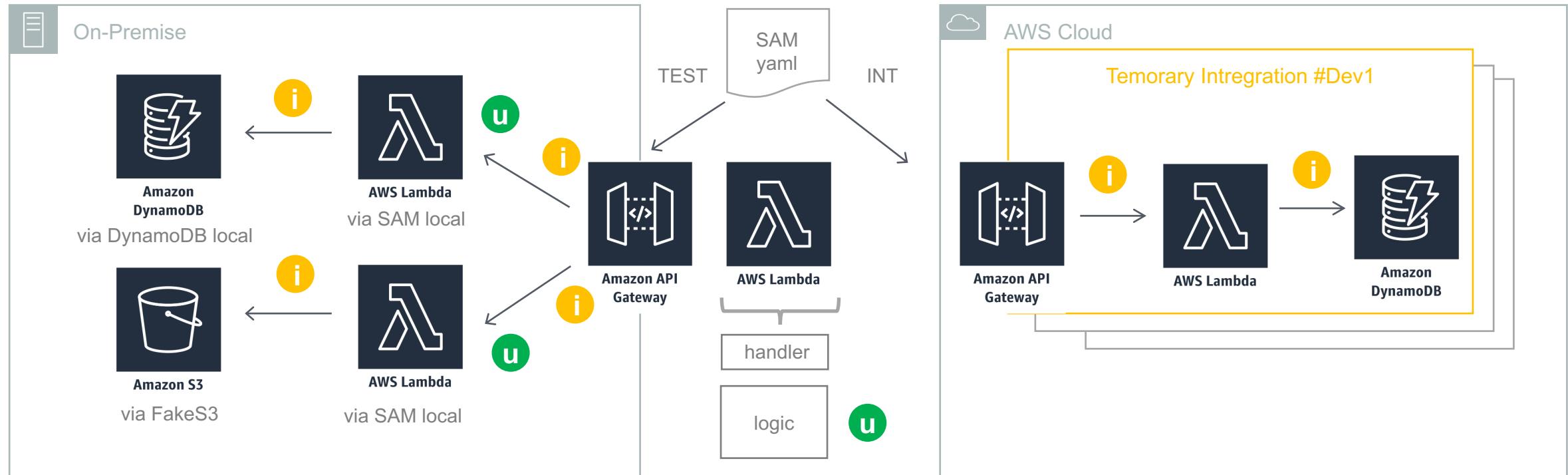
```
Last login: Fri Oct 25 13:13:24 on ttys007
Larss-MBP-4:~ larson$ aws --endpoint-url=http://localhost:8000 dynamodb list-tables
{
  "TableNames": [
    "cities",
    "imageinfo",
    "users"
  ]
}
Larss-MBP-4:~ larson$
```

\$ aws --endpoint-url=http://localhost:8000 dynamodb list-tables

Testing Best Practices

- u Kandidat für Unit Tests
- i Kandidat für Integration Tests
- e Kandidat für End-to-End Tests

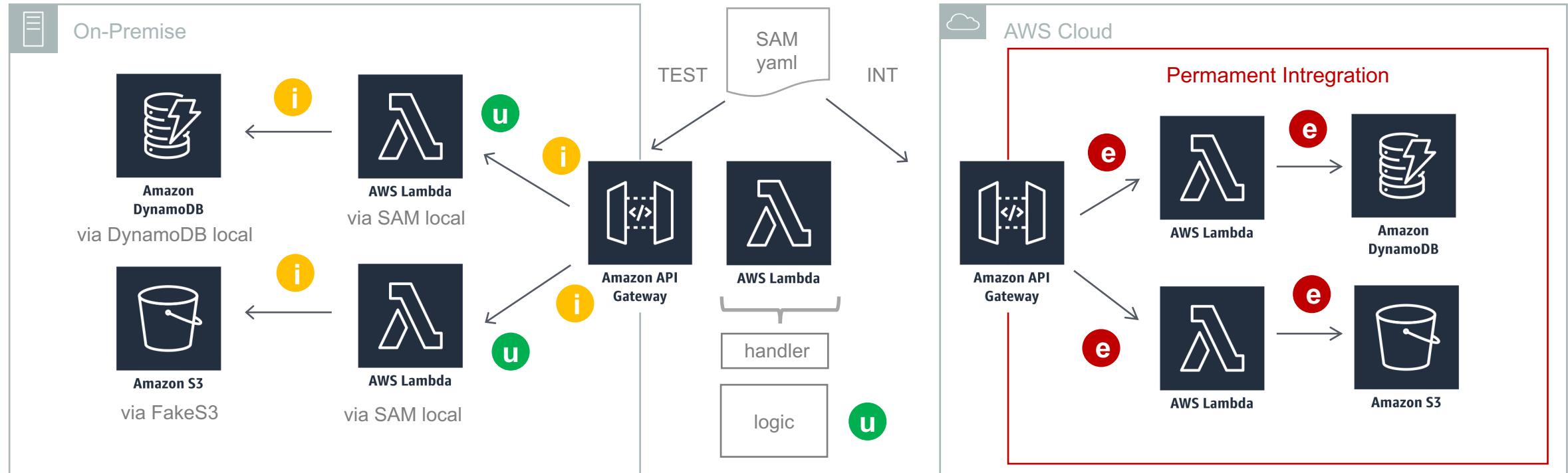
#6 temporäre Integration-Cloud für partielle Integration Tests



Testing Best Practices

- u Kandidat für Unit Tests
- i Kandidat für Integration Tests
- e Kandidat für End-to-End Tests

#7 permanente Integration-Cloud für End-to-End Tests



Lessons learned: Serverless U/I/A-Testing



SERVERLESS U/I/A-TESTING

Lessons Learned

- **Separation of Concerns**
- **SAM** zum lokalen Starten von Lambdas
- **Locale Cloud** zur lokalen Simulation
- **CloudFormation** zum Aufsetzen von Cloud ENVs
- **Nur die Cloud verhält sich wie die Cloud!**



Agenda

- Hello World & Set-up*
- Order Management*
- Order Processing*
- API Gateway*
- Testing*



*Was ihr aus dem heutigen Workshop **mitnehmen** solltet:*

- **eine Lambda** kommt nie allein
- je **kleiner** desto besser
- je **kürzer** desto besser
- **Coldstart** als Herausforderung
- **Testen** auf allen Ebenen!
- **IaC** ist dein (Cloud-)Freund



**„Going Serverless“
ist keine Frage der
Technologie, sondern eine
Architekturentscheidung!**



Zeit für
Fragen?
Immer!



#WISSENTEILEN
by open knowledge GmbH

Vielen
Dank!

Lars Röwekamp, CIO New Technologies
 @_openKnowledge | @mobileLarson



BILDNACHWEIS

Folie 01: © Biletsky Egeniy, iStockphoto.com

Folie 05: © Ljubco, iStockphoto.com

Folie 06: © Orba Aljia, iStockphoto.com

Folie 44: © foxaon1987, Shuttertock.com

Folie 90: © Framework Wonderland, Shutterstock.com

All other pictures, drawings and icons originate from

- pexels.com, pixabay.com, unsplash.com,
- flaticon.com

or were made by my own.