AMBASSADOR

Simplifying developer experience with new features in AWS Step Functions

Håkon Eriksen Drange

Principal Cloud Architect, Sopra Steria

Disclaimer: Standing on the shoulders of giants

RECOMMENDED MATERIAL FOR SERVERLESS ENTHUSIASTS

API402

Building advanced workflows with AWS Step Functions

Eric Johnson
Principal Developer Advocate, Serverless
Amazon Web Services

SVS401

Best practices for serverless developers

Julian Wood
(he/him)
Principal Serverless Developer Advocate
AWS

Ran Isenberg
(he/him)
AWS Hero & Principal Architect
CyberArk

AWS re:Invent 2024 - Serverless Compute

by AWS Events

Playlist • 35 videos • 3,653 views

https://serverlessland.com/reinvent2024/api402

https://serverlessland.com/reinvent2024/svs401

https://www.youtube.com/playlist?list=PL2yQDdvlhXf_Ezjnq7A7LfHBgCYSqzrZS

Why go serverless?

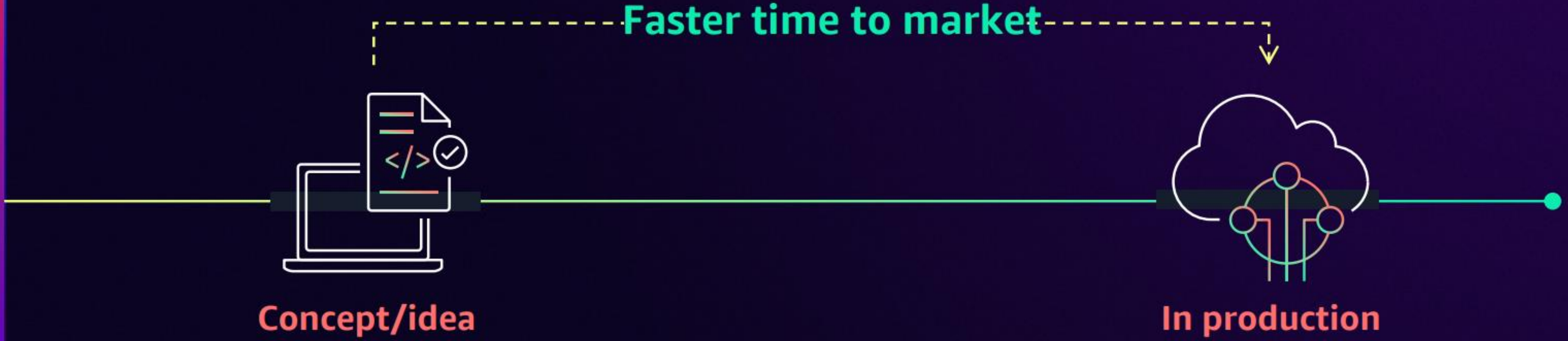
Comparison of operational responsibility for compute

More opinionated

Less opinionated

	AWS manages	Customer manages
AWS Lambda Serverless functions	• Data source integrations • Physical hardware, software, networking, and facilities • Provisioning	• Application code
AWS Fargate Serverless containers	• Container orchestration, provisioning • Cluster scaling • Physical hardware, host OS/kernel, networking, and facilities	• Application code • Data source integrations • Security config and updates, network config, management tasks
ECS/EKS Container-management as a service	• Container orchestration control plane • Physical hardware software, networking, and facilities	• Application code • Data source integrations • Work clusters • Security config and updates, network config, firewall, management tasks
EC2 Infrastructure-as-a-Service	• Physical hardware software, networking, and facilities	• Application code • Data source integrations • Scaling • Security config and updates, network config, management tasks • Provisioning, managing scaling and patching of servers

Why go serverless?



A faster way to get to customer value

AWS Serverless spectrum

INCLUDES

AWS OFFERS A WIDE PORTFOLIO OF SERVERLESS SERVICES

Compute

- AWS Lambda
- AWS Fargate
- Amazon ECS
- AWS App Runner

Storage

- Amazon S3
- Amazon EFS

Workflows and Integrations

- Amazon EventBridge
- AWS Step Functions
- Amazon API Gateway
- AWS AppSync
- Amazon SQS
- Amazon SNS
- Amazon Kinesis

Databases and Analytics

- Amazon Aurora Serverless
- Amazon DynamoDB
- Amazon OpenSearch Service
- Amazon Bedrock
- Amazon QuickSight
- AWS Glue
- Amazon Redshift

The serverless challenge

Question: When your application is powered by multiple connected services, how do you build, track, inspect, visualize, and orchestrate those connections?

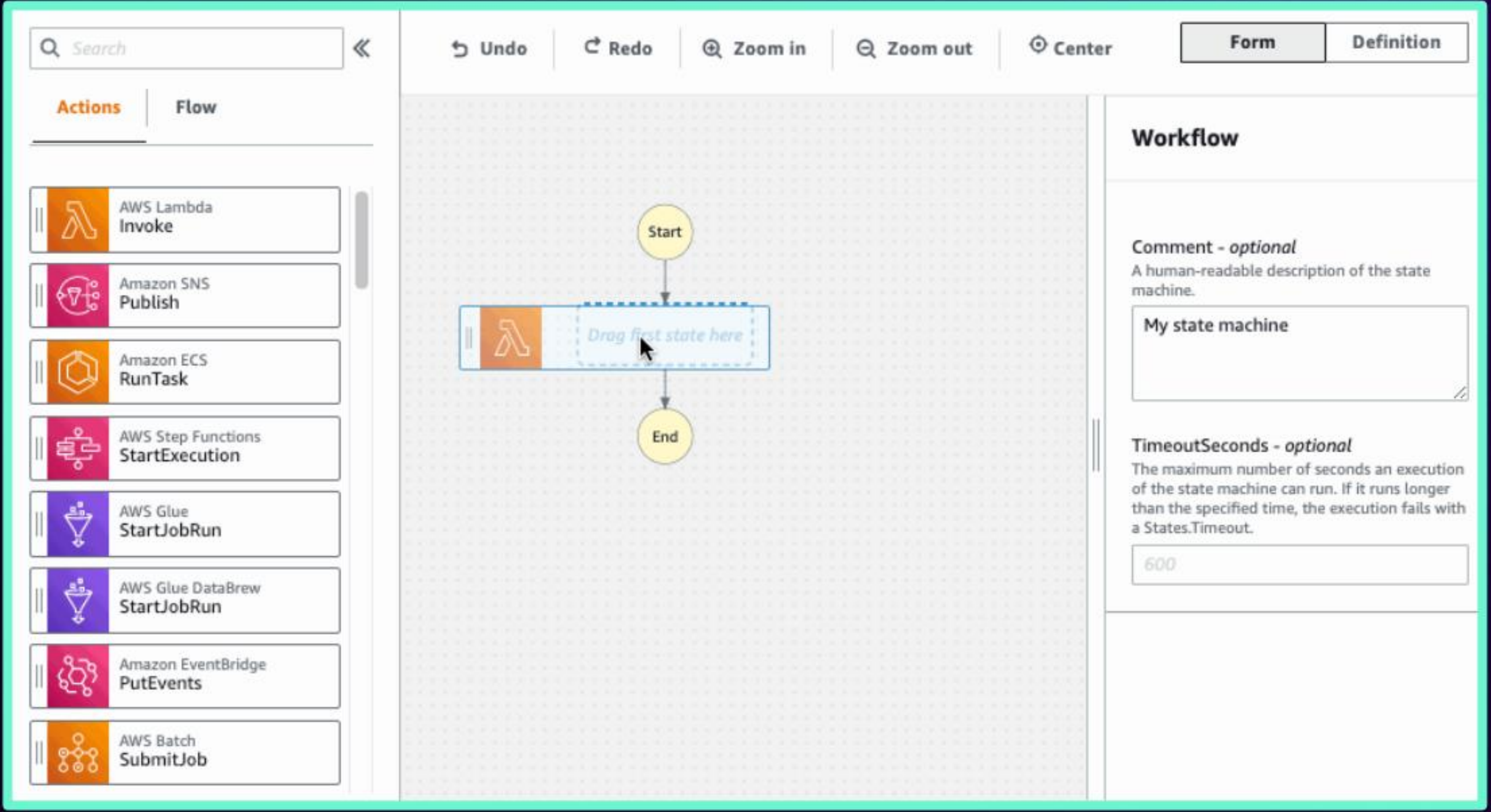
Answer: AWS Step Functions

A SERVERLESS, LOW-CODE VISUAL WORKFLOW SERVICE

- Pay-per-use
- Scales automatically
- Fully managed

- Drag and drop or ASL
- Built-in error handling

- Integrates with over 200 AWS services



Workflow

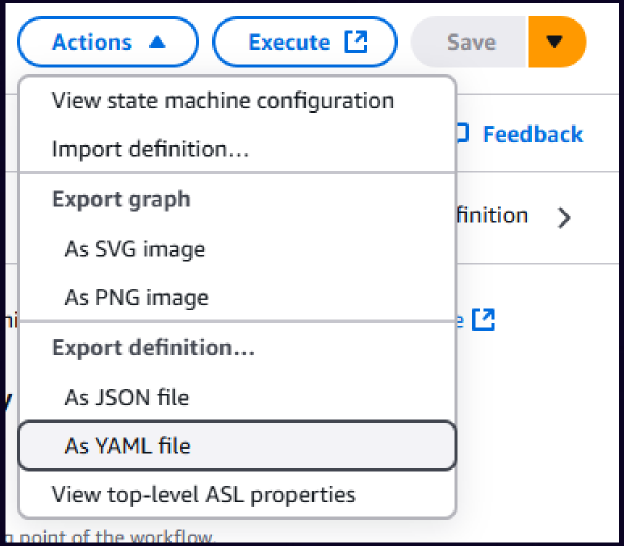
Comment - *optional*
A human-readable description of the state machine.

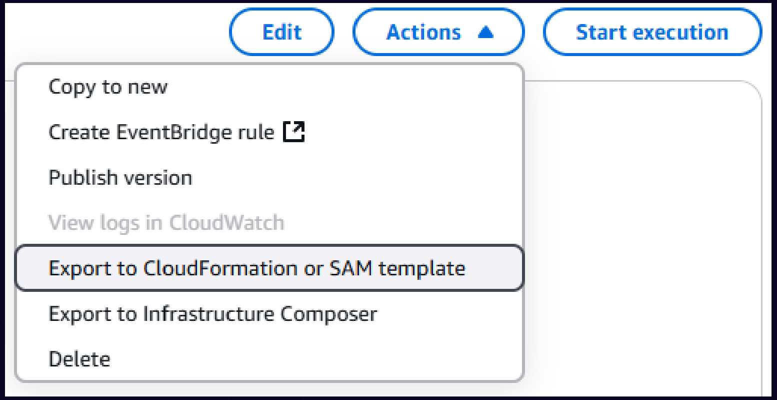
My state machine

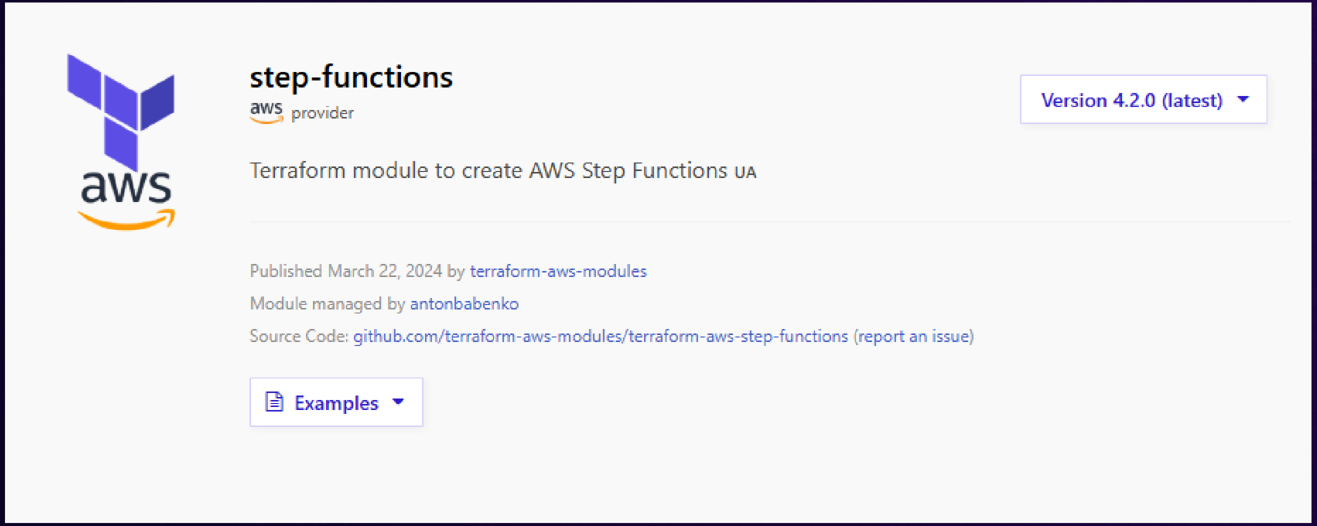
TimeoutSeconds - *optional*
The maximum number of seconds an execution of the state machine can run. If it runs longer than the specified time, the execution fails with a States.Timeout.

600

AWS Step Functions IaC







step-functions

aws provider

Version 4.2.0 (latest)

Terraform module to create AWS Step Functions UA

Published March 22, 2024 by terraform-aws-modules

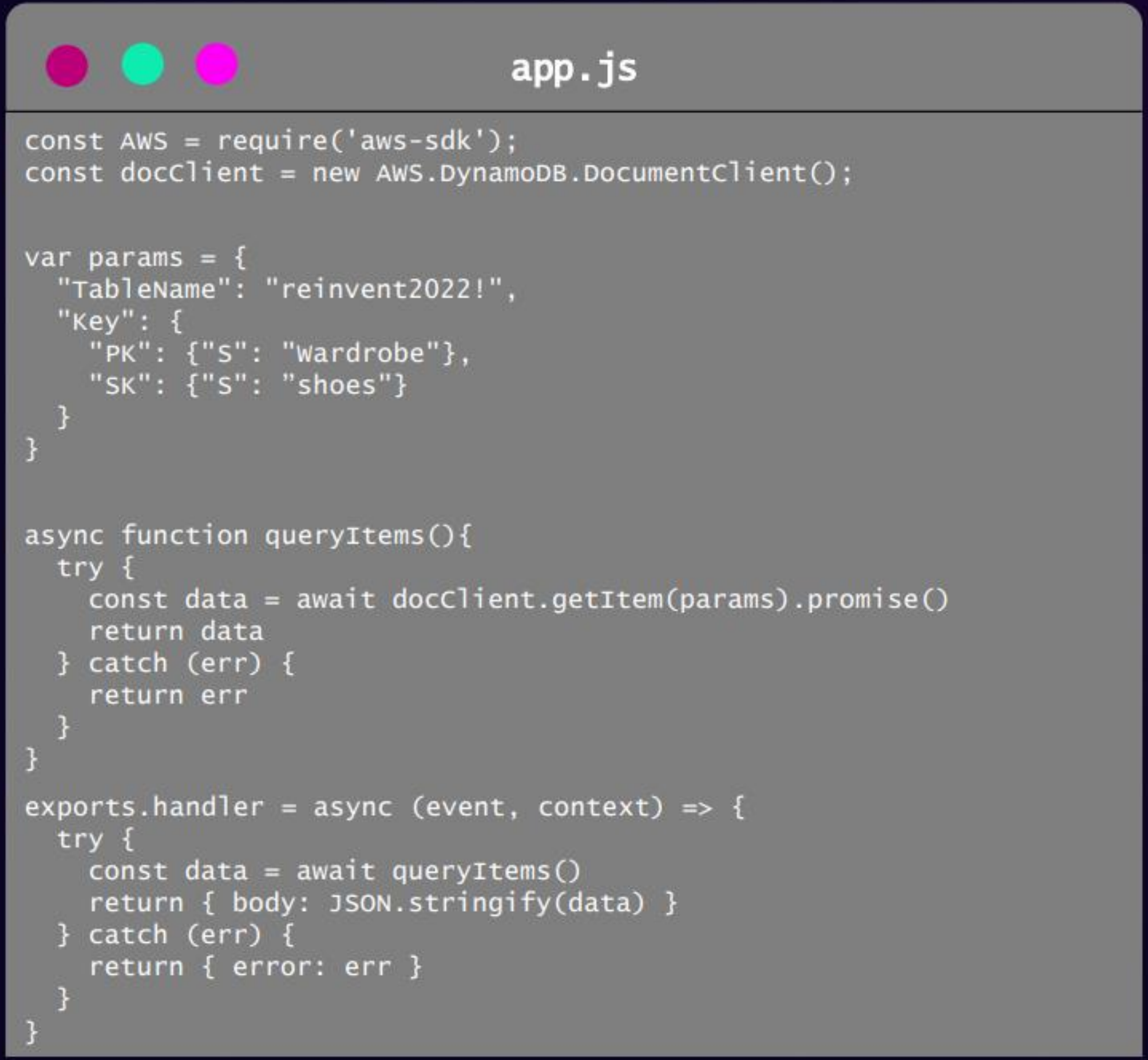
Module managed by antonbabenko

Source Code: github.com/terraform-aws-modules/terraform-aws-step-functions (report an issue)

Examples

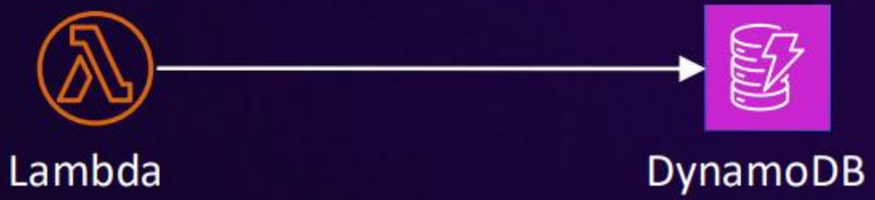
https://github.com/terraform-aws-modules/terraform-aws-step-functions

From code to workflow


app.js

```
const AWS = require('aws-sdk');
const docClient = new AWS.DynamoDB.DocumentClient();


var params = {
  "TableName": "reinvent2022!",
  "Key": {
    "PK": {"S": "Wardrobe"},
    "SK": {"S": "shoes"}
  }
}


async function queryItems(){
  try {
    const data = await docClient.getItem(params).promise()
    return data
  } catch (err) {
    return err
  }
}

exports.handler = async (event, context) => {
  try {
    const data = await queryItems()
    return { body: JSON.stringify(data) }
  } catch (err) {
    return { error: err }
  }
}
```

An AWS Lambda function that queries Amazon DynamoDB has multiple lines of code

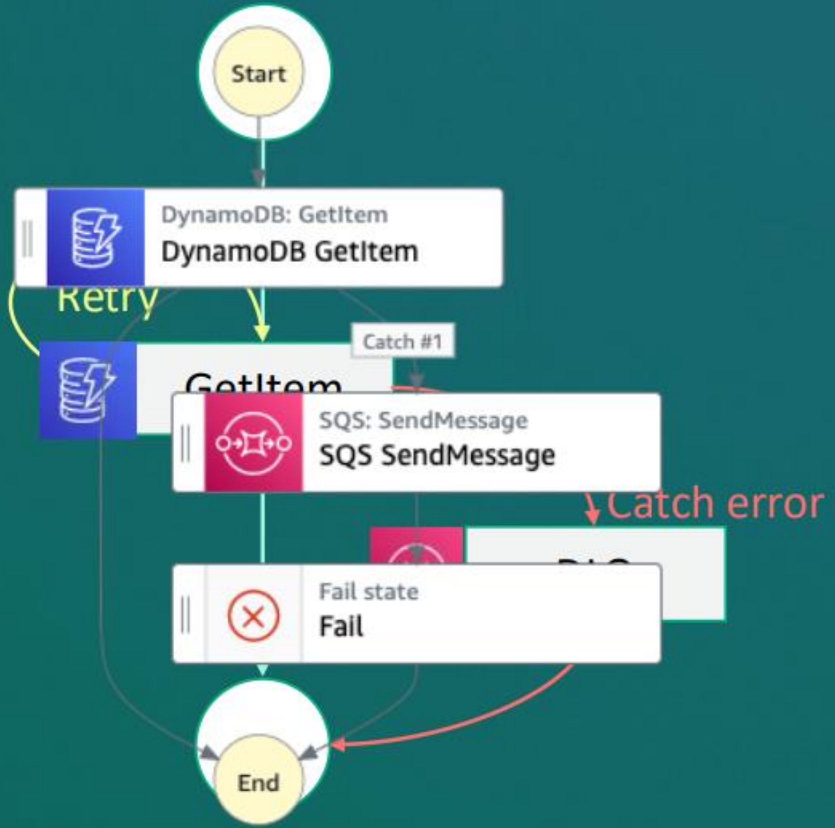


From code to workflow

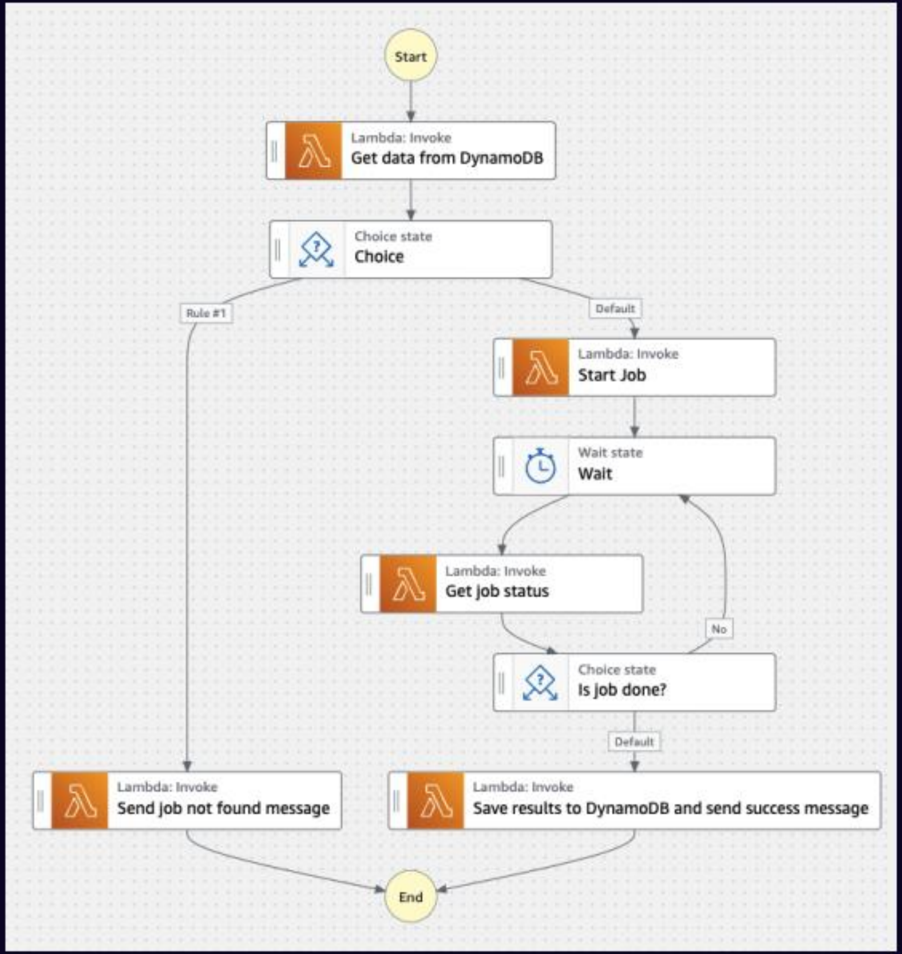
app.js

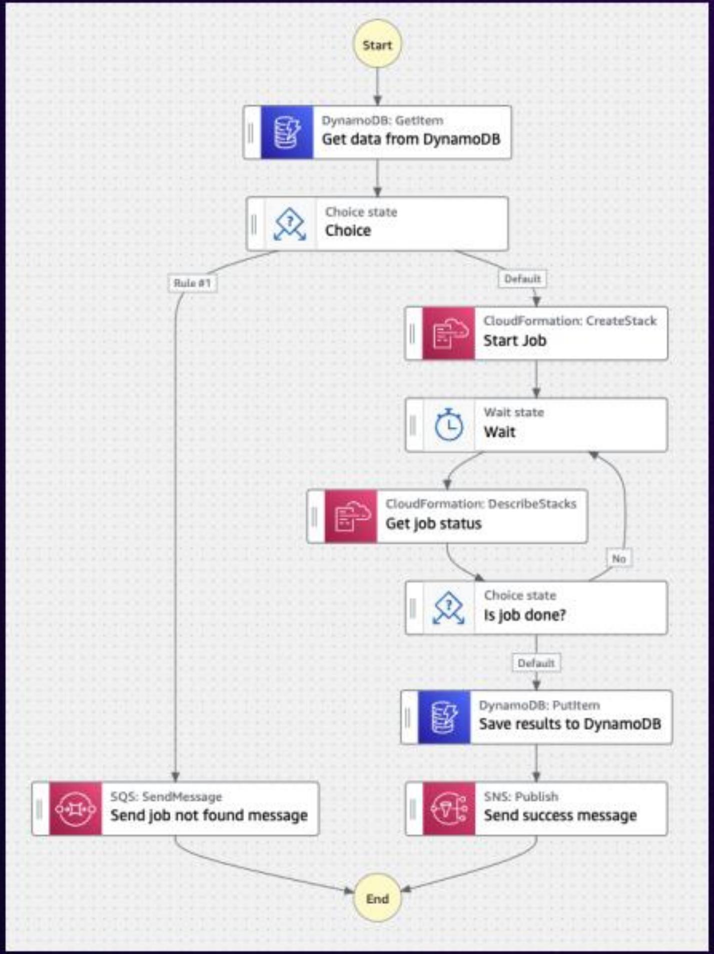
```
const AWS = require('aws-sdk');
const docClient = new AWS.DynamoDB.DocumentClient();


var params = {
  "TableName": "reinvent2022!",
  "Key": {
    "PK": {"S": "Wardrobe"},
    "SK": {"S": "shoes"}
  }
}


async function queryItems(){
  try {
    const data = await docClient.getItem(params).promise()
    return data
  } catch (err) {
    return err
  }
}

exports.handler = async (event, context) => {
  try {
    const data = await queryItems()
    return { body: JSON.stringify(data) }
  } catch (err) {
    return { error: err }
  }
}
```

Even a single-task "workflow" adds value with built-in error handling, catch, retry, observability, reduction of custom code, and centralized logging of each workload

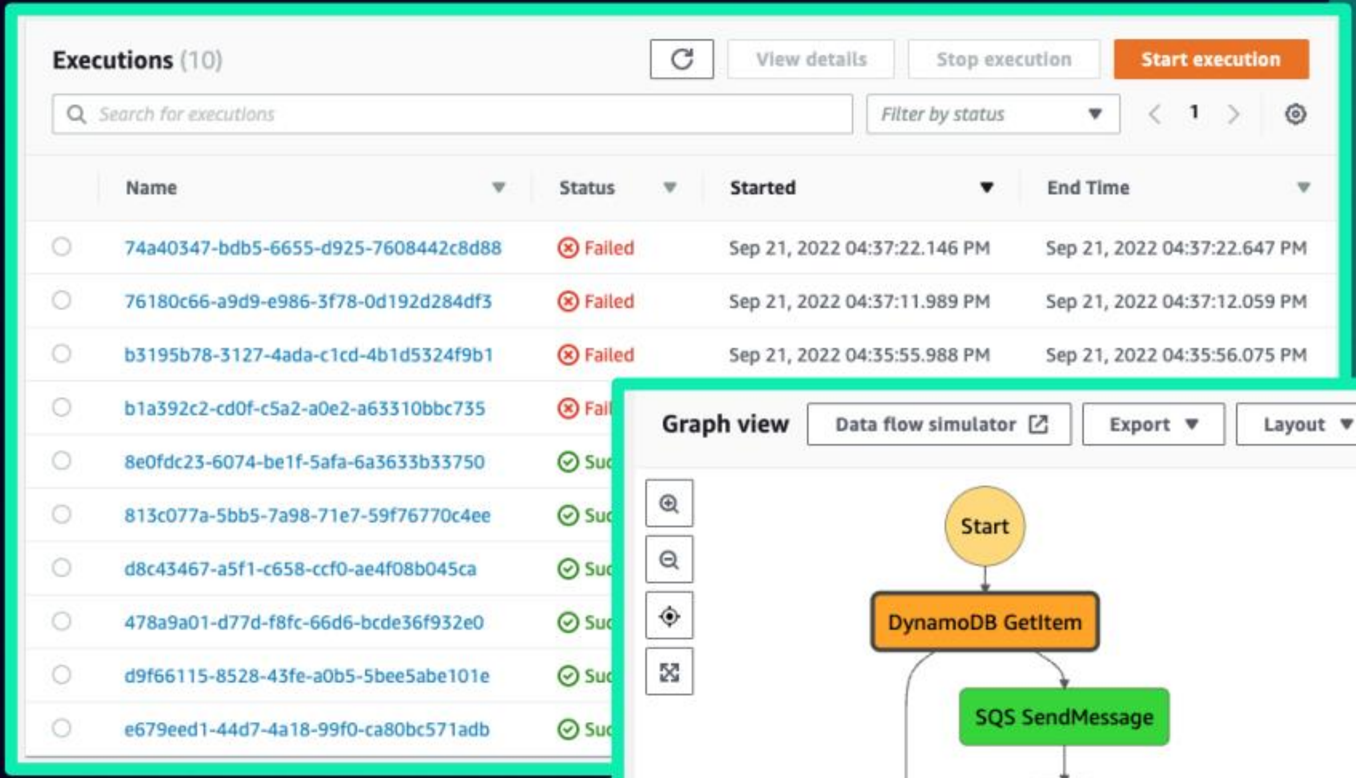


Step Functions SDK integrations



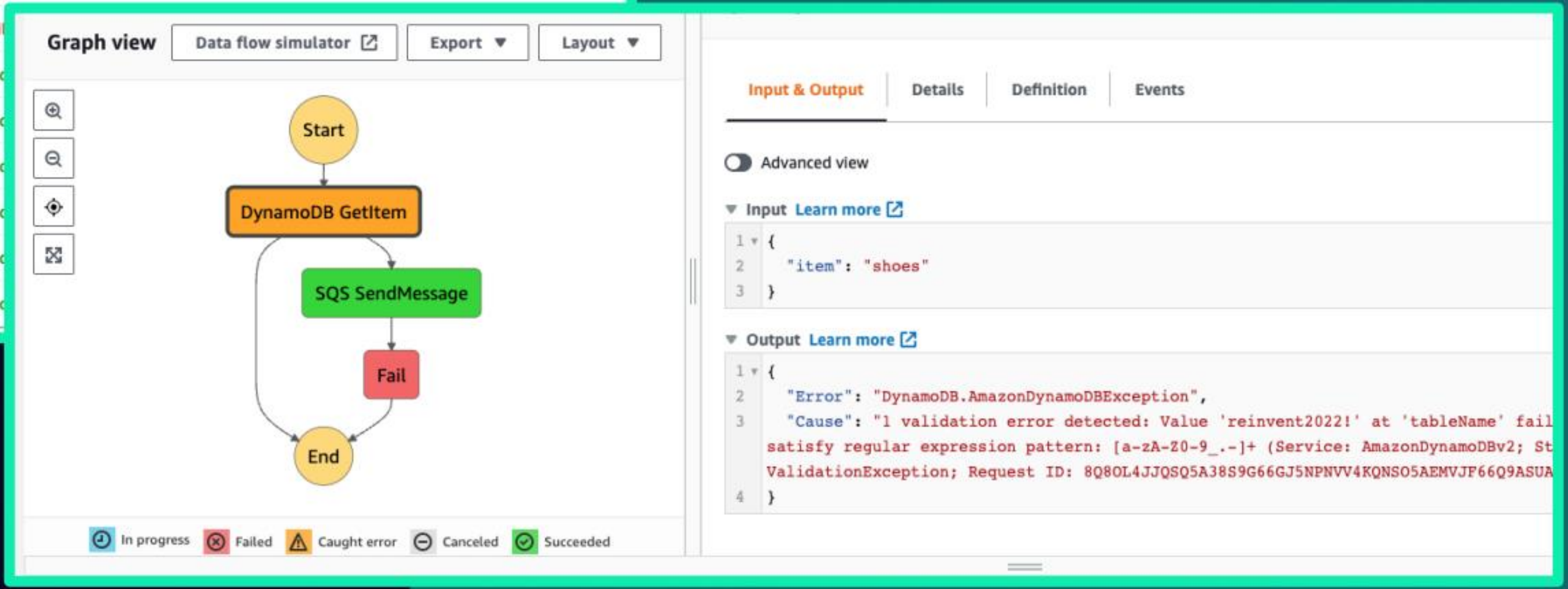


From code to workflow



	Name	Status	Started	End Time
○	74a40347-bdb5-6655-d925-7608442c8d88	⊗ Failed	Sep 21, 2022 04:37:22.146 PM	Sep 21, 2022 04:37:22.647 PM
○	76180c66-a9d9-e986-3f78-0d192d284df3	⊗ Failed	Sep 21, 2022 04:37:11.989 PM	Sep 21, 2022 04:37:12.059 PM
○	b3195b78-3127-4ada-c1cd-4b1d5324f9b1	⊗ Failed	Sep 21, 2022 04:35:55.988 PM	Sep 21, 2022 04:35:56.075 PM
○	b1a392c2-cd0f-c5a2-a0e2-a63310bbc735	⊗ Fail		
○	8e0fdc23-6074-be1f-5afa-6a3633b33750	✓ Suc		
○	813c077a-5bb5-7a98-71e7-59f76770c4ee	✓ Suc		
○	d8c43467-a5f1-c658-ccf0-ae4f08b045ca	✓ Suc		
○	478a9a01-d77d-f8fc-66d6-bcde36f932e0	✓ Suc		
○	d9f66115-8528-43fe-a0b5-5bee5abe101e	✓ Suc		
○	e679eed1-44d7-4a18-99f0-ca80bc571adb	✓ Suc		

Drill down to trace the execution path of every workload request



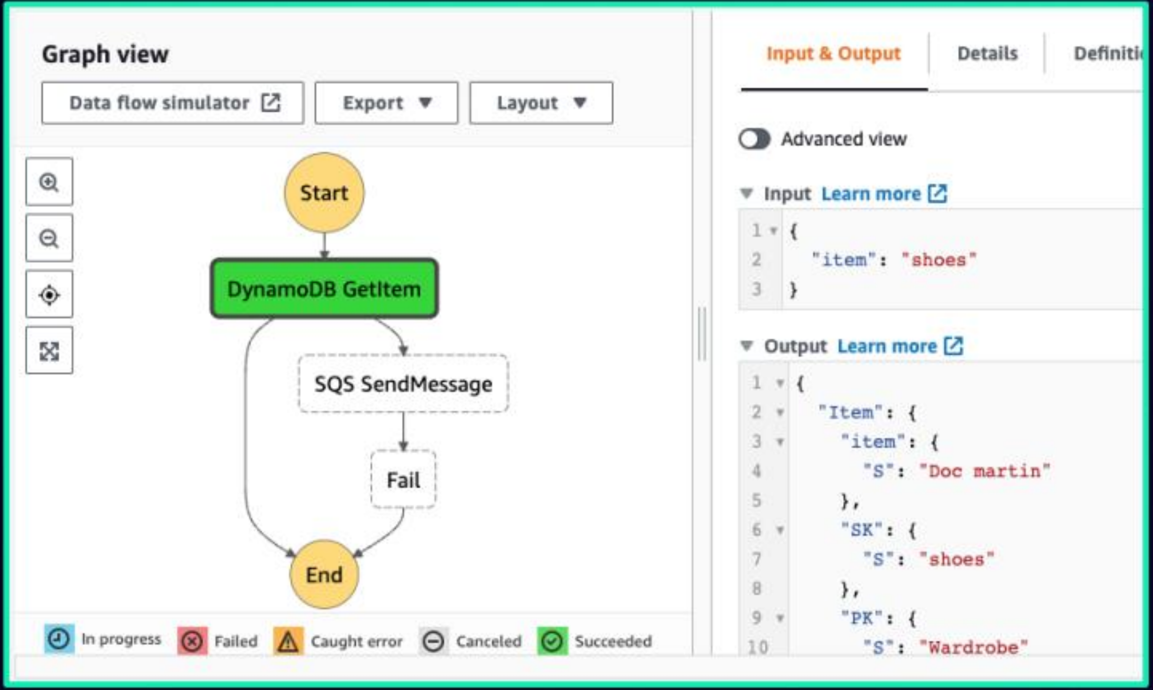
Input Learn more

```
{
  "item": "shoes"
}
```

Output Learn more

```
{
  "Error": "DynamoDB.AmazonDynamoDBException",
  "Cause": "1 validation error detected: Value 'reinvent2022!' at 'tableName' fail
satisfy regular expression pattern: [a-zA-Z0-9_.-]+ (Service: AmazonDynamoDBv2; St
ValidationException; Request ID: 8Q8OL4JJQSQ5A38S9G66GJ5NPNVV4KQNSO5AEMVJF66Q9ASUA
}
```

From code to workflow

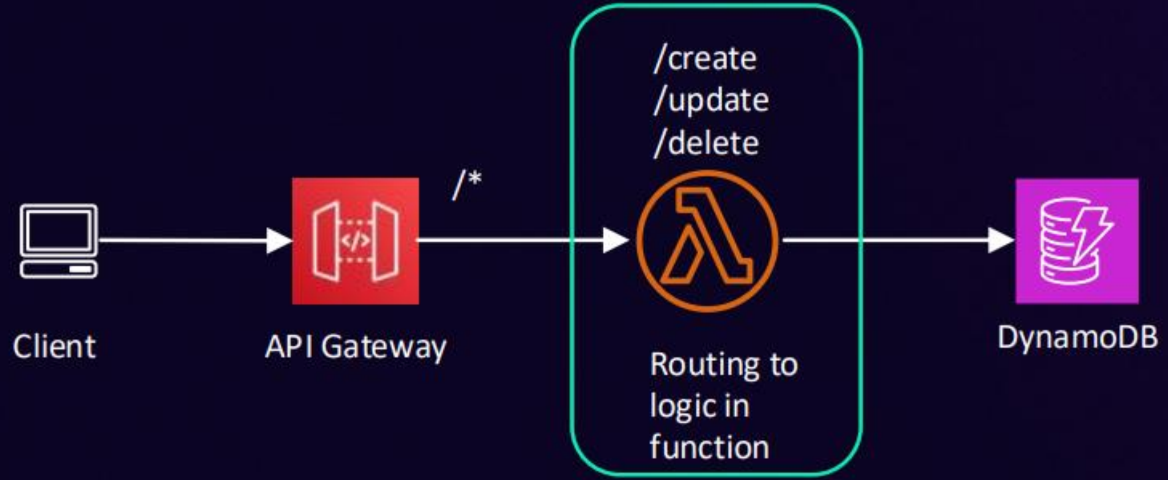


```
{
  "item": "shoes"
}
```

```
{
  "Item": {
    "item": {
      "S": "Doc martin"
    },
    "SK": {
      "S": "shoes"
    },
    "PK": {
      "S": "Wardrobe"
```

Examine the input and output of each task for each request

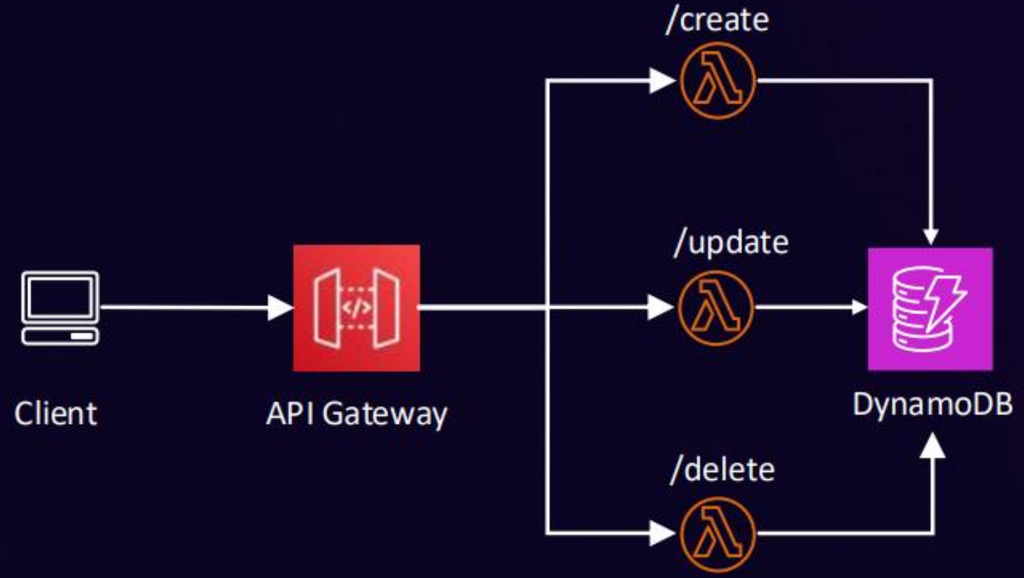
Breaking apart a “Lambda-lith”

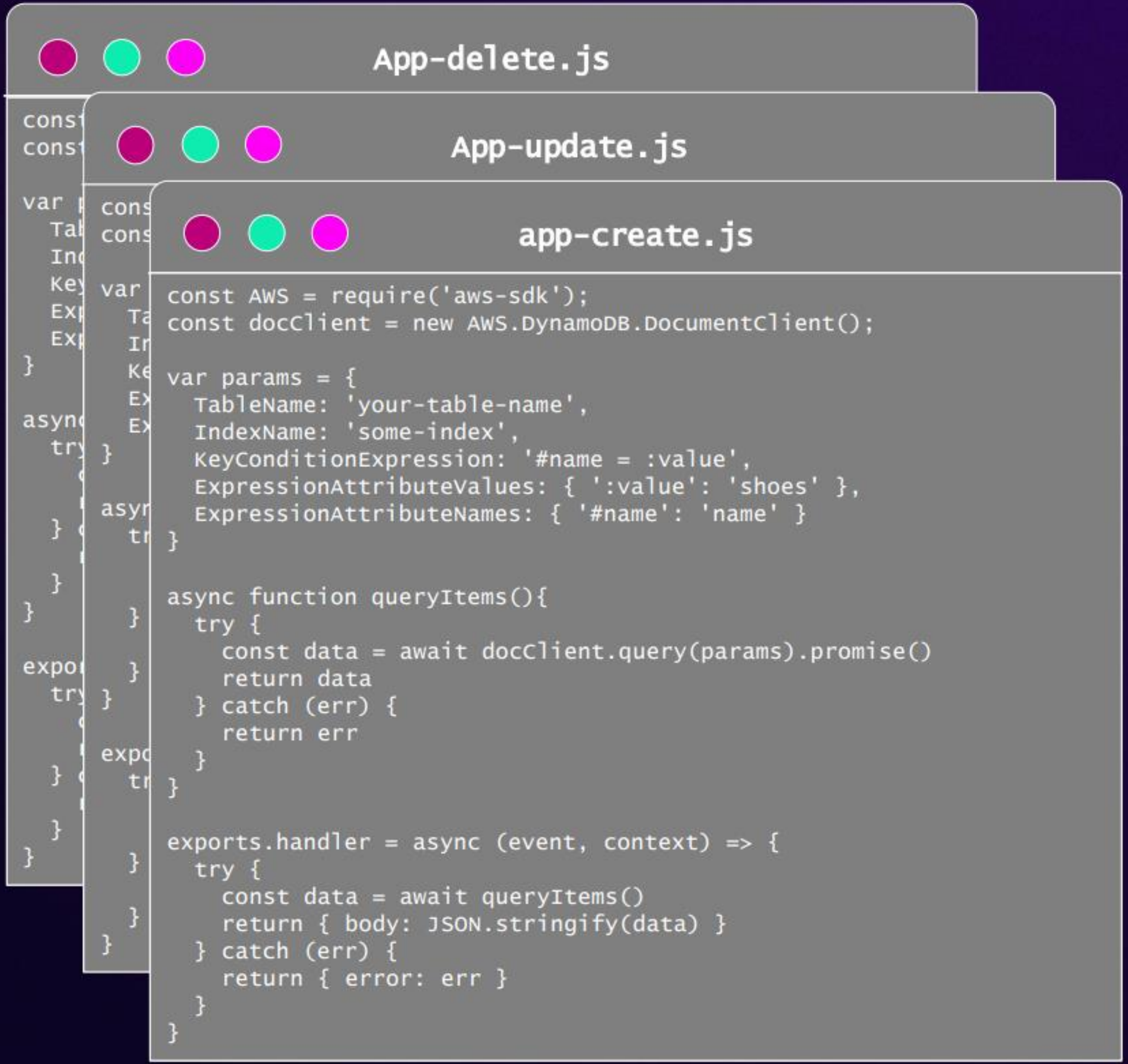


Amazon API Gateway routes all requests to a single Lambda function that runs the appropriate code based on its route configuration

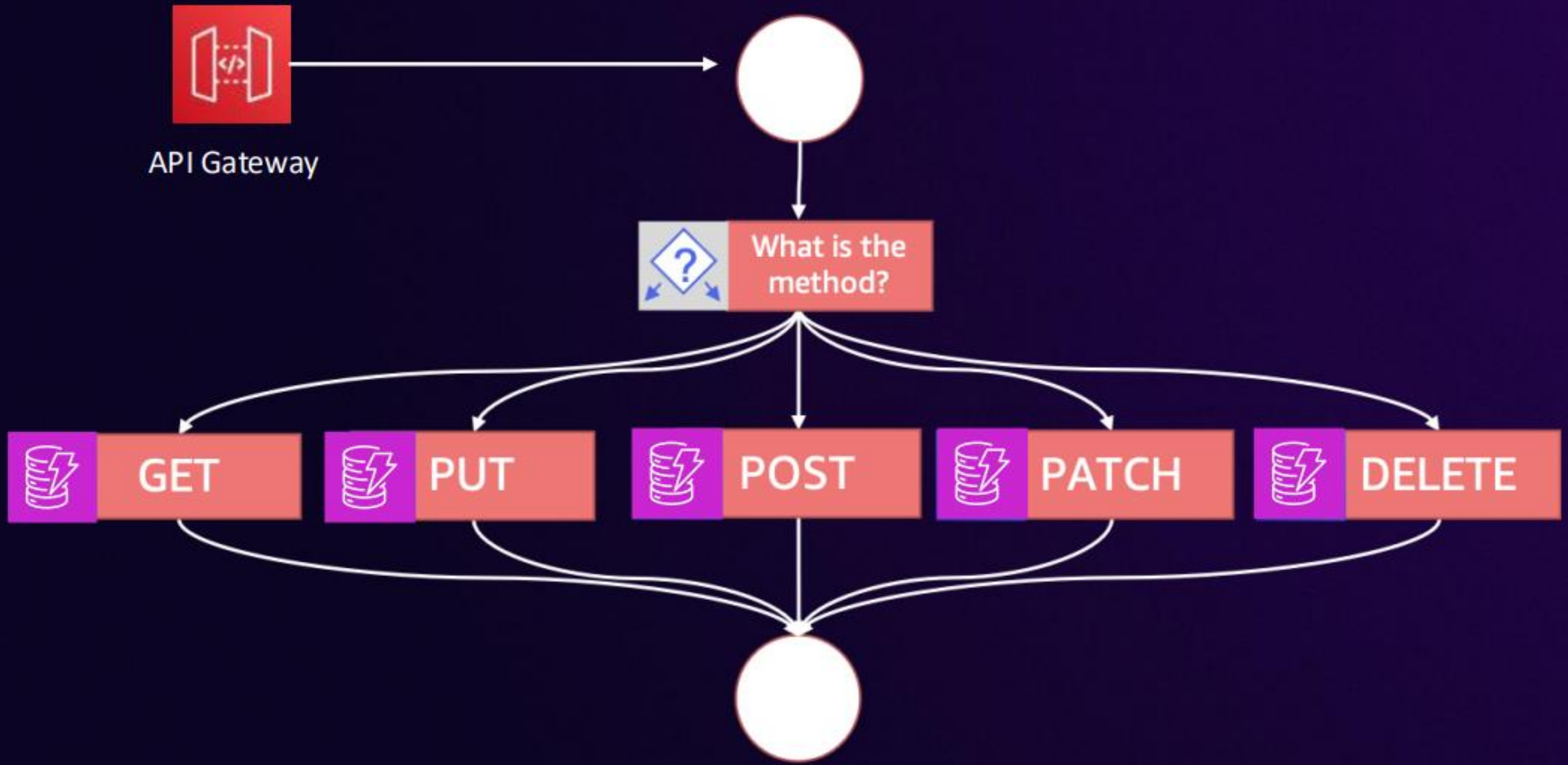
- Security permissions applied to the whole
- Performance setting applied to the whole
- Duration and space limits applied to the whole

Micro Lambda





```
const AWS = require('aws-sdk');
const docClient = new AWS.DynamoDB.DocumentClient();

var params = {
  TableName: 'your-table-name',
  IndexName: 'some-index',
  KeyConditionExpression: '#name = :value',
  ExpressionAttributeValues: { ':value': 'shoes' },
  ExpressionAttributeNames: { '#name': 'name' }
}

async function queryItems(){
  try {
    const data = await docClient.query(params).promise()
    return data
  } catch (err) {
    return err
  }
}

exports.handler = async (event, context) => {
  try {
    const data = await queryItems()
    return { body: JSON.stringify(data) }
  } catch (err) {
    return { error: err }
  }
}
```



The “REST” easy

Combine with API Gateway and Step Functions Synchronous Express Workflows to create a low-latency, scalable API backend

Simplicity

Step Functions intrinsic functions

Arrays

JSON data manipulation

Encoding and decoding

Math operations

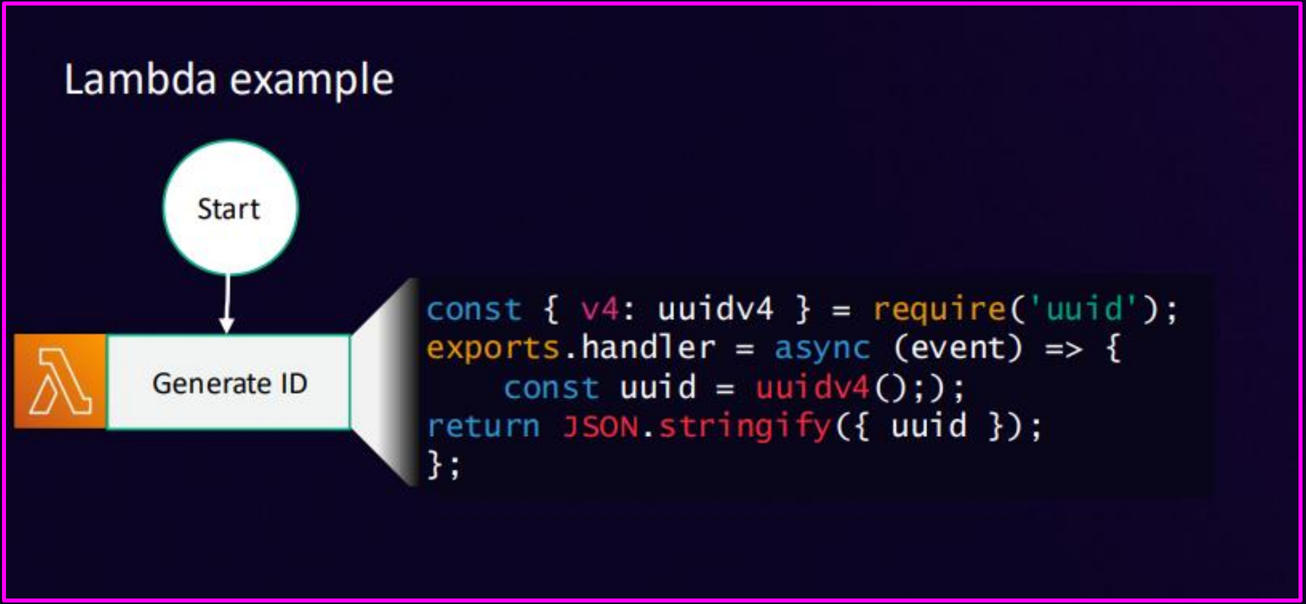
String operations

Unique identifier generation

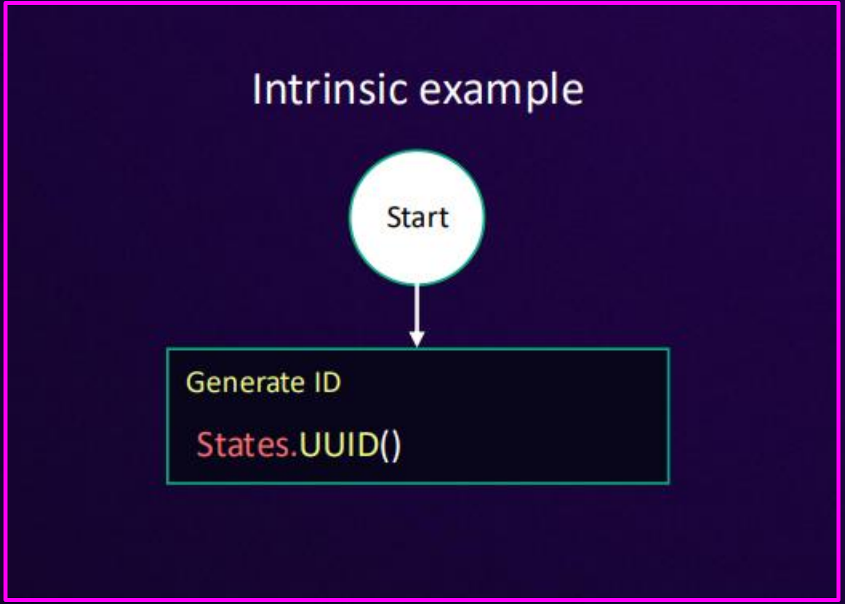
Reducing cost: Standard Workflows

USE INTRINSIC FUNCTIONS INSTEAD OF COMPUTE SERVICES TO PERFORM DATA TRANSFORMATIONS

Generating a unique ID



```
const { v4: uuidv4 } = require('uuid');
exports.handler = async (event) => {
    const uuid = uuidv4(););
return JSON.stringify({ uuid });
};
```



Arrays

JSON data manipulation

Unique identifier generation

String operations

Math operations

Encoding and decoding

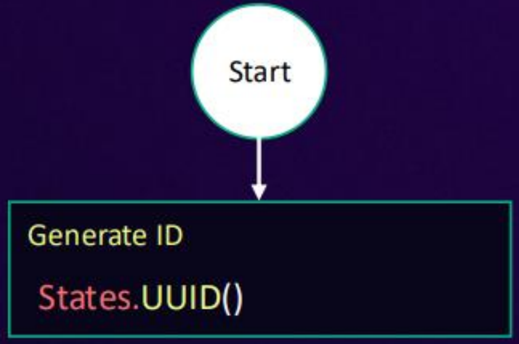
Reducing cost: Standard Workflows

USE INTRINSIC FUNCTIONS INSTEAD OF COMPUTE SERVICES TO PERFORM DATA TRANSFORMATIONS

Generating a unique ID

Intrinsic example

No invocation delays
No invocation cost
Less code to manage



Arrays

JSON data manipulation

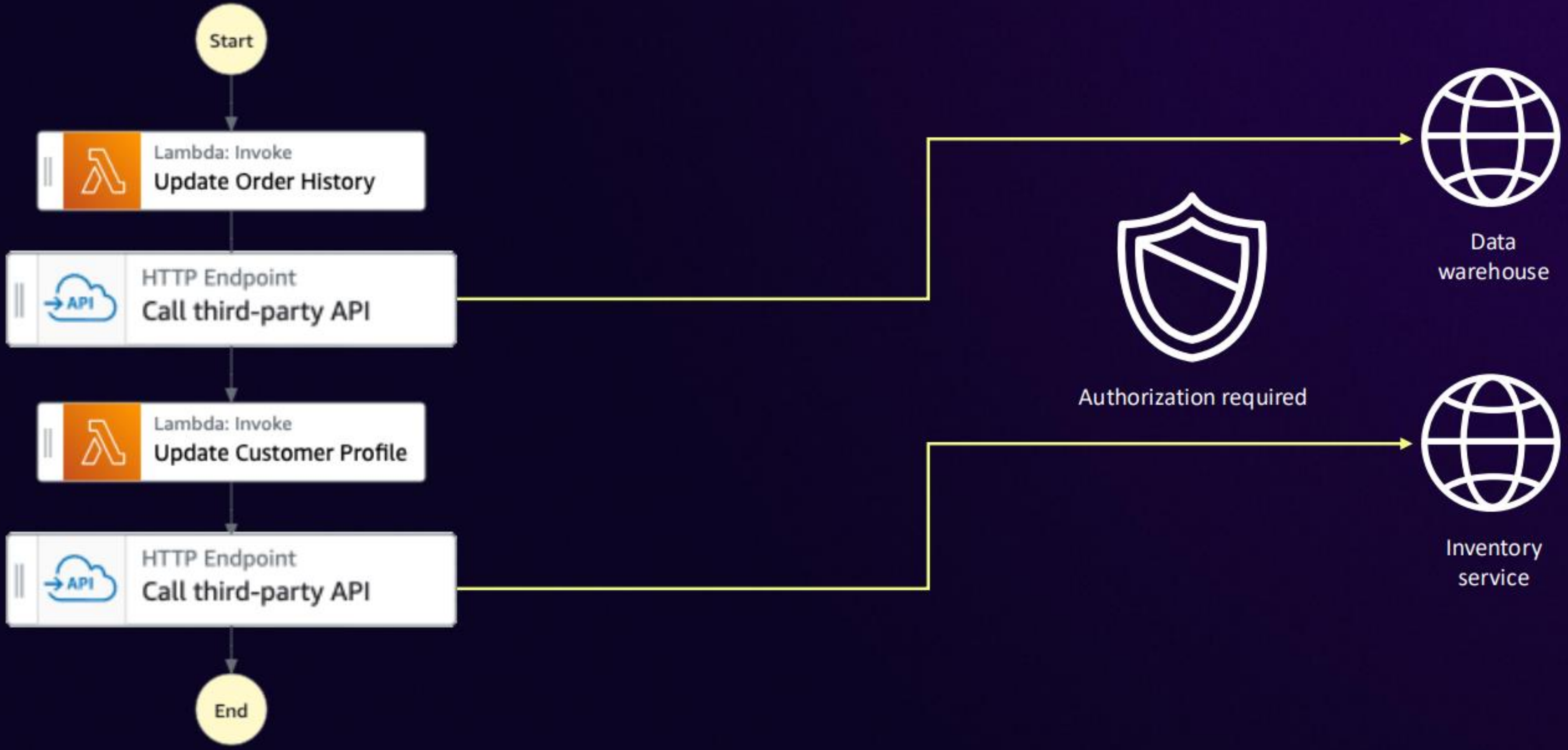
Unique identifier generation

String operations

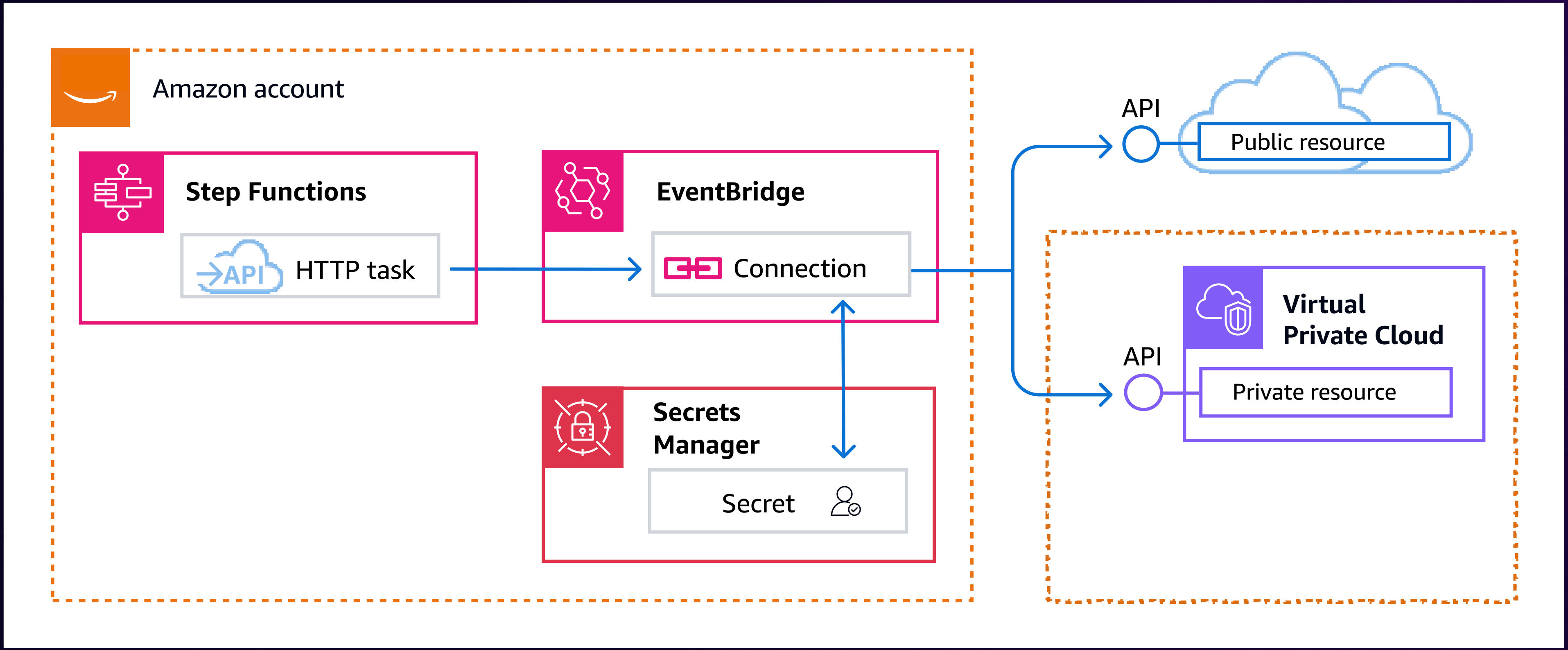
Math operations

Encoding and decoding

HTTP API integrations



HTTP API integrations



AWS Step Functions: JSONata and Variables

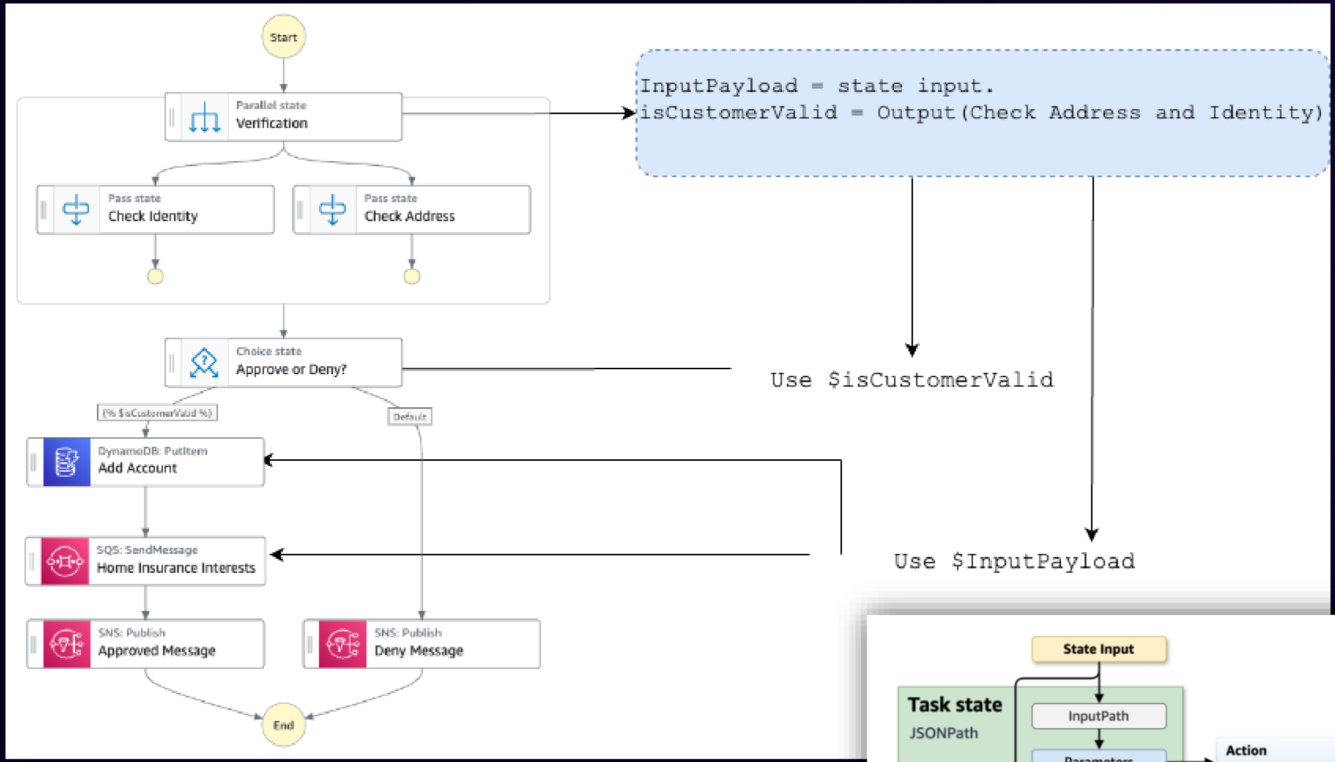
SIMPLIFY STATE PAYLOAD MANAGEMENT AND DATA TRANSFORMATION IN STATE MACHINES

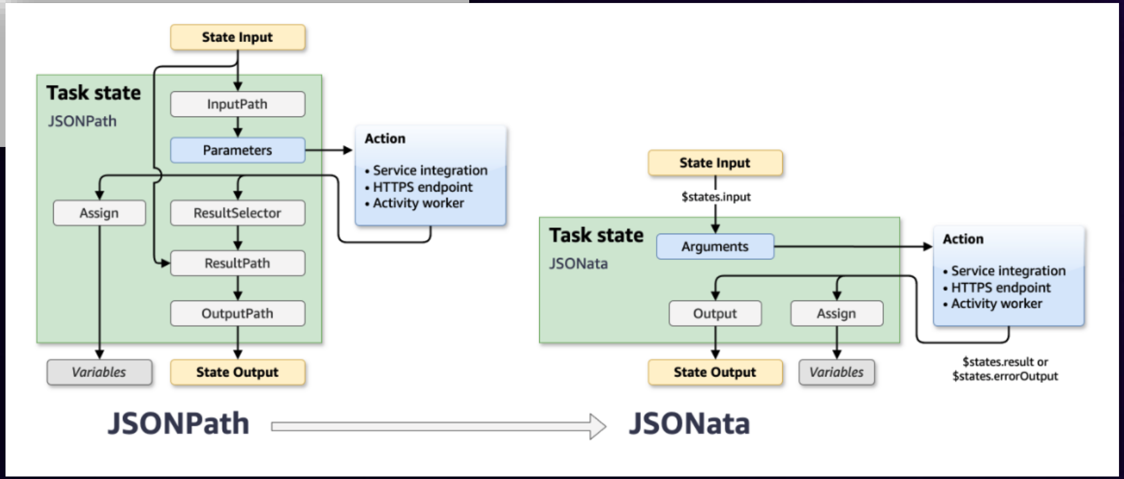
GA

Nov, 22nd

pre-re:Invent

All Regions

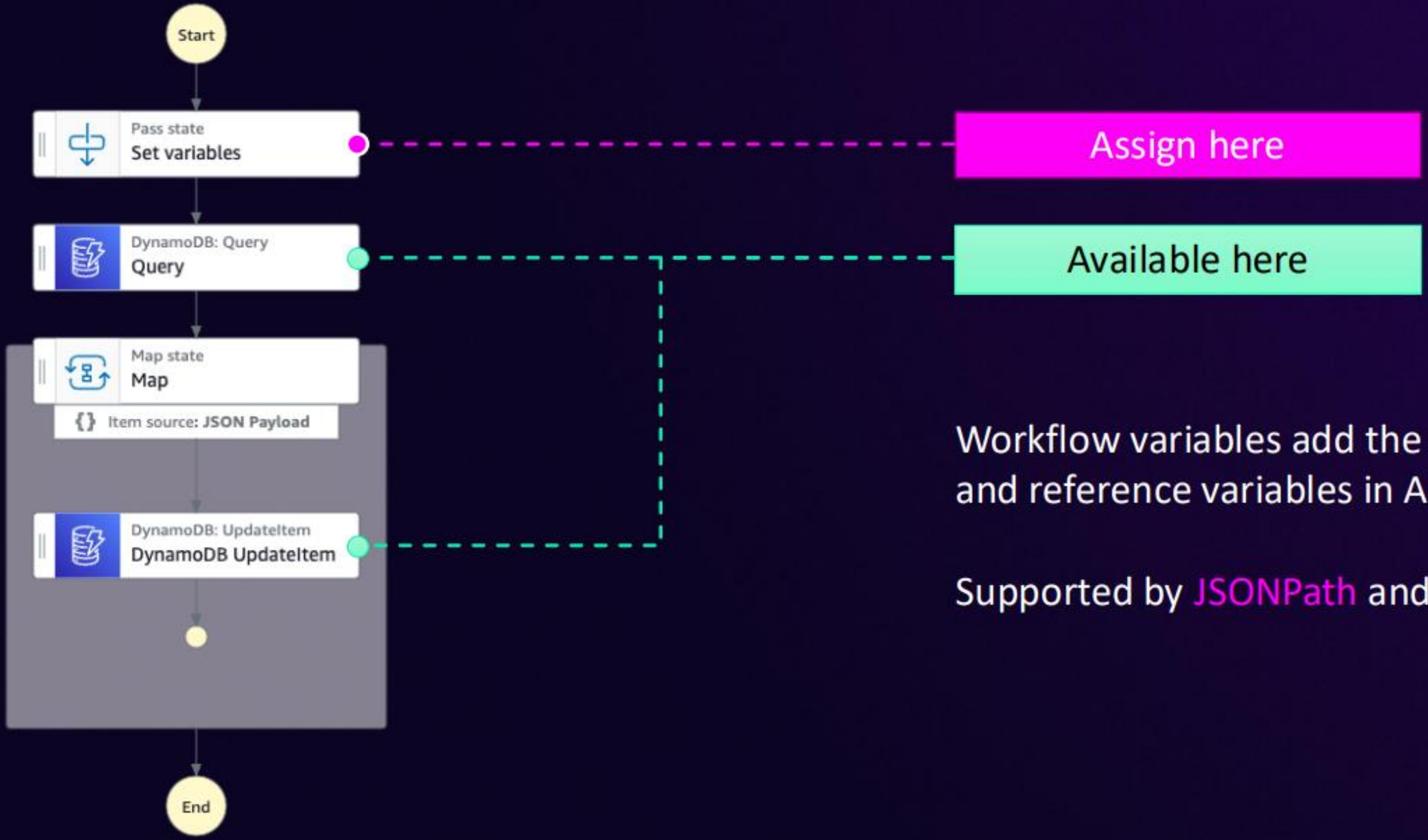




AWS Compute Blog Post

Step Function Workflow Variables

New



Workflow variables add the ability to assign and reference variables in ASL.

Supported by JSONPath and JSONata

Workflow Variable assignment

New

```
"Set variables": {
  "Type": "Pass",
  "Next": "Query",
  "Assign": {
    "oldOwner": ”{% $states.input.oldOwner %}",
    "newOwner": ”{% $states.input.newOwner %}",
    "table": "S12D"
  }
}
```

New

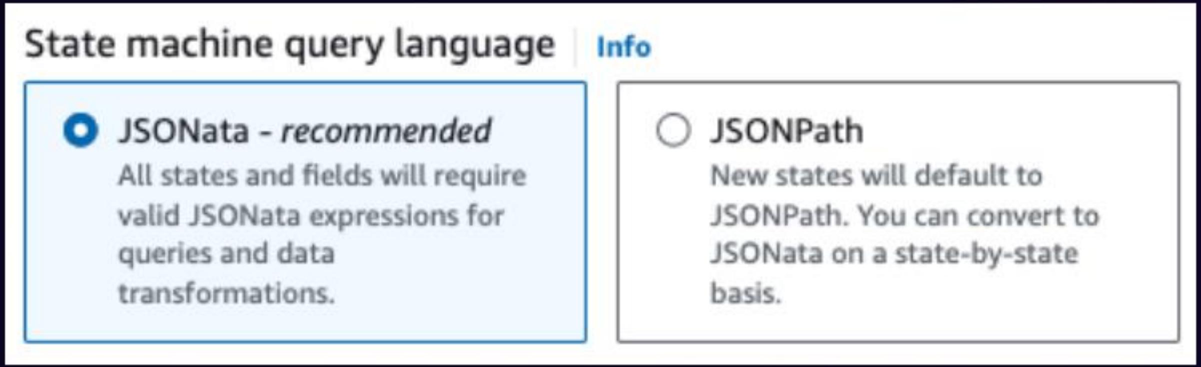
Workflow Variable reference

Syntax: $<variable_name>

```
"Query": {
  "Type": "Task",
  "Parameters": {
    "TableName": "{% $table %}",
    "IndexName": "OwnerIndex",
    "KeyConditionExpression": "#owner = :owner",
    "Limit": 10,
    "ExpressionAttributeNames": {"#owner": "owner"},
    "ExpressionAttributeValues": {
      ":owner": {"S": "{% $oldOwner %}"}
    }
  },
  ...
}
```

JSONata support for data transformation

JSONata is a powerful query and expression language to select and transform data in your workflows

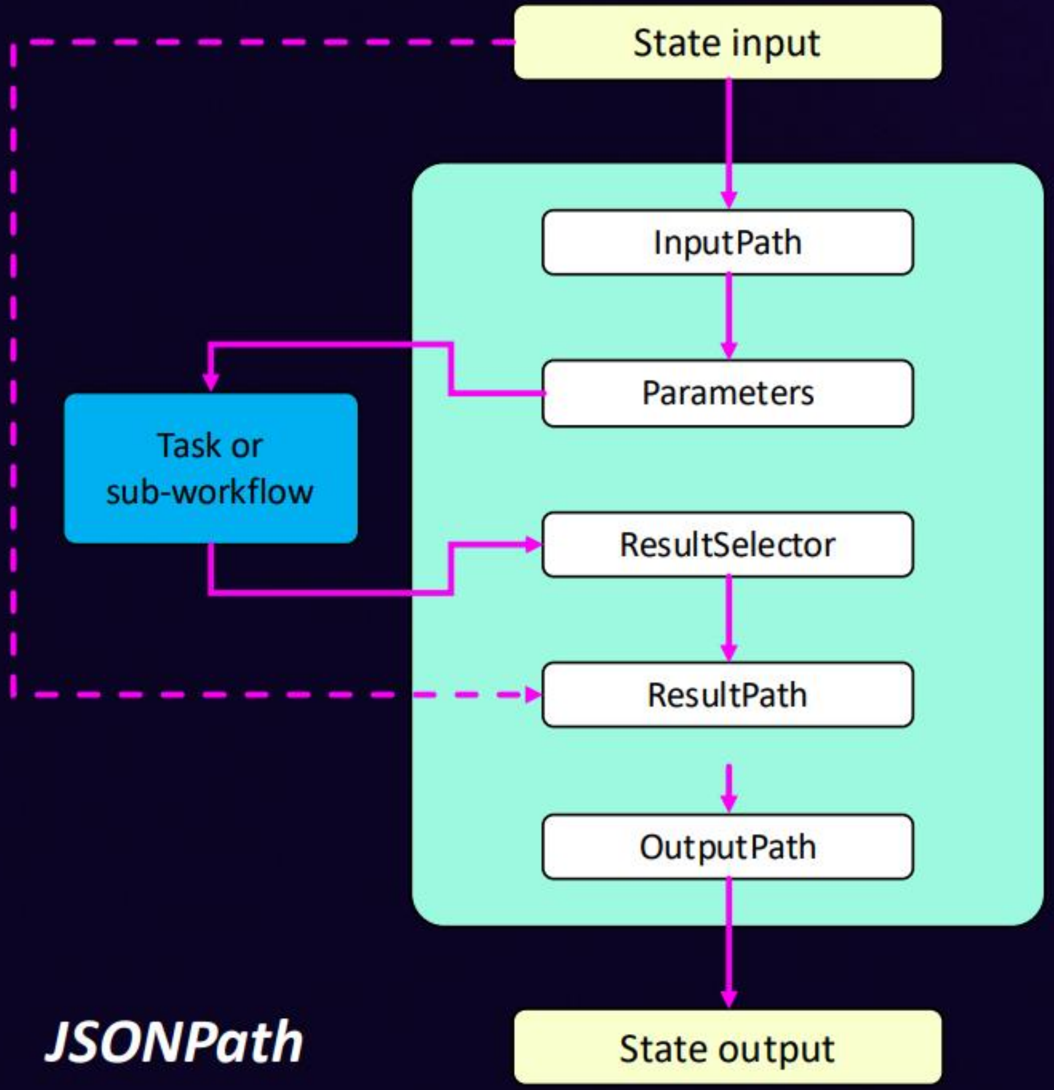

State machine query language | Info

[✓ JSONata - *recommended*]
All states and fields will require valid JSONata expressions for queries and data transformations.

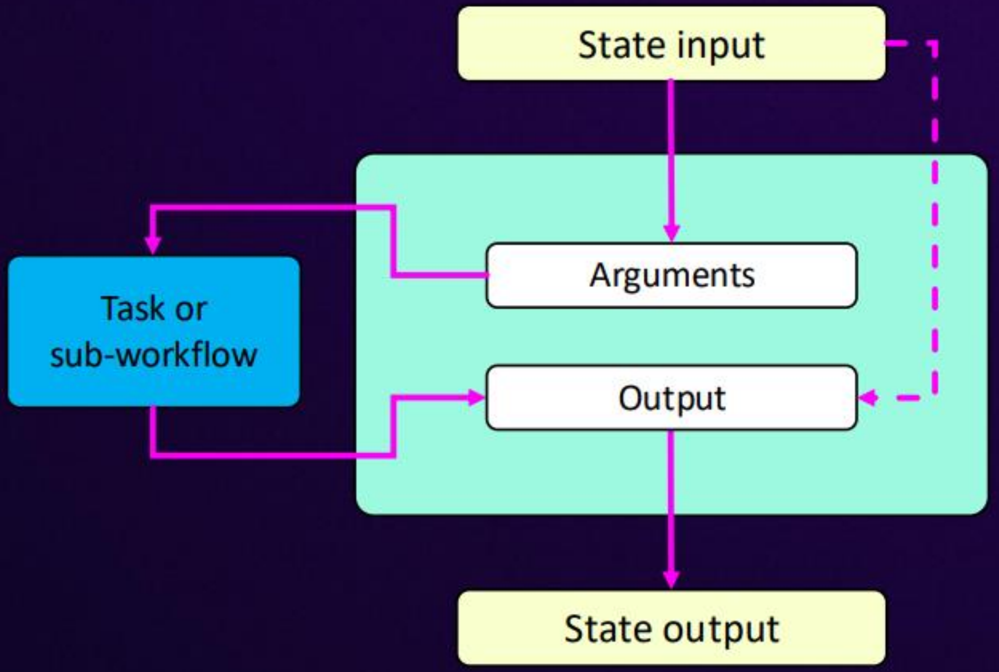
JSONPath
New states will default to JSONPath. You can convert to JSONata on a state-by-state basis.

New

Simplified task state



JSONPath



JSONata

New

JSONata syntax

```
{
  "FullName": "{% $states.input.FirstName & ' ' & $states.input.LastName %}"
}
```

Wrap the JSONata string in {% %}

Drop the use of .$ in the key name

New

JSONata reserved variables

```
{
  “$states”:  {
    “input”: “raw input to the state”,
    “result”: “Results from the task if successful”,
    “errorOutput”: “Results from task if errored”,
    “context”: “the context object”
  }
}
```

New

JSONata native functions

NON EXHAUSTIVE (NOT EVEN CLOSE)

String functions

$string()
$length()
$substring()
$substringBefore()
$substringAfter()
$uppercase()
$lowercase()
$trim()
$pad()
$contains()
$split()
$join()
$match()
$replace()
$eval()
$base64encode()
$base64decode()
$encodeUrlComponent()
$encodeUrl()
$decodeUrlComponent()
$decodeUrl()

Number functions

$number()
$abs()
$floor()
$ceil()
$round()
$power()
$sqrt()
$random()
$formatNumber()
$formatBase()
$formatInteger()
$parseInteger()

Numeric aggregation functions

$sum()
$max()
$min()
$average()

Array functions

$count()
$append()
$sort()
$reverse()
$shuffle()
$distinct()
$zip()

Object functions

$keys()
$lookup()
$spread()
$merge()
$sift()
$each()
$error()
$assert()
$type()

New

Step Function JSONata

$partition - partition a large array

$range - generate an array of values.

$hash - calculate the hash value of a given input.

$random - return a random number n where 0 ≤ n < 1

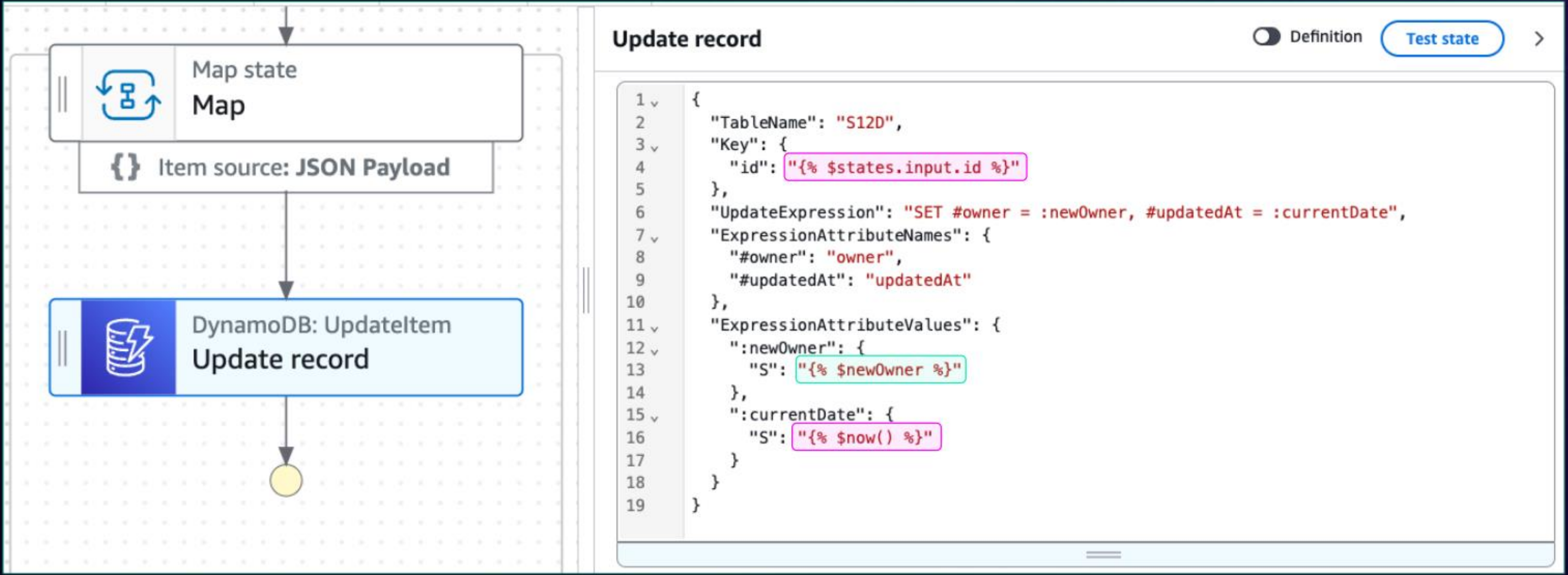
$uuid - generate a uuid

$parse - deserialize JSON strings

$now() - for timestamp generation

New

Step Function JSONata example



Update record

```
{
  "TableName": "S12D",
  "Key": {
    "id": "{% $states.input.id %}"
  },
  "UpdateExpression": "SET #owner = :newOwner, #updatedAt = :currentDate",
  "ExpressionAttributeNames": {
    "#owner": "owner",
    "#updatedAt": "updatedAt"
  },
  "ExpressionAttributeValues": {
    ":newOwner": {
      "S": "{% $newOwner %}"
    },
    ":currentDate": {
      "S": "{% $now() %}"
    }
  }
}
```

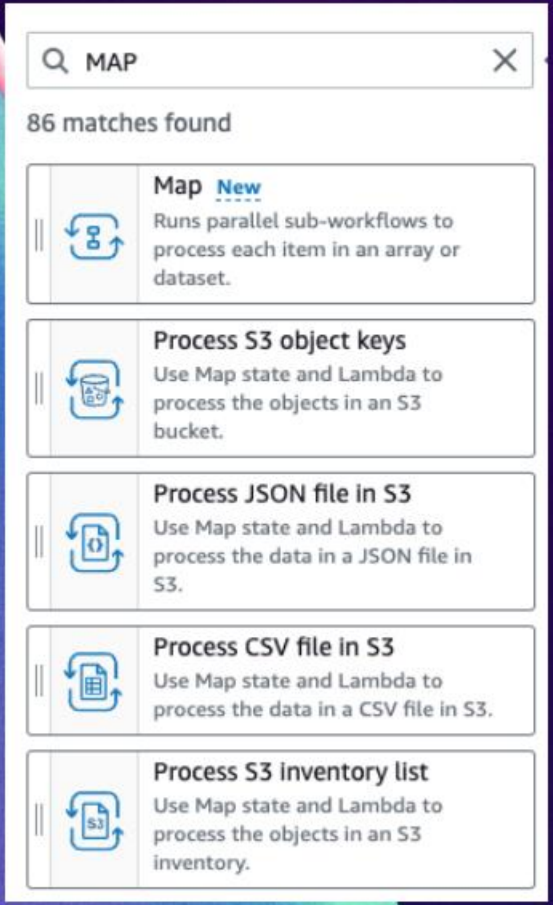
Distributed Map state

Coordinate large-scale parallel workloads

Iterate over millions of Amazon S3 objects, such as logs, images, or JSON or CSV files

Up to 10,000 parallel executions

Invoke Lambda or Amazon ECS/AWS Fargate technology for large-scale on-demand serverless compute



MAP

86 matches found

Map New
Runs parallel sub-workflows to process each item in an array or dataset.

Process S3 object keys
Use Map state and Lambda to process the objects in an S3 bucket.

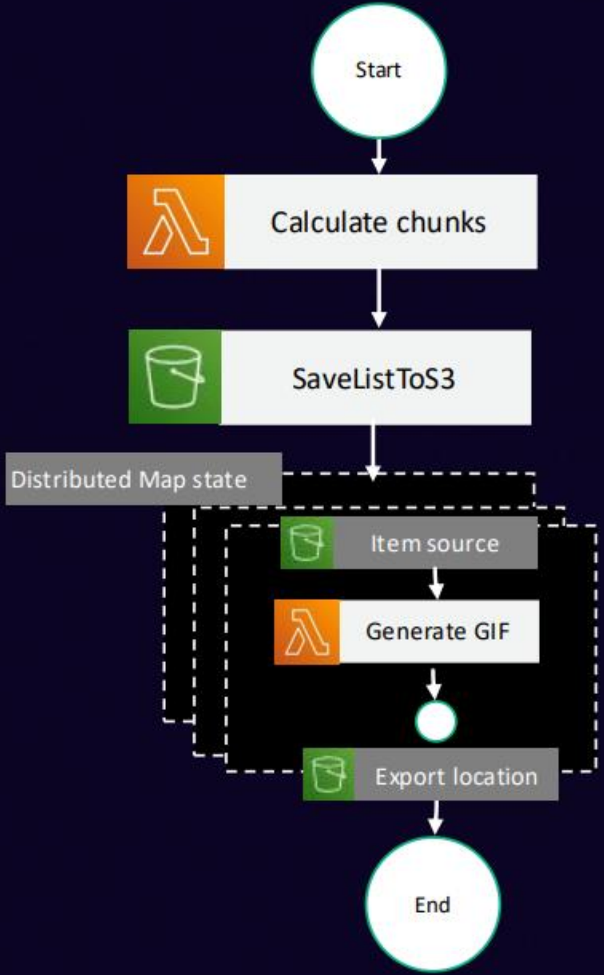
Process JSON file in S3
Use Map state and Lambda to process the data in a JSON file in S3.

Process CSV file in S3
Use Map state and Lambda to process the data in a CSV file in S3.

Process S3 inventory list
Use Map state and Lambda to process the objects in an S3 inventory.

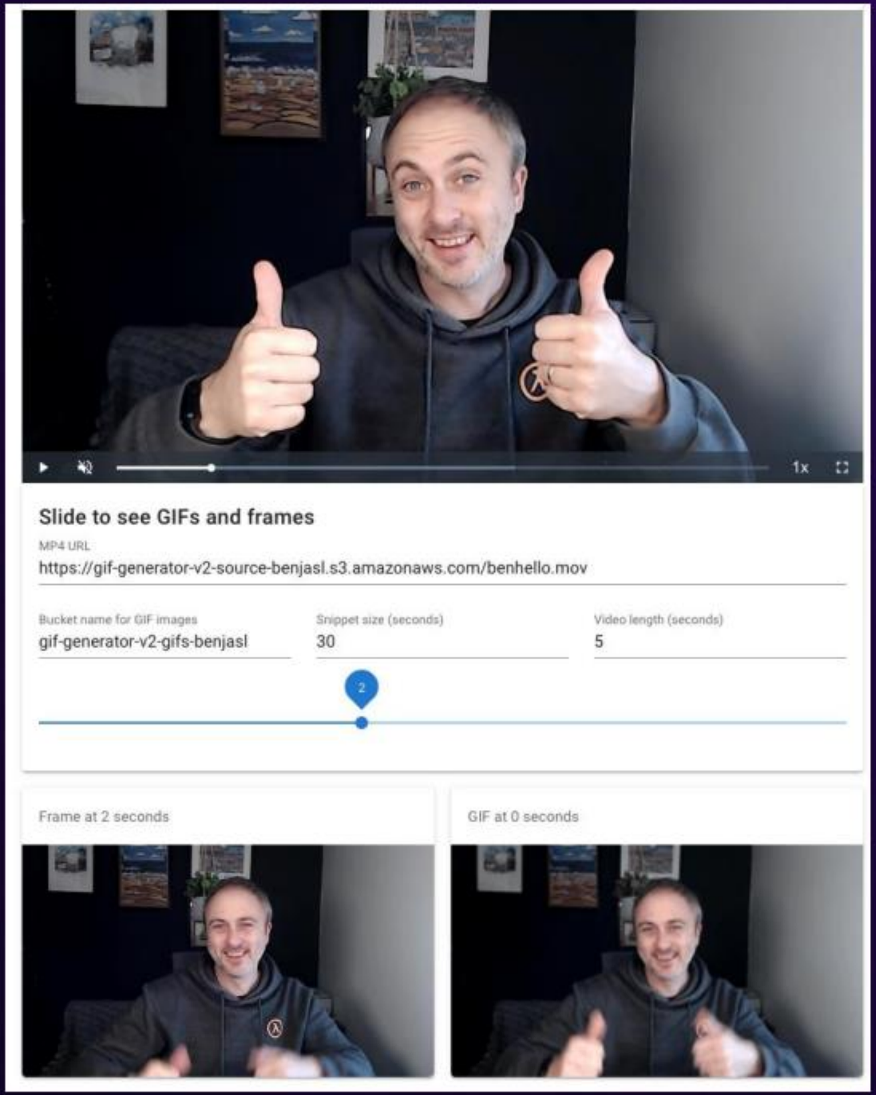
Step Functions Distributed Map

Each iteration runs as a separate child workflow with its own execution history and payload limits



A serverless GIF generator

Convert an .mp4 from Amazon S3 into multiple GIFs for timeline scrubbing


Slide to see GIFs and frames

MP4 URL
https://gif-generator-v2-source-benjasl.s3.amazonaws.com/benhello.mov

Bucket name for GIF images
gif-generator-v2-gifs-benjasl

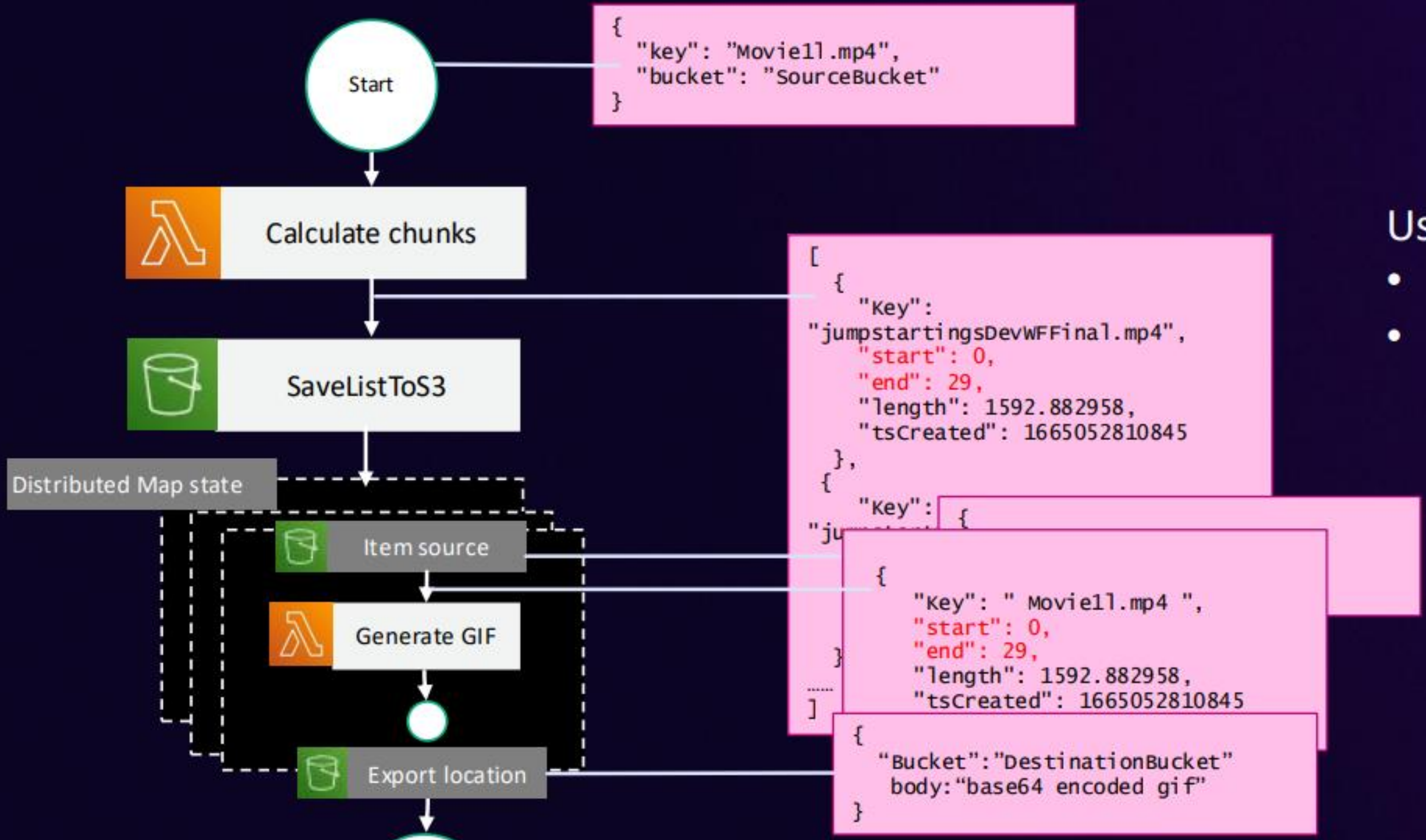
Snippet size (seconds)
30

Video length (seconds)
5

Frame at 2 seconds

GIF at 0 seconds

Scalability



```
{
  "key": "Movie11.mp4",
  "bucket": "SourceBucket"
}
```

```
[
  {
    "Key":
"jumpstartingsDevWFFinal.mp4",
    "start": 0,
    "end": 29,
    "length": 1592.882958,
    "tsCreated": 1665052810845
  },
 {
    "Key":
"ju
  }
......
]
```

```
{
```

```
  {
    "Key": " Movie11.mp4 ",
    "start": 0,
    "end": 29,
    "length": 1592.882958,
    "tsCreated": 1665052810845
```

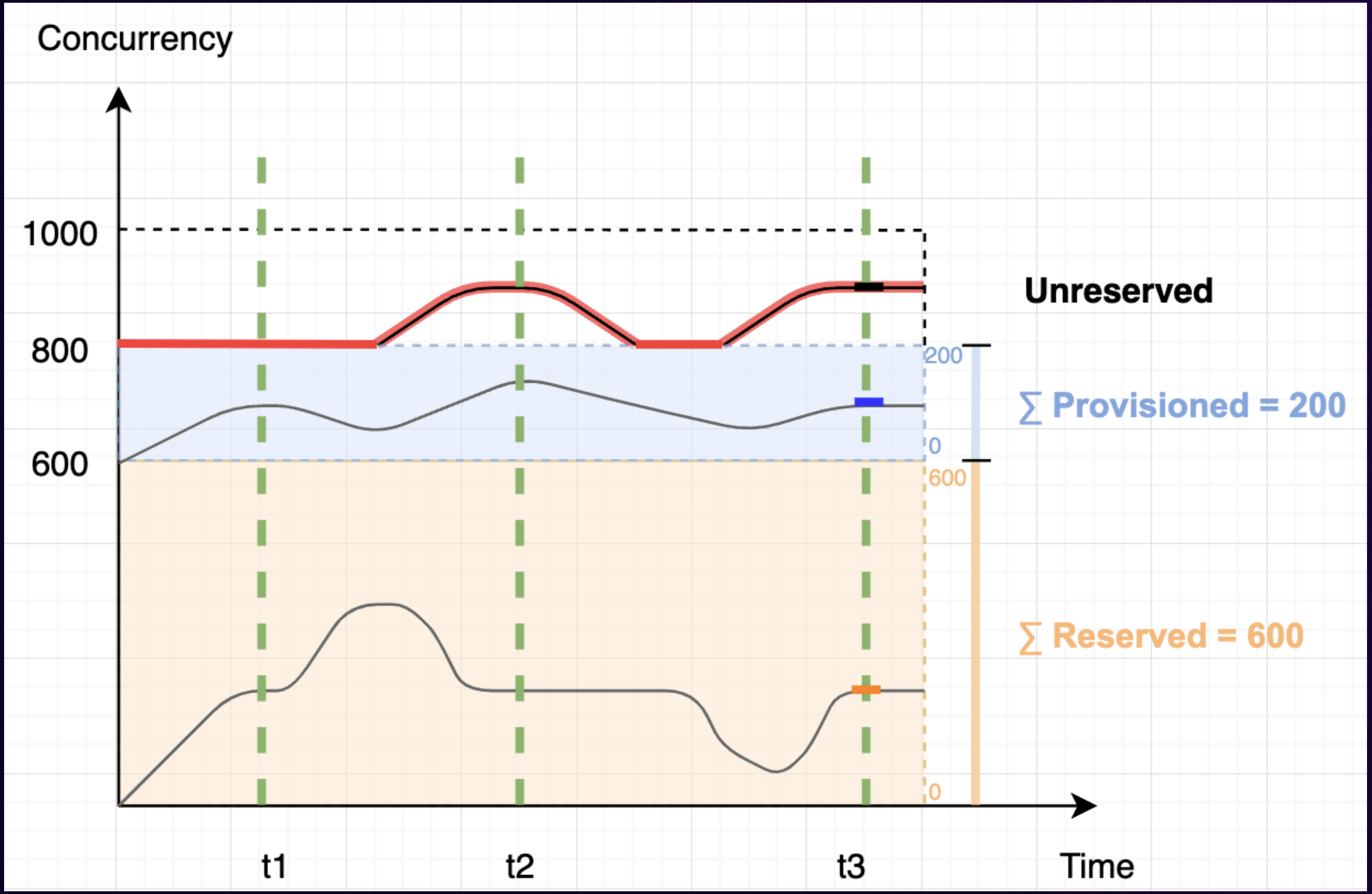
```
{
  "Bucket":"DestinationBucket"
  body:"base64 encoded gif"
}
```

Using a Distributed Map state for this workload

- Not limited to 40 concurrent executions
- Imports and exports directly from/to Amazon S3

Scalability

Why is Distributed MAP state a big deal?



See https://docs.aws.amazon.com/lambda/latest/dg/lambda-concurrency.html#concurrency-quotas

Thank you!

Håkon Eriksen Drange

https://hedrange.com/

https://www.linkedin.com/in/haakondrange/