SERVERLESS BOSTON & NYC

AWS STEP FUNCTIONS DEEP DIVE

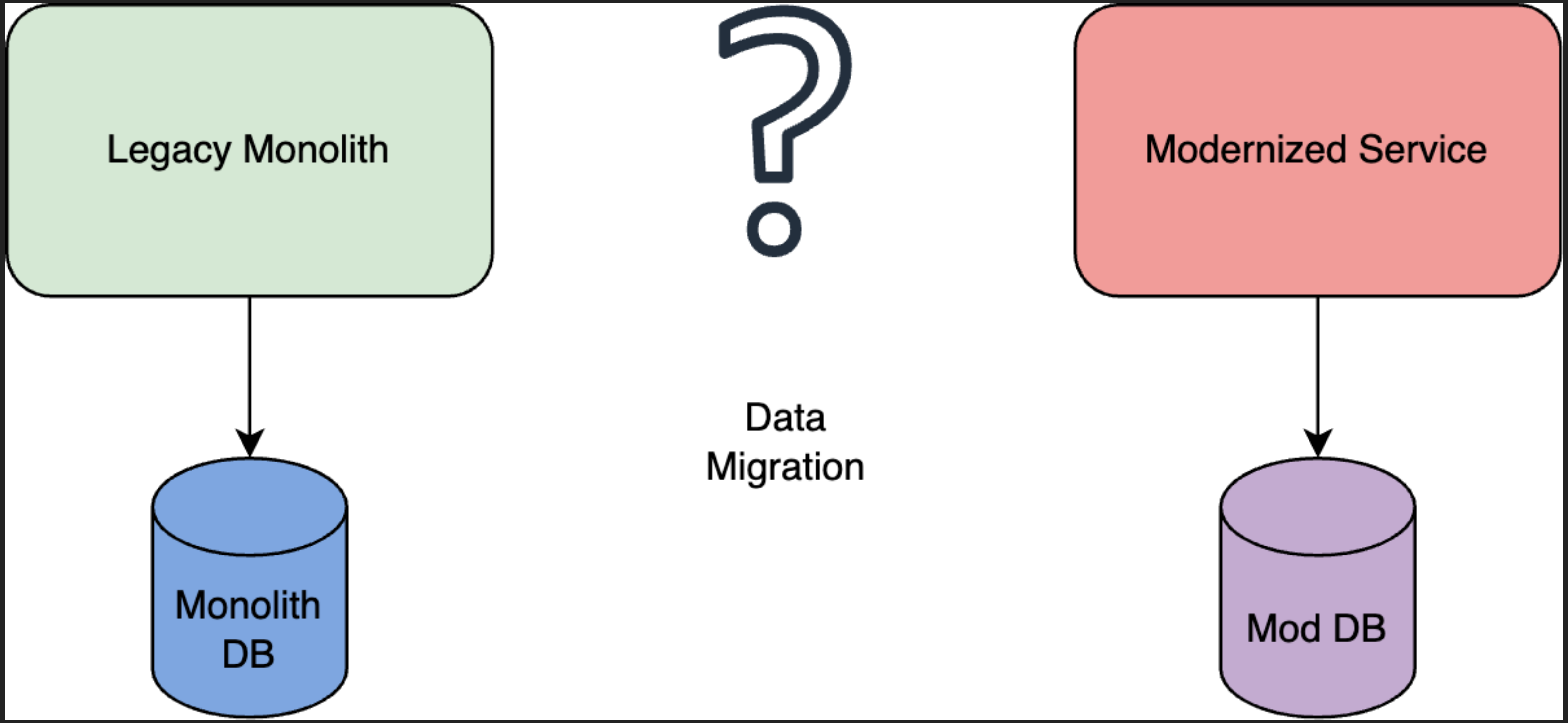
AGENDA

- Why Step Functions?
- Complex Workflows & Observability
- Error-handling & Retry
- Synchronous Invocation & Express Workflows
- Intrinsics & Service Integrations
- Task Tokens & Wait
- Map & Parallel States
- The Learning Curve
- Summary & Discussion

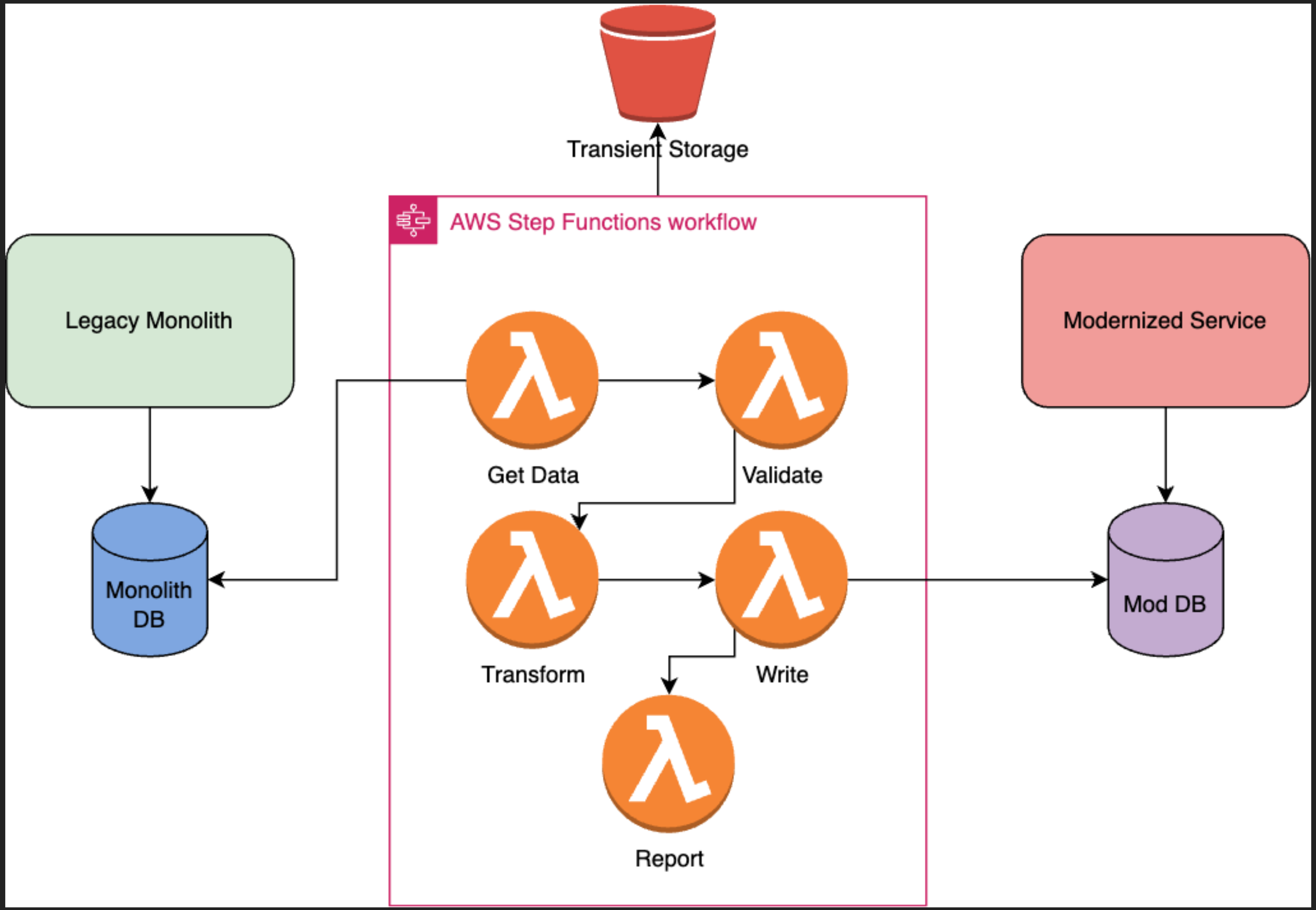
WHY STEP FUNCTIONS?

- Started using Step Functions in 2019 to handle complex data migrations
- Used for reporting engine on top of multiple databases, different printing options, and fan-out
- Step Functions let us structure, control, and monitor the steps in these workflows

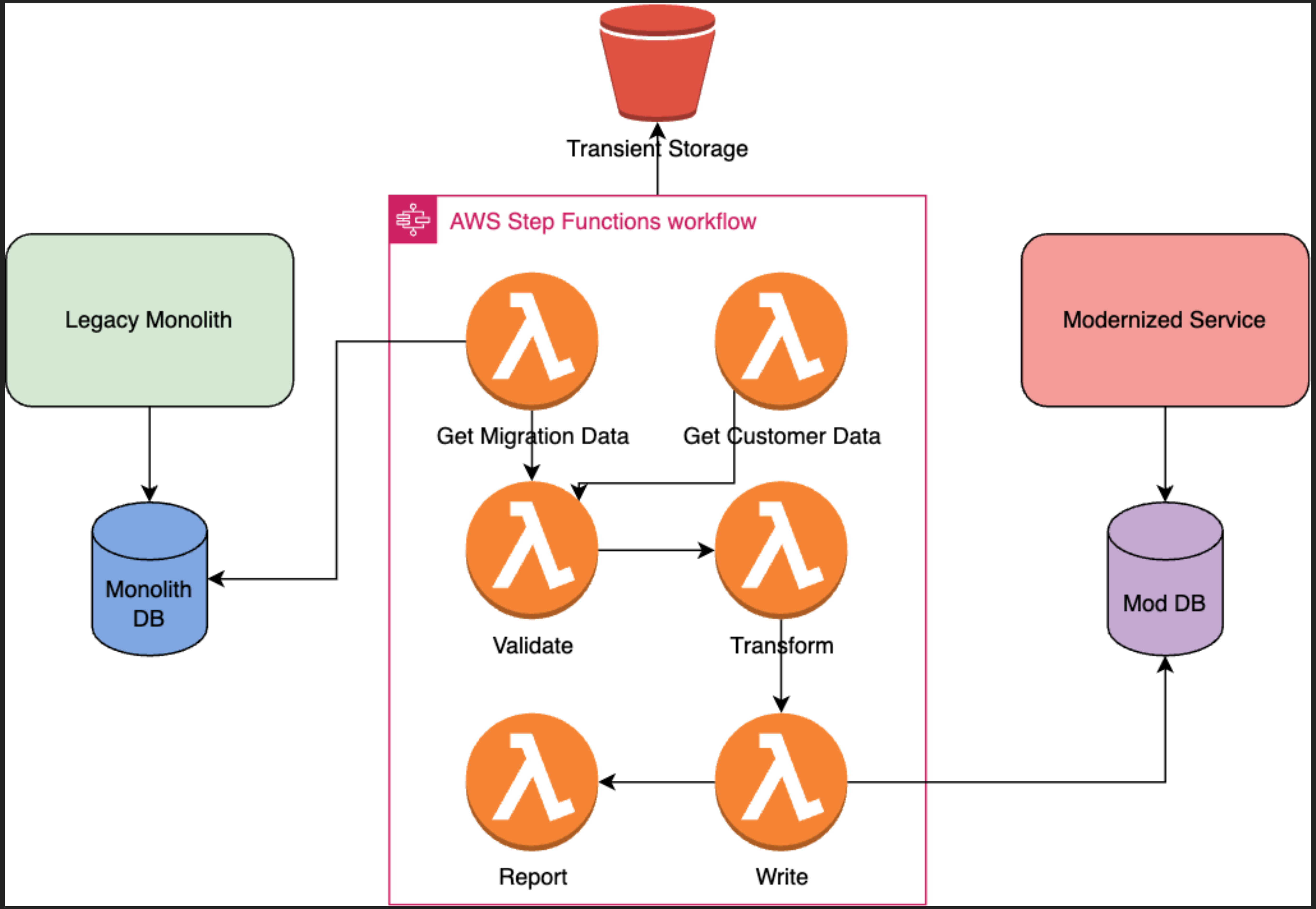
WHY STEP FUNCTIONS?



WHY STEP FUNCTIONS?



WHY STEP FUNCTIONS?



WHY STEP FUNCTIONS?

- Defined start and end for a workflow
- Fan-out and fan-in with a managed service
- Error-handling and retry (because "Everything fails all the time")
- Small and replaceable chunks of code
- Worth the learning curve

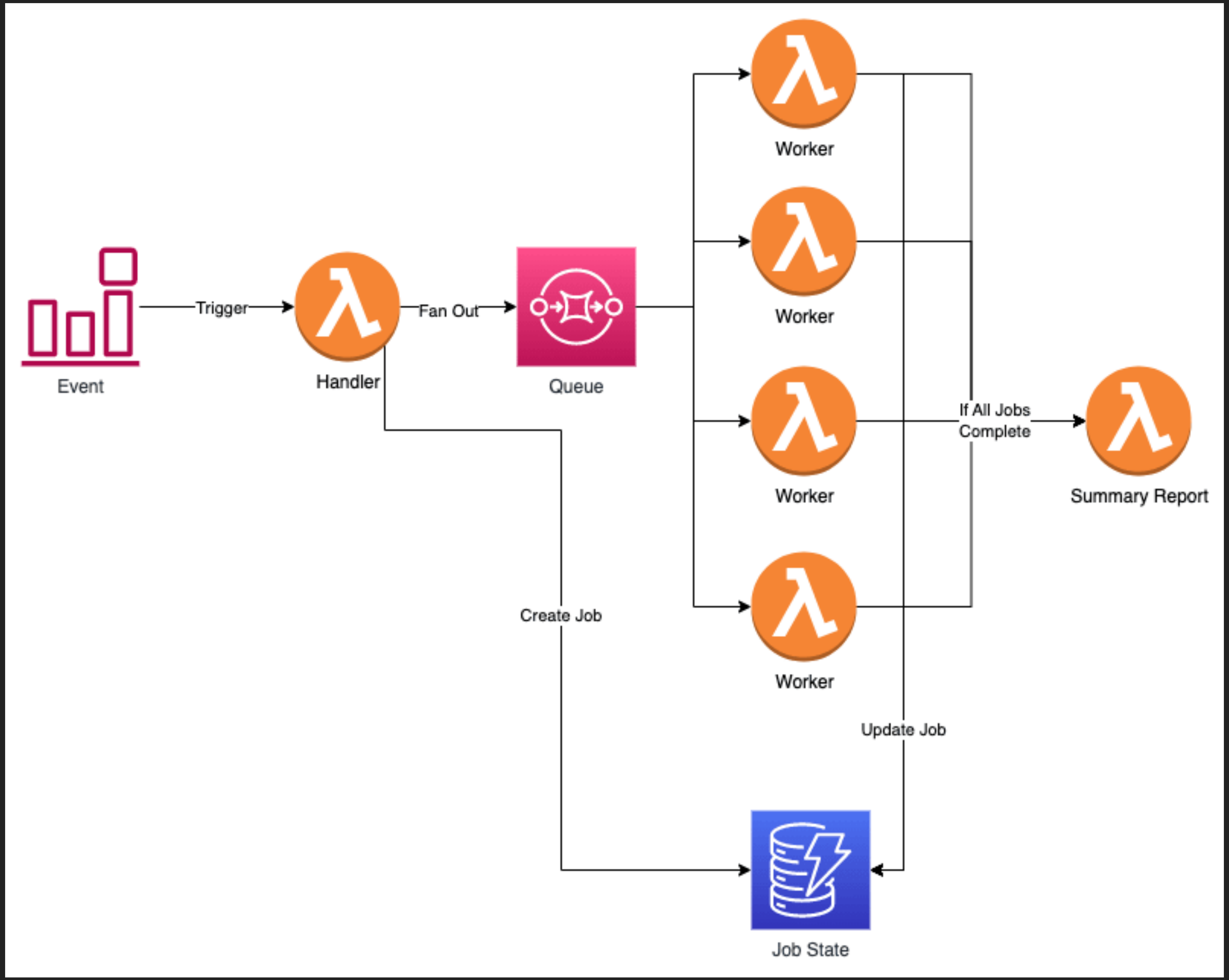
Matt Morgan for AWS Community Builders
Posted on Feb 27

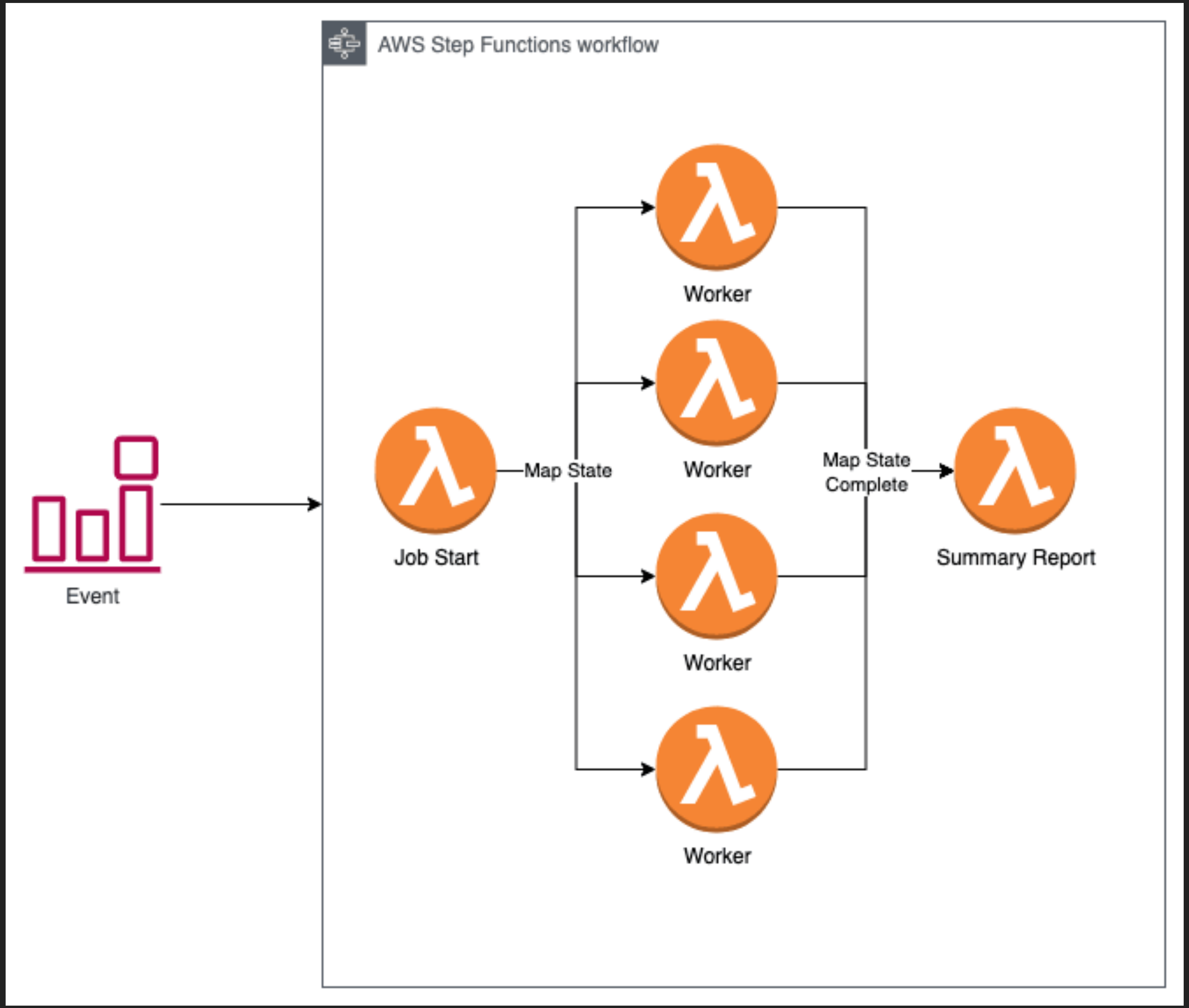
❤️ 30 🦄 2 🤯 1 🙌 1 🔥 2

Avoiding the Serverless Workflow Antipattern

#aws #serverless #stepfunctions #sqs

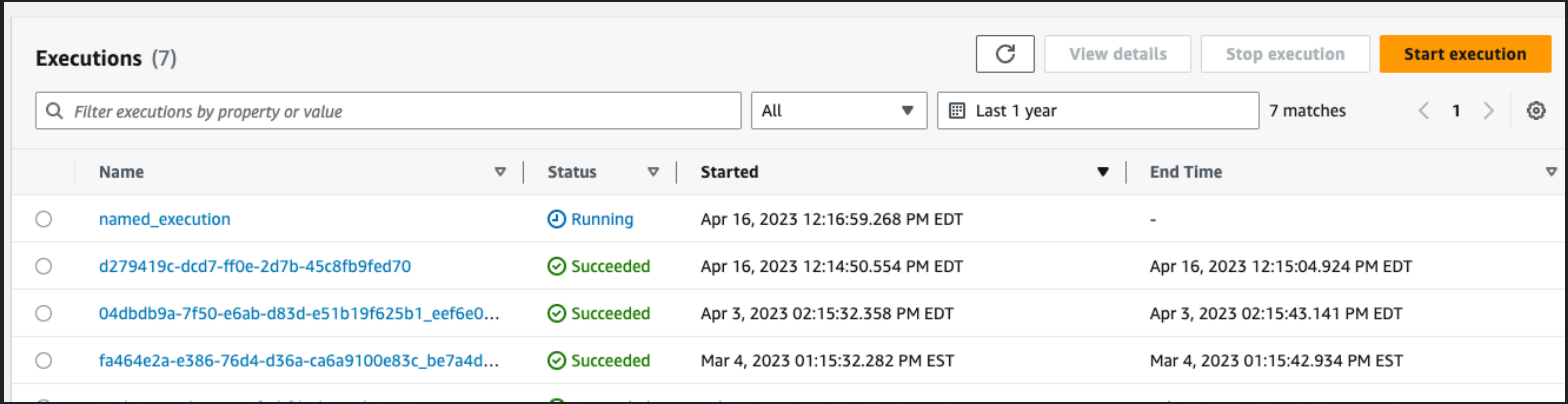
WHY STEP FUNCTIONS?





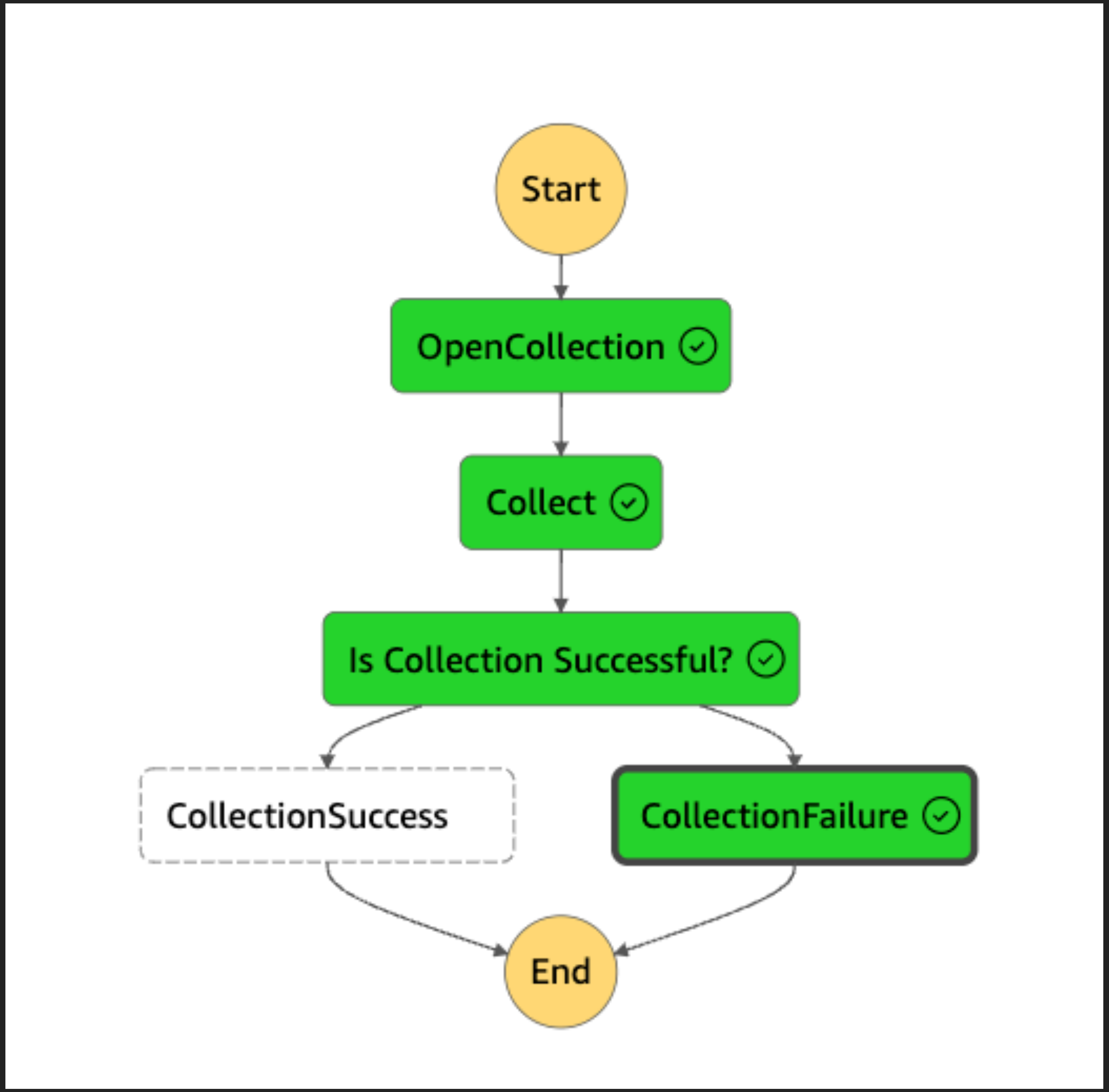
COMPLEX WORKFLOWS & OBSERVABILITY

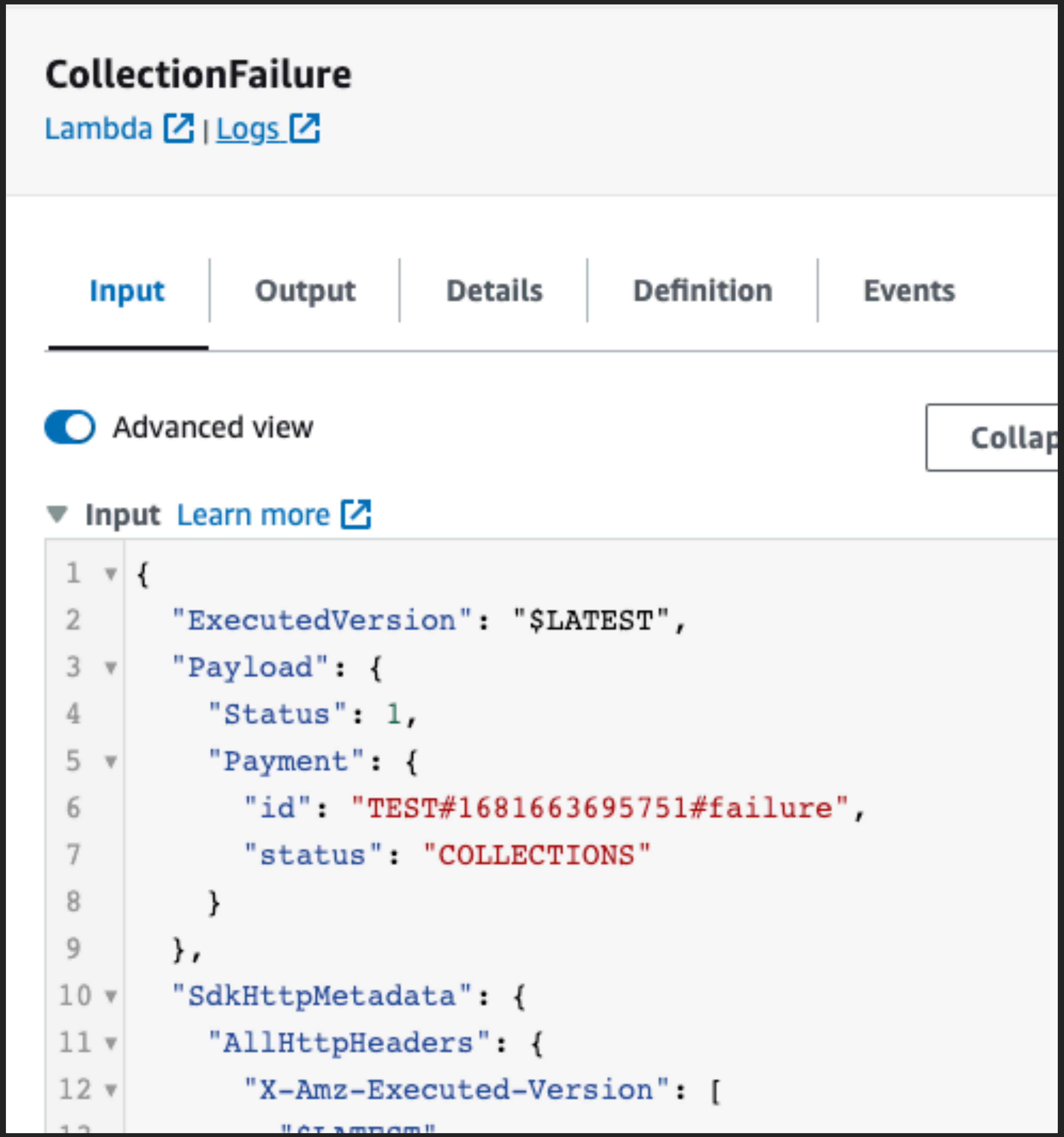
- RUNNING | SUCCEEDED | FAILED | TIMED_OUT | ABORTED



Name	Status	Started	End Time
named_execution	Running	Apr 16, 2023 12:16:59.268 PM EDT	-
d279419c-dcd7-ff0e-2d7b-45c8fb9fed70	Succeeded	Apr 16, 2023 12:14:50.554 PM EDT	Apr 16, 2023 12:15:04.924 PM EDT
04dbdb9a-7f50-e6ab-d83d-e51b19f625b1_eef6e0...	Succeeded	Apr 3, 2023 02:15:32.358 PM EDT	Apr 3, 2023 02:15:43.141 PM EDT
fa464e2a-e386-76d4-d36a-ca6a9100e83c_be7a4d...	Succeeded	Mar 4, 2023 01:15:32.282 PM EST	Mar 4, 2023 01:15:42.934 PM EST

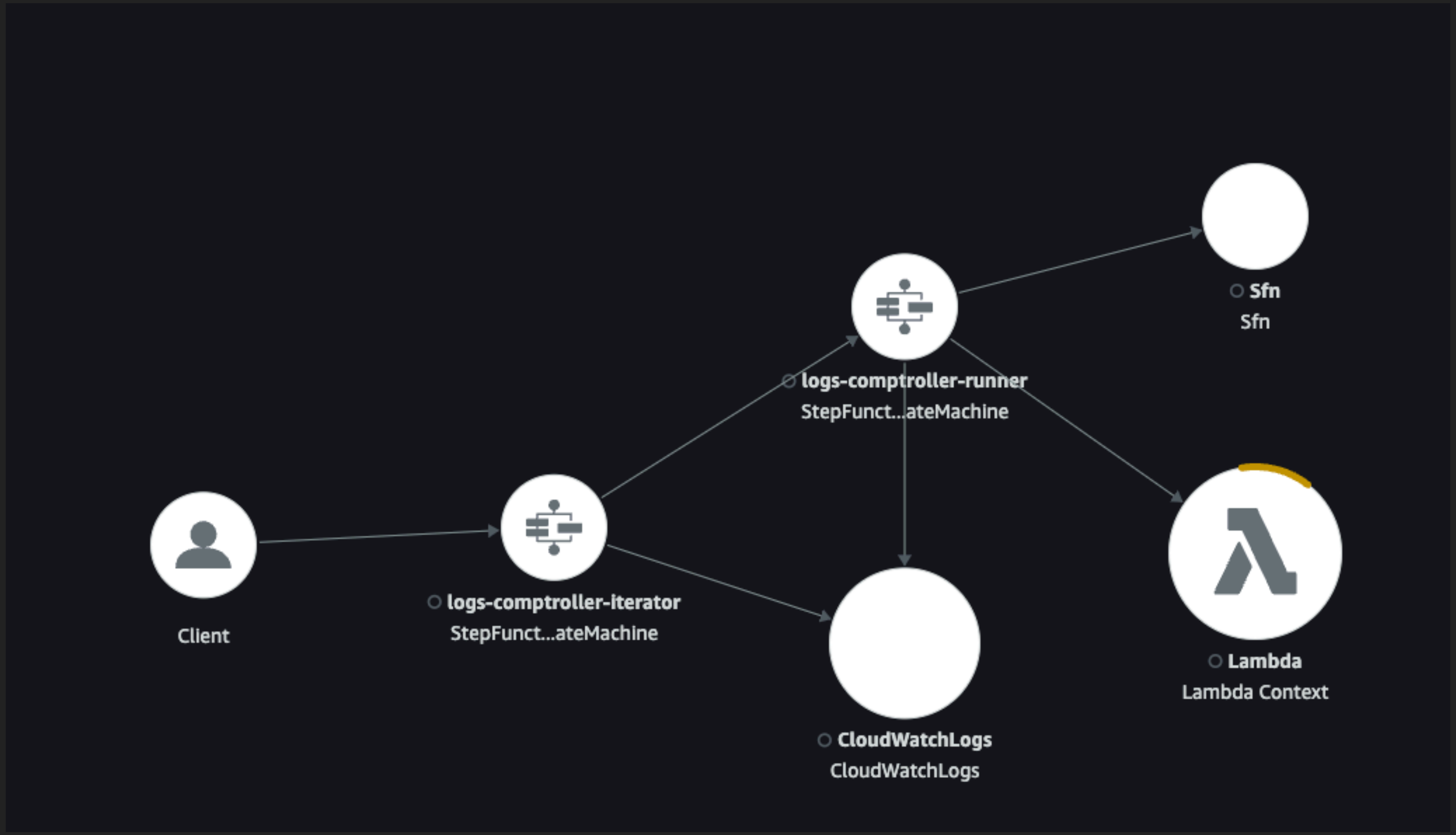
COMPLEX WORKFLOWS & OBSERVABILITY

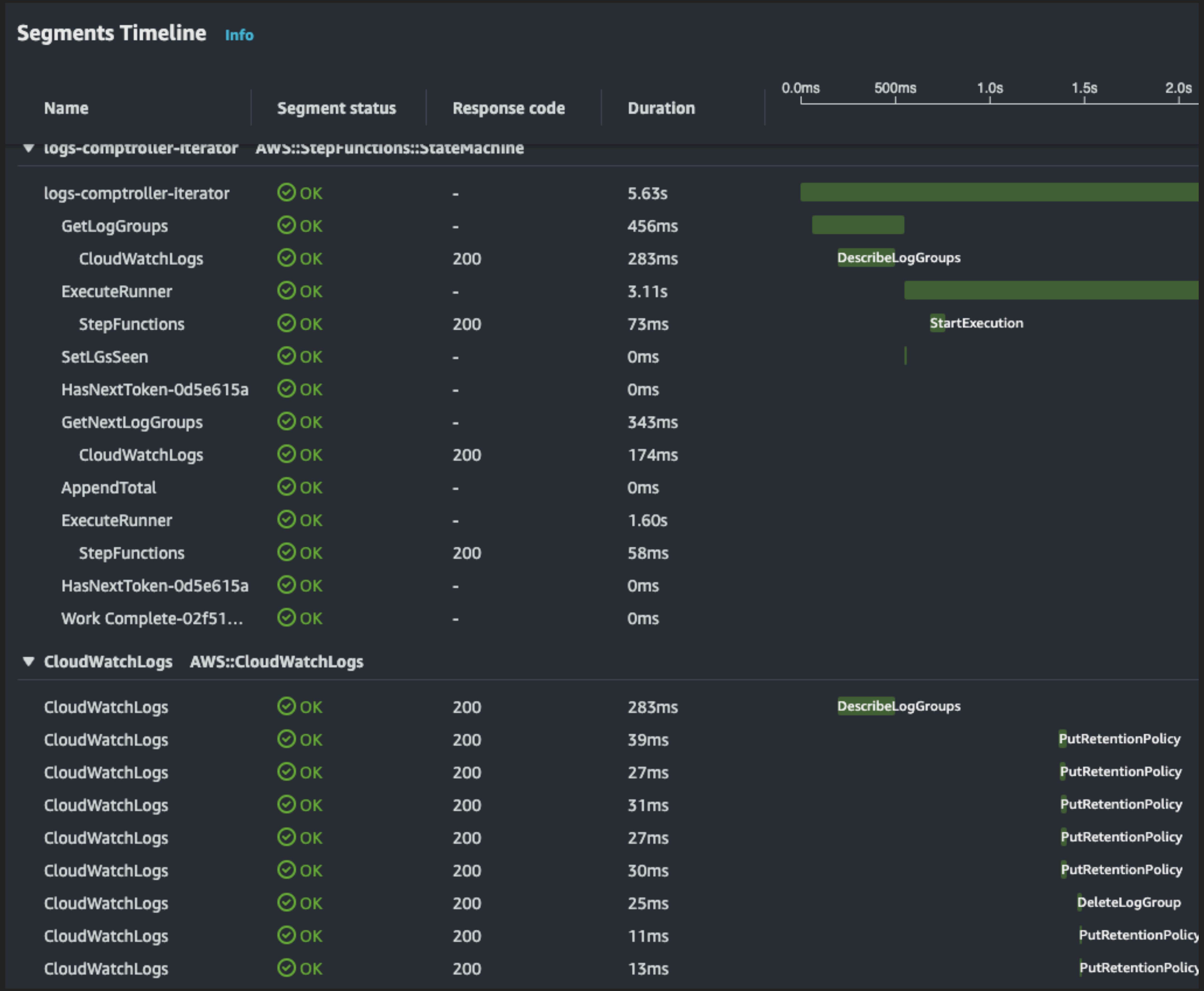




```
{
  "ExecutedVersion": "$LATEST",
  "Payload": {
    "Status": 1,
    "Payment": {
      "id": "TEST#1681663695751#failure",
      "status": "COLLECTIONS"
    }
  },
  "SdkHttpMetadata": {
    "AllHttpHeaders": {
      "X-Amz-Executed-Version": [
```

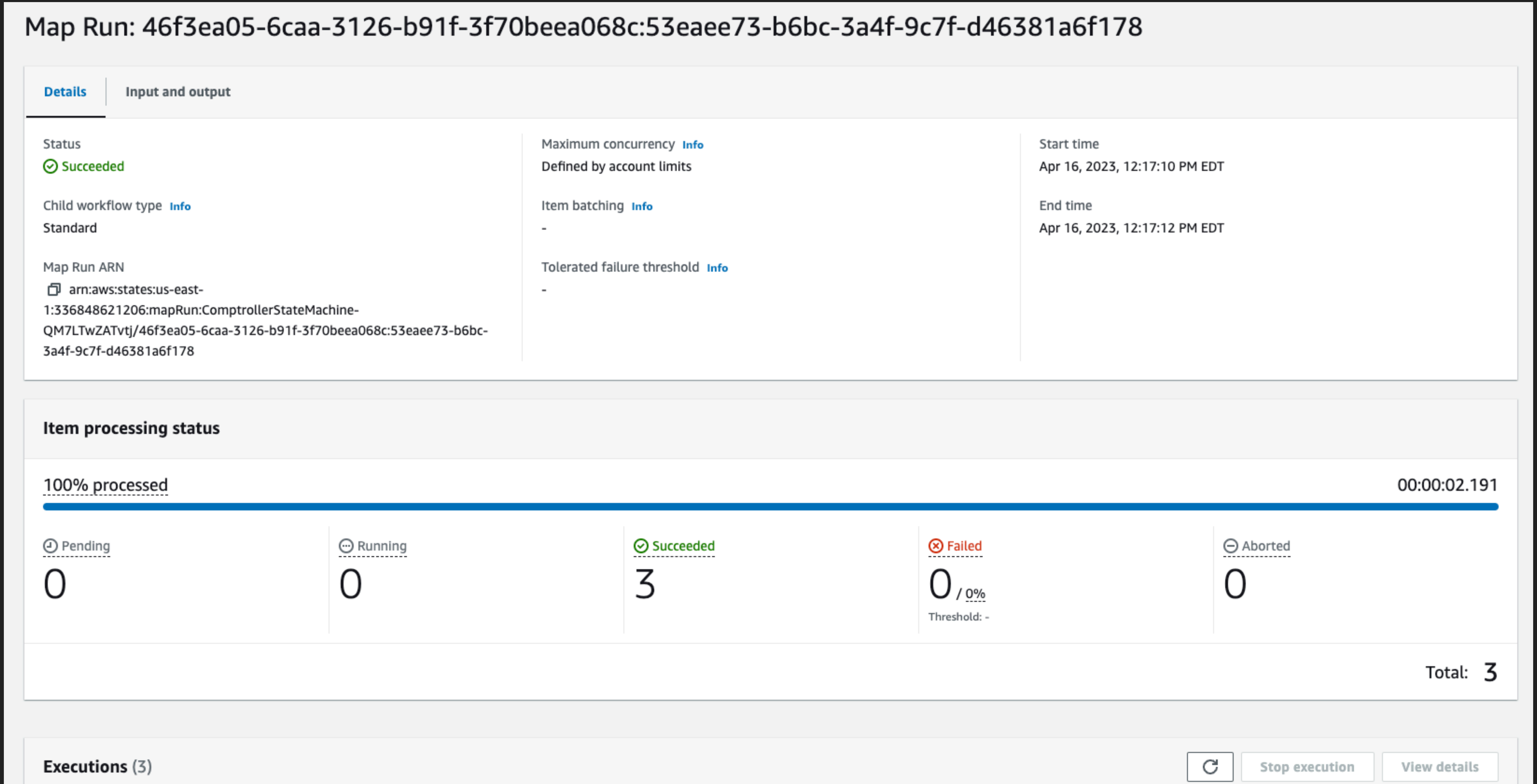
COMPLEX WORKFLOWS & OBSERVABILITY




Segments Timeline Info

Name	Segment status	Response code	Duration	0.0ms 500ms 1.0s 1.5s 2.0s
▼ logs-comptroller-iterator AWS::StepFunctions::StateMachine				
logs-comptroller-iterator	OK	-	5.63s	
GetLogGroups	OK	-	456ms	
CloudWatchLogs	OK	200	283ms	DescribeLogGroups
ExecuteRunner	OK	-	3.11s	
StepFunctions	OK	200	73ms	StartExecution
SetLGsSeen	OK	-	0ms	
HasNextToken-0d5e615a	OK	-	0ms	
GetNextLogGroups	OK	-	343ms	
CloudWatchLogs	OK	200	174ms	
AppendTotal	OK	-	0ms	
ExecuteRunner	OK	-	1.60s	
StepFunctions	OK	200	58ms	
HasNextToken-0d5e615a	OK	-	0ms	
Work Complete-02f51...	OK	-	0ms	
▼ CloudWatchLogs AWS::CloudWatchLogs				
CloudWatchLogs	OK	200	283ms	DescribeLogGroups
CloudWatchLogs	OK	200	39ms	PutRetentionPolicy
CloudWatchLogs	OK	200	27ms	PutRetentionPolicy
CloudWatchLogs	OK	200	31ms	PutRetentionPolicy
CloudWatchLogs	OK	200	27ms	PutRetentionPolicy
CloudWatchLogs	OK	200	30ms	PutRetentionPolicy
CloudWatchLogs	OK	200	25ms	DeleteLogGroup
CloudWatchLogs	OK	200	11ms	PutRetentionPolicy
CloudWatchLogs	OK	200	13ms	PutRetentionPolicy

COMPLEX WORKFLOWS & OBSERVABILITY



Map Run: 46f3ea05-6caa-3126-b91f-3f70beea068c:53eaee73-b6bc-3a4f-9c7f-d46381a6f178

Status
Succeeded

Child workflow type Info
Standard

Map Run ARN
arn:aws:states:us-east-1:336848621206:mapRun:ComptrollerStateMachine-QM7LTwZATvtj/46f3ea05-6caa-3126-b91f-3f70beea068c:53eaee73-b6bc-3a4f-9c7f-d46381a6f178

Maximum concurrency Info
Defined by account limits

Item batching Info
-

Tolerated failure threshold Info
-

Start time
Apr 16, 2023, 12:17:10 PM EDT

End time
Apr 16, 2023, 12:17:12 PM EDT

Item processing status

100% processed 00:00:02.191

Pending	Running	Succeeded	Failed	Aborted
0	0	3	0 / 0% Threshold: -	0

Total: 3

Executions (3)

ERROR-HANDLING & RETRY

- A word on idempotence
- “Everything fails all the time”
- Retry & backoff
- Error-handling is at the Task, Parallel, and Map states (no global catch at the moment)
- Sample code will retry after 1, 2, 4, 8, 16, 32, 64, 128, 256 and 512 seconds.

```
"AddRetention": {
    "Retry": [
      {
        "ErrorEquals": [
          "States.ALL"
        ],
        "MaxAttempts": 10
      }
    ],
    "Type": "Task",
    "Resource": "arn:aws:states:::aws-sdk:cloudwatchlogs:putRetentionPolicy",
    "Parameters": {
      "LogGroupName.$": "$.LogGroupName",
      "RetentionInDays": 1
    }
  }
```

ERROR-HANDLING & RETRY

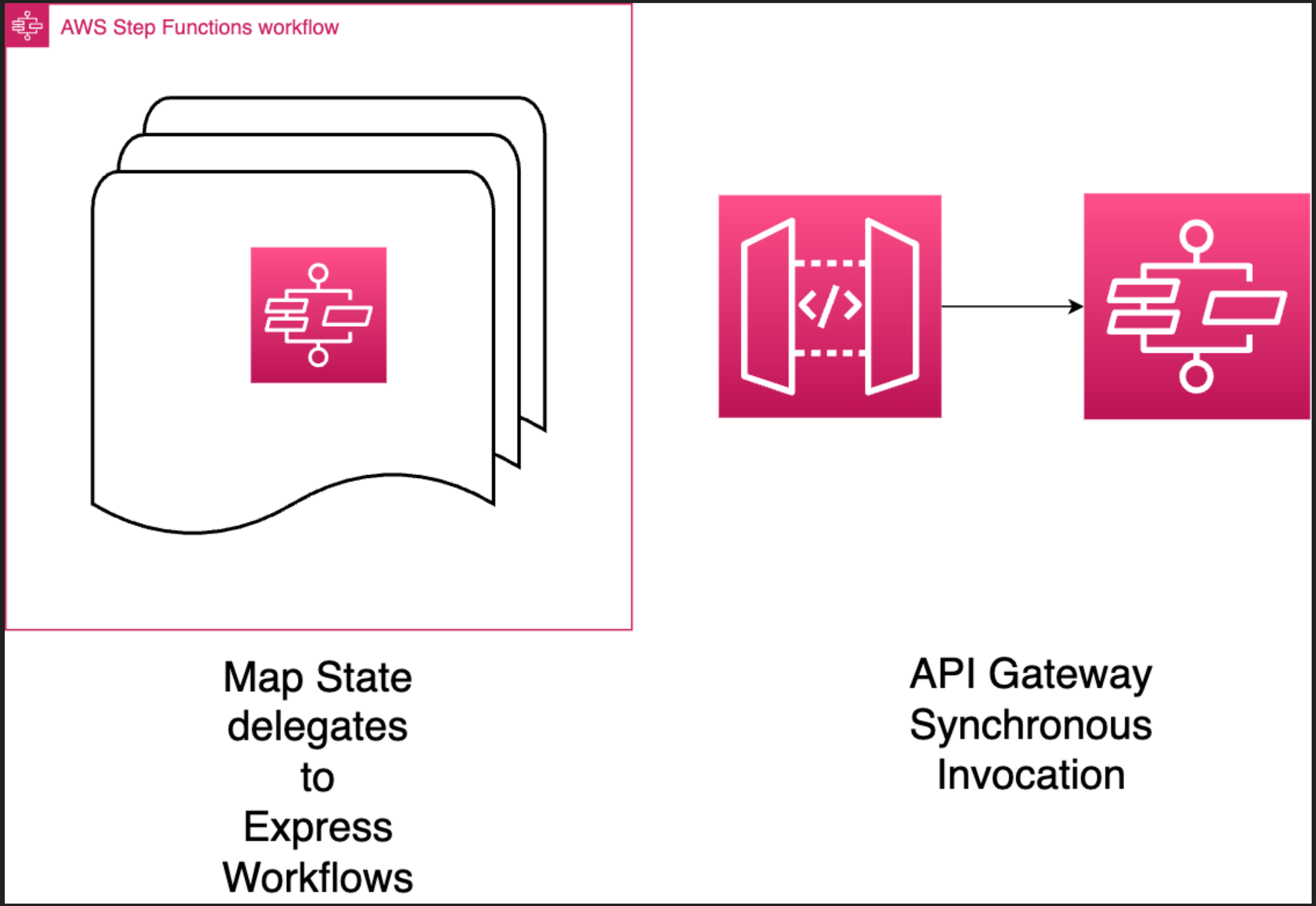
- Catch and Retry pair well with service integrations and predictable errors.
- “Single-branch parallel” for global error-handling - #awswishlist for proper global error-handling
- Uncaught errors will cause execution failure

```
"DeleteLogGroup": {
        "Catch": [
         {
           "ErrorEquals": [
             "CloudWatchLogs.ResourceNotFoundException"
           ],
          }
        ],
       "Retry": [
         {
           "ErrorEquals": [
             "CloudWatchLogs.CloudWatchLogsException",
             "CloudWatchLogs.OperationAbortedException"
           ],
           "MaxAttempts": 10
         }
        ],
        "Resource": "arn:aws:states:::aws-sdk:cloudwatchlogs:deleteLogGroup",
        "Parameters": {
          "LogGroupName.$": "$.LogGroupName"
        }
}
```

SYNCHRONOUS INVOCATION & EXPRESS WORKFLOWS

- Sometimes async doesn’t do it
- Break down complex workflows with a synchronous invocation
- Express workflows has a different billing model and different limitations
- You can combine both Standard and Express workflows and gain the best of both worlds!

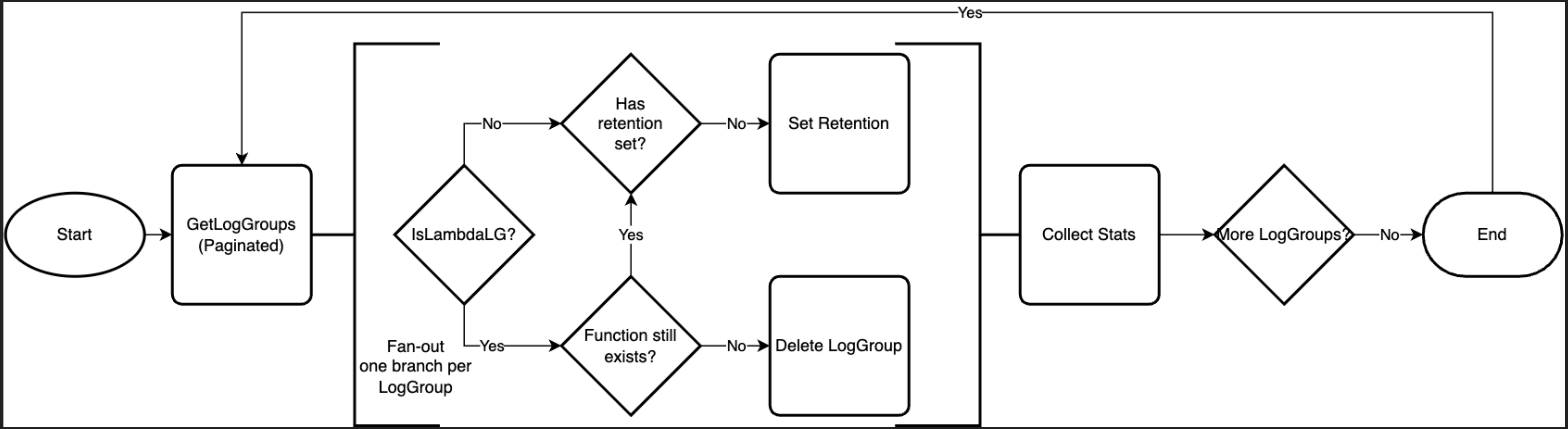
SYNCHRONOUS INVOCATION & EXPRESS WORKFLOWS


AWS Step Functions workflow
Map State delegates to Express Workflows
API Gateway Synchronous Invocation

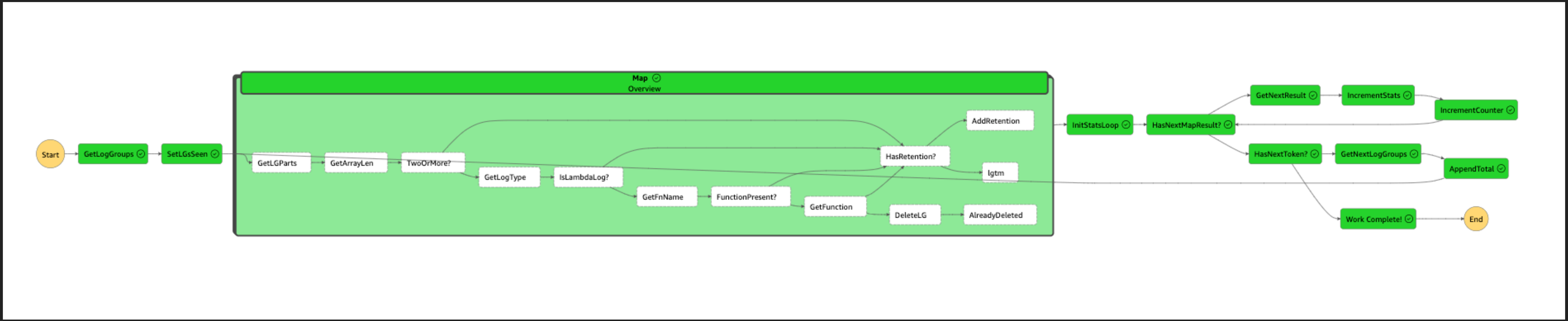
INTRINSICS & SERVICE INTEGRATIONS

- No cold start or other overhead
- No runtime sdk abstraction
- No runtime deprecation
- No swallowed errors
- Great for distributed state machines!

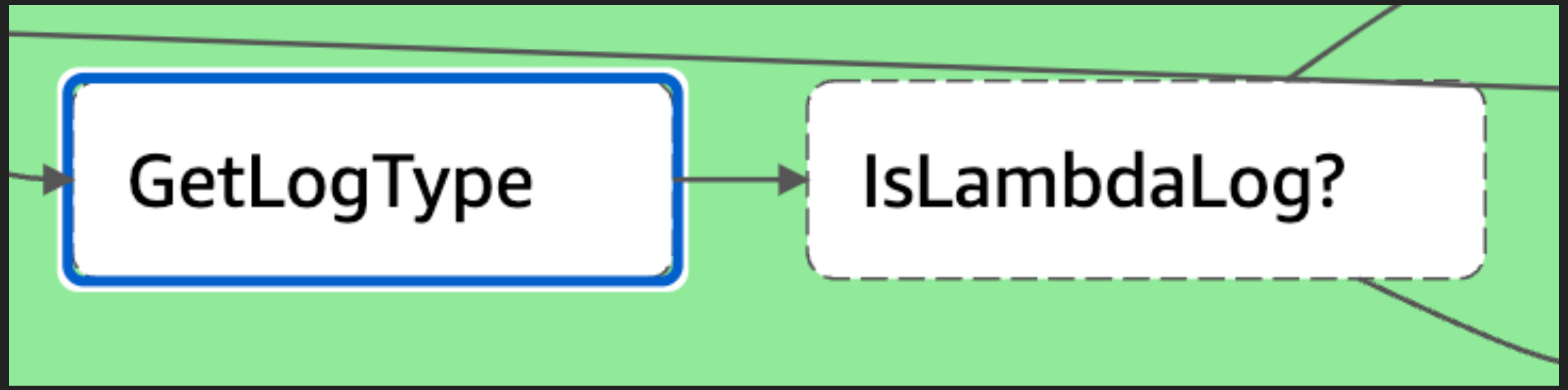
INTRINSICS & SERVICE INTEGRATIONS

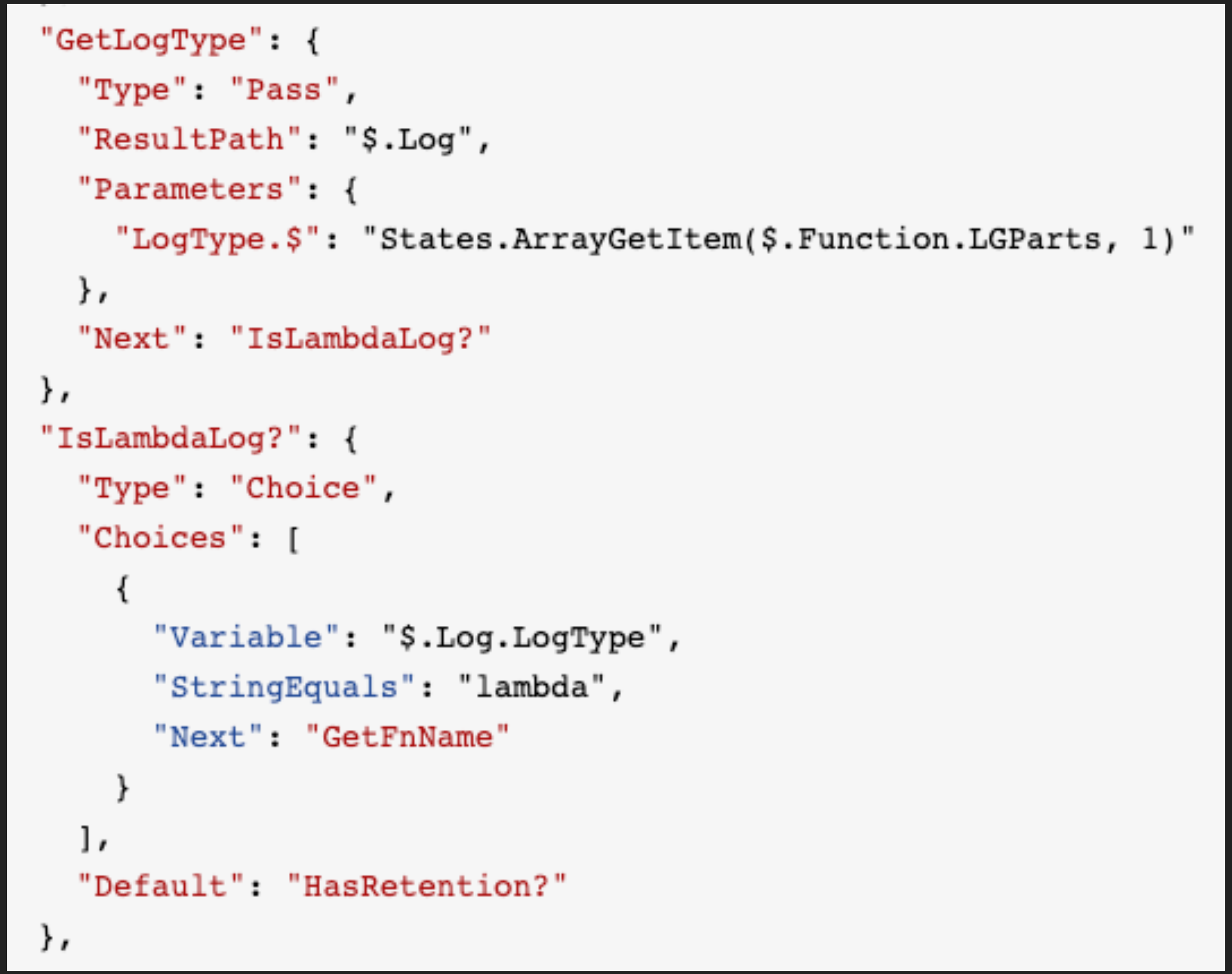


INTRINSICS & SERVICE INTEGRATIONS



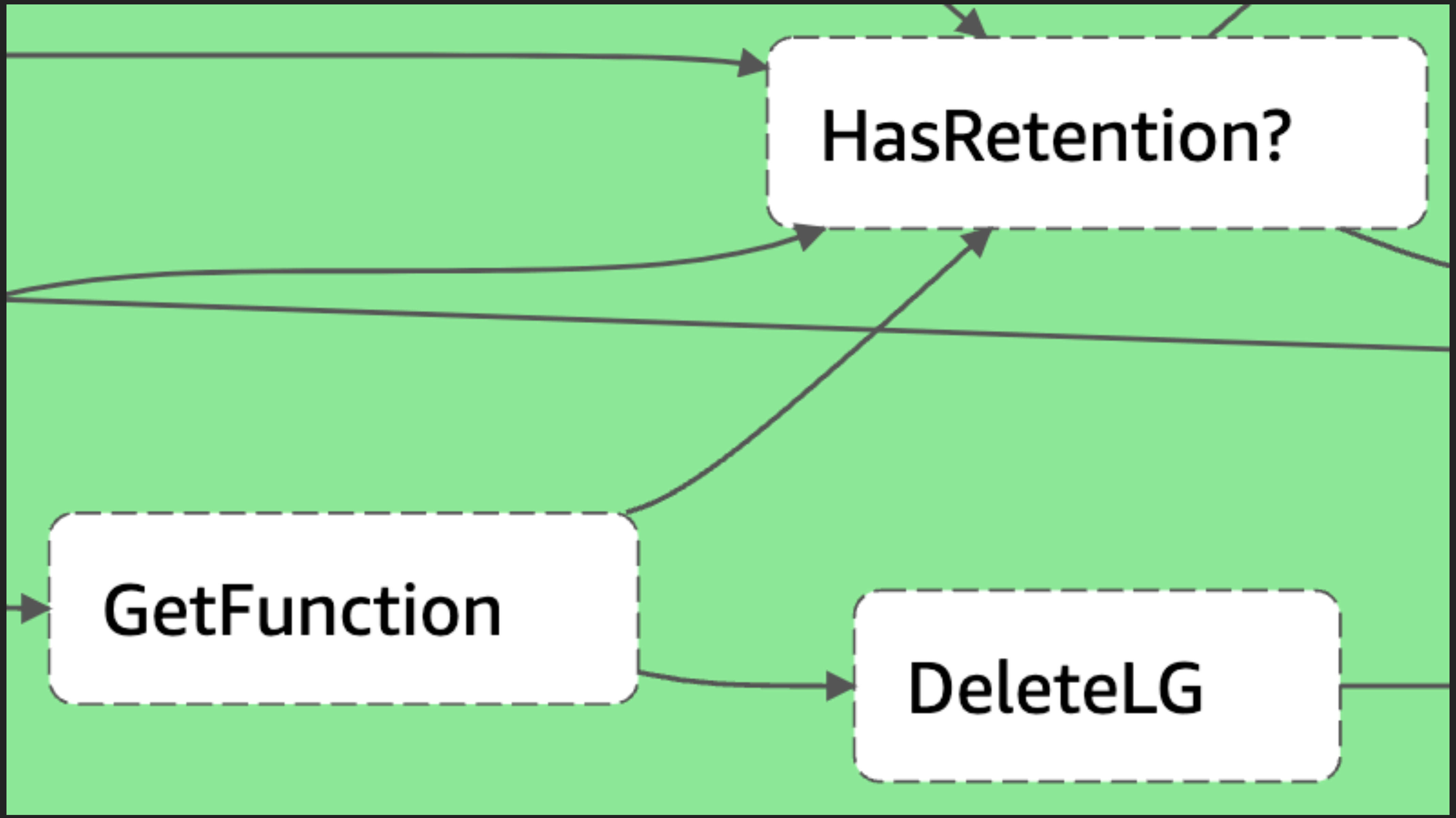
INTRINSICS & SERVICE INTEGRATIONS

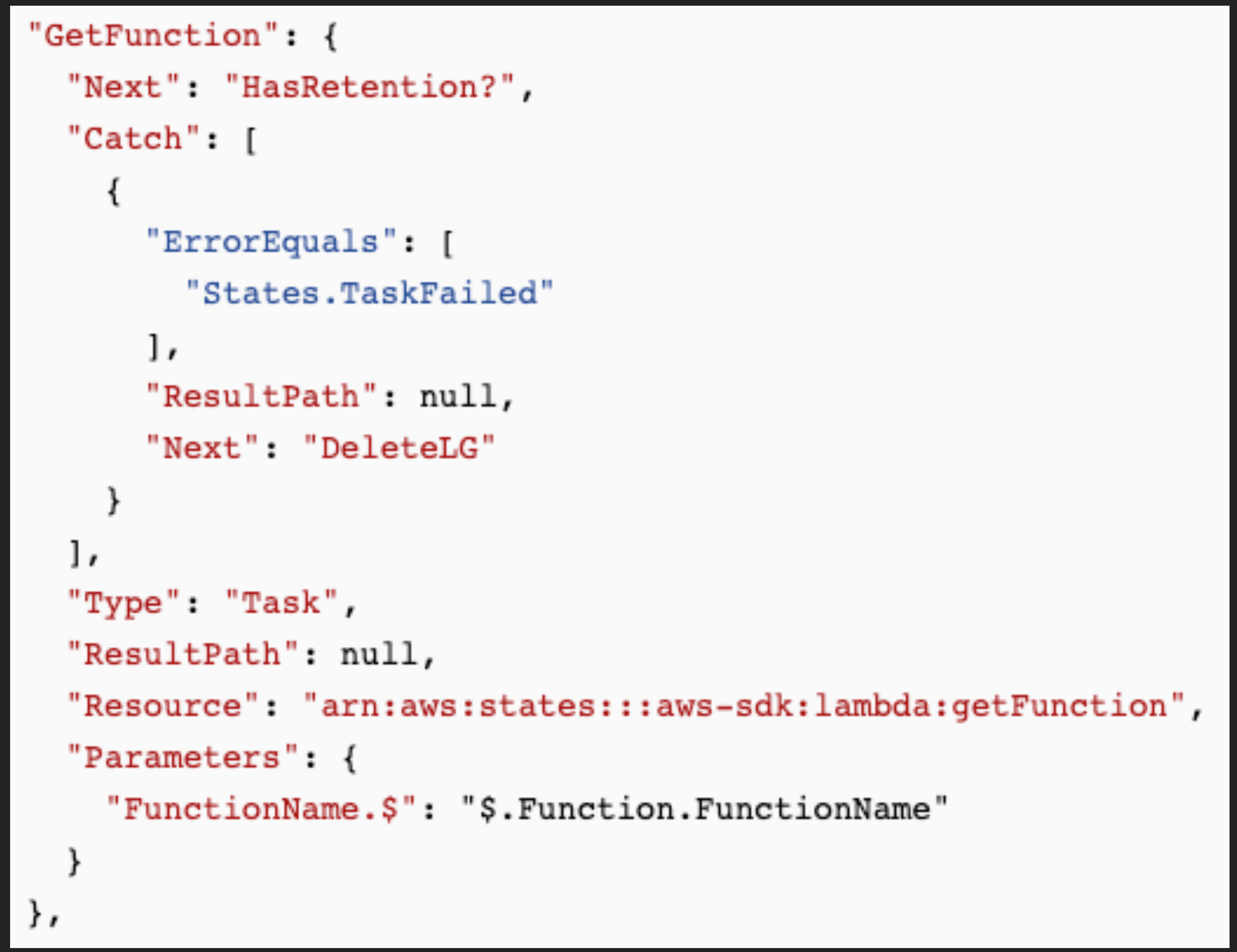




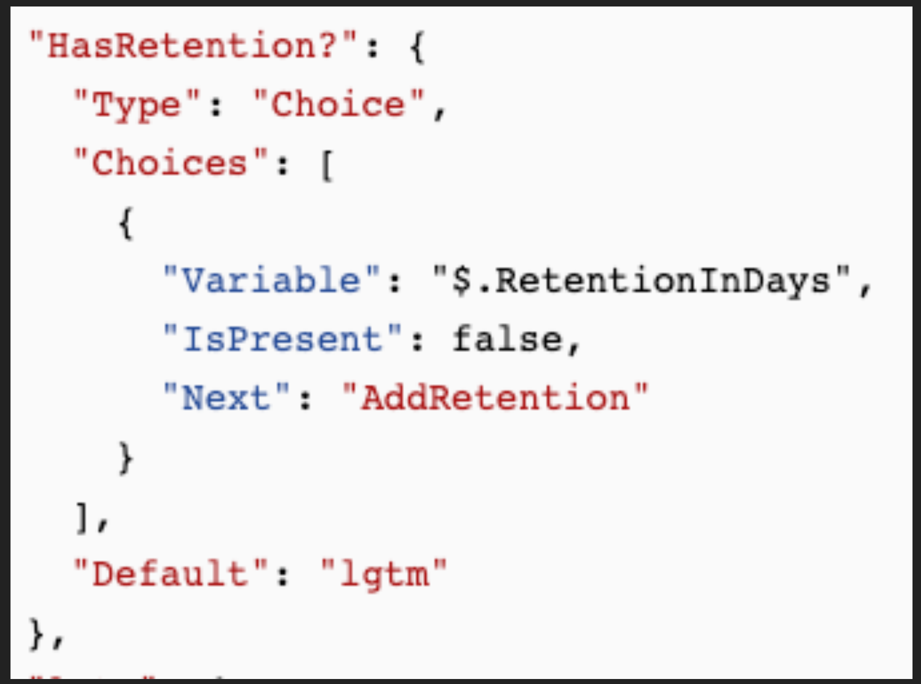
```
"GetLogType": {
  "Type": "Pass",
  "ResultPath": "$.Log",
  "Parameters": {
    "LogType.$": "States.ArrayGetItem($.Function.LGParts, 1)"
  },
  "Next": "IsLambdaLog?"
},
"IsLambdaLog?": {
  "Type": "Choice",
  "Choices": [
    {
      "Variable": "$.Log.LogType",
      "StringEquals": "lambda",
      "Next": "GetFnName"
    }
  ],
  "Default": "HasRetention?"
},
```

INTRINSICS & SERVICE INTEGRATIONS

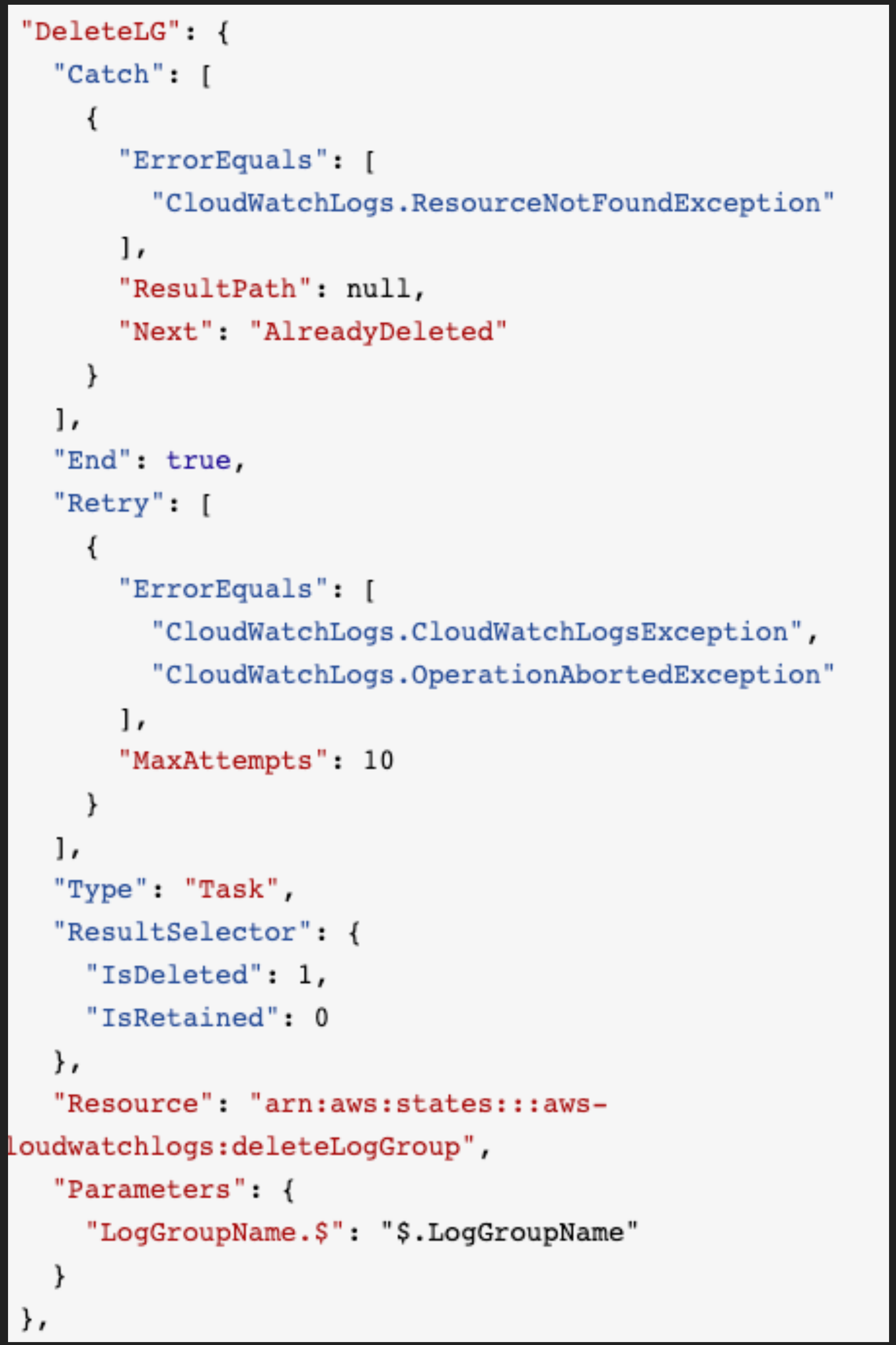




```
"GetFunction": {
  "Next": "HasRetention?",
  "Catch": [
    {
      "ErrorEquals": [
        "States.TaskFailed"
      ],
      "ResultPath": null,
      "Next": "DeleteLG"
    }
  ],
  "Type": "Task",
  "ResultPath": null,
  "Resource": "arn:aws:states:::aws-sdk:lambda:getFunction",
  "Parameters": {
    "FunctionName.$": "$.Function.FunctionName"
  }
},
```



```
"HasRetention?": {
  "Type": "Choice",
  "Choices": [
    {
      "Variable": "$.RetentionInDays",
      "IsPresent": false,
      "Next": "AddRetention"
    }
  ],
  "Default": "lgtm"
},
```



```
"DeleteLG": {
  "Catch": [
    {
      "ErrorEquals": [
        "CloudWatchLogs.ResourceNotFoundException"
      ],
      "ResultPath": null,
      "Next": "AlreadyDeleted"
    }
  ],
  "End": true,
  "Retry": [
    {
      "ErrorEquals": [
        "CloudWatchLogs.CloudWatchLogsException",
        "CloudWatchLogs.OperationAbortedException"
      ],
      "MaxAttempts": 10
    }
  ],
  "Type": "Task",
  "ResultSelector": {
    "IsDeleted": 1,
    "IsRetained": 0
  },
  "Resource": "arn:aws:states:::aws-
loudwatchlogs:deleteLogGroup",
  "Parameters": {
    "LogGroupName.$": "$.LogGroupName"
  }
},
```

TASK TOKENS & WAIT

- Standard workflows have a maximum duration of one year!
- Wait steps can also specify a timestamp.
- Wait for Task Token is a great way to fire off an async process.

TASK TOKENS & WAIT

S3

- /my-customer-files
  - /customer1 // millions of objects
  - /customer2 // millions of objects
  - /customer3 // billions of objects
  - …tens of thousands of customers

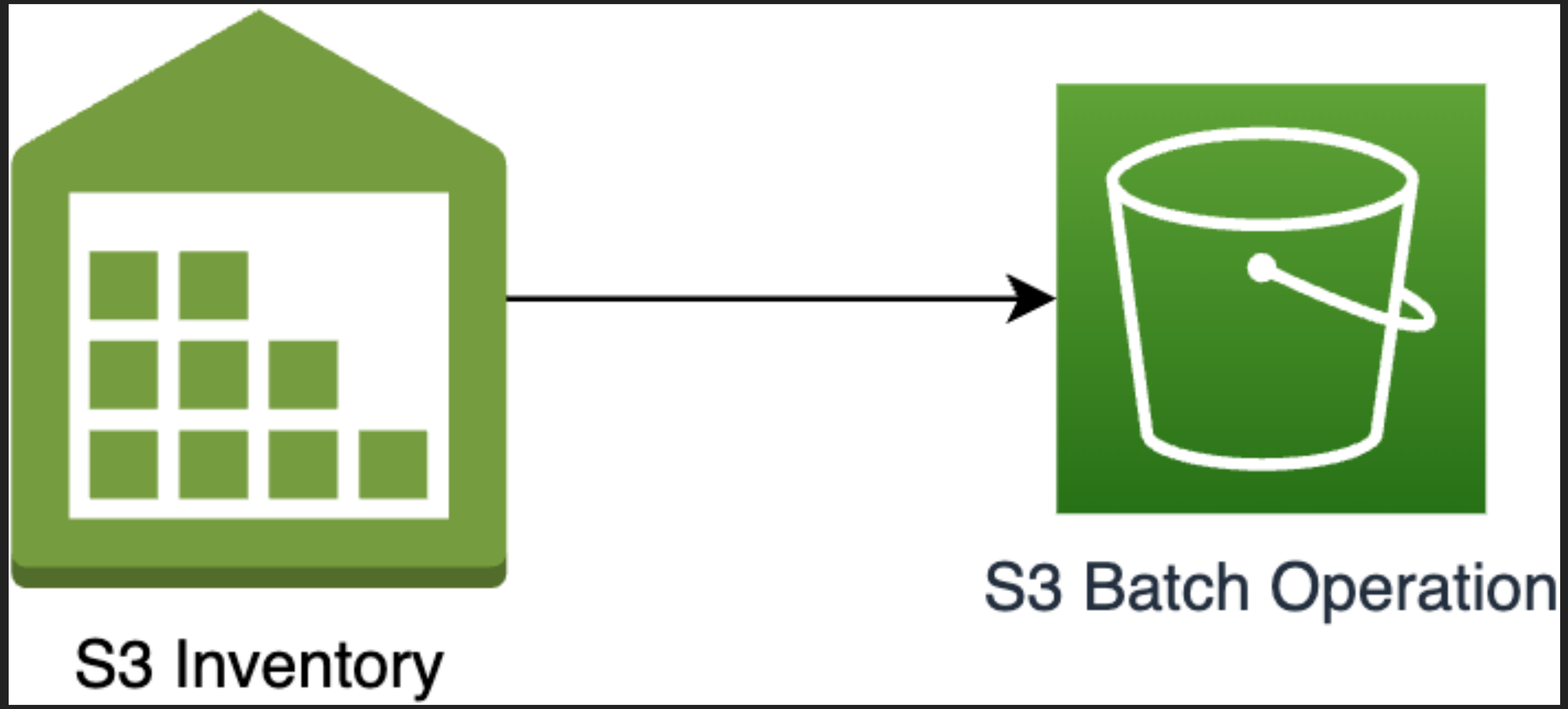
TASK TOKENS & WAIT

S3API can't handle *billions* of objects. (1 billion objects = 1 million API calls)

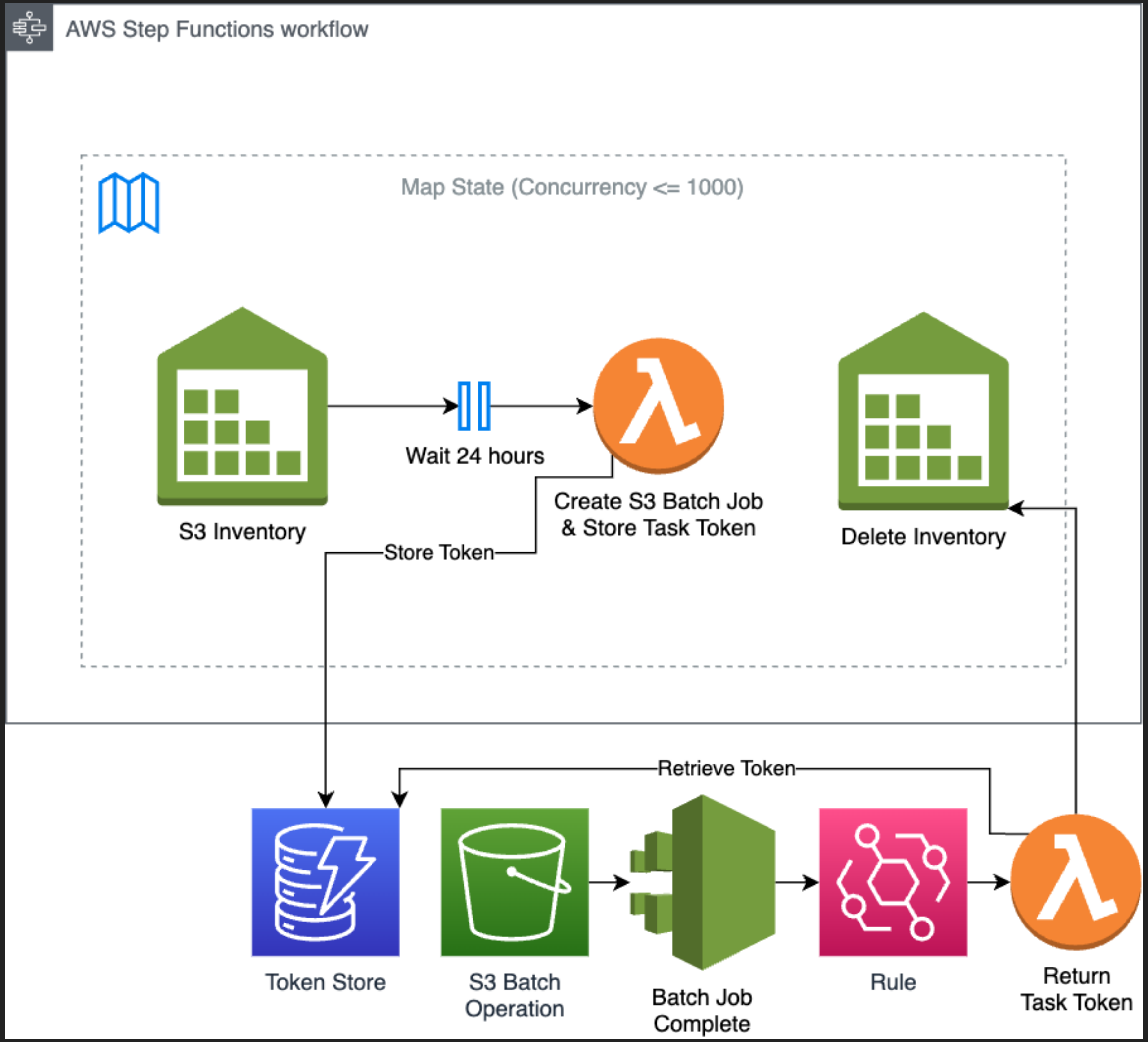
S3 Batch Operations requires an inventory.

Inventories have to be scheduled for daily or weekly operation.

A bucket has a hard limit of 1000 inventories.



TASK TOKENS & WAIT



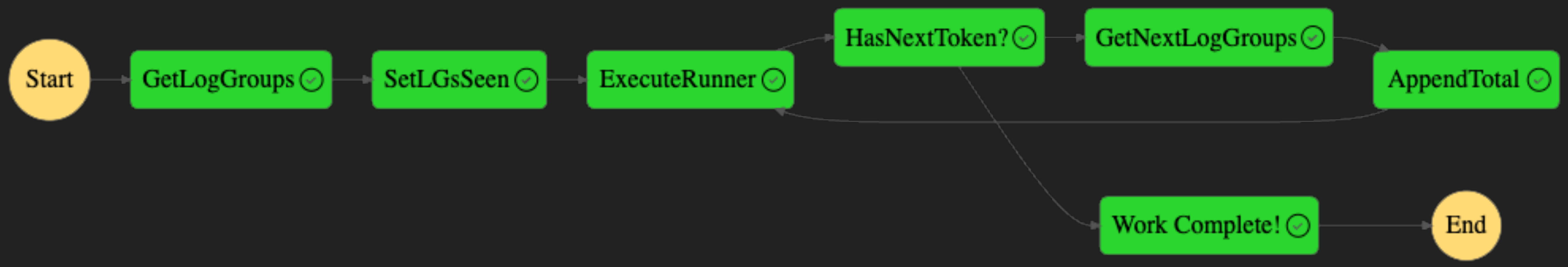
MAP AND PARALLEL STATES

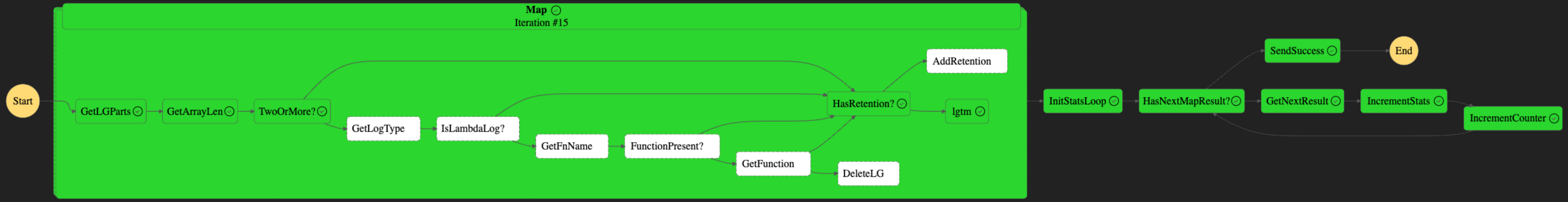
- Map States - not just fan-out but also fan-in!
- Parallel States
- Supports nesting (watch execution limit)
- Distributed Map is a game-changer - see the March meeting up with Adam and Dustin:
  https://sga.com/events
  https://www.meetup.com/serverless-boston/events/292096634/
  https://www.meetup.com/serverless-nyc/events/292096525/

THE LEARNING CURVE

- Amazon States Language, the DSL for Step Functions https://states-language.net/
- Data Flow Simulator https://aws.amazon.com/about-aws/whats-new/2021/04/aws-step-functions-adds-new-data-flow-simulator-for-modelling-input-and-output-processing/
- Workflow Studio https://docs.aws.amazon.com/step-functions/latest/dg/workflow-studio.html
- Community Tools https://dev.to/aws-builders/supercharge-your-stepfunctions-productivity-with-local-file-system-sync-in-workflow-studio-4ab9 https://matthewbonig.com/2022/02/19/step-functions-and-the-cdk/

SUMMARY & DISCUSSION





https://mattmorgan.cloud