API001

I didn’t know AWS Step Functions did that!

Benjamin Smith

Principal Developer Advocate, Serverless
AWS

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

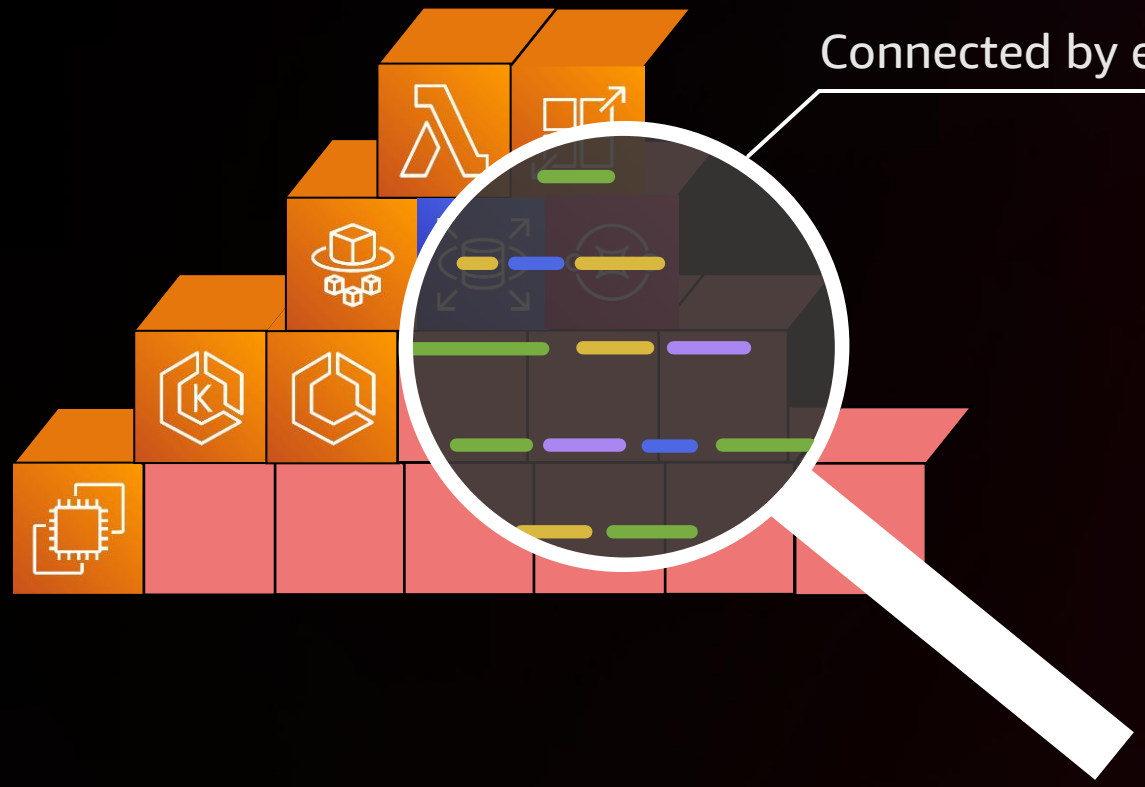


Nice to meet you !

- Ben Smith
  - Email: benjasl@amazon.com
  - Twitter: @benjamin_l_s
- Most interested in
  - Event-driven architectures with Step Functions
  - Serverless and PHP
- Random fact

  I once taught David Hasselhoff to ski

Managed



Manual

The

Serverless

Event-driven pyramid in 2023

aws

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

The serverless challenge

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

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

AWS Step Functions is different

AWS Lambda

Amazon API Gateway

Amazon DynamoDB

Amazon EC2

Amazon EventBridge

Amazon S3

Amazon ECS

Amazon SNS

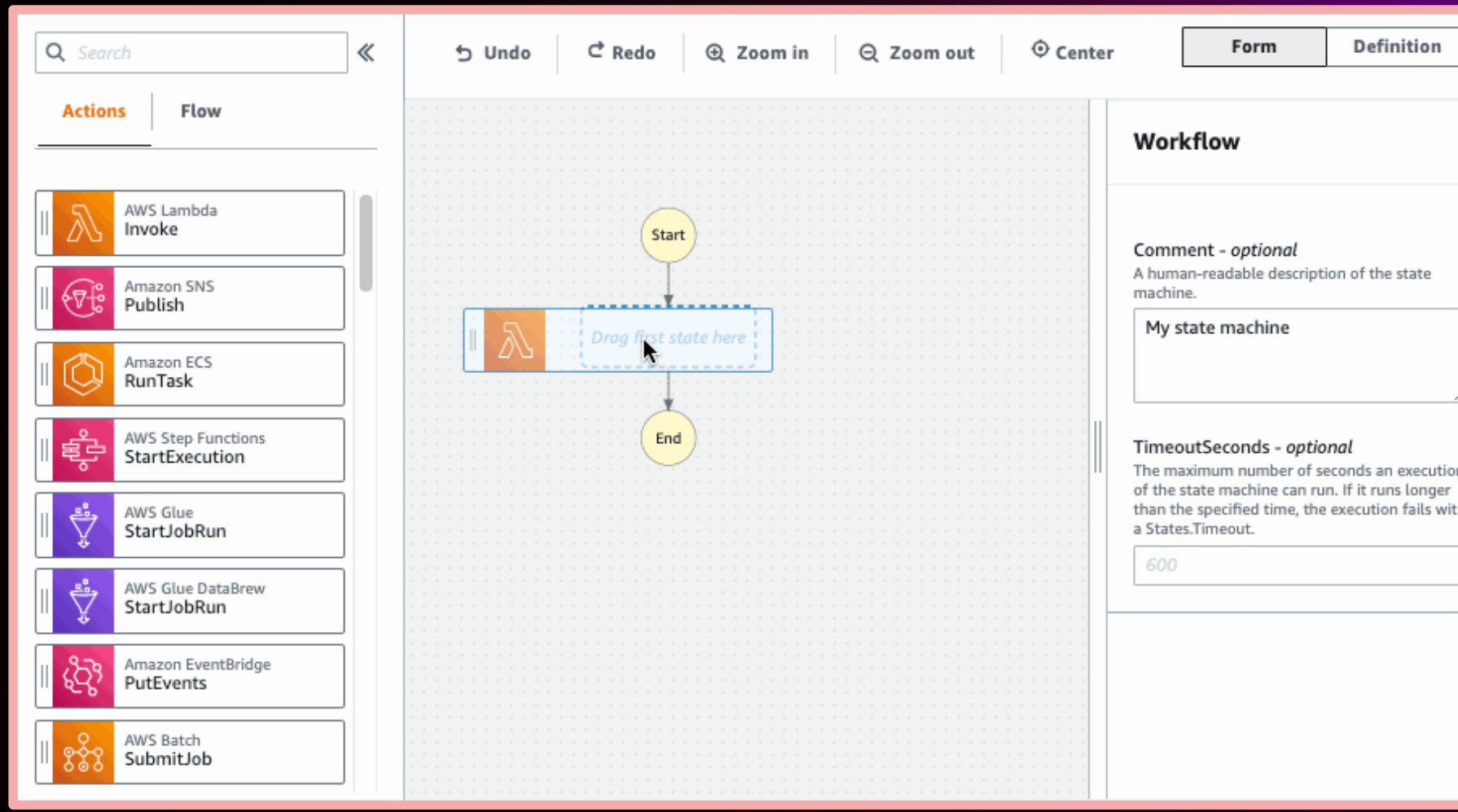
Amazon Kinesis

Amazon EKS

Amazon SQS

AWS IoT Core

AWS Fargate



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

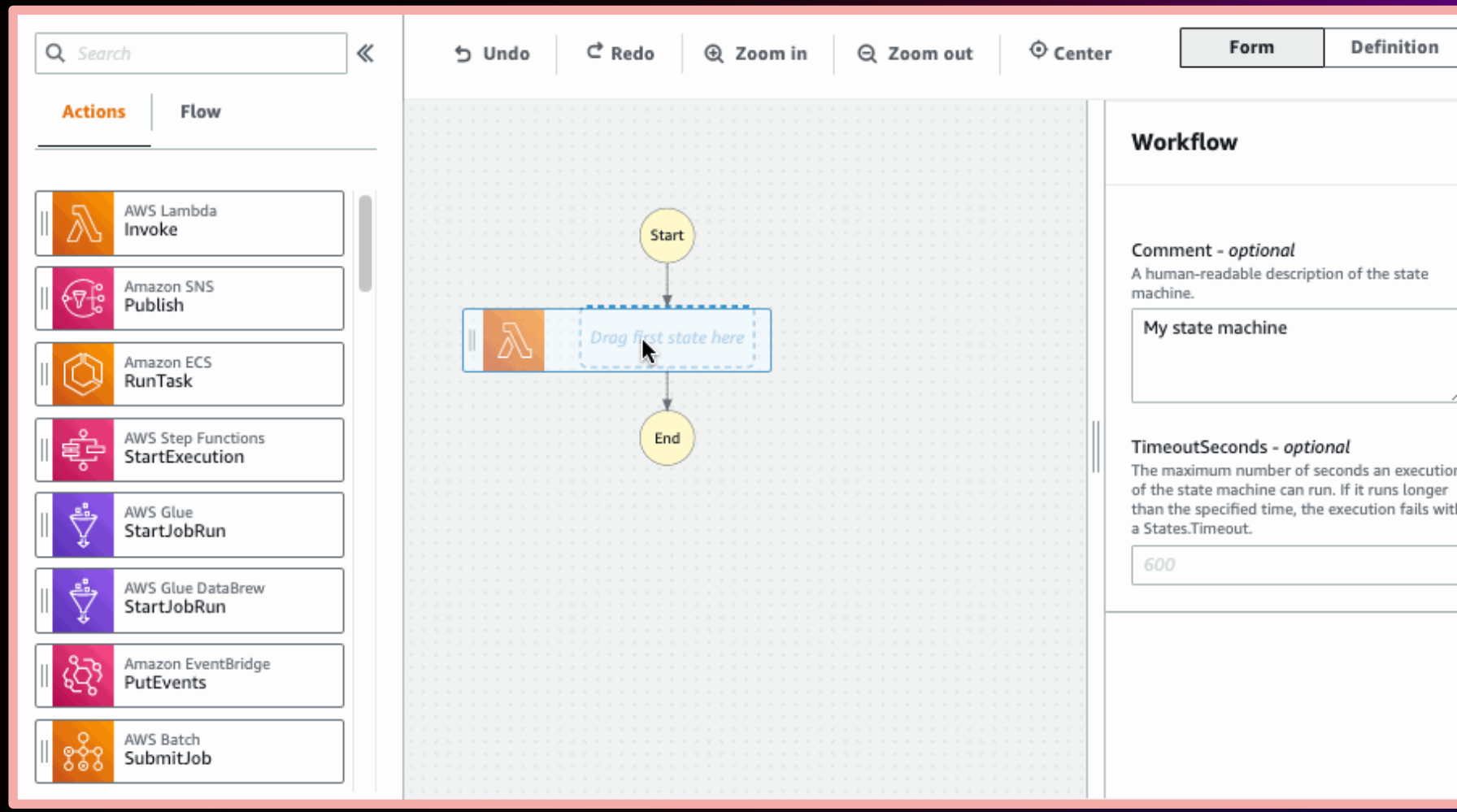
AWS Step Functions is different

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



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

“Step Functions first, Step Functions always”

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

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 }
  }
}
```

A Lambda function that queries Amazon DynamoDB has multiple lines of code



AWS Lambda

Amazon DynamoDB

aws

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

From code to workflow

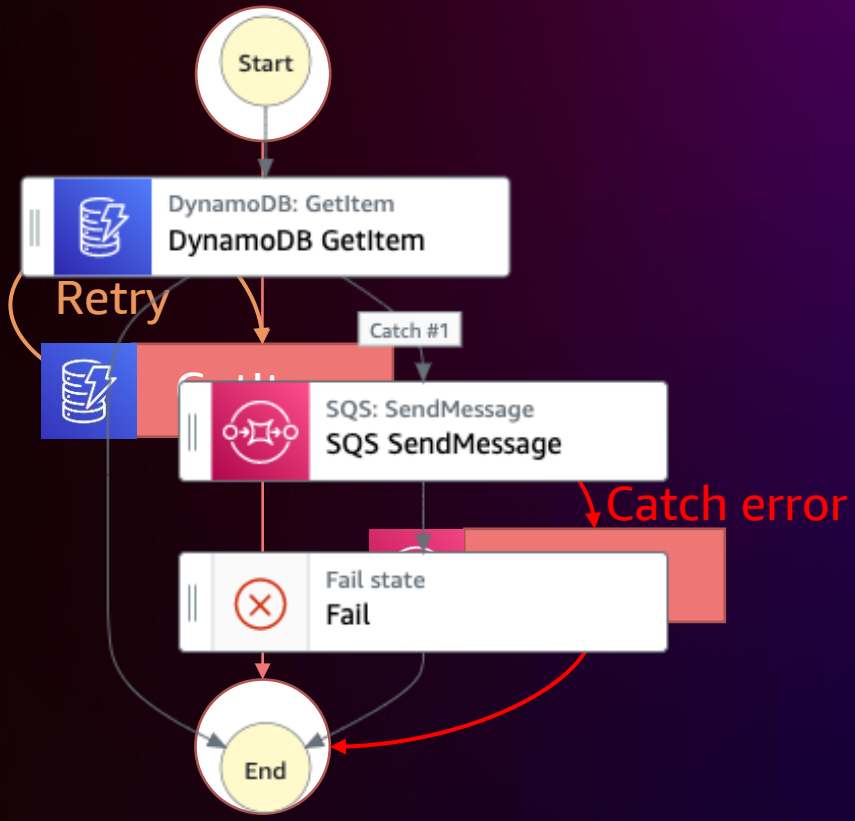
app.js

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


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

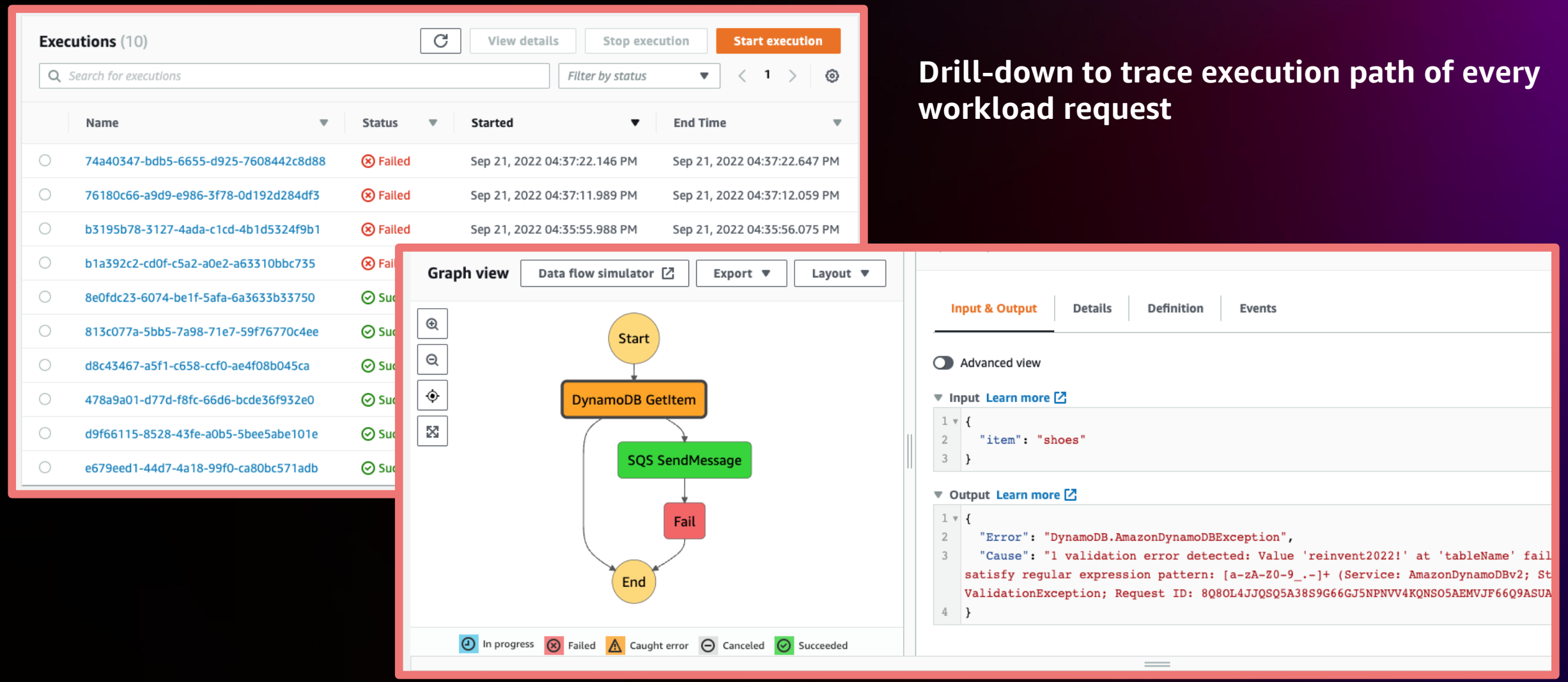

async function queryItems(){
  try {
    const data = await docClient.getItem(params).promise()
    return data
  } catch (err) {
    return err
  }
}
exports.handler = async (event, context) => {
  try {
    const data = await queryItems()
    return { body: JSON.stringify(data) }
  } catch (err) {
    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



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

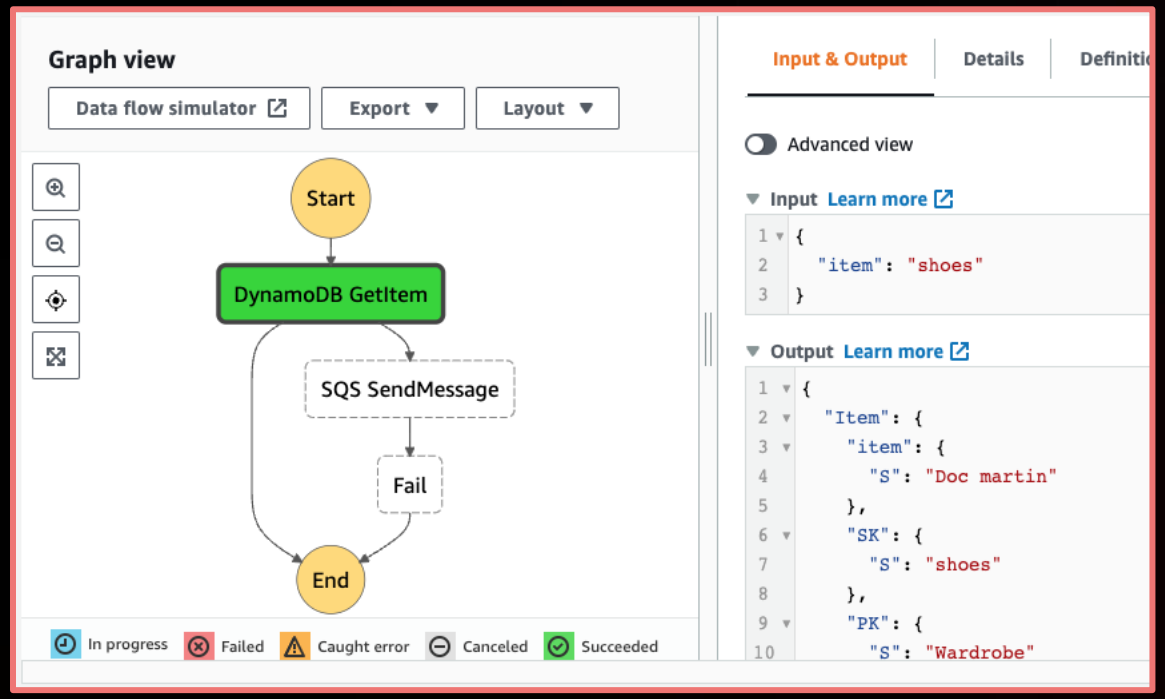
From code to workflow



Drill-down to trace execution path of every workload request

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

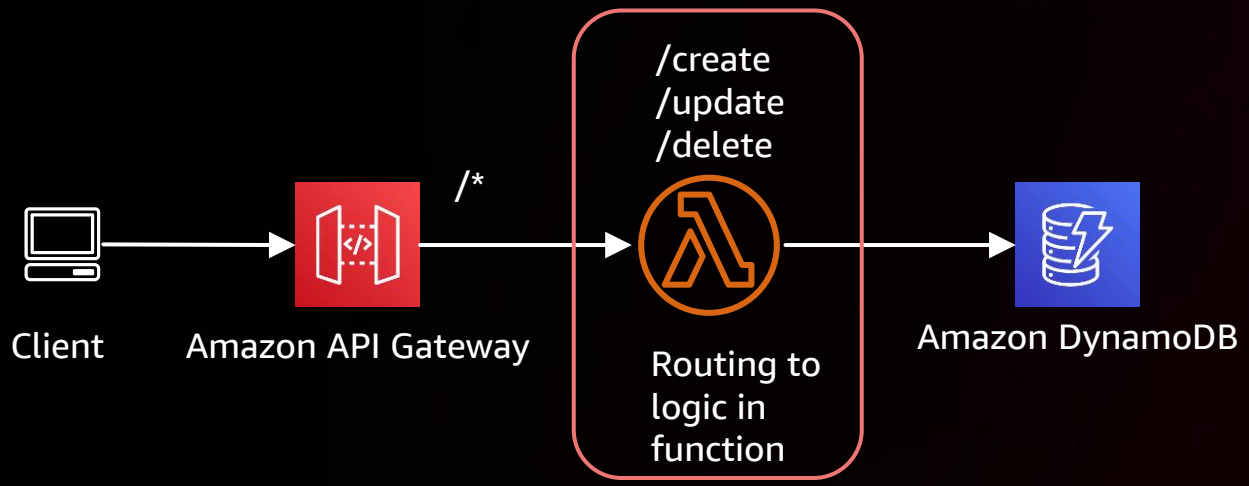
From code to workflow



Examine the input and output of each task for each request

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

Breaking apart a “Lambda-lith”

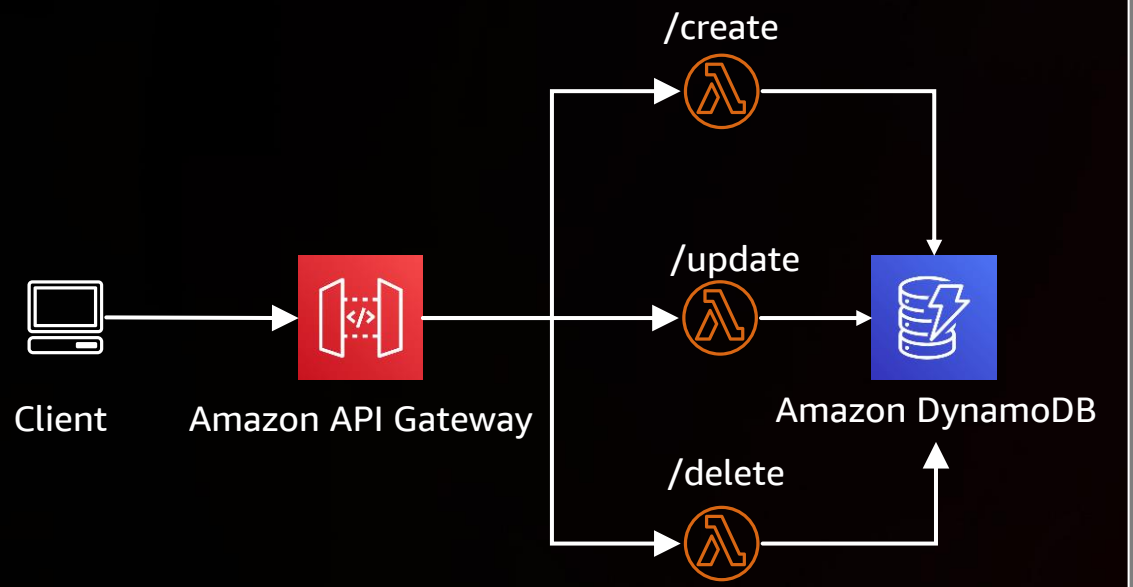


Amazon API Gateway routes all requests to a single Lambda function that runs the appropriate code based on its route configuration

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

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

Micro Lambda



App-delete.js

App-update.js

app-create.js

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

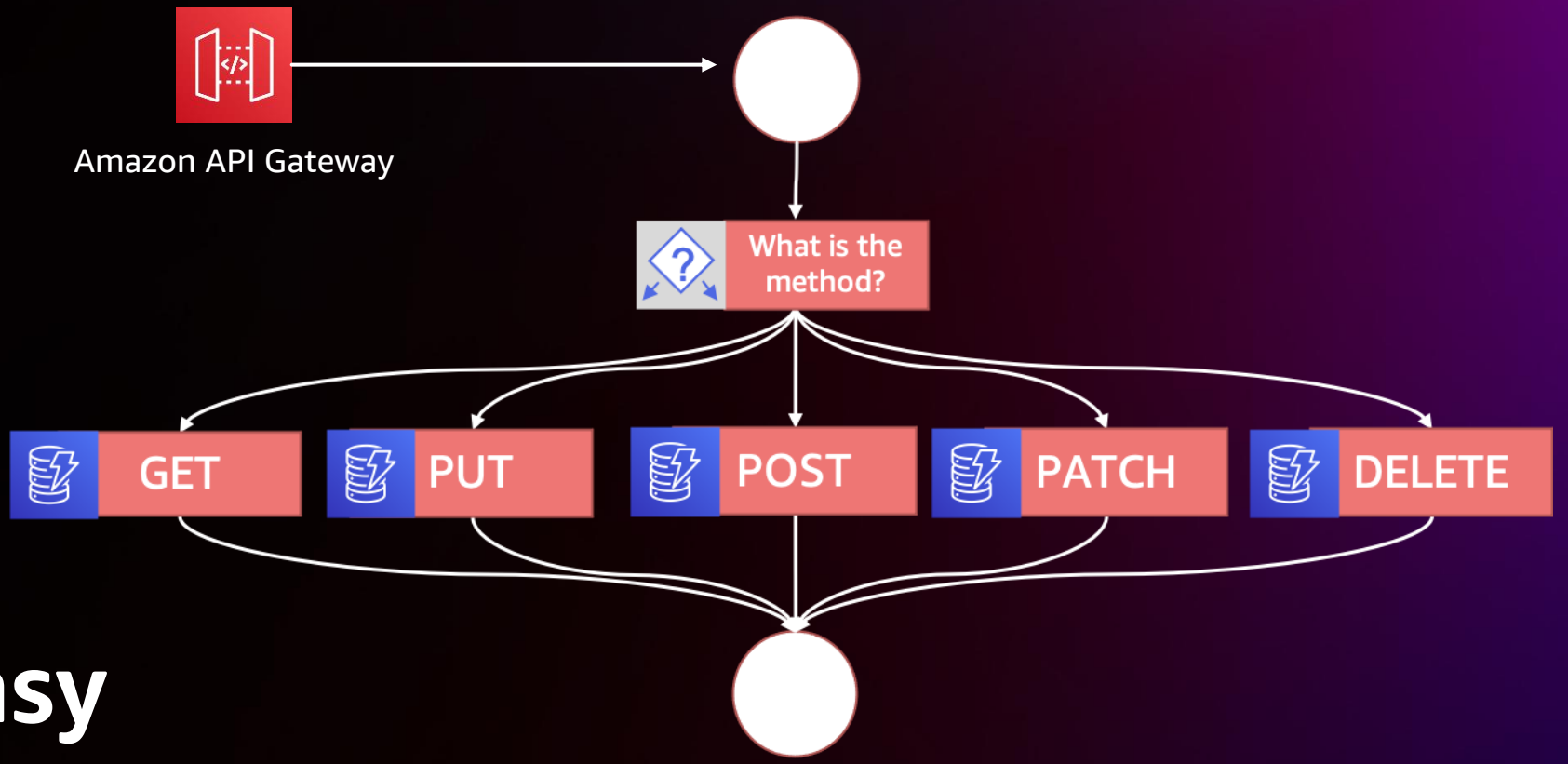
var params = {
  TableName: 'your-table-name',
  IndexName: 'some-index',
  KeyConditionExpression: '#name = :value',
  ExpressionAttributeValues: { ':value': 'shoes' },
  ExpressionAttributeNames: { '#name': 'name' }
}

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 }
  }
}
```

aws

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



The “REST” easy

Combine with API Gateway and synchronous Express Workflows to create low latency, scalable API backend

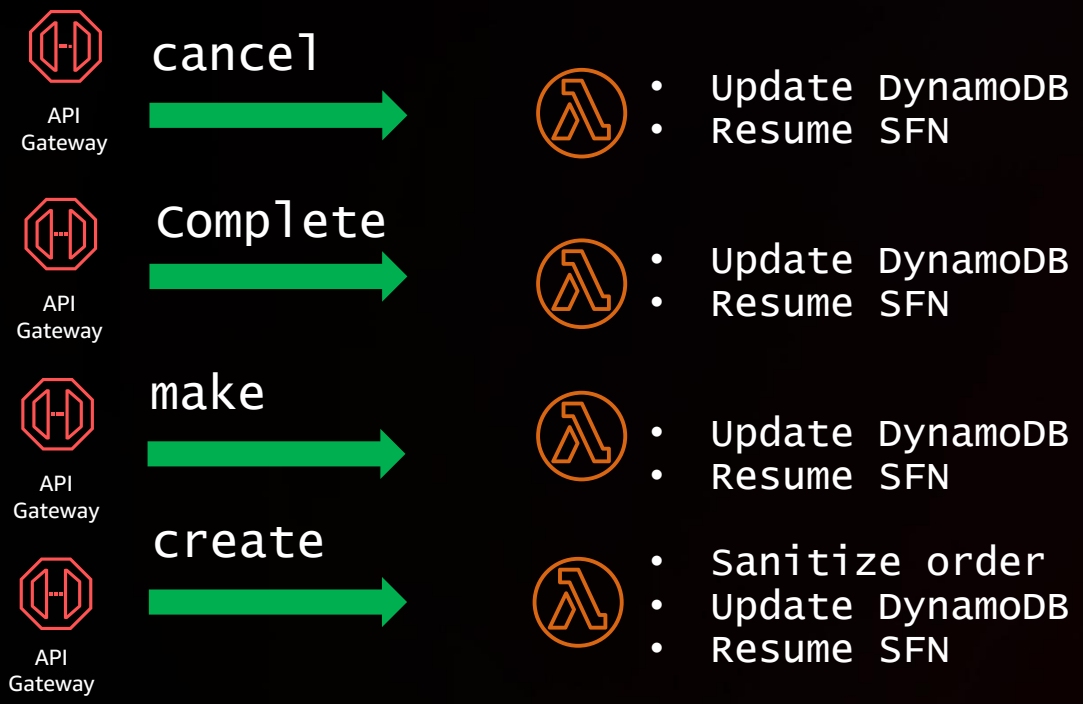
Simplicity

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

Version 1

Order Manager service as a Lambda function

Each operation invokes a Lambda function



Application grew more complex over time, performing multiple tasks to handle increasingly complex business logic, leading to

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

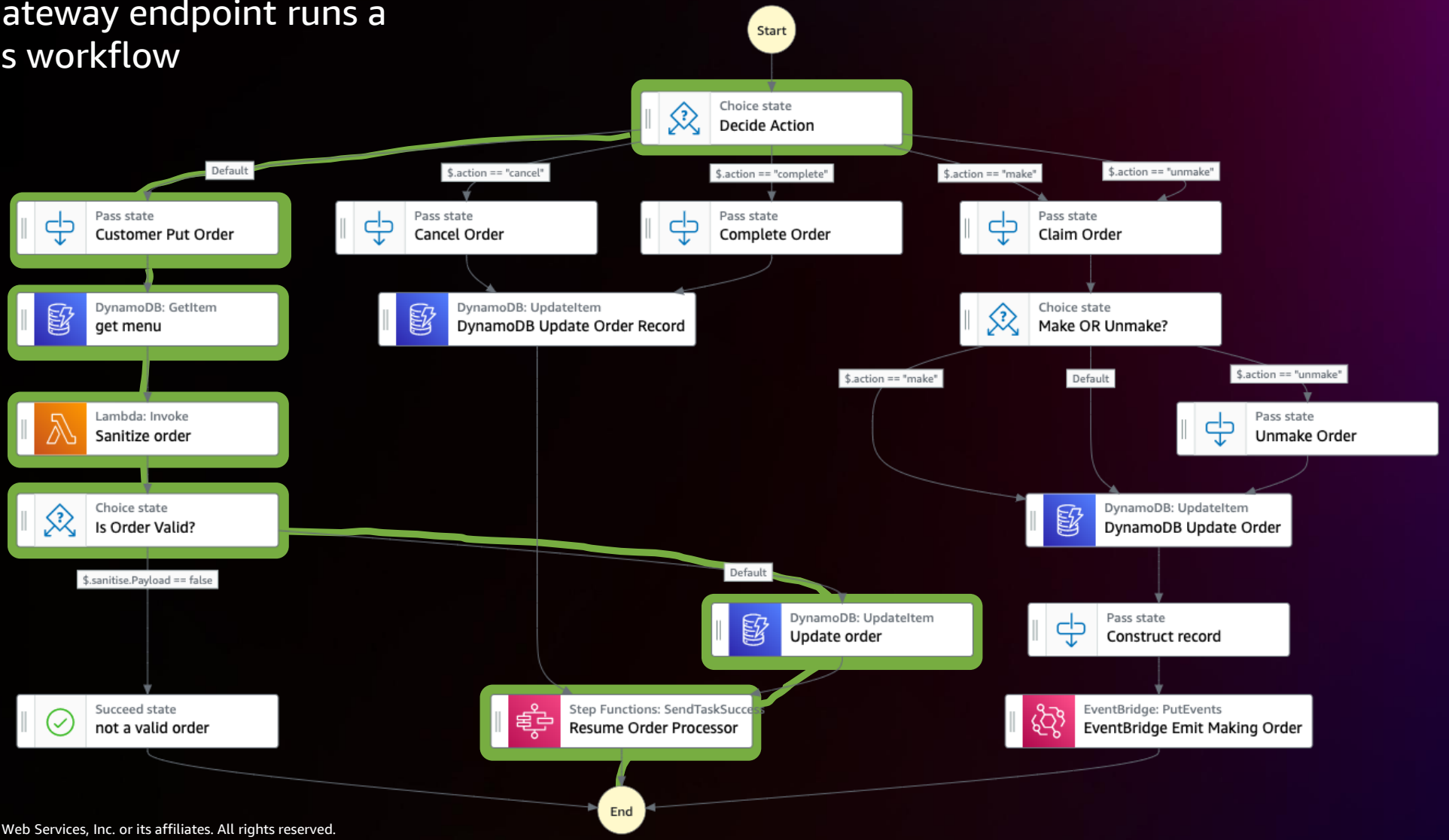
aws

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

Version 2

Order Manager service as a workflow

A single API Gateway endpoint runs a Step Functions workflow



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

Step Functions at scale

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

Parallelism

Doing lots of things at once

Same input different steps

VS

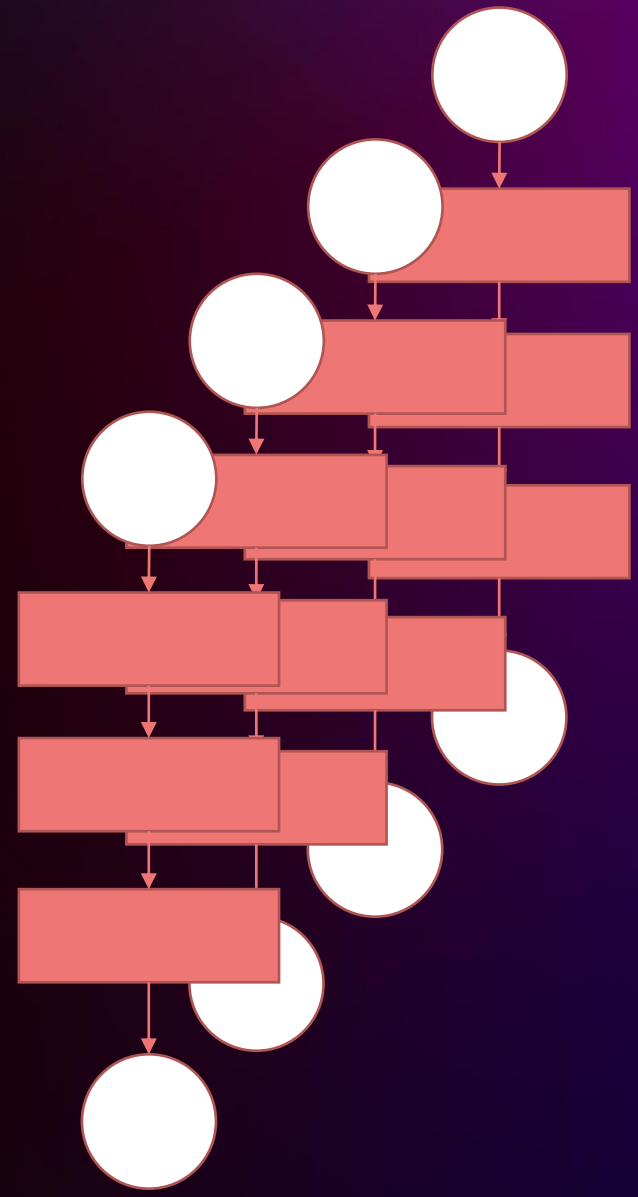
Concurrency

Dealing with lots of things at once

Same steps different input

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

Parallel invocations, lots of independent workflows doing their own thing



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

Parallel invocations

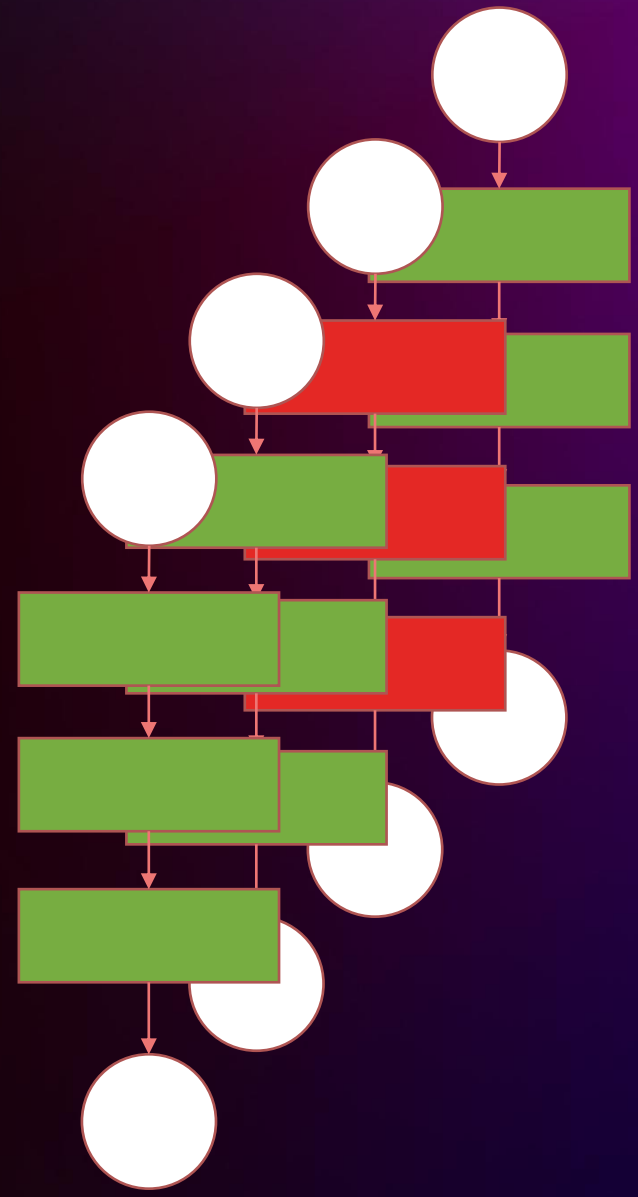
Same ASL definition executing at the same time, with different data

Each process is independent

Failure in one process doesn't impact another

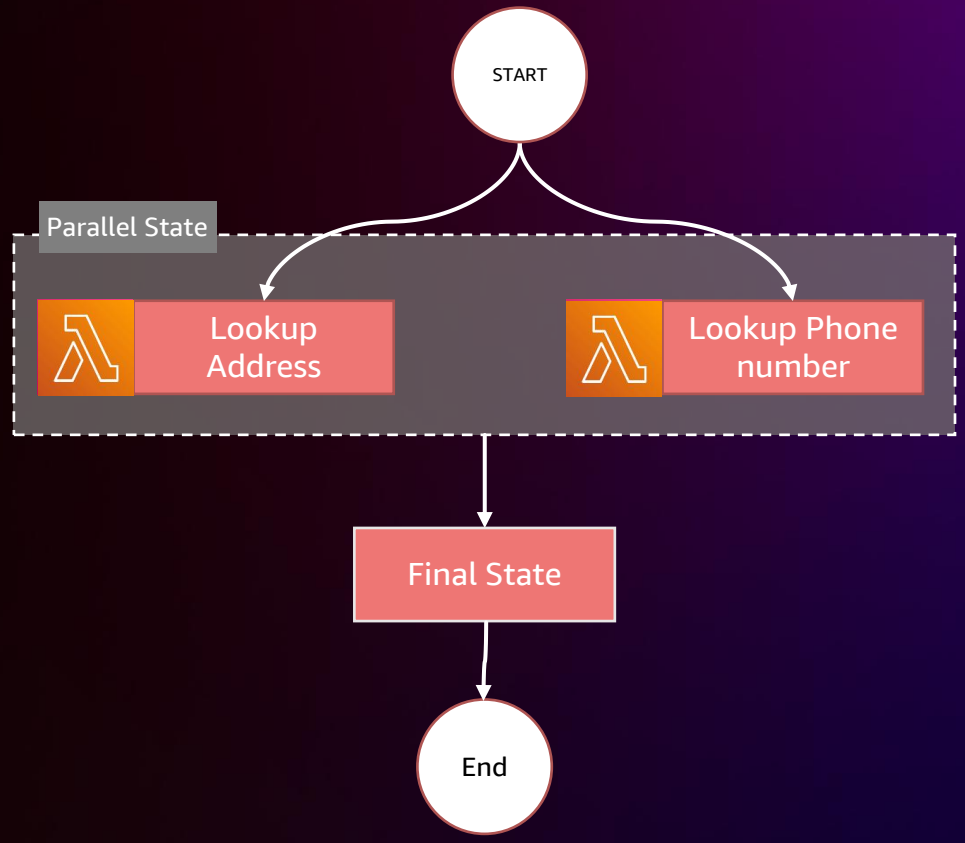
E.g., remediation workflow for noncompliant resource

Limitation: account quote (mostly soft limits)



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

Parallel state:
Execute a fixed number of branches in parallel

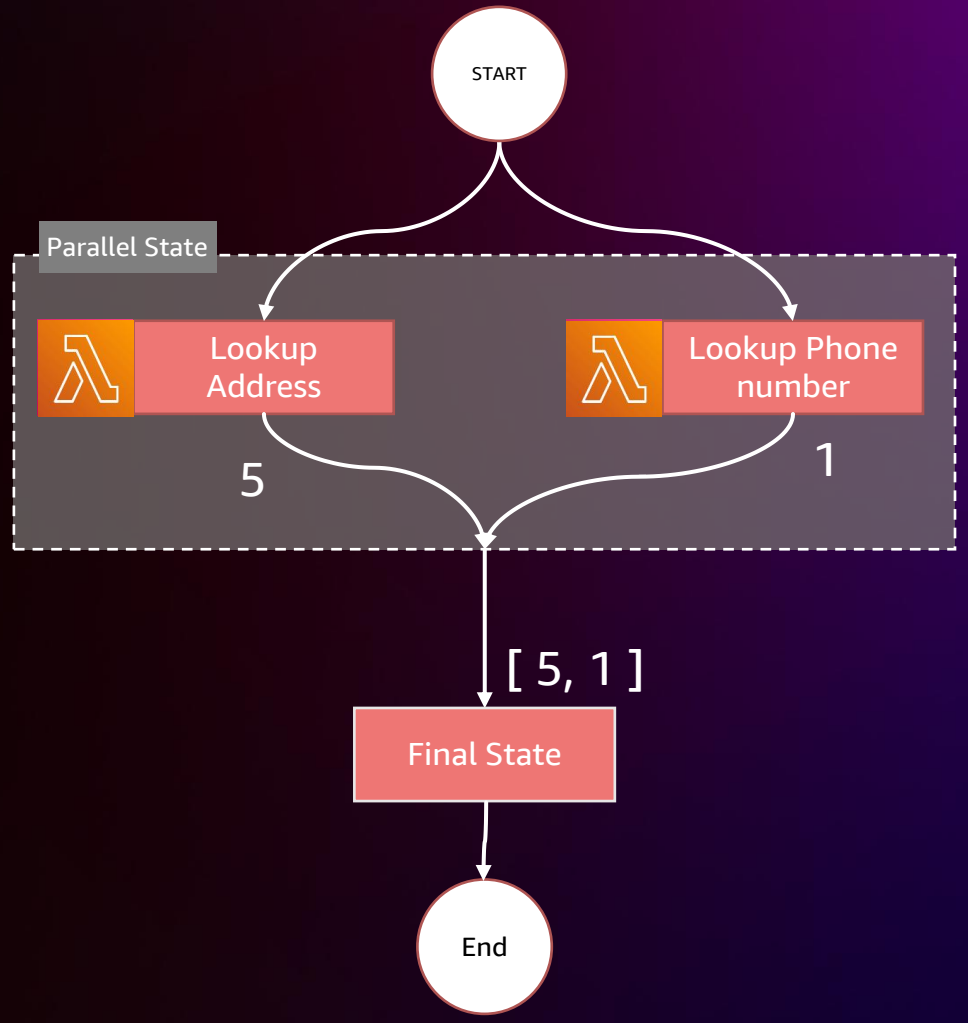


© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Parallel state

Executes multiple branches of steps using the same input

Map state output is an array with one element for each branch result

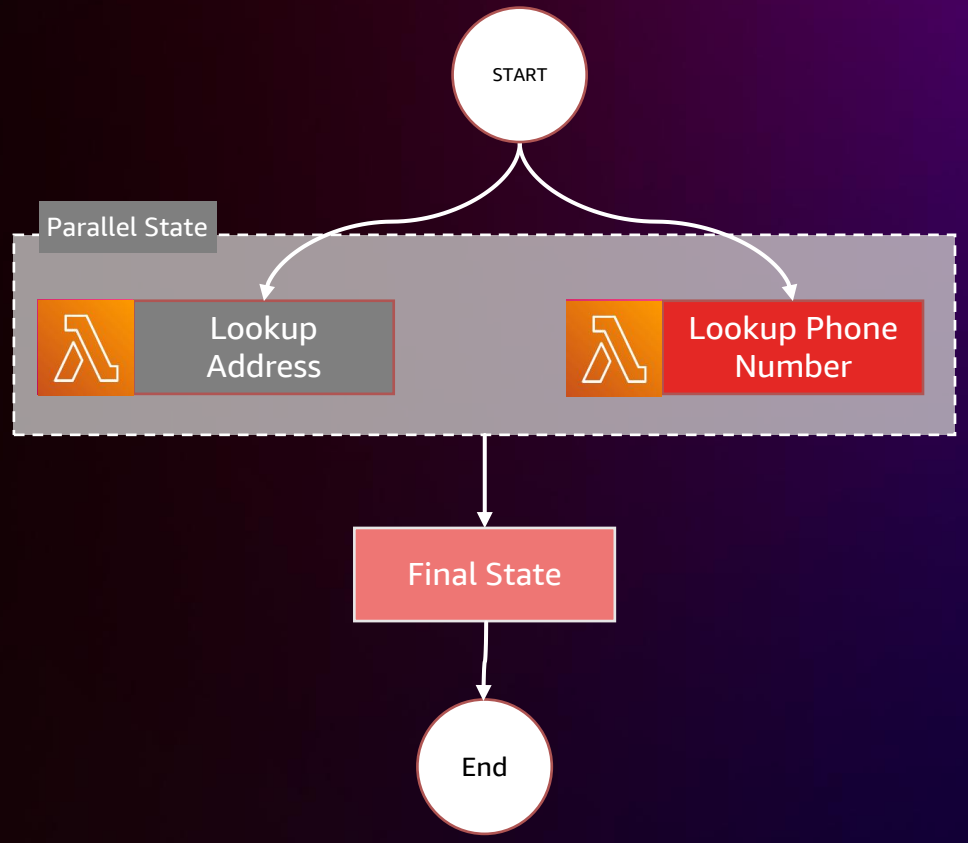


© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Parallel state

Wait until all branches terminate (reach a terminal state) before processing the next state

If any branch fails because of an unhandled error or by transitioning to a fail state, the entire parallel state is considered to have failed and all of its branches are stopped

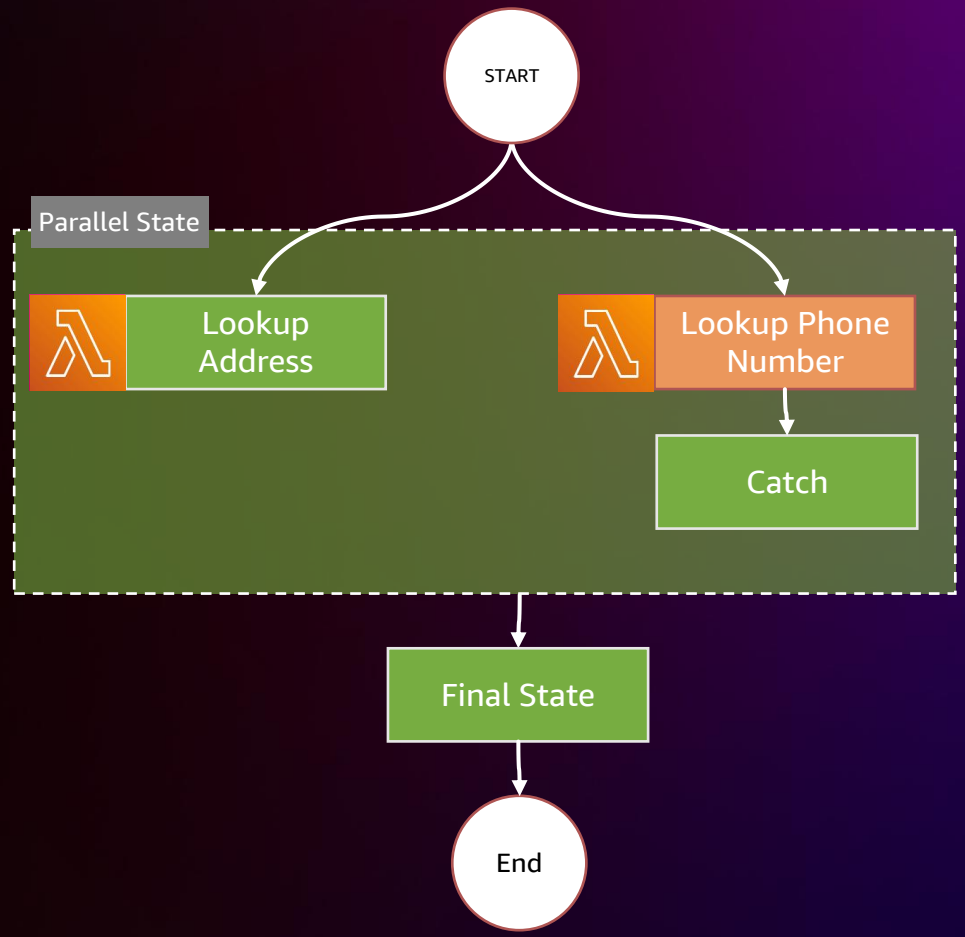


aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Parallel state

Catch errors to allow other branches to complete

All execution results are aggregated into the final state

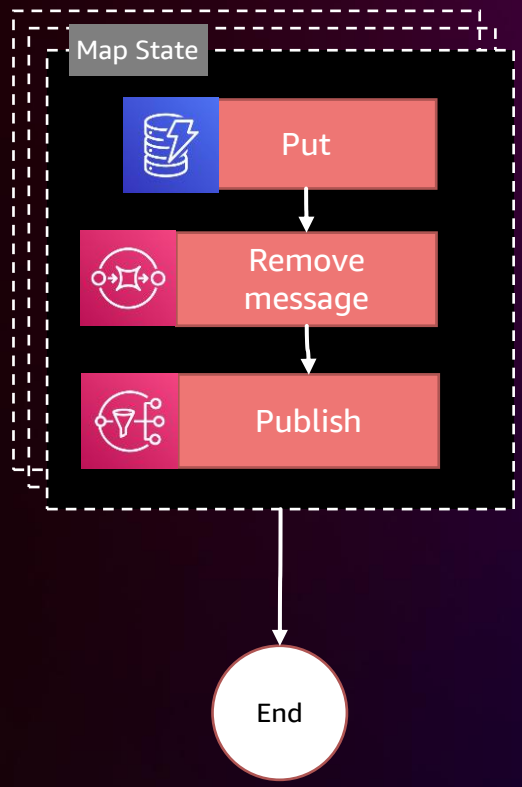


© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Dynamic parallelism

A sub-workflow to accomplish work faster by using more resources at the same time

Limited to 32,768 characters



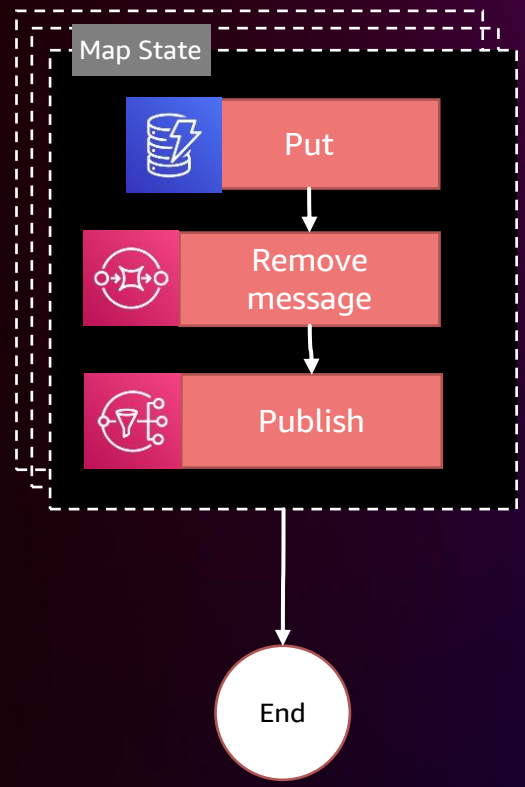
aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Dynamic parallelism

Execute same steps for multiple entries of an input array

MaxConcurrency controls number of iterations that will run in parallel

Set to 1 for single-threaded integrations
Set to 0 for maximum parallelism

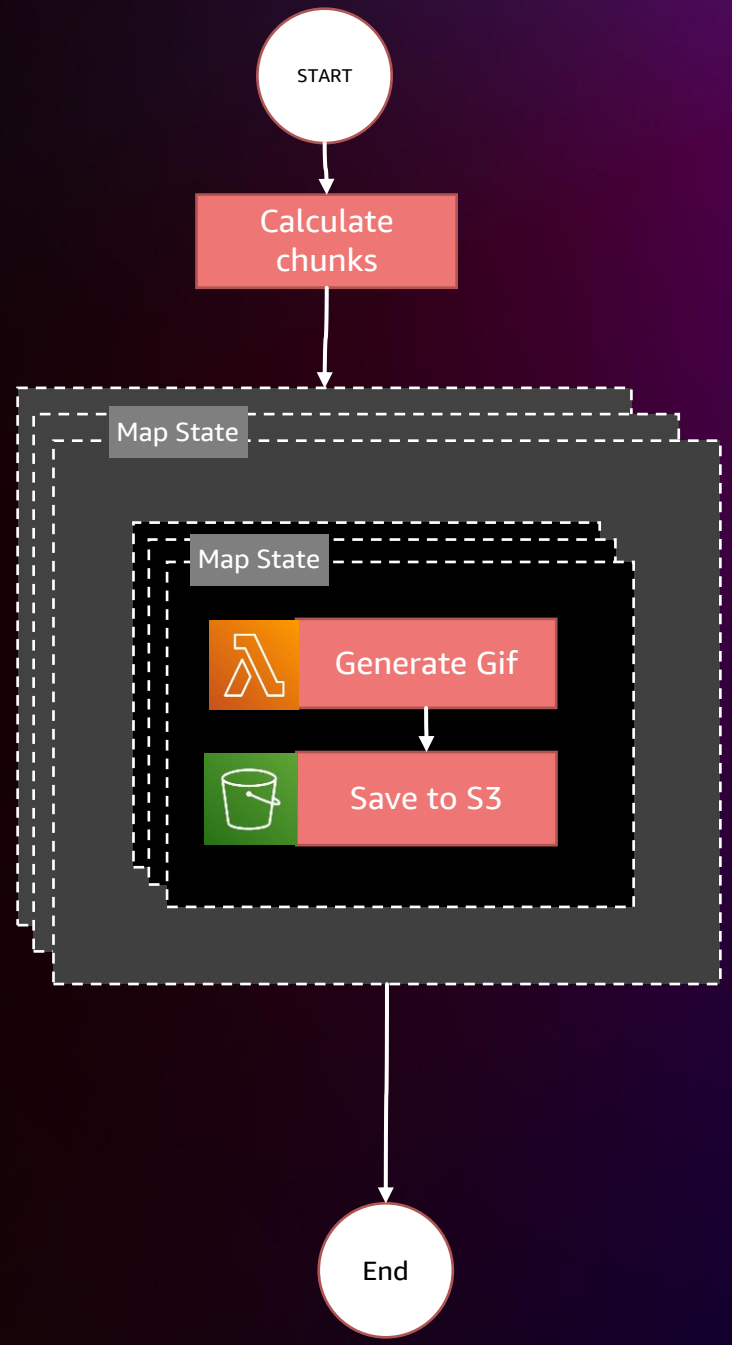


aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

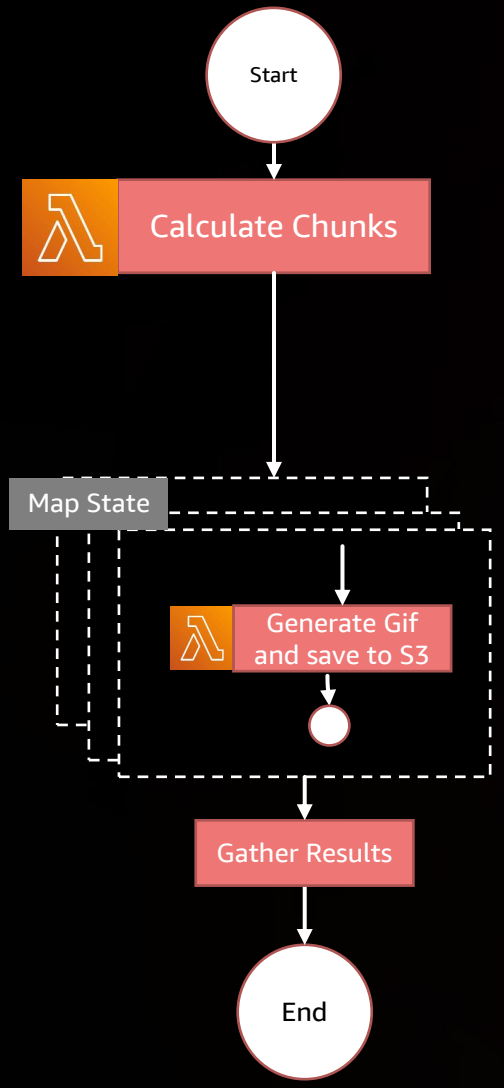
Map state limitations

When your input array has more than 40 items, some iterations may not begin until previous iterations have completed

To achieve a greater concurrency, consider using nested state machines that cascade map states. For example, to achieve a concurrency of 1,024, you could build a state machine that contains a map state that iterates 32 times, then nest that state machine into the map state of a higher level state machine that iterates 32 times

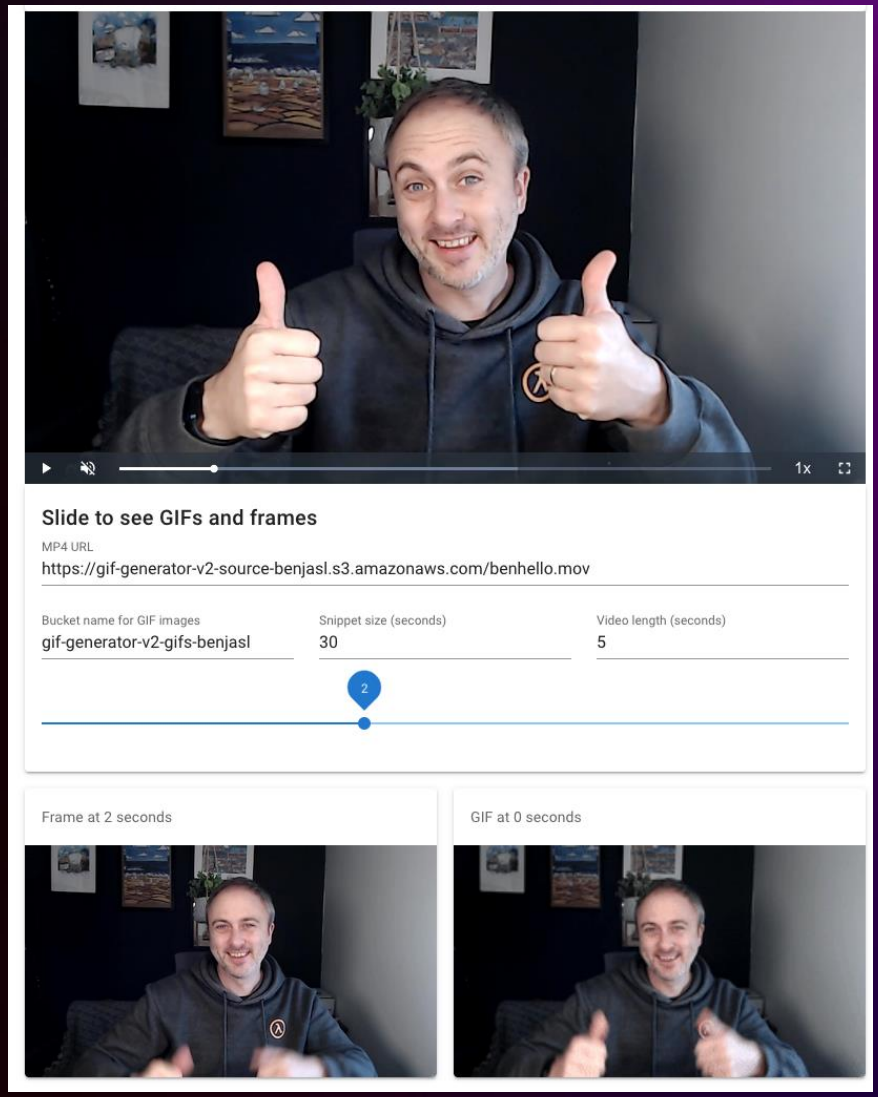


aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.



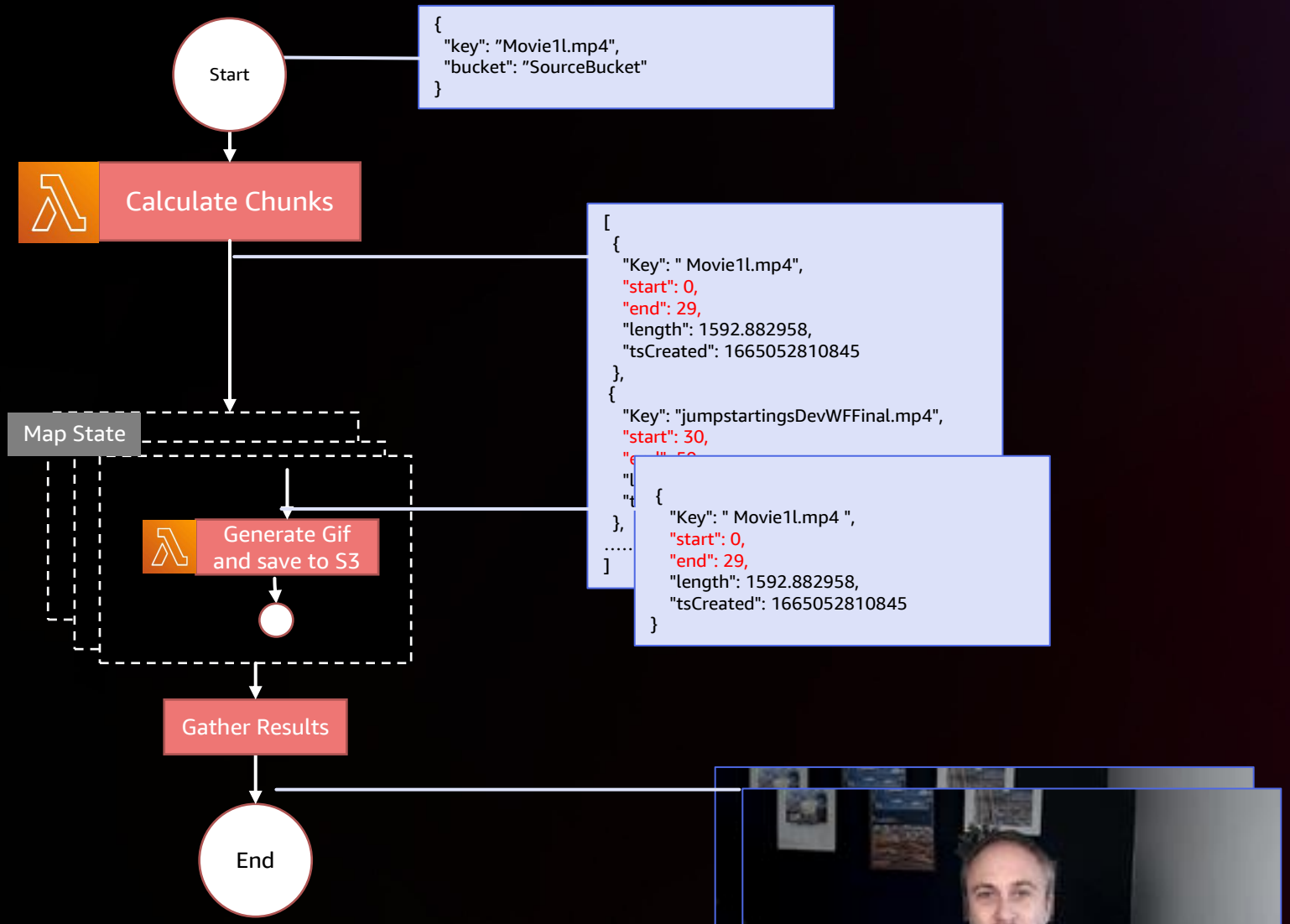
A serverless Gif generator

Convert an .mp4 from S3 into multiple Gifs for timeline scrubbing



Scalability

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





Scalability

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

Edit generateGif Info

Cancel | Apply and exit

Search

Actions | Flow

MOST POPULAR

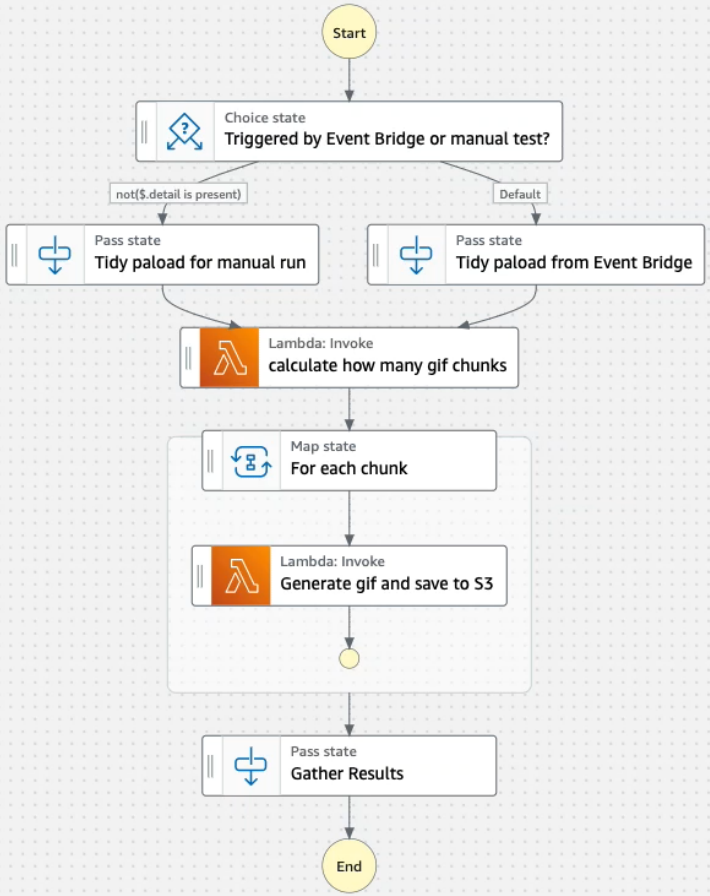
- AWS Lambda Invoke
- Amazon SNS Publish
- Amazon ECS RunTask
- AWS Step Functions StartExecution
- AWS Glue StartJobRun

COMPUTE

- Amazon Data Lifecycle Manager
- Amazon EBS
- Amazon EC2
- AWS EC2 Instance Connect
- Elastic Inference
- Elastic Load Balancing V1
- Elastic Load Balancing V2
- Recycle Bin for EBS
- Amazon Lightsail
- AWS Lambda
- AWS Batch
- Elastic Beanstalk
- AWS Serverless Application Re...

Undo | Redo | Zoom in | Zoom out | Center | Duplicate | Delete

Import/Export | Form | Definition



Workflow

Start at
Choose which state is the starting point of the workflow

Triggered by Event Bridge or manual test?

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

A description of 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.

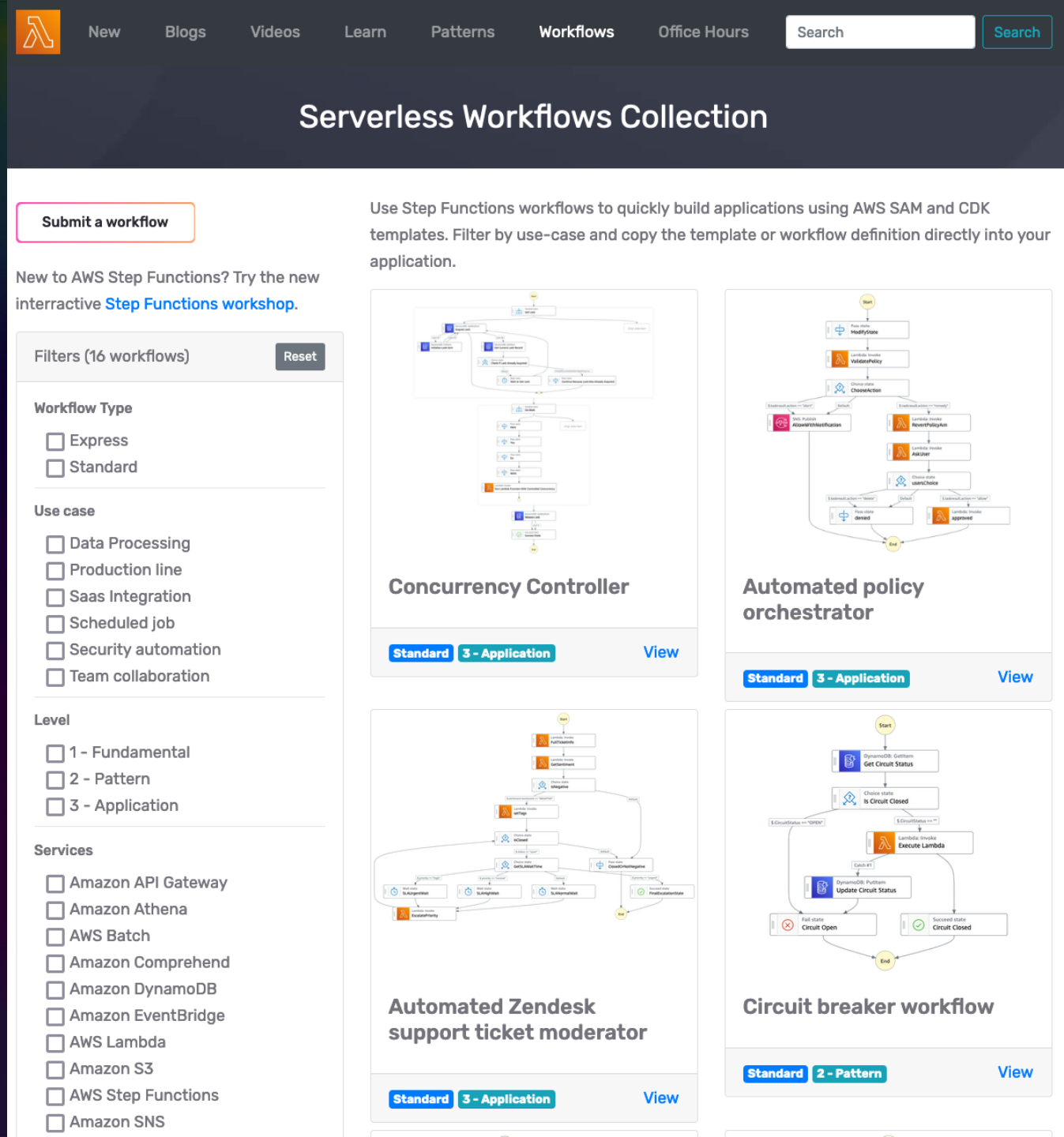
600

Introducing the Serverless Workflows collection

Discover, deploy, and share Step Functions workflows

https://s12d.com/workflows

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





serverlesspresso

A re:Invent coffee bar built on serverless technologies by the serverless DA team.

In the Expo Hall

aws

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

Thank you!

Benjamin Smith
@Benjamin_l_s



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

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