aws INNOVATE

MODERN APPLICATIONS EDITION

20 October, 2022

Handle errors and retries for event-driven applications and workflows

Calvin Ngo

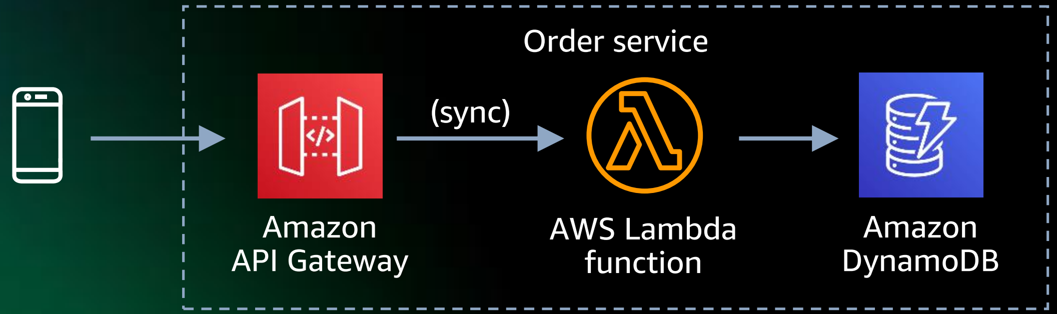
Developer Specialist Solution Architect
Amazon Web Services

```
{
        “message”: “Internal Server Error”
}
```

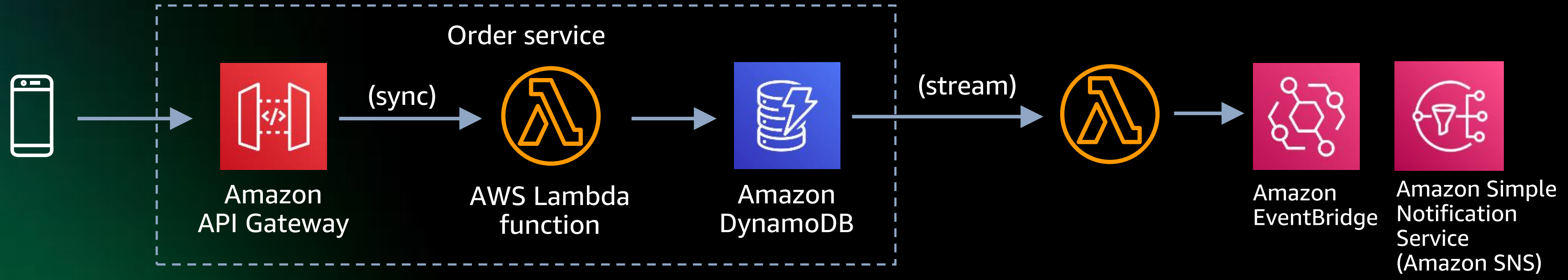
Agenda

- Reference architecture
- Invocation types
- Retry and failure modes
- How to enhance errors visibility
- Externalized orchestration

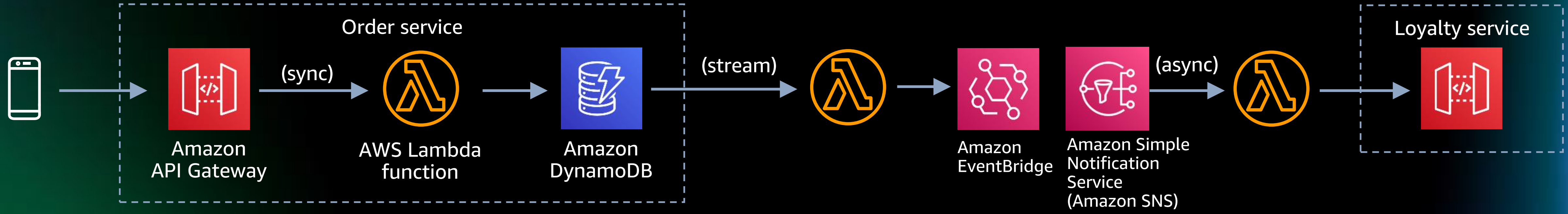
Reference architecture (ecommerce)



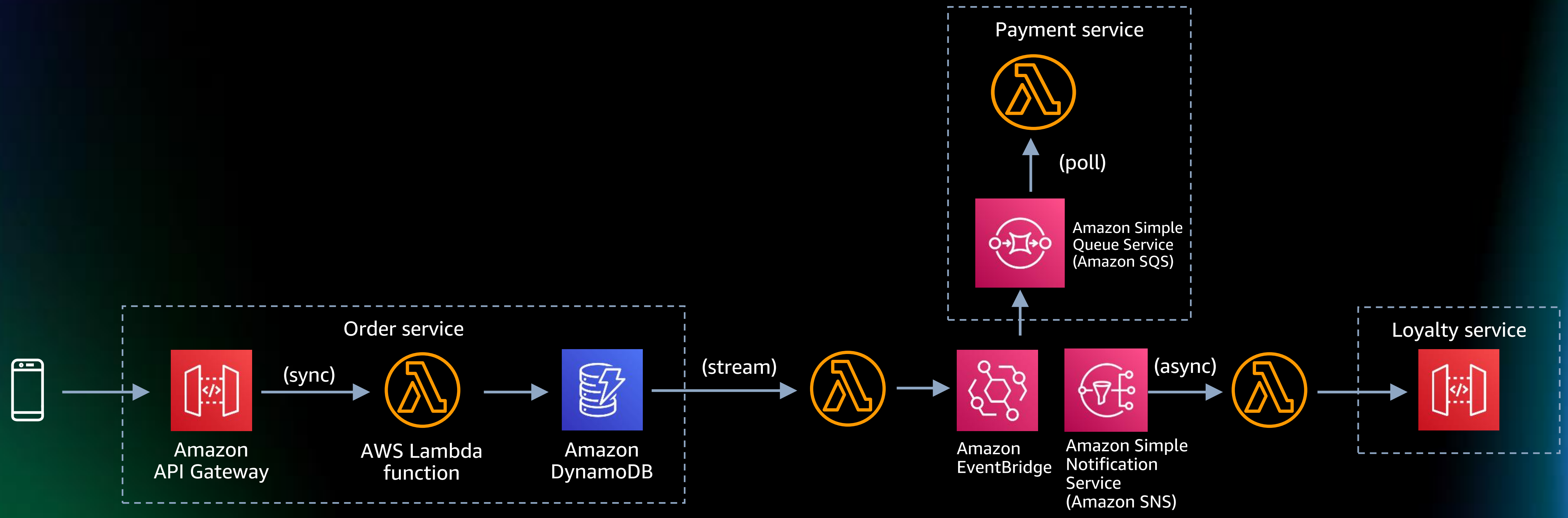
Reference architecture (ecommerce)



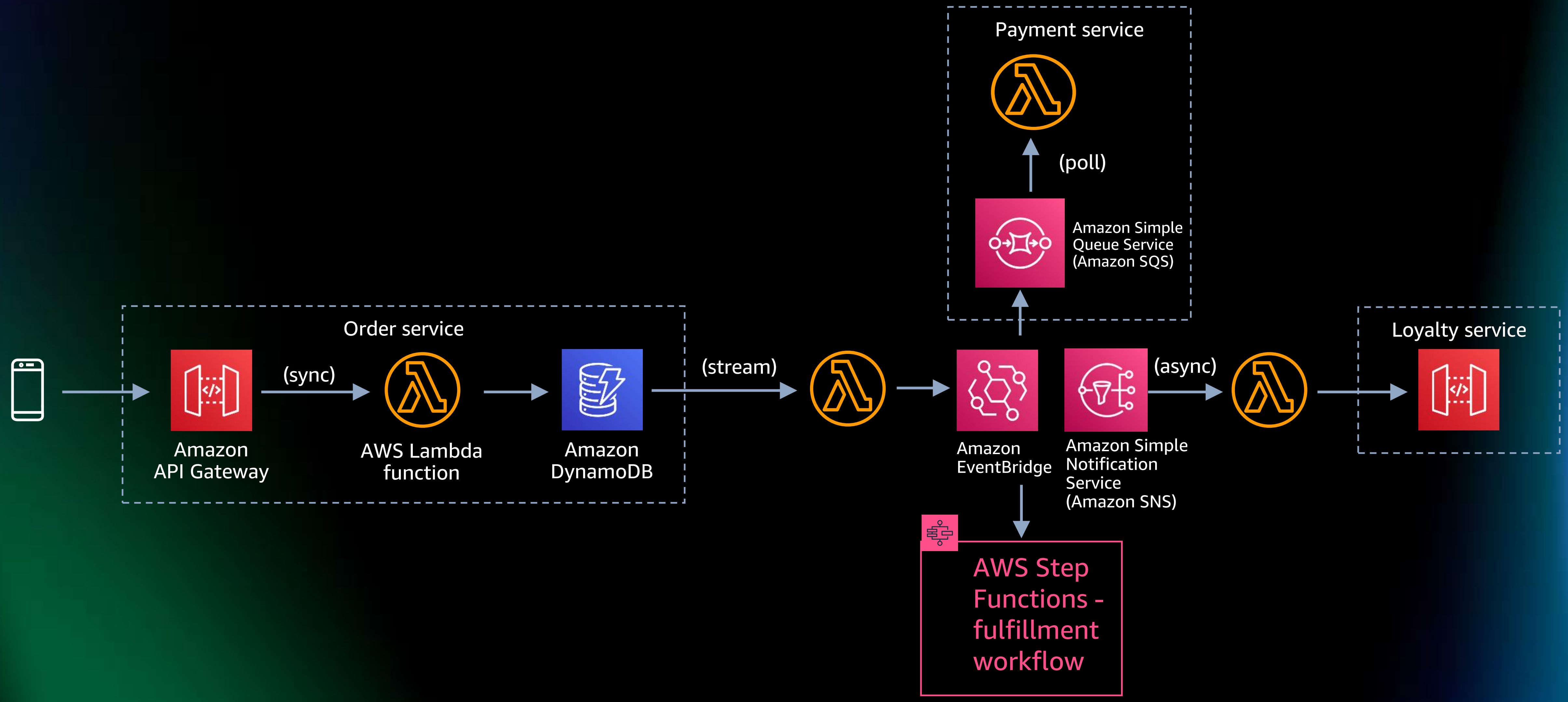
Reference architecture (ecommerce)



Reference architecture (ecommerce)



Reference architecture (ecommerce)


Payment service

(poll)

Amazon Simple Queue Service (Amazon SQS)

Order service

(sync)

Amazon API Gateway

AWS Lambda function

Amazon DynamoDB

(stream)

Amazon EventBridge

Amazon Simple Notification Service (Amazon SNS)

(async)

Loyalty service

AWS Step Functions - fulfillment workflow

AWS Lambda Power tools

“A suite of utilities for Lambda functions to ease adopting best practices such as tracing, structured logging, custom metrics, and more.”

https://awslabs.github.io/aws-lambda-powertools-python/

https://awslabs.github.io/aws-lambda-powertools-java/

#1: Synchronous invocations

Amazon
API Gateway

No retries

Two types of “errors”

Throw custom error types for client errors (e.g. 400)

- Function invocation must succeed

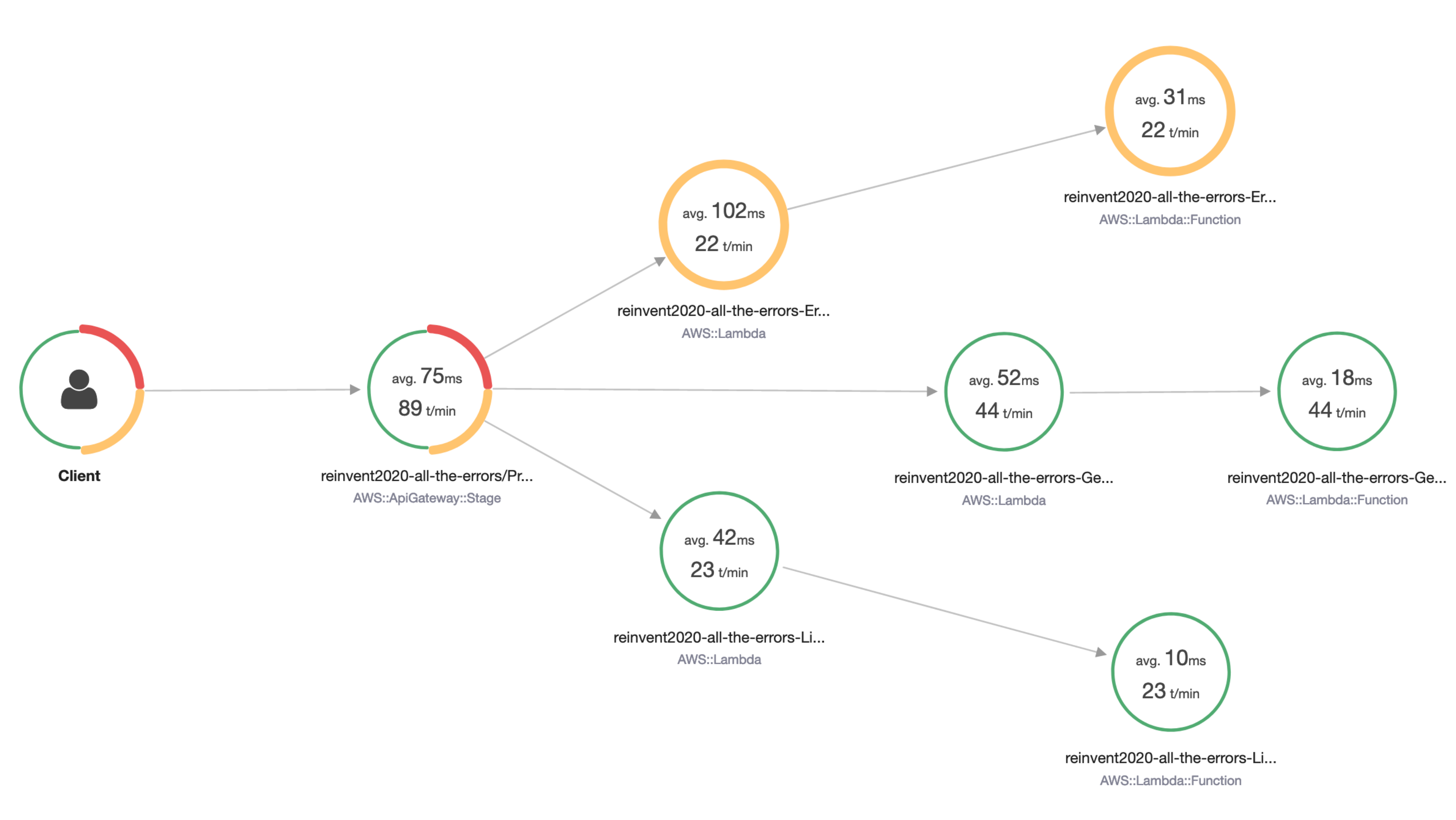
Don’t catch function errors in handler (e.g. AccessDeniedError)

- Use a centralized middleware

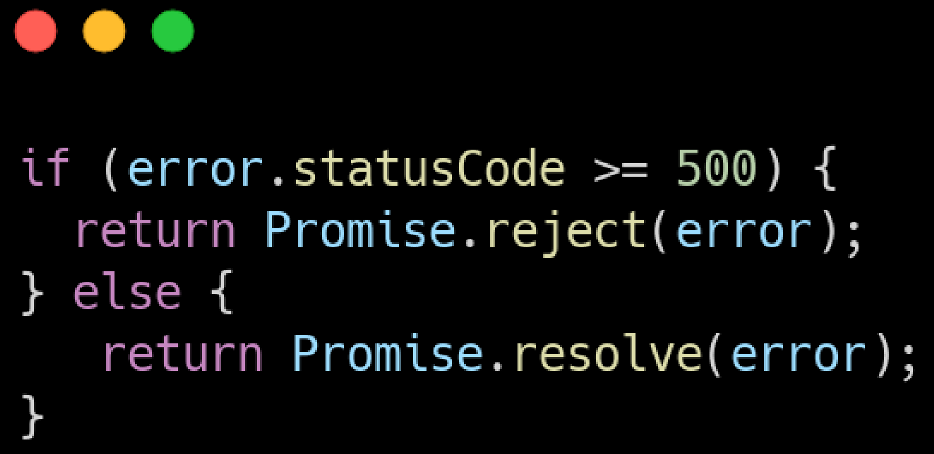
Demo

Why AWS X-Ray tracing

AWS X-Ray



Pro-tip for Node.js promises



```
if (error.statusCode >= 500) {
  return Promise.reject(error);
} else {
   return Promise.resolve(error);
}
```

#2: Asynchronous invocations

Amazon SNS

Amazon Simple Storage Service (Amazon S3)

Amazon EventBridge

2 retries by default

- Configurable with `MaximumRetryAttempts`

Max event age is 6h by default

- Configurable with `MaximumEventAge (6s to 6h)`

Throw all errors to enable alarms and retries

- Function invocation must fail
- Use a middleware to log everything and re-throw

AWS Lambda destinations

Two options: `onSuccess` or `onFailure`

Supported destinations:

Amazon SNS

Amazon SQS

Amazon EventBridge

AWS Lambda

Destination payloads will include metadata

Original payload, function name, # of retries

No need to modify the handler

Demo

#3: Streaming event sources

Amazon DynamoDB

Amazon Kinesis

Batch of records (up to 10k)

Retry the whole batch until message expires

- Blocking behaviour

Same error logic as async (throw everything)

- Also supports Lambda Destinations and retry config

One more option: `BisectBatchOnFunctionError`

Keyword: Idempotence

Demo

New: custom checkpoints

Amazon DynamoDB

Amazon Kinesis

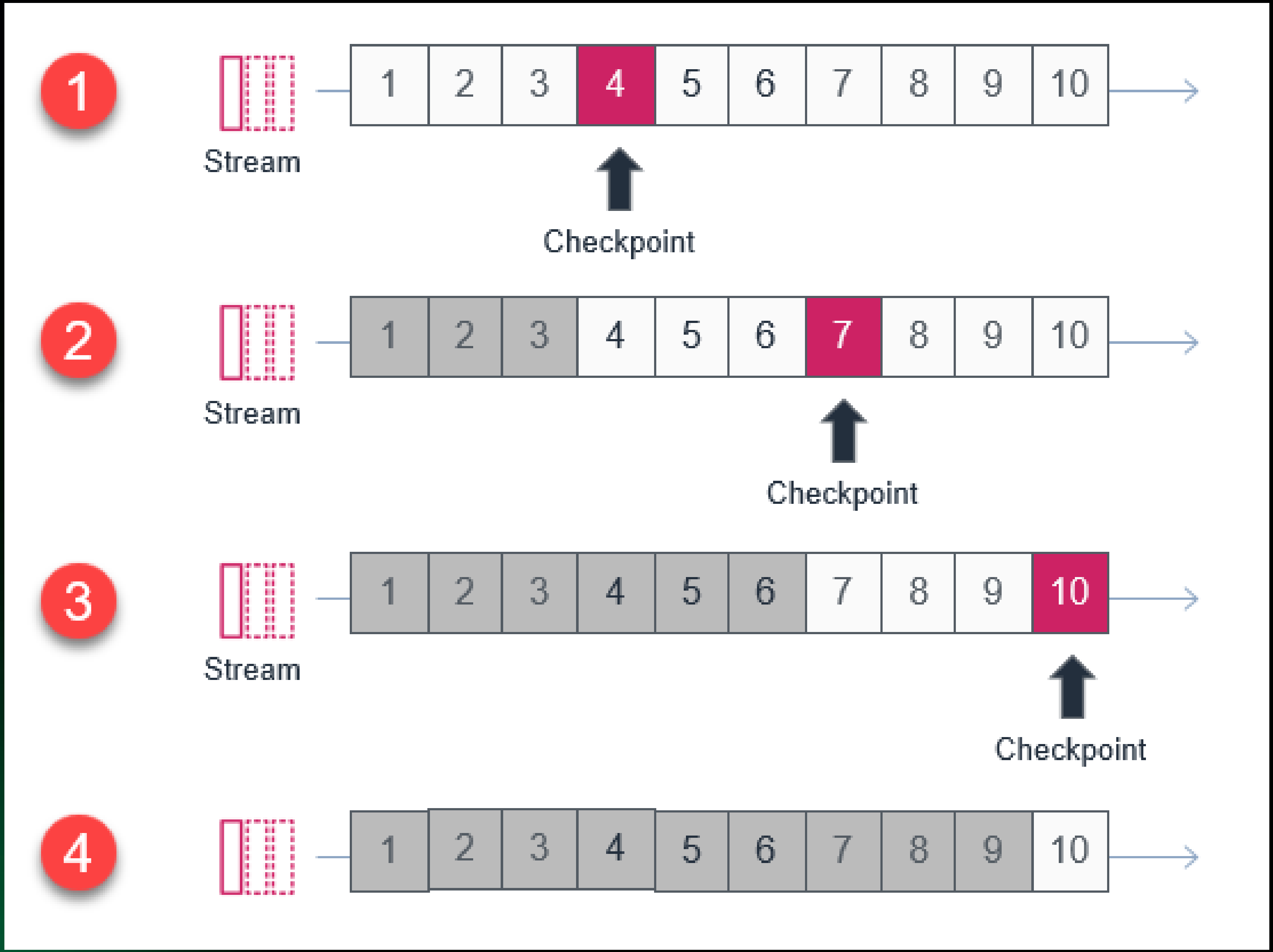
Retry a failed batch, excluding processed records

Configure EventSourceMapping

```
    FunctionResponseTypes: [ReportBatchItemFailures]
```

Update error handler (instead of re-raising)

```
    return {
        "batchItemFailures": [ {
            "itemIdentifier": item_sequence_number
        }]
    }
```


1
Stream
1 2 3 4 5 6 7 8 9 10
Checkpoint
2
Stream
1 2 3 4 5 6 7 8 9 10
Checkpoint
3
Stream
1 2 3 4 5 6 7 8 9 10
Checkpoint
4
1 2 3 4 5 6 7 8 9 10

#4: Poll event sources

Amazon SQS

Very similar to streaming sources (batch)

- But it is synchronous (no Lambda Destinations)
- Use dead-letter queues (DLQs) on Amazon SQS for failed messages

Max batch size: 10k

- Up to 5 baches concurrently
- Max 10 records for first-in, first-out (FIFO) queues

Accumulate more messages in your batch with a time window

- Use `MaximumBatchingWindowInSeconds`
- Must be >1sec to go beyond 10 messages per batch
- Formula: Visibility timeout >= 6 x Function timeout + window

Amazon SQS batch – partial failure

AWS Lambda cleans all messages for you, if batch is successful

In case of partial failure, it is your responsibility

- Delete successfully processed messages
- `DeleteMessage` or `DeleteMessageBatch`

Demo

#5: Orchestration with AWS Step Functions

AWS Step Functions

Built-in support by Amazon States Language

Configurable retry behavior

- Only for some errors
- `MaxAttempts`, `IntervalSeconds`, and `BackoffRate`

Catch and handle error types explicitly

- `Catch`, `ErrorEquals`, and `Next`

Don’t forget to use `Fail` states to mark executions as Failed

Demo

Visit the Modern Applications resource hub

Dive deeper with these resources to help you develop an effective plan for your modernization journey.

- Build modern applications on AWS
- Business value of cloud modernization
- An introduction to event-driven architectures
- Accelerate full-stack web and mobile app development
- Determining the total cost of ownership: Comparing serverless and server-based technologies
- Building event-driven architectures with AWS
- Continuous learning, continuous modernization

https://tinyurl.com/modern-apps-aws

Visit resource hub

AWS Training and Certification

Get started with Free Digital Training for you and your team today

Achieve key milestones and plan your next steps with the AWS Modern Application skills training

Access 500+ free digital courses with AWS Skill Builder

Earn an industry-recognized credential:
AWS Certified Developer – Associate
AWS Certified DevOps – Professional

Create a self-paced learning roadmap
AWS ramp-up guide - Developer
AWS ramp-up guide - DevOps

Thank you for attending AWS Innovate Modern Applications Edition

We hope you found it interesting! A kind reminder to **complete the survey.** Let us know what you thought of today's event and how we can improve the event experience for you in the future.

- aws-apj-marketing@amazon.com
- twitter.com/AWSCloud
- facebook.com/AmazonWebServices
- youtube.com/user/AmazonWebServices
- slideshare.net/AmazonWebServices
- twitch.tv/aws

Thank you!