



AWS  
re:Invent

**A R C 3 0 7 - R**

# Serverless architectural patterns and best practices

**Heitor Lessa**

Principal Serverless Lead, Well-Architected  
Amazon Web Services

# Agenda

Recap on key practices

Patterns – Starring “Call Me Maybe”, The “cherry-pick” and more

# What we're not covering

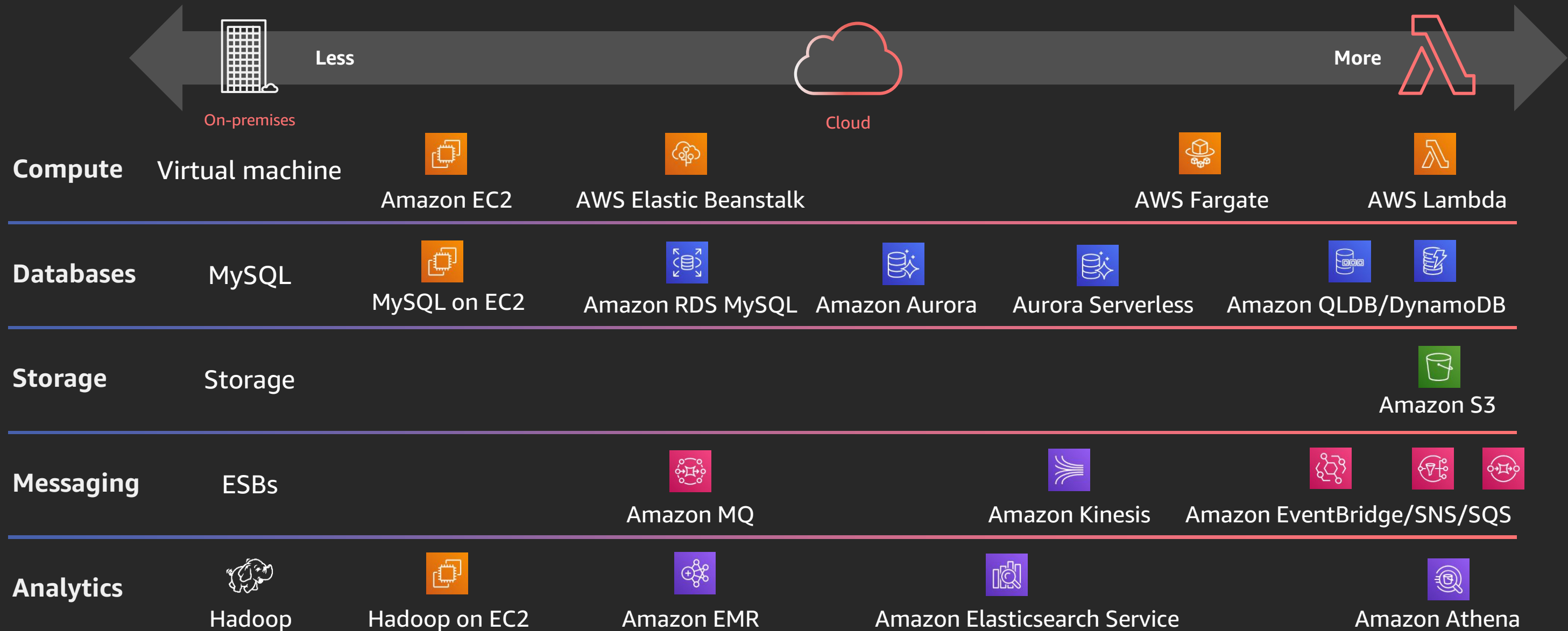
Introduction to AWS serverless platform

Patterns for serverless AI/ML

Best practices for serverless at scale

# Before we start

# AWS operational responsibility models



# Quick recap – Key practices

# Quick recap: An easier getting-started experience

Lambda > Applications > Create application


## Create a Lambda application

An AWS Lambda application is a combination of Lambda functions, event sources, and other resources that work together to perform tasks. Choose an option below to create an application with sample code and a continuous delivery pipeline.

### Choose a sample application

#### Serverless API backend


A RESTful web API that uses DynamoDB to manage state.

Made by: AWS  Uses: API Gateway, DynamoDB, Lambda

Runtime: Node.js 10.x

#### File processing


Use Amazon S3 to trigger AWS Lambda to process data immediately after an upload. For example, you can use Lambda to thumbnail images, transcode videos, index files, process content, and aggregate and filter data.

Made by: AWS  Uses: S3, Lambda

Runtime: Node.js 10.x

#### Scheduled job


Schedule AWS Lambda functions using AWS CloudWatch events. This application creates a Lambda function that is triggered by a scheduled event.

Made by: AWS  Uses: CloudWatch, Lambda

Runtime: Node.js 10.x

#### Notifications processing


Use a Lambda function to subscribe to an Amazon SNS topic. When a message is published to an SNS topic that has a Lambda function subscribed to it, the function is invoked.

Made by: AWS  Uses: SNS, Lambda

Runtime: Node.js 10.x

#### Queue processing

Use an AWS Lambda function to process messages from an Amazon SQS queue. With this application, you can offload tasks from one or more EC2 instances to a Lambda function.

Made by: AWS  Uses: SQS, Lambda


Runtime: Node.js 10.x

### Serverless API backend

Use Lambda with API Gateway and DynamoDB to build fault-tolerant web APIs that scale automatically and run only when needed.


#### Serverless API backend

A RESTful web API that uses DynamoDB to manage state.

Made by: AWS  Uses: API Gateway, DynamoDB, Lambda

Runtime: Node.js 10.x

#### Architecture



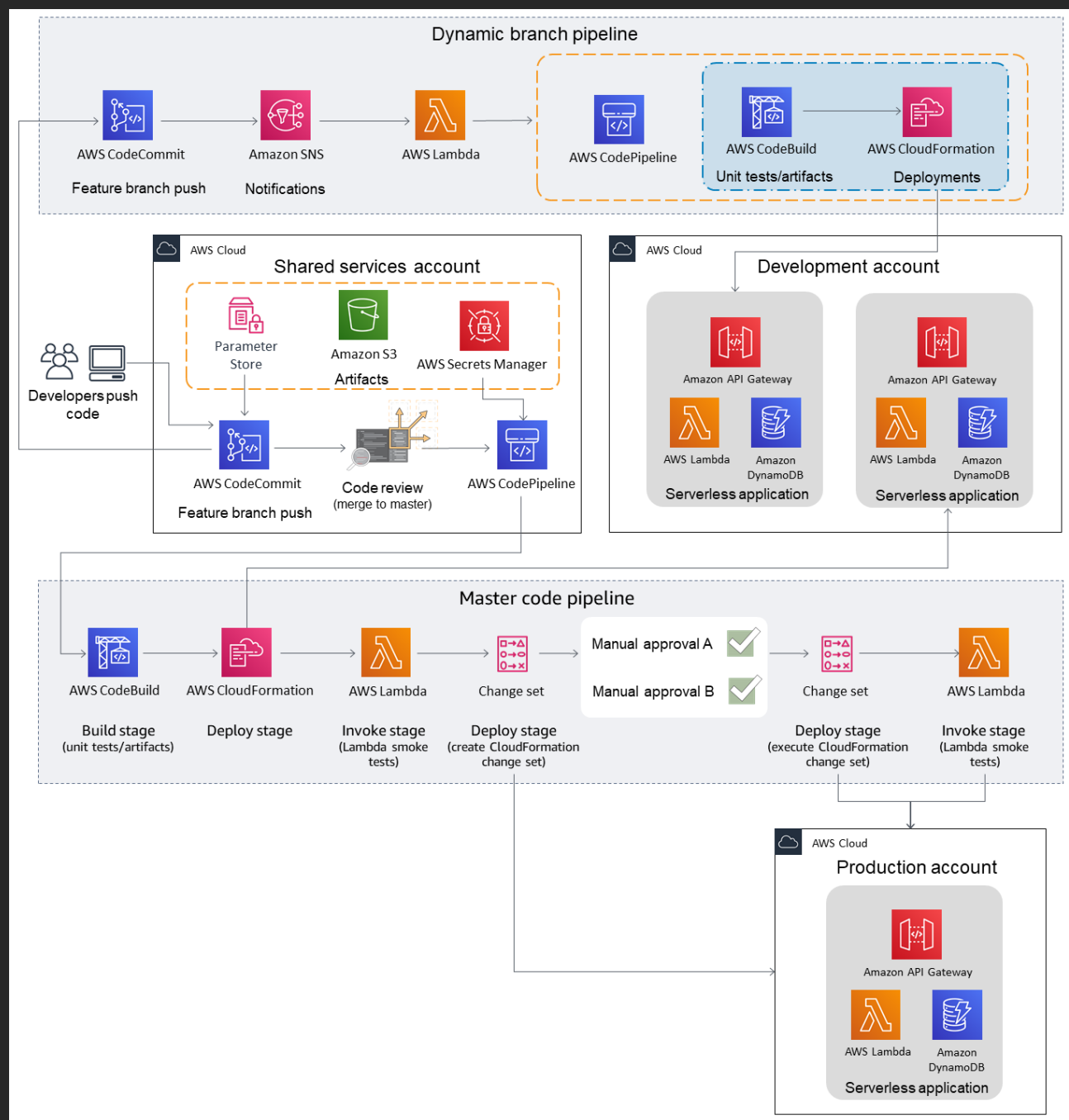
```
graph LR; AG[Amazon API Gateway] --> L[AWS Lambda x 3]; L --> DB[Amazon DynamoDB]
```

#### Services used

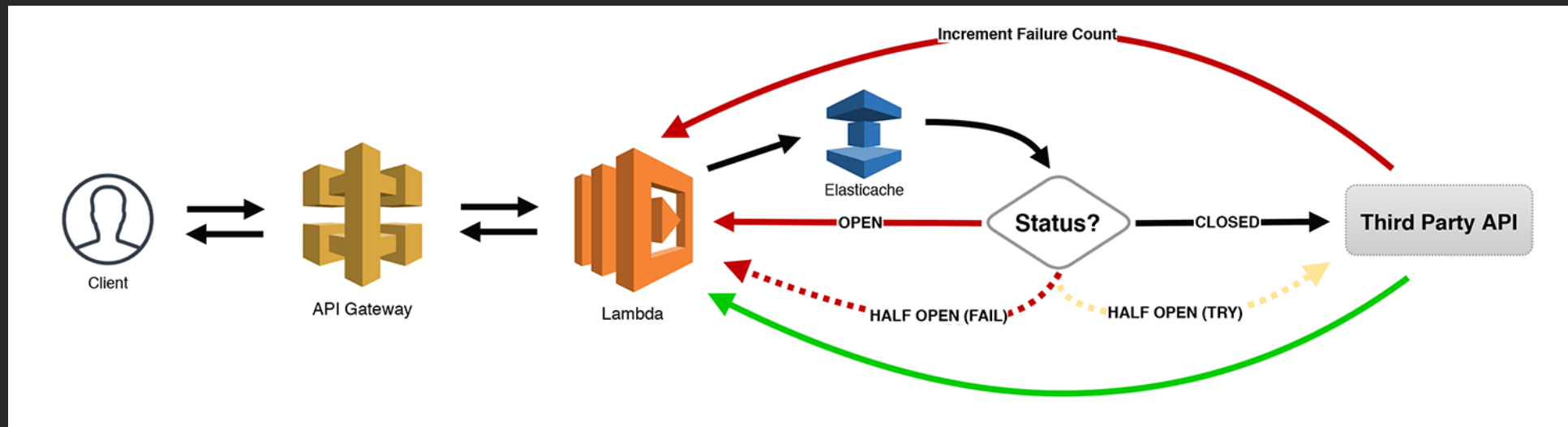
Source control CodeCommit or GitHub	Continuous delivery CodePipeline	Application resources API Gateway DynamoDB Lambda
Build and test CodeBuild	Deployment AWS CloudFormation	



# Quick recap: Serverless CI/CD for enterprise



# Quick recap: Serverless microservices patterns



Circuit Breaker and many more by Jeremy Daly



# Quick recap: Lambda memory tuning

AWS Lambda Power Tuning Results



**Best Cost**  
1536MB

**Best Time**  
3008MB

**Worst Cost**  
3008MB

**Worst Time**  
128MB



# Performance test – Memory tuning, edge to regional API

▶ STATISTICS

Expand all groups | Collapse all groups

Requests ^	🔄 Executions					🕒 Response Time (ms)							
	Total ↕	OK ↕	KO ↕	% KO ↕	Cnt/s ↕	Min ↕	50th pct ↕	75th pct ↕	95th pct ↕	99th pct ↕	Max ↕	Mean ↕	Std Dev ↕
Global Information	59221	59195	26	0%	140.002	18	158	433	813	1051	7655	294	307
Retrieve Profile	12459	12459	0	0%	29.454	246	429	519	682	802	2300	460	117
Search Rand Flights	8495	8495	0	0%	20.083	19	26	28	38	60	412	28	10
Search Flights	8426	8426	0	0%	19.92	18	26	28	38	62	220	28	9
Stripe token	8495	8495	0	0%	20.083	304	349	364	415	1448	7655	370	188
Charge API	8495	8469	26	0%	20.083	421	781	829	1131	2005	4841	831	228
List bookings	4382	4382	0	0%	10.359	22	36	44	63	90	185	40	14
Process booking	8469	8469	0	0%	20.021	49	94	105	131	179	787	96	25

Retrieve profile

**P99** – 802ms to 237ms

**Min** – 246ms to 70ms

▶ STATISTICS

Expand all groups | Collapse all groups

Requests ^	Executions					Response Time (ms)							
	Total ⇅	OK ⇅	KO ⇅	% KO ⇅	Cnt/s ⇅	Min ⇅	50th pct ⇅	75th pct ⇅	95th pct ⇅	99th pct ⇅	Max ⇅	Mean ⇅	Std Dev ⇅
Global Information	58379	58378	1	0%	138.012	18	106	334	537	683	3531	190	219
Retrieve Profile	12422	12422	0	0%	29.366	70	127	152	207	237	1244	137	41
Search Rand Flights	8268	8268	0	0%	19.546	19	26	28	33	67	257	28	10
Search Flights	8506	8506	0	0%	20.109	18	26	28	38	67	268	28	11
Stripe token	8268	8267	1	0%	19.546	182	349	364	416	1488	3531	369	145
List bookings	4381	4381	0	0%	10.357	21	34	42	64	103	179	39	15
Charge API	8267	8267	0	0%	19.544	471	561	602	693	1770	2831	598	195
Process booking	8267	8267	0	0%	19.544	44	93	106	131	171	443	96	22

# Quick recap: Saga within the serverless airline

Flight App

LGW ↔ MAD

Review your selection

DEPARTURE

LGW  
London Gatwick

16 JAN 2019

MAD  
Madrid Barajas

08:00

2h15m

11:15

400 EUR

Flight No #1812

Payment details

Name

Name on card

Country

Postcode

Postcode

Card number

1234 1234 1234 1234

Expiry date

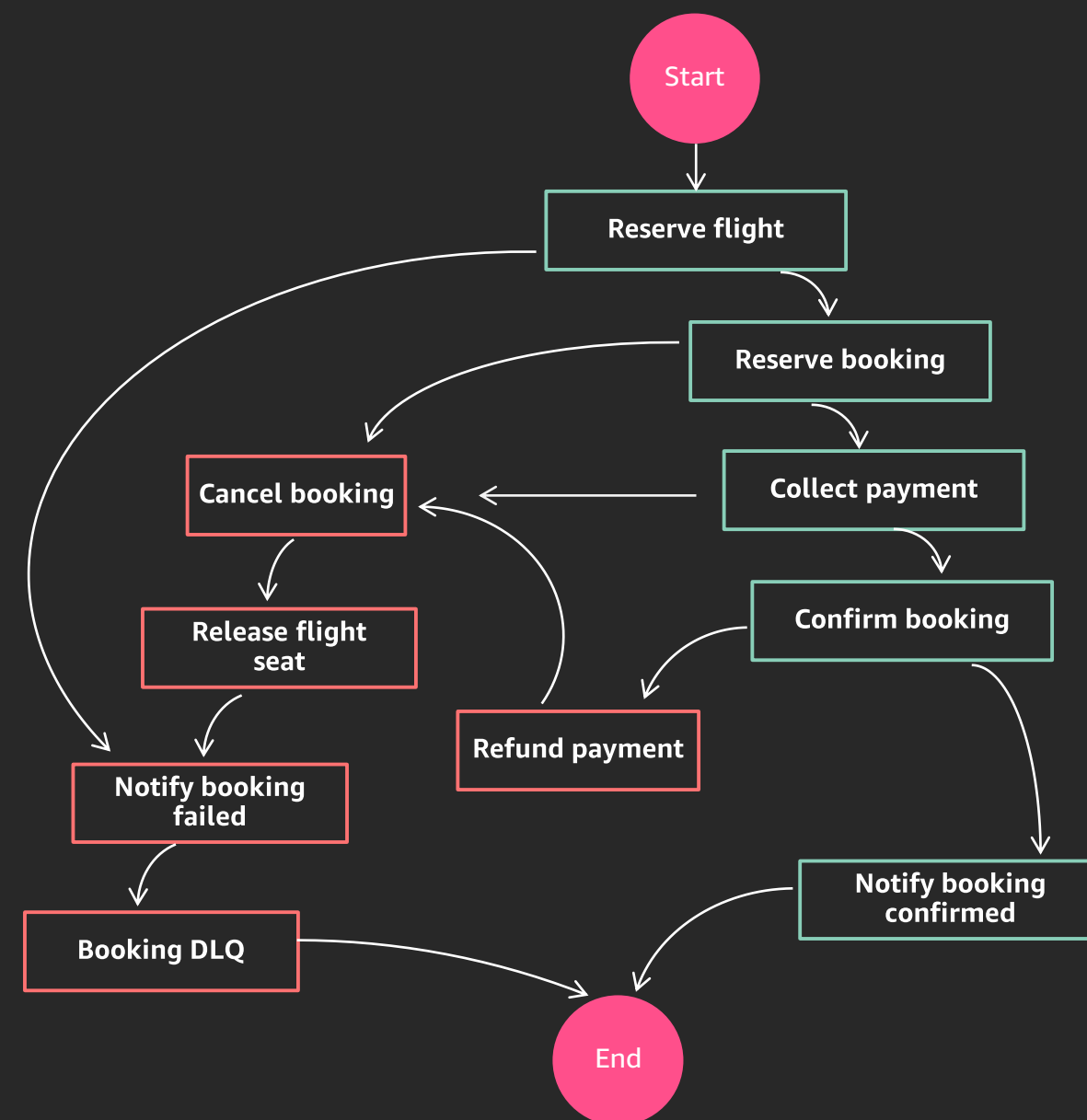
MM / YY

CVC

CVC

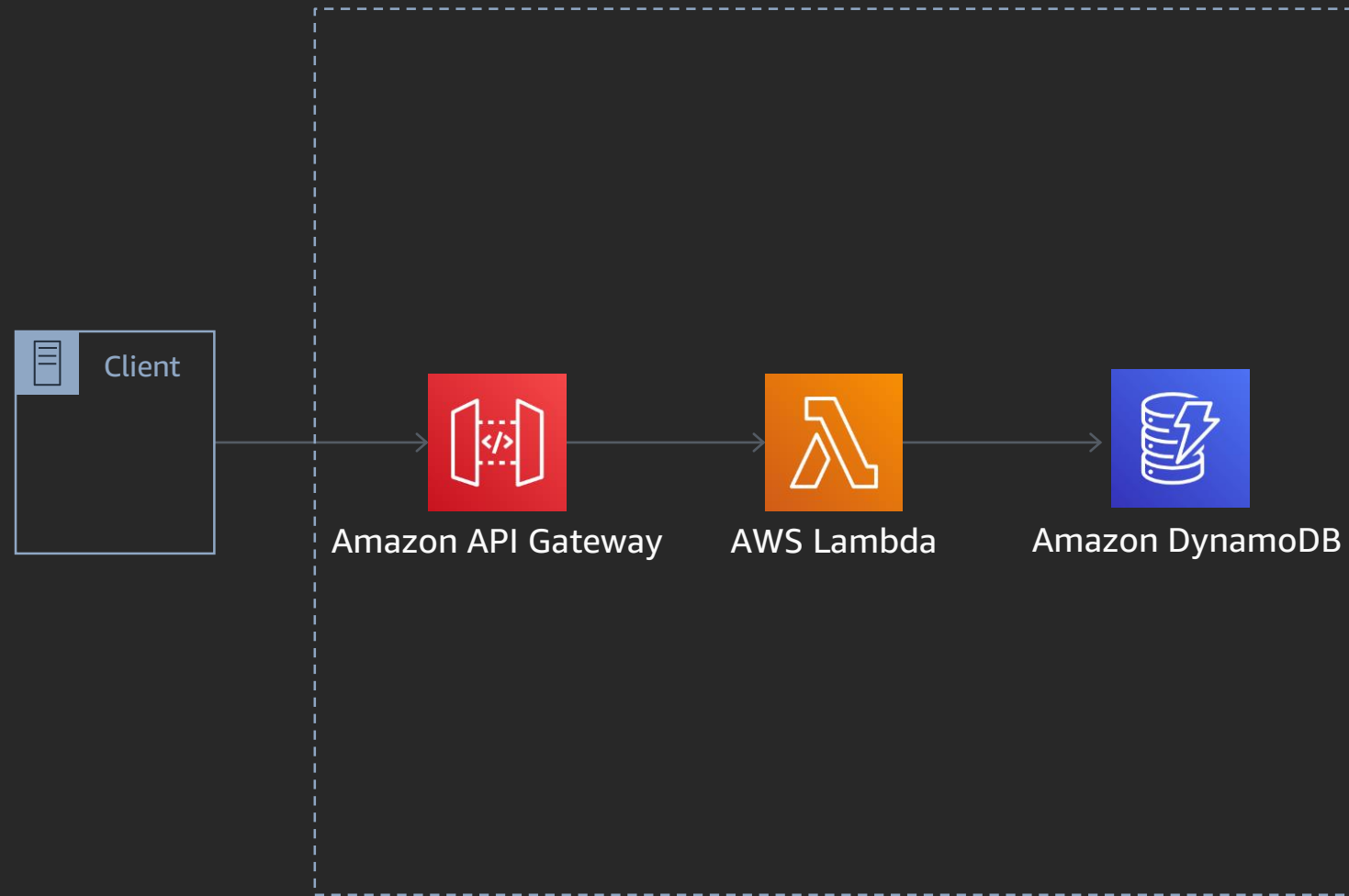
AGREE AND PAY NOW >

Process booking state machine



# Pattern: The comfortable “REST”

# Pattern: The comfortable “REST”



OPERATIONS

RELIABILITY

SECURITY

PERFORMANCE

COST

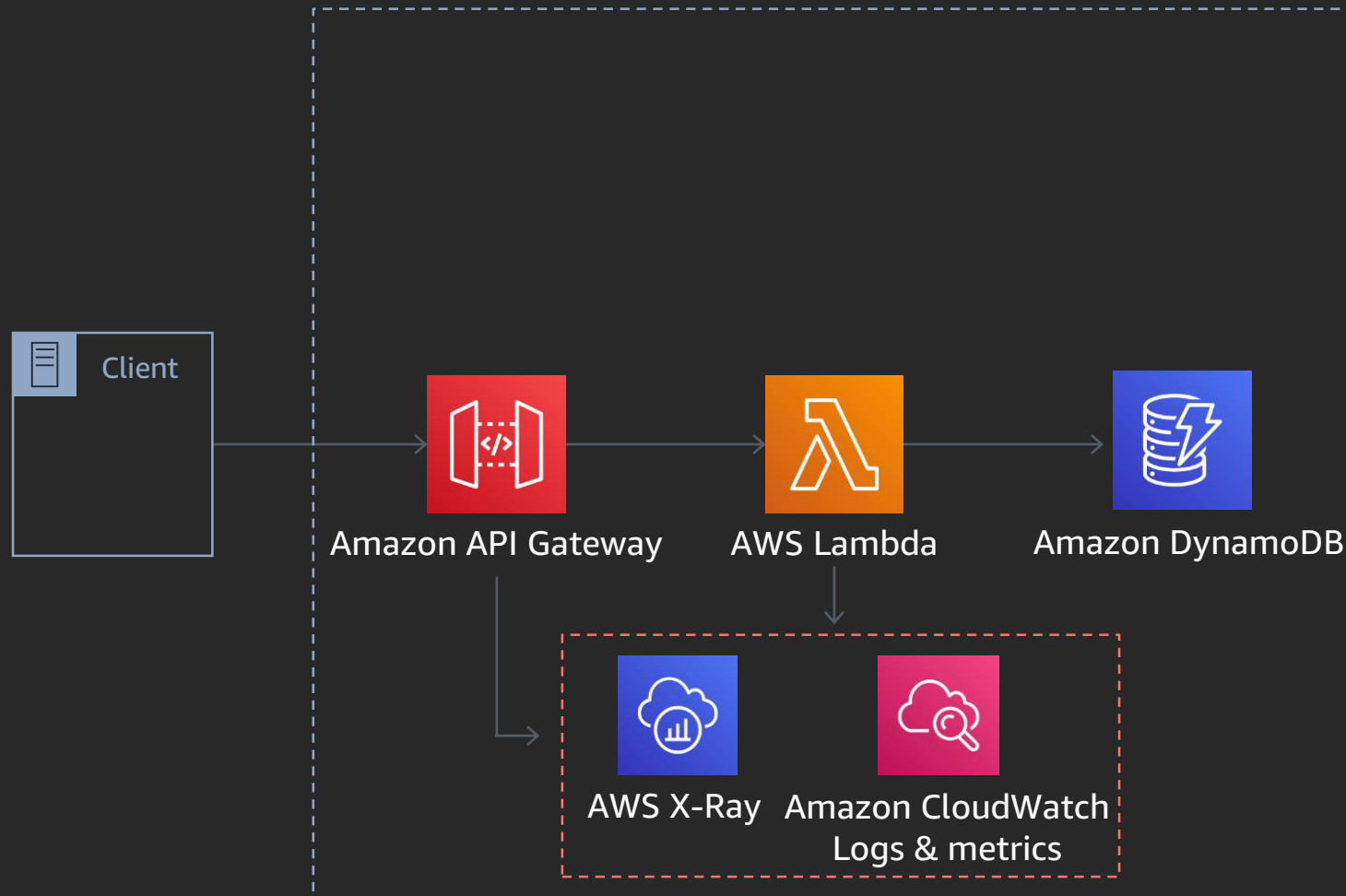


# Pattern: The comfortable “REST”

## Best practices

Enable access logs, structure logs and instrument your code

Create metrics async with CloudWatch Embedded Metric Format new



OPERATIONS

RELIABILITY

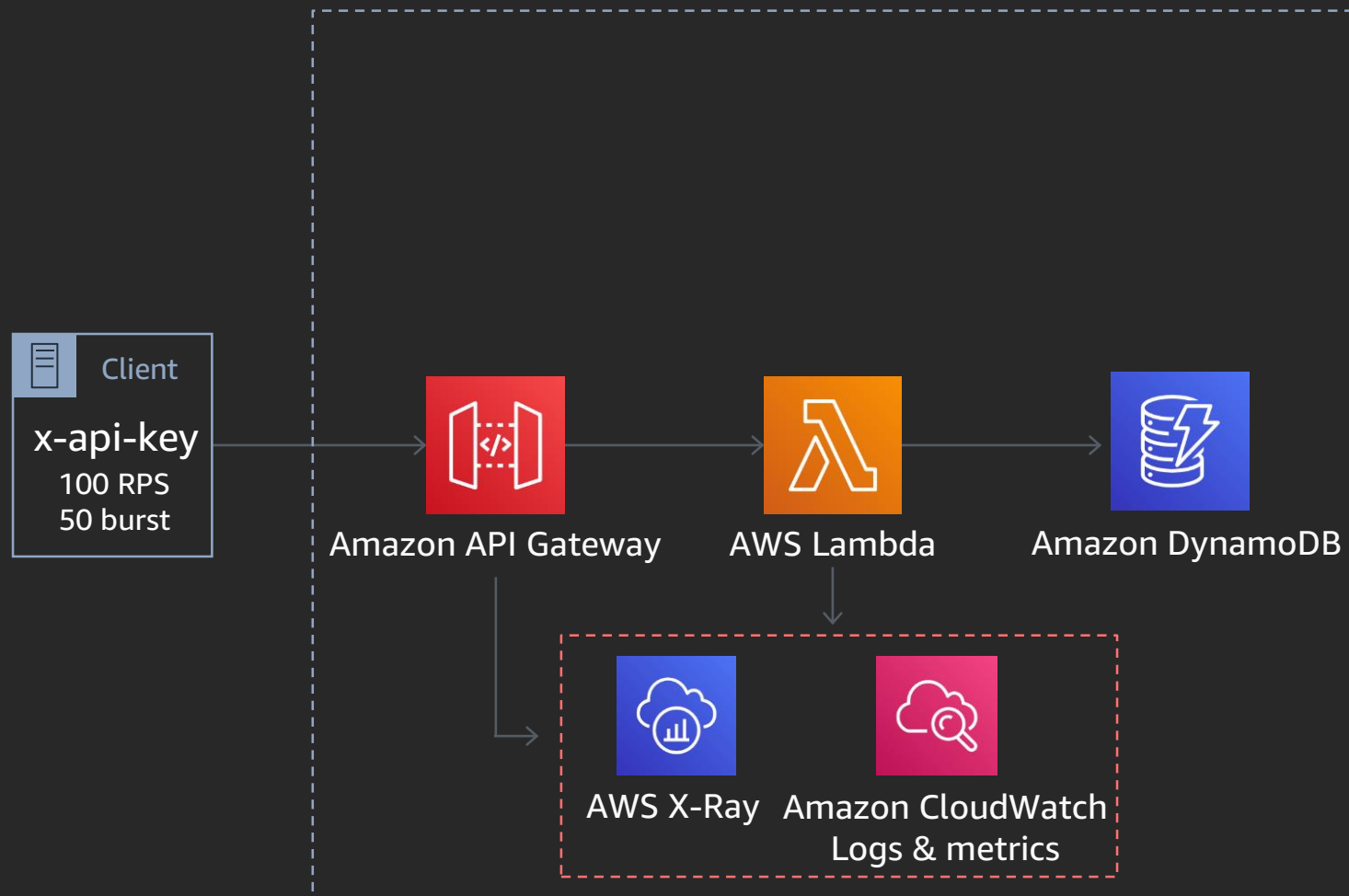
SECURITY

PERFORMANCE

COST



# Pattern: The comfortable “REST”



## Best practices

- Enable access logs, structure logs and instrument your code
- Create metrics async with CloudWatch Embedded Metric Format new
- Regulate inbound access rates

OPERATIONS

RELIABILITY

SECURITY

PERFORMANCE

COST

# Pattern: The comfortable “REST”



## Best practices

- Enable access logs, structure logs and instrument your code
- Create metrics async with CloudWatch Embedded Metric Format new
- Regulate inbound access rates
- Authorize consumers. Manage secrets with AWS Secrets Manager

OPERATIONS

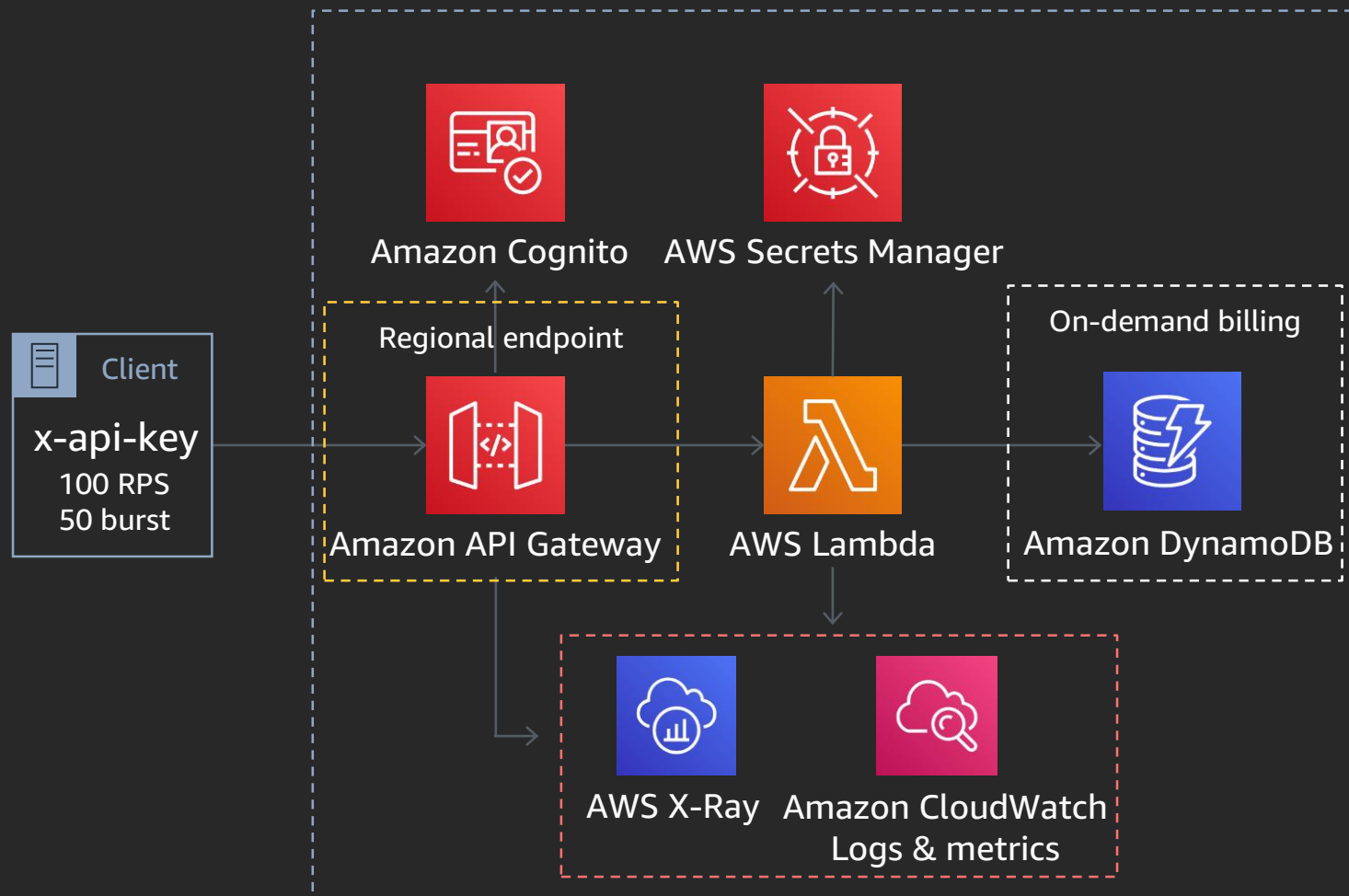
RELIABILITY

SECURITY

PERFORMANCE

COST

# Pattern: The comfortable “REST”



## Best practices

- Enable access logs, structure logs and instrument your code
- Create metrics async with CloudWatch Embedded Metric Format new
- Regulate inbound access rates
- Authorize consumers. Manage secrets with AWS Secrets Manager
- On-demand tables support up to 40K read/write request units
- Regional endpoints support HTTP2

OPERATIONS

RELIABILITY

SECURITY

PERFORMANCE

COST

# Pattern: The comfortable “REST”



## Best practices

- Enable access logs, structure logs and instrument your code
- Create metrics async with CloudWatch Embedded Metric Format new
- Regulate inbound access rates
- Authorize consumers. Manage secrets with AWS Secrets Manager
- On-demand tables support up to 40K read/write request units
- Regional endpoints support HTTP2
- Use Lambda Power Tuning for perf/cost tuning

OPERATIONS

RELIABILITY

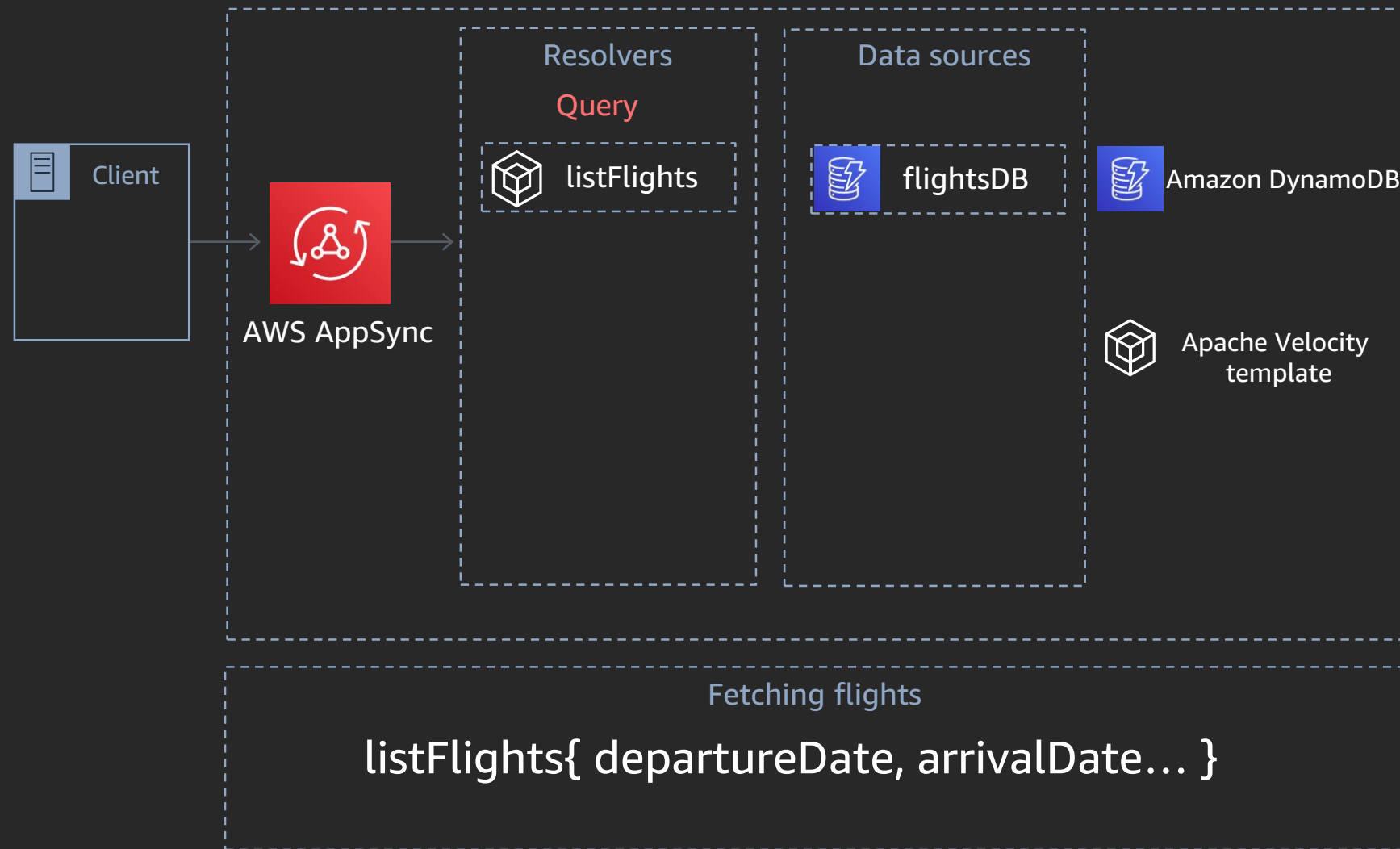
SECURITY

PERFORMANCE

COST

# Pattern: The “cherry-pick”

# Pattern: The “cherry-pick” (GraphQL API)



OPERATIONS

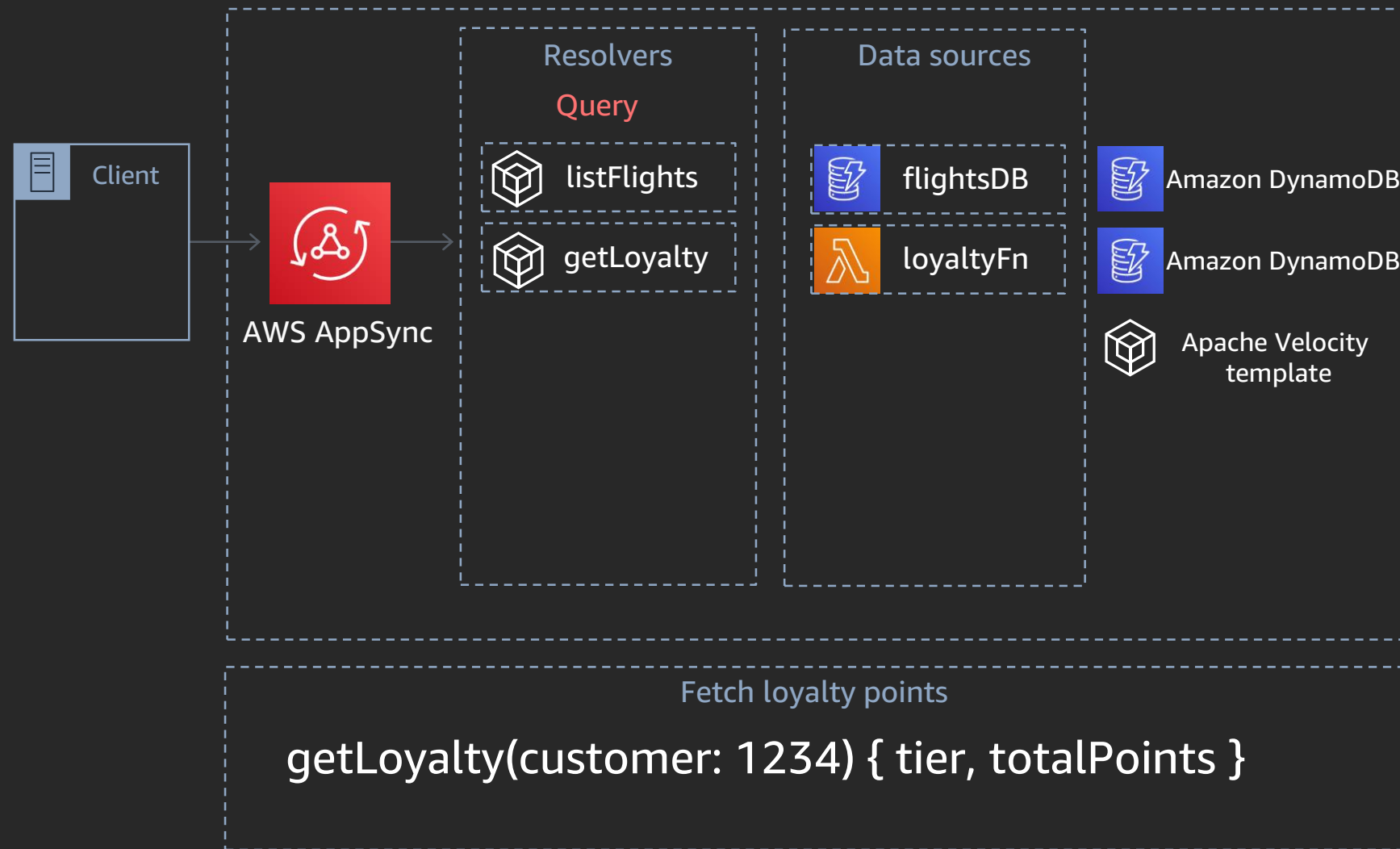
RELIABILITY

SECURITY

PERFORMANCE

COST

# Pattern: The “cherry-pick” (GraphQL API)



## Best practices

Use Lambda for complex logic

OPERATIONS

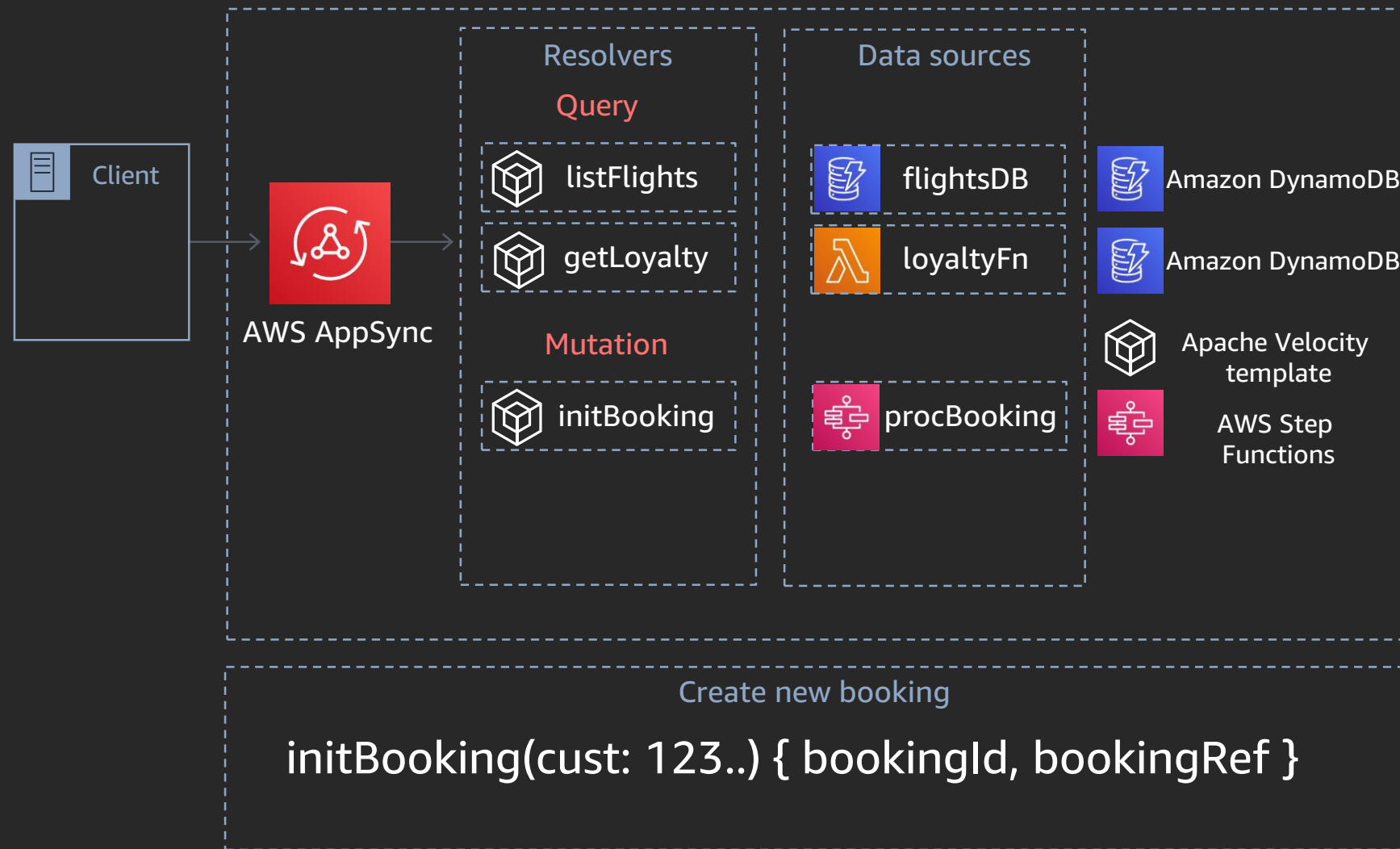
RELIABILITY

SECURITY

PERFORMANCE

COST

# Pattern: The “cherry-pick” (GraphQL API)



## Best practices

- Use Lambda for complex logic
- Use state machines for long transactions. Pipeline resolvers for simpler transactions

OPERATIONS

RELIABILITY

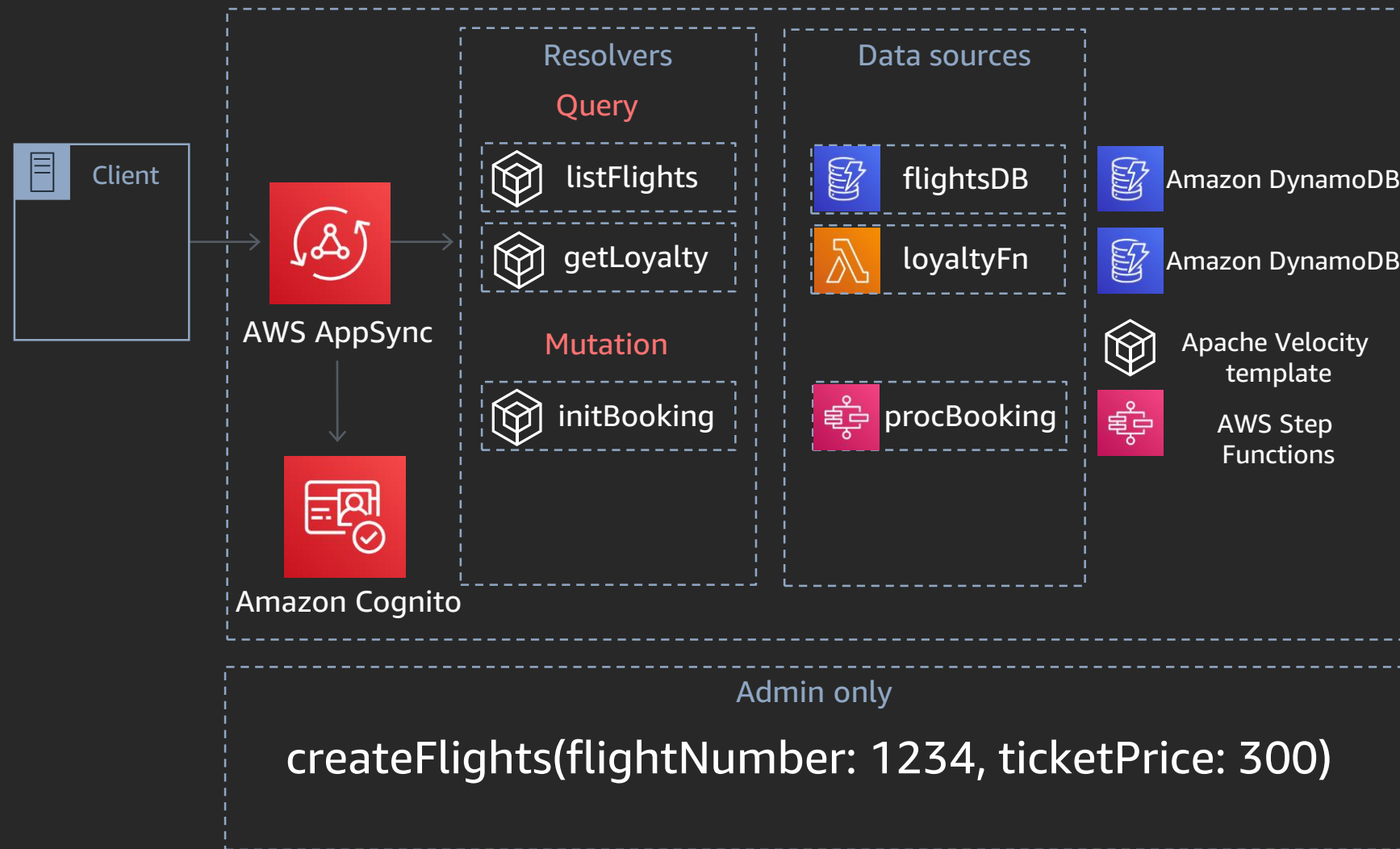
SECURITY

PERFORMANCE

COST



# Pattern: The “cherry-pick” (GraphQL API)



## Best practices

- Use Lambda for complex logic
- Use state machines for long transactions. Pipeline resolvers for simpler transactions
- Enforce authorization at API, data field and operation level

OPERATIONS

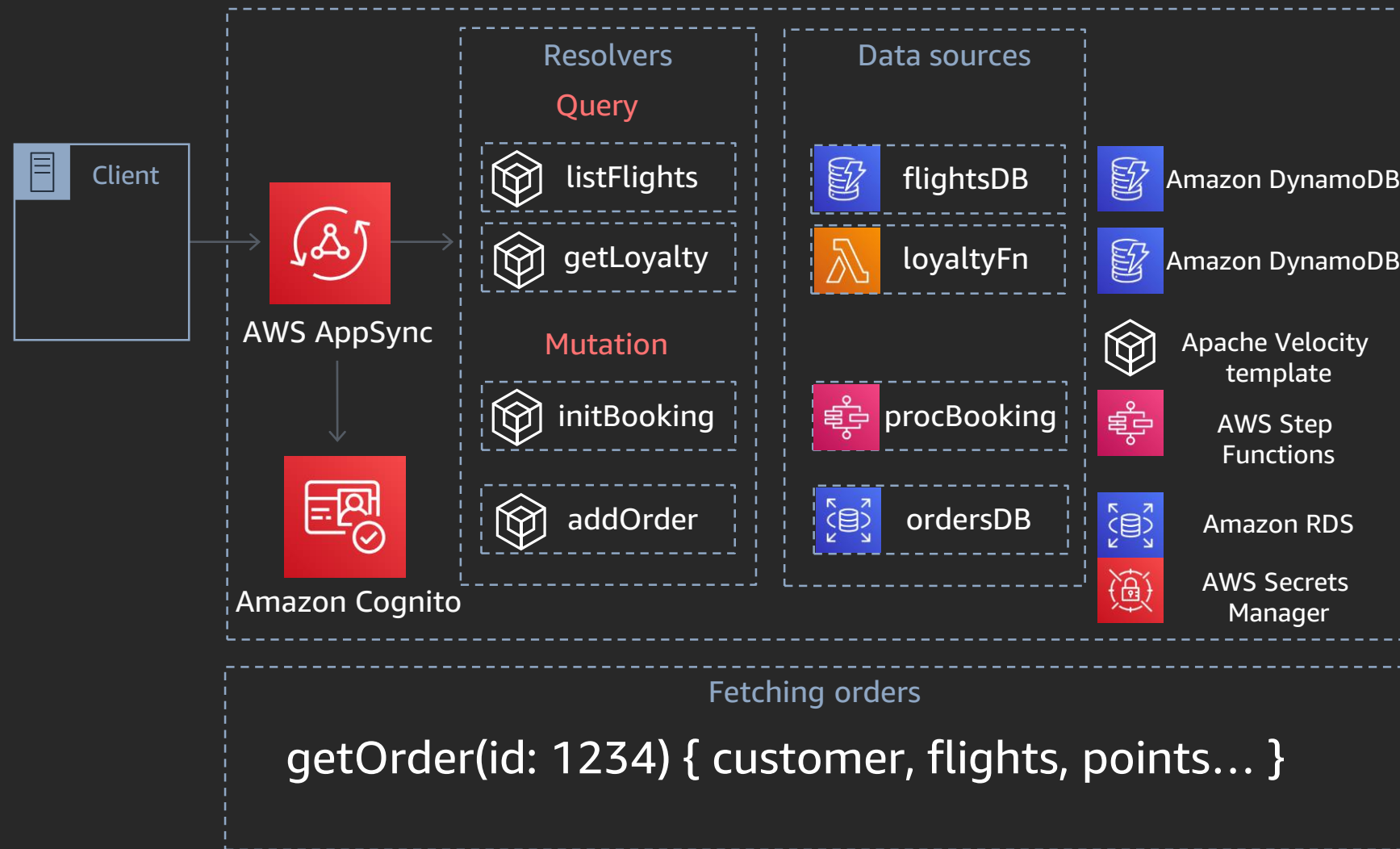
RELIABILITY

SECURITY

PERFORMANCE

COST

# Pattern: The “cherry-pick” (GraphQL API)



## Best practices

- Use Lambda for complex logic
- Use state machines for long transactions. Pipeline resolvers for simpler transactions
- Enforce authorization at API, data field and operation level
- Use purpose-built databases

OPERATIONS

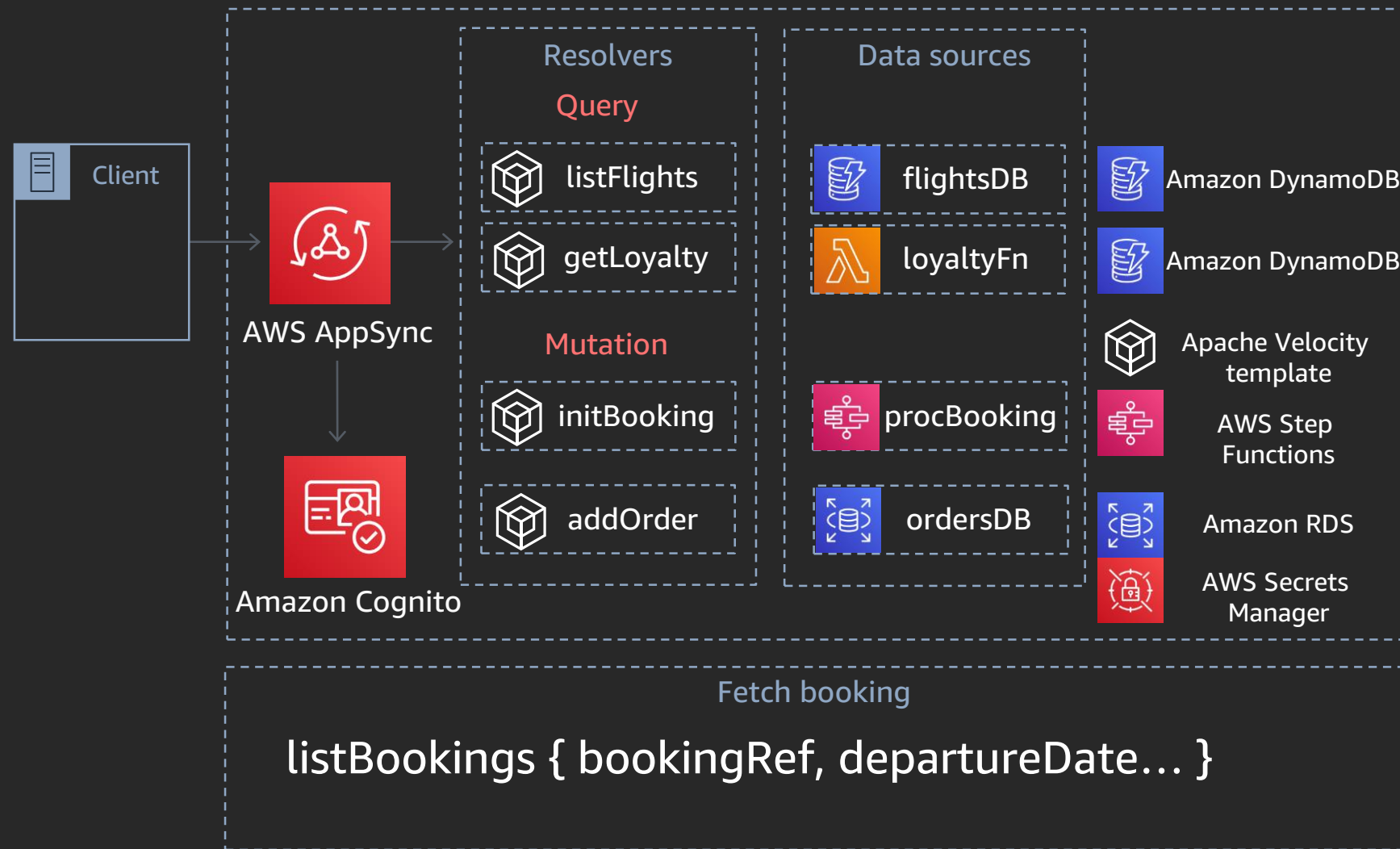
RELIABILITY

SECURITY

PERFORMANCE

COST

# Pattern: The “cherry-pick” (GraphQL API)



## Best practices

- Use Lambda for complex logic
- Use state machines for long transactions. Pipeline resolvers for simpler transactions
- Enforce authorization at API, data field and operation level
- Use purpose-built databases
- Select only data you need
- Enable caching new

OPERATIONS

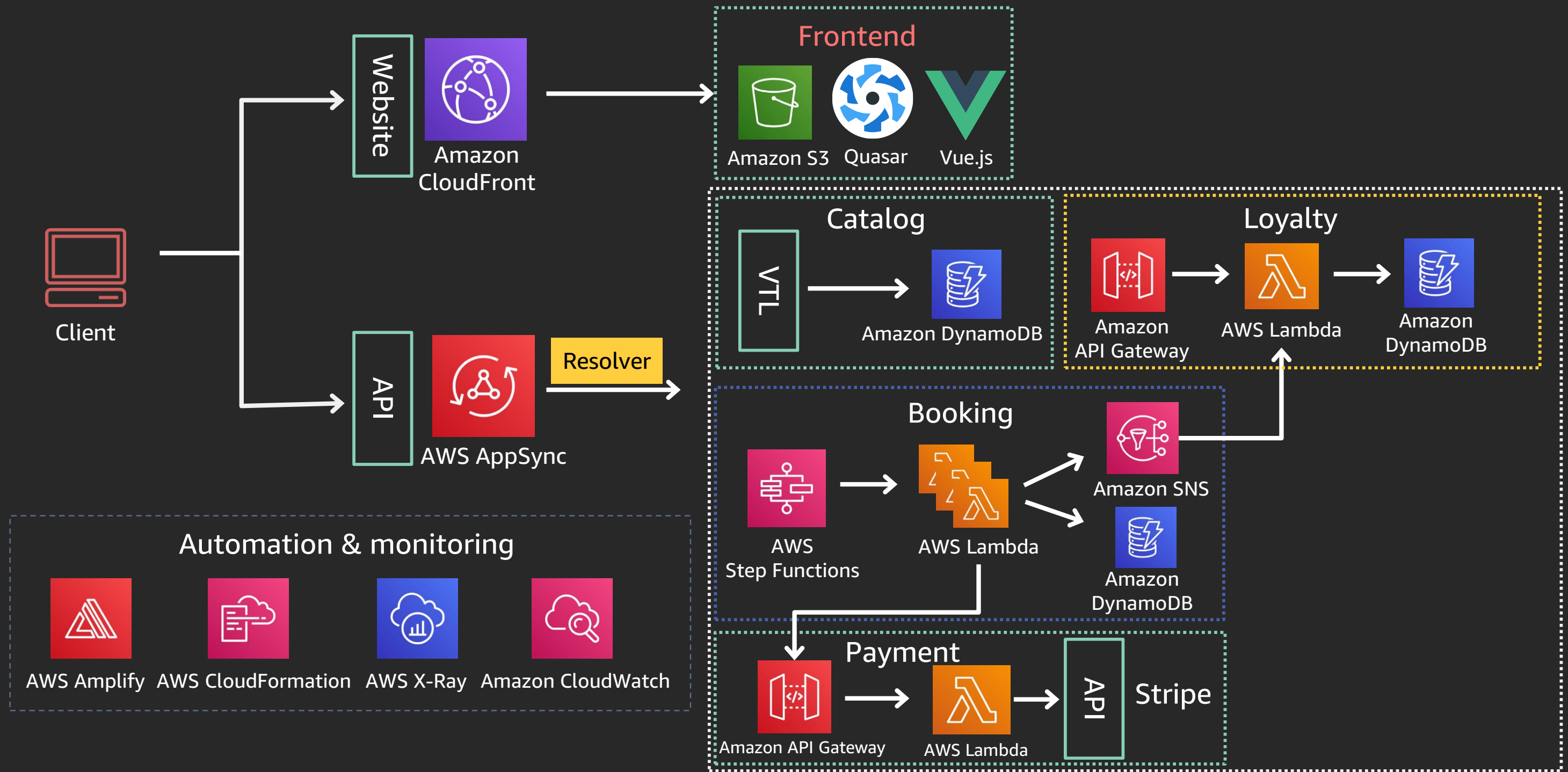
RELIABILITY

SECURITY

PERFORMANCE

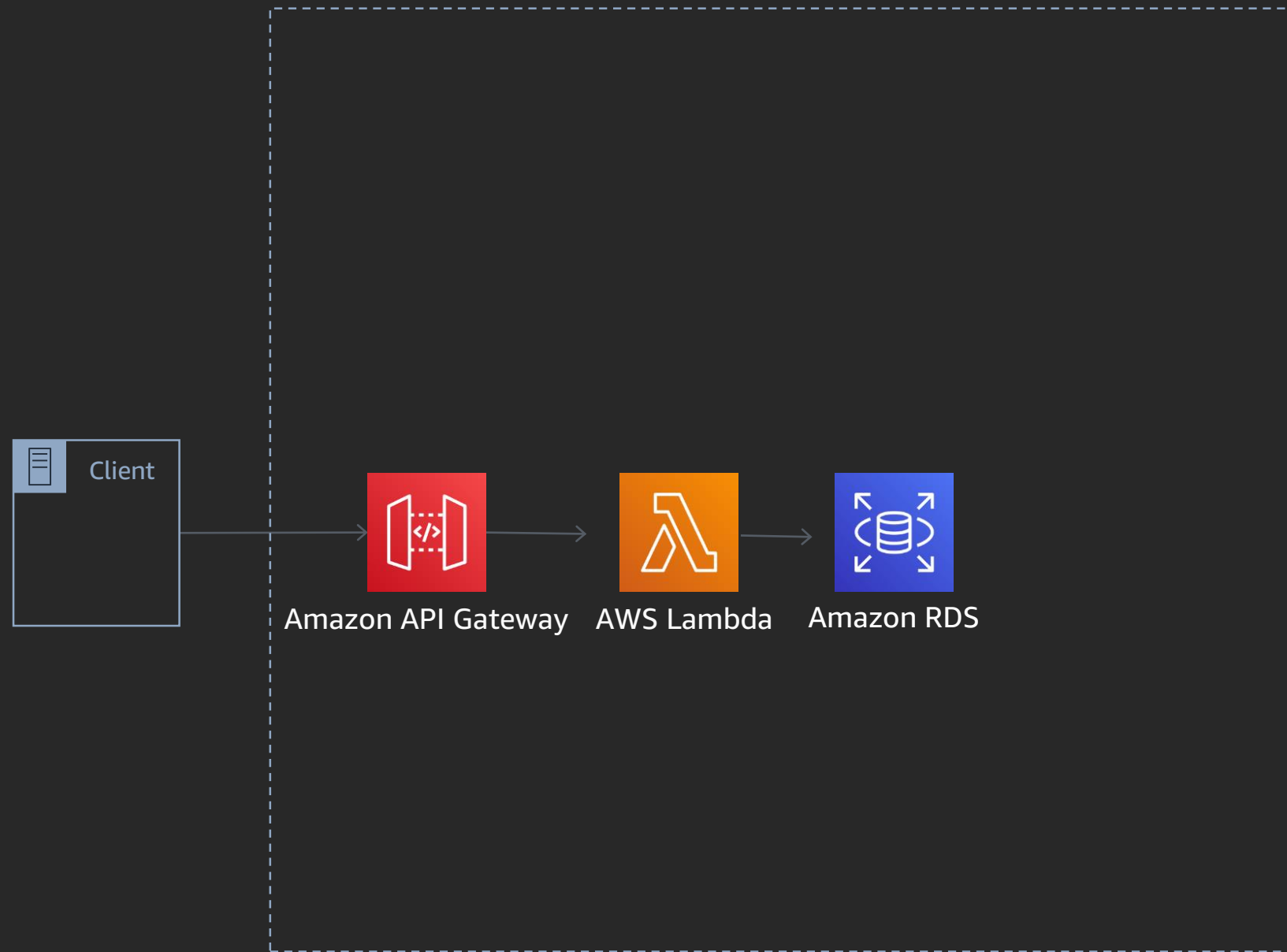
COST

# Practical example: Serverless Airline



# Pattern: Call me, “Maybe”

# Pattern: Call me, "Maybe" (Webhook)



OPERATIONS

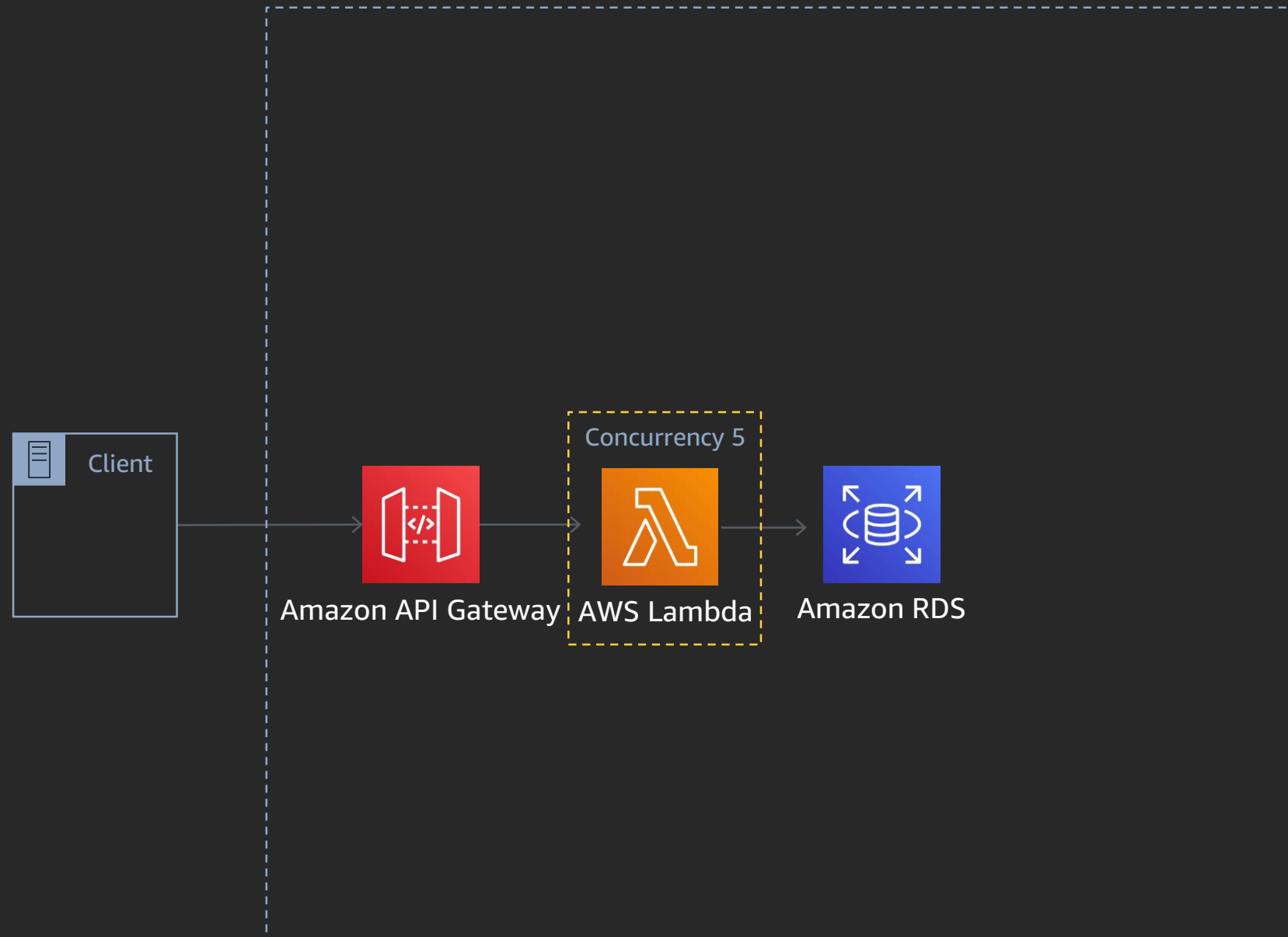
RELIABILITY

SECURITY

PERFORMANCE

COST

# Pattern: Call me, "Maybe" (Webhook)



## Best practices

Limit concurrency to protect non-scalable/stateful downstream services

OPERATIONS

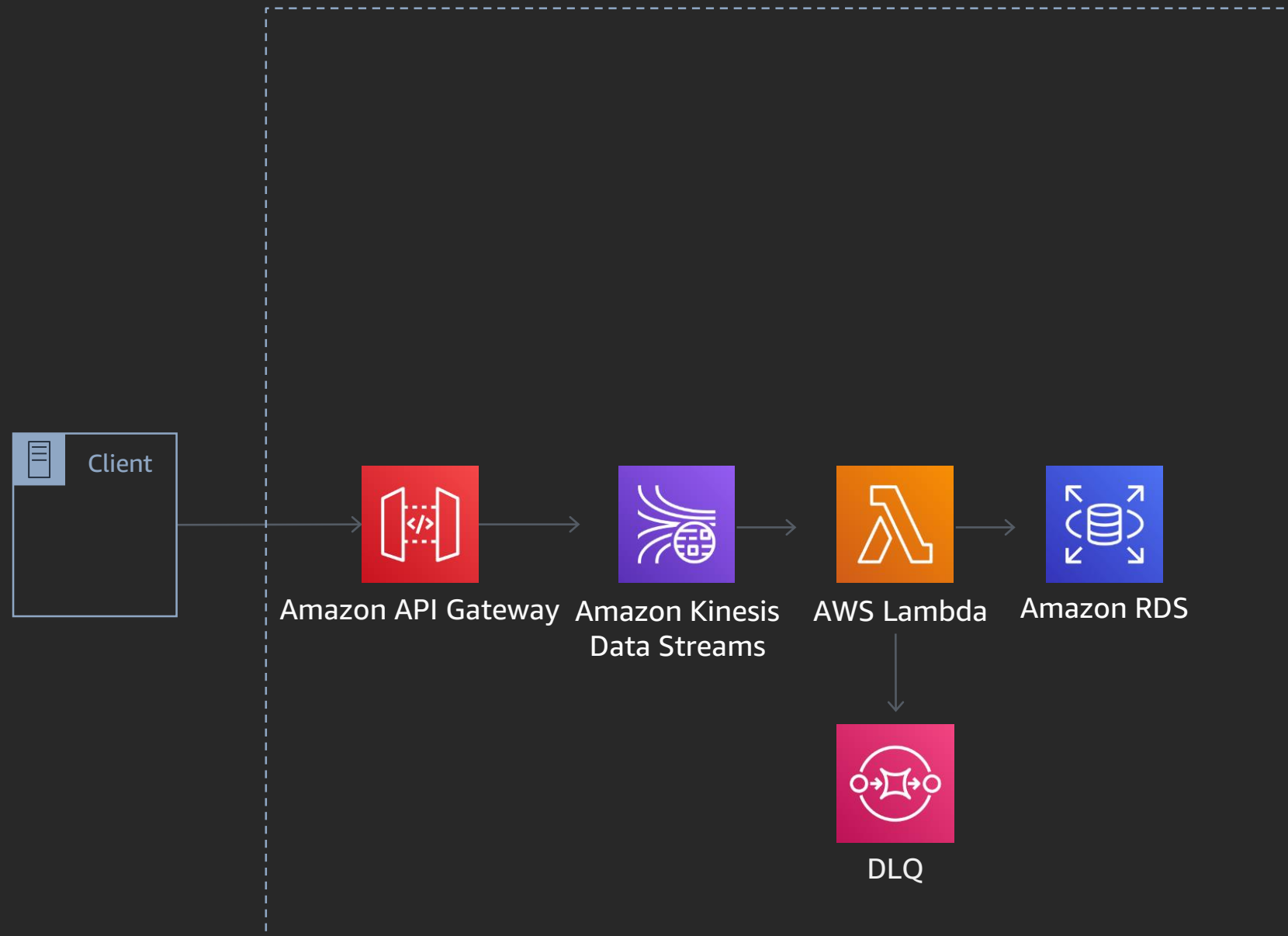
RELIABILITY

SECURITY

PERFORMANCE

COST

# Pattern: Call me, "Maybe" (Webhook)



## Best practices

- Limit concurrency to protect non-scalable/stateful downstream services

- Kinesis as a buffer + a better mechanism to limit concurrency

- Use Lambda Destinations for failed requests; set max retries new

OPERATIONS

RELIABILITY

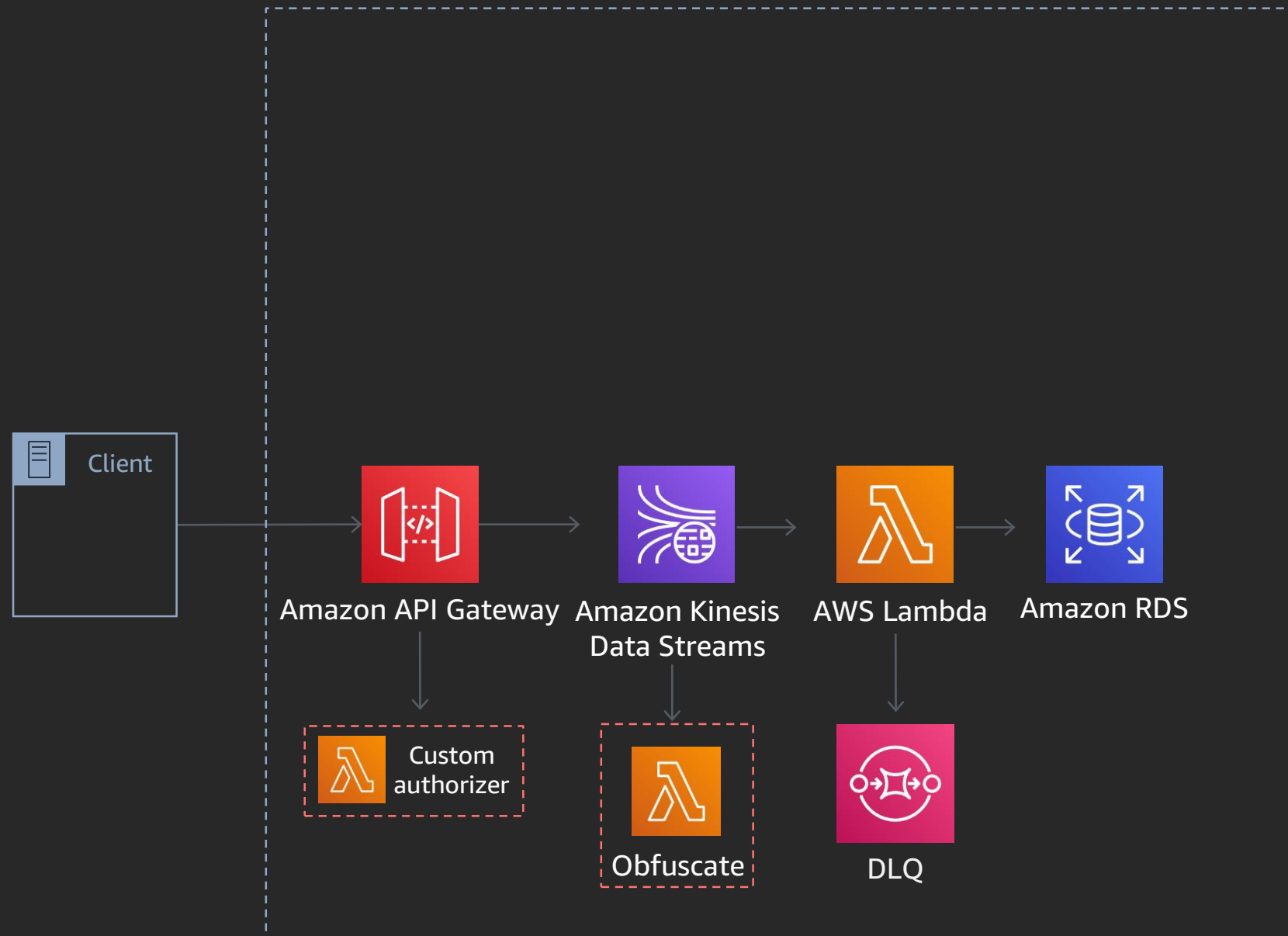
SECURITY

PERFORMANCE

COST



# Pattern: Call me, "Maybe" (Webhook)



## Best practices

- Limit concurrency to protect non-scalable/stateful downstream services

- Kinesis as a buffer + a better mechanism to limit concurrency

- Use Lambda Destinations for failed requests; set max retries new

- Enforce authorization and obfuscate sensitive data on the stream

OPERATIONS

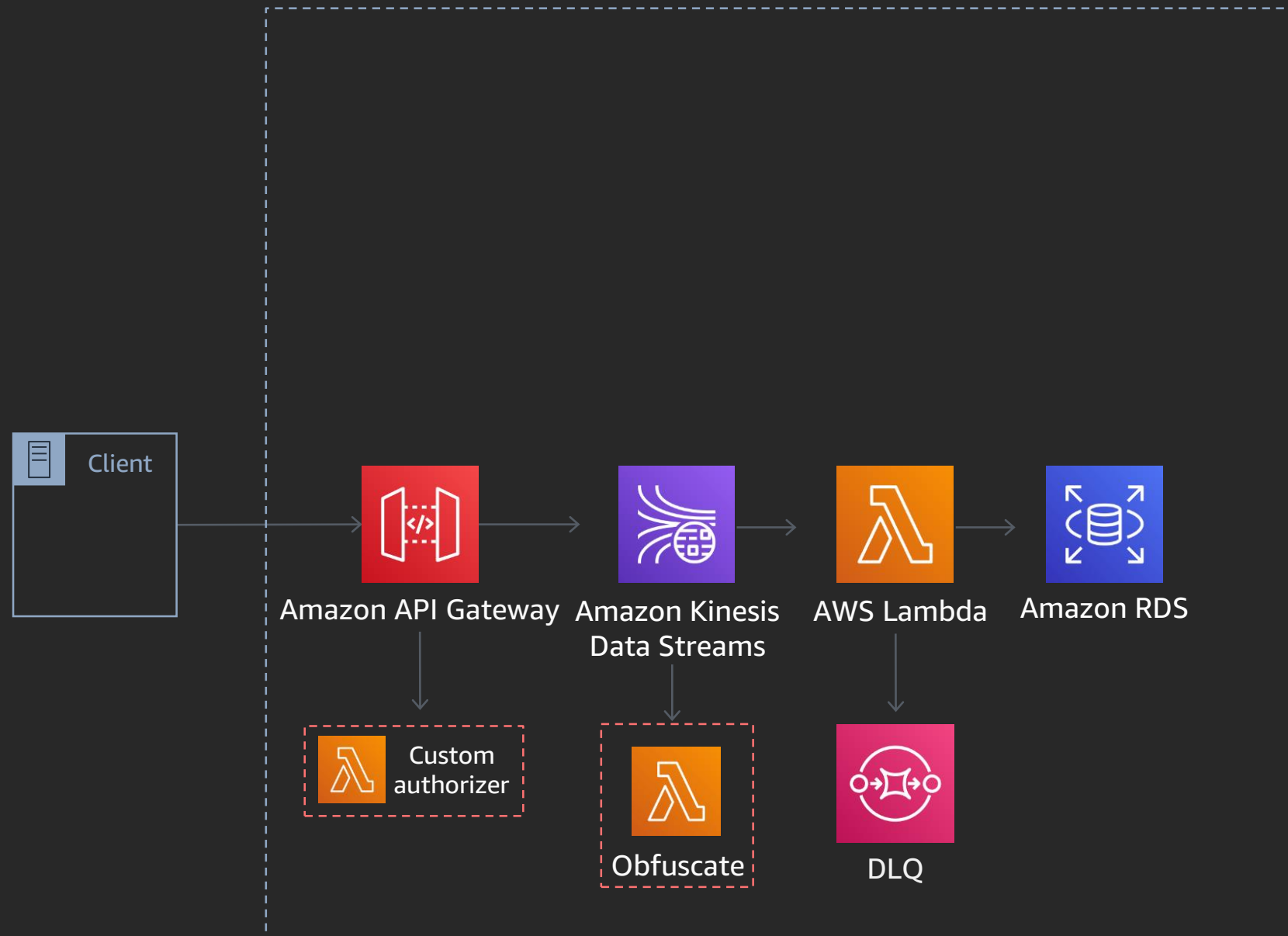
RELIABILITY

SECURITY

PERFORMANCE

COST

# Pattern: Call me, "Maybe" (Webhook)



## Best practices

- Limit concurrency to protect non-scalable/stateful downstream services

- Kinesis as a buffer + a better mechanism to limit concurrency

- Use Lambda Destinations for failed requests; set max retries new

- Enforce authorization and obfuscate sensitive data on the stream

- For low-volume traffic, Kinesis can batch records for up to 5 minutes

OPERATIONS

RELIABILITY

SECURITY

PERFORMANCE

COST

# Pattern: Call me, "Maybe" (Webhook)



## Best practices

- Limit concurrency to protect non-scalable/stateful downstream services

- Kinesis as a buffer + a better mechanism to limit concurrency

- Use Lambda Destinations for failed requests; set max retries new

- Enforce authorization and obfuscate sensitive data on the stream

- For low-volume traffic, Kinesis can batch records for up to 5 minutes

- Alternatively, DynamoDB+SQS to easily scale webhooks

OPERATIONS

RELIABILITY

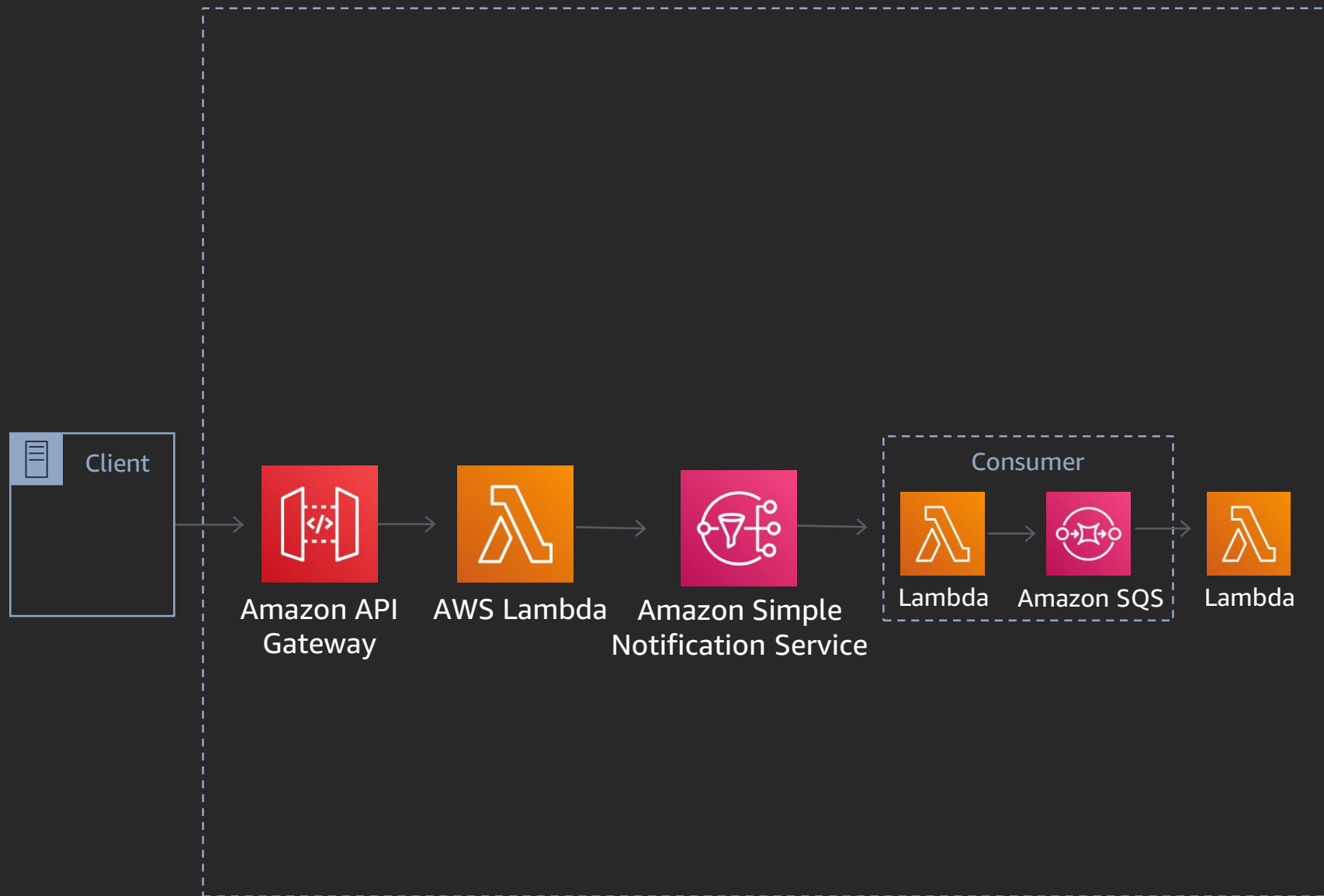
SECURITY

PERFORMANCE

COST

# Pattern: The big “Fan”

# Pattern: The big "Fan" (fan-out)



OPERATIONS

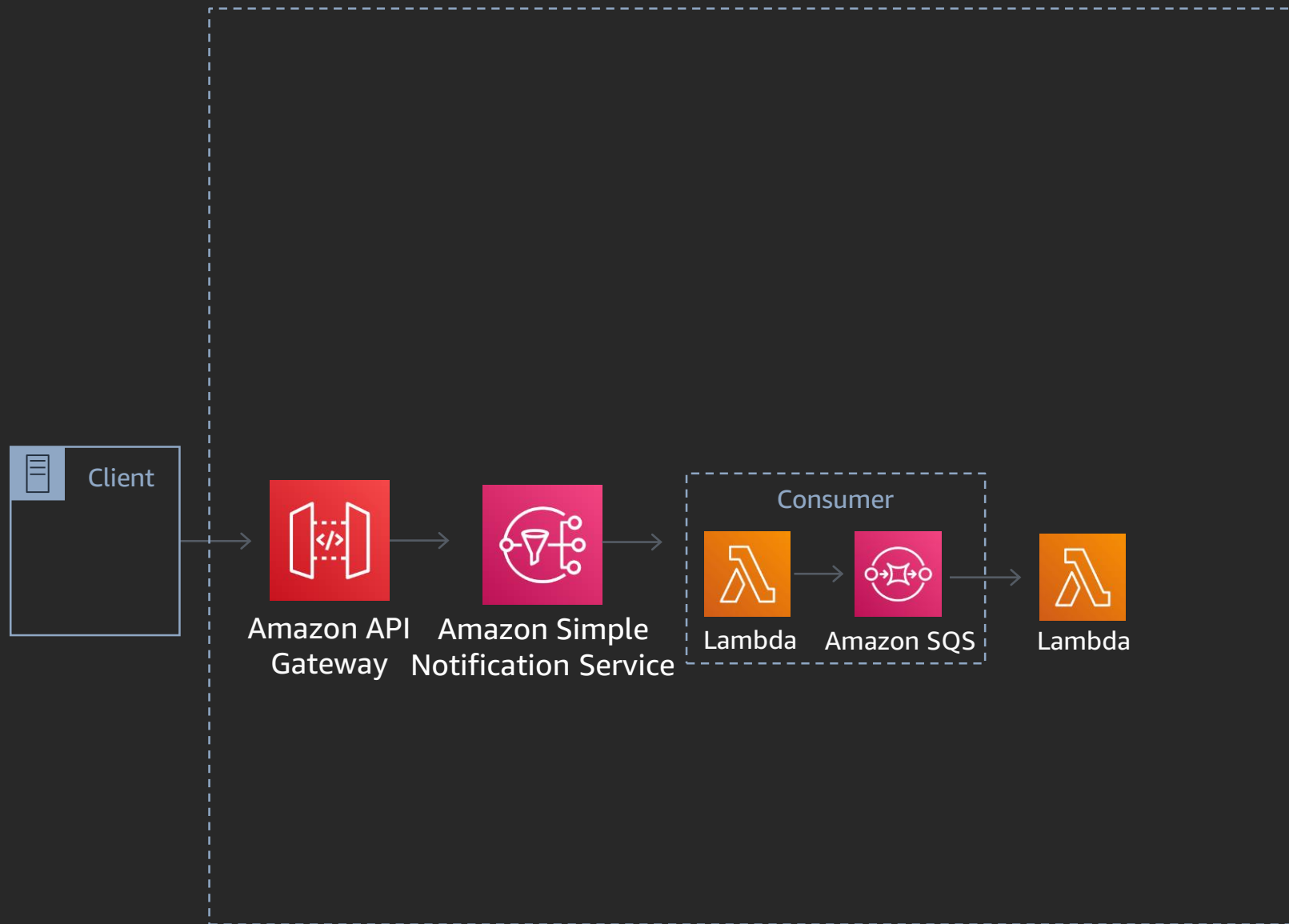
RELIABILITY

SECURITY

PERFORMANCE

COST

# Pattern: The big "Fan" (fan-out)



## Best practices

- API Gateway can integrate with AWS services directly

OPERATIONS

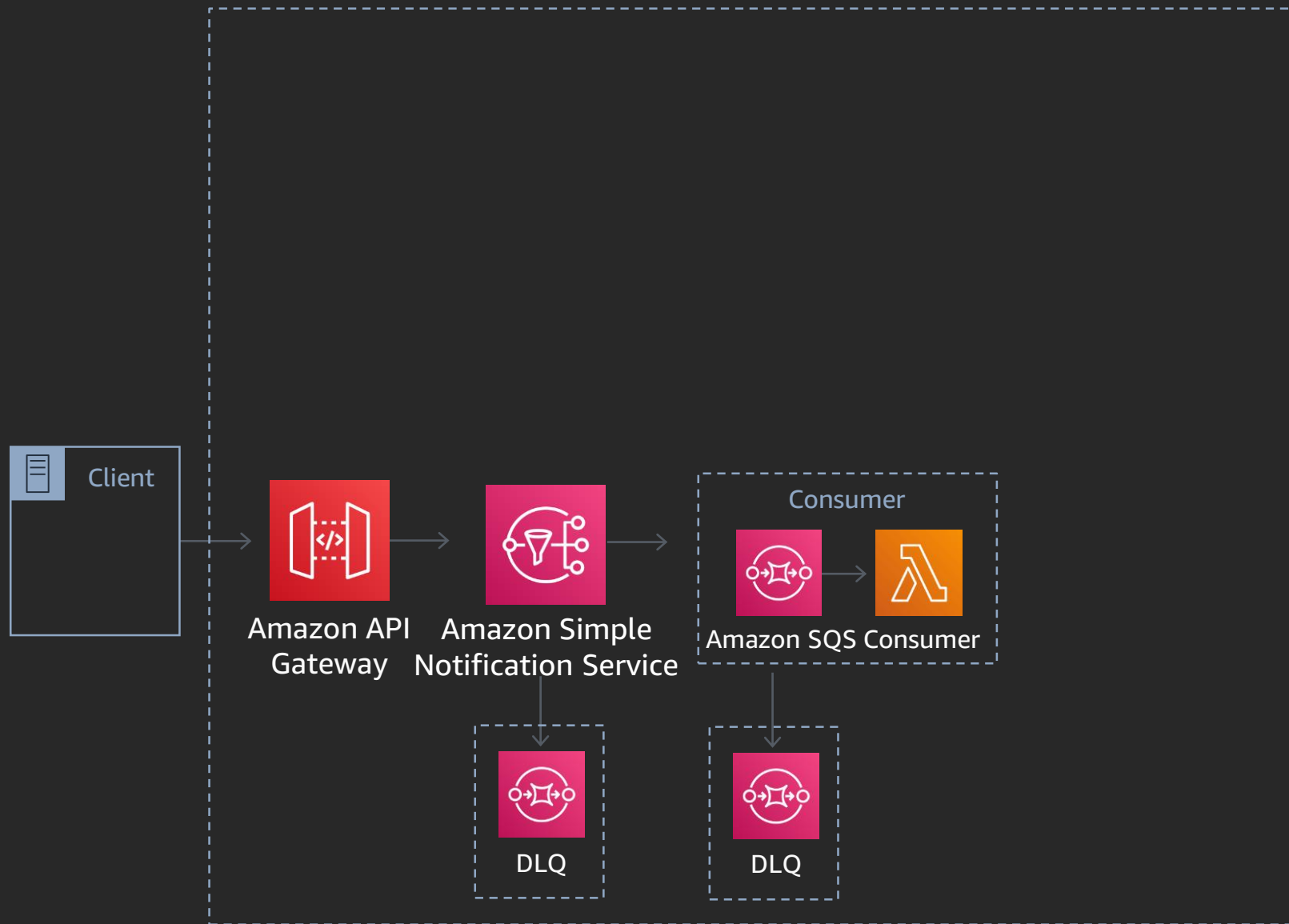
RELIABILITY

SECURITY

PERFORMANCE

COST

# Pattern: The big "Fan" (fan-out)



## Best practices

- API Gateway can integrate with AWS services directly
- Integrate with Amazon SQS for higher durability, batching, and DLQ new

OPERATIONS

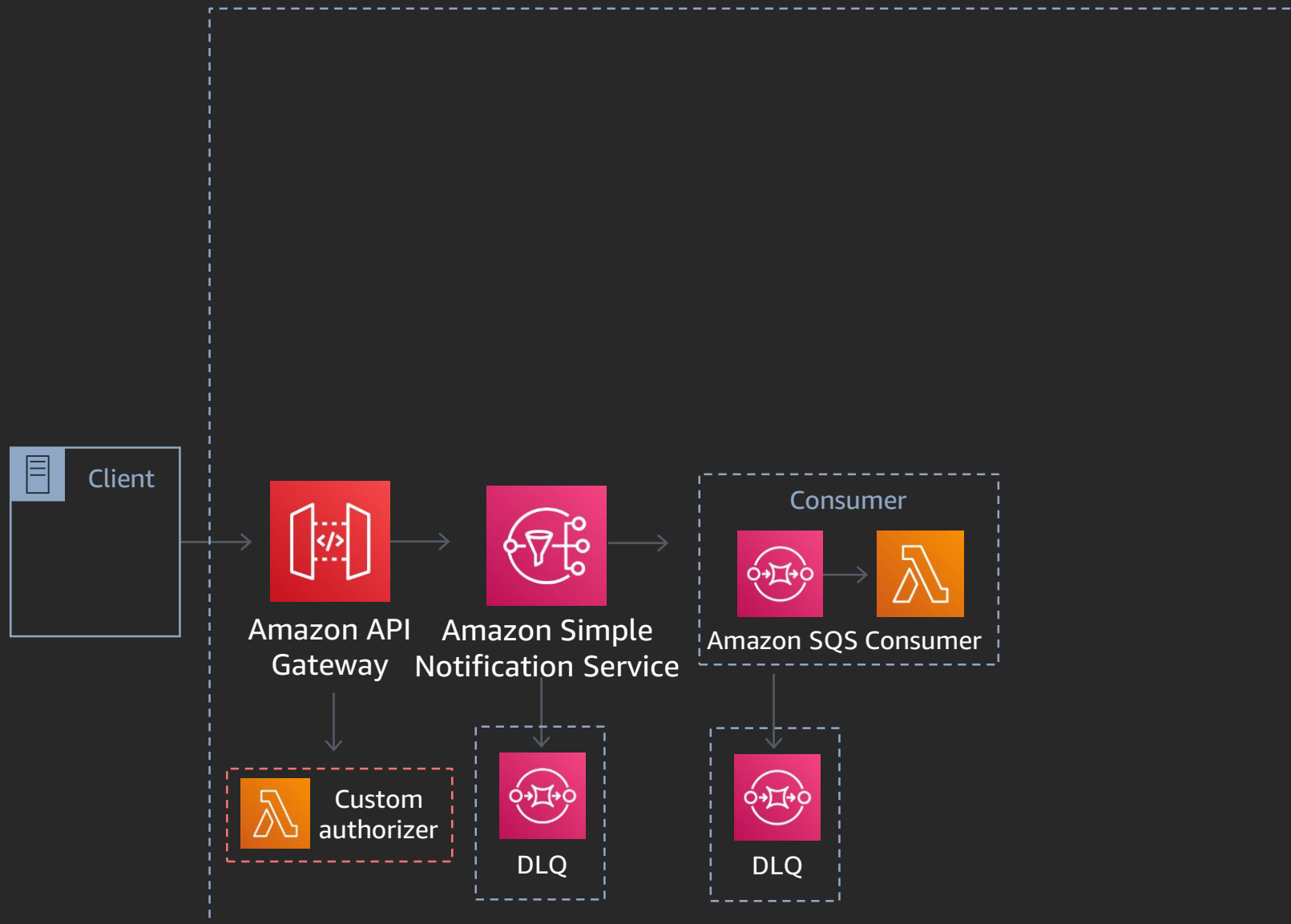
RELIABILITY

SECURITY

PERFORMANCE

COST

# Pattern: The big “Fan” (fan-out)



## Best practices

- API Gateway can integrate with AWS services directly
- Integrate with Amazon SQS for higher durability, batching, and DLQ new
- Enforce authorization. Verify signature of Amazon SNS messages

OPERATIONS

RELIABILITY

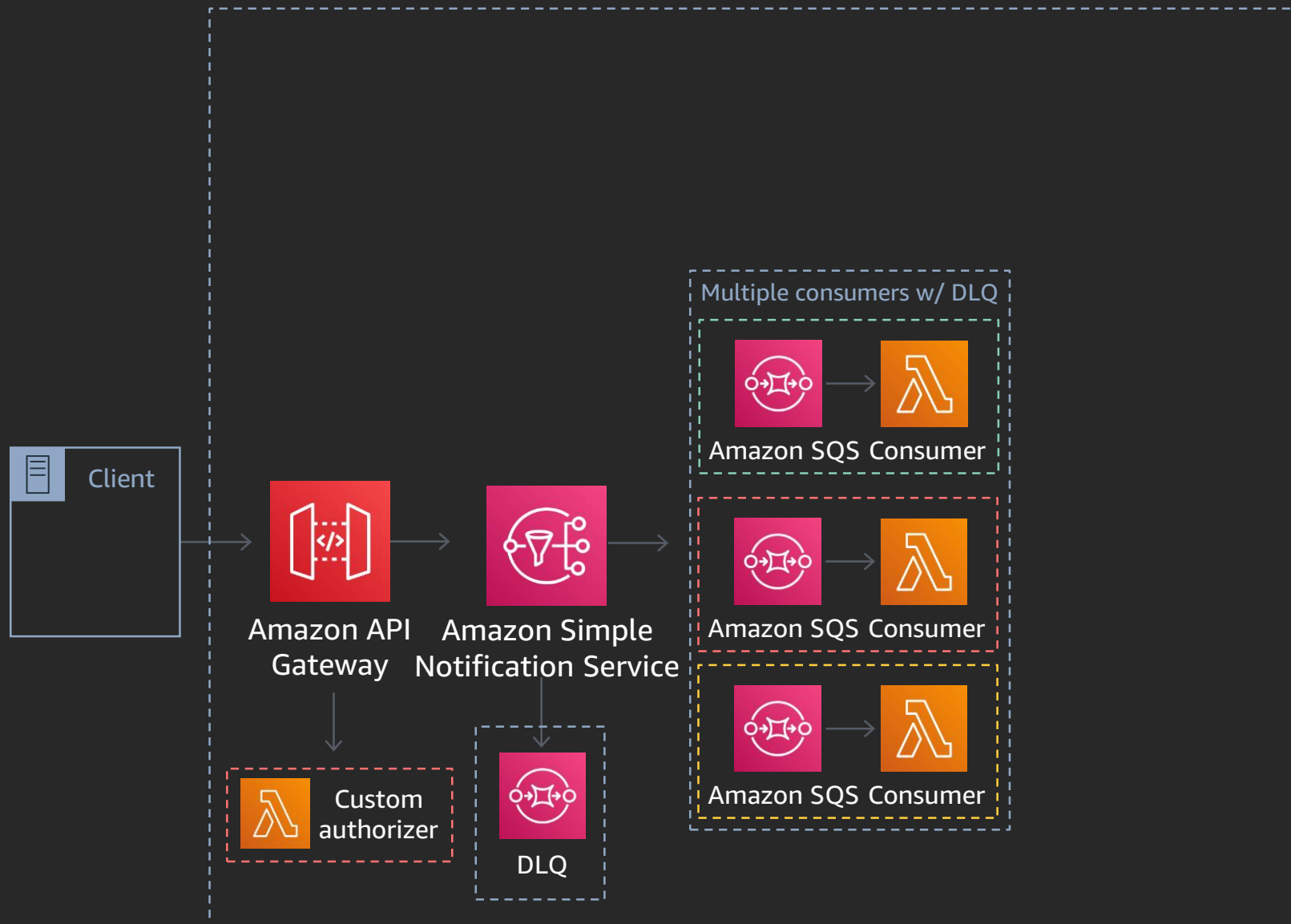
SECURITY

PERFORMANCE

COST



# Pattern: The big "Fan" (fan-out)



## Best practices

- API Gateway can integrate with AWS services directly
- Integrate with Amazon SQS for higher durability, batching, and DLQ new
- Enforce authorization. Verify signature of Amazon SNS messages

OPERATIONS

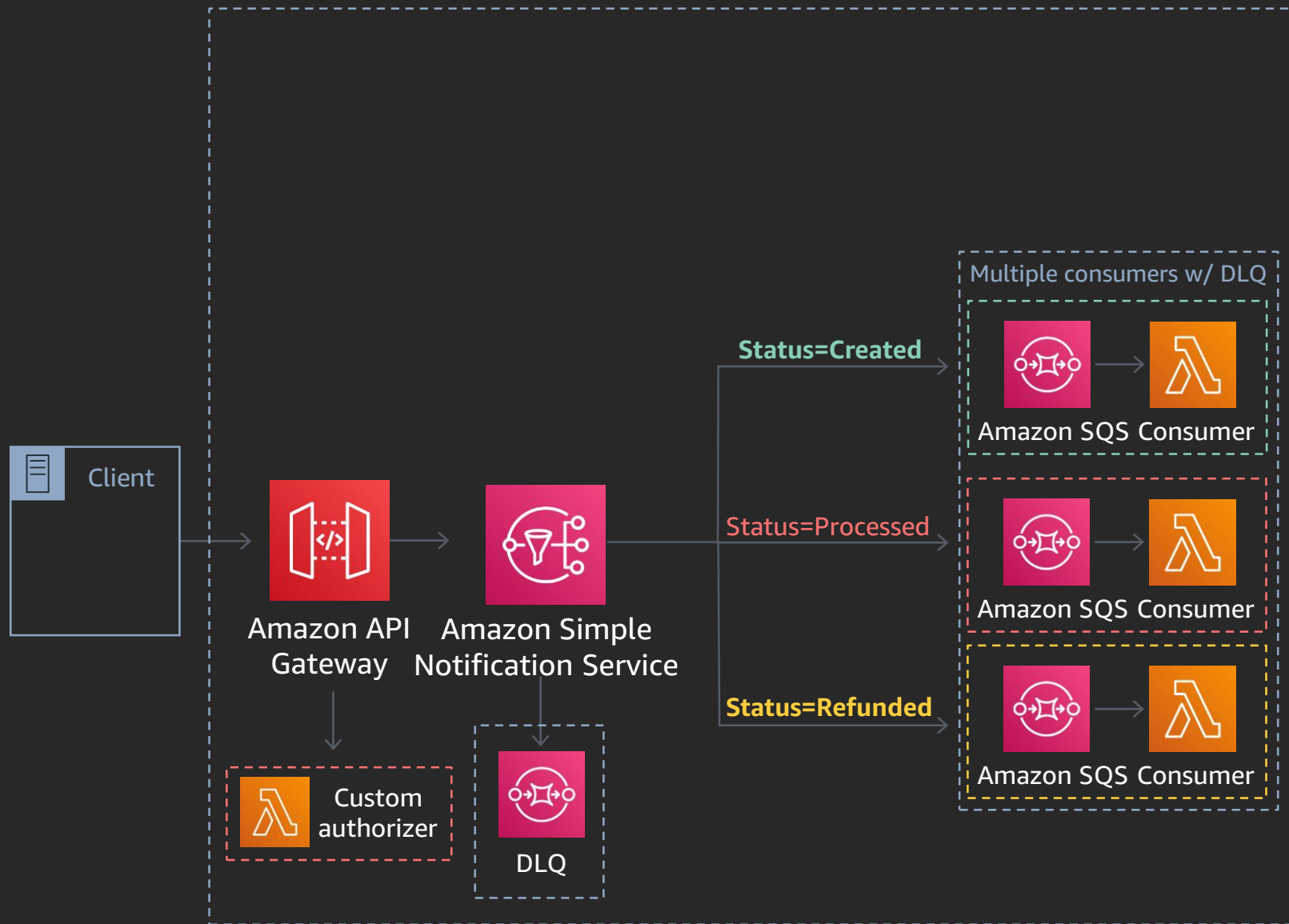
RELIABILITY

SECURITY

PERFORMANCE

COST

# Pattern: The big “Fan” (fan-out)



## Best practices

- API Gateway can integrate with AWS services directly
- Integrate with Amazon SQS for higher durability, batching, and DLQ new
- Enforce authorization. Verify signature of Amazon SNS messages
- Use message filtering for efficient processing

OPERATIONS

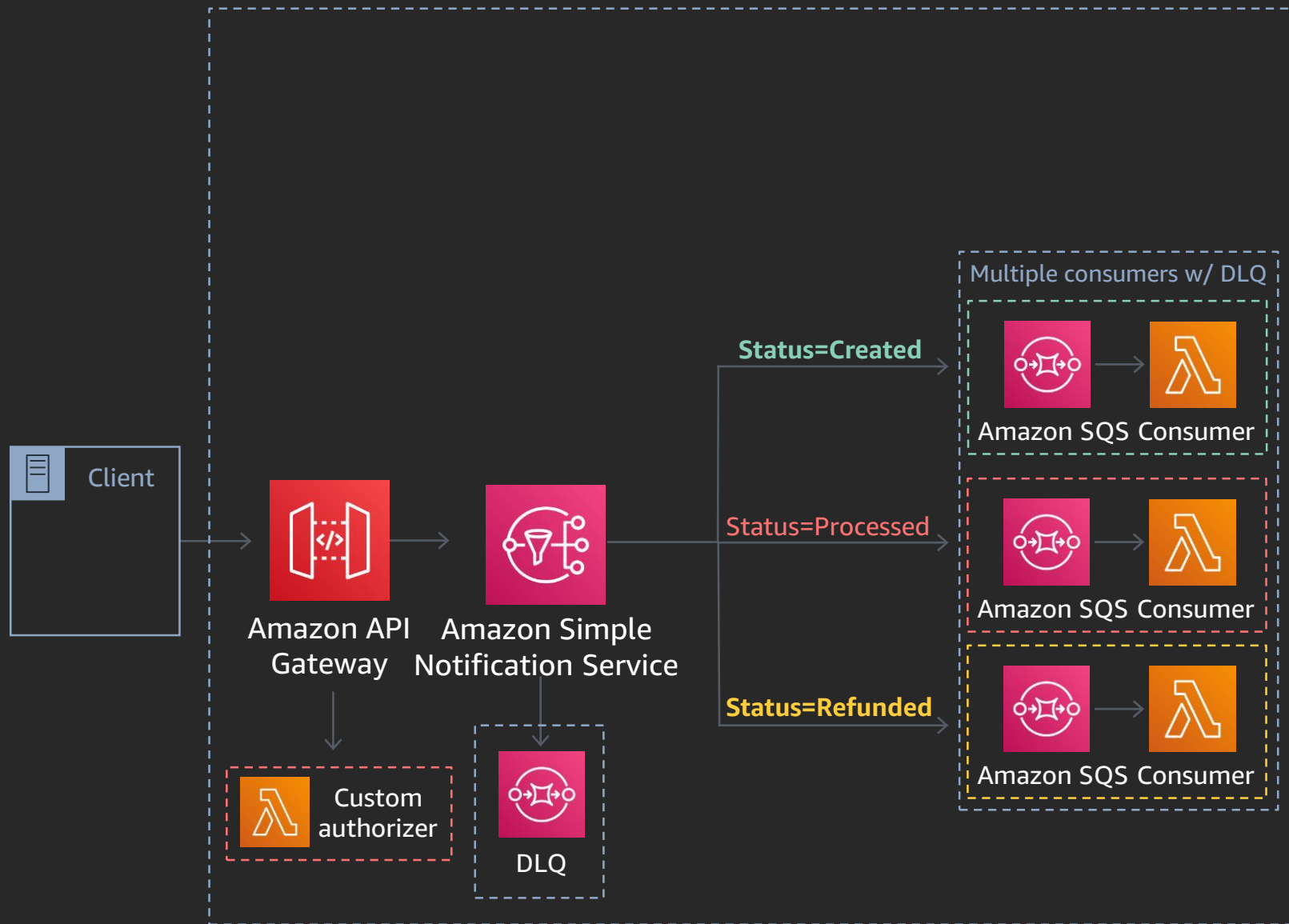
RELIABILITY

SECURITY

PERFORMANCE

COST

# Pattern: The big "Fan" (fan-out)



## Best practices

- API Gateway can integrate with AWS services directly
- Integrate with Amazon SQS for higher durability, batching, and DLQ new
- Enforce authorization. Verify signature of Amazon SNS messages
- Use message filtering for efficient processing
- Compress and aggregate messages when possible

OPERATIONS

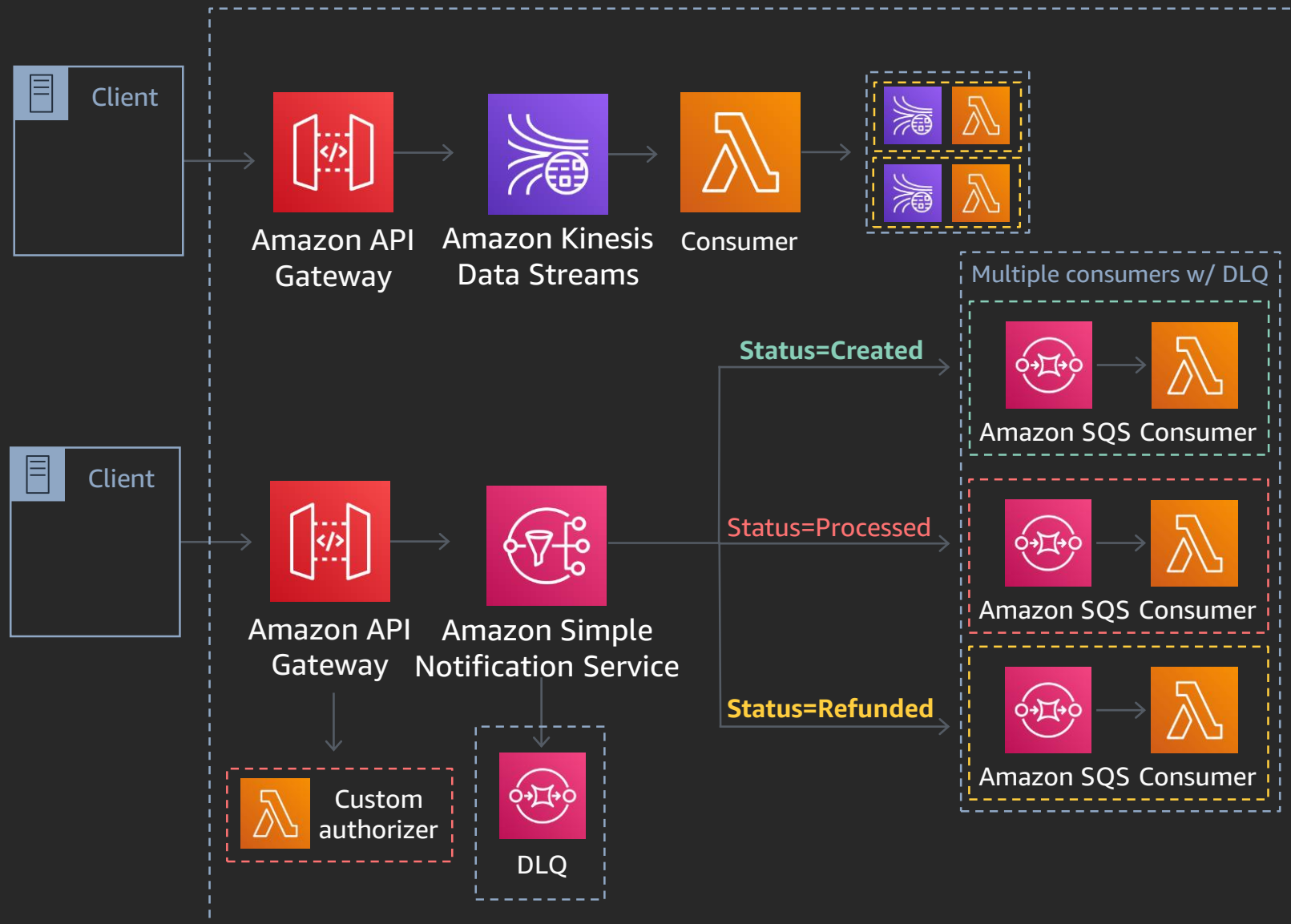
RELIABILITY

SECURITY

PERFORMANCE

COST

# Pattern: The big "Fan" (fan-out)



## Best practices

- API Gateway can integrate with AWS services directly
- Integrate with Amazon SQS for higher durability, batching, and DLQ new
- Enforce authorization. Verify signature of Amazon SNS messages
- Use message filtering for efficient processing
- Compress and aggregate messages when possible
- Consider Kinesis for larger payloads

OPERATIONS

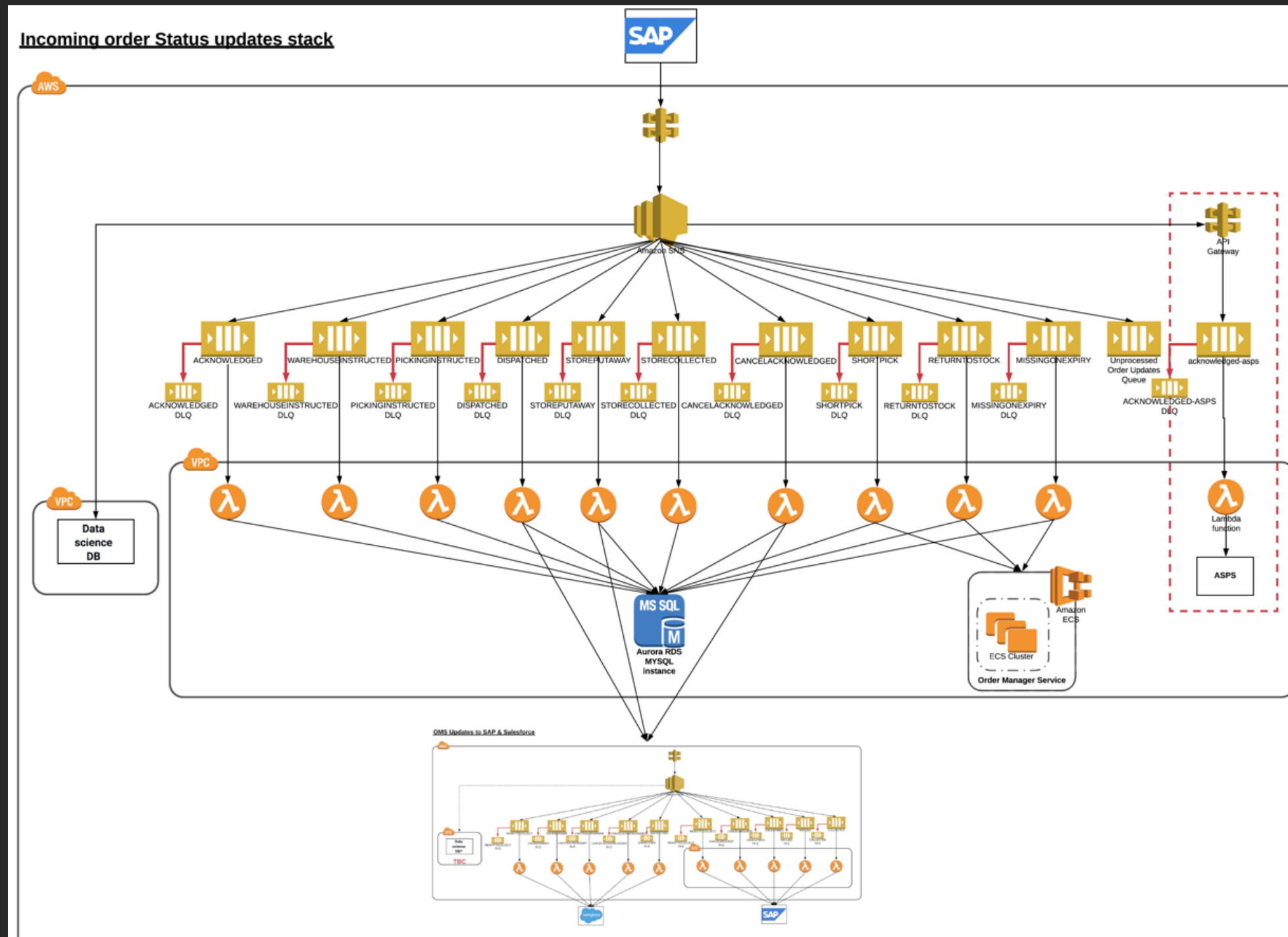
RELIABILITY

SECURITY

PERFORMANCE

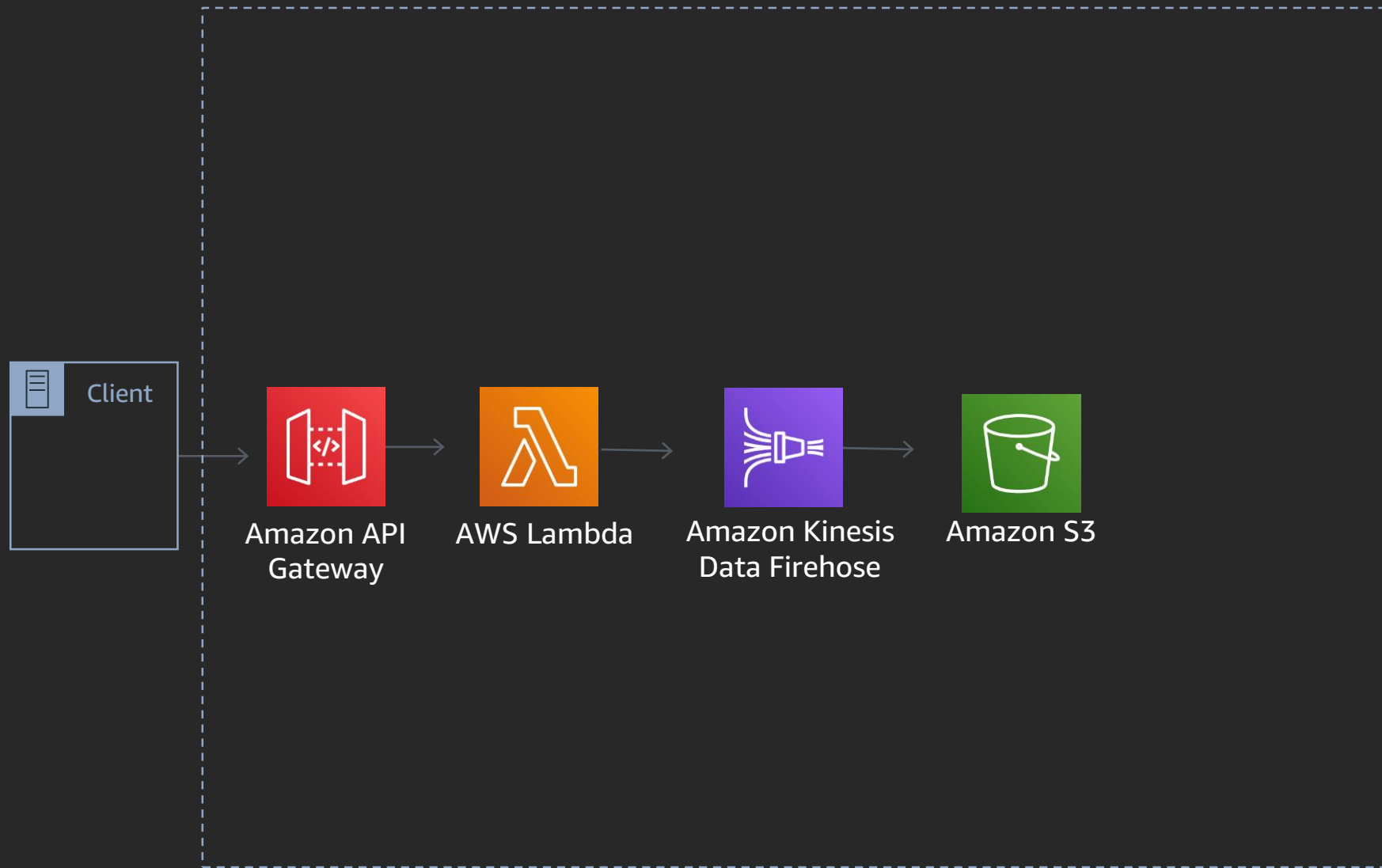
COST

# Practical example: Dunelm



# Pattern: They say “I’m a Streamer”

# Pattern: They say "I'm a Streamer" (streaming)



OPERATIONS

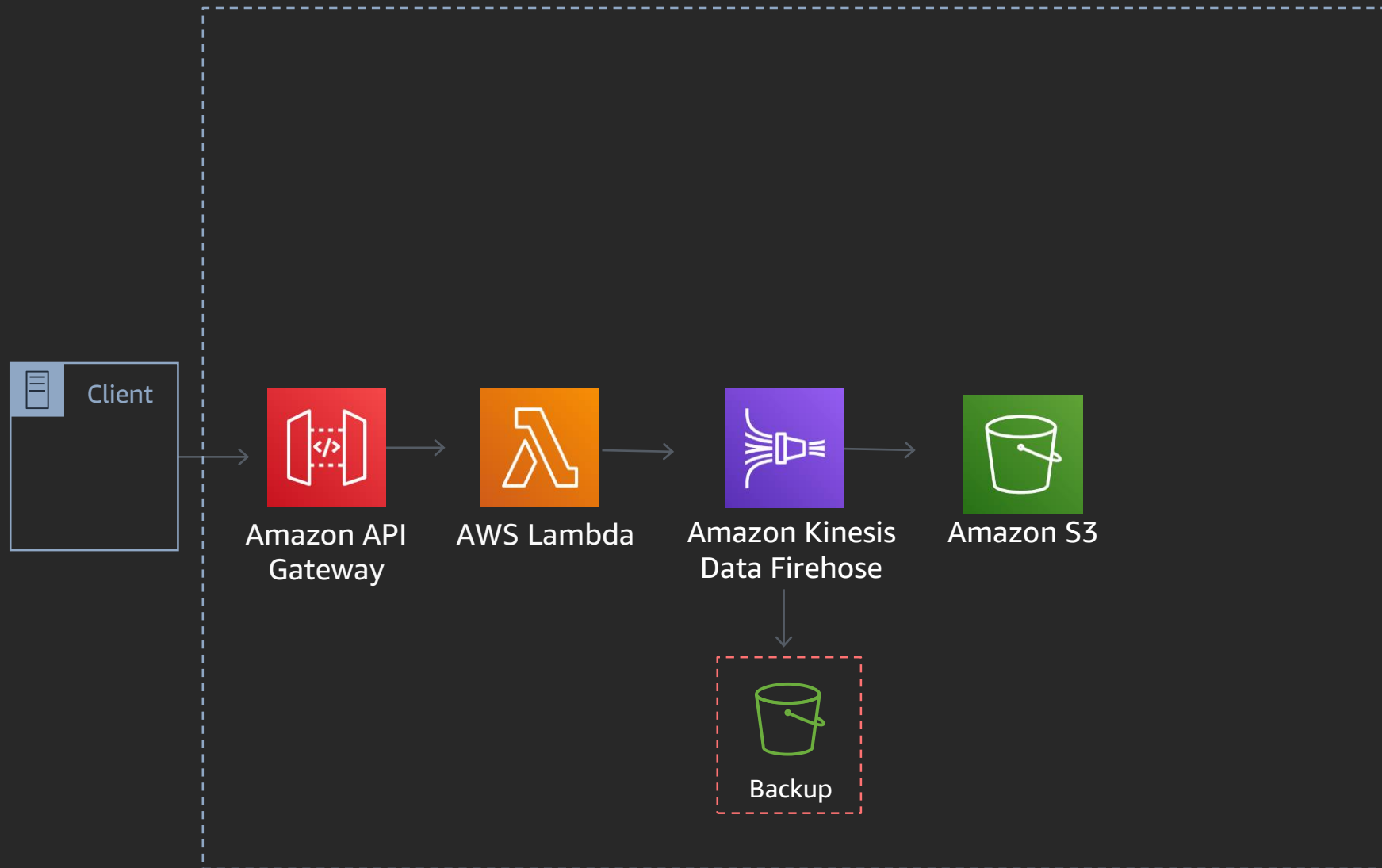
RELIABILITY

SECURITY

PERFORMANCE

COST

# Pattern: They say “I’m a Streamer” (streaming)



## Best practices

- Enable source stream record backup

OPERATIONS

RELIABILITY

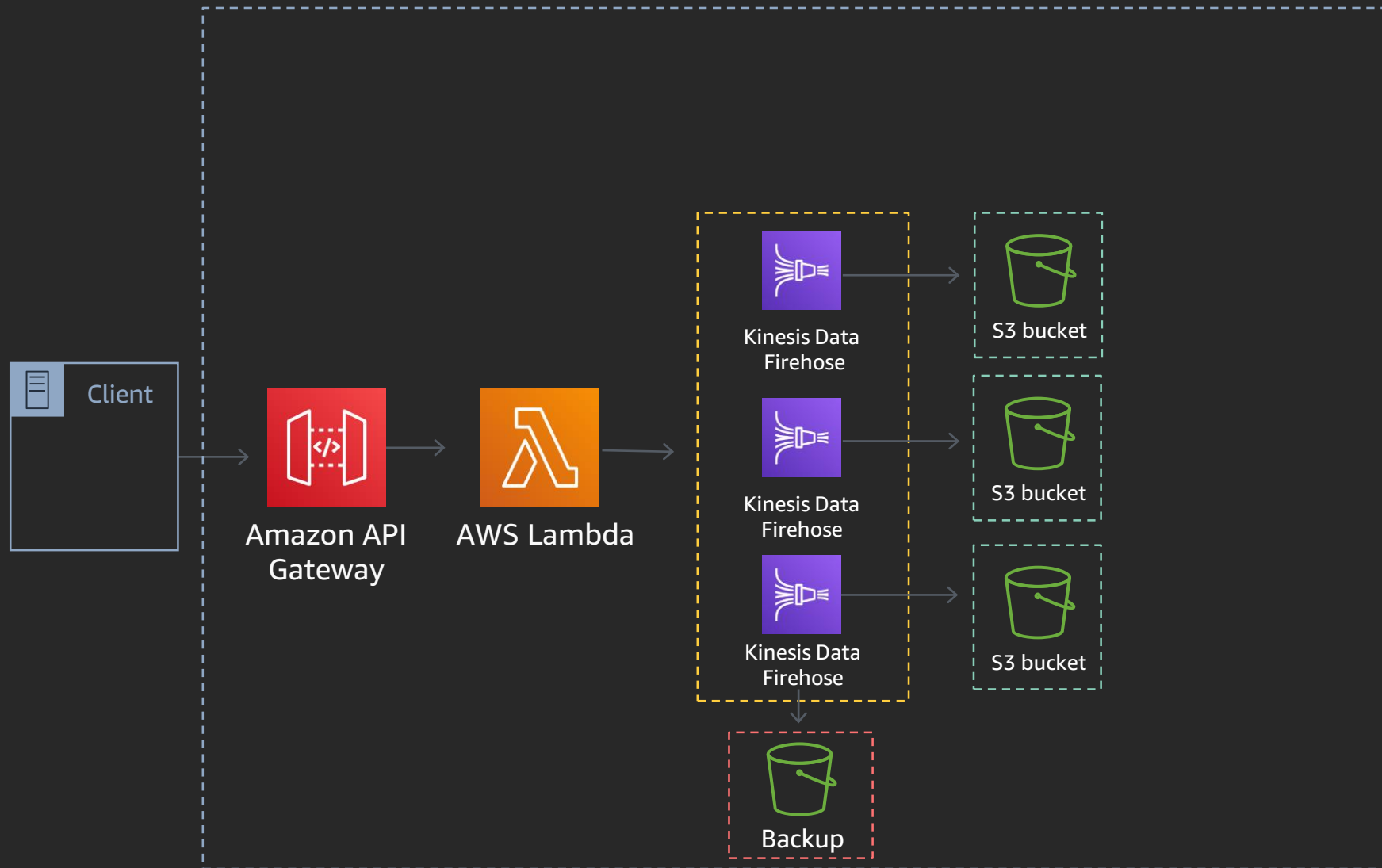
SECURITY

PERFORMANCE

COST



# Pattern: They say “I’m a Streamer” (streaming)



## Best practices

- Enable source stream record backup
- Favor dedicated Data Firehose per context/domain

OPERATIONS

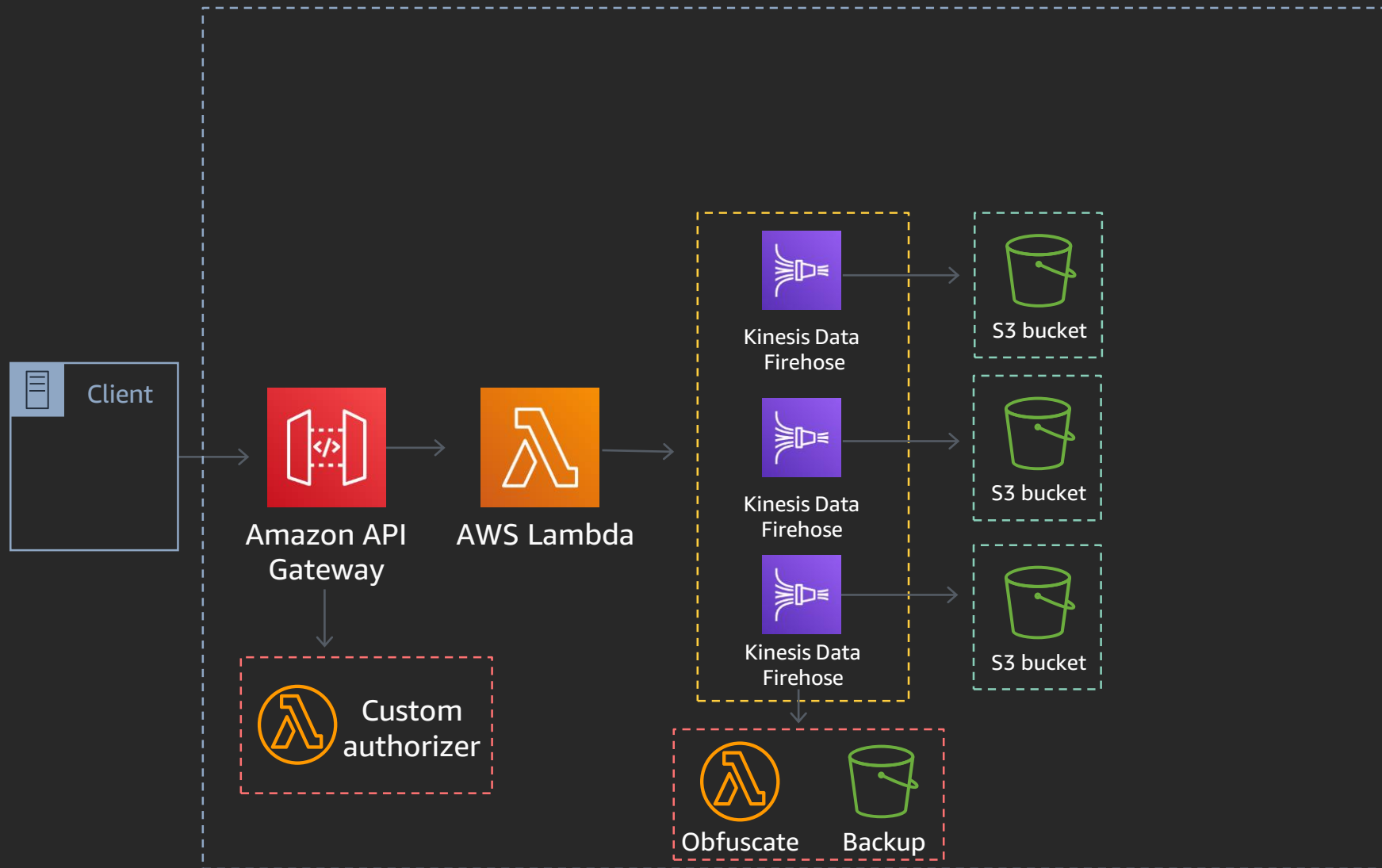
RELIABILITY

SECURITY

PERFORMANCE

COST

# Pattern: They say "I'm a Streamer" (streaming)



## Best practices

- Enable source stream record backup
- Favor dedicated Data Firehose per context/domain
- Enforce authorization
- Obfuscate/remove sensitive stream data

OPERATIONS

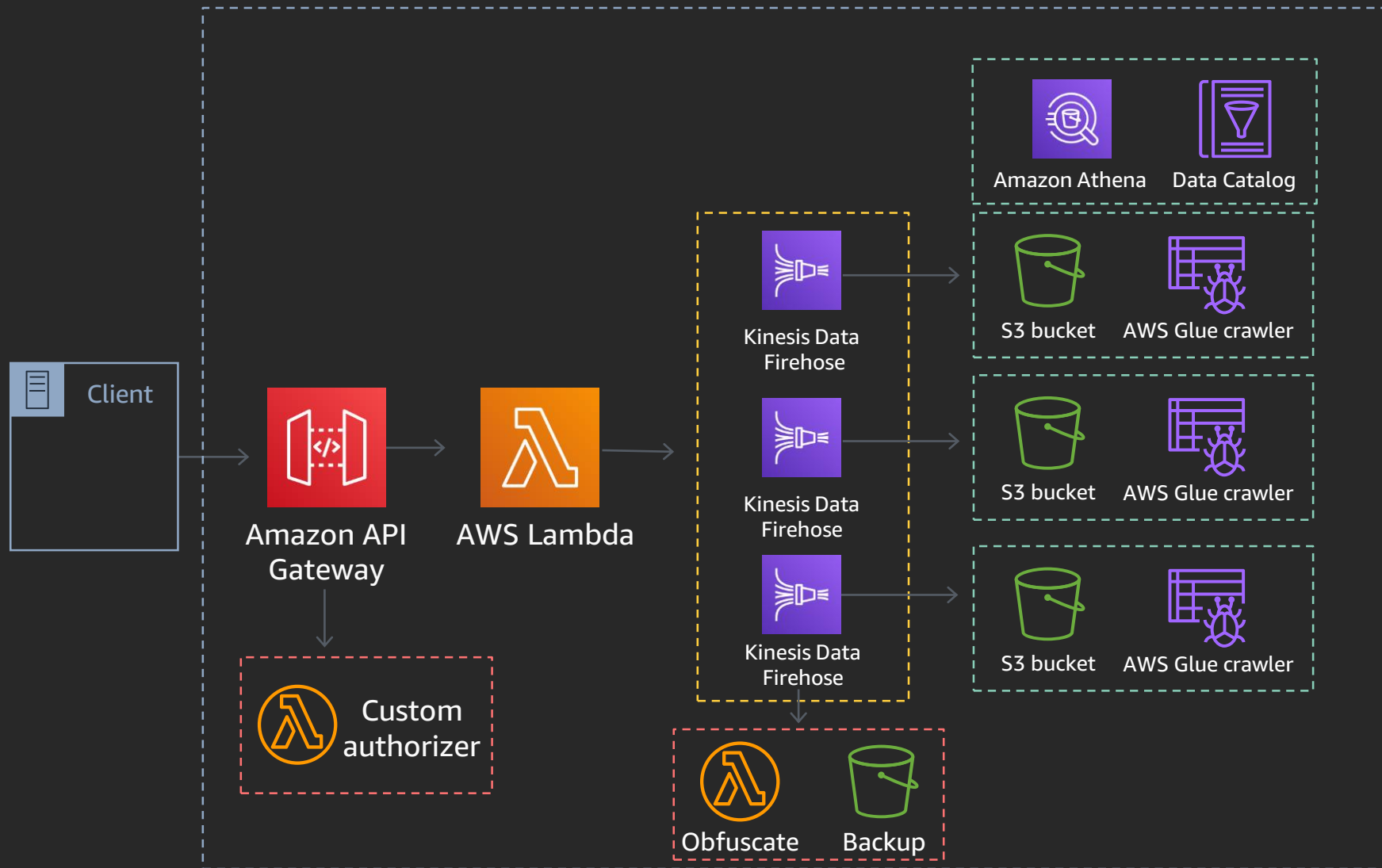
RELIABILITY

SECURITY

PERFORMANCE

COST

# Pattern: They say "I'm a Streamer" (streaming)



## Best practices

- Enable source stream record backup
- Favor dedicated Data Firehose per context/domain
- Enforce authorization
- Obfuscate/remove sensitive stream data
- Enable Parquet transformation. Use Glue to discover data schema and Athena to query

OPERATIONS

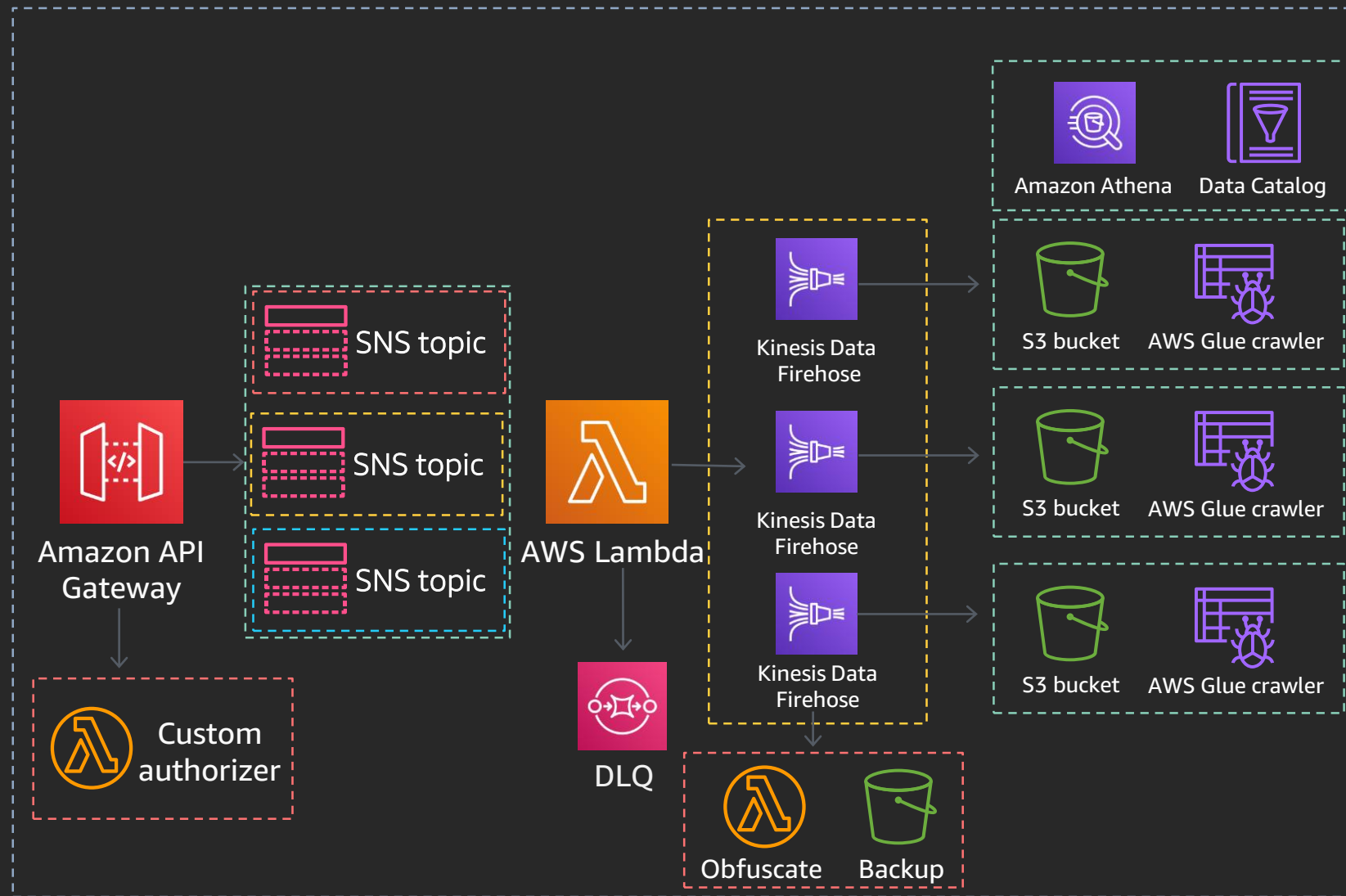
RELIABILITY

SECURITY

PERFORMANCE

COST

# Pattern: They say “I’m a Streamer” (streaming)



## Best practices

- Enable source stream record backup
- Favor dedicated Data Firehose per context/domain
- Enforce authorization
- Obfuscate/remove sensitive stream data
- Enable Parquet transformation. Use Glue to discover data schema and Athena to query
- Use message filtering to prevent unwanted events. Tune buffer/compression

OPERATIONS

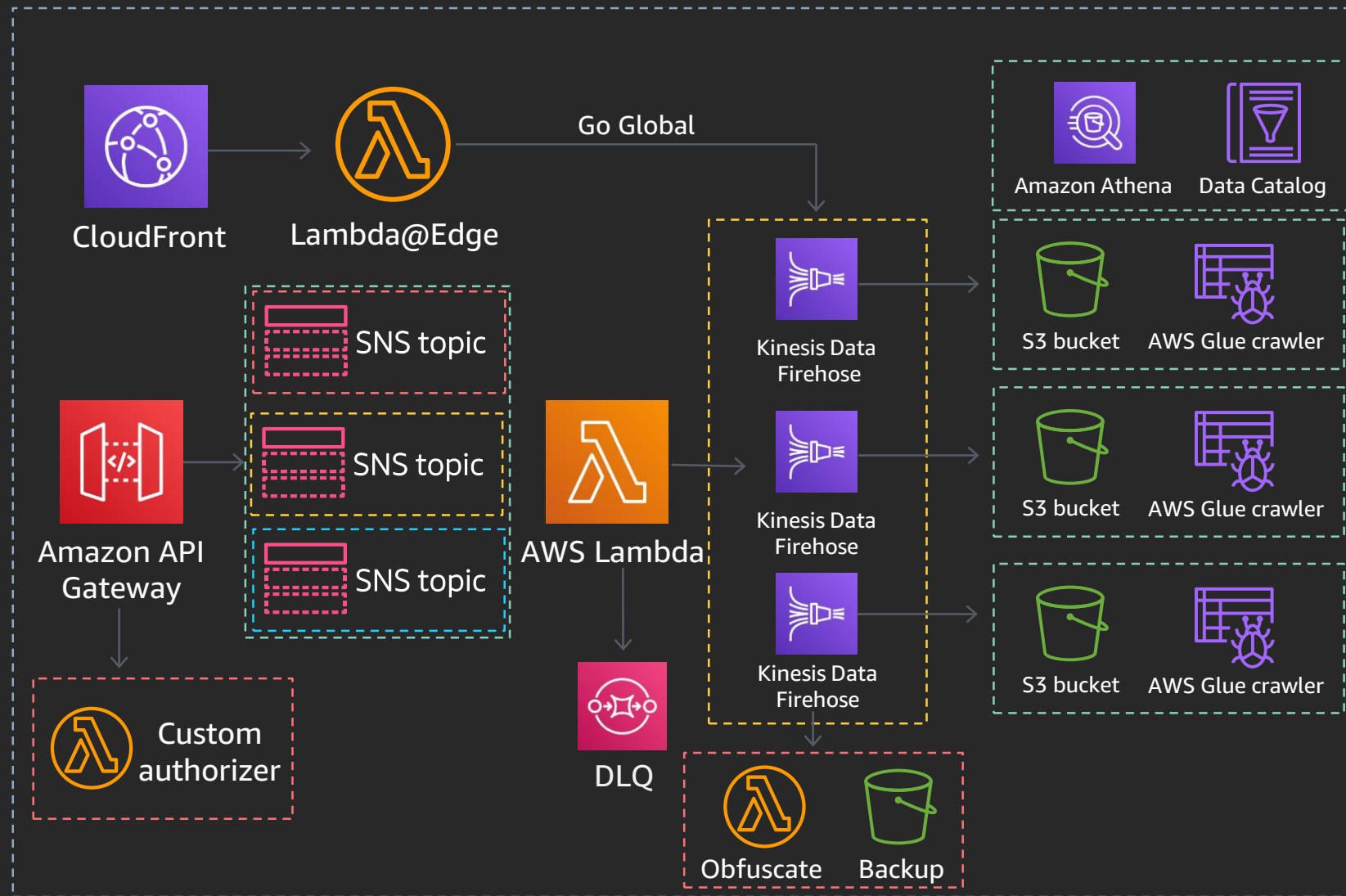
RELIABILITY

SECURITY

PERFORMANCE

COST

# Pattern: They say “I’m a Streamer” (streaming)



## Best practices

- Enable source stream record backup
- Favor dedicated Data Firehose per context/domain
- Enforce authorization
- Obfuscate/remove sensitive stream data
- Enable Parquet transformation. Use Glue to discover data schema and Athena to query
- Use message filtering to prevent unwanted events. Tune buffer/compression

OPERATIONS

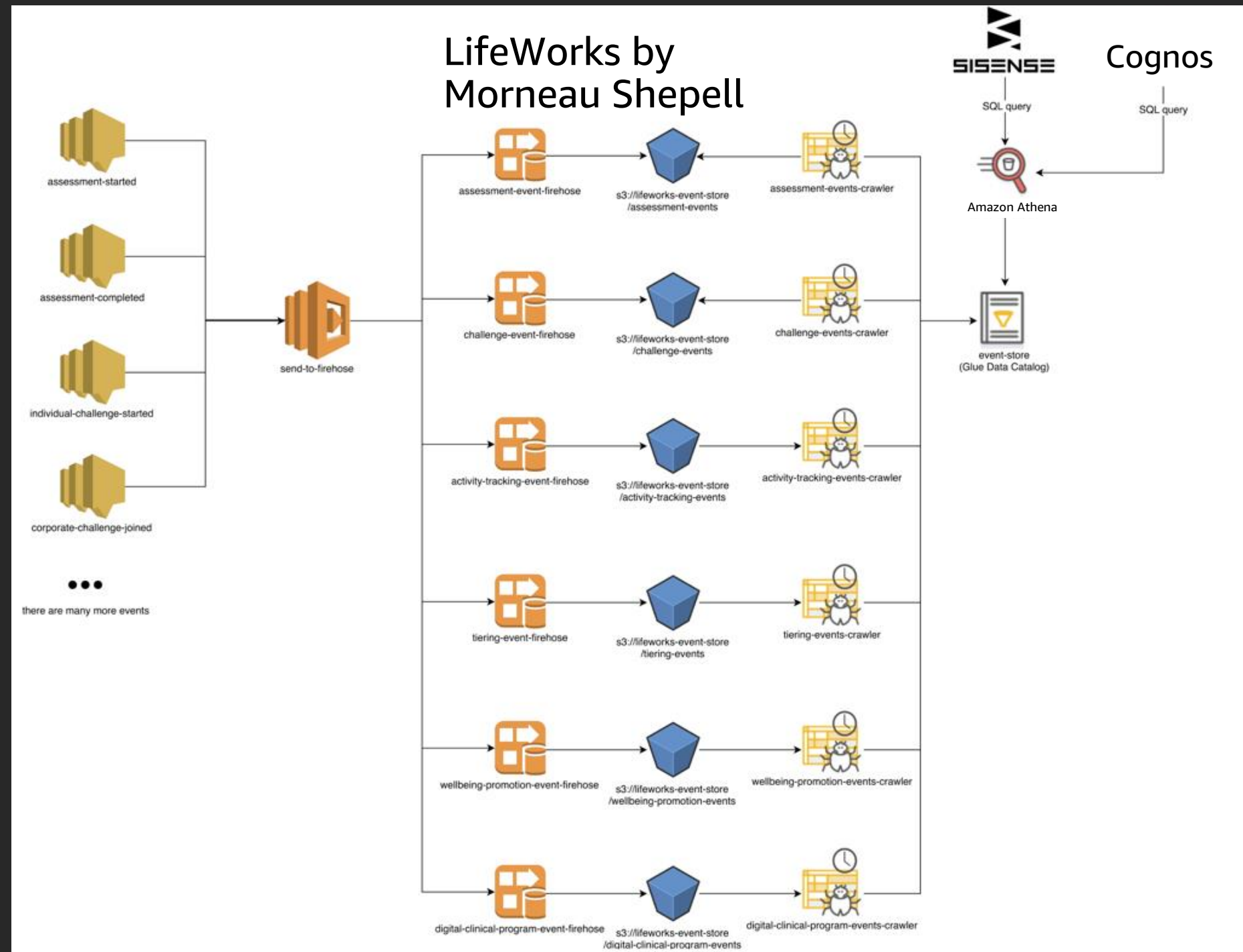
RELIABILITY

SECURITY

PERFORMANCE

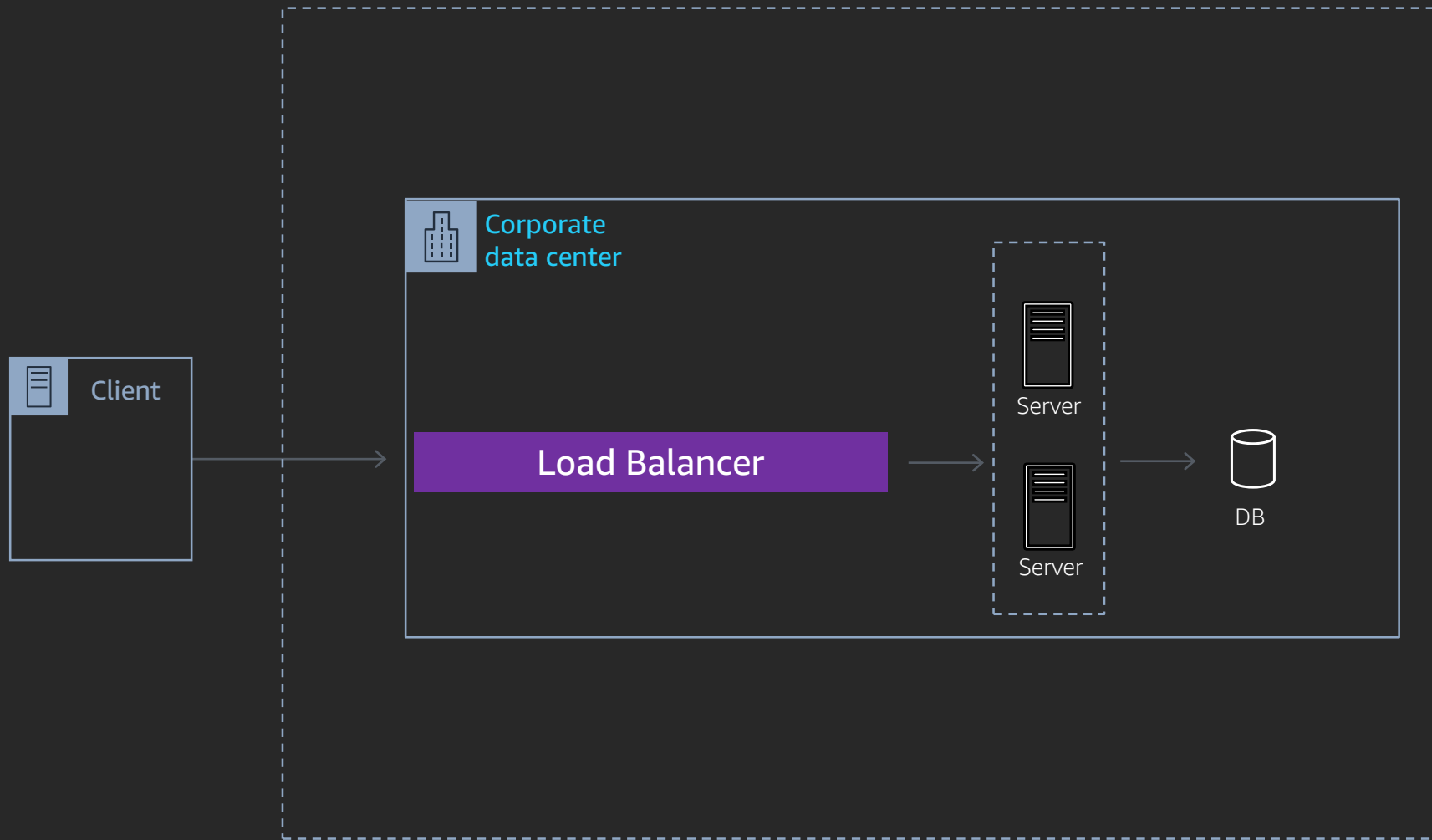
COST

# Practical example: LifeWorks



# Pattern: The “Strangler”

# Pattern: The “Strangler”



OPERATIONS

RELIABILITY

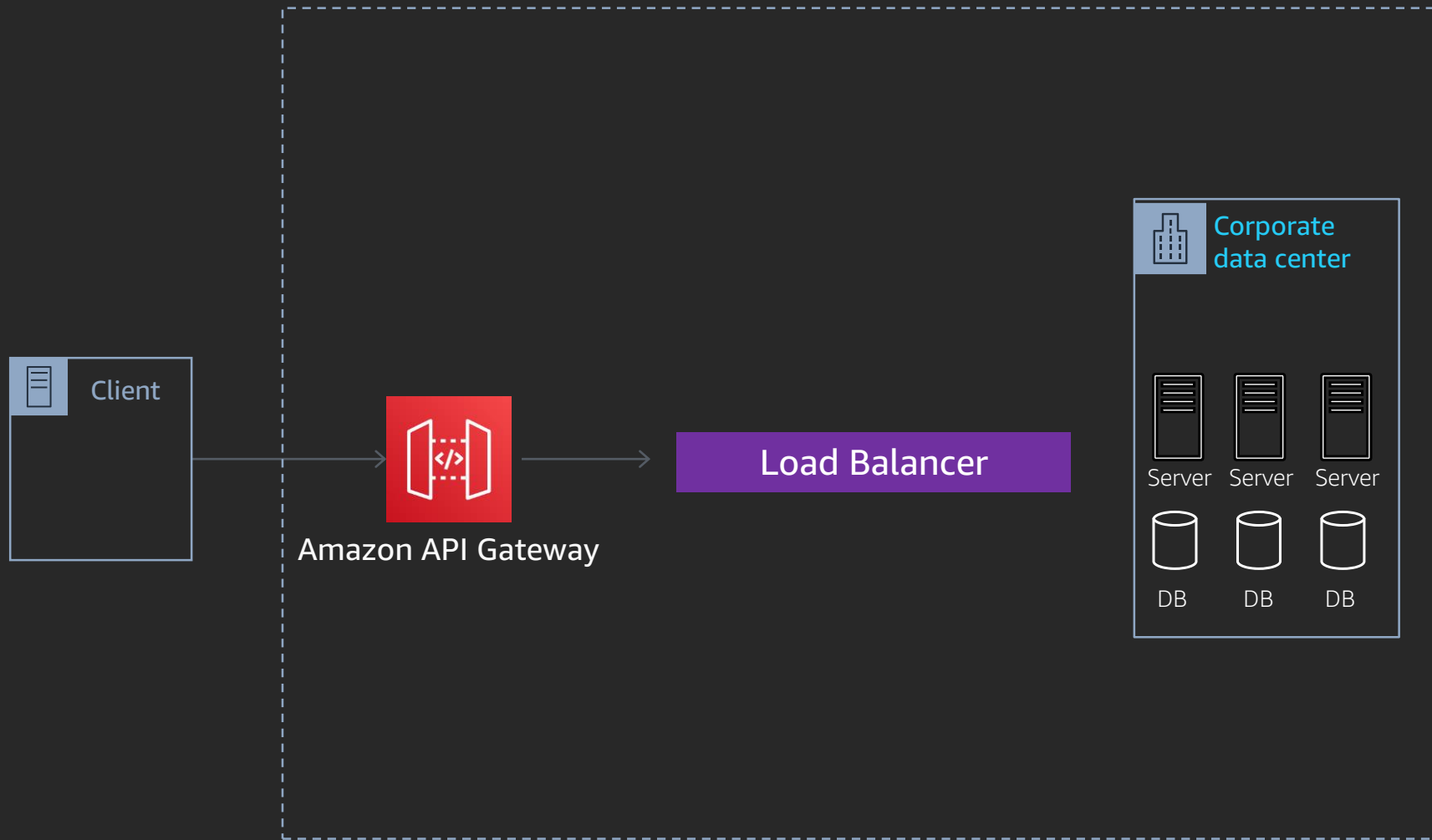
SECURITY

PERFORMANCE

COST



# Pattern: The “Strangler”



OPERATIONS

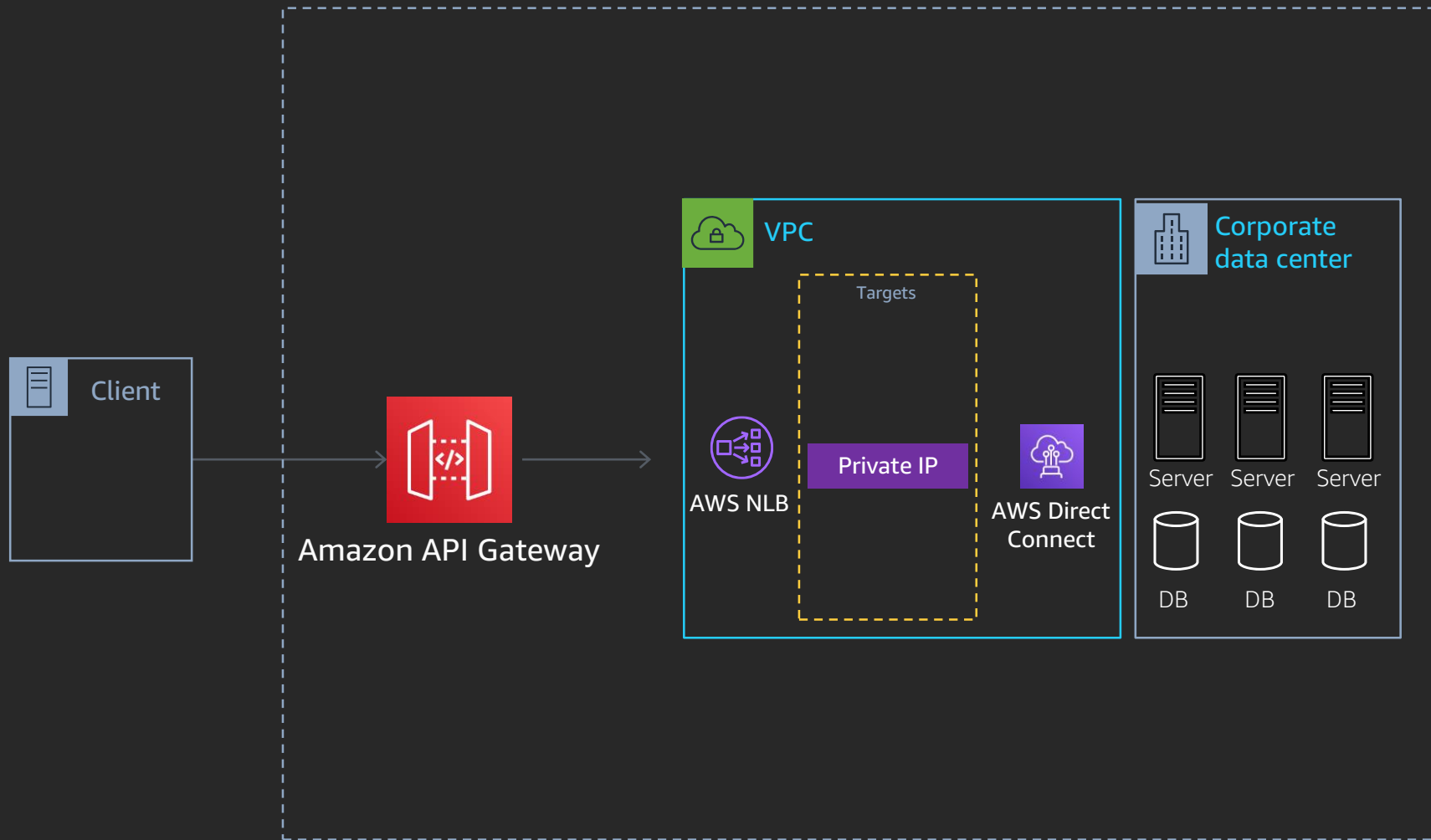
RELIABILITY

SECURITY

PERFORMANCE

COST

# Pattern: The “Strangler”



OPERATIONS

RELIABILITY

SECURITY

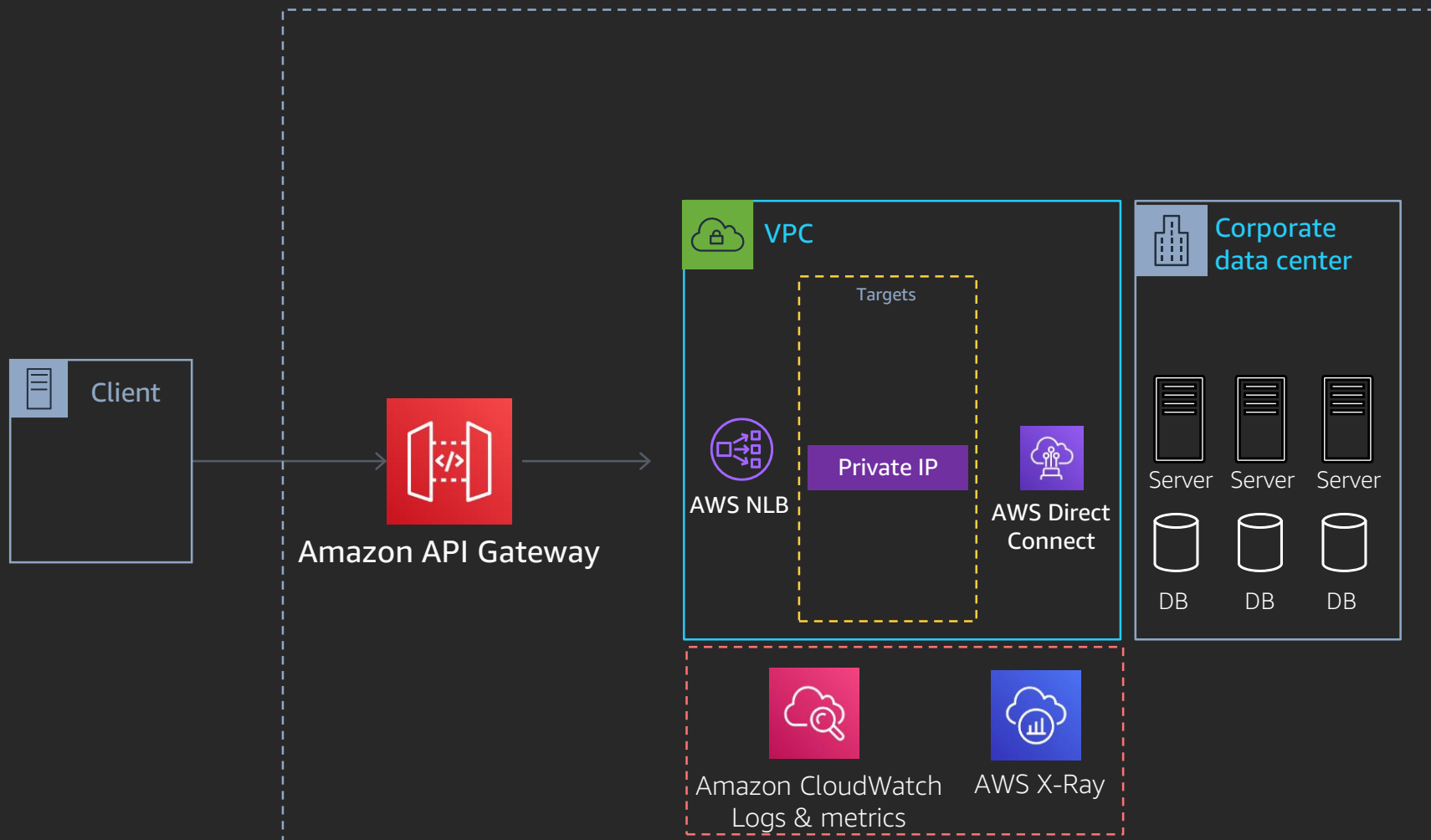
PERFORMANCE

COST

# Pattern: The “Strangler”

## Best practices

Centralize logs, metrics, and distributing tracing



OPERATIONS

RELIABILITY

SECURITY

PERFORMANCE

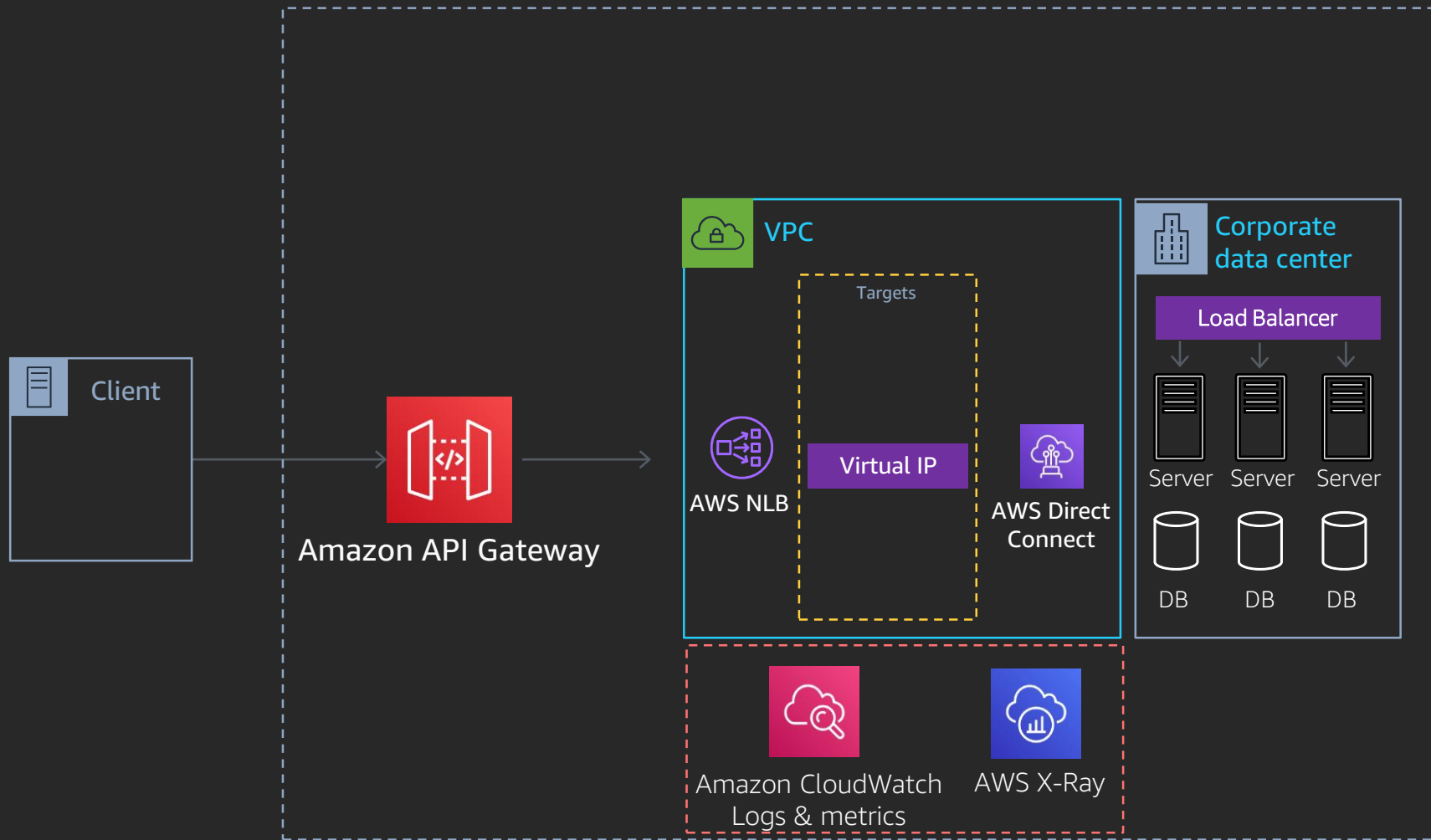
COST

# Pattern: The “Strangler”

## Best practices

- Centralize logs, metrics, and distributing tracing

- Use a corporate Load balancer virtual IP to send traffic to



OPERATIONS

RELIABILITY

SECURITY

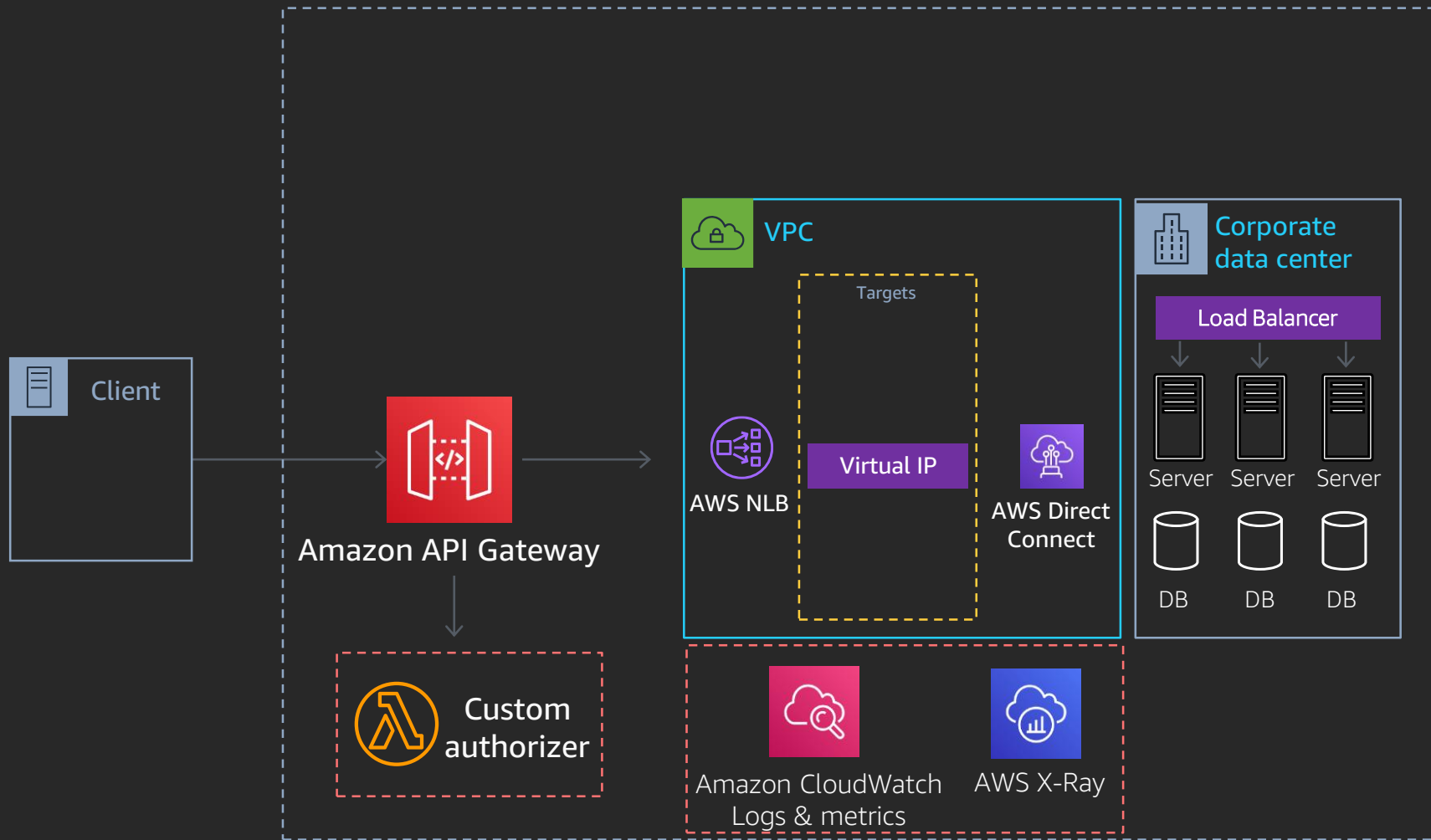
PERFORMANCE

COST

# Pattern: The “Strangler”

## Best practices

- Centralize logs, metrics, and distributing tracing
- Use a corporate Load balancer virtual IP to send traffic to
- Enforce authorization



OPERATIONS

RELIABILITY

SECURITY

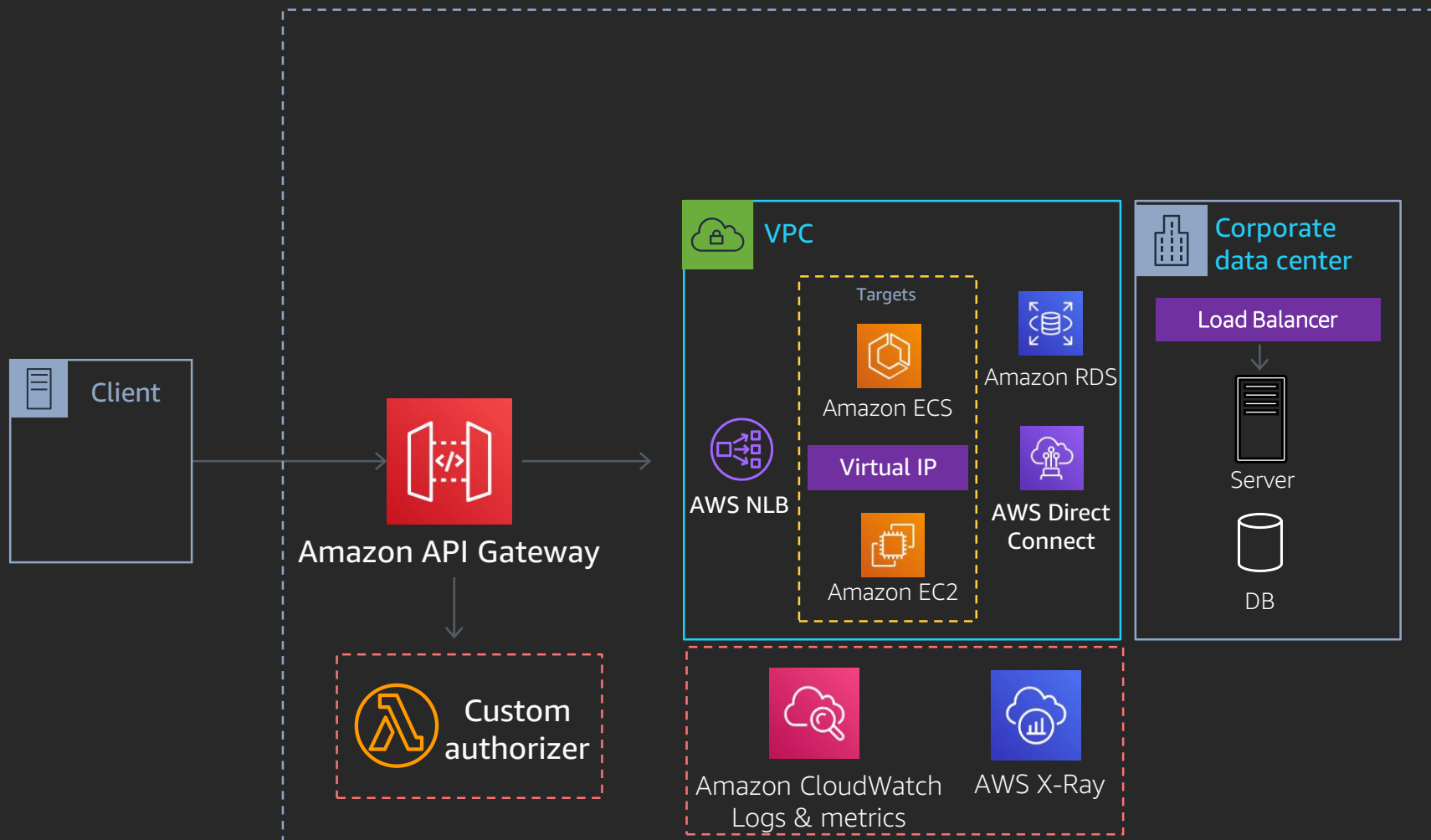
PERFORMANCE

COST

# Pattern: The “Strangler”

## Best practices

- Centralize logs, metrics, and distributing tracing
- Use a corporate Load balancer virtual IP to send traffic to
- Enforce authorization
- Gradually shift functionalities to newer compute/database platforms



OPERATIONS

RELIABILITY

SECURITY

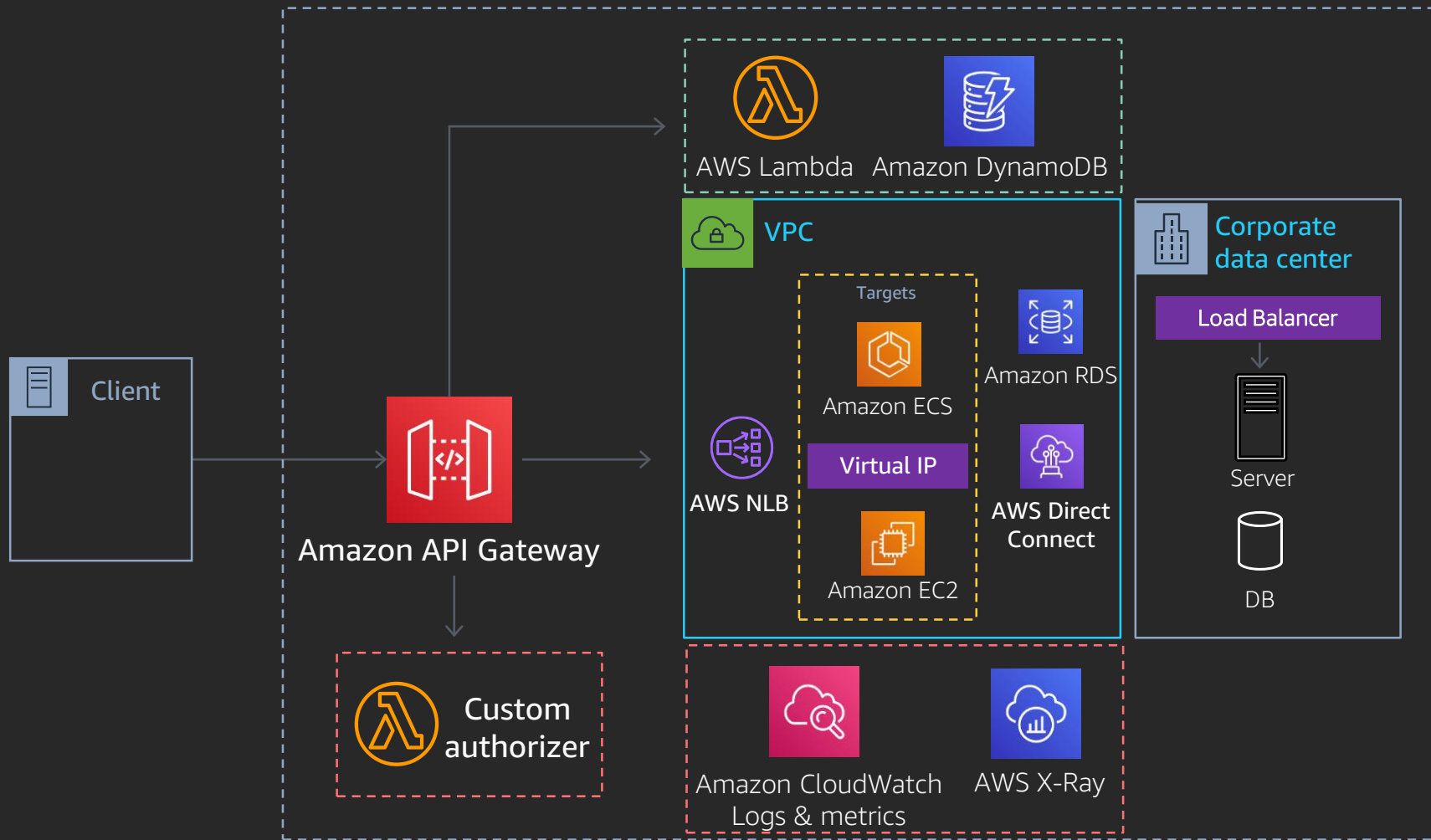
PERFORMANCE

COST

# Pattern: The “Strangler”

## Best practices

- Centralize logs, metrics, and distributing tracing
- Use a corporate Load balancer virtual IP to send traffic to
- Enforce authorization
- Gradually shift functionalities to newer compute/database platforms
- Use serverless for new functionalities



OPERATIONS

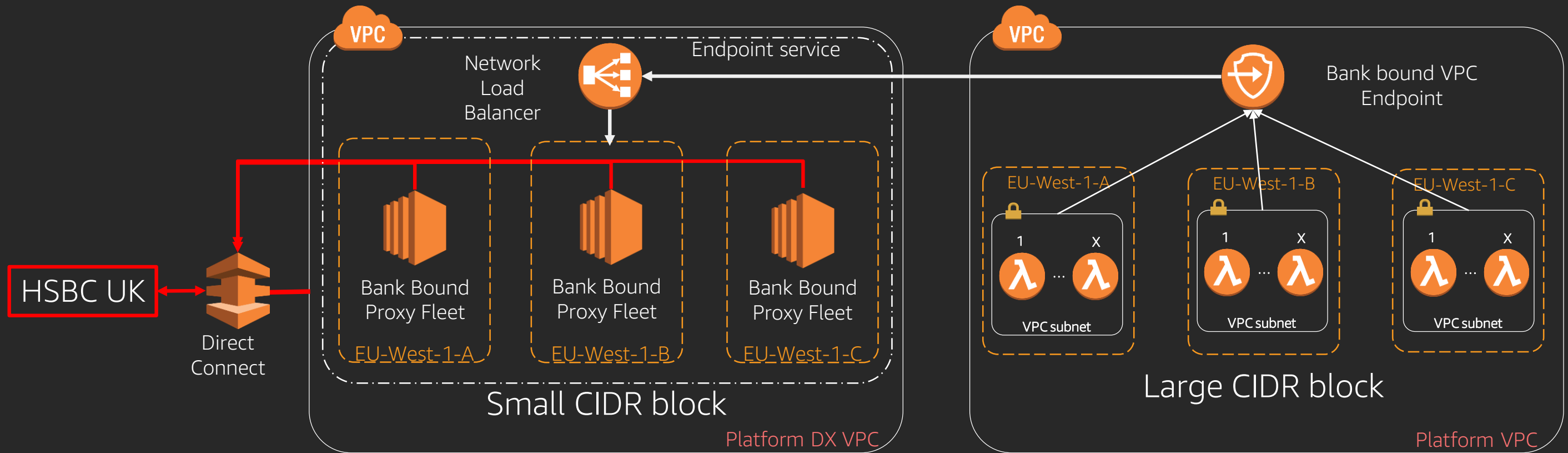
RELIABILITY

SECURITY

PERFORMANCE

COST

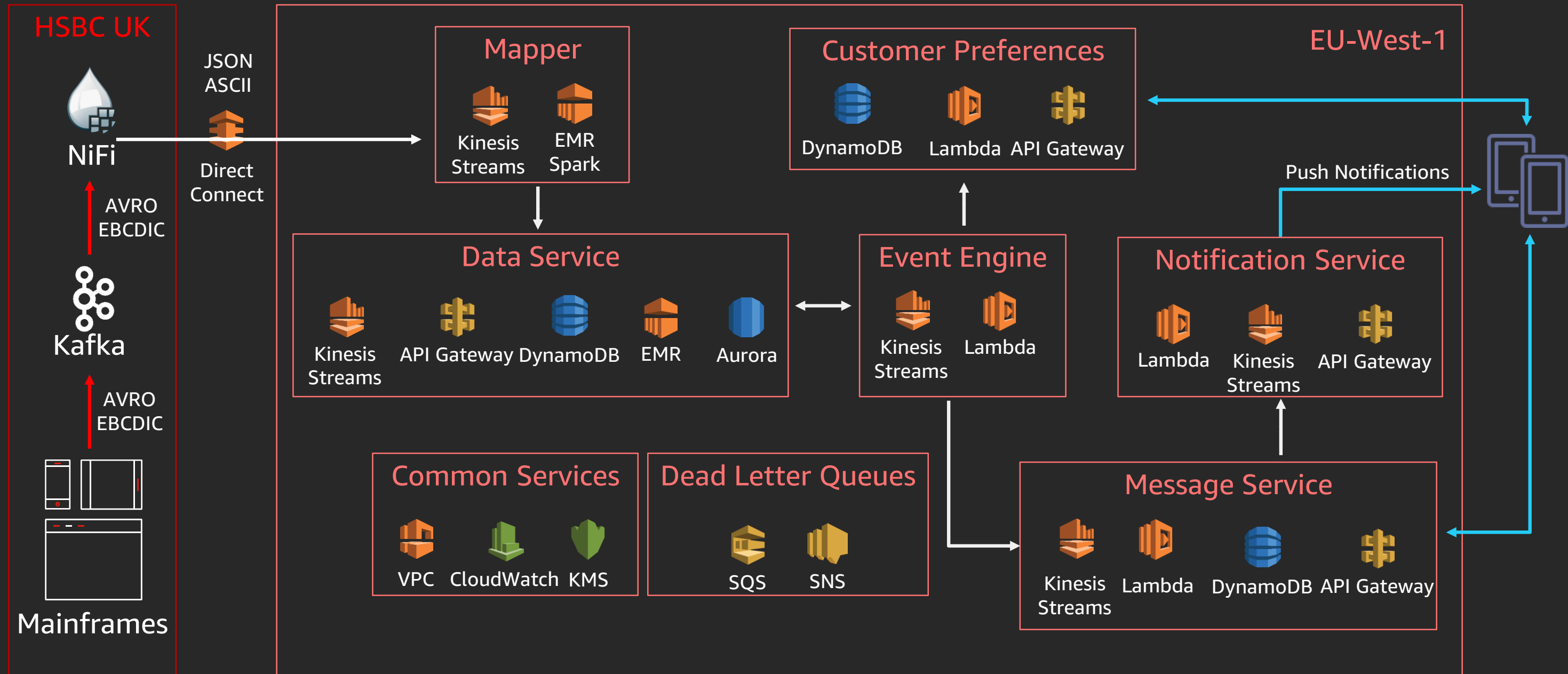
# Practical example: HSBC Part 1



- As VPC attached Lambda function scales, subnets must have available IP addresses to match the number of ENIs = large CIDR block required to your VPC
- Access to on-premise provided via VPC endpoint which encapsulates a set of proxy servers located on a VPC with Direct Connect = small CIDR used on VPC connected to on-premise

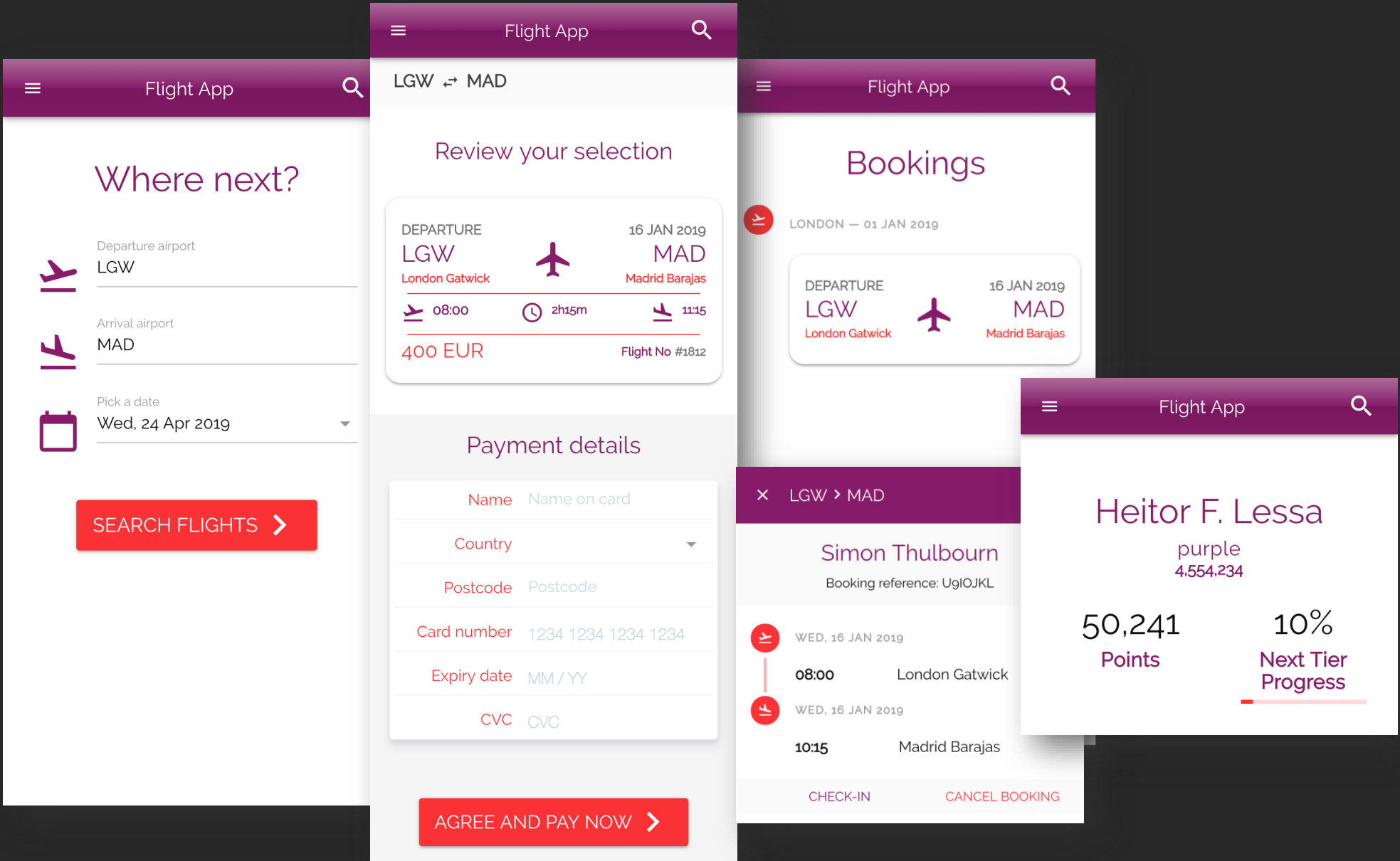


# Practical example: HSBC Part 2



# Summary

# Serverless airline – Multiple patterns/practices



# Summary

## OPERATIONS

Understand the health and lifecycle of your application

## RELIABILITY

Build resiliency and protect non-serverless resources

## SECURITY

Focus on managing security boundaries and AppSec

## PERFORMANCE

Optimize for low, steady, and/or peaks

## COST

Factor in development, people, opportunity, and maintenance cost

# Related breakouts

SVS311 Serverless at scale: design patterns and optimizations

SVS401-R Optimizing your serverless applications

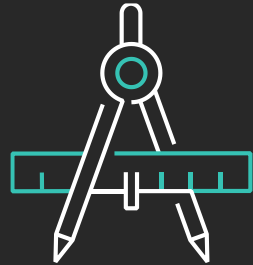
SVS403-R Best practices for AWS Lambda and Java

API304 Scalable serverless event-driven applications using Amazon SQS and Lambda

SVS309 Architecting and operating resilient serverless systems at scale

# Learn to architect with AWS Training and Certification

Resources created by the experts at AWS to propel your organization and career forward



Free foundational to advanced digital courses cover AWS services and teach architecting best practices



Classroom offerings, including Architecting on AWS, feature AWS expert instructors and hands-on labs



Validate expertise with the **AWS Certified Solutions Architect - Associate** or **AWS Certification Solutions Architect - Professional** exams

Visit [aws.amazon.com/training/path-architecting/](https://aws.amazon.com/training/path-architecting/)



Please complete the session  
survey in the mobile app.

# Thank you!

**Heitor Lessa**

@heitor\_lessa