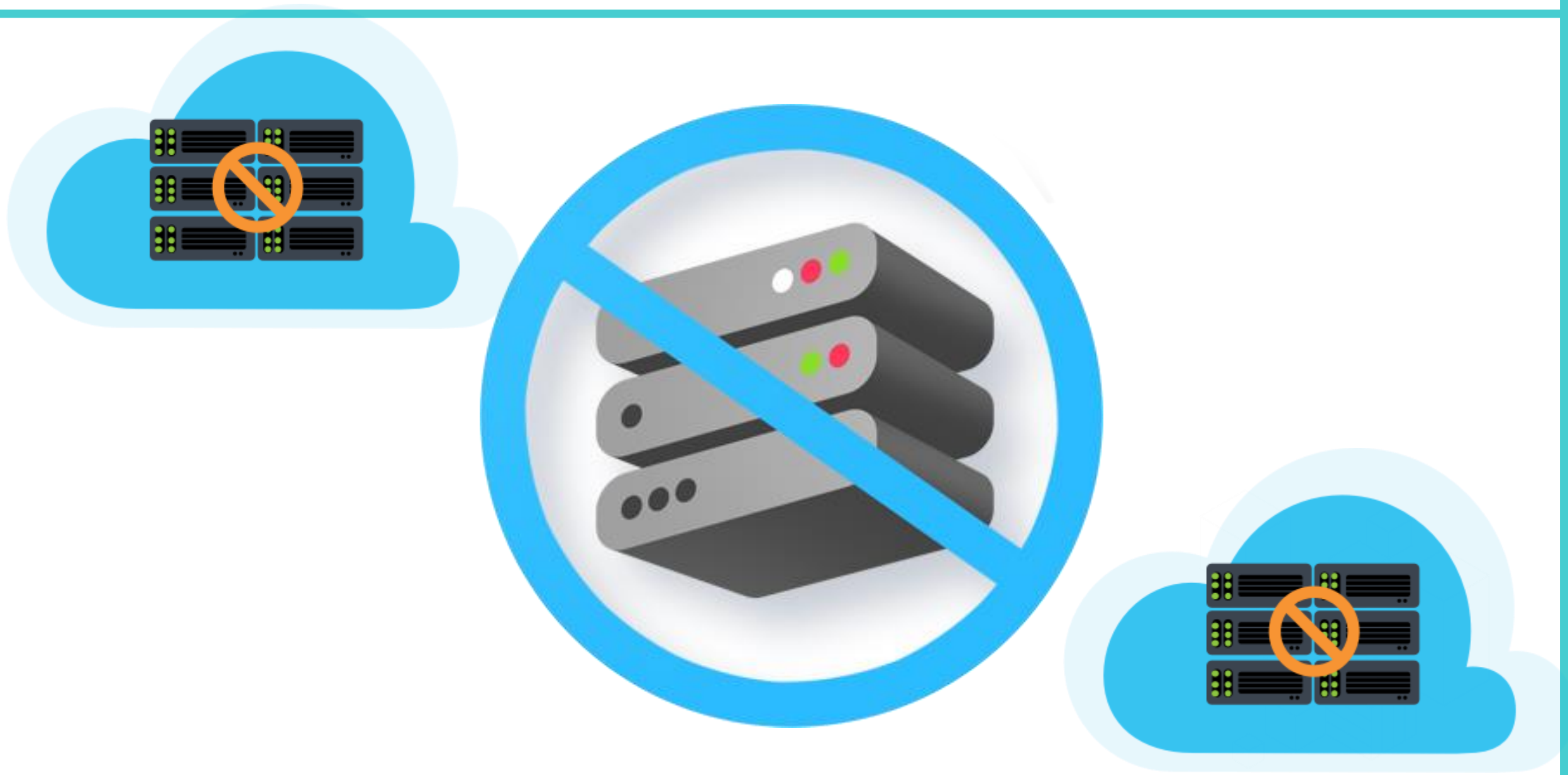
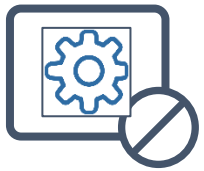
Serverless

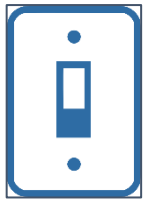


Why Serverless?



No infrastructure provisioning,
no management

Automatic scaling

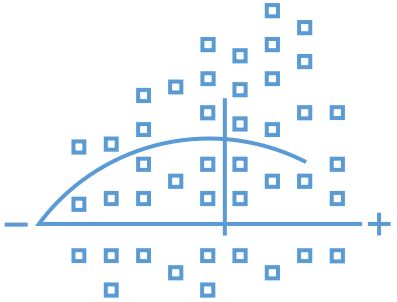


Pay for value

Highly available and secure

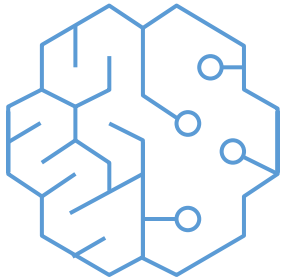
What are customers building with Serverless?

IT Automation



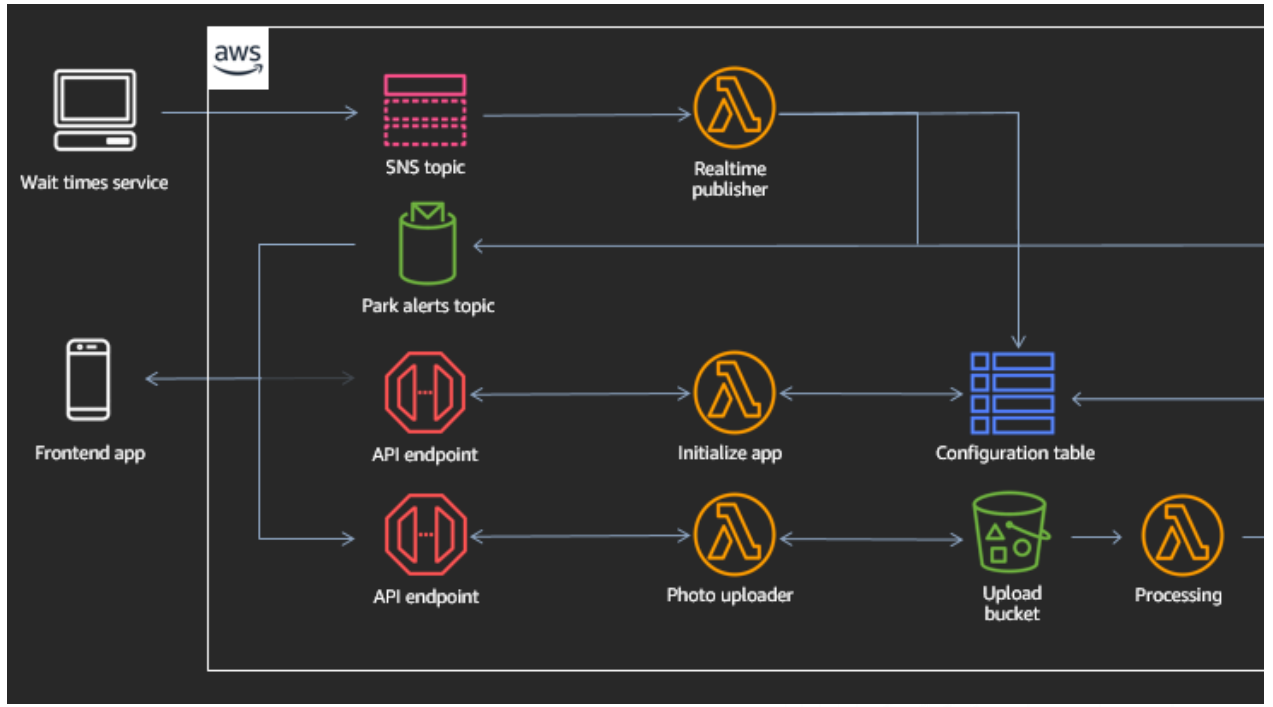
Data processing

Web applications



Machine Learning

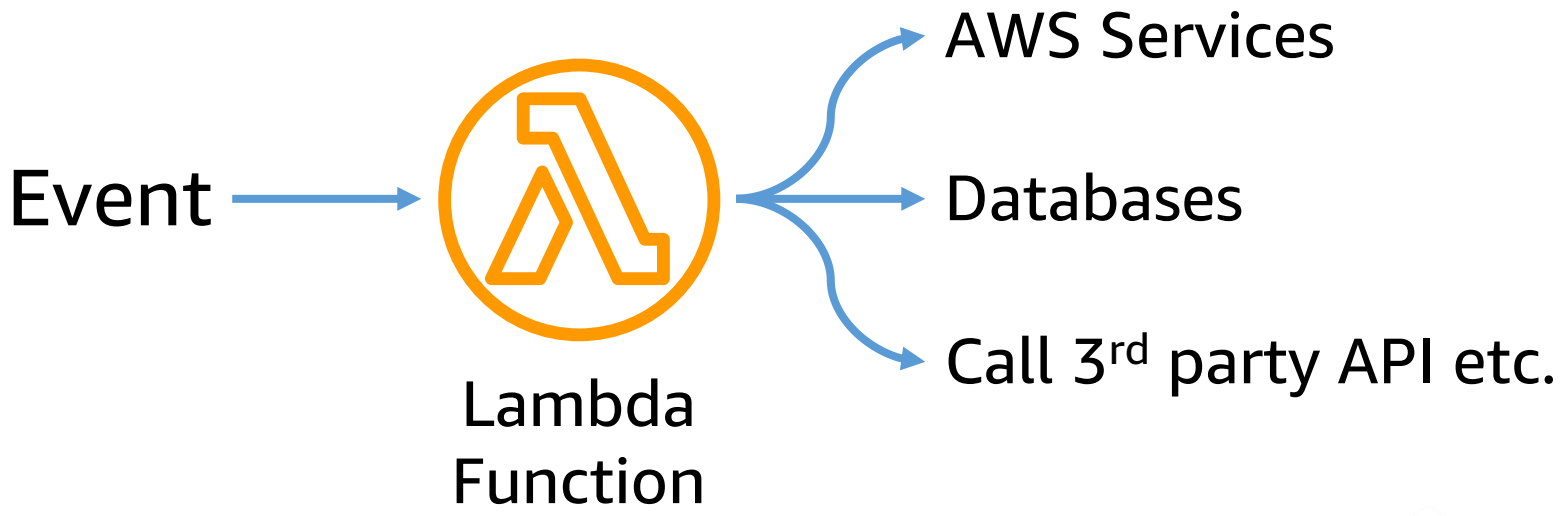
Small components, loosely coupled



Why Serverless?

- Customers across many industries face the problem of large-scale, real-time data processing
- In many cases, no single server is sufficient to handle the data volumes
- Building scalable and reliable distributed systems is difficult!
- Serverless AWS services such as Amazon DynamoDB, AWS Lambda, SNS, SQS, EventBridge, Amazon Kinesis provide building blocks to make it easy, efficient, and reliable

What is Lambda? – High level view

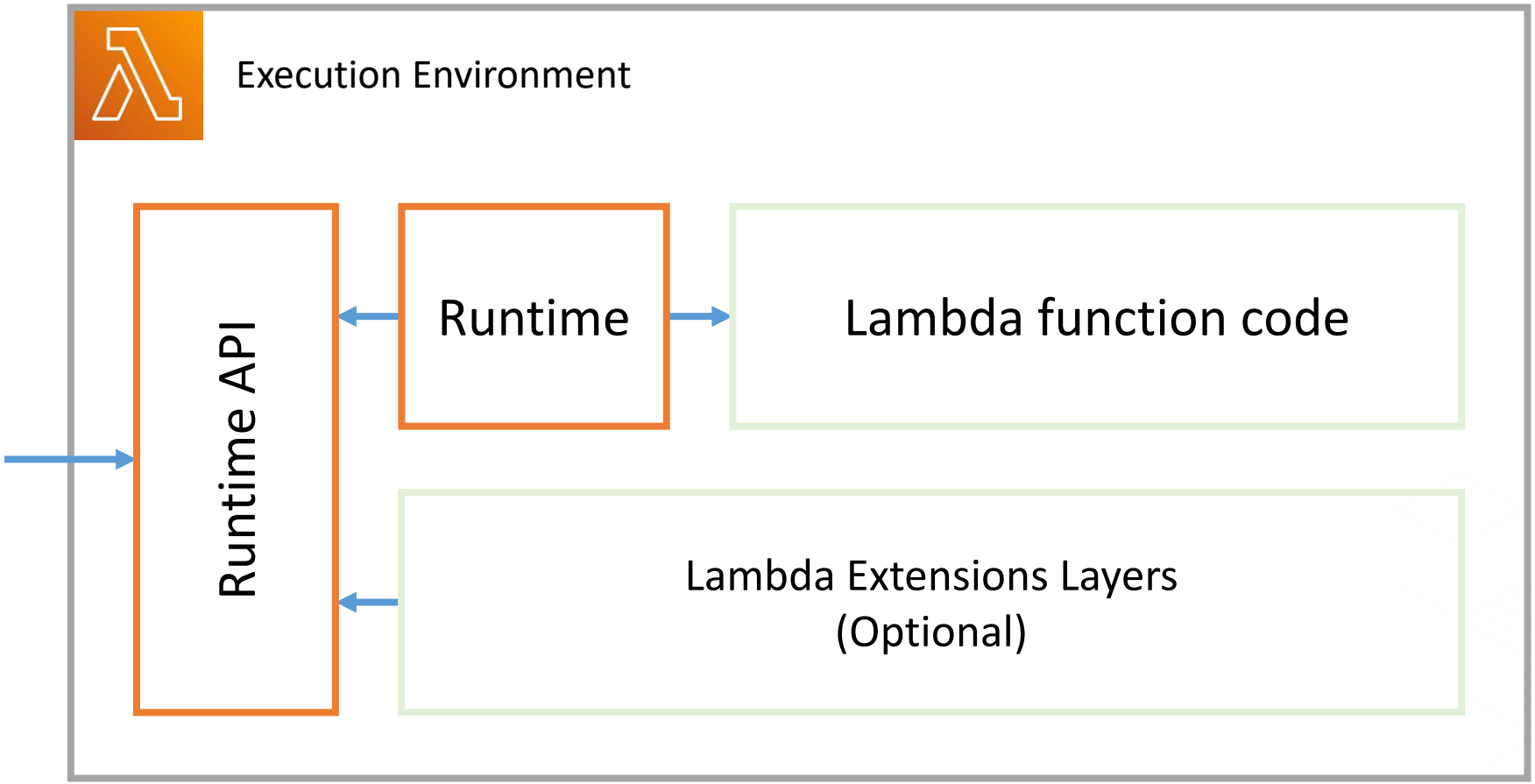


Python
NodeJS/Javascript
Java
.NET/C#
Ruby
BYOL
Container images

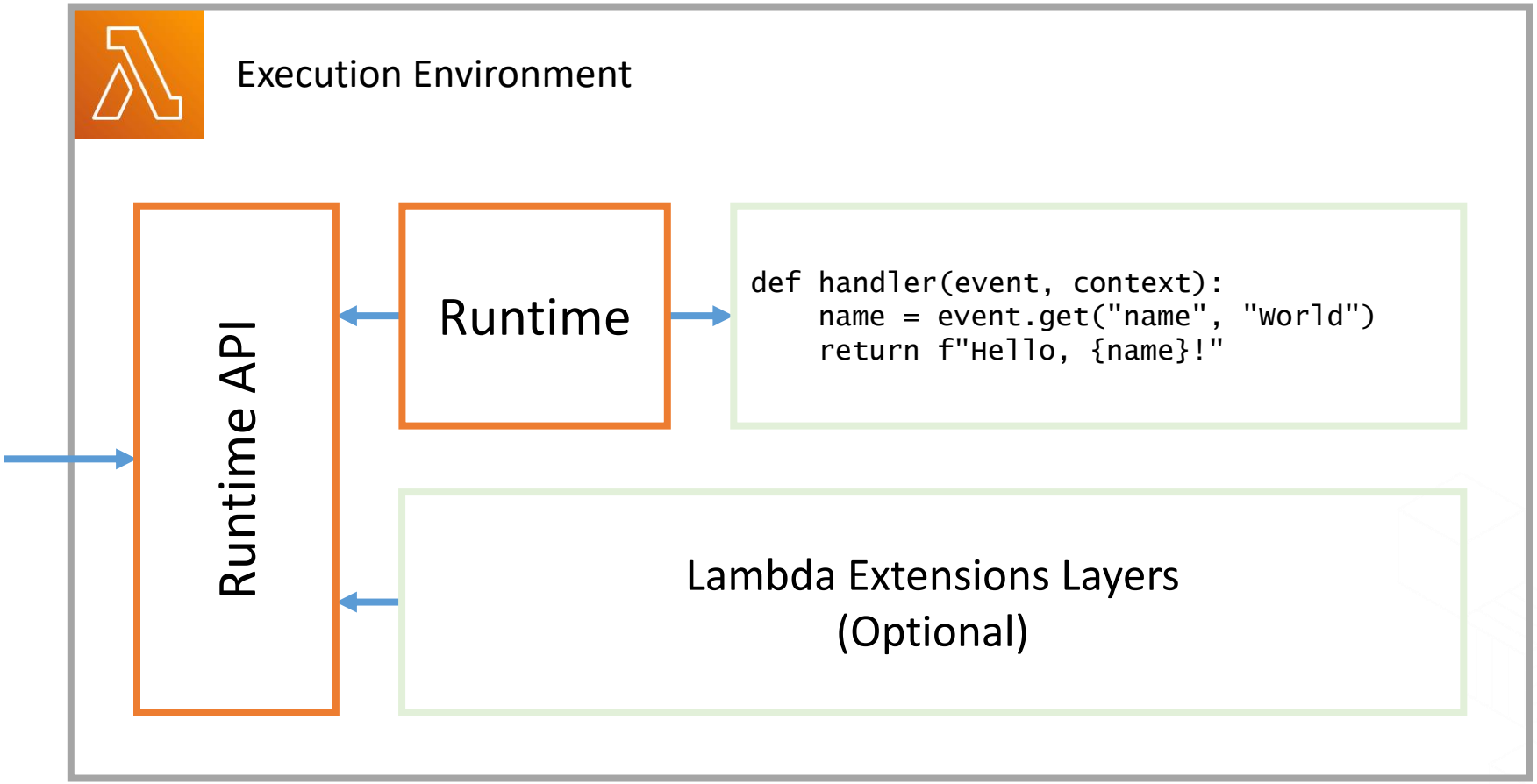
Firecracker

Secure and fast microVMs
for serverless computing

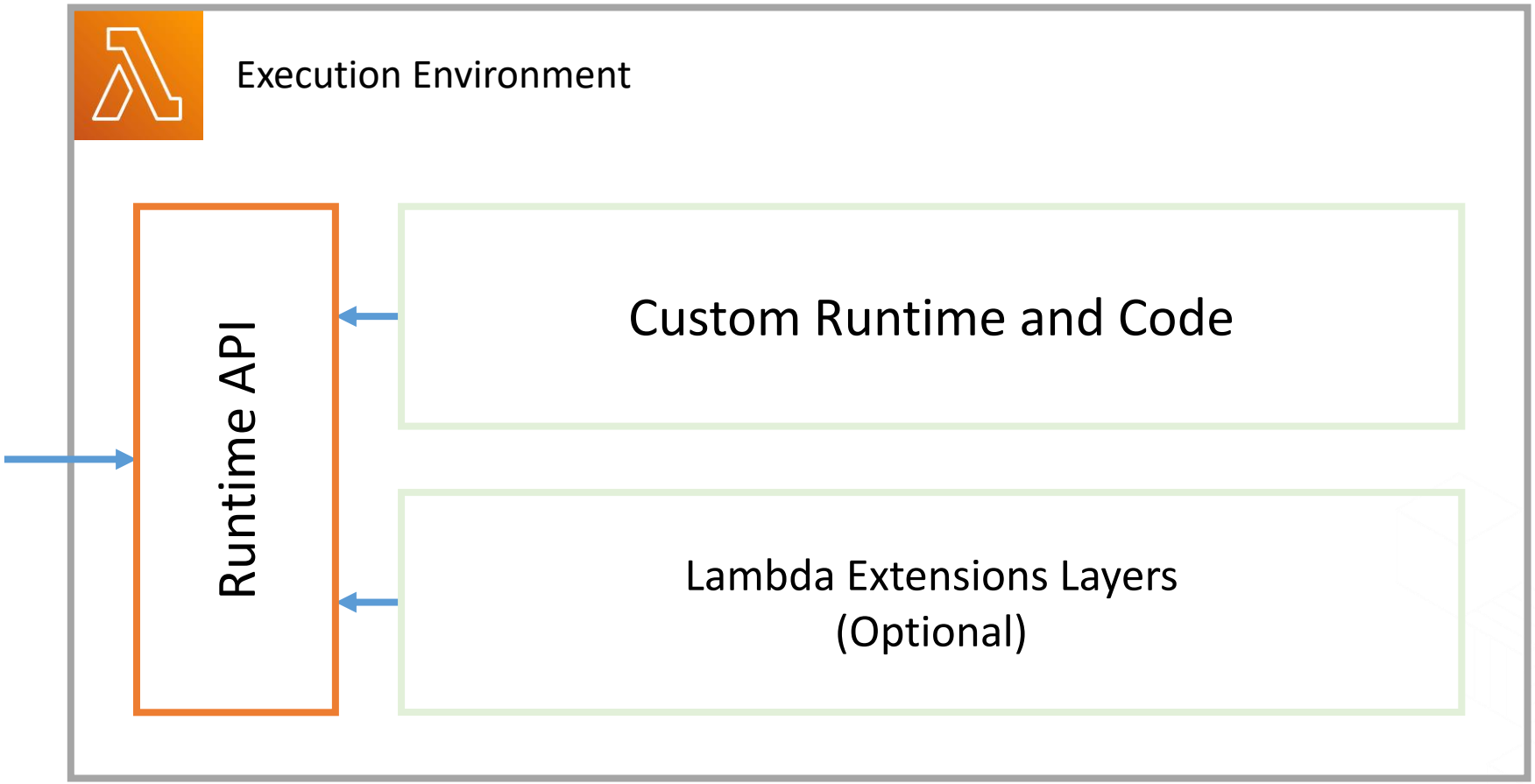
Handling requests (managed runtime)



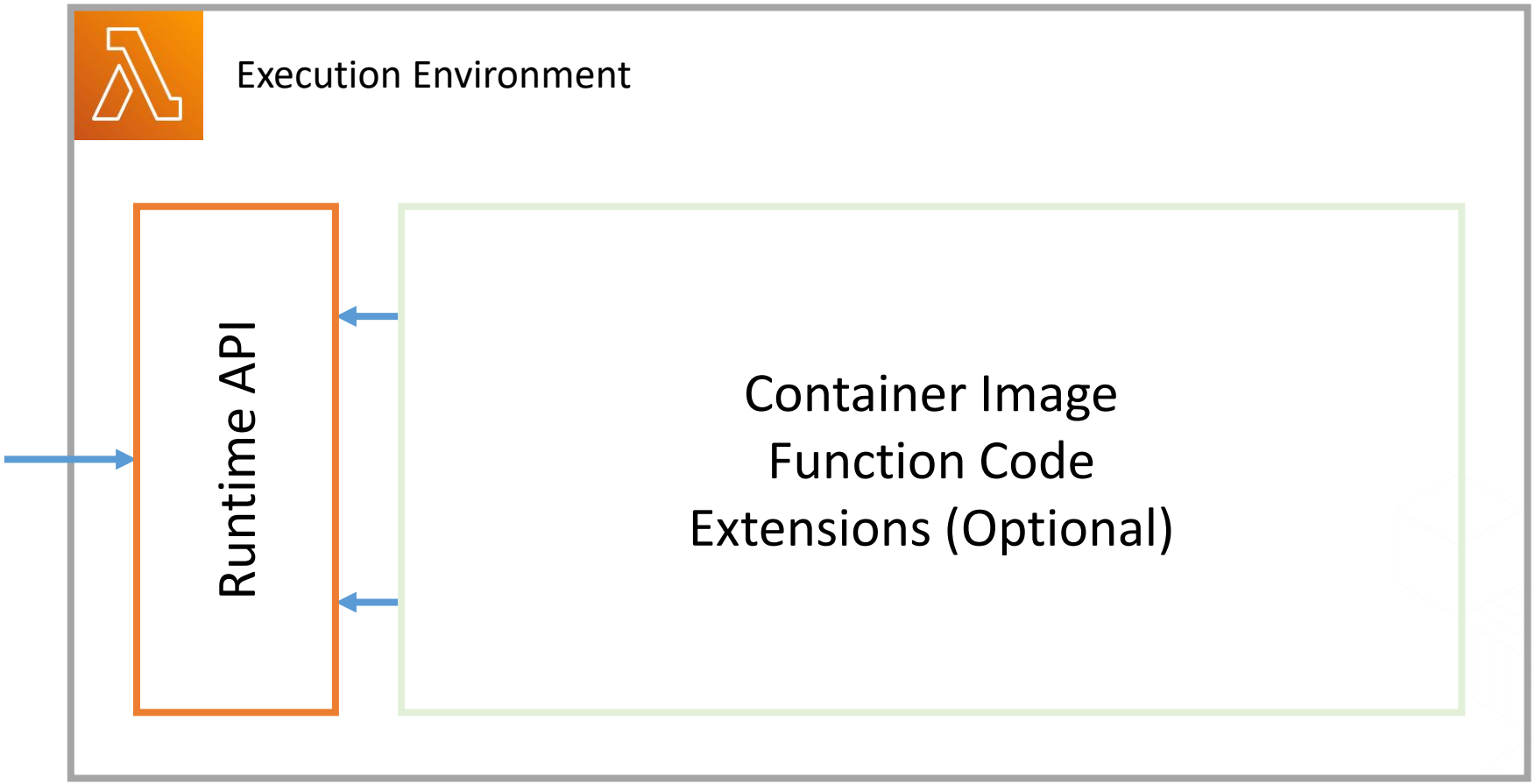
Handling requests (managed runtime)



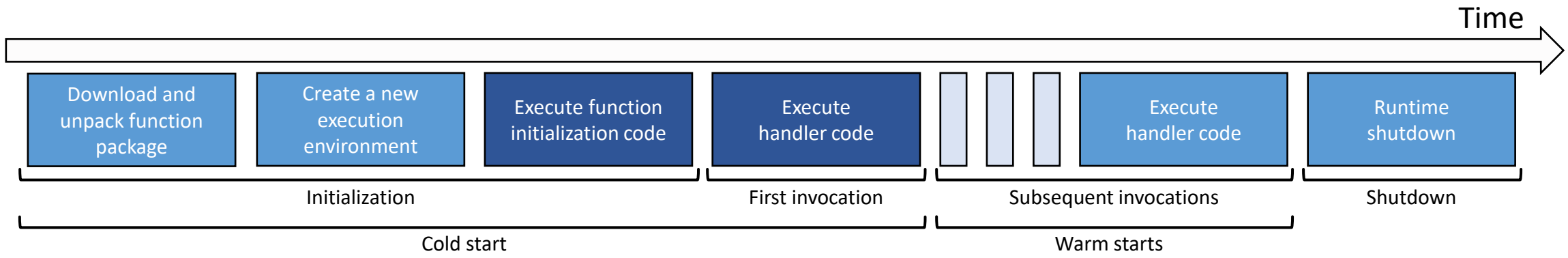
Handling requests (custom runtime)



Handling requests (container images)



Lambda Execution Lifecycle



Init: create a new or thaw existing execution environment. For new environments, run the function's initialization code.

⇨

Invoke: Run the function handler code, return response. Freeze execution environment until next invocation

⇨

Shutdown: triggered when execution environment is destroyed, e.g. if the function does not receive invocations for a period of time

AWS Lambda Invocation Model

Synchronous
request/response


Application Load Balancer


Amazon API Gateway


AWS Lambda function URL

request event

response


AWS Lambda function

Asynchronous
event


Amazon EventBridge


Amazon S3

event

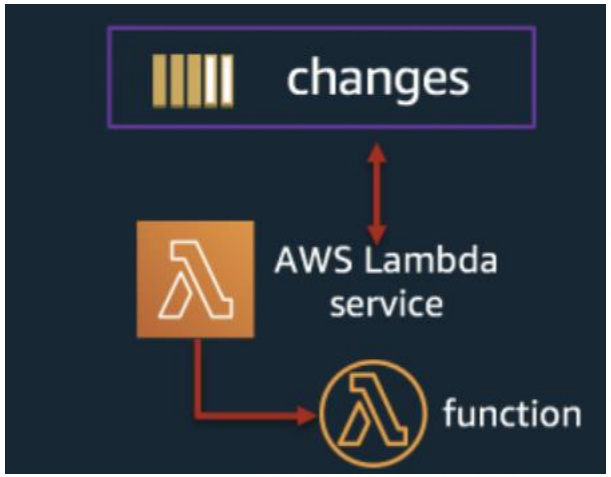
event queue


AWS Lambda function

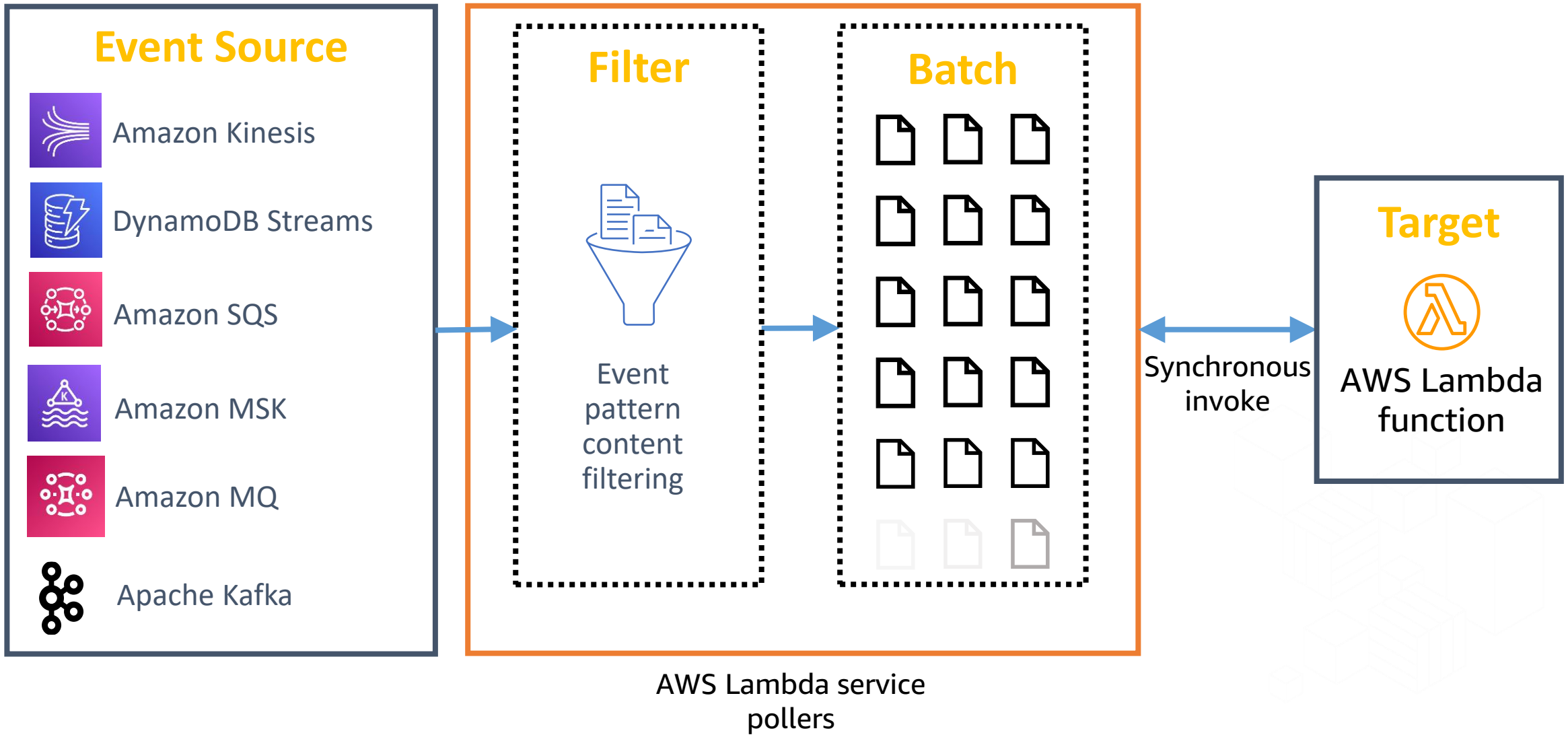
Stream
(poll based)


Amazon DynamoDB

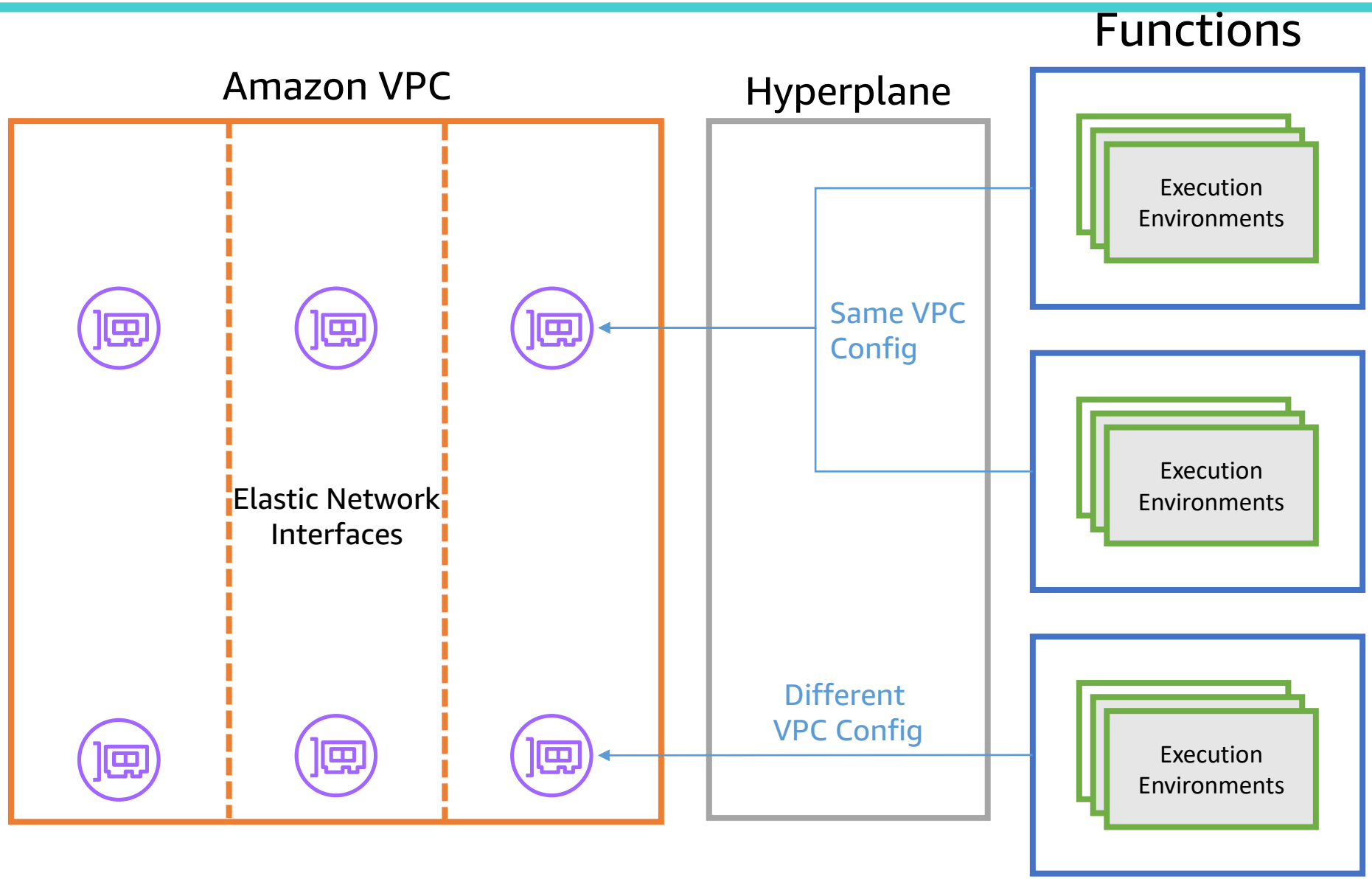

Amazon Kinesis



Lambda event source mappings



Lambda & VPCs



Lambda functions are not deployed in your VPC.

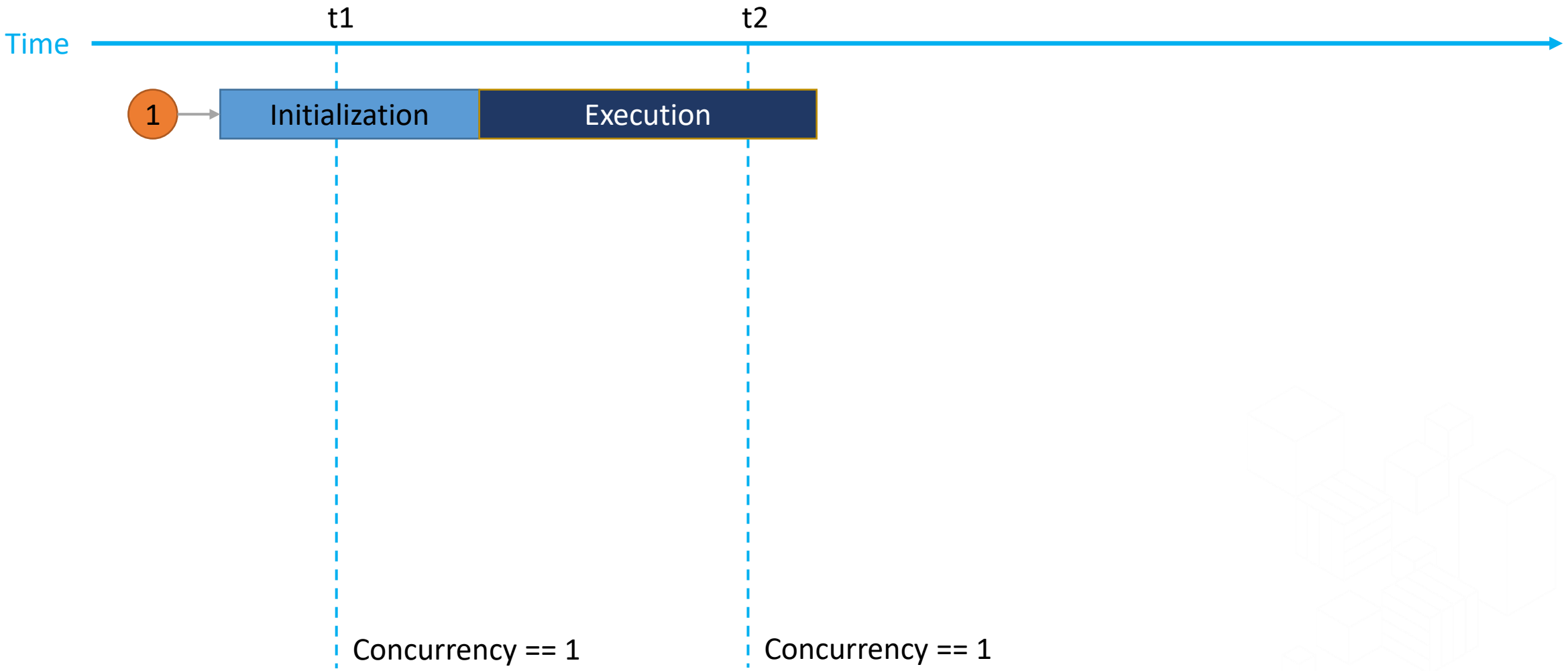
ENIs are tied to security group and subnet pairing.

Lambda concurrency

Lambda concurrency

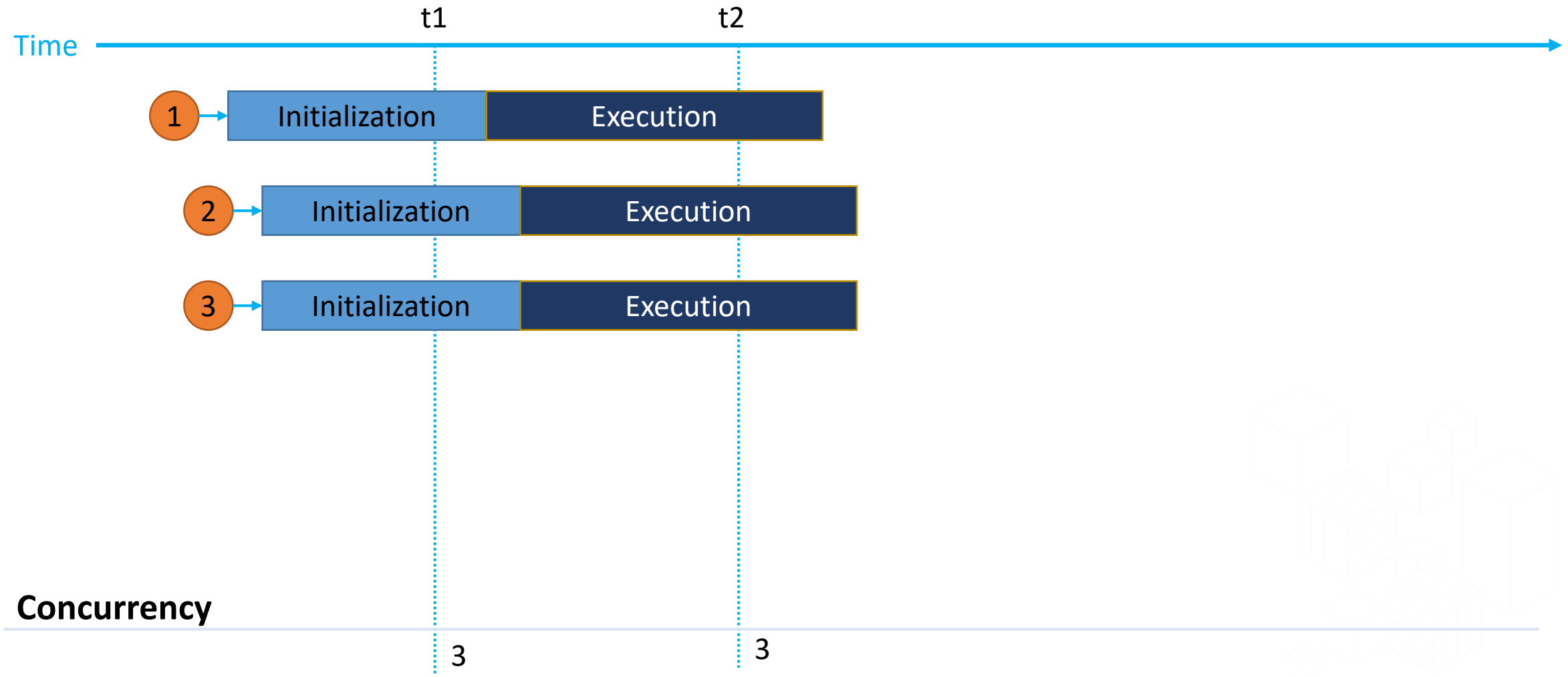
Lambda concurrency

Lambda concurrency

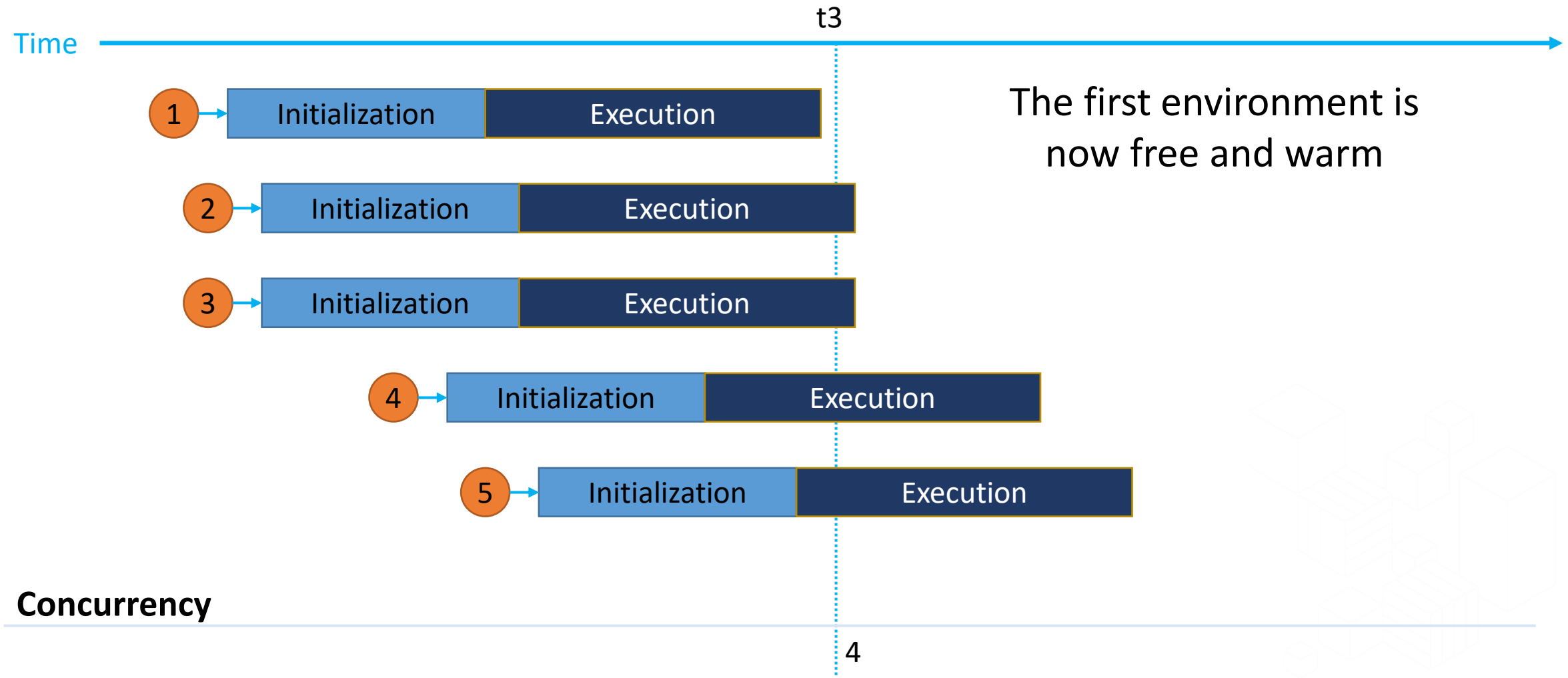


Lambda concurrency

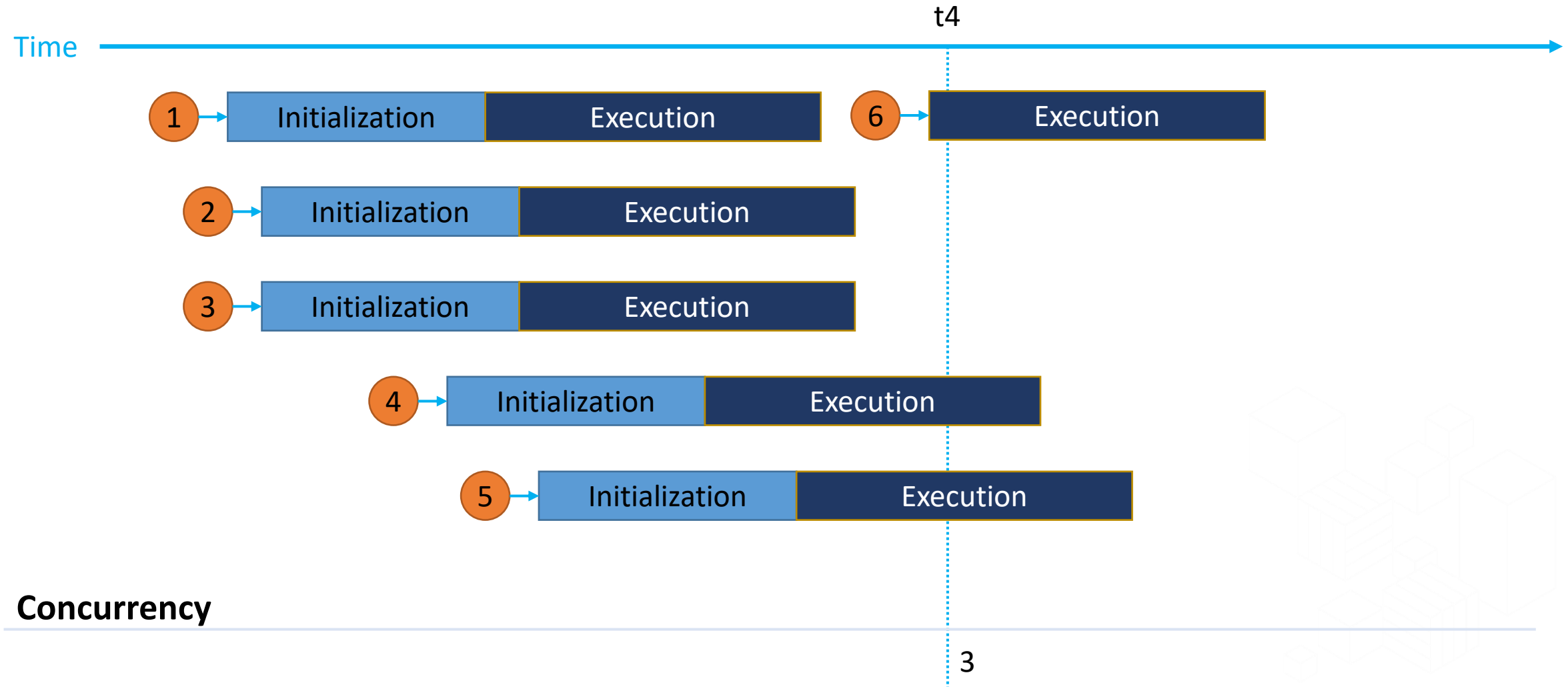
Lambda concurrency



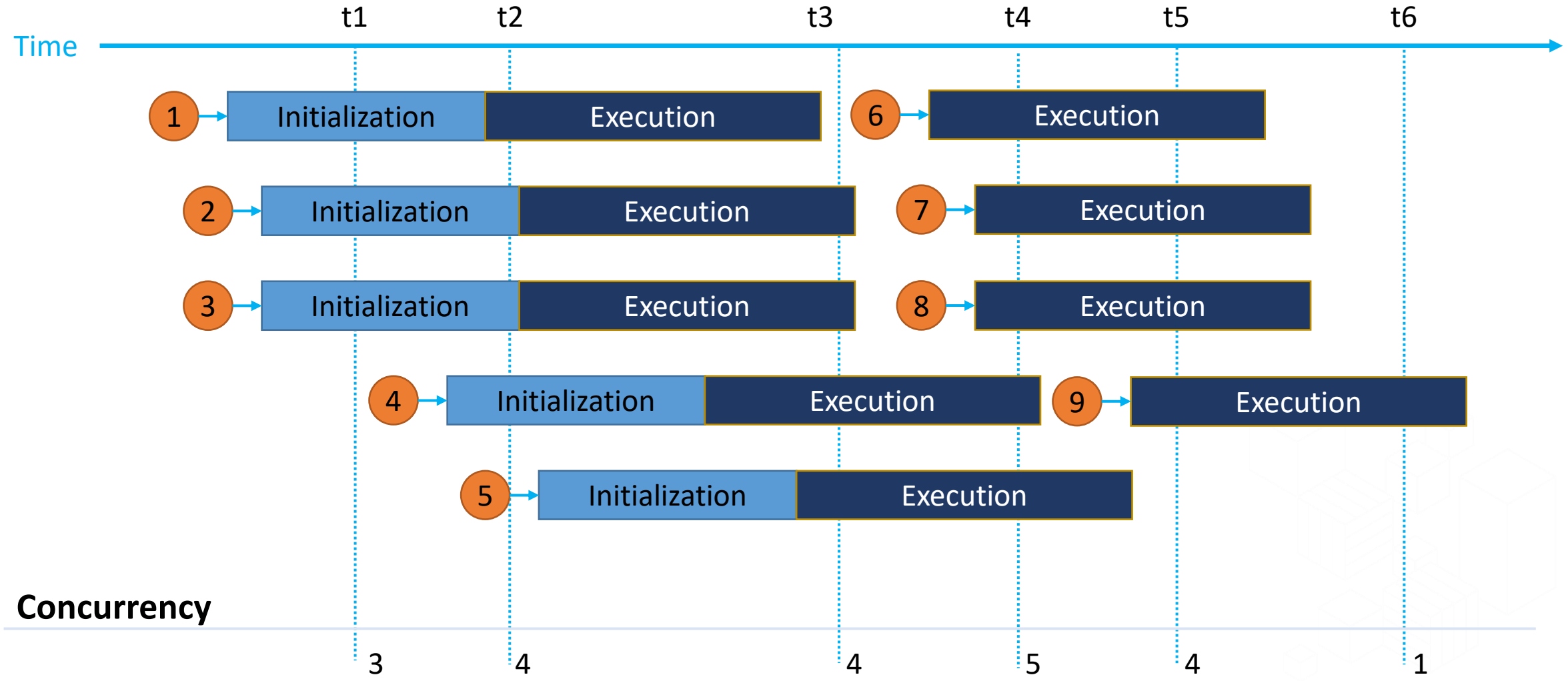
Lambda concurrency



Lambda Concurrency



Lambda concurrency



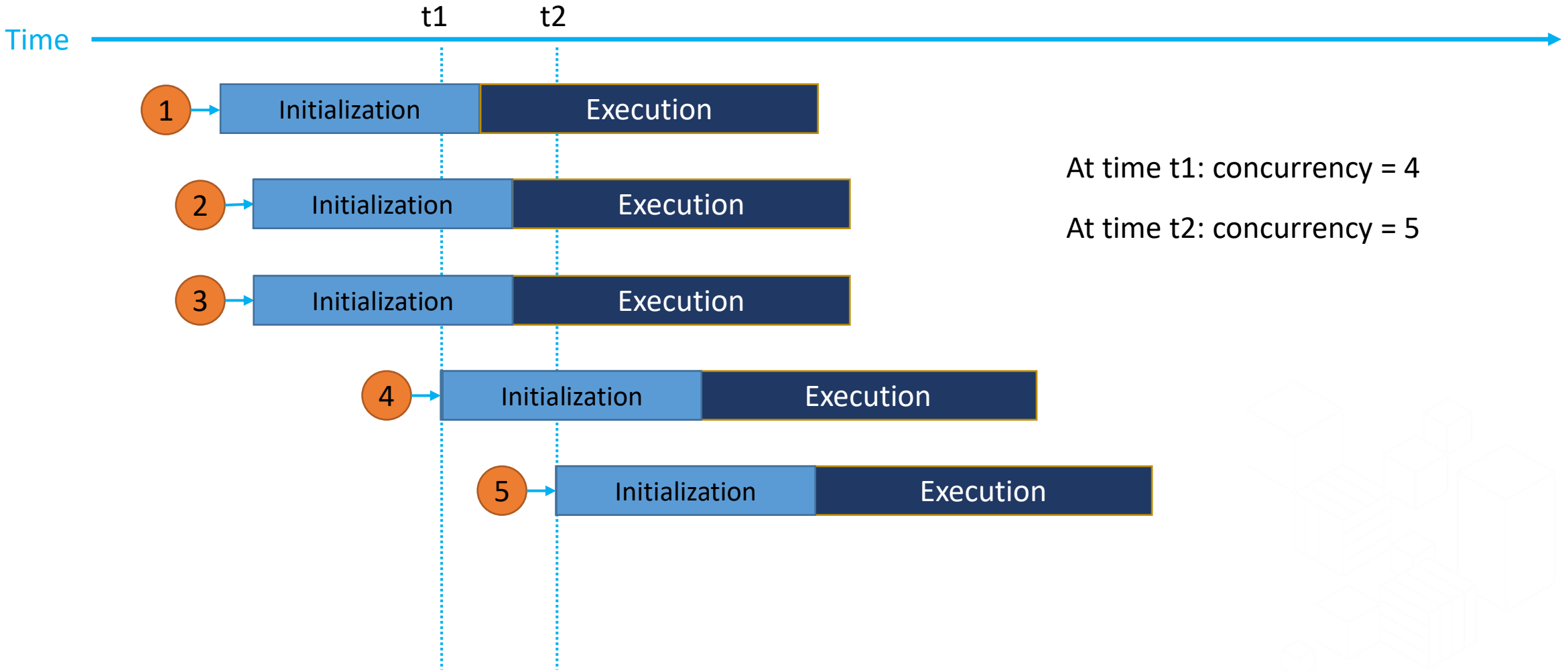
Lambda concurrency Calculation

Lambda concurrency = requests per second (RPS) x duration in seconds

Long-running functions require more concurrency

- Examples
- 10,000 RPS x 1,000 ms = 10,000 concurrency
- 10,000 RPS x 500 ms = 5,000 concurrency
- 10,000 RPS x 100 ms = 1,000 concurrency

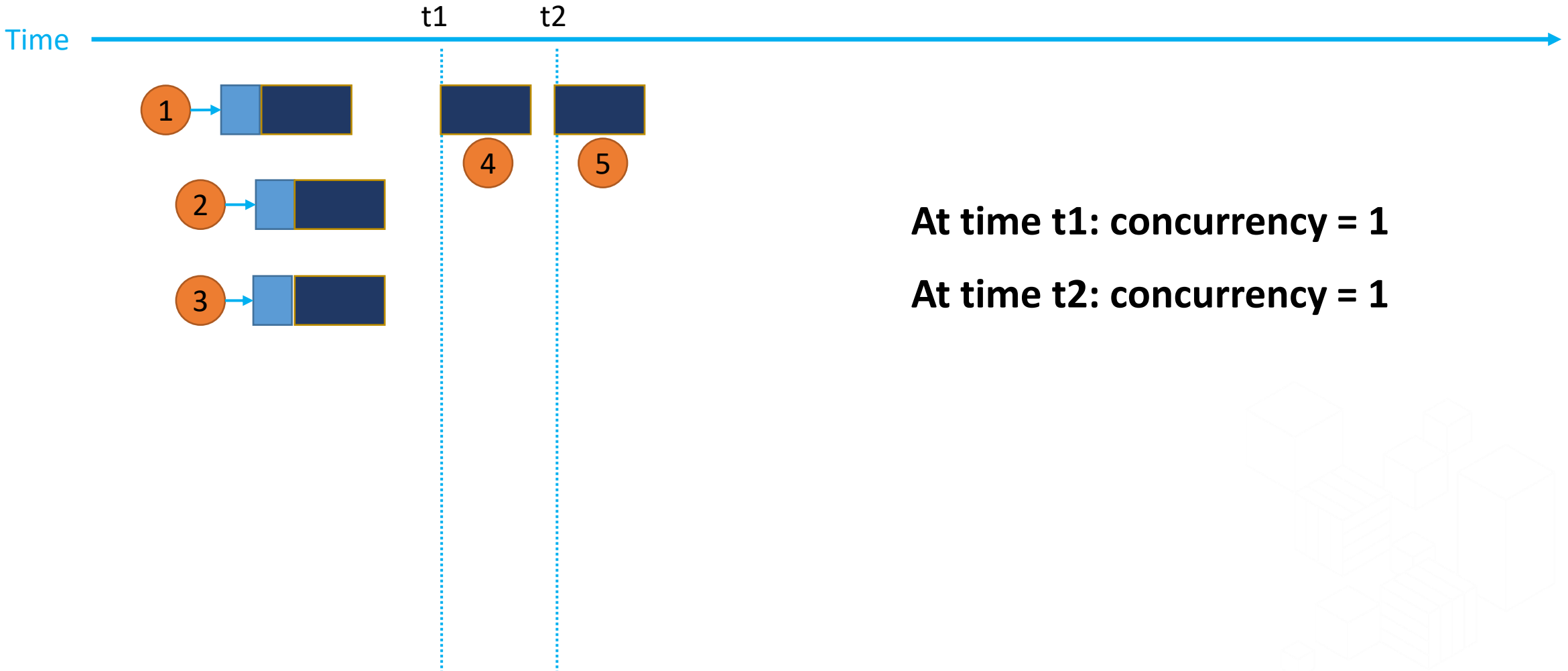
Decreasing concurrency



At time t1: concurrency = 4

At time t2: concurrency = 5

Decreasing concurrency



At time t1: concurrency = 1

At time t2: concurrency = 1

Lambda Scaling Quotas

Account concurrency

Maximum concurrency in a given region across all functions.

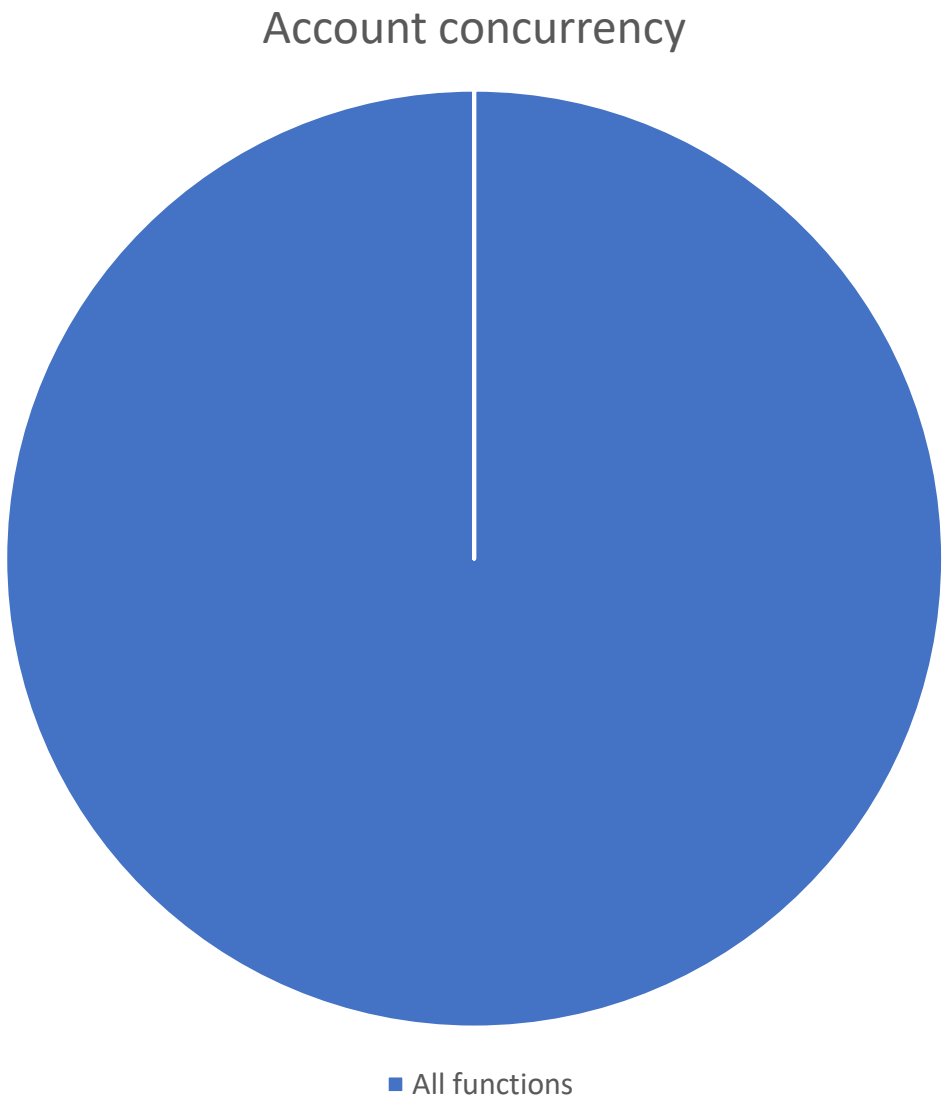
1000 in all regions

This can be increased

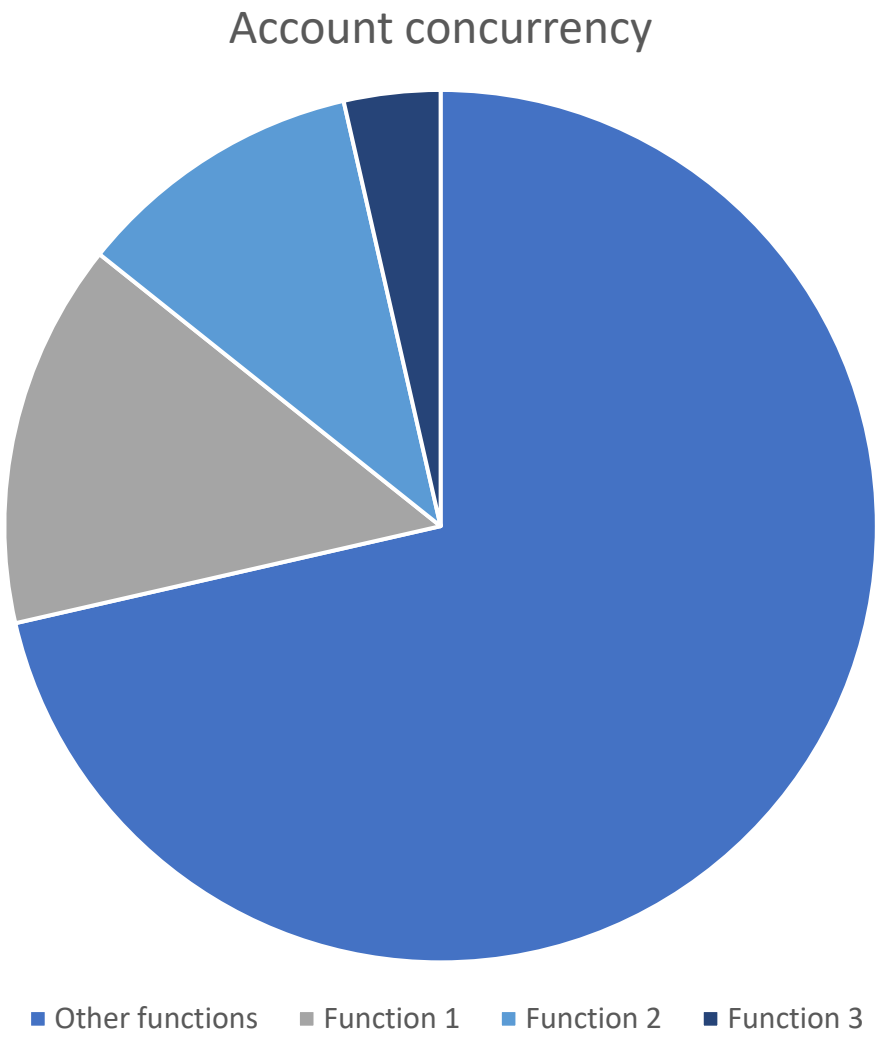
Function quota

Scaling rate per function, in each region.
1000 new concurrent executions every 10 seconds
This can NOT be increased

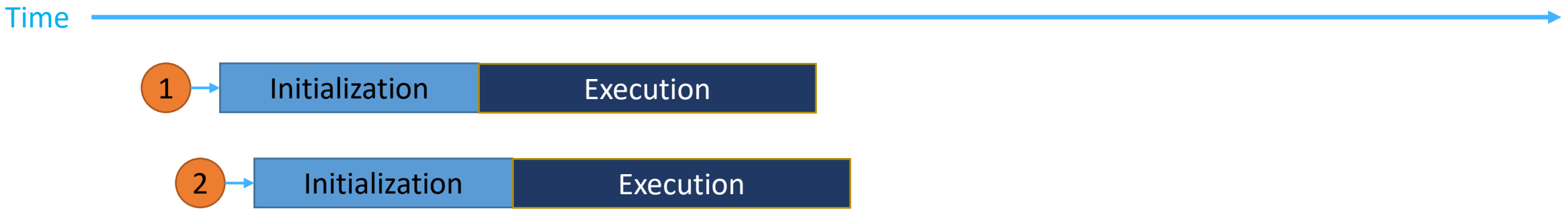
Reserved concurrency



Reserved concurrency

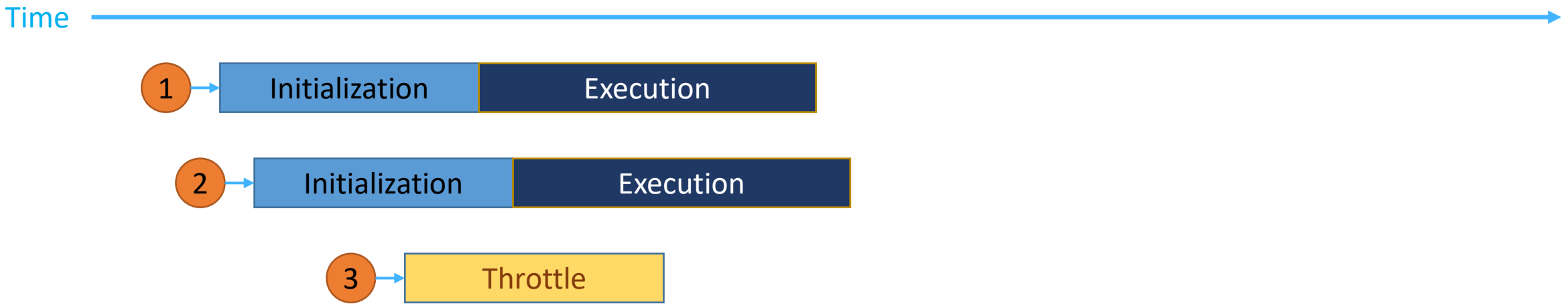


How Lambda scales: Reserved concurrency



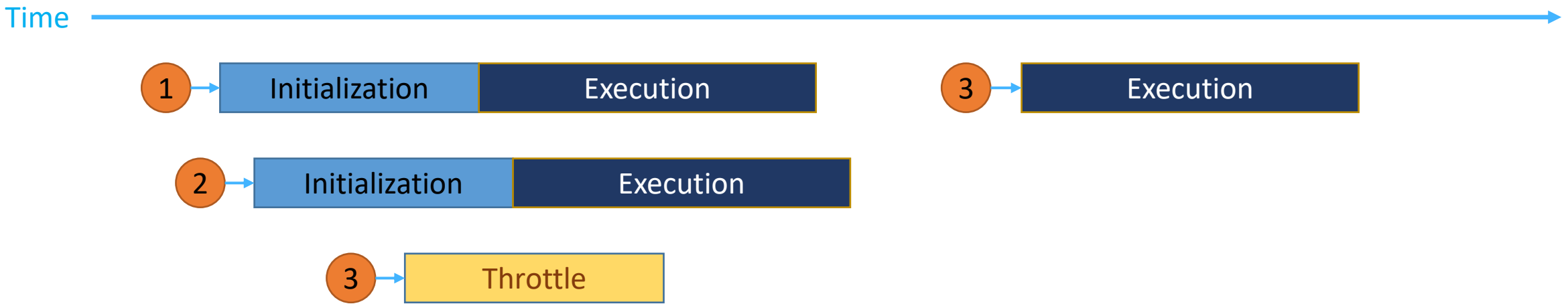
Reserved Concurrency = 2

How Lambda scales: Reserved concurrency



Reserved Concurrency = 2

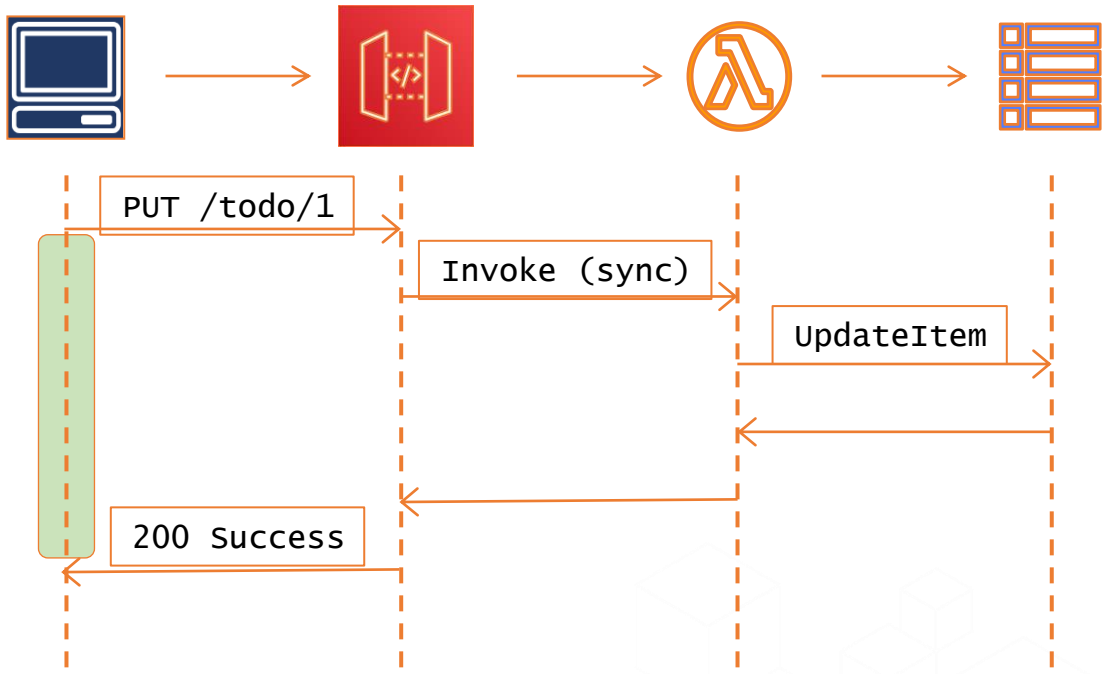
How Lambda scales: Reserved concurrency



Reserved Concurrency = 2

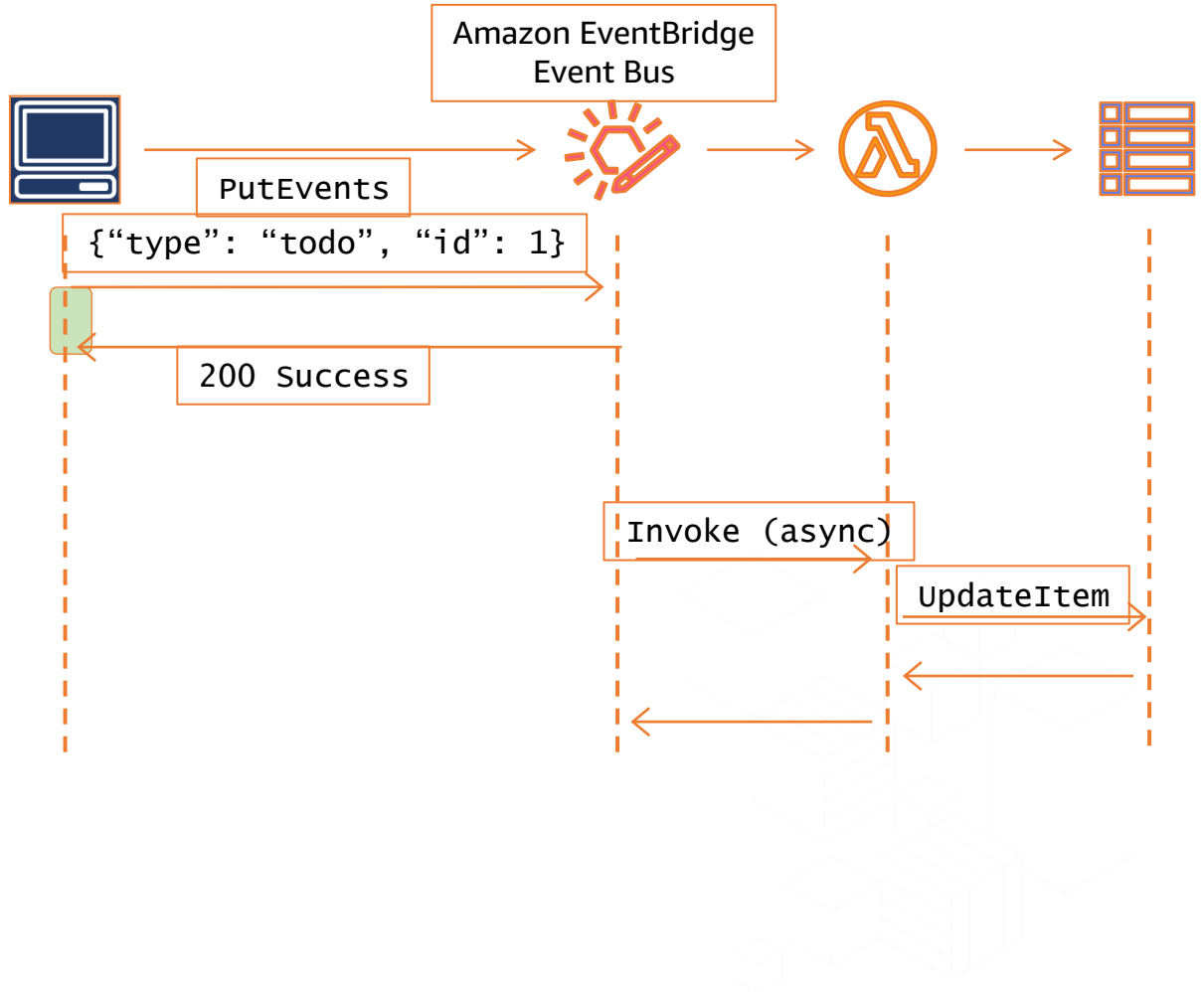
API Driven Architecture

- API defines the interface
  - e.g. REST, GraphQL
- Caller expects an immediate response
  - Response contains the result of the work
  - Generally, under 30 seconds
  - Synchronous processing
- Client must implement error handling, retry logic



Event Driven Architecture

- Event payload defines the interface
  - e.g. JSON
- Response is “message received” (or not)
  - Asynchronous processing
  - Process can be long running (beyond 30s)
  - Updating client (e.g. UI) can be more challenging
- Higher resiliency, durability
  - Driven by messaging service
- Built-in retries, configurable to use case



Why Customers are moving to event-driven applications

1

Speed & agility

Move faster. Build and deploy services independently.

2

Resiliency

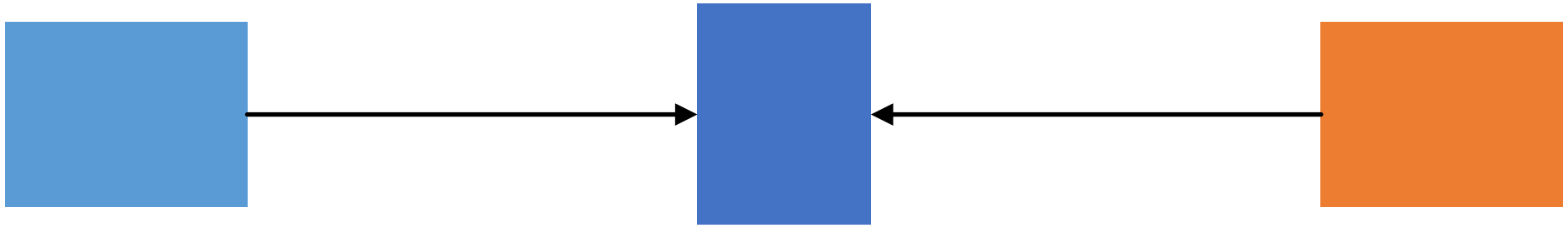
Loosely coupled systems can run and fail independently.

3

Scalability

Minimize waiting time through async and parallel processing.

EDA Components



Producer

Event producers are systems that detect a change in state or notice updates and publish those facts. Application code, a database or a time-based trigger can serve as event producers, among other things.

Event broker

An event broker is a meeting place between the producers and consumers. The broker is how events are exchanged.

Consumer

Event consumers are systems that listen for events and use them for their purposes. It's possible, and common, for a consumer to receive an event, perform some work and publish its event in response.

Event Brokers

Pull-based

Queues

- Amazon SQS
- Amazon MQ

Streams

- Kinesis
- Kafka/MSK

Push-based

Routers

- EventBridge
- SNS

EDA on AWS

Amazon SQS

Messaging

- Durable and scalable
Fully managed
Comprehensive security

Amazon SNS

Notifications

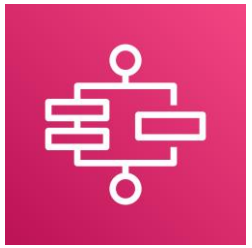
- Performance at scale
Fully managed
Enterprise-ready

Amazon EventBridge

Choreography

- Event filtering

Managed & scalable
SaaS integration



AWS
Step Functions

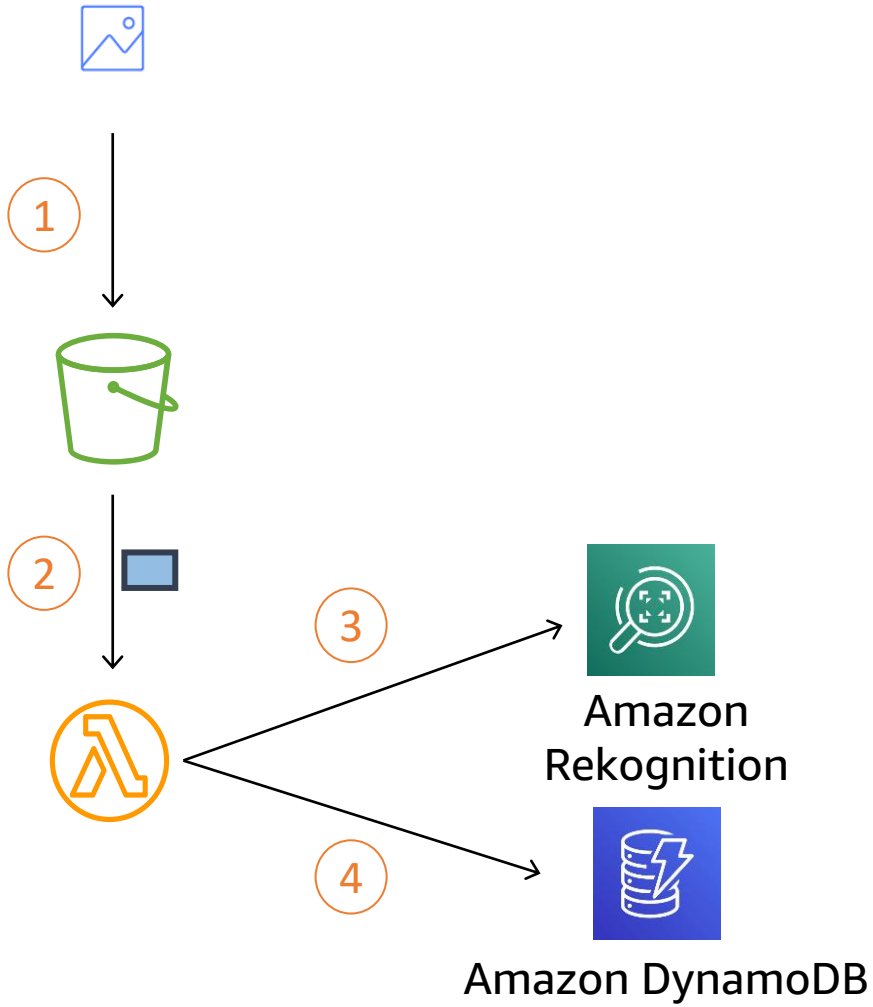
Orchestration

- Sequencing

Parallel execution
State management

Processing File Uploads

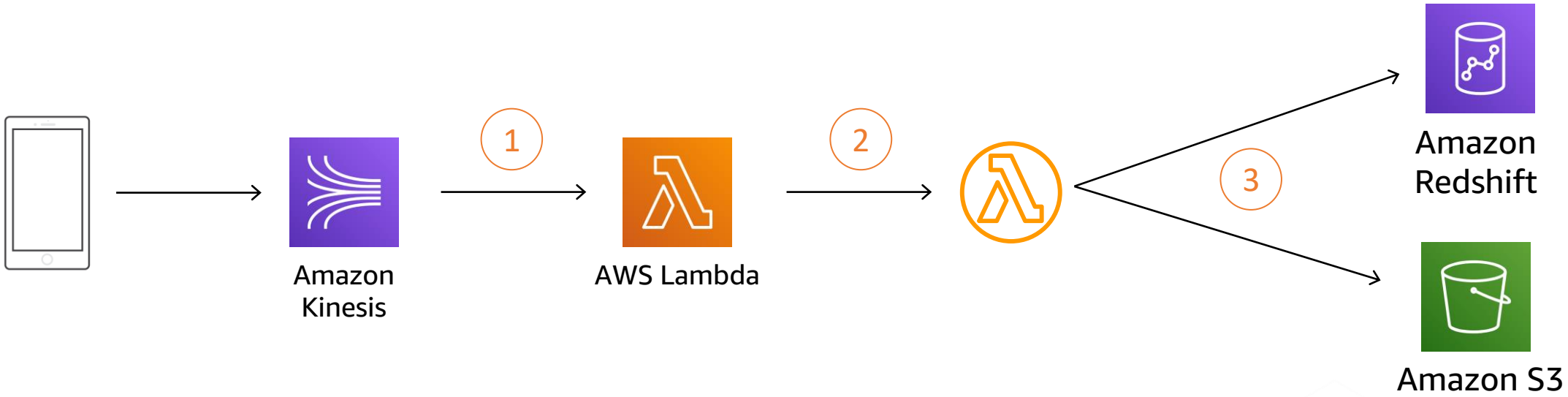
Resize photo, extract text, translate, etc.



1. Object uploaded to Amazon S3 Bucket
2. Asynchronous invoke of Lambda function, event payload includes:
   - Bucket name
   - Object key
3. Analyze photo with Amazon Rekognition
4. Store image details and results of analysis

Streaming Data ingestion and storage

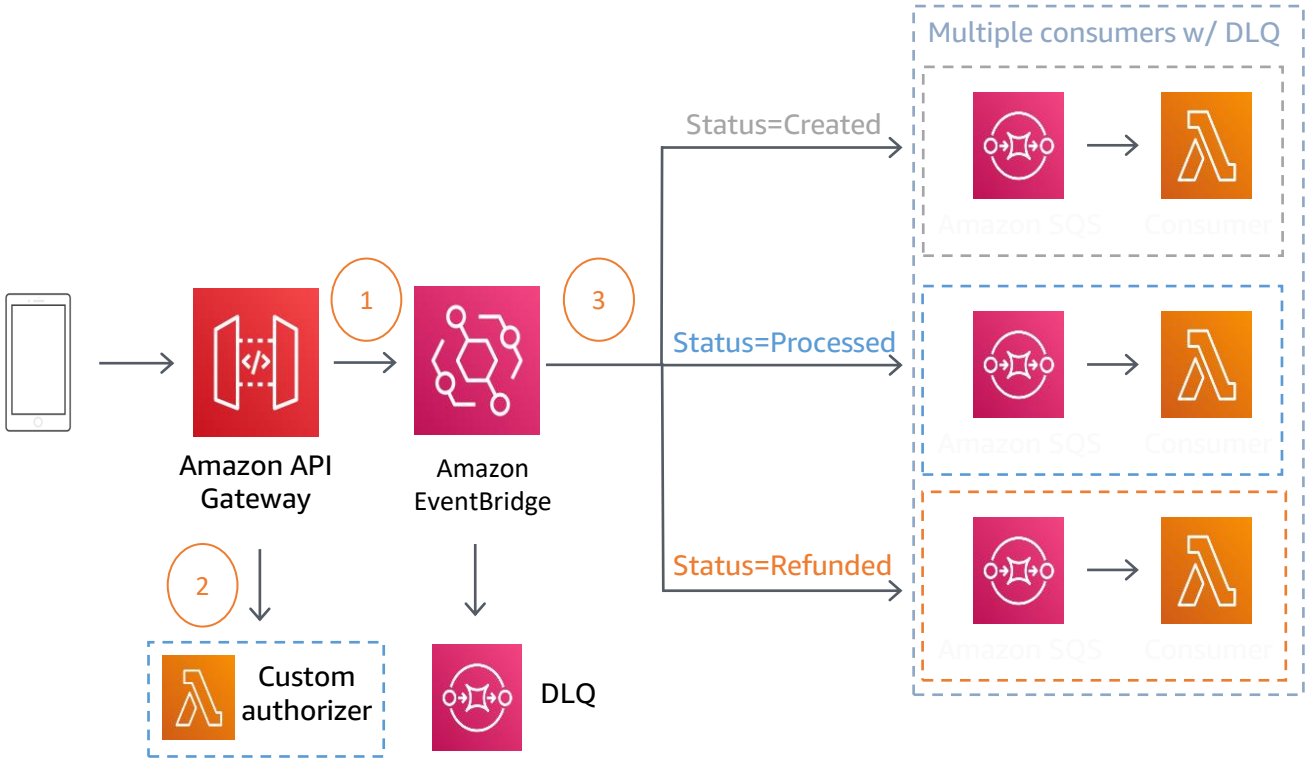
Consume , Process and Store



1. Lambda service polls Kinesis Data Stream for messages
2. Function is synchronously invoked with batches of messages
3. Function processes and/or pushes data to downstream data stores

Fan out

Push updates to multiple subscribers



1. “Storage first”: integrate API Gateway directly to EventBridge
2. Enforce authorization
3. Use routing for efficient processing