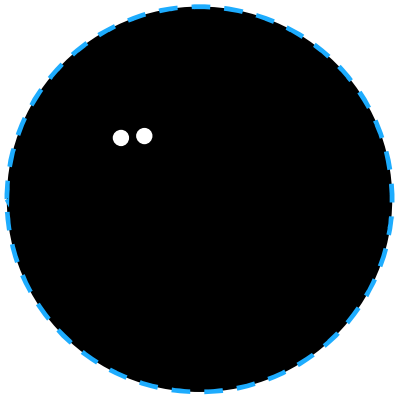
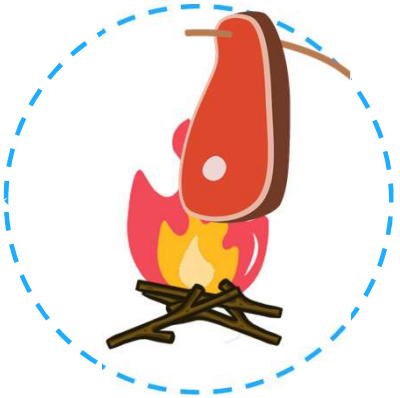
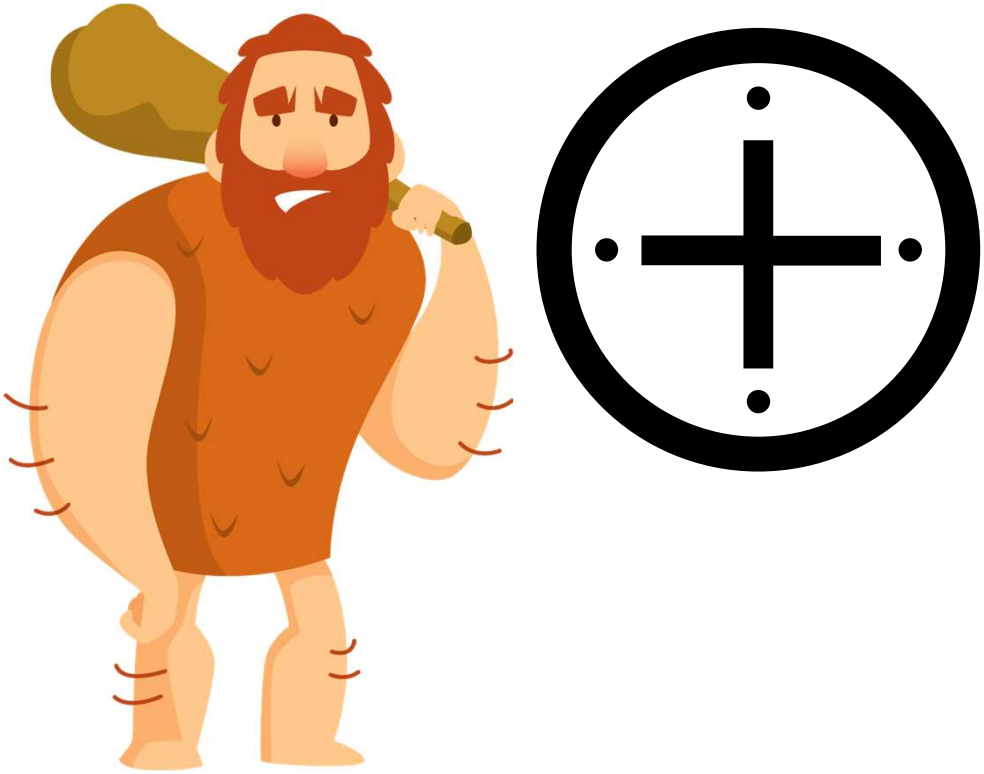
# Lambda Service Basics





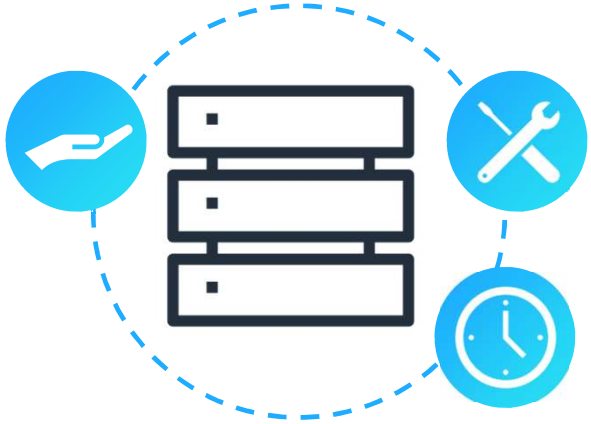


# AWS Lambda

# Bring your own Code

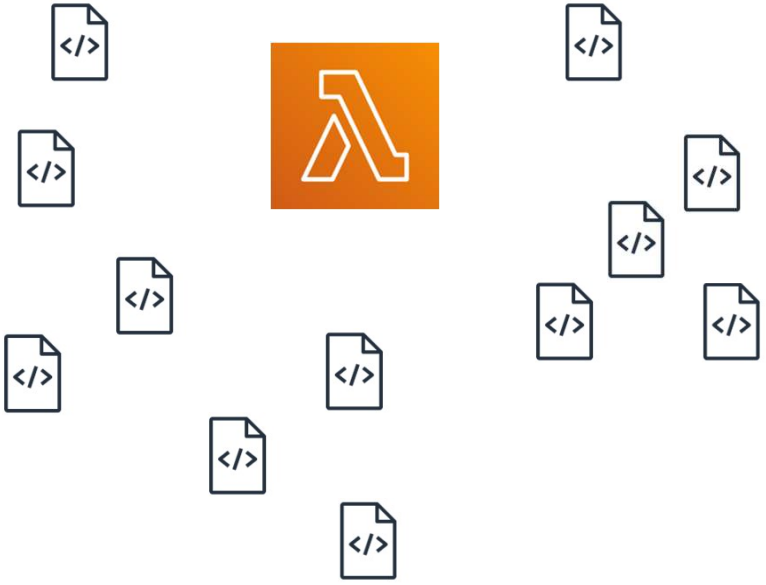
# What is Lambda?

Serverless
Compute
Service



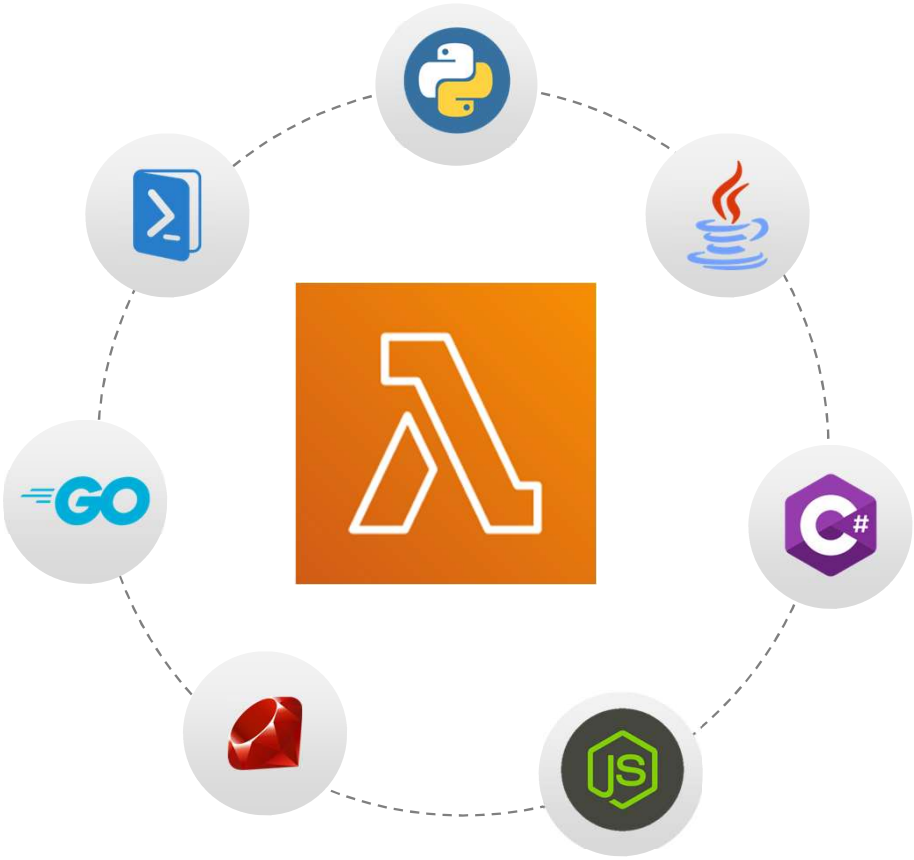
# What is Lambda?

# What is Lambda?

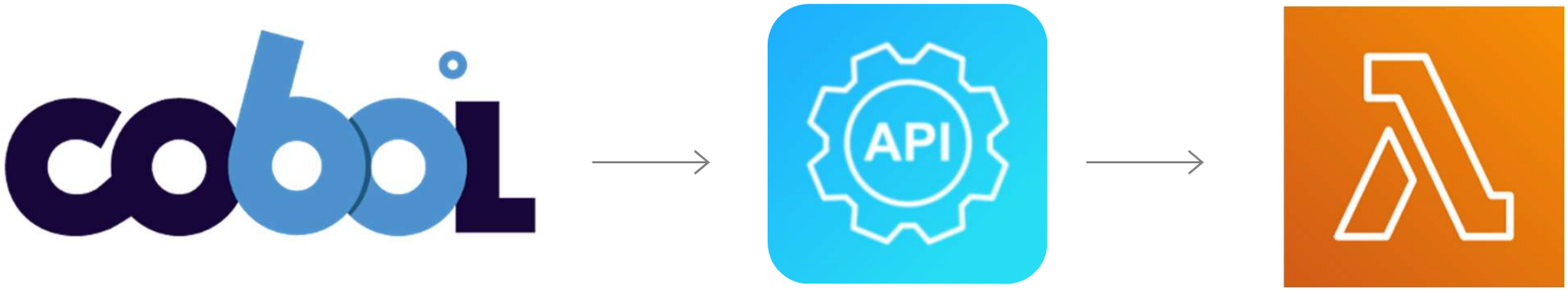


# What is Lambda?

# Supported Languages



# Supported Languages



# Supported Languages

# Summary

## What is AWS Lambda?

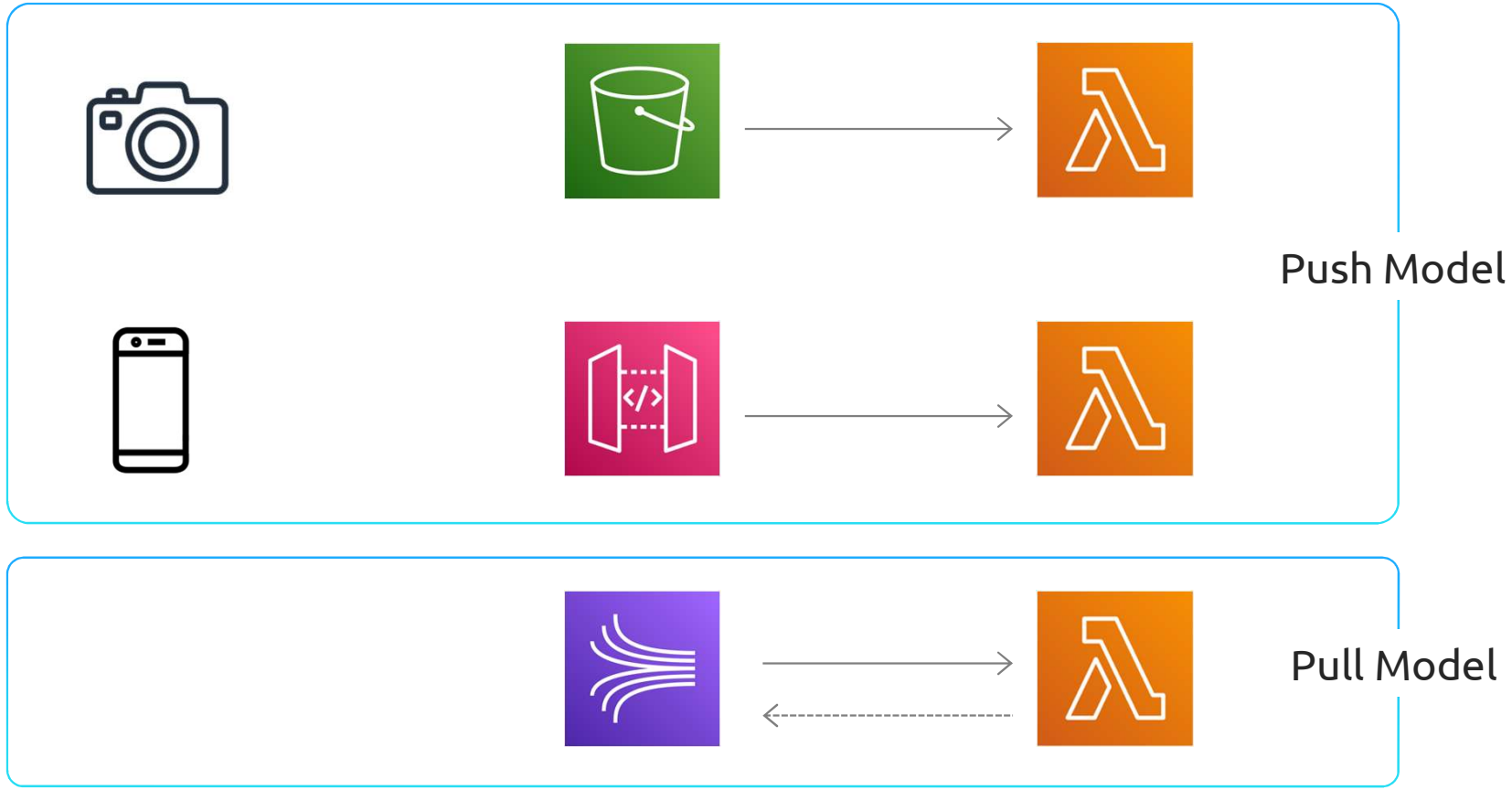
AWS Lambda is a serverless computing service that allows you to run programs without having to worry about provisioning, maintaining, or waiting for servers to be built

## AWS Lambda Features

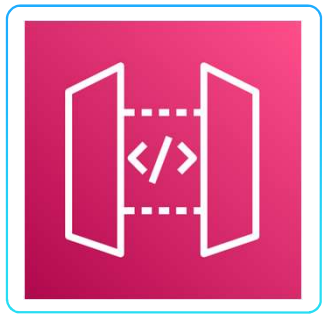
- Serverless: Run code without provisioning or maintaining a server.
- Automatic Scaling: Scale your applications automatically as per the workload.
- Pay per use: Billed per millisecond of use.
- Consistency: Performance consistency is achieved by selecting the right memory size for the function.
- Language support: AWS Lambda supports multiple programming languages.

# Event Sources

# Event Sources



# Push Model Source Types : Synchronous



API Gateway

http request

Erroneous http request
Response

http response

Retries

# Push Model Source Types : Asynchronous

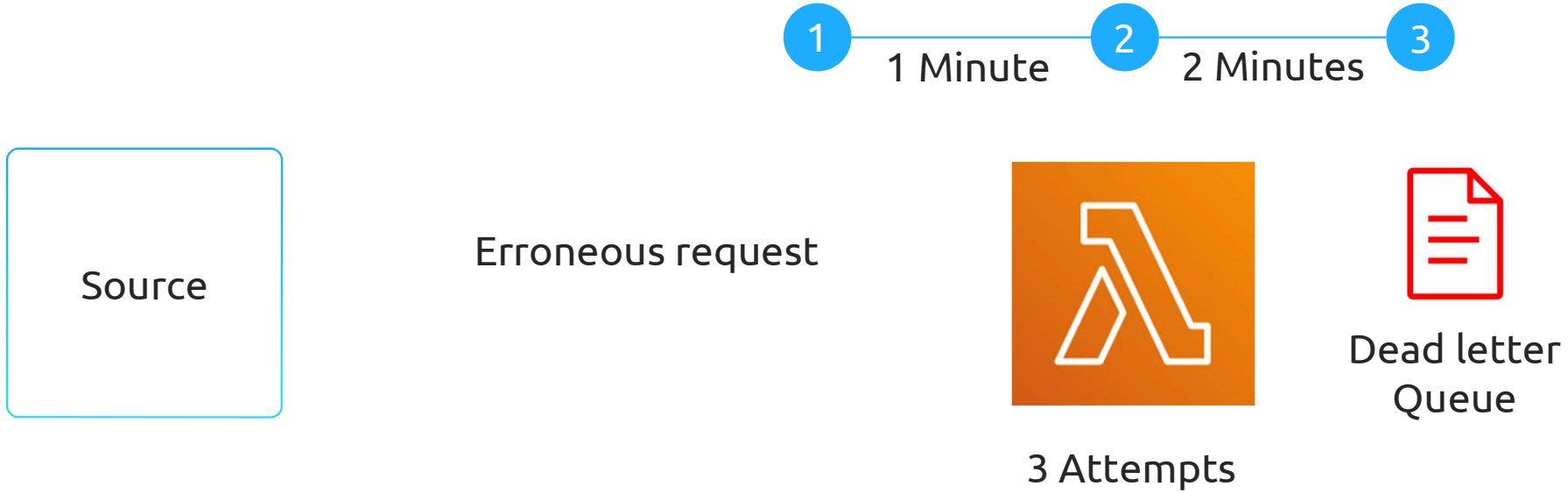
Include code to handle retries

Source

Retries

Queue

# Push Model Source Types : Asynchronous



# Push Model Source Types : Asynchronous

Source

Destinations

Eventbridge

Simple Notification Service

# Push Model Source Types

## Synchronous

Cloudformation

Cloudfront

API Gateway

Cognito

## Asynchronous

Cloudwatch

Eventbridge

S3

SNS

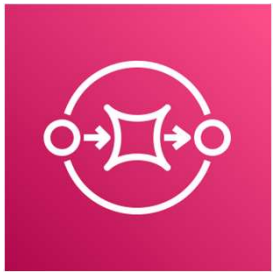
# Pull Model Source Types

## Streams

Dynamo DB

Kinesis Data Stream

## Queues

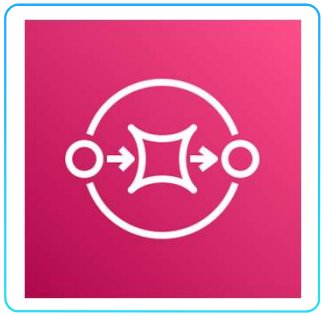


Amazon Simple Queueing Service

# Pull Model Source Types : Streams

Streams

# Pull Model Source Types : Queues



Amazon Simple Queueing Service

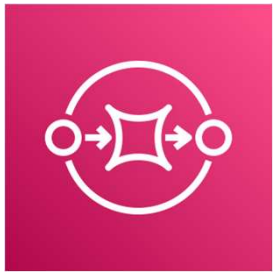
Queue

# Pull Model Source Types

Lambda Function

# Pull Model Source Types

## Queues



Amazon Simple Queueing Service

## Streams

Dynamo DB

Kinesis Data Stream

# Pull Model Source Types : Streams

Streams

Retry

# Pull Model Source Types : Queues

# Summary

## Event Source

AWS Lambda Event Source is any AWS Service, custom application, a stream of data or queue which triggers a Lambda function to run.

## Lambda Event Sources Type

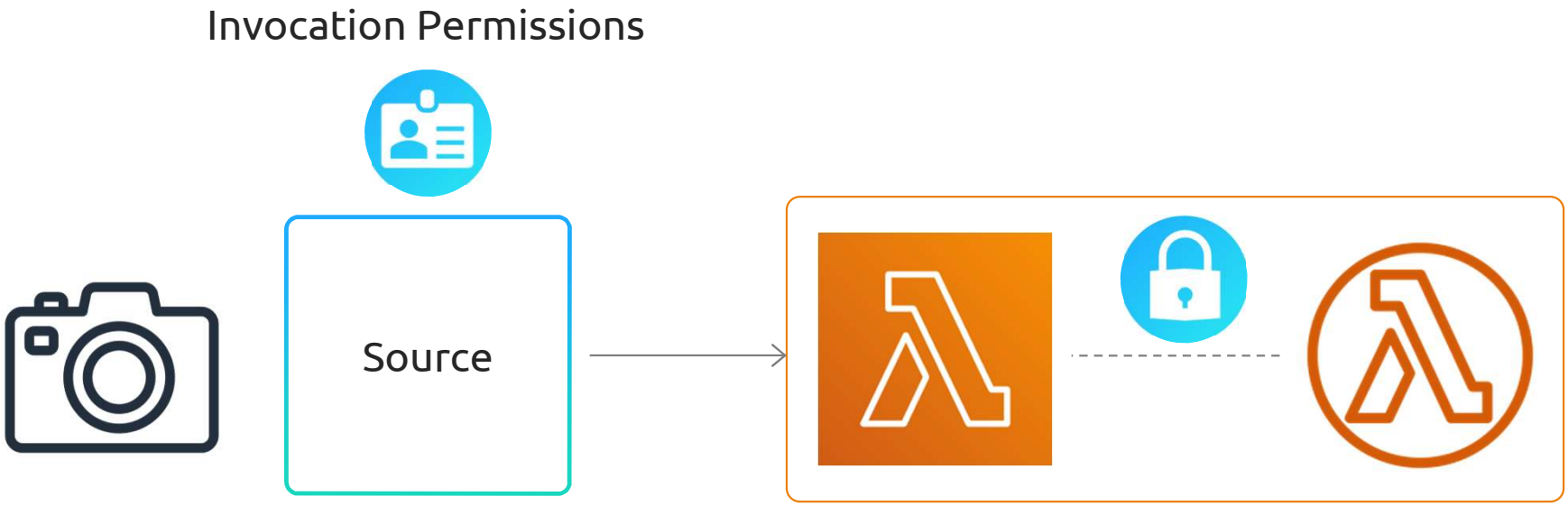
- Push-based model
  - Synchronous
  - Asynchronous
- Pull-based model
  - Stream
  - Queue

# Summary

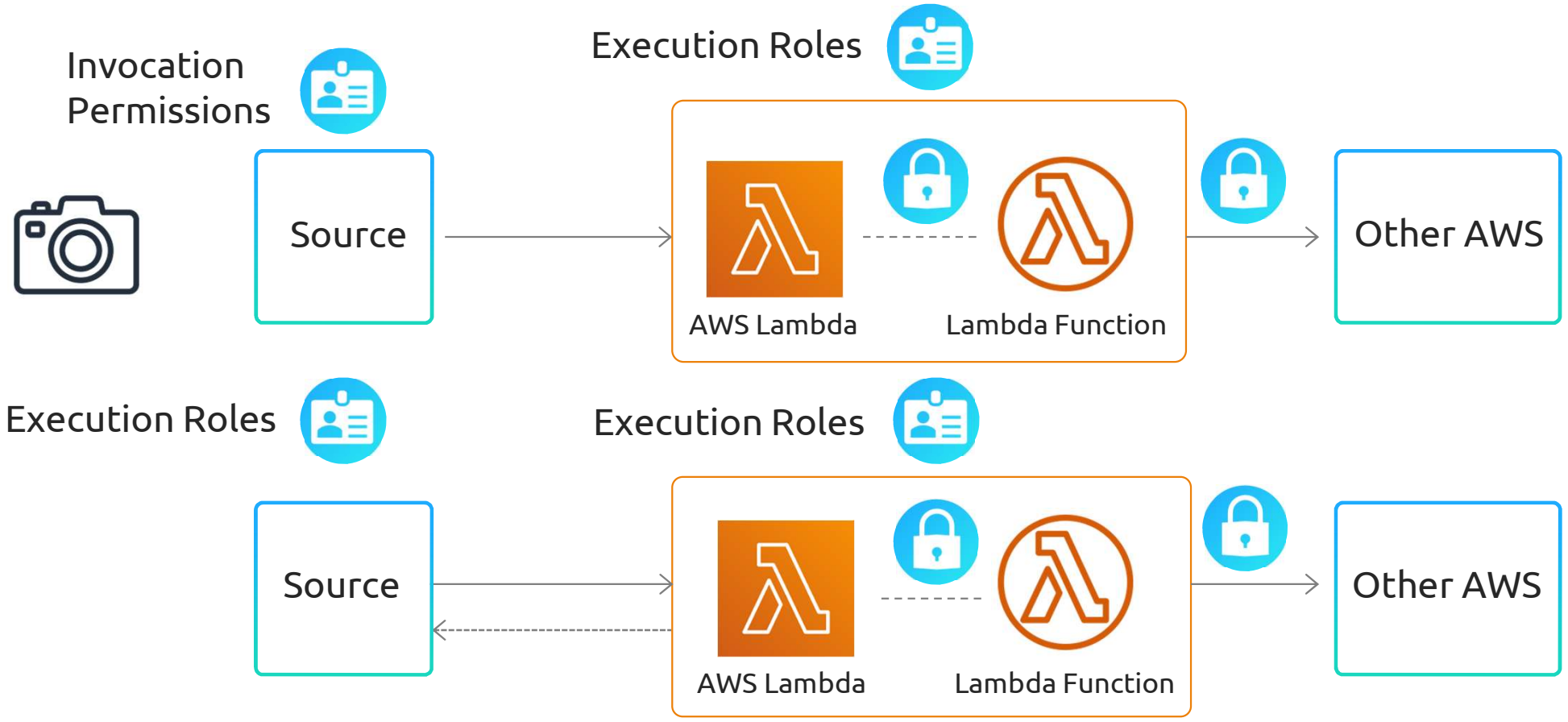
- Push-based model: Other service directly triggers Lambda and tells it to activate when something happens.
  - Synchronous: Lamba returns a response back to the event source.
  - Asynchronous: For asynchronous invocations, Lambda does handle retries. After Lambda is invoked by the source, it will first place the event into a queue and immediately sends a success response back to the source.
- Pull-based model: Information is flowing through a stream or put in a queue that Lambda periodically polls, and it then goes into action when certain events are detected .
  - Stream: Lambda will stop polling while it retries the message.
  - Queue: Lambda will return the message to the queue if the invocation fails or times out but will also keep retrying until it’s either successful or expires.

# Access Permissions

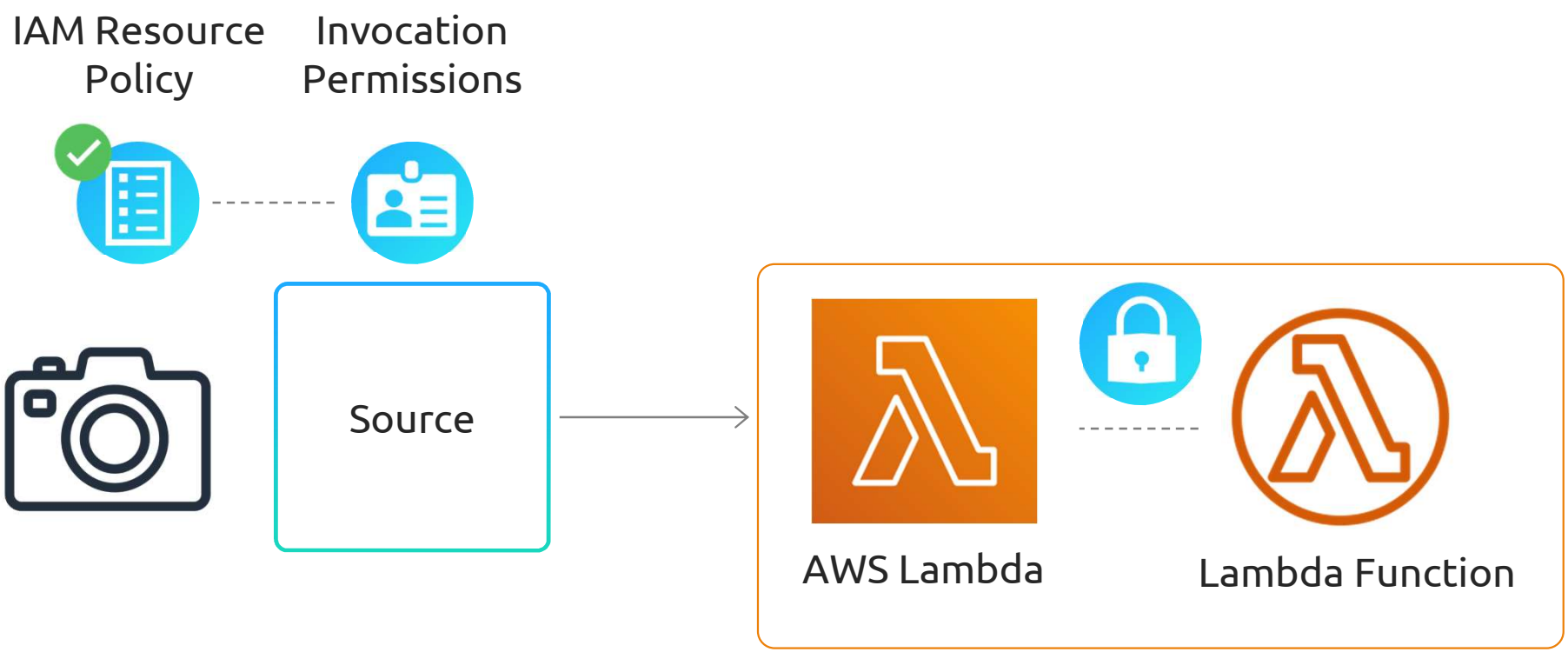
# Access Permissions



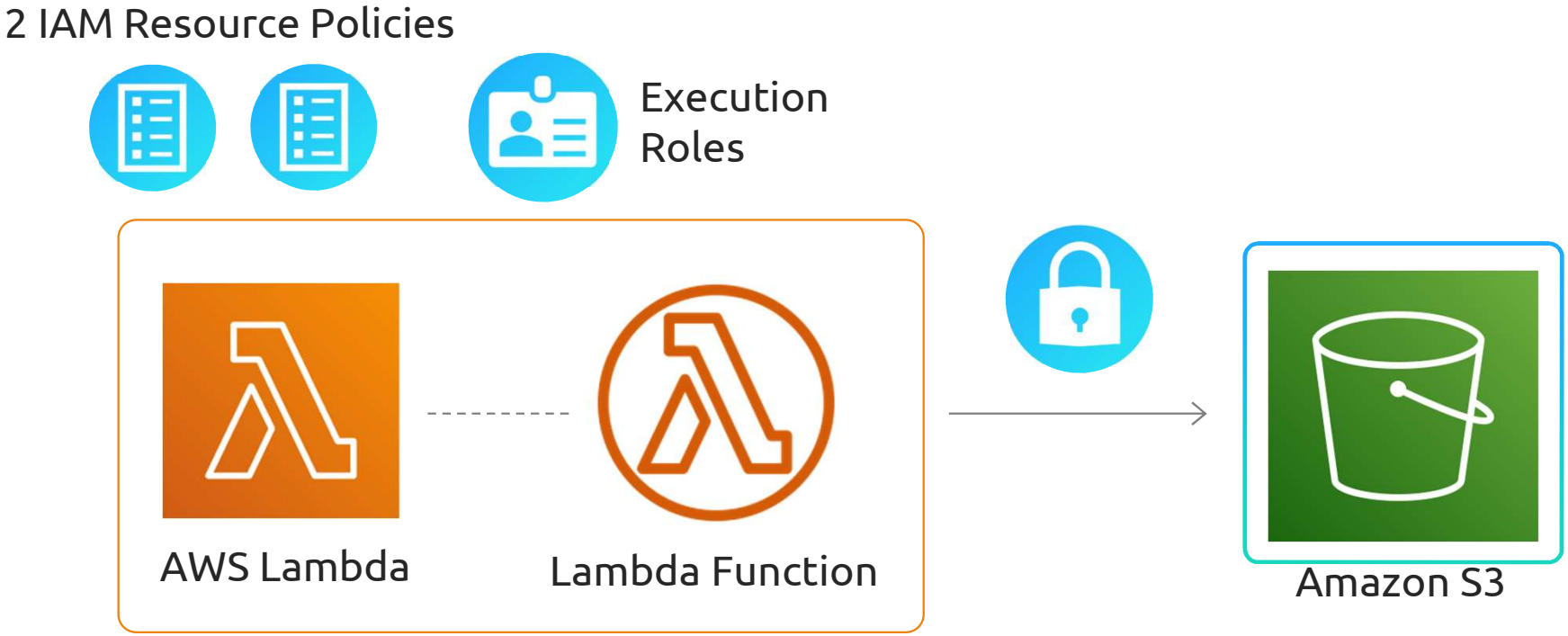
# Access Permissions



# Invocation Permissions



# Execution Roles



# IAM Policy

Terminal

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ExampleSourceFunctionArn",
            "Effect": "Allow",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::lambda_bucket/*",
            "Condition": {
                "ArnEquals": {
                    "lambda:SourceFunctionArn":
"arn:aws:lambda:us-east-1:123456789012:function:source_lambda"
                }
            }
        }
    ]
}
```

# Trust Policy

Terminal

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

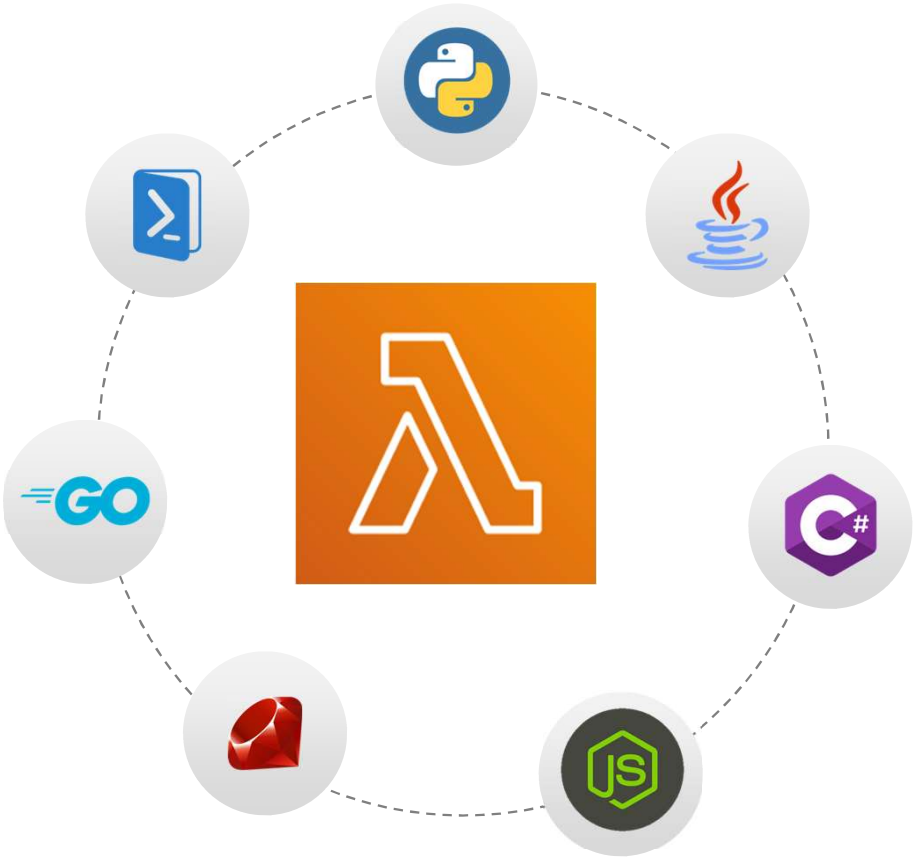
# Summary

## Access Permissions

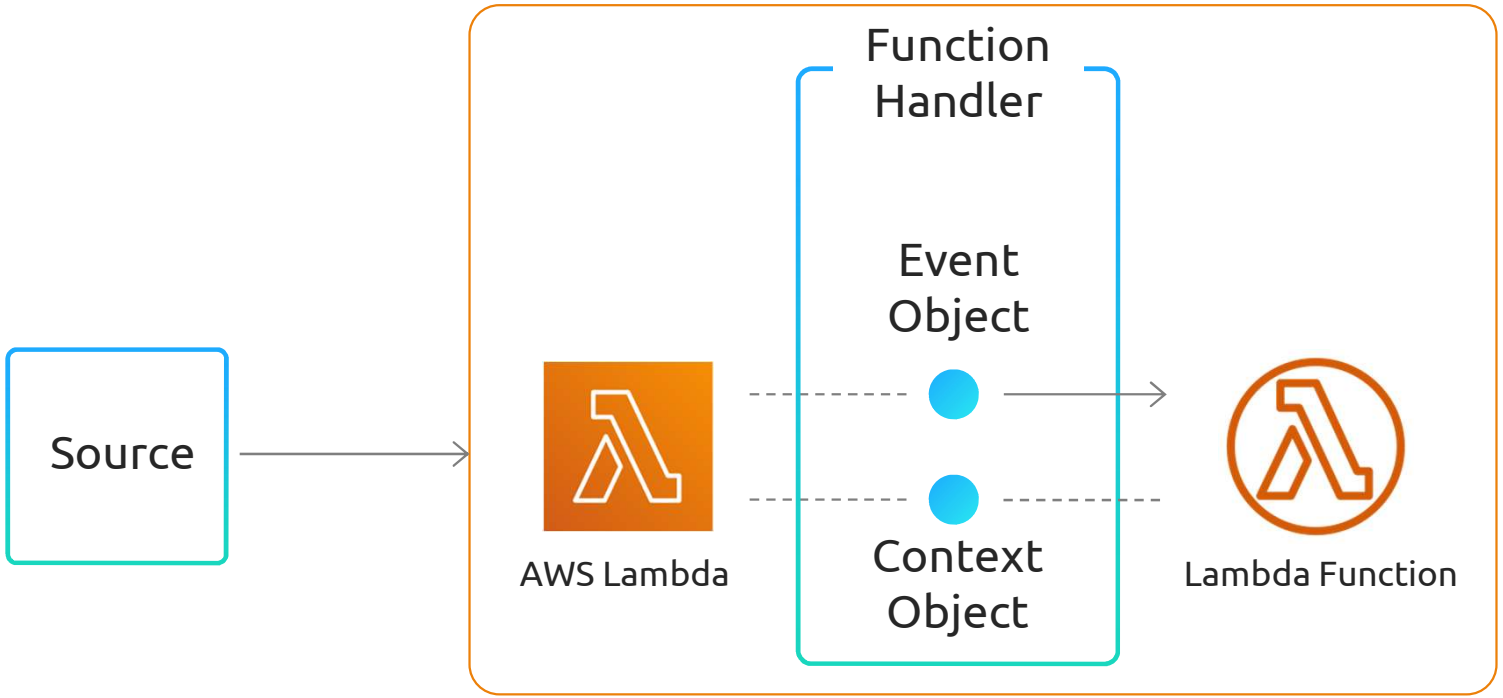
- Security is crucial when dealing with Lambda because of its power to run code and take actions that affect other AWS services.
- Two types of security permissions are needed for Lambda: Invocation permissions and Execution roles.
- Invocation permissions are only needed for push event sources and are granted through an IAM resource policy automatically created when configuring an AWS service as an event source.
- Execution roles grant Lambda permissions to interact with other AWS services. They include an IAM policy defining what Lambda is allowed to do and a Trust policy allowing Lambda to assume the execution role and perform actions on the other service.
- Adding the execution role to your Lambda function is the final step in granting permissions and policies.

# Functions

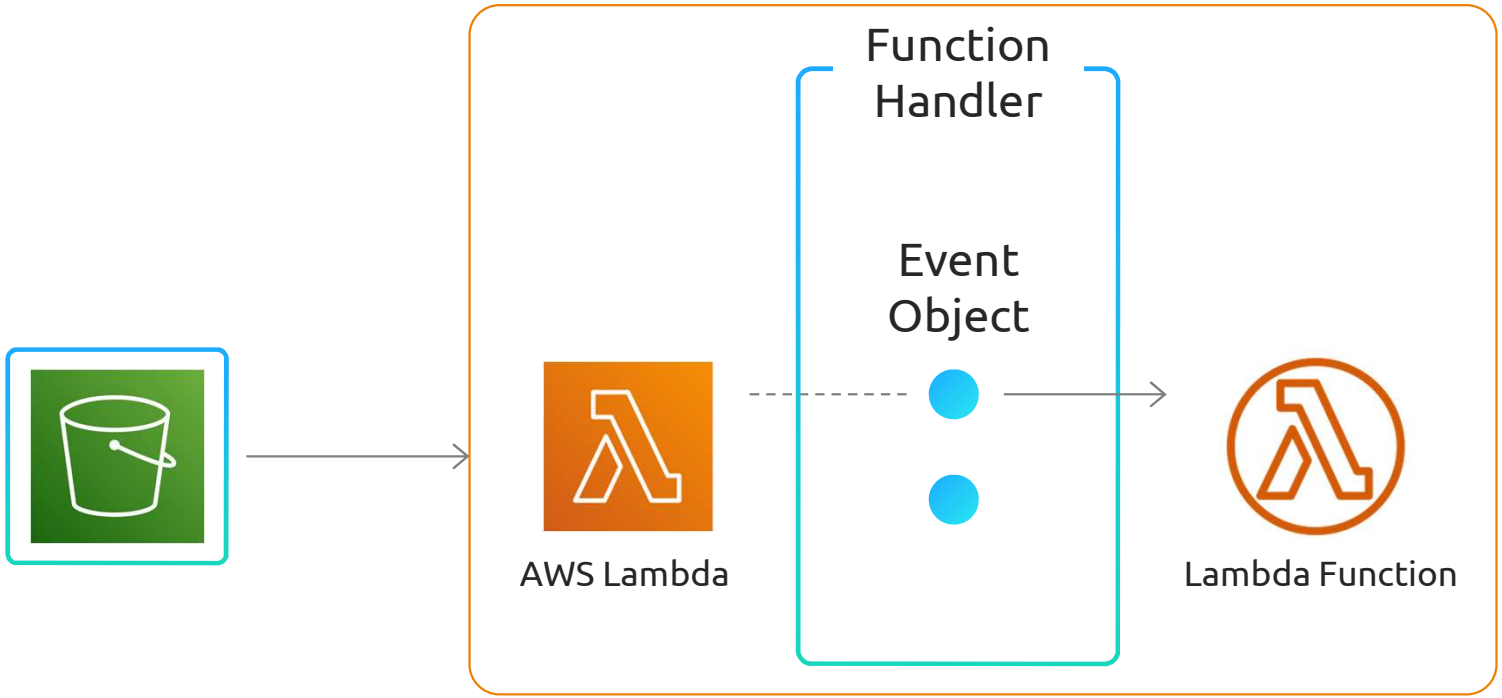
# Functions



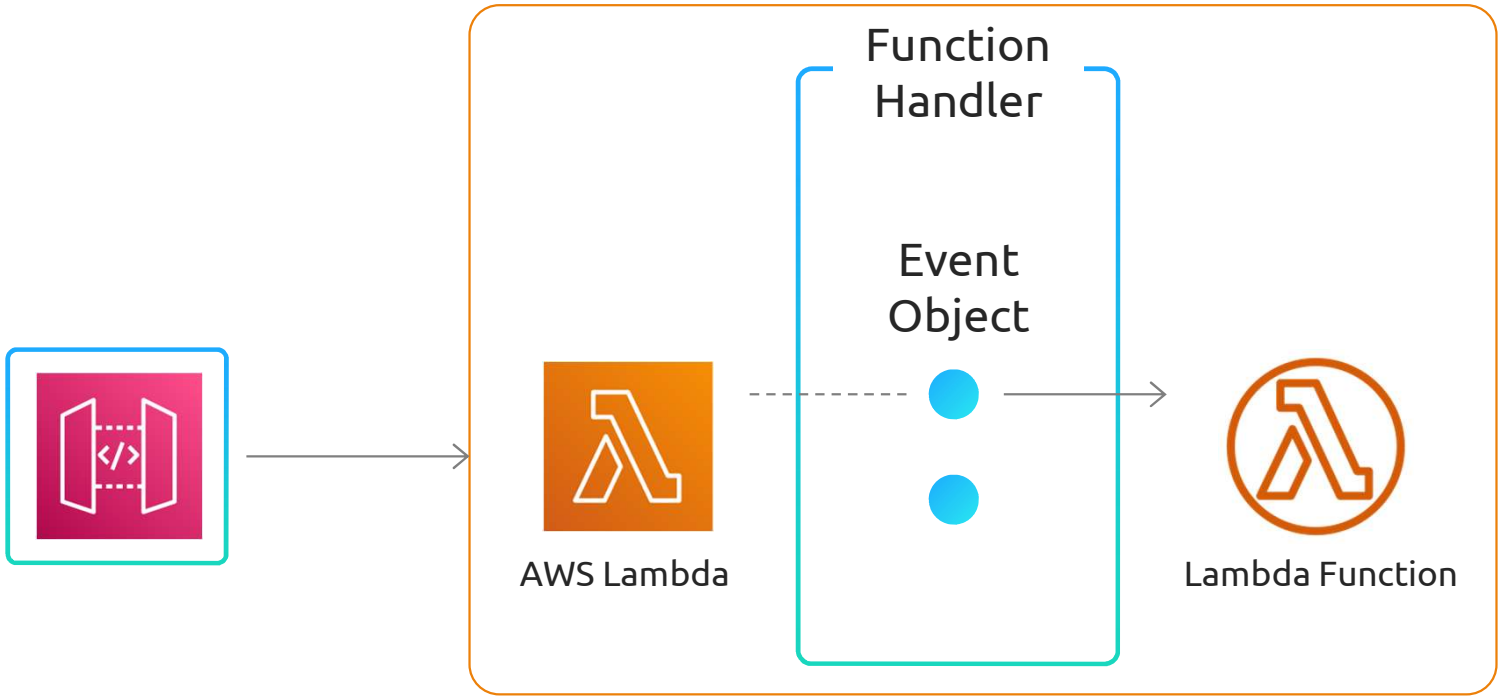
# Functions



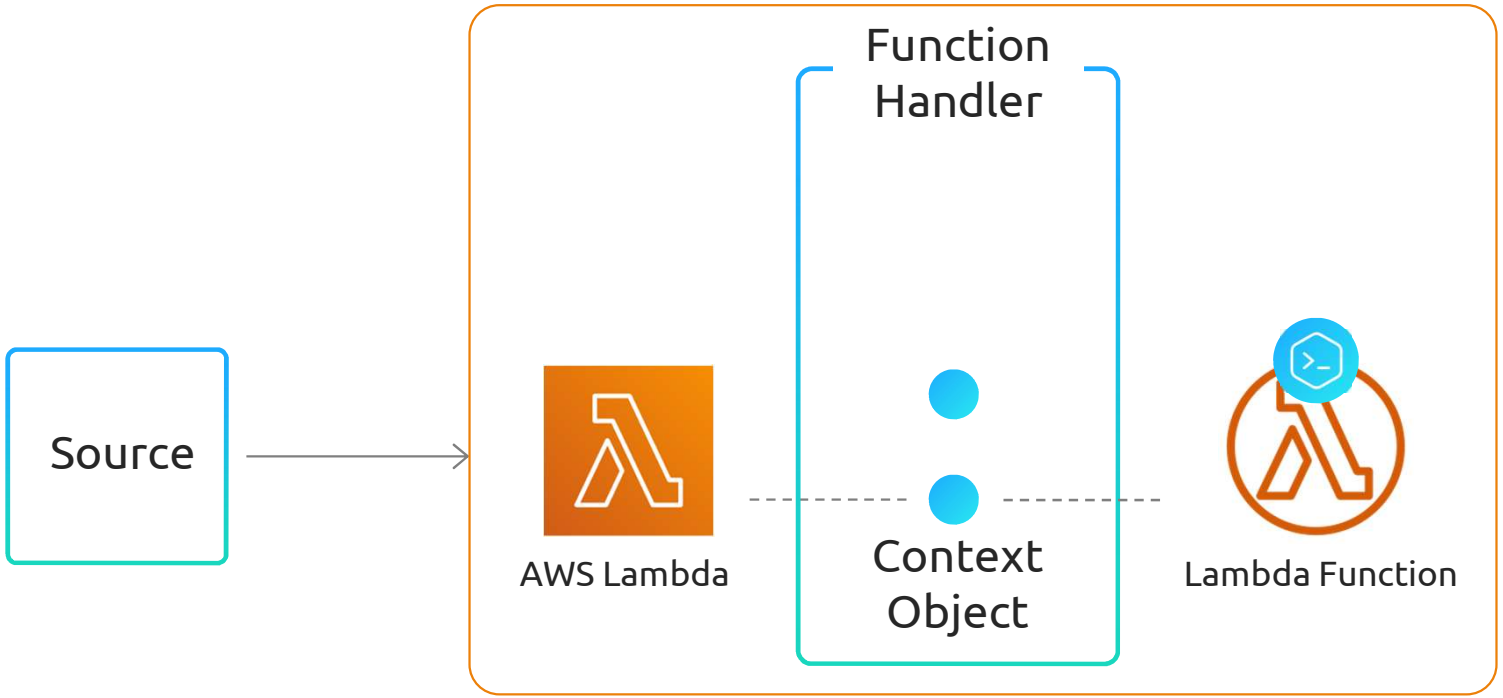
# Functions



# Functions



# Functions



lambda_function.py

```python
def lambda_handler(event, context):
    message = 'Hello {} {}!'.format(event['first_name'], event['last_name'])
    return { 'message' : message }
```

## AWS Lambda Developers Guide:

https://docs.aws.amazon.com/lambda/latest/dg/welcome.html

# Functions

## Best Practices:

https://docs.aws.amazon.com/lambda/latest/dg/best-practices.html

- Writing Functions
- Testing
- Performance
- Using Lambda with specific services
- Working with streams
- And more...

# Summary

## AWS Lambda Functions

- Lambda contains a Function that executes the programming code in one of the supported programming languages.
- The user can bring the code without changing its bulk, but a function handler must be included.
- The function handler has two objects: the event object and the context object.
- Event Object: Allows the event source to pass information to the Lambda function.
- Context Object: Contains information specific to the runtime environment of the programming language used and allows communication within the function.
- The configuration of the code specifics depends on the programming language used.

# Pricing

# Pricing

## Start Free

### Monthly Free Tier:

1 Million Requests

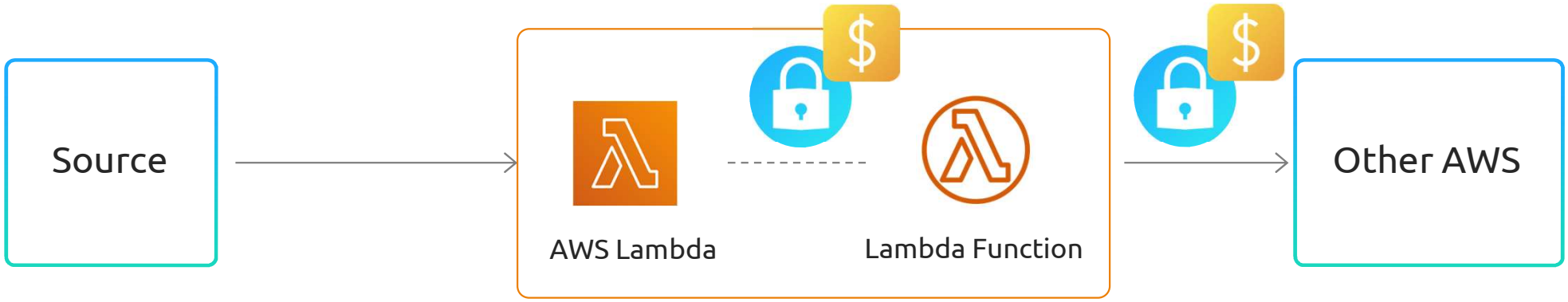
400, 000 Gigabit seconds

# Pricing Components

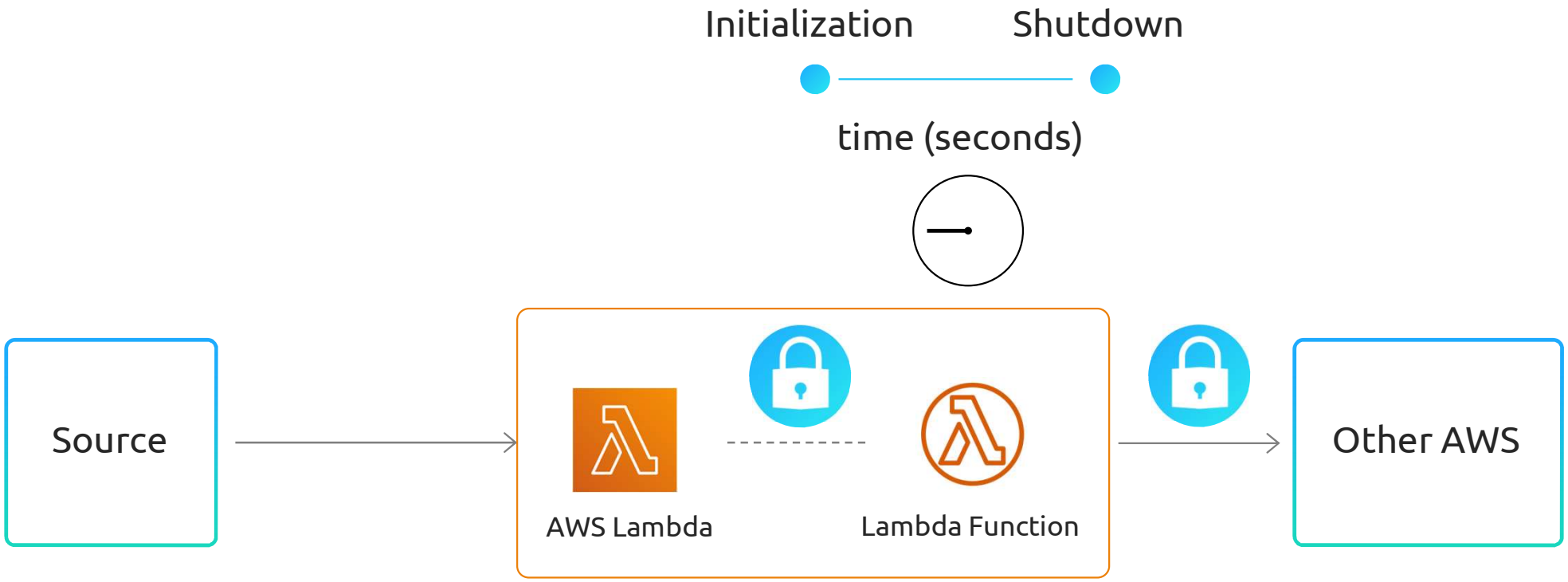
## Parts of Lambda Cost

1. Number of Requests
2. Gigabit Seconds (Amount of Time X Amount of Resources)

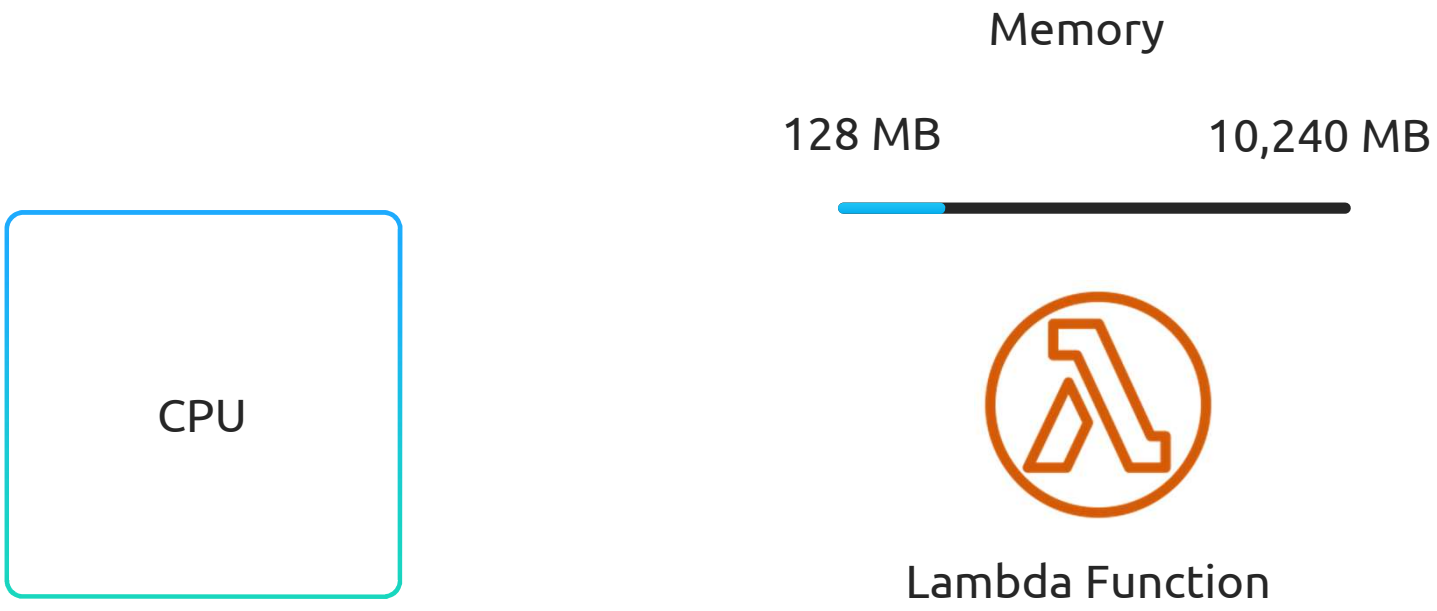
# Lambda Requests



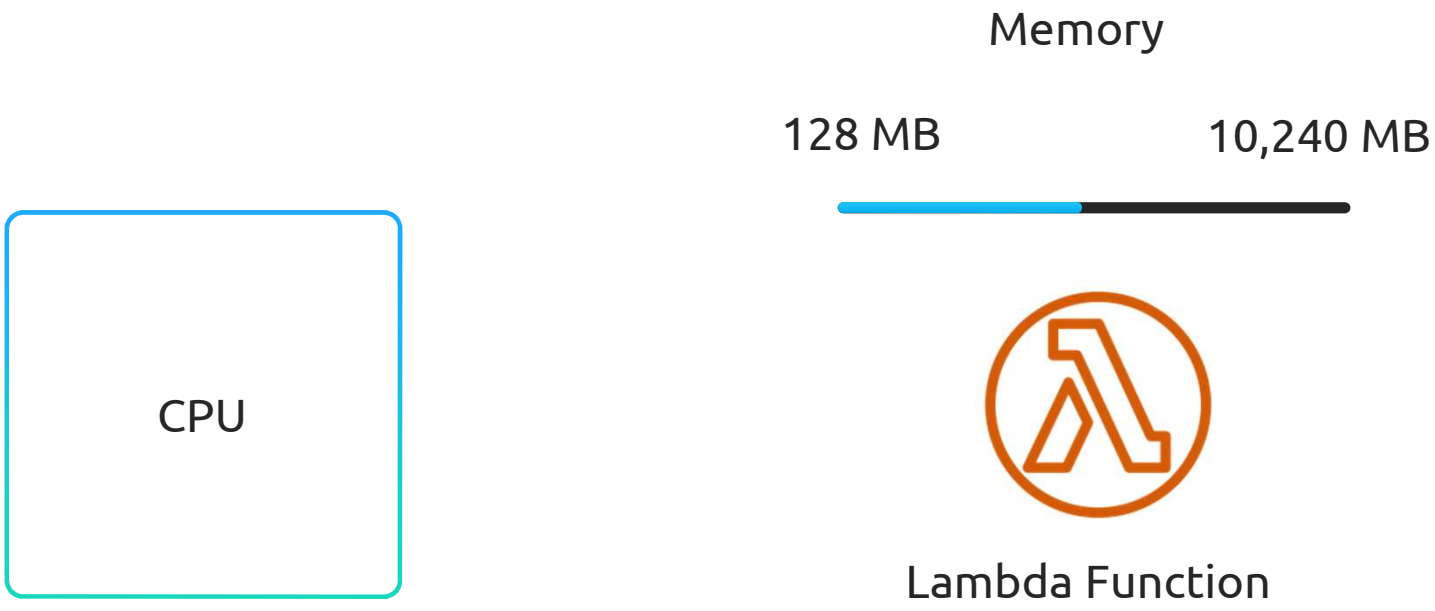
# Gigabit Seconds



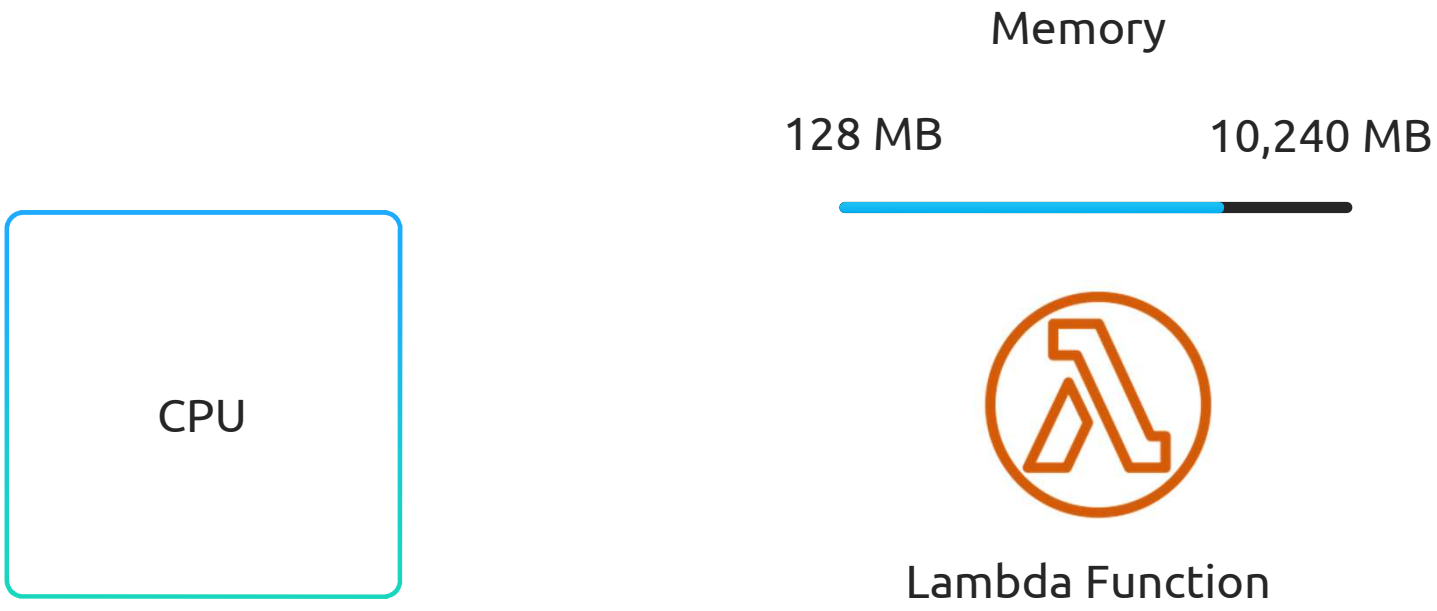
# Gigabit Seconds



# Gigabit Seconds



# Gigabit Seconds



# Gigabit Seconds

CPU

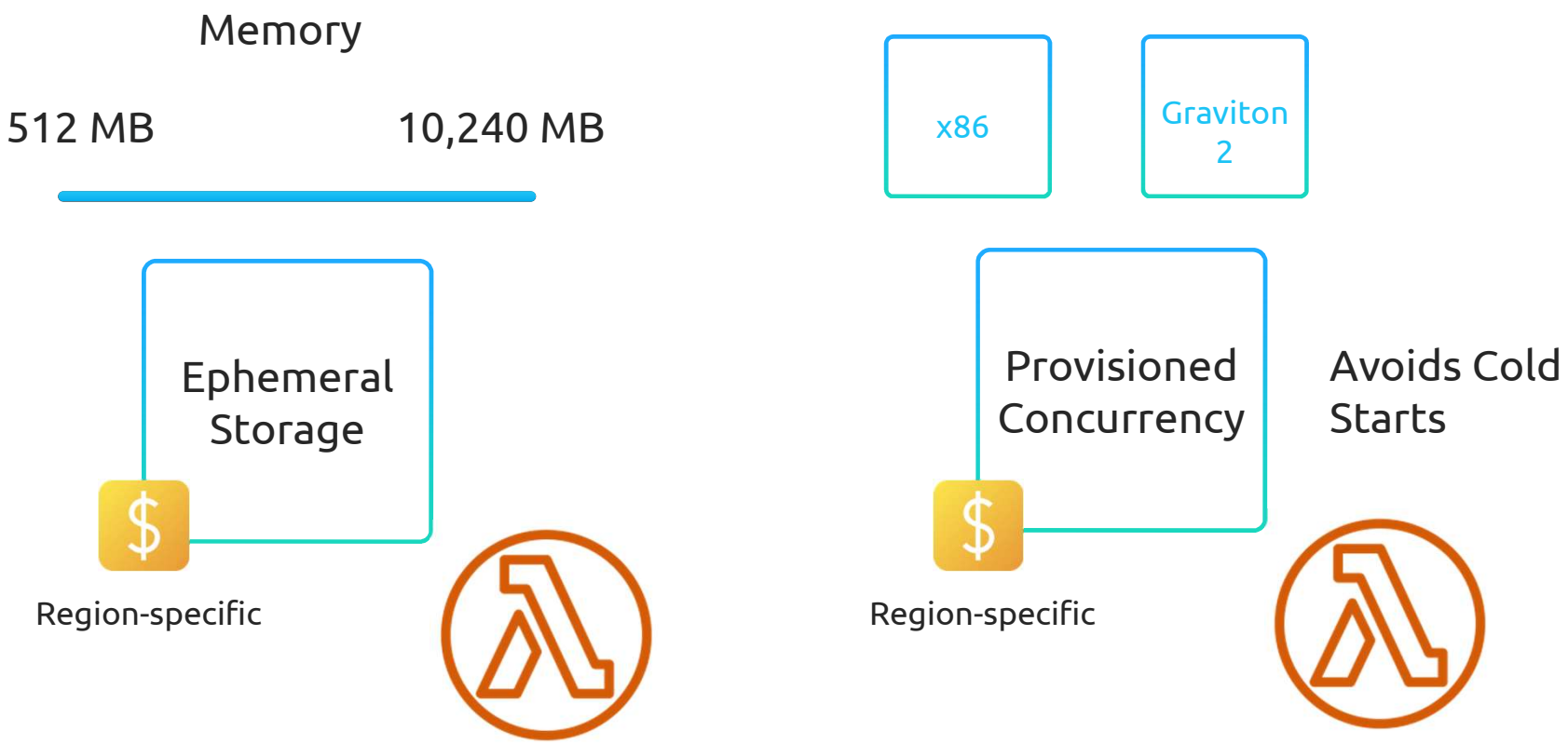
Memory

128 MB 10,240 MB

Lambda Function

# Pricing Components

# Pricing Components



# Lambda Pricing Example

AWS Calculator Link:

https://calculator.aws

# Summary

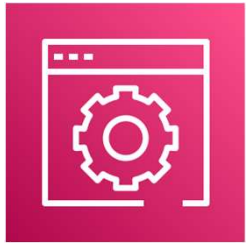
## AWS Lambda Pricing

- Lambda's free tier allows you to use 1 million requests and 400,000GB seconds per month.
- The main components of Lambda pricing are the number of requests and Gigabit Seconds.
- A Lambda request starts each time it executes the function in response to an event trigger.
- Gigabit seconds are the amount of time your function runs multiplied by the amount of memory it uses.
- The amount of memory you allocate to the function determines the amount of CPU power allocated by Lambda.
- There are additional costs for including ephemeral storage and provisioned concurrency to your Lambda functions.

# Prerequisites

# Prerequisites

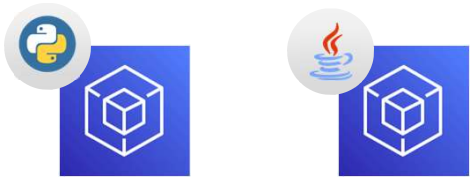
## Ways to configure Lambda



AWS
Management Console

AWS Command Line
Interface (CLI)

Software Development
Kits (SDKs)



# Prerequisites

## Ways to configure Lambda

Infrastructure
as Code

**JSON** or **YAML** File

AWS
Cloudformation

AWS
Serverless Application Model

# Prerequisites

https://aws.amazon.com/console/

# Prerequisites

AWS Command Line Interface (CLI)

# Prerequisites

AWS Command Line Interface (CLI)

Download the application at:

https://aws.amazon.com/console/

Supported on:

Windows
MacOS
Linux

# Prerequisites

AWS Command Line Interface (CLI)

Download the application at:

https://aws.amazon.com/cli

Supported on:

Windows
MacOS
Linux

# Prerequisites

AWS Command Line Interface (CLI)

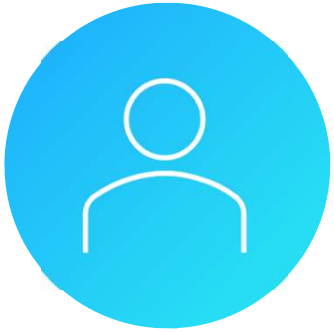
Download the application at:

https://aws.amazon.com/console/

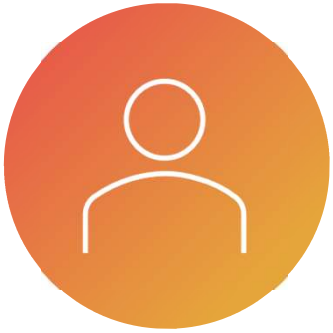
Supported on:

Windows
MacOS
Linux

# Prerequisites



Root User Account



Administrative User Account

# Summary

## Ways to configure Lambda

AWS
Management Console

AWS Command Line
Interface (CLI)

Software Development
Kits (SDKs)

# Create Basic Function Demo

## Environment

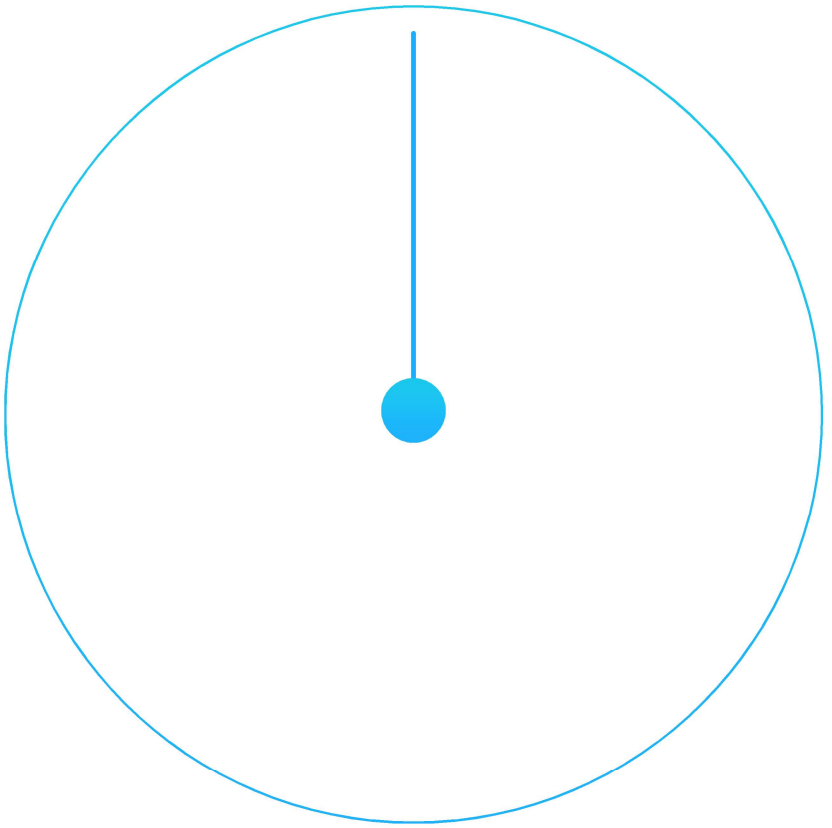
AWS CLI

Visual Studio Code
(or equivalent)


AWS Console

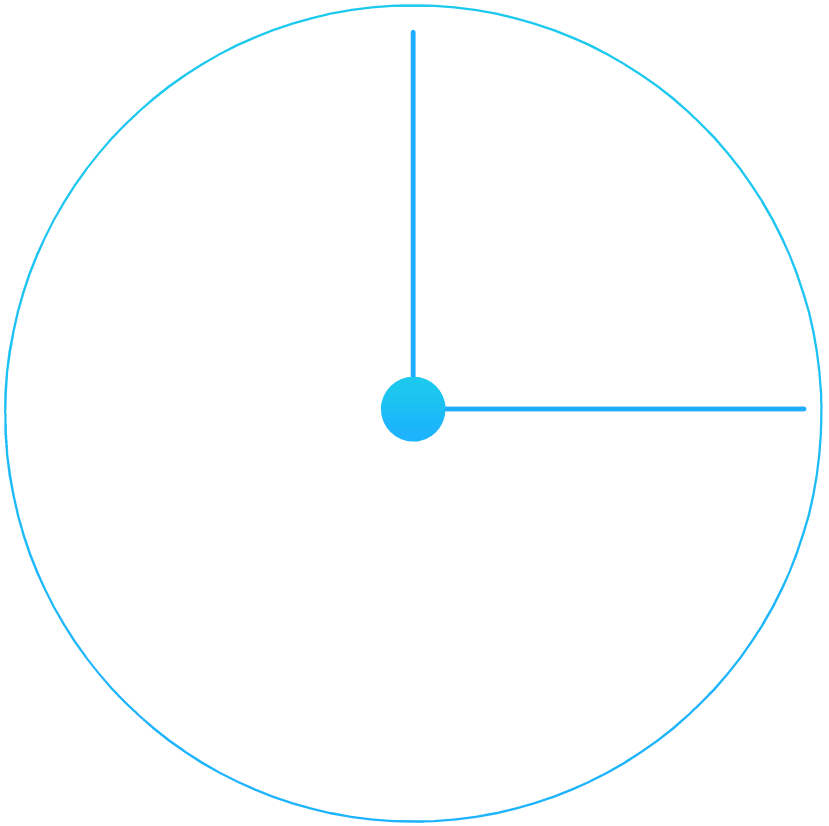
# Limitations

# Limitations



# Limitations

Can only run for 15 minutes



# Limitations

Can only run for 15 minutes

10 GB RAM Limit

10 GB Storage Limit

# Limitations

- Can only run for 15 minutes
- 10 GB RAM Limit
- 10 GB Storage Limit
- 1000 Concurrent Execution

1000

# Limitations

- Can only run for 15 minutes
- 10 GB RAM Limit
- 10 GB Storage Limit
- 1000 Concurrent Execution

3000

Region-specific

# Limitations

Can only run for 15 minutes

10 GB RAM Limit

10 GB Storage Limit

1000 Concurrent Execution

For more information on Limits and Restrictions visit:

https://docs.aws.amazon.com/lambda/latest/dg/gettingstarted-limits.html

# Summary

## Limitations

- Lambda function can only run for 15 minutes maximum.
- Functions must adhere to a 10 GB RAM and 10 GB storage limit.
- Ephemeral storage is the temporary space for functions to read and write data.
- Lambda automatically scales based on load and can handle up to 1000 concurrent executions running at the same time.
- There is a burst feature that allows temporary increase in concurrent executions up to 3000 in some regions.
- Requesting an increase to the default 1000 concurrent execution limit is possible but only granted by AWS if required and based on your use case.

# Monitoring

# Monitoring

Amazon
Cloudwatch

X-Ray
Services

# Amazon Cloudwatch

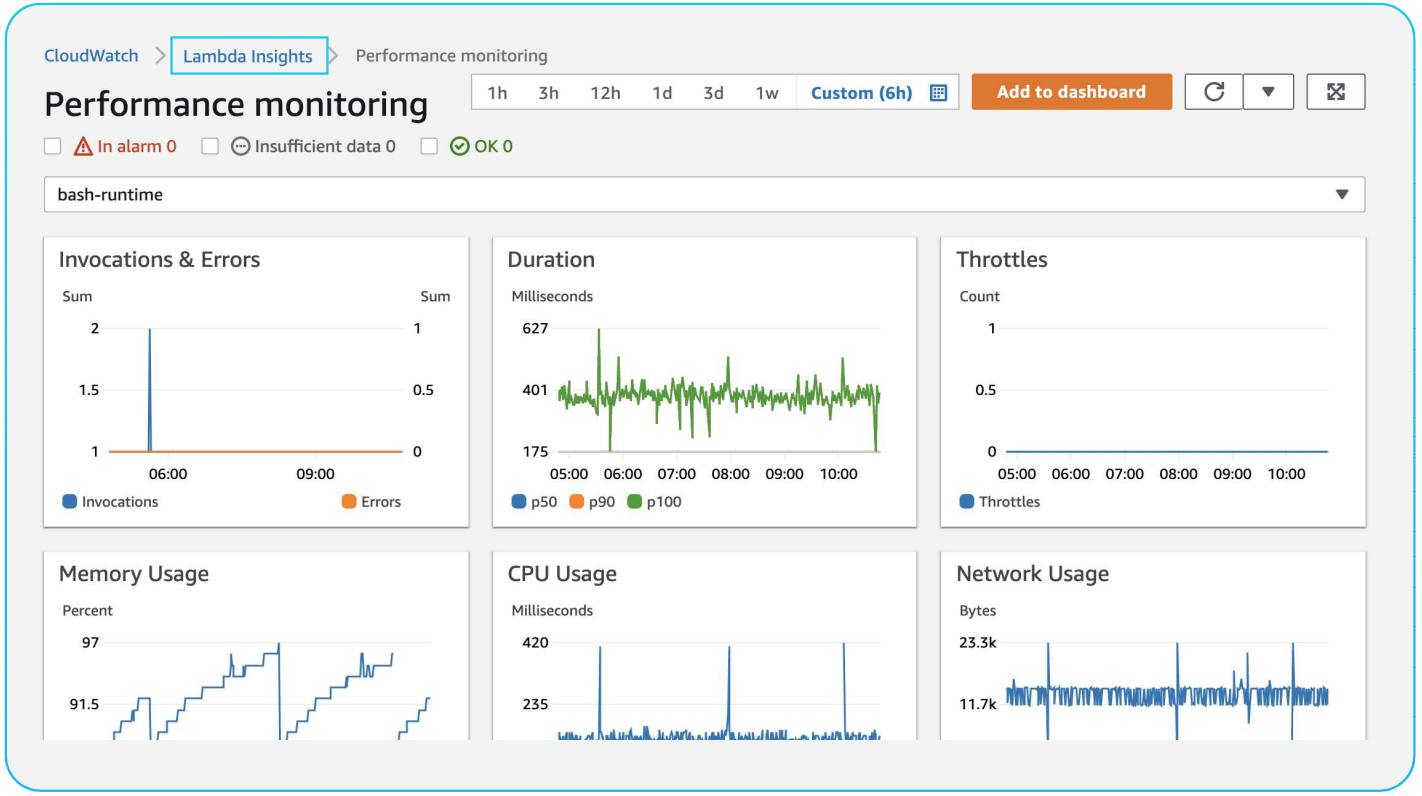
## Automatically monitors:

- Number of requests
- Duration per request
- Number of requests resulting to error

## Additional Metrics Available:

- Throttles due to concurrency limit
- Delay pulling data from stream sources referred to as “Iterator Age”
- Number of events in dead letter queue
- Detailed concurrent execution metrics

# Lambda Insights



# AWS X-ray

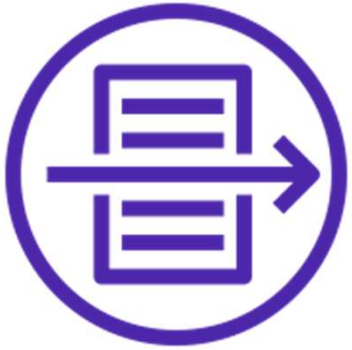
## Monitoring and troubleshooting tool

- Visual mapping
- Identify performance bottlenecks and errors
- Trace Lambda function path

# Monitoring

Transmission Control Protocol (TCP)

Internet Protocol (IP)



Flow Logs

# Monitoring

# Summary

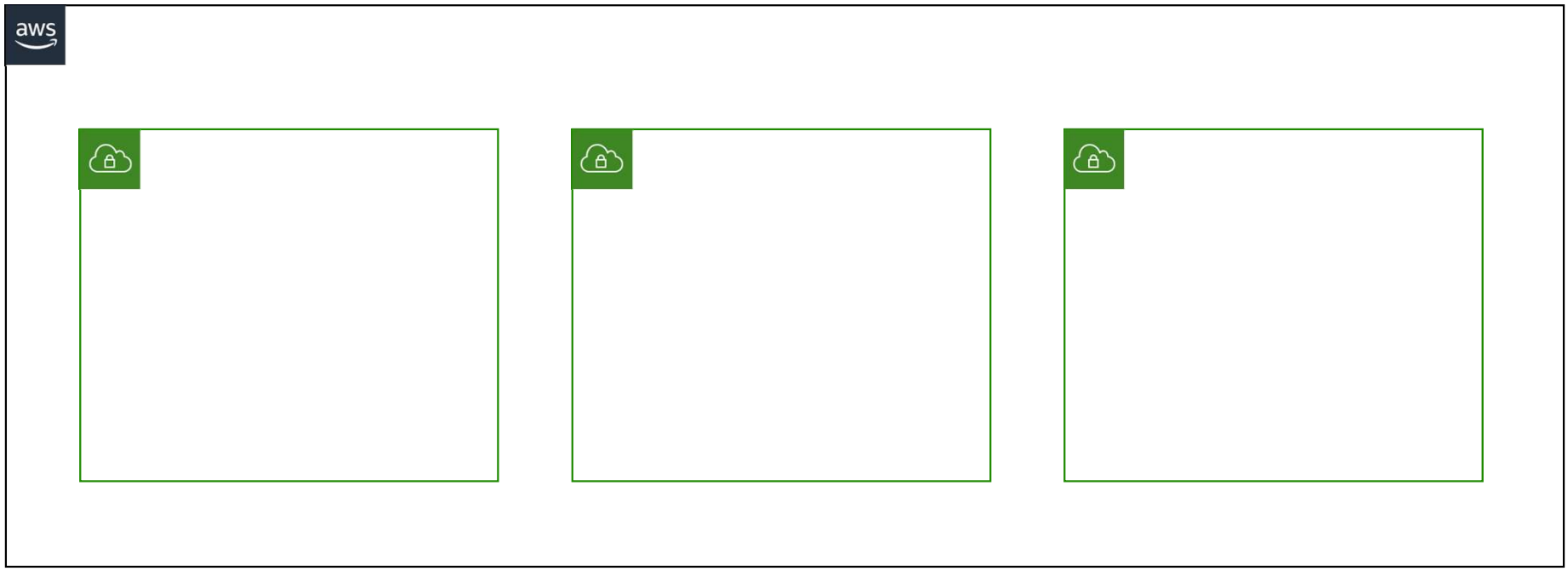
## Monitoring

- Lambda integrates with CloudWatch and X-Ray for monitoring and troubleshooting.
- CloudWatch monitors the number of invocations, duration of each request, errors, and other metrics.
- Lambda Insights is an additional monitoring extension that provides aggregate information for all functions in your account or region.
- X-Ray is a monitoring and troubleshooting tool that allows you to view requests as they travel through your application and identify bottlenecks and errors.
- TCP/IP and network traffic information are unavailable by default for Lambda but may be available through Flow Logs if Lambda runs inside a Virtual Private Cloud (VPC).

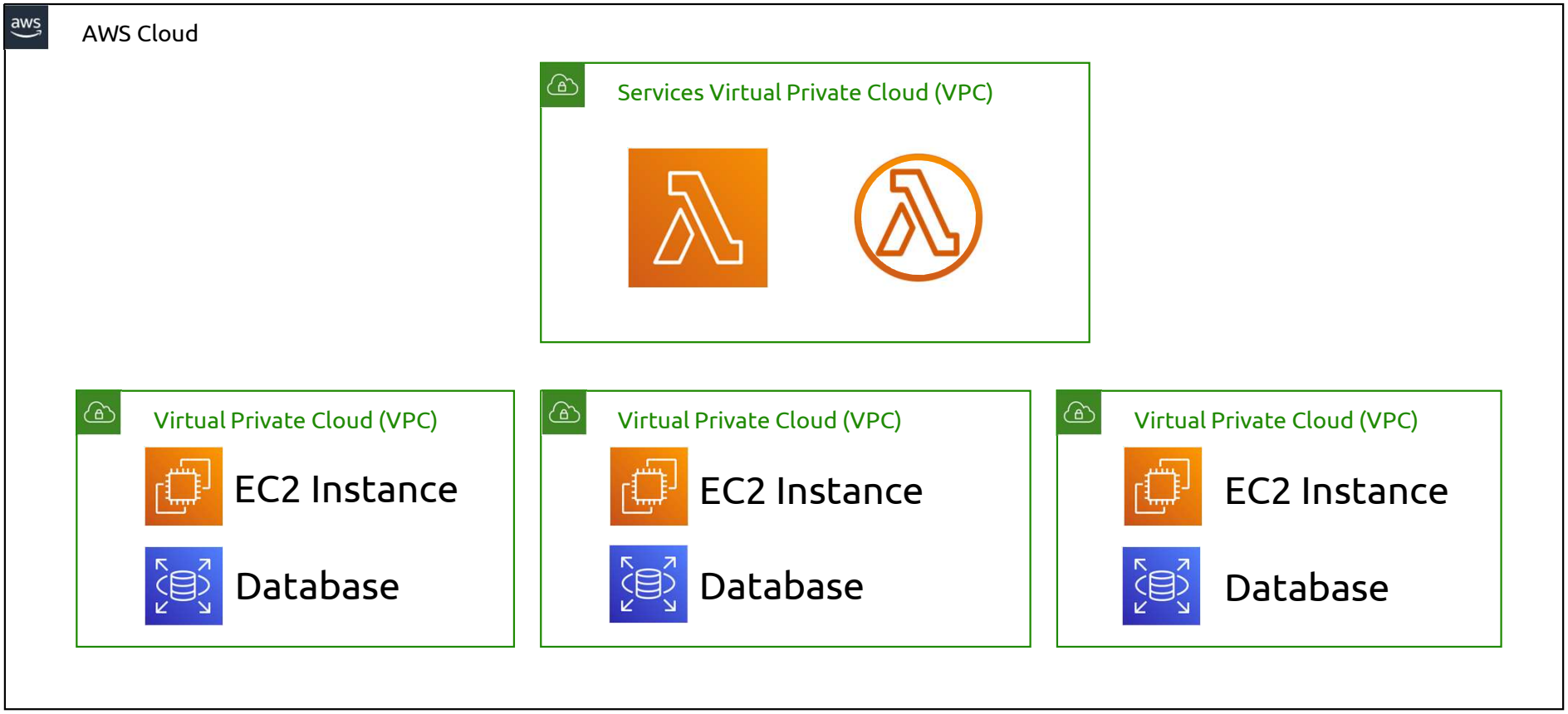
# Lambda Networking

# Lambda Networking

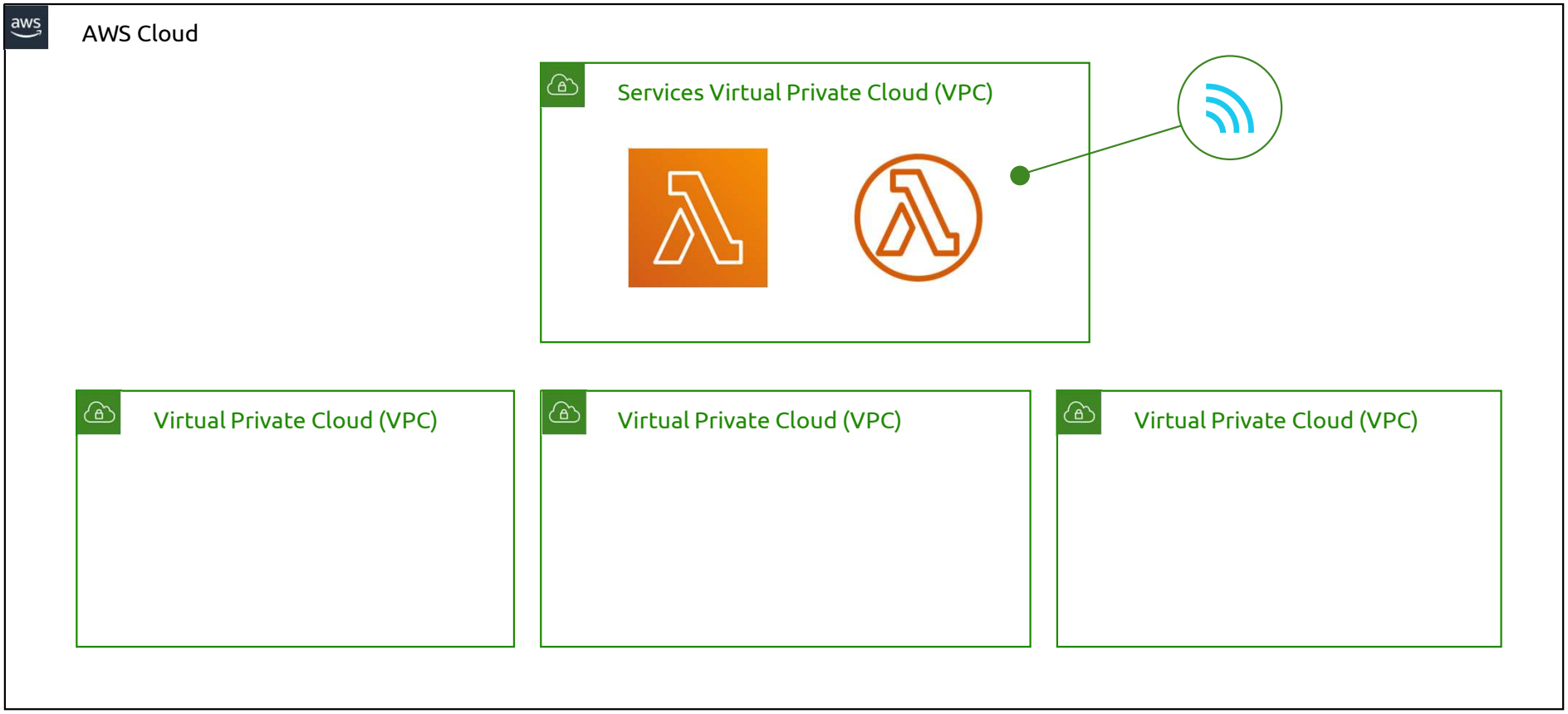
# Lambda Networking



# Lambda Networking

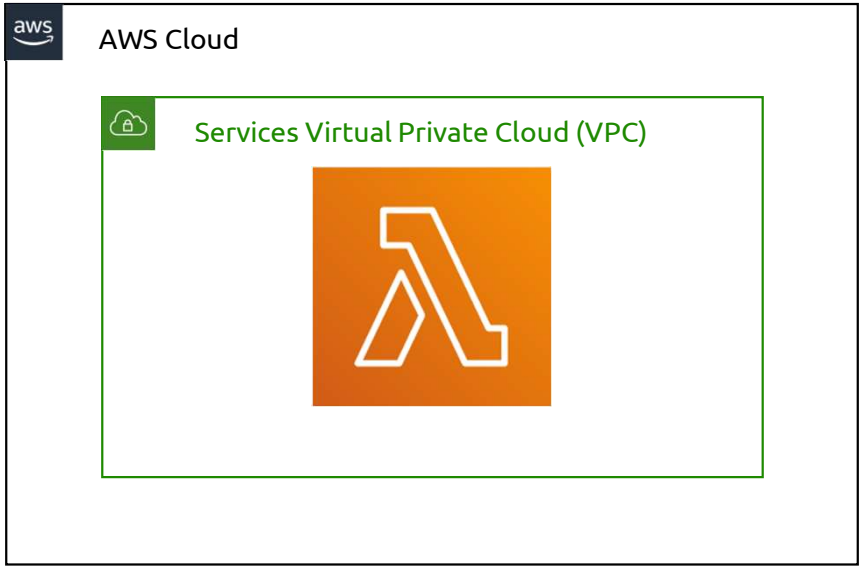


# Lambda Networking

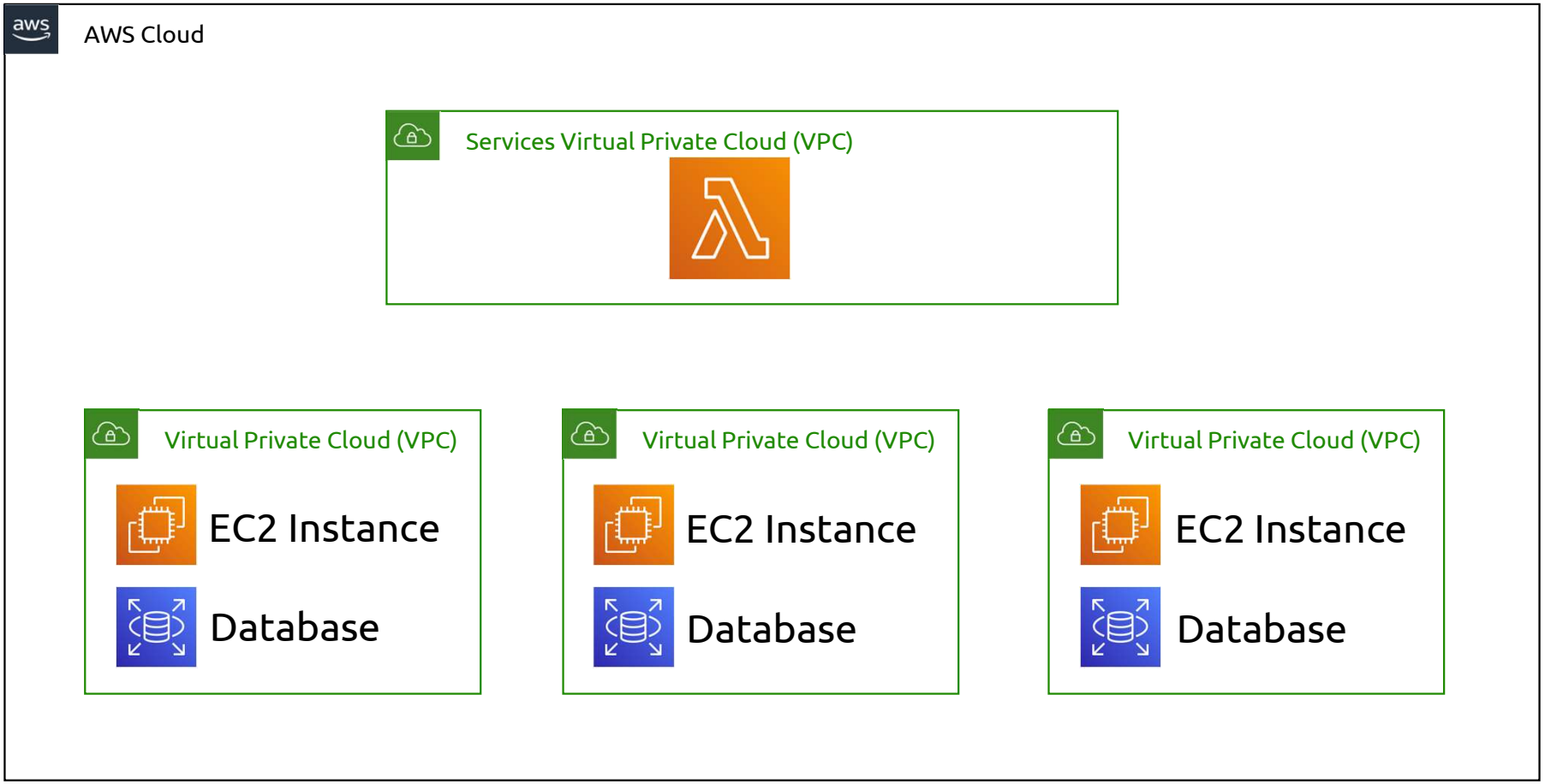


# Lambda Networking

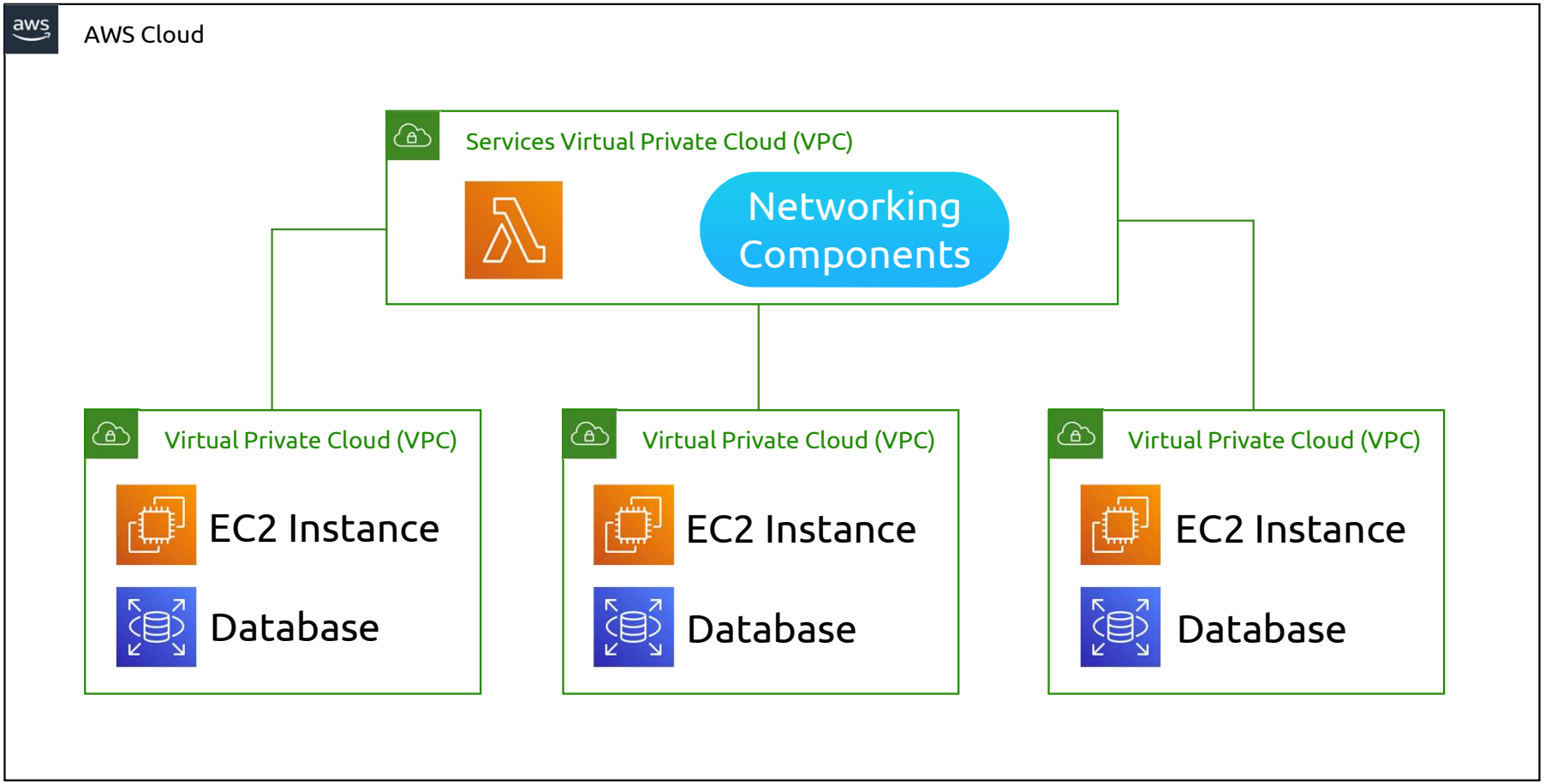
# Lambda Networking



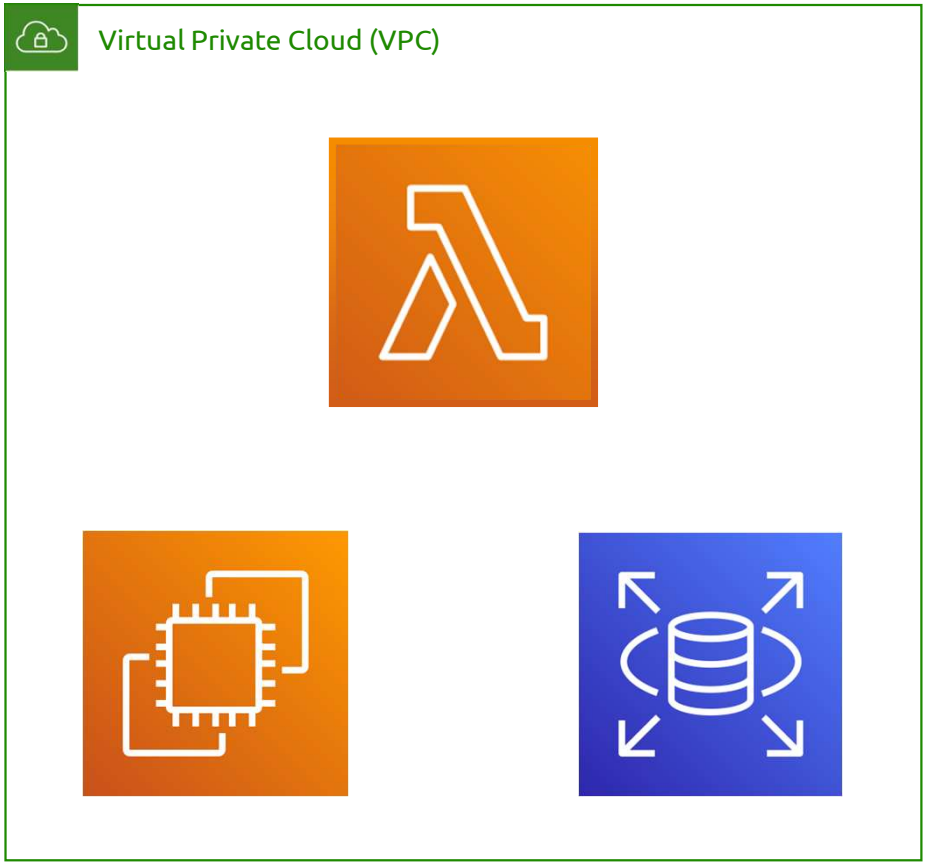
# Lambda Networking



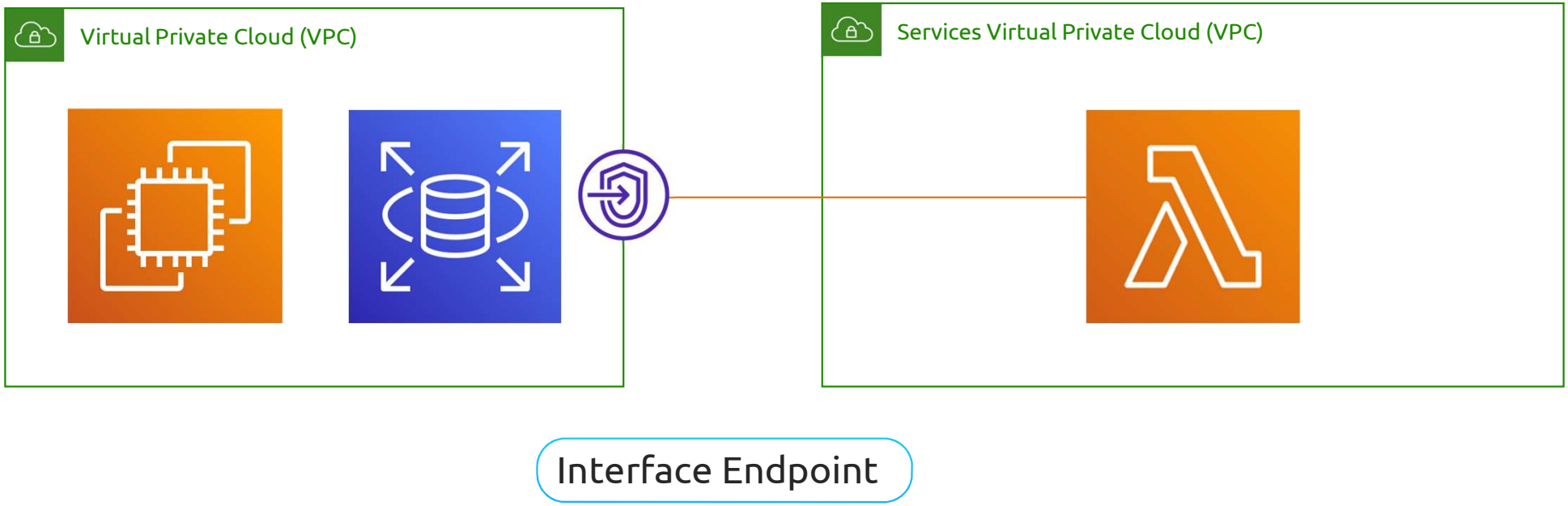
# Lambda Networking



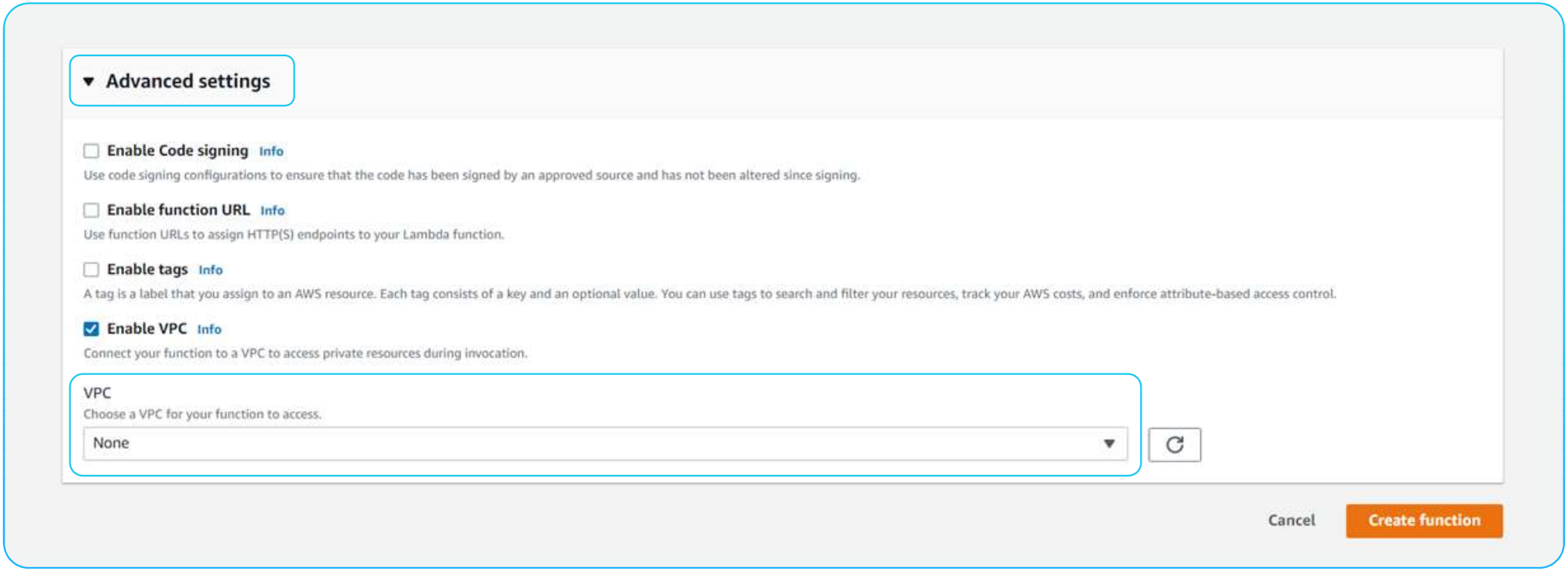
# Lambda Networking



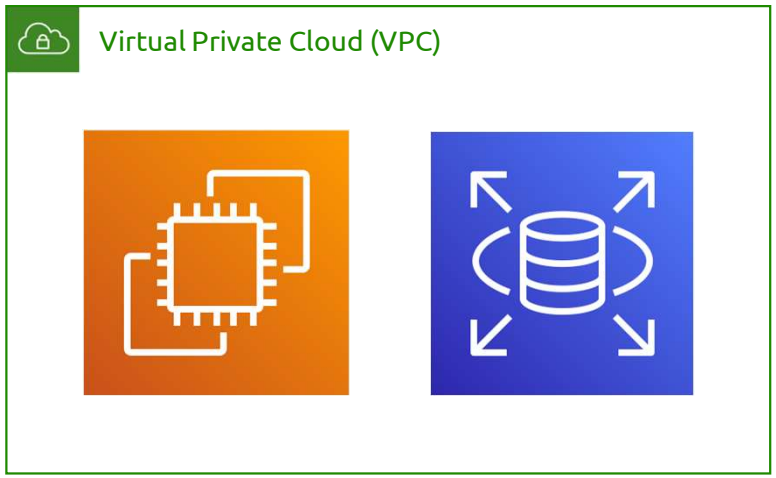
# Lambda Networking

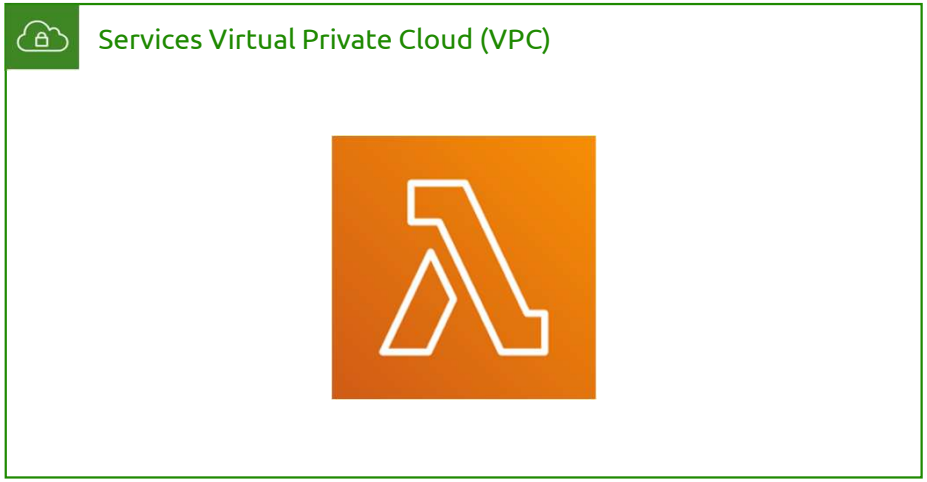


# Lambda Networking

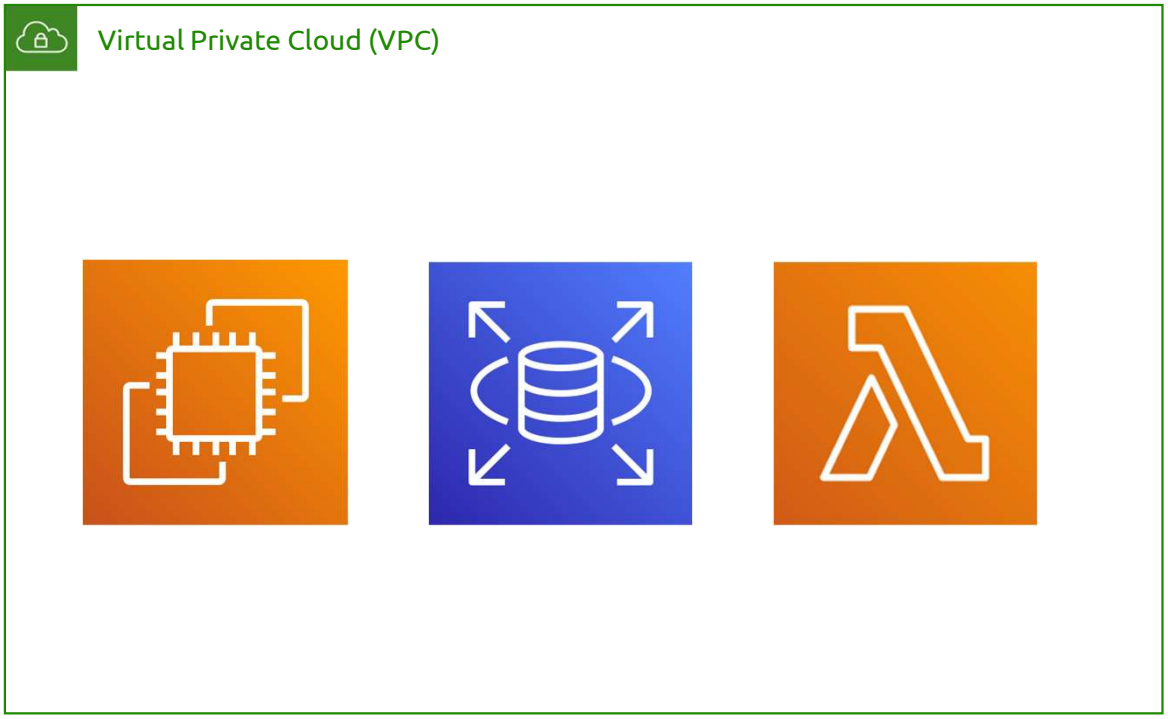


# Lambda Networking

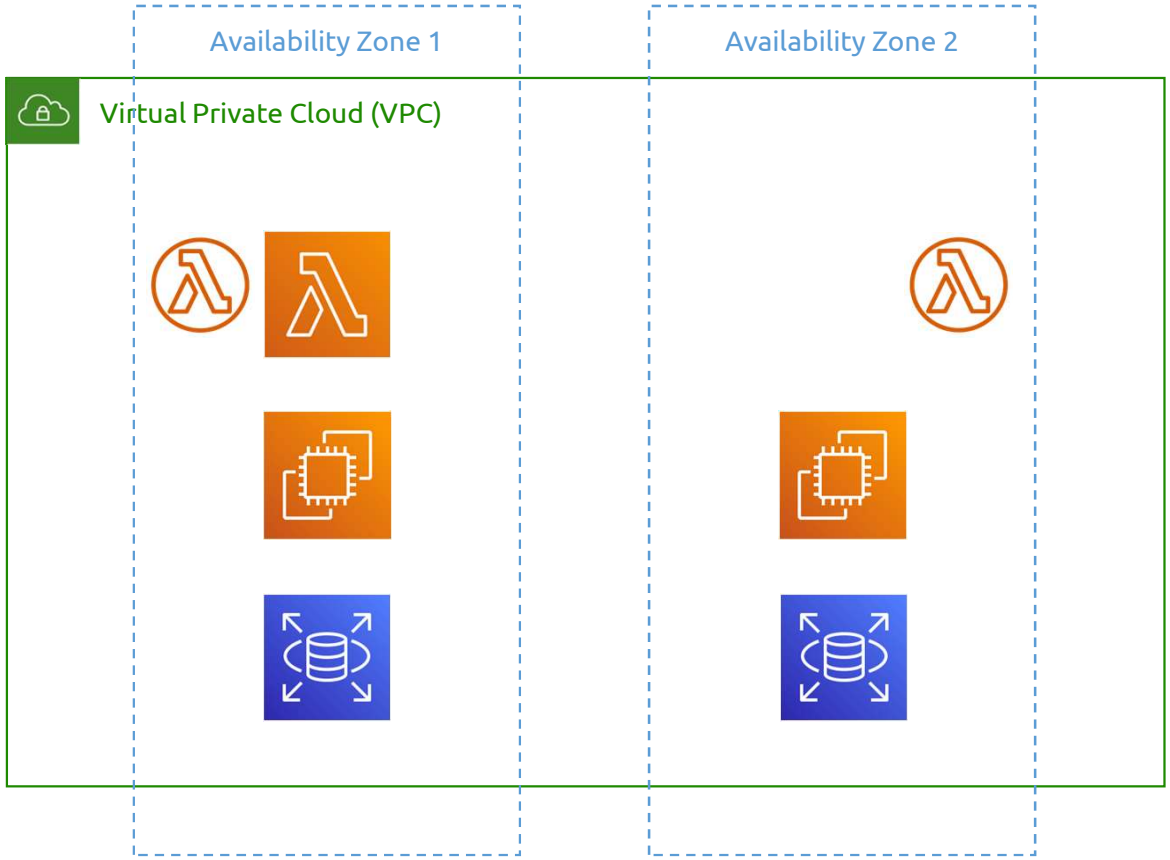


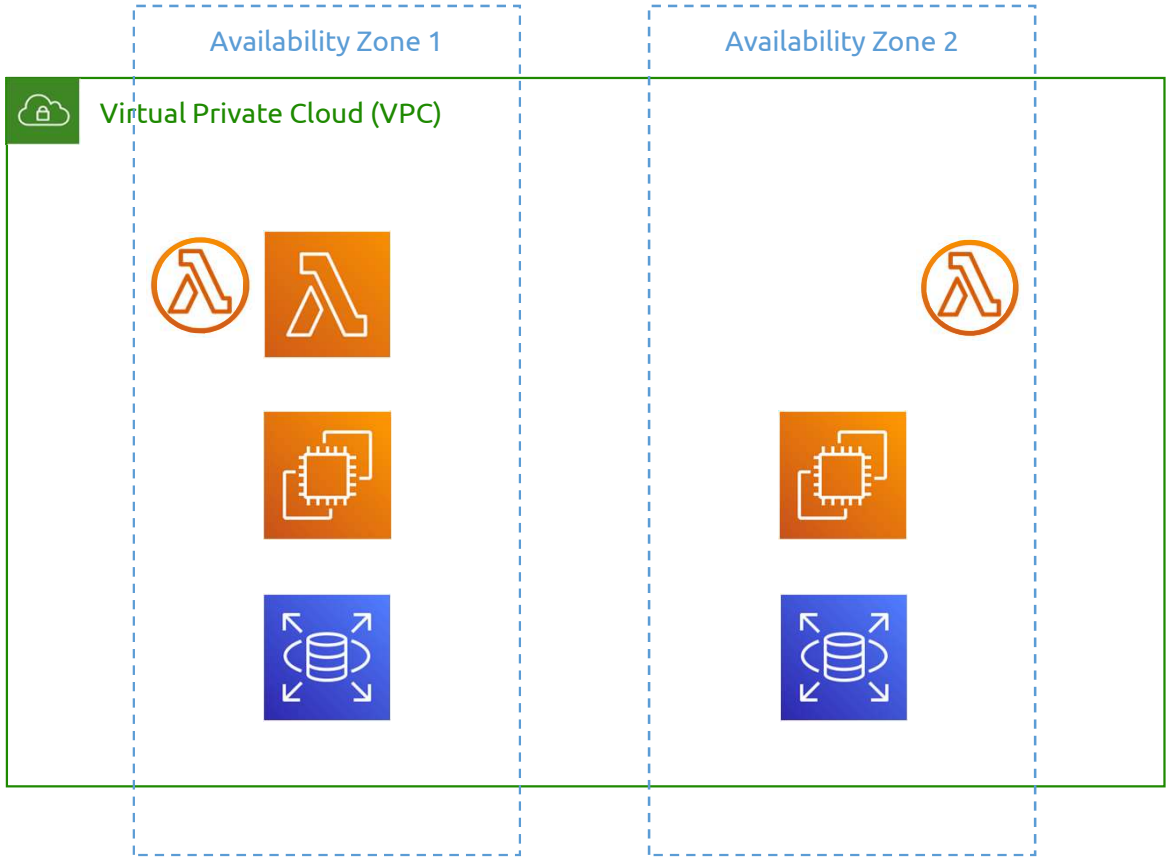
# Lambda Networking



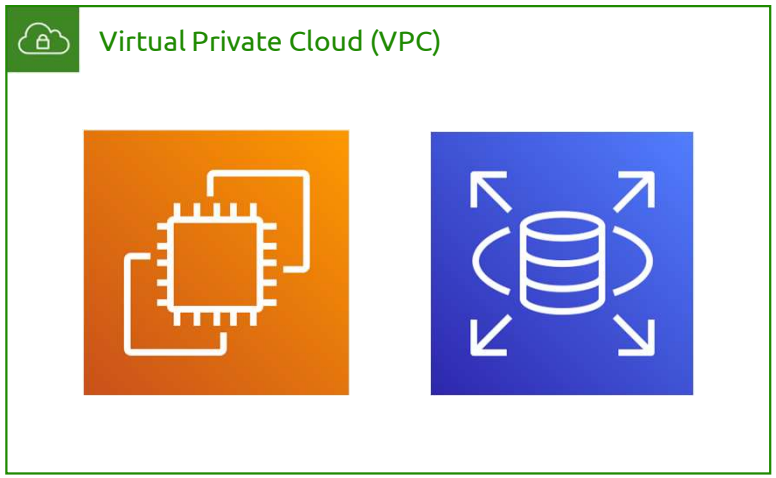
# Lambda Networking

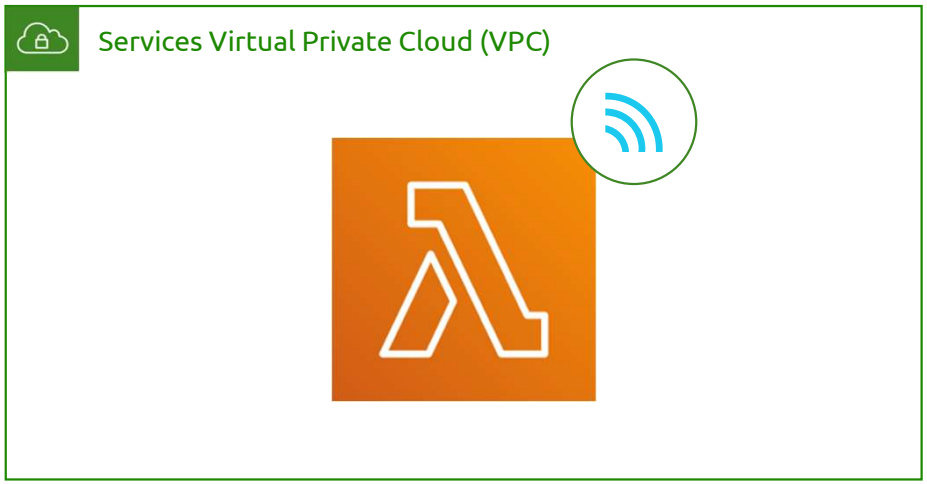


# Lambda Networking

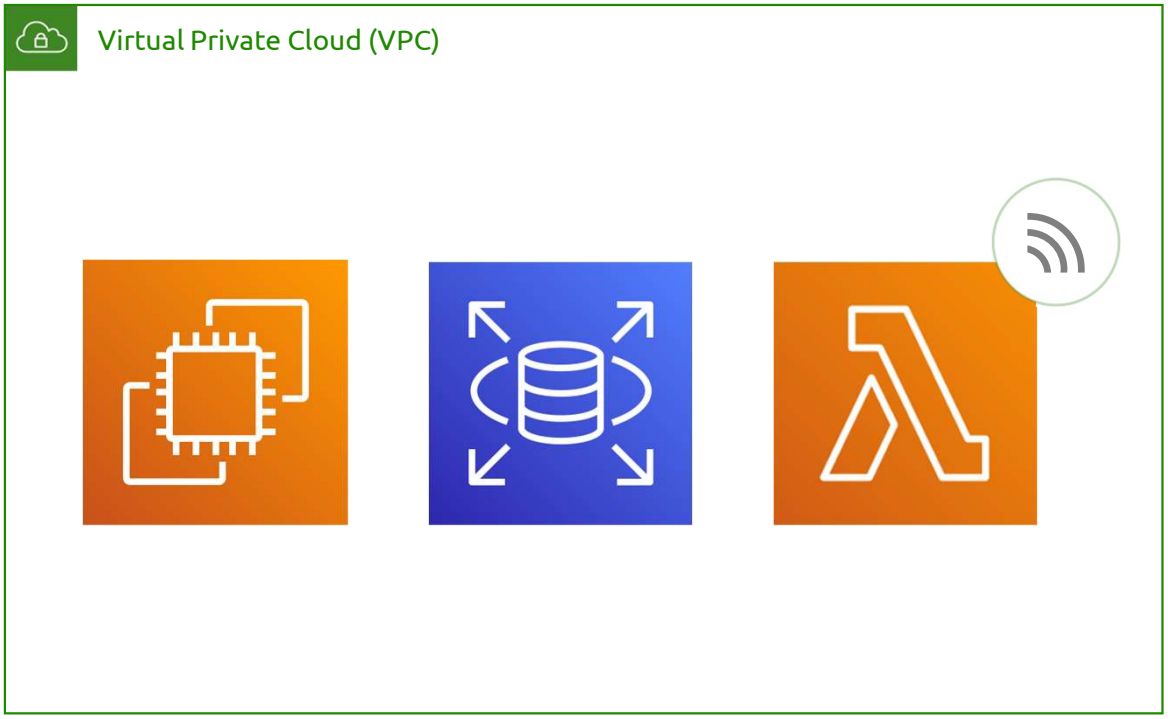


# Lambda Networking

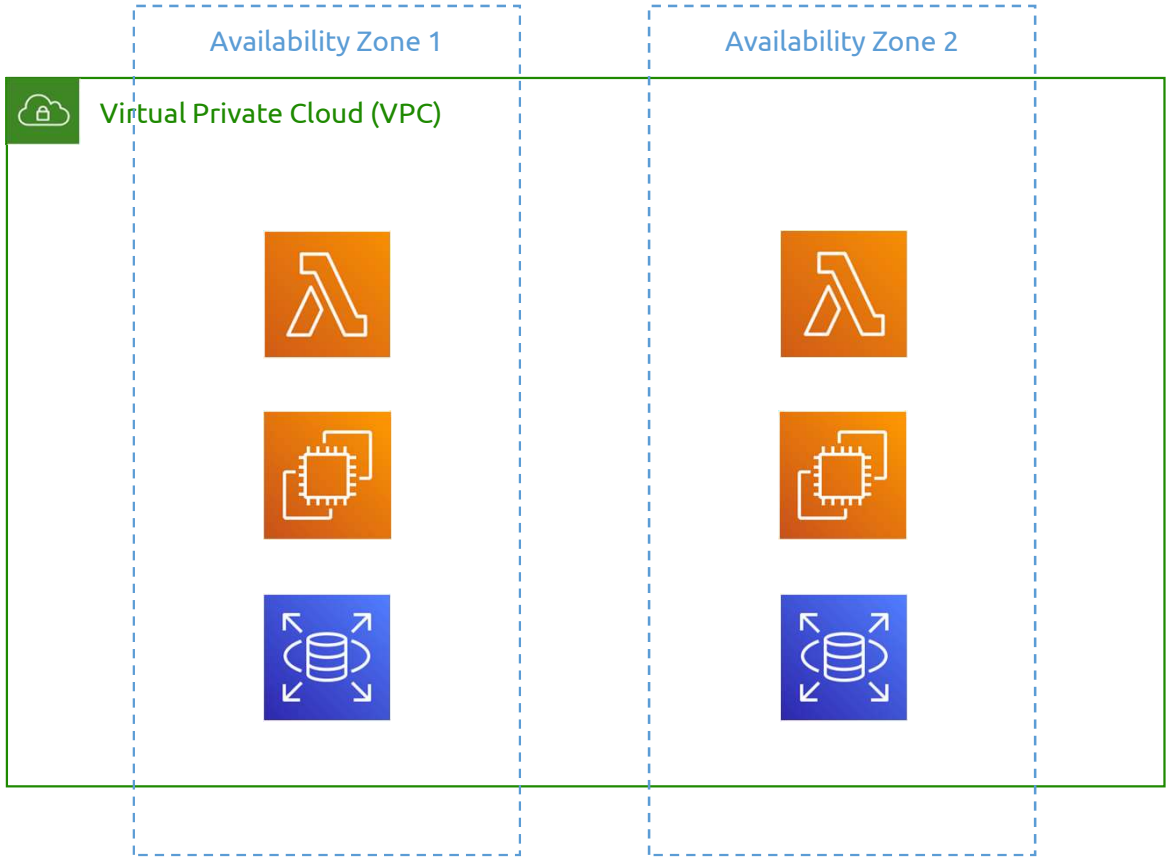


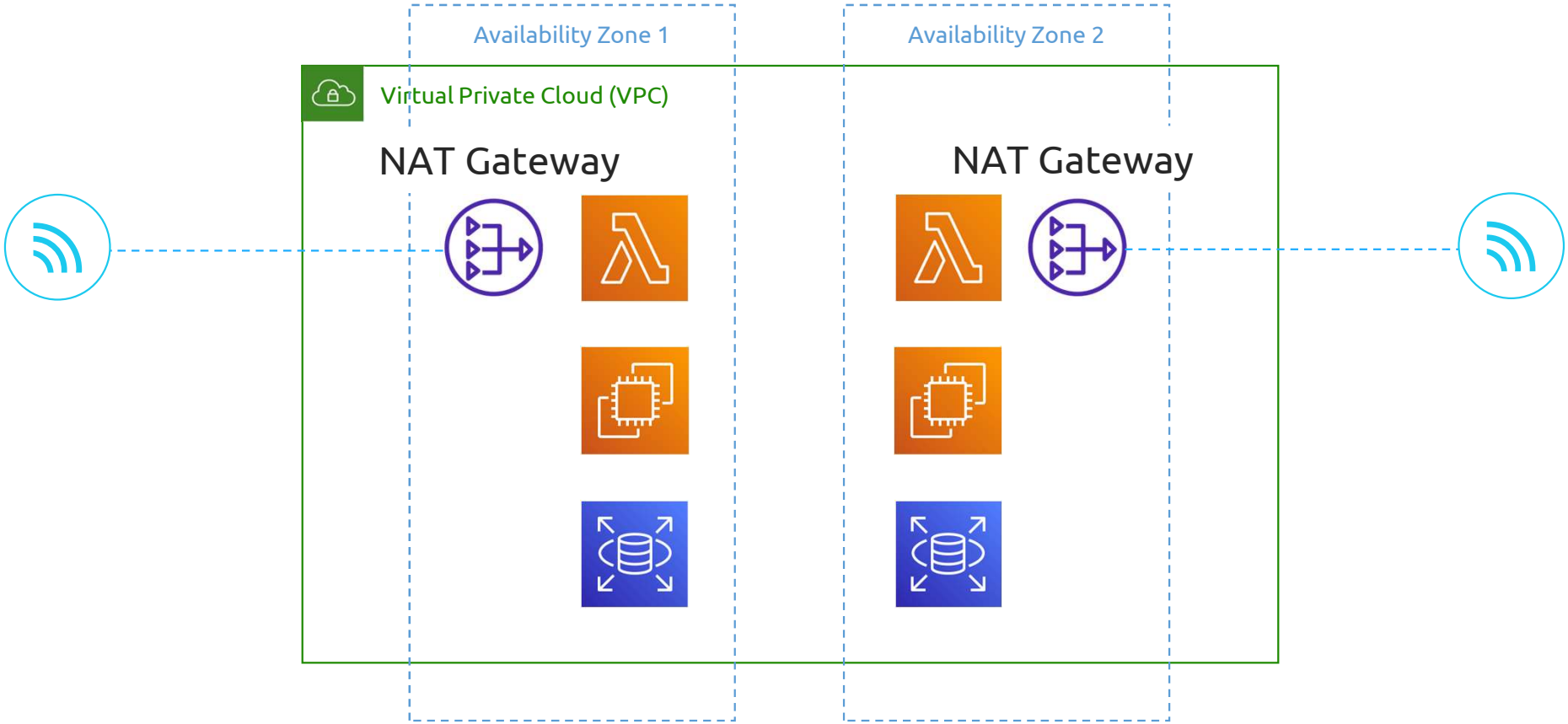
# Lambda Networking



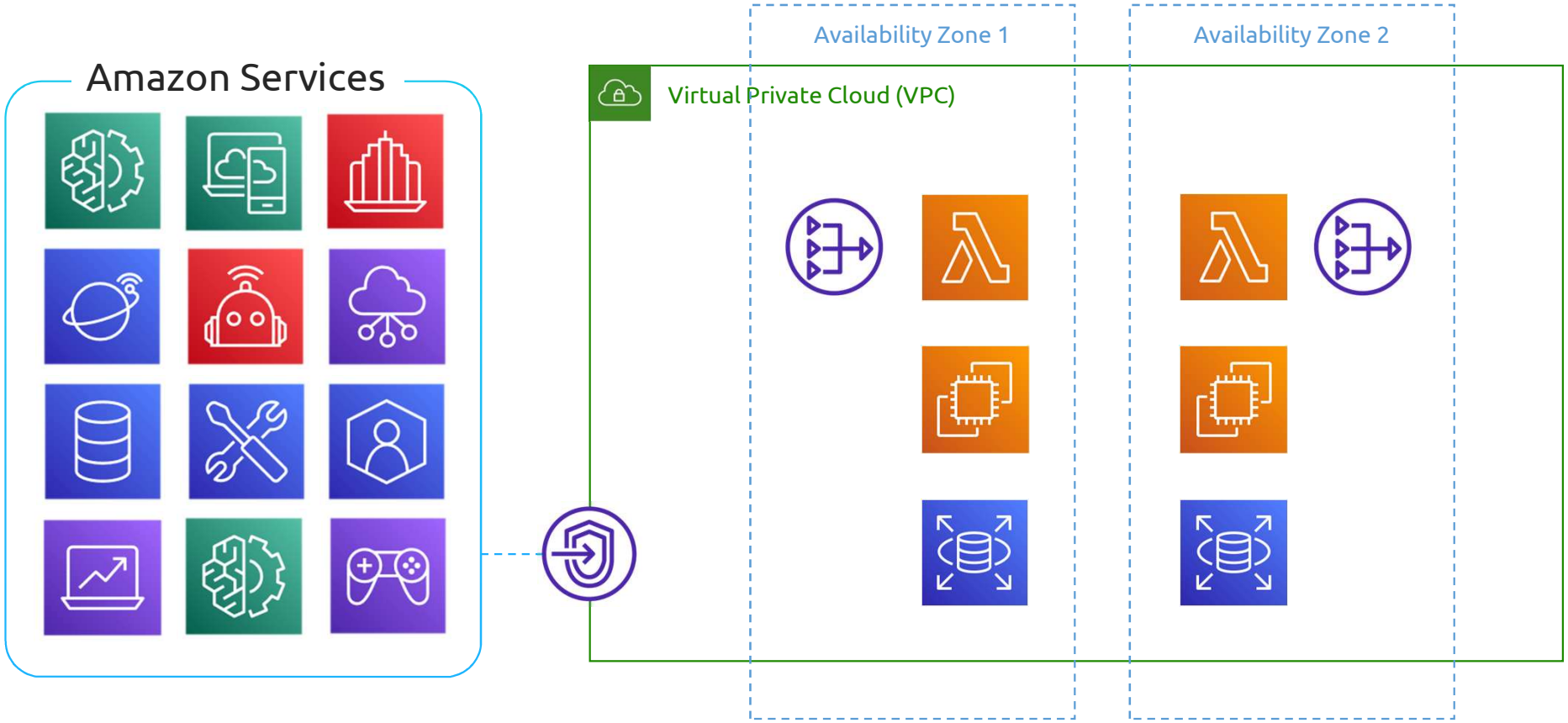
# Lambda Networking



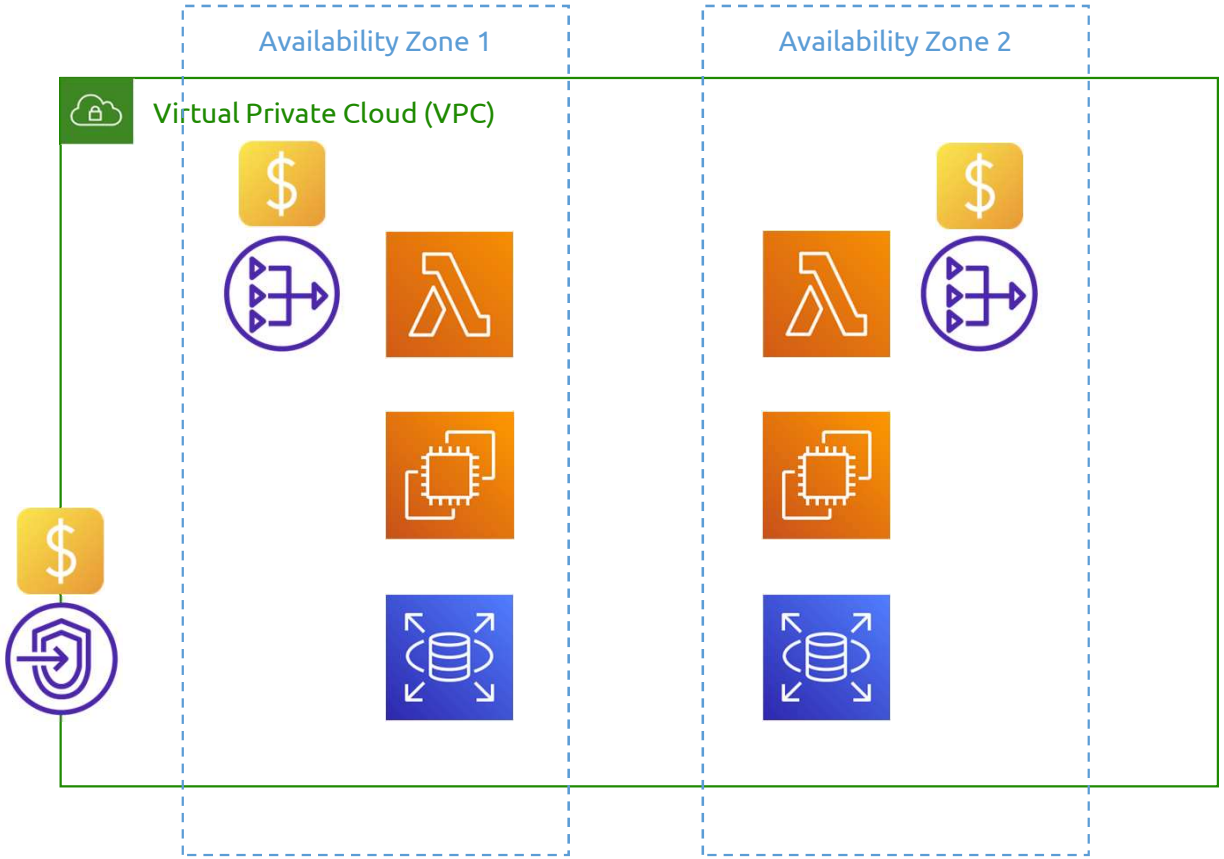
# Lambda Networking



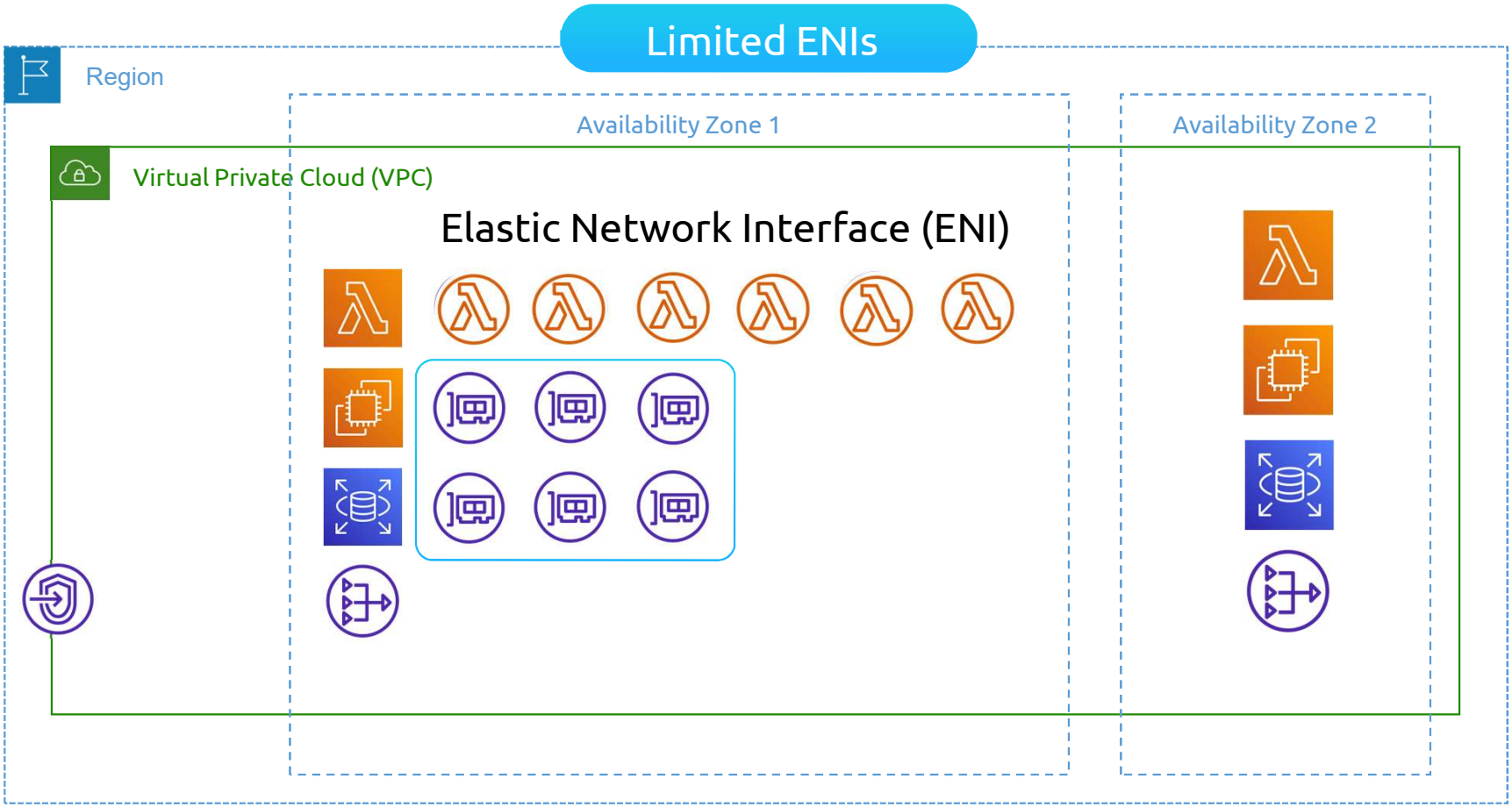
# Lambda Networking



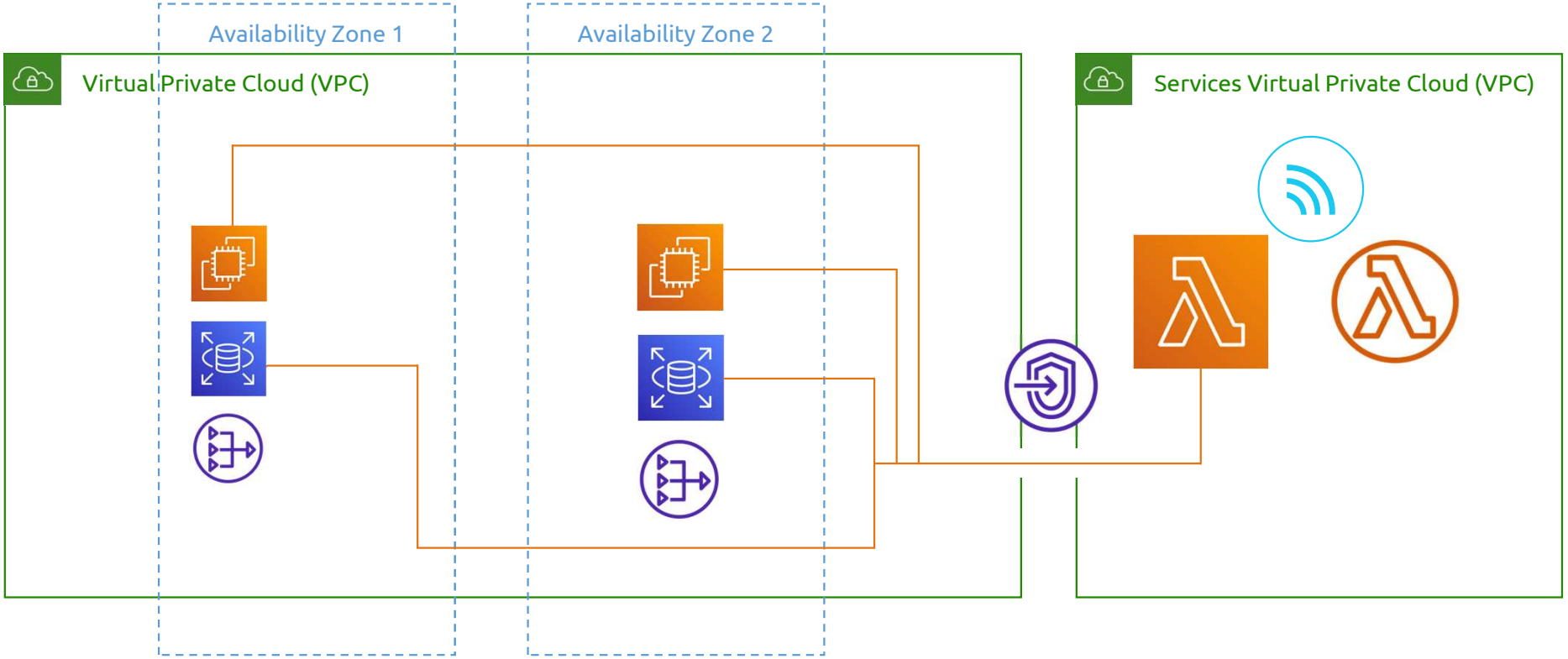
# Lambda Networking



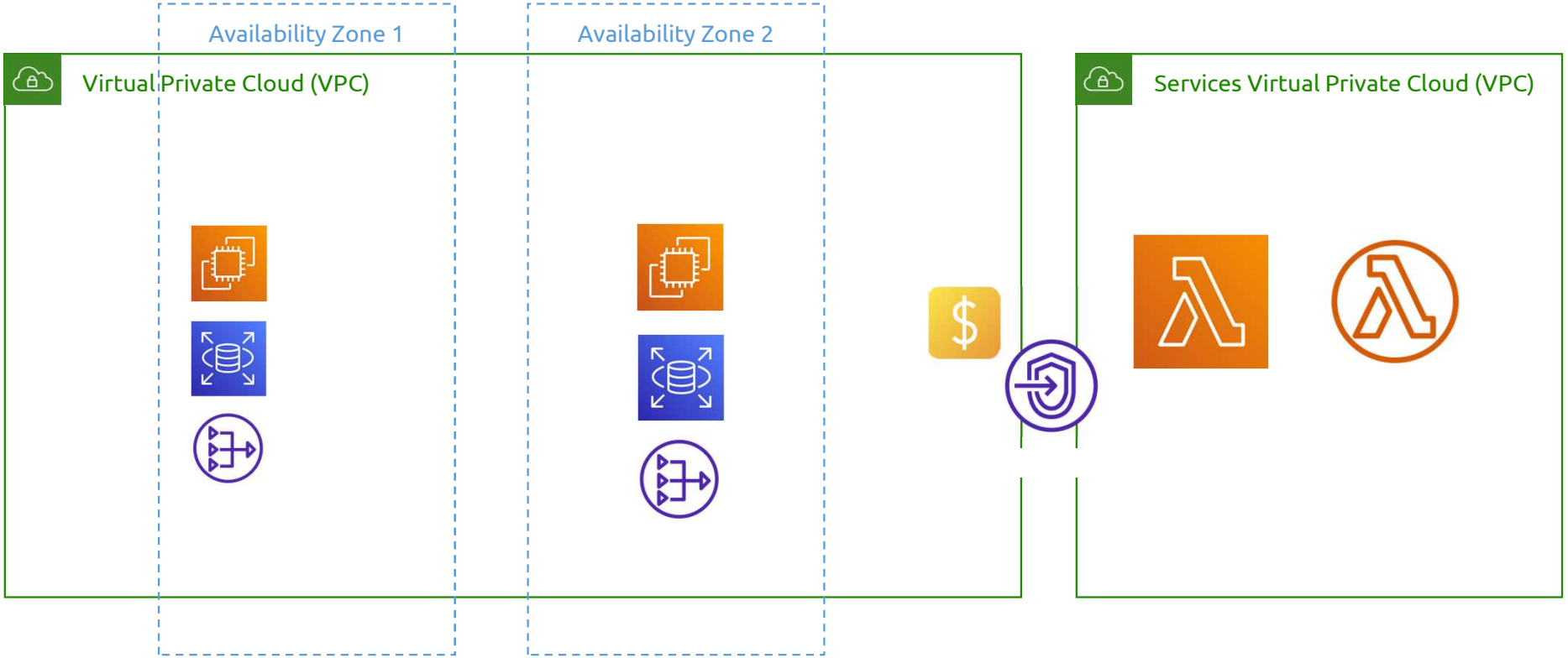
# Lambda Networking



# Lambda Networking



# Lambda Networking



# Summary

## Networking

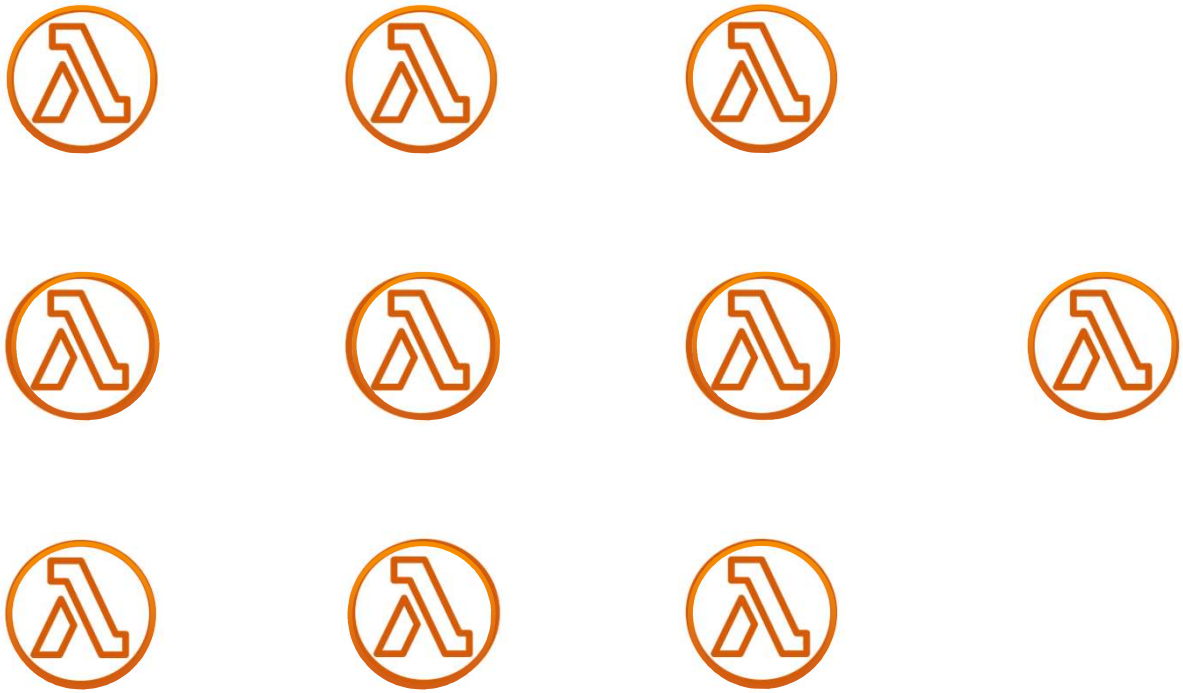
- Lambda functions are placed in a special area in AWS called a services VPC which are the isolated networks in the AWS cloud that holds certain resources.
- The services VPC has access to the internet and other AWS services.
- The services VPC cannot communicate with resources inside private VPCs.
- Two options for adding networking for Lambda: Running Lambda inside your own private VPC or adding a private connection between your VPC and Lambda using an interface endpoint.
- Running Lambda in your own VPC requires high availability design, a NAT gateway for internet connectivity, and adding endpoints to access other AWS services.
- Running Lambda in your own VPC also affects the number of concurrent Lambda executions and adds to the cost.

# Reserved and Unreserved Concurrency

# Concurrency

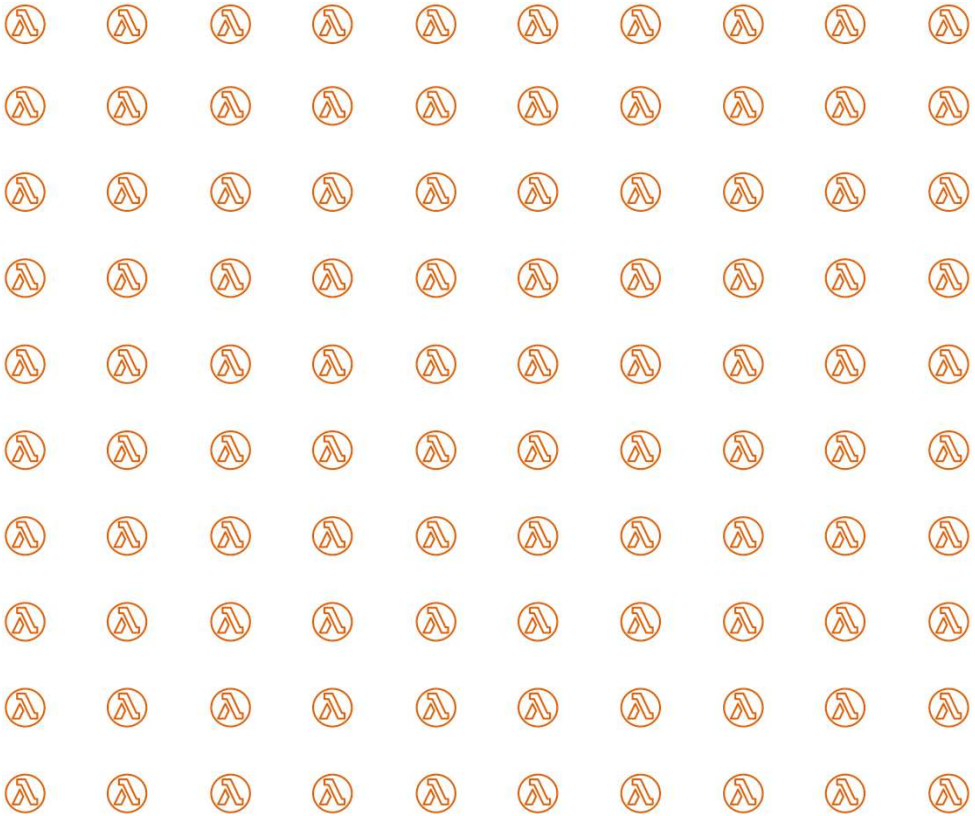
1

# Concurrency



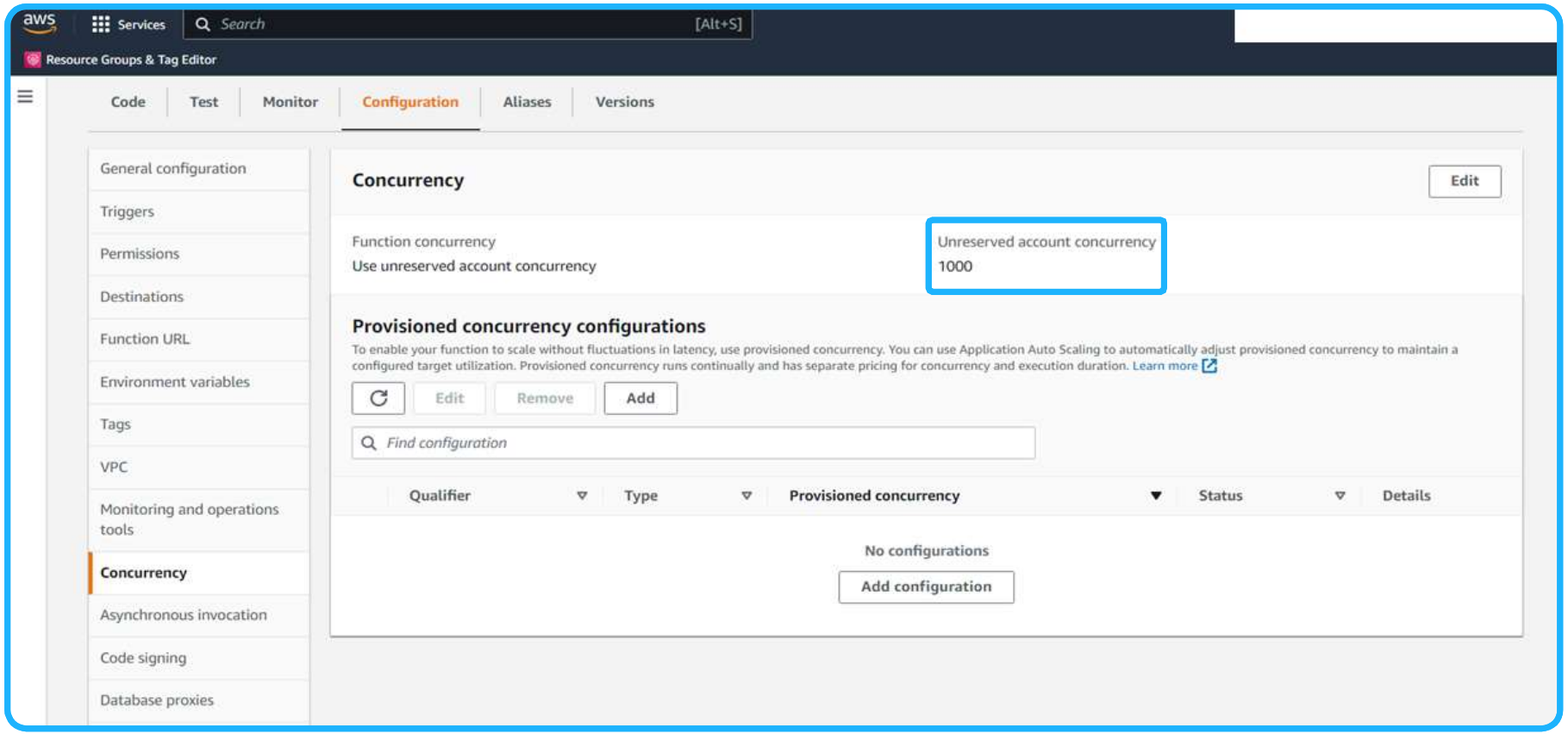
10

# Concurrency

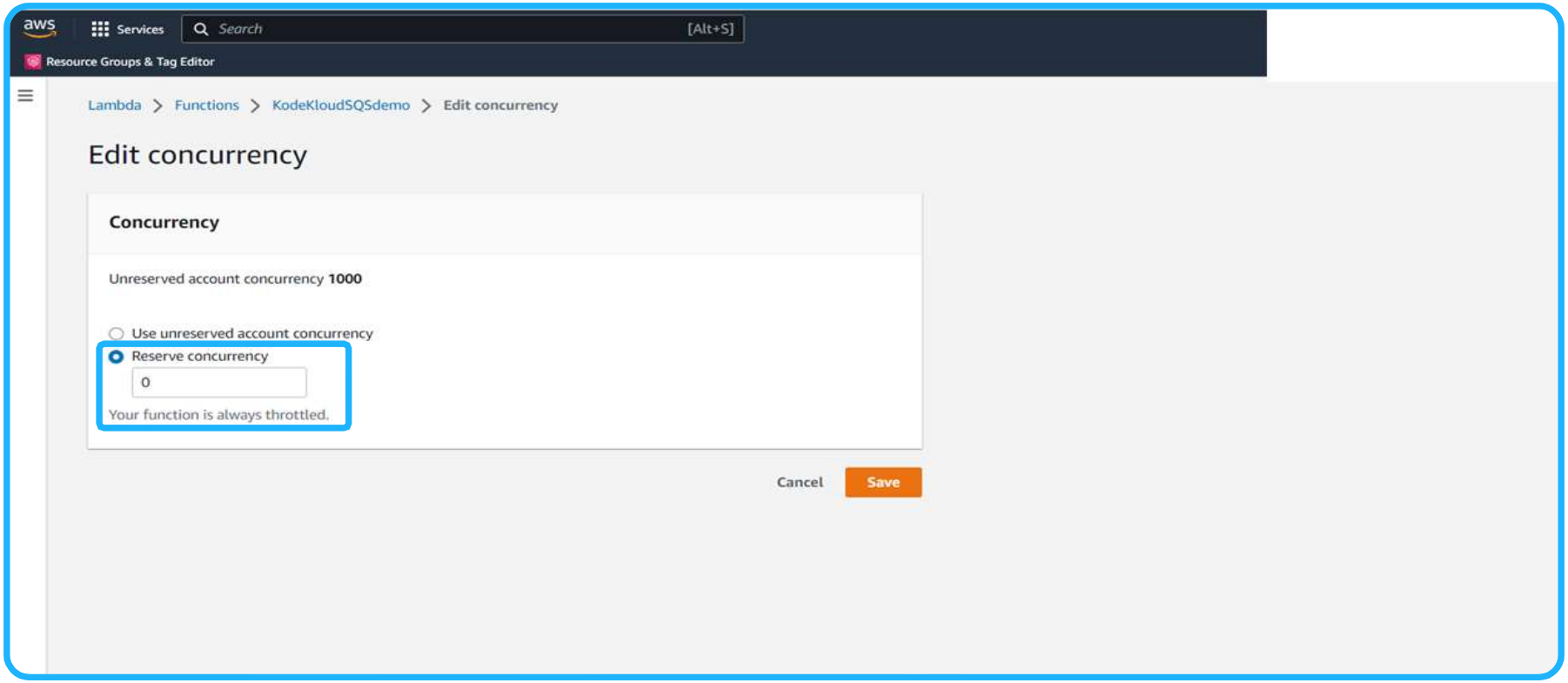


100

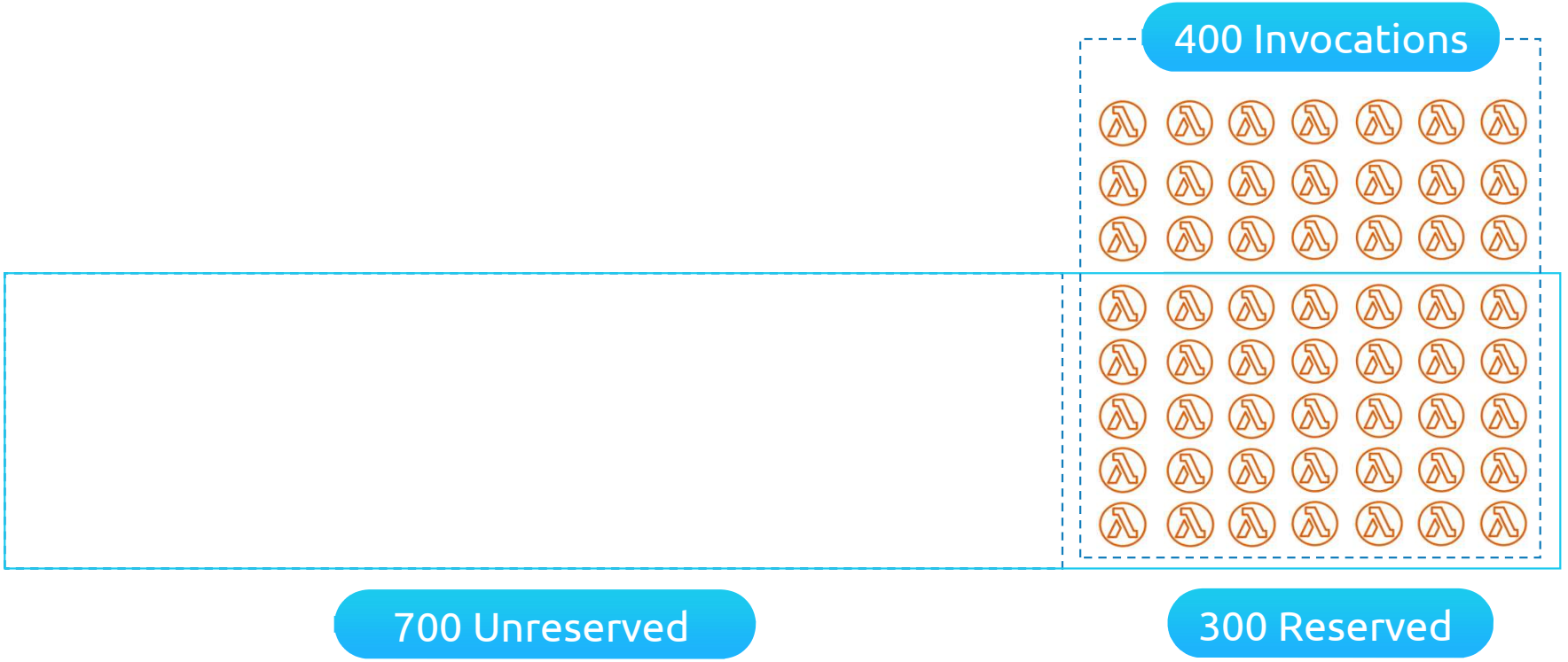
# Unreserved Concurrency



# Reserved Concurrency

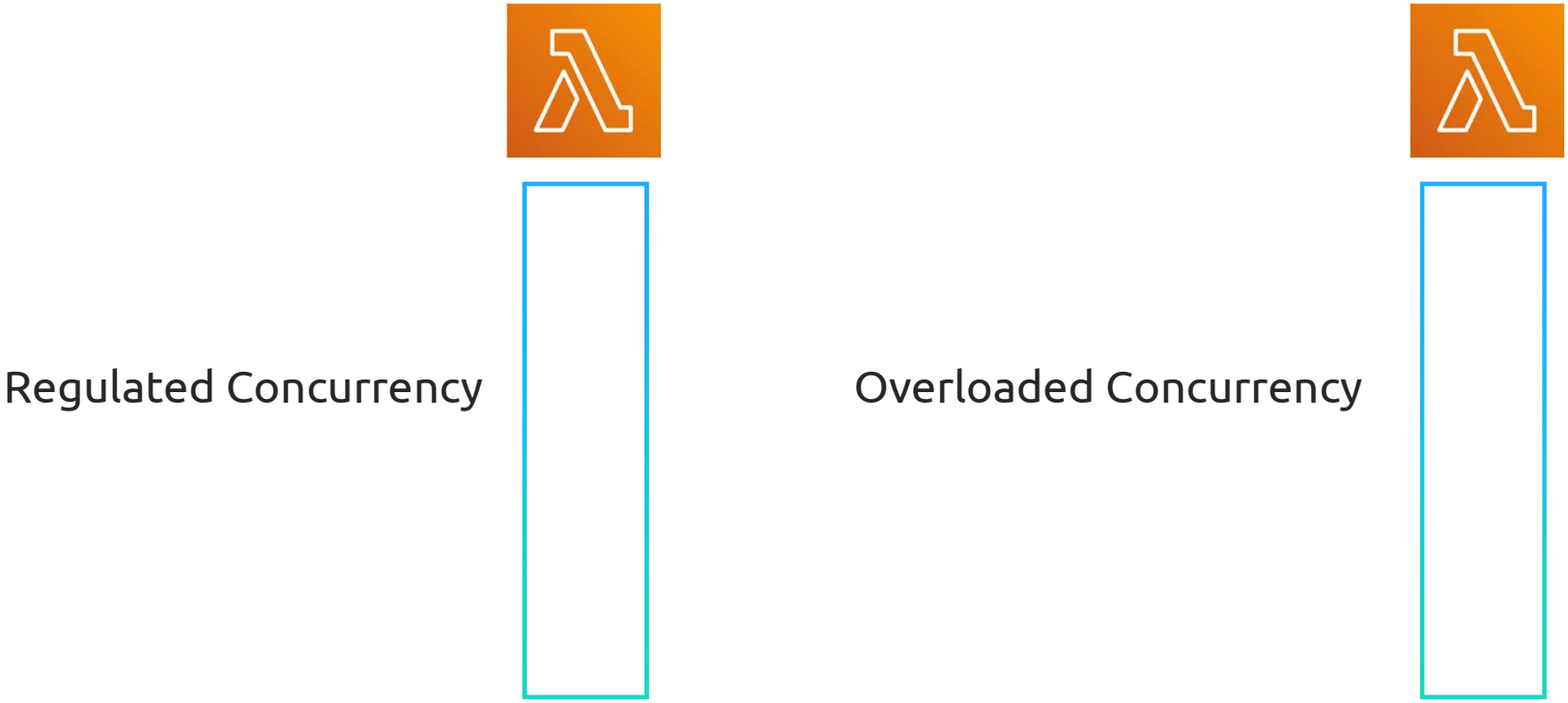


# Reserved Concurrency

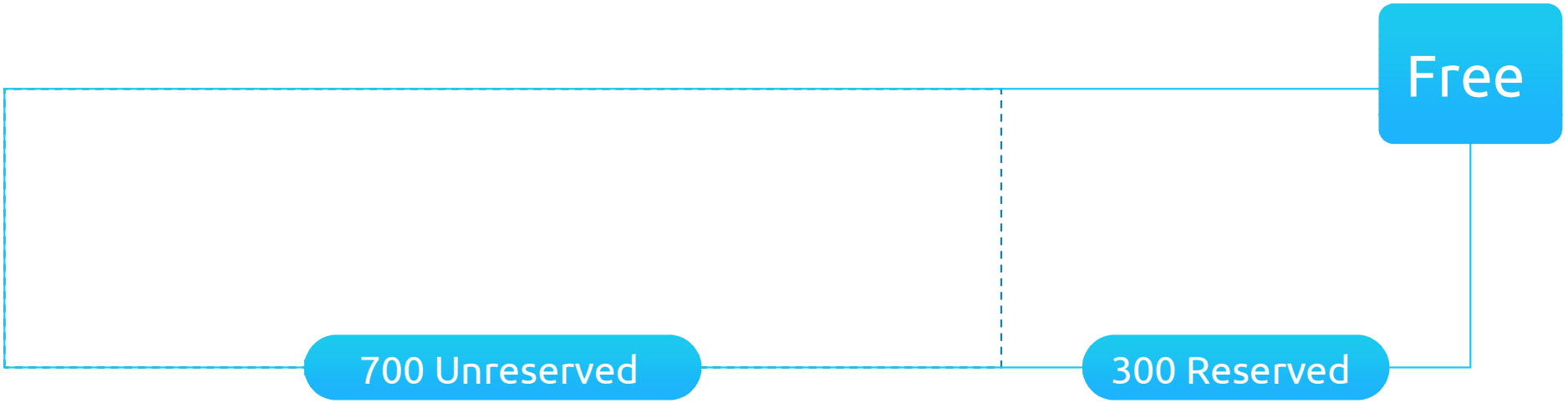



1000 (default account concurrency)

# Reserved Concurrency



# Reserved Concurrency



# Reserved Concurrency

1000

# Reserved Concurrency

3000

In some Regions

Limited Time

# Summary

## Concurrency

- AWS Lambda has limits on how many functions can run simultaneously.
- Concurrency settings control which functions get priority within those limits.
- You can configure concurrency settings in the Configuration tab within your function settings.
- Unreserved Account Concurrency represents the maximum number of Lambda functions you can run simultaneously across your account, usually set to 1000.

# Summary

## Concurrency

- Reserved concurrency sets aside a certain number of your limit for a specific function.
- Reserved concurrency guarantees a certain number of invocations available to a function at all times but also sets a maximum ceiling for that function.
- Setting reserved concurrency is free and can help prevent overloading downstream resources.
- You can temporarily scale higher than your unreserved concurrency limit, up to 3000 in some regions, for a limited time.

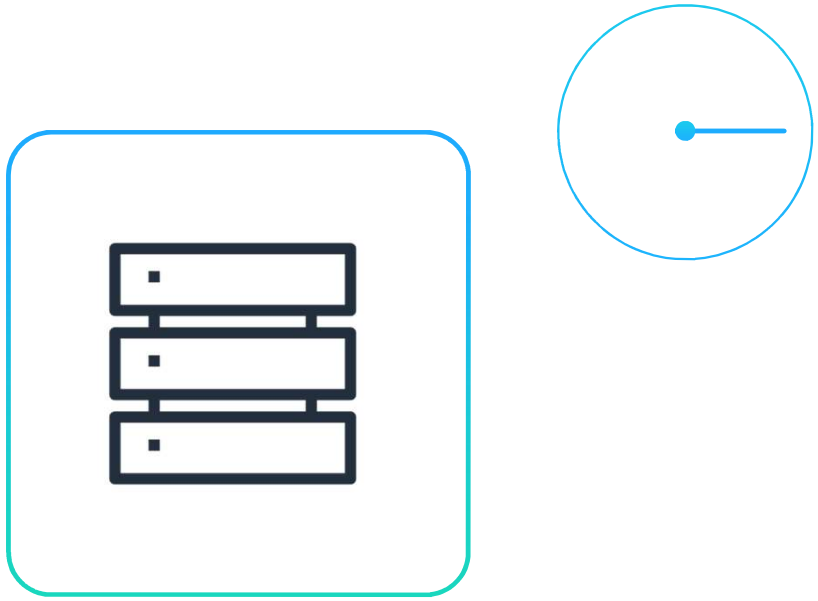
# Provisioned Concurrency

# Are there any downsides?

# Are we removing benefits from servers?

# Lambda Benefits

# Physical Server Benefits



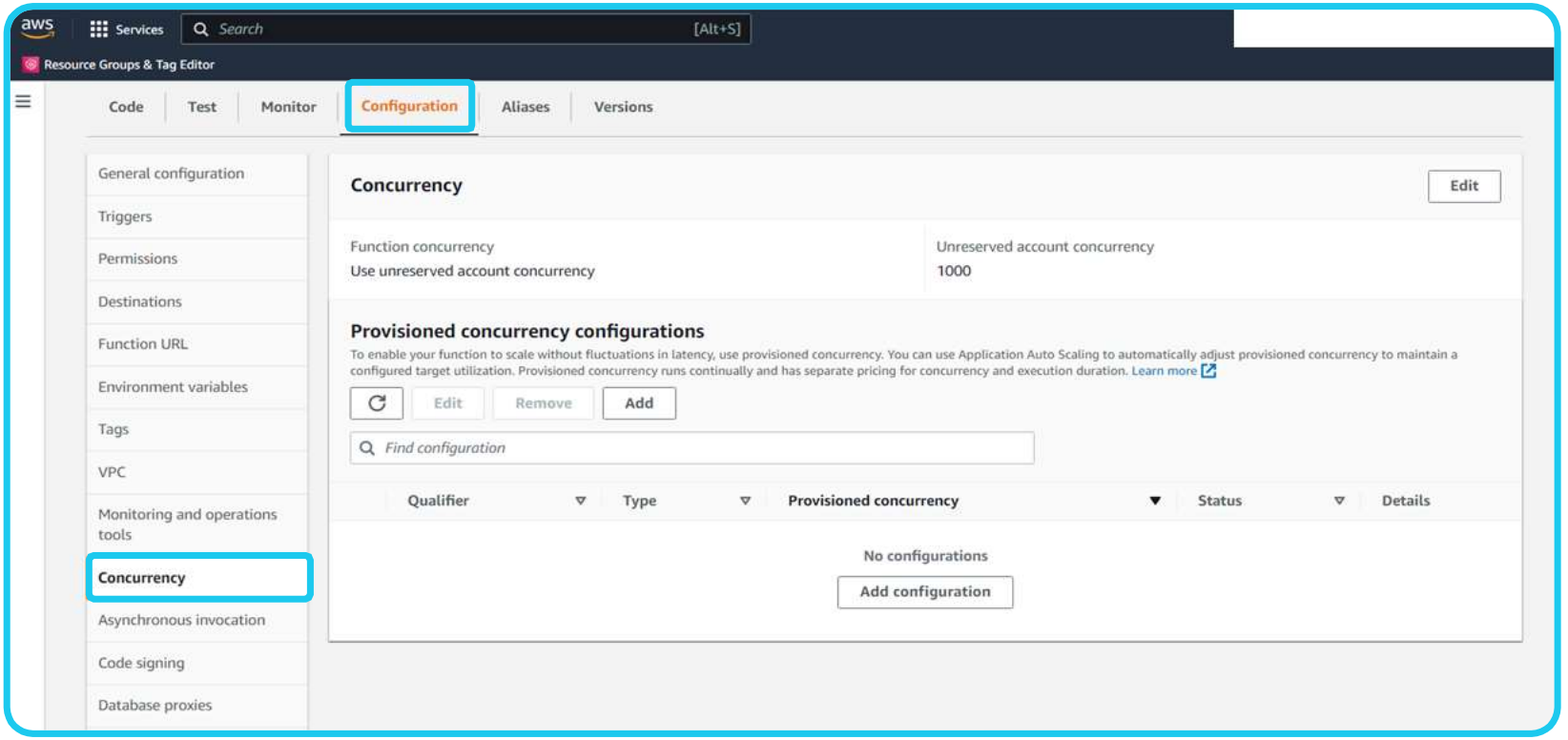
# Event Triggers

# Execution Latency

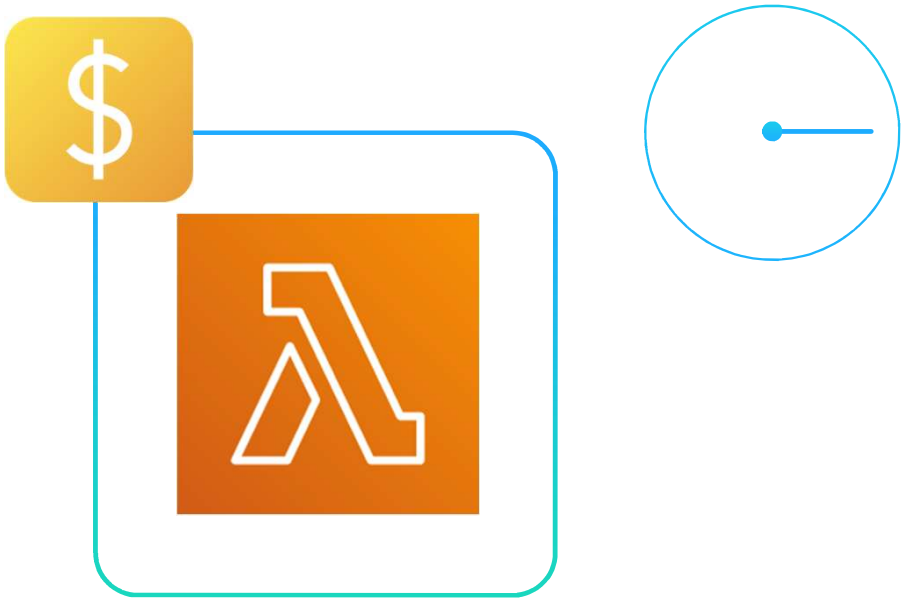
# Provisioned Concurrency

# Provisioned Concurrency

# Provisioned Concurrency



# Provisioned Concurrency



# Summary

## Provisioned Concurrency

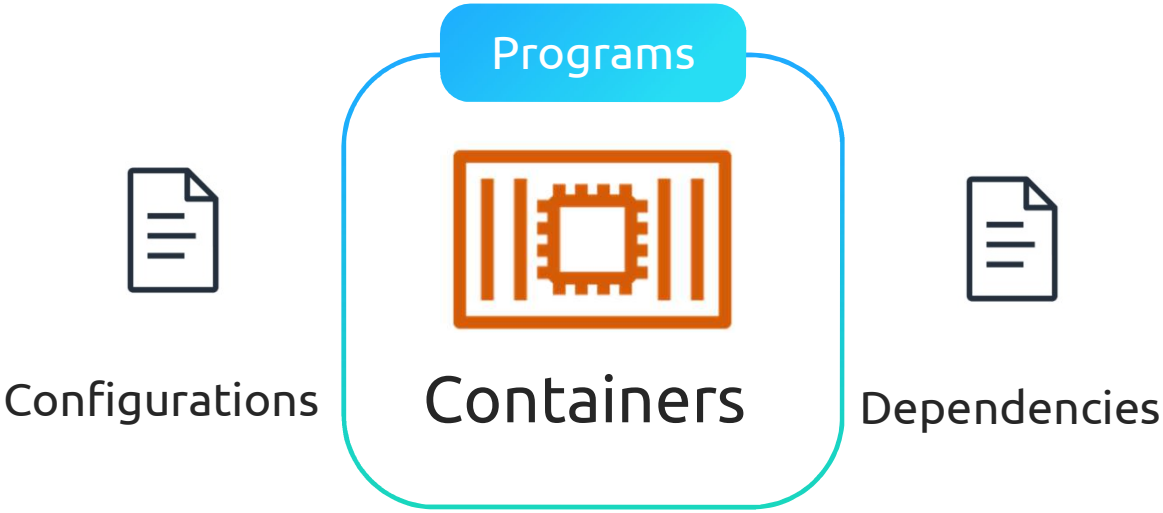
- Provisioned Concurrency is a solution to avoid execution latency in Lambda, where you tell Lambda to keep a certain number of functions running at all times and ready to execute.
- Execution latency might occur due to cold starts if the function wasn't running before the event.
- Provisioned Concurrency can be configured using the concurrency option in the function's configuration tab.
- Provisioned Concurrency adds cost as it requires functions to be kept running permanently.
- Aliases and versions are used to assign Provisioned Concurrency.
- The benefits of removing servers include not having to patch, maintain, and operate them and not having to wait for them to be built to deliver applications.

# Lambda Containers

New

# Lambda Containers

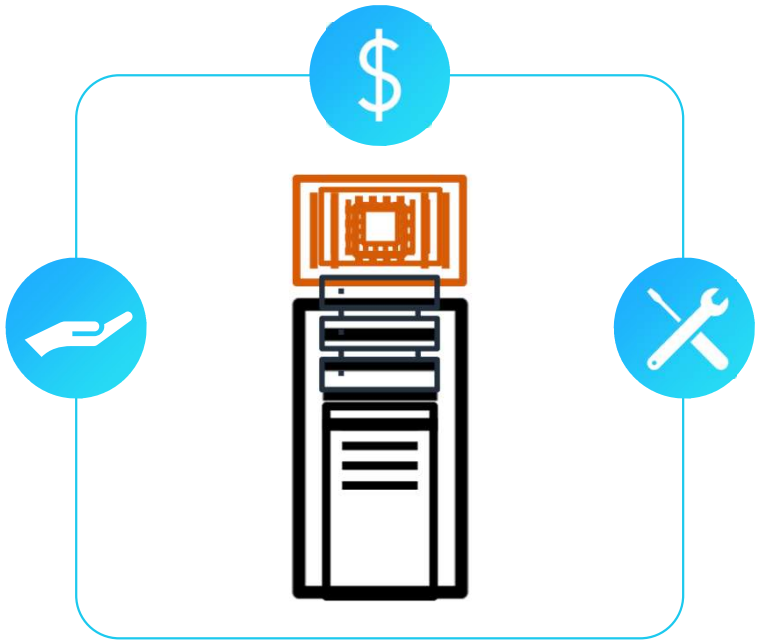
# Lambda Containers



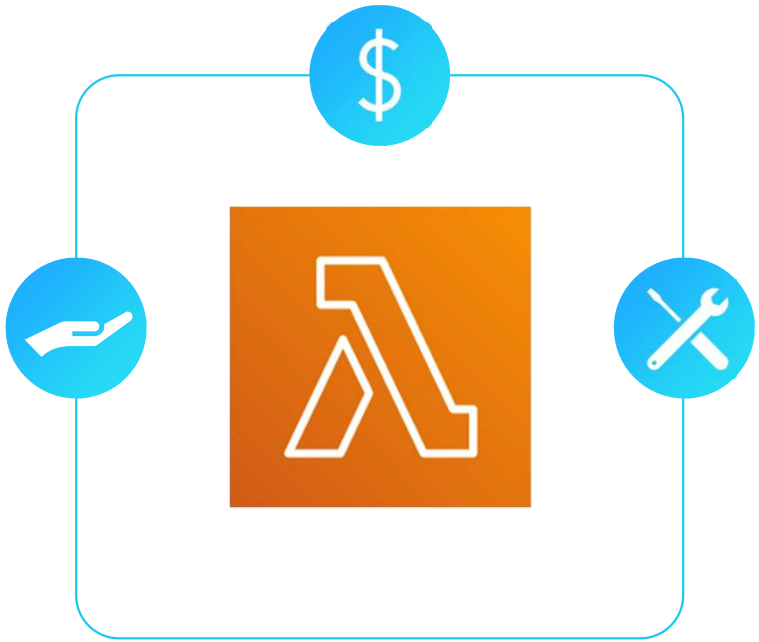
# Lambda Containers

# Lambda Containers

# Lambda Containers



# Lambda Containers



# Lambda Containers

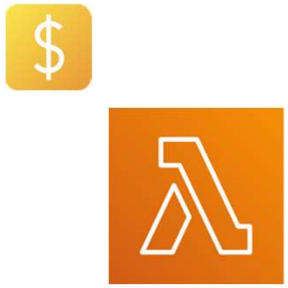
Kubernetes

ECS Autoscaling

# Lambda Containers

# Lambda Containers

# Lambda Containers

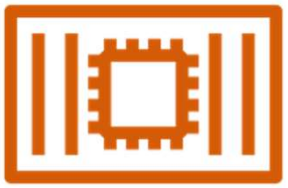


# Lambda Containers

10 GB

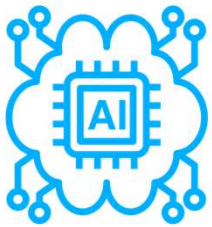
250 MB

# Lambda Containers

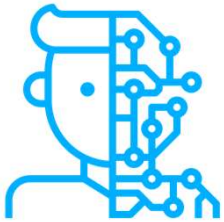


**10 GB**

# Lambda Containers

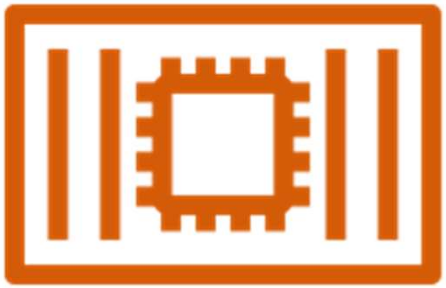



Artificial Intelligence



Machine Learning

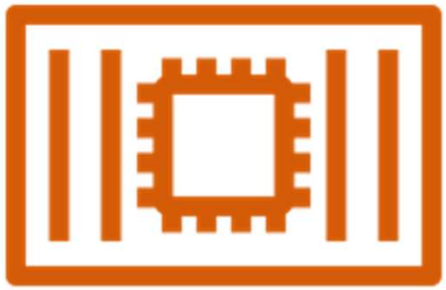
Big Data Analytics



# Lambda Containers

Base image

Code



Base image

Runtime Interface Emulator

Container App

# Summary

## Lambda Containers

- Containers package all necessary code and dependencies to run a program into a small bundle that can be run on any machine or OS.
- Containers are popular due to their portability and convenience but are still typically run on servers.
- Lambda allows for the running of containers, which offers benefits such as removing server-related issues, automatic scaling, and cost efficiency.
- Containers on Lambda allow for setting environment variables, specifying startup parameters, and connecting to a VPC.

# Summary

## Lambda Containers

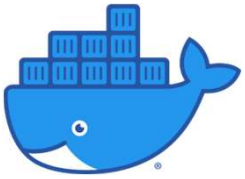
- AWS Fargate can be used with containers on Lambda for even more cost efficiency and operational simplicity.
- Lambda supports container file sizes up to 10GB, allowing powerful use cases such as AI and big data analytics.
- AWS provides base images and a runtime interface emulator to help users integrate containers with Lambda.
- Containers on Lambda may only be suitable for some use cases due to networking limitations or specific dependencies that cannot be packaged.

# Create Basic Function Demo

## Environment

AWS CLI

Visual Studio Code
(or equivalent)



Docker Desktop
(or equivalent)


AWS Console

# Container Demo

Links:

https://docs.aws.amazon.com/lambda/latest/dg/images-create.html#images-create-from-base

https://aws.amazon.com/blogs/aws/new-for-aws-lambda-container-image-support/

# Create a Basic Function Demo

## Environment

AWS CLI

Visual Studio Code
(or equivalent)


AWS Console

# Create a Basic Function Demo

## Links

AWS Command Line Interface Program
https://aws.amazon.com/cli/

Lambda CLI Documentation
https://aws.amazon.com/lambda/latest/dg/gettingstarted-awscli.html