AWS Lambda Beginner to Expert

HELLO!

I'm Daniel

Senior Software Engineer

youtube.com/c/beabetterdev

COURSE INTRODUCTION

Approach, Teaching Format, Content, Pre-requisites

Course Introduction

- Teach you the fundamentals of AWS Lambda
  - Emphasis on the important bits
- Visual Learning
  - Console Walkthroughs
  - Cloud Development Kit for Infrastructure as Code

Course Contents

Introduction

What is a Lambda Function?

A Brief Tour of the Console

How does it work?

Key Concepts

Configuration

Deployments

Monitoring

Advanced Concepts

Invocation Types

Throttling & Concurrency

Performance Tuning

Good to Knows

IaC with CDK

Pricing

Service Integrations

Event Processing & Workflows

APIs & Adhoc Jobs

Timed Jobs

Who Are You?

- Developers
- Data Engineers & Scientists
- Devops
- Other Industry Professional
- Folks starting a new job and looking to ramp up quickly

Pre-requisites

- **Python Fundamentals**
  - Functions, variables primitives, etc...
- **General Understanding of AWS**
- **Python, PIP, AWS CLI, CDK**
- **AWS Account**

The Value of Learning Lambda

- Massive Industry Adoption
- Applications in Many Job Families
- Extensive Community Support
- Lower Cost vs Traditional Bare Metal

Introduction to AWS Lambda

Agenda

- What is AWS Lambda?
- Why is Lambda Useful?
- Disadvantages
- Who Uses Lambda?

History of AWS Lambda

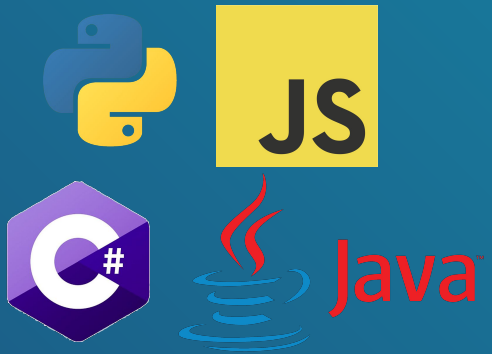
Early 2000s	Mid to Late 2000s	2014+
Traditional Company Specific Data Centers	Early Cloud Infrastructure (EC2)	Enhanced Cloud Infrastructure

What is AWS Lambda?

- **Compute Service**
- **Run code at scale without worrying about servers**
- **You write *Functions* - the primary unit of Lambda**
- **Useful in many applications such as**
  - API Hosting
  - Event Processing
  - Ad-hoc or timer based jobs

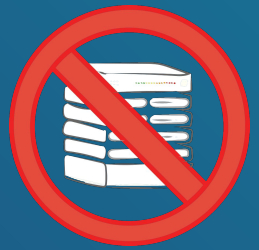
A Typical Workflow

1. **Create a Function**
2. **Write & Upload Your Code**
3. **Run Your Function**

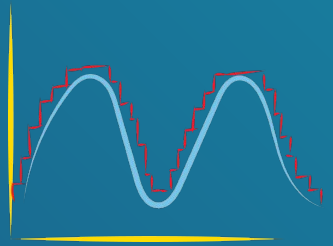


Why is Lambda Useful?

No Servers to Manage



Autoscaling



Pay For What You Use

Performance

Service Integrations

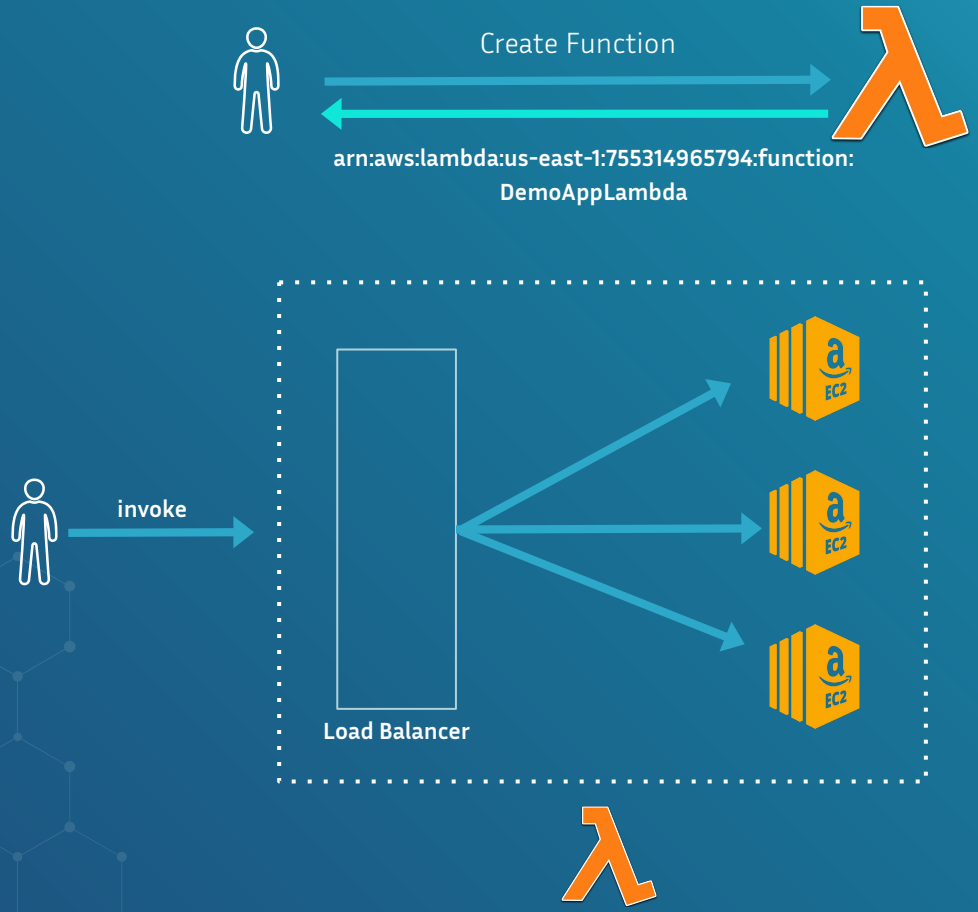
Easy To Use

“With AWS Lambda, you gain flexibility at the expense of control.”

Users of AWS Lambda

How Lambda Works Under The Hood

Under The Hood



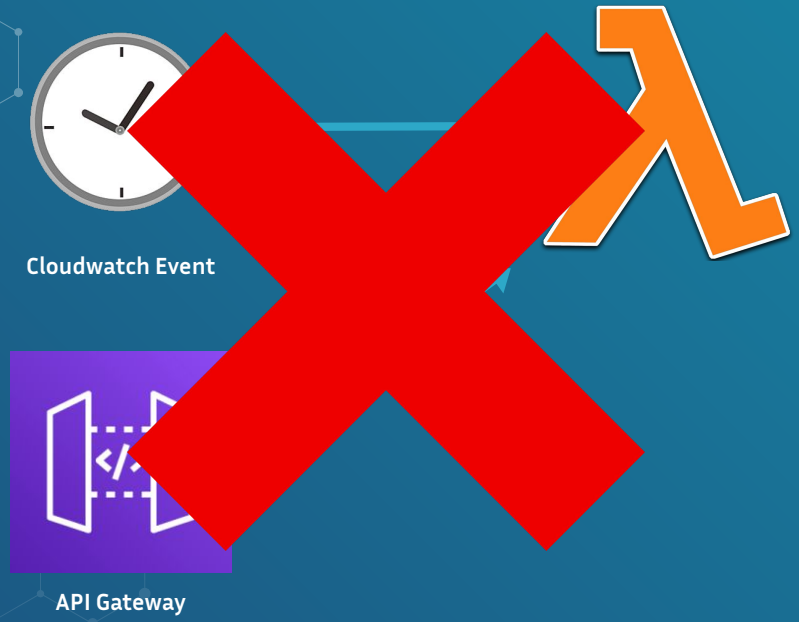
Core Concepts - Triggers


Not like this...

What is a Trigger?

- Means to Invoke your Function
- SDK, CLI, Console, Service Integration
- Can have many triggers
- Similar concept called *Event Source Mapping*
- Different Triggers have different scaling / retry behaviour

Don’t Try to Do This



Common Triggers / Event Source Mappings

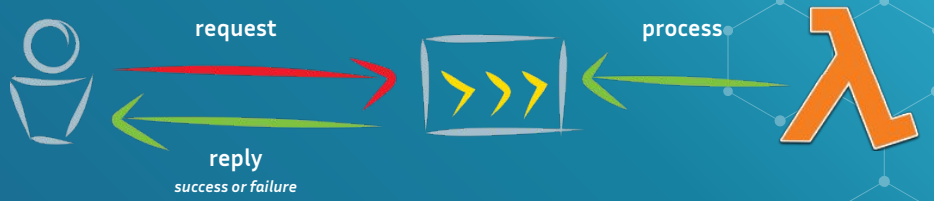
- **Cloudwatch Events** for Timer based jobs
- **API Gateway** for REST APIs
- **SNS** or **SQS** for event processing
- **DynamoDB Streams** for Record Changes
- **S3** for Object Creation / Updates

Core Concepts - Invocation Types

Synchronous and Asynchronous

Synchronous

Asynchronous



How do we handle failure?

Dead Letter Queue

Why does this matter?

- Synchronous can have cost implications
- Asynchronous can be more suitable for fault tolerant use cases
- Different Failure Behaviours

Real Life Example

- Event Processing System With Multiple Lambdas calling each other
- Initially Used Synchronous… Whoops
- Switched to Async and Voila…

Core Concepts - Duration / Timeouts


We all have our Limits

Lambda Duration Limits

- Maximum 15 minutes before automatic timeout
- This means Lambda are not suitable for long running processes
- Lambdas will automatically Timeout when the configured limit is reached
- Default is 3 seconds - Configurable via Console

Pro Tips

- Longer timeouts aren’t always better
- Background (asynchronous) processes are also terminated

Core Concepts - Memory

Who Remembers These?

Lambda Memory

- Defines the amount of memory available to your function at runtime
- Range between 128 MB and 10,240 MB
- Changing Memory is the most significant performance lever available to you
- Increasing memory determines instance type behind the scenes

Pro Tips

- Increasing memory setting can reduce the impact of cold start
- Can also help with Networking Connectivity Speed
- Memory increase doesn’t result in linear performance gain
- You should try and find the sweet spot to balance performance and invocation duration

Side Topic - Pricing

Invocation Count

The number of times you invoke the function

$.20 per 1 Million Requests

Memory

The amount of memory you provision

Memory (MB)	Price per 1ms
128	$0.0000000021
512	$0.0000000083
1024	$0.0000000167
1536	$0.0000000250
2048	$0.0000000333
3072	$0.0000000500
4096	$0.0000000667
5120	$0.0000000833
6144	$0.0000001000
7168	$0.0000001167
8192	$0.0000001333
9216	$0.0000001500
10240	$0.0000001667

Duration

How long your invocations run for

$.0000166667 per GB Second

3 Million Invocations

512 MB Memory

1 Second Duration

$18.74 / month

30 Million Invocations

128 MB Memory

200ms Duration

$11.63 / month

To Estimate Your Costs

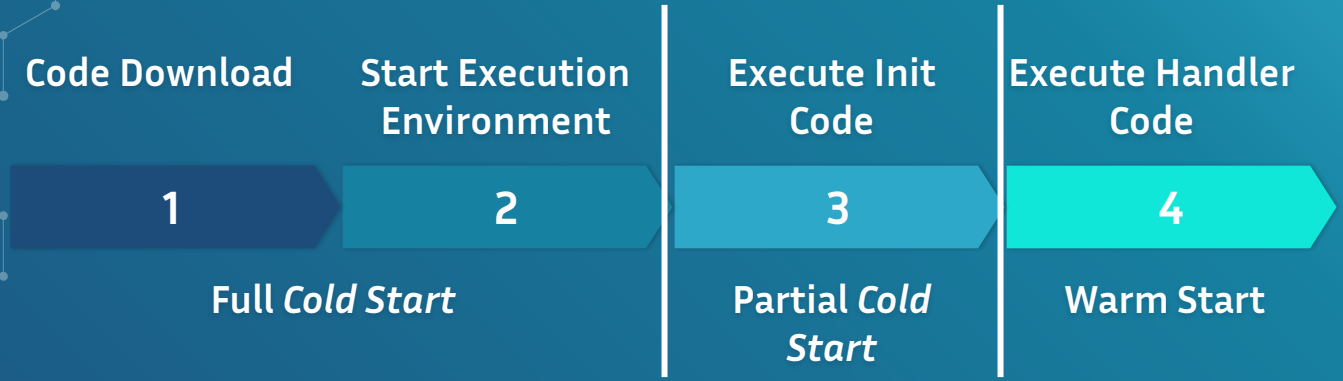
- Use the Cost Calculator!
- https://aws.amazon.com/lambda/pricing/

Pro Tips

- Start with low memory settings
- Once you collect enough invocation history, use the Compute Optimizer Tool
- 1 Million free requests per month and 400,000 GB Seconds

Core Concepts - Function Execution & Cold Start

Decomposing a Function Execution




```
1  //1 - Import required libraries
2  var AWSXRay = require('aws-xray-sdk-core')
3  var captureMySQL = require('aws-xray-sdk-mysql')
4  var mysql = captureMySQL(require('mysql2'))
5  var sequelize = require('sequelize')
6
7  //2 - Extract environment variables from runtime
8  const username = process.env.databaseUser
9  const password = process.env.databasePassword
10 const host = process.env.databaseHost
11
12 //Function Entry Point
13 exports.handler = async (event) => {
14     //3 - Create Database Connection
```

Cold Start also occurs while scaling up

Strategies to Minimize Cold Start

- Minimize number of library dependencies
- Only import what you need
- Raise Memory Configuration
- Utilize Provisioned Concurrency

Example - Querying a Database

```
//1 - Import required libraries
var AWSXRay = require('aws-xray-sdk-core')
var captureMySQL = require('aws-xray-sdk-mysql')
var mysql = captureMySQL(require('mysql2'))
var sequelize = require('sequelize')

//2 - Extract environment variables from runtime
const username = process.env.databaseUser
const password = process.env.databasePassword
const host = process.env.databaseHost

//Function Entry Point
exports.handler = async (event) => {
    //3 - Create Database Connection
    var connection = mysql.createConnection({ //
      host     : host,
      user     : username,
      password : password,
      database : 'lambdadb'
    });

    //4 - Define Query
    var query = "SELECT * FROM Customers";
    var result;

    //5 - Perform Query Using Connection
    connection.query(query, function (error, results, fields) {
      if (error) throw error
      console.log("Ran query: " + query)
      for (result in results)
        console.log(results[result])
    })
}
```

Core Concepts - Concurrency & Throttling


32
BX GIÁP BÁT ⇒ NHỔN

What is Lambda Concurrency?

- # of requests being served at a given moment
- New containers are spawned for each concurrent request
- Concurrency is a major scaling consideration, and can cause applications to fail due to *Throttling*
- Default 1000 units of concurrency per AWS account per region
- Unreserved, Reserved, and Provisioned

Throttling aka *RateExceeded*

- Throttling is when Lambda rejects a request
- Occurs when in flight invocations exceeds available concurrency

Unreserved	Reserved	Provisioned
Common concurrency pool	Dedicated concurrency pool	Dedicated and ‘always on’ concurrency pool

Core Concepts - Reserved & Unreserved Concurrency

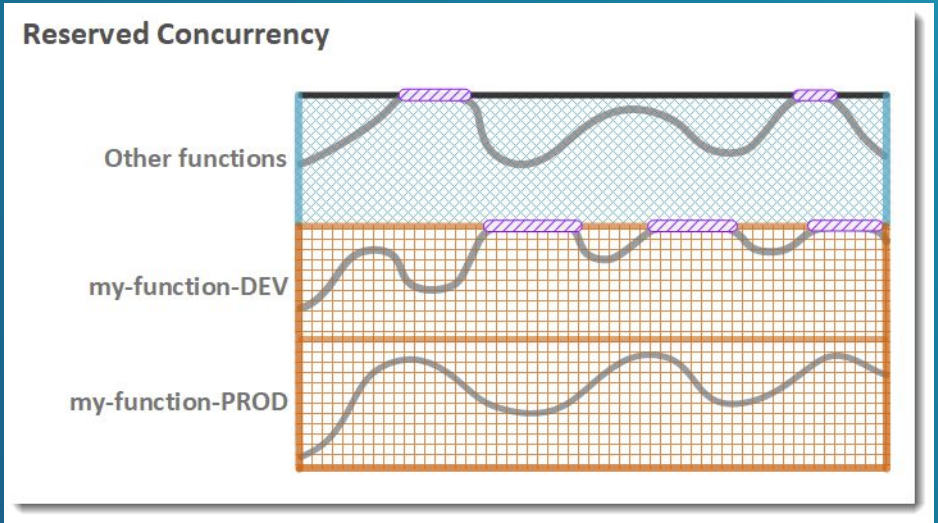
Unreserved Concurrency

- Free for all among all functions in an account per region
- If one function consumes all concurrency, the others will get throttled
- Limit can be raised via AWS Support Ticket

Reserved Concurrency

- A portion is always reserved for a specific function, even if there are no invocations
- You can use reserved concurrency to minimize or maximize your processing rate
- Good for predictable resource allocation
- Bad in terms of utilization of resources

Reserved



Function Concurrency

Reserved Concurrency

Unreserved Concurrency

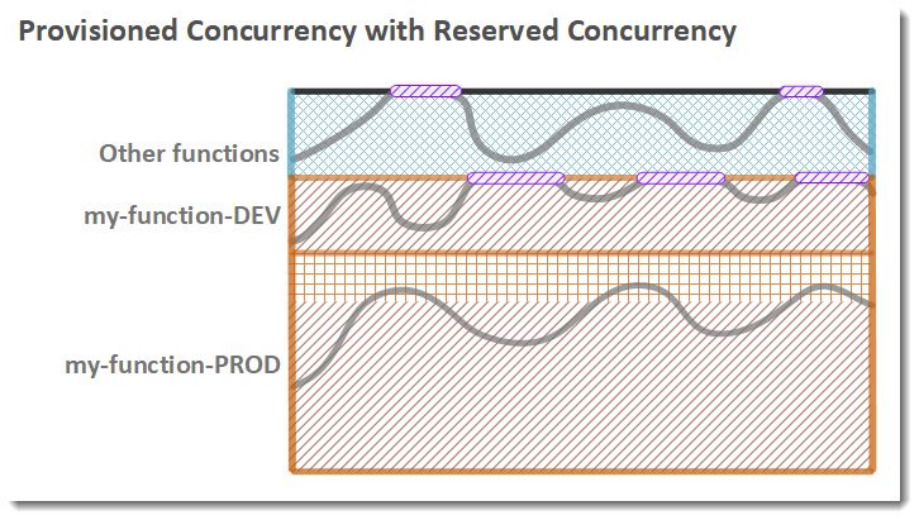
Throttling

Core Concepts - Provisioned Concurrency

Provisioned Concurrency

- Pool of concurrency that is ‘always on’
- Solves the Cold Start problem, *kinda*
- Supports autoscaling policies based on usage
- Very expensive (34$/month for 5 units always on)
- Blurs the line between serverless and metal

Provisioned with Reserved



Function Concurrency

Reserved Concurrency

Provisioned Concurrency

Unreserved Concurrency

Throttling

Pro Tips

- Alarm on throttles for early indicators of issues
- Evaluate your concurrency needs and plan accordingly
- Have your clients use Exponential Backoff to avoid retry storms
- Raising your Memory Limit can help, but *be careful*
- Use a small amount of provisioned concurrency to mitigate cold starts for latency sensitive apps

Core Concepts - Versions


Change

Versions

- Versions are snapshots of the code and settings of a function.
- Code and settings get locked to a version on publish
- Allow you to have multiple ‘versions’ of a function at the same time - useful for quick rollbacks
- New uploads default to $LATEST version

arn:aws:lambda:us-east-1:755314965794:function:DemoAppLambda:1

Versions - Example

V1

V2

V3

Version

1	$LATEST
2	$LATEST
3	$LATEST

Core Concepts - Aliases

Aliases

- An Alias is a named pointer to a specific Version

  arn:aws:lambda:us-east-1:755314965794:function:DemoAppLambda:1

- Useful for ‘beta’ testing, or ‘pre-prod’ environments

Aliases - Example

V1

V2

V3

Version		Alias
1	$LATEST	prod
2	$LATEST	
3	$LATEST	prod

Advanced Concept - Version/Alias Weights

- You can assign weights to different versions using a Version/Alias
- *90% of Traffic to Version 1, 10% to Version 2*
- Useful for version validation prior to full blown deployment
- You need to use Aliases/Versions in order to use *Provisioned Concurrency*

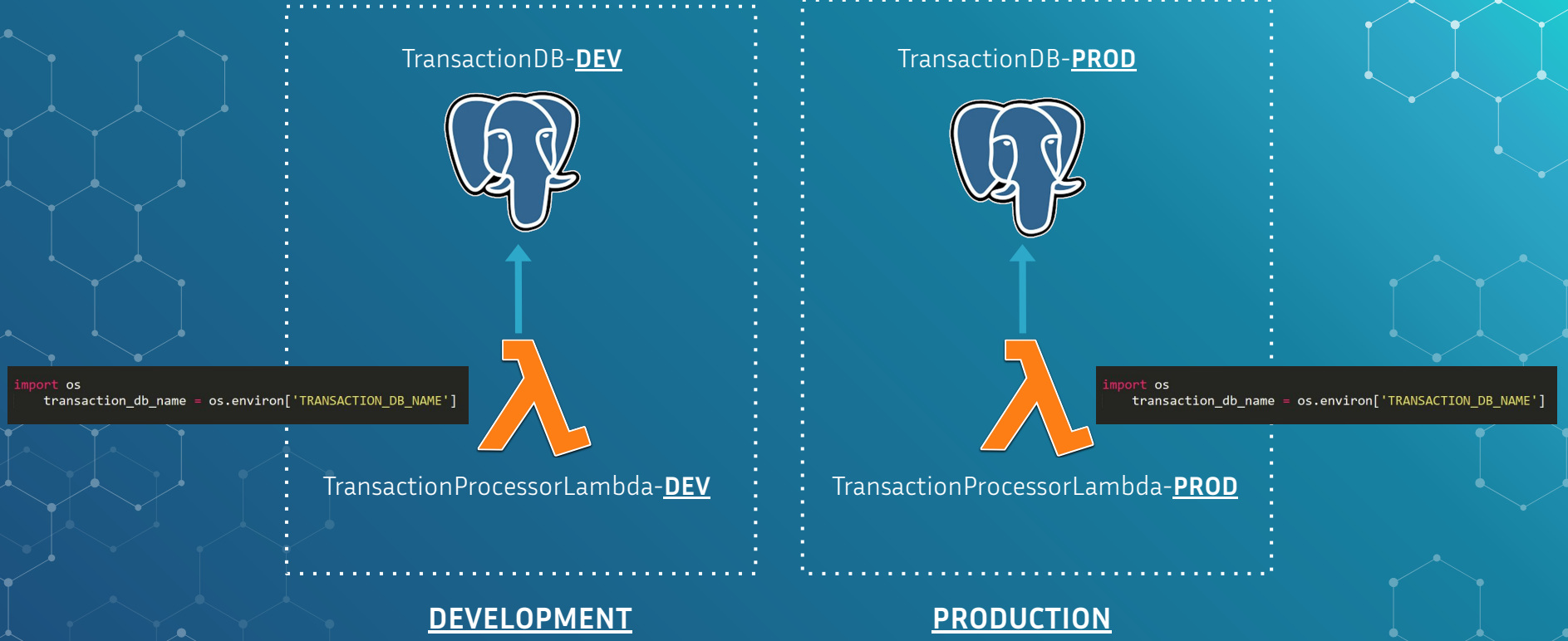
Core Concepts - Environment Variables

Environment Variables

- Pairs of Strings with a key and value
- Allow you to adjust function behavior without touching your code
- *Examples*
  - STAGE
  - CLIENT_ENDPOINT_URL
  - DATABASE_CONNECTION_STRING

```
import os
    db_connection_string = os.environ['DATABASE_CONNECTION_STRING']
    region = os.environ['AWS_REGION']
```

An Example


TransactionDB-DEV
TransactionDB-PROD
import os
transaction_db_name = os.environ['TRANSACTION_DB_NAME']
import os
transaction_db_name = os.environ['TRANSACTION_DB_NAME']
TransactionProcessorLambda-DEV
TransactionProcessorLambda-PROD
DEVELOPMENT
PRODUCTION

Core Concepts - VPC & Networking

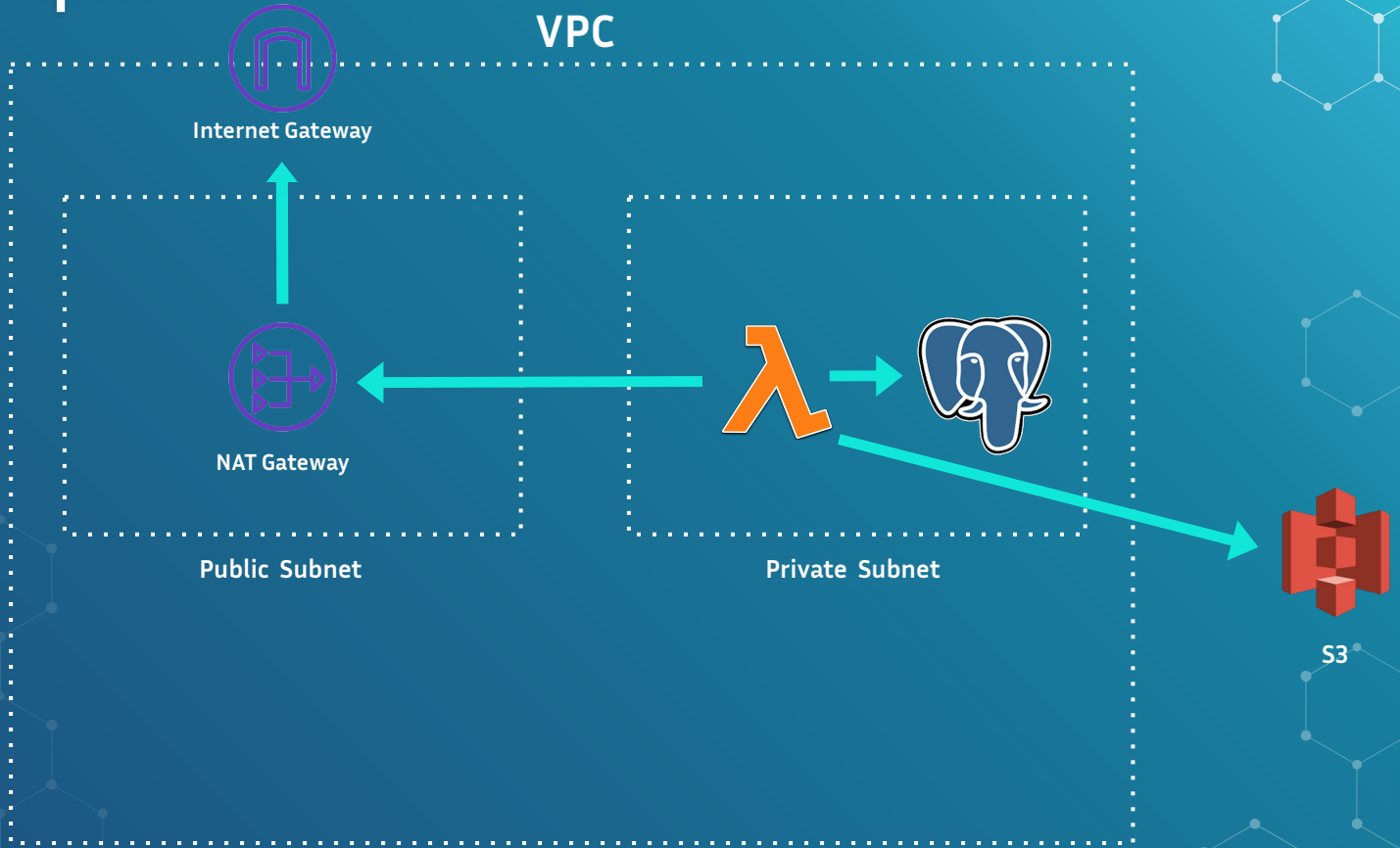
VPC - Quick Refresher

- Stands for Virtual Private Cloud
- Isolated and private network within AWS
- Public and Private Subnets
- *Critical* for security and apps with compliance guidelines

VPC and Lambda

- Only Necessary when your function needs to access resources not accessible over public internet
- Example: RDS or ElasticSearch in private subnet
- Behind the scenes, Lambda creates ENIs for each subnet the function is deployed into
- No longer contributes to cold start latency
- VPC Endpoints can be used to communicate with *some* AWS services privately

Example


VPC
Internet Gateway
NAT Gateway
Public Subnet
Private Subnet
S3

You Specify:

VPC

Subnets

Security Groups

Lambda Creates:

Elastic Network Interface

Exposing Your Function to Other VPCs

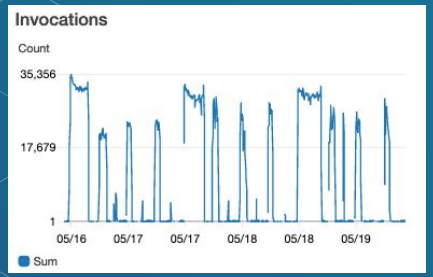
- Use VPC Endpoints to privately connect to AWS Lambda
- Does not expose your function to the public internet
- You can limit *who* from the corresponding VPC can call your function

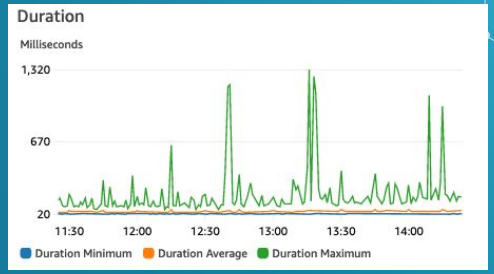
Good To Knows

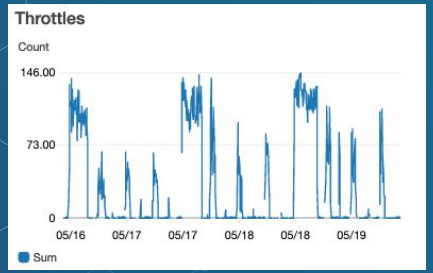
- Launching in VPC can lead to longer startup times due to ENI connection overhead

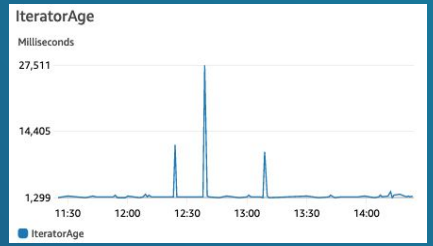
Monitoring - Key Metrics

Metrics to Monitor









+ any other business metrics!

Important Tip

Metrics don’t help if you’re not looking at them

SETUP ALARM ON YOUR METRICS!

Other Notable Metrics

- ConcurrentExecutions
- UnreservedConcurrentExecutions

Monitoring - Logging

Logging

- Writing to stout automatically writes logs to Cloudwatch
- Log lines by default include metadata about the execution

2021-03-06T15:26:51.080-05:00	START RequestId: 1e379bd7-8723-4b12-ab14-efd904a317c2 Version: $LATEST
2021-03-06T15:26:53.712-05:00	* Serving Flask app "app" (lazy loading)
2021-03-06T15:26:53.712-05:00	* Environment: production
2021-03-06T15:26:53.712-05:00	WARNING: Do not use the development server in a production environment.
2021-03-06T15:26:53.712-05:00	Use a production WSGI server instead.
2021-03-06T15:26:53.712-05:00	* Debug mode: off
2021-03-06T15:26:53.772-05:00	* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
2021-03-06T15:26:54.086-05:00	END RequestId: 1e379bd7-8723-4b12-ab14-efd904a317c2
2021-03-06T15:26:54.086-05:00	REPORT RequestId: 1e379bd7-8723-4b12-ab14-efd904a317c2 Duration: 3003.39 ms Billed Duration: 3000 ms Memory Size: 128 MB Max Memory Used: …
2021-03-06T15:26:54.086-05:00	2021-03-06T20:26:54.086Z 1e379bd7-8723-4b12-ab14-efd904a317c2 Task timed out after 3.00 seconds

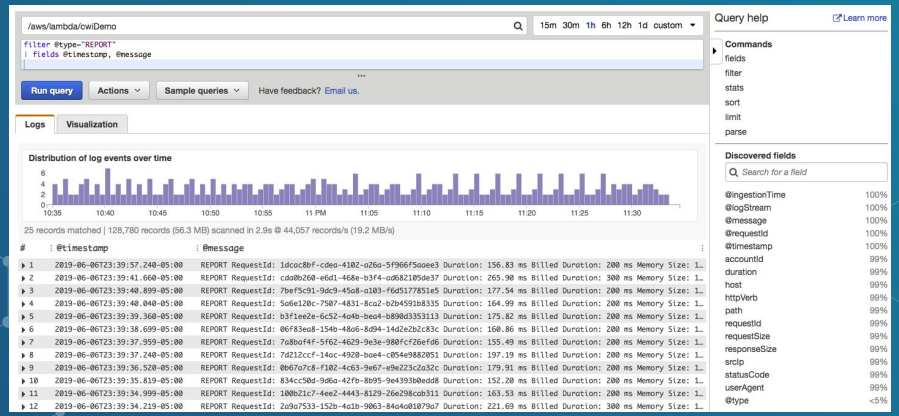
Metric Filters

- Allow you to write patterns (think regex) to extract metrics from log lines
- **Much** cheaper than using PutMetric
- More suitable for high TPS applications

***LATENCY - CreditCardServiceLatency* 990ms**
ERROR - InvalidArgumentException 1

Cloudwatch Logs Insights

- Search for patterns across multiple functions at once
- Pay per use (amount of data scanned)
- Great ‘Example Queries’ that you can build from

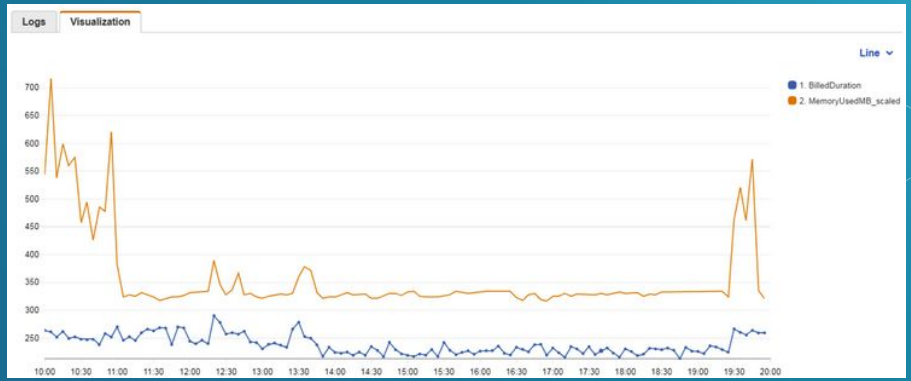


Cloudwatch Logs Insights Examples

```
fields (@maxMemoryUsed/1024/1024) as MemoryMB
| filter @type="REPORT" and (MemoryMB>=62 and MemoryMB<=70)
| stats count(@billedDuration) by MemoryMB, @billedDuration
```

#	MemoryMB	@billedDuration	count(@billedDuration)
1	69.6182	300	7
2	69.6182	400	21
3	69.6182	500	4
4	69.6182	200	16
5	68.6646	1000	1

```
filter @type="REPORT"
| stats avg(@billedDuration) as BilledDuration,
avg(@maxMemoryUsed/1024/102.4) as MemoryUsedMB_scaled by bin(5m)
```



Pro Tips

- Don’t be overly verbose with Logging
- Embed log lines with important ids to make tracing easier
- Use Cloudwatch Logs Insights for fuzzy searching
- Set up a log retention policy or archive regularly

```
[INFO] 2021-08-01 18:00:04 Lambda Execution Start
[INFO] 2021-08-01 18:00:05 PURCHASE Transaction with TransactionId=90a0fce1 and VALUE=10.00
[INFO] 2021-08-01 18:00:05 [TransactionId=90a0fce1] Processing PURCHASE
[INFO] 2021-08-01 18:00:06 [TransactionId=90a0fce1] Saved to DB
[INFO] 2021-08-01 18:00:07 [TransactionId=90a0fce1] Notified Subscribers via SNS
[INFO] 2021-08-01 18:00:08 [TransactionId=90a0fce1] Processing Complete
```

Monitoring - Lambda Insights

What is Lambda Insights?

- Performance Monitoring tool for Lambda
- Collects, aggregates & summarizes system level metrics (CPU, Memory, Network Usage)
- Embedded dashboard

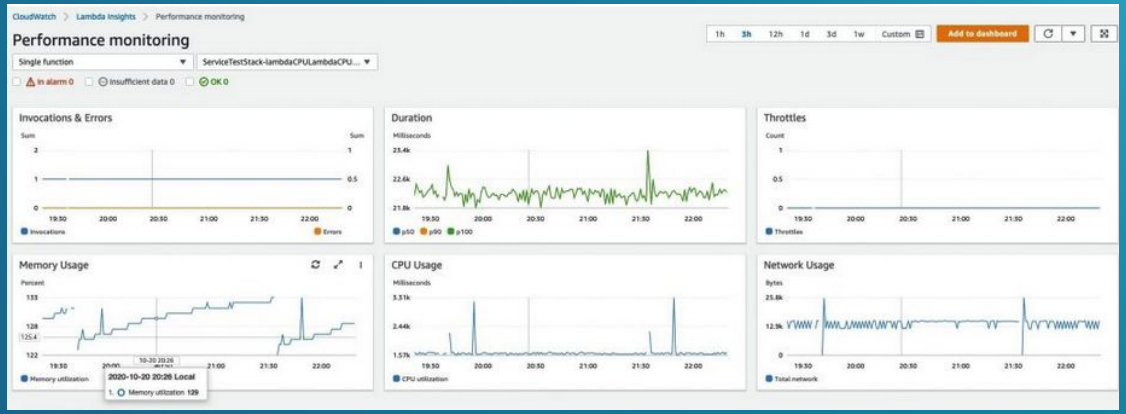
How it works

- Activating it adds a Lambda Extension to your function. Logs get emitted on every invocation.
- Metrics get sent to cloudwatch and extracted with metric filter patterns

Log Entry Example

2021-01-15T10:27:57.336-08:00 {"runtime":"Custom","total_network":84,"used_memory_avg"

```
{
    "runtime": "Custom",
    "total_network": 84,
    "used_memory_avg": 112,
    "request_id": "49d58d40-a0af-4cda-ab11-cf37ca27325d",
    "agent_memory_avg": 7134549,
    "total_memory": 128,
    "cpu_total_time": 370,
    "agent_memory_max": 7778304,
    "used_memory_max": 125,
    "fd_use": 384,
    "tmp_max": 551346176,
    "memory_utilization": 98,
    "function_name": "Services-StepFnlambdasteppriceLessThan556D8B304A-OLERCJ9Y1R7F",
    "version": "$LATEST",
    "agent_version": "1.0.54.0",
    "trace_id": "1-6001de56-4f9b0bff0c00348c2f757547",
    "threads_max": 10,
    "event_type": "performance",
    "fd_max": 18781,
    "tmp_used": 538333184,
    "used_memory_samples": 6,
    "cpu_user_time": 170,
    "tx_bytes": 42,
    "rx_bytes": 42,
```



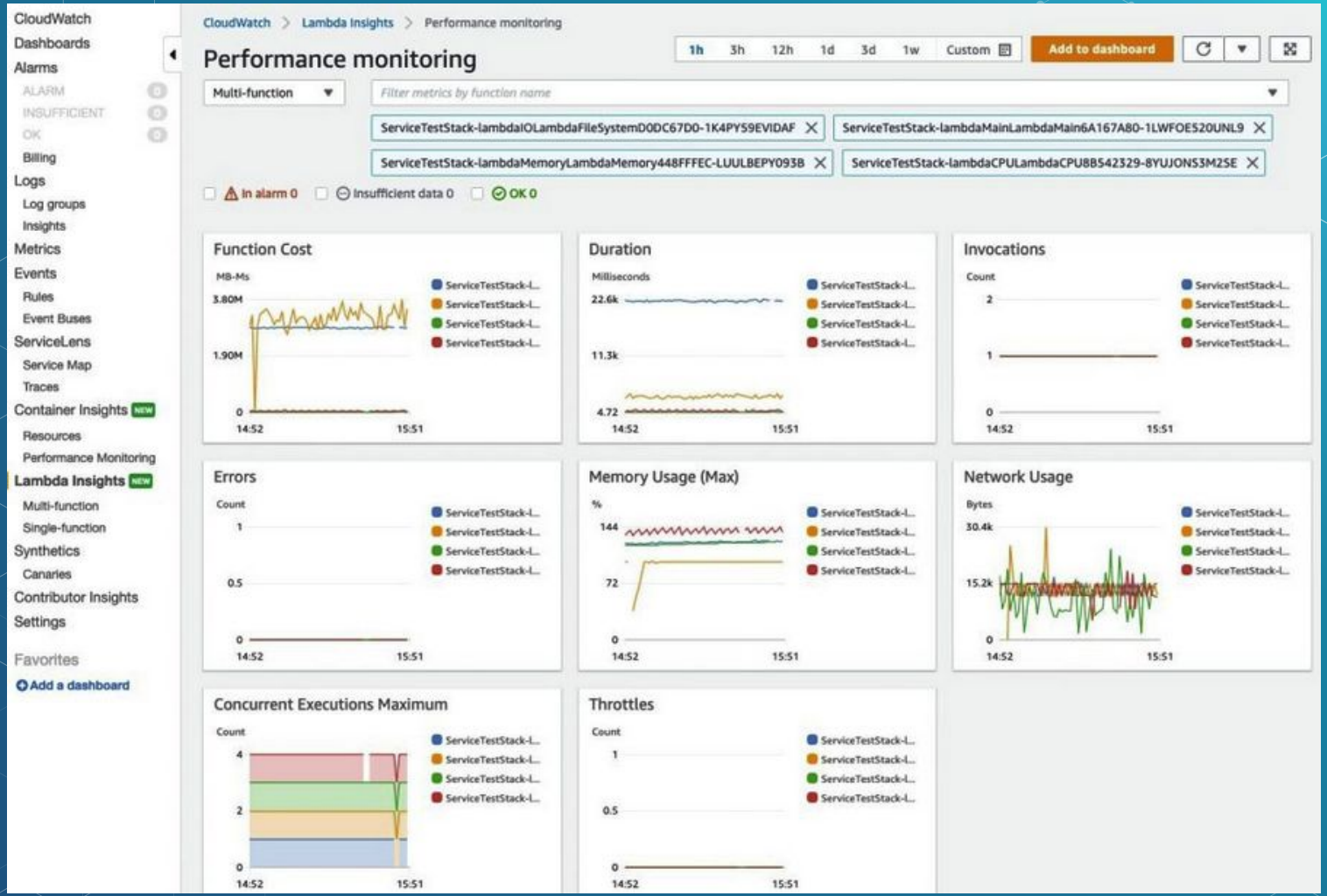
Most recent 1000 invocations (8)

View performance or application logs of this function, or select a few invocations to view logs for selected invocations.

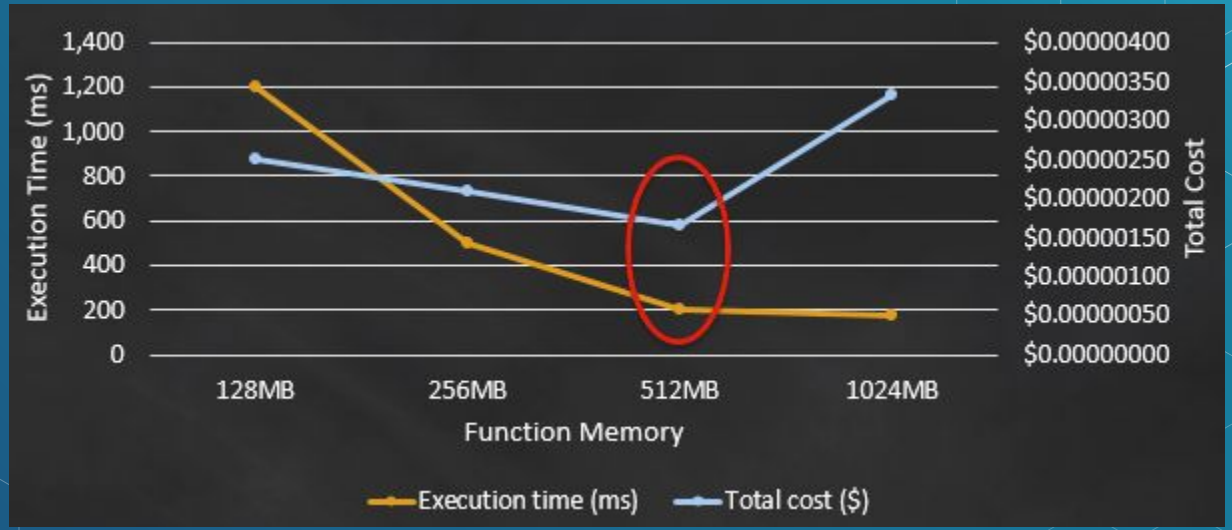
View performance logs | View application logs

1

Timestamp	Request ID	Trace	Error	Init duration	Duration	Memory %	CPU time	Network IO
2021-05-21 01:44:59 (UTC-04:00)	e1bda395-48bf-...	-	-	-	4ms	37%	50ms	1281 bytes
2021-05-21 01:39:23 (UTC-04:00)	c07d588b-c9ce-...	-	-	-	4ms	37%	10ms	6638 bytes
2021-05-21 01:39:23 (UTC-04:00)	38f00008-63e2-...	-	-	-	3ms	37%	10ms	-
2021-05-21 01:39:23 (UTC-04:00)	4d4386d6-b08b...	-	-	-	3ms	37%	20ms	576 bytes
2021-05-21 01:39:22 (UTC-04:00)	f5fb5273-8df3-...	-	-	-	20ms	37%	10ms	4740 bytes
2021-05-21 01:39:22 (UTC-04:00)	6637b76b-679f...	-	-	-	4ms	37%	10ms	3568 bytes
2021-05-21 01:39:21 (UTC-04:00)	7da05d90-41d5...	-	-	-	4ms	37%	-	3333 bytes
2021-05-21 01:39:21 (UTC-04:00)	17d0e85a-cd26-...	-	-	-	16ms	37%	-	10558 bytes


CloudWatch
Dashboards
Alarms
ALARM 0
INSUFFICIENT 0
OK 0
Billing
Logs
Log groups
Insights
Metrics
Events
Rules
Event Buses
ServiceLens
Service Map
Traces
Container Insights NEW
Resources
Performance Monitoring
Lambda Insights NEW
Multi-function
Single-function
Synthetics
Canaries
Contributor Insights
Settings
Favorites
Add a dashboard
CloudWatch > Lambda Insights > Performance monitoring
Performance monitoring
1h 3h 12h 1d 3d 1w Custom
Add to dashboard
Multi-function
Filter metrics by function name
ServiceTestStack-lambdaIOLambdaFileSystemD0DC67D0-1K4PYS9EVIDAF
ServiceTestStack-lambdaMainLambdaMain6A167A80-1LWFOES20UNL9
ServiceTestStack-lambdaMemoryLambdaMemory448FFFEC-LUULBEPY093B
ServiceTestStack-lambdaCPULambdaCPU8B542329-8YUJONS3M2SE
In alarm 0
Insufficient data 0
OK 0
Function Cost
MB-Ms
3.80M
1.90M
0
14:52
15:51
Duration
Milliseconds
22.6k
11.3k
4.72
Invocations
Count
2
1
0
Errors
1
0.5
Memory Usage (Max)
%
144
72
Network Usage
Bytes
30.4k
15.2k
Concurrent Executions Maximum
4
2
Throttles
ServiceTestStack-L...

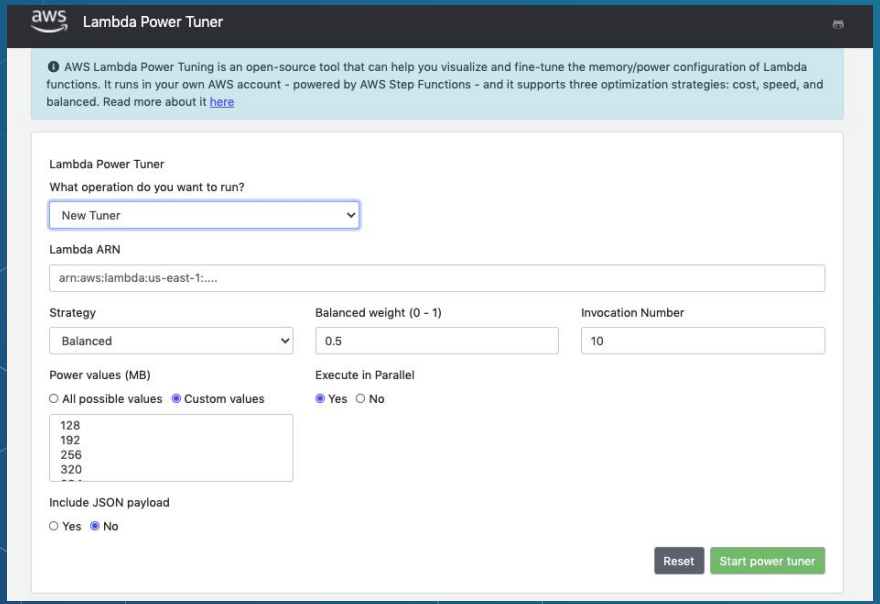
Performance Optimization

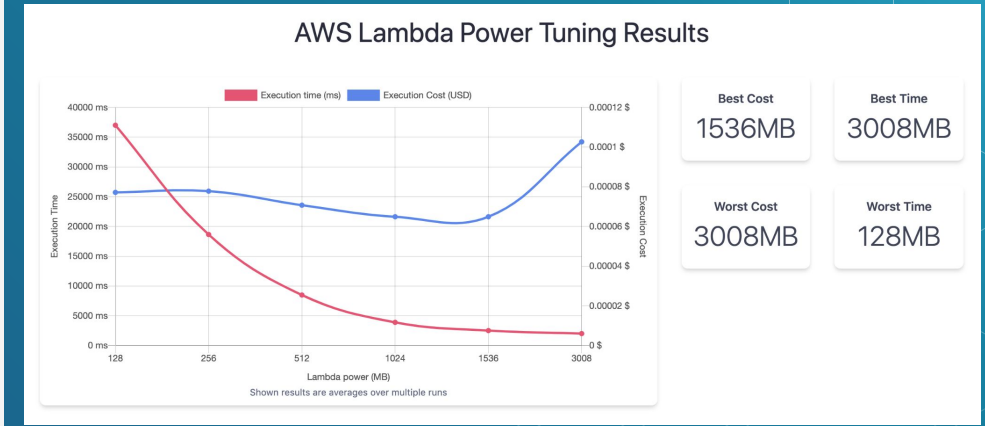


How can we pick the right value?

Option 1: Trial and Error

Option 2: Lambda Power Tuning



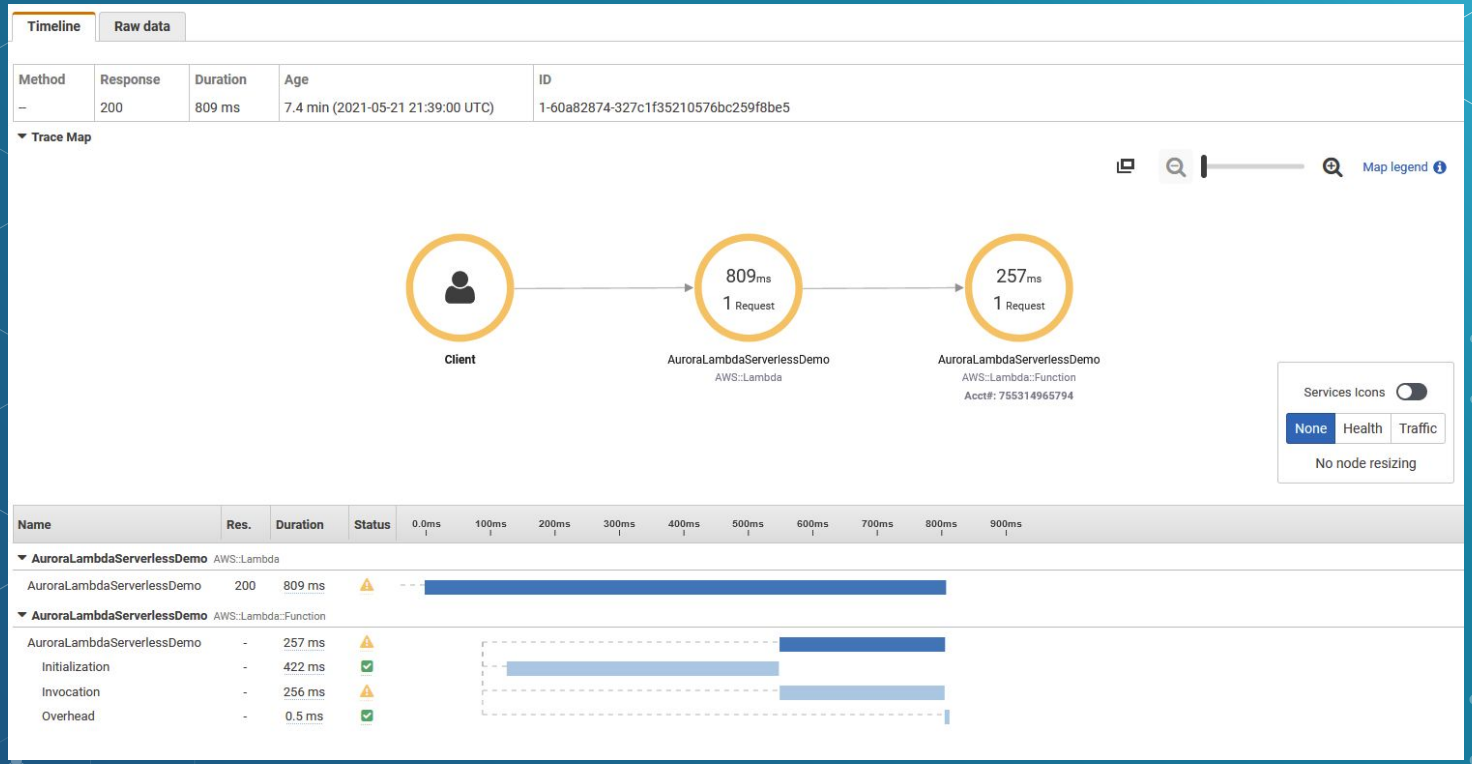


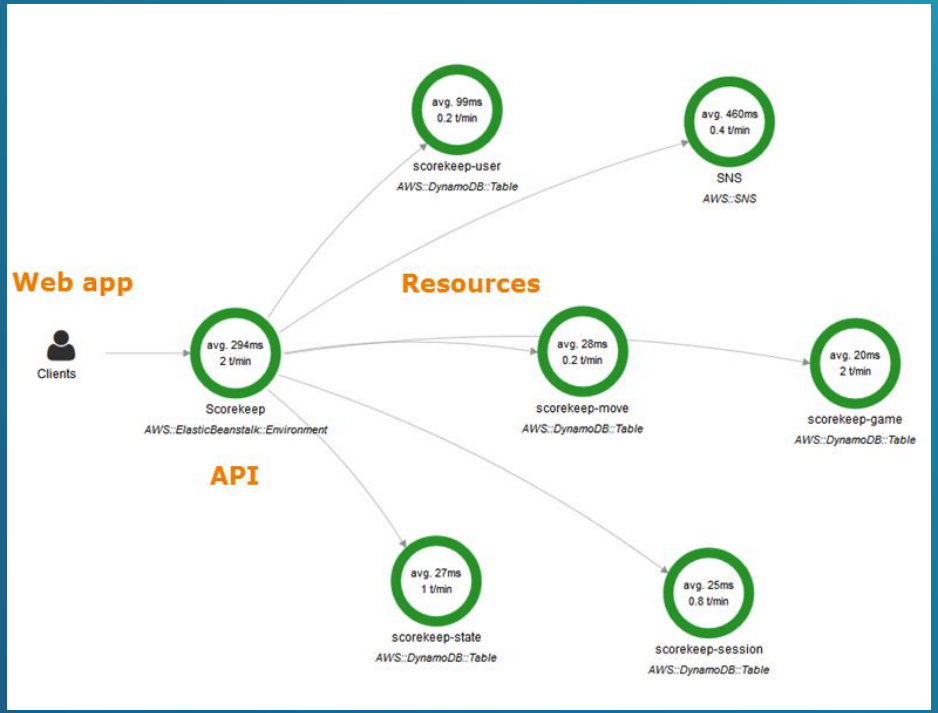
Logging - Hands On Example

Monitoring - X Ray Tracing

What is X Ray Tracing?

- Application level debugging feature
- Generates Trace information on your invocations
- Visualization aid for understanding how much time is spent where
- You can add custom *segments* to your code for function traces


Timeline
Raw data
Method
Response
Duration
Age
ID
–
200
809 ms
7.4 min (2021-05-21 21:39:00 UTC)
1-60a82874-327c1f35210576bc259f8be5
Trace Map
Map legend
809ms
1 Request
257ms
1 Request
Client
AuroraLambdaServerlessDemo
AWS::Lambda
AuroraLambdaServerlessDemo
AWS::Lambda::Function
Acct#: 755314965794
Services Icons
None
Health
Traffic
No node resizing
Name
Res.
Duration
Status
0.0ms
100ms
200ms
300ms
400ms
500ms
600ms
700ms
800ms
900ms
AuroraLambdaServerlessDemo AWS::Lambda
AuroraLambdaServerlessDemo 200 809 ms
AuroraLambdaServerlessDemo AWS::Lambda::Function
AuroraLambdaServerlessDemo - 257 ms
Initialization - 422 ms
Invocation - 256 ms
Overhead - 0.5 ms


avg. 99ms
0.2 t/min
scorekeep-user
AWS::DynamoDB::Table
avg. 460ms
0.4 t/min
SNS
AWS::SNS
Web app
Resources
Clients
avg. 294ms
2 t/min
Scorekeep
AWS::ElasticBeanstalk::Environment
avg. 28ms
0.2 t/min
scorekeep-move
AWS::DynamoDB::Table
avg. 20ms
2 t/min
scorekeep-game
AWS::DynamoDB::Table
API
avg. 27ms
1 t/min
scorekeep-state
AWS::DynamoDB::Table
avg. 25ms
0.8 t/min
scorekeep-session
AWS::DynamoDB::Table

Pro Tips

- Its expensive
  - $5.00 / 1 million traces ***recorded***
  - $0.50 / 1 million traces ***retrieved***
  - $0.50 / 1 million traces ***scanned***
- More useful for ad-hoc debugging
- Consider using sampling

Advanced Concepts - DLQ

What is a DLQ?

- **D**ead **L**etter **Q**ueue
- Sometimes dependencies fail and messages cannot be processed...
- DLQ is a destination to temporarily hold failed invocation message content
- After a number of attempts, messages can be sent to the DLQ for you to act on

Example

Method 1

1. invoke

2. On failure, send to queue

Amazon Simple Queue Service DLQ

4. Invoke

3. Once issue is fixed, poll queue for failed messages

Method 2

EventSource Queue

1. invoke

2. On failure, lamda automatically sends to queue

EventSource **DLQ**

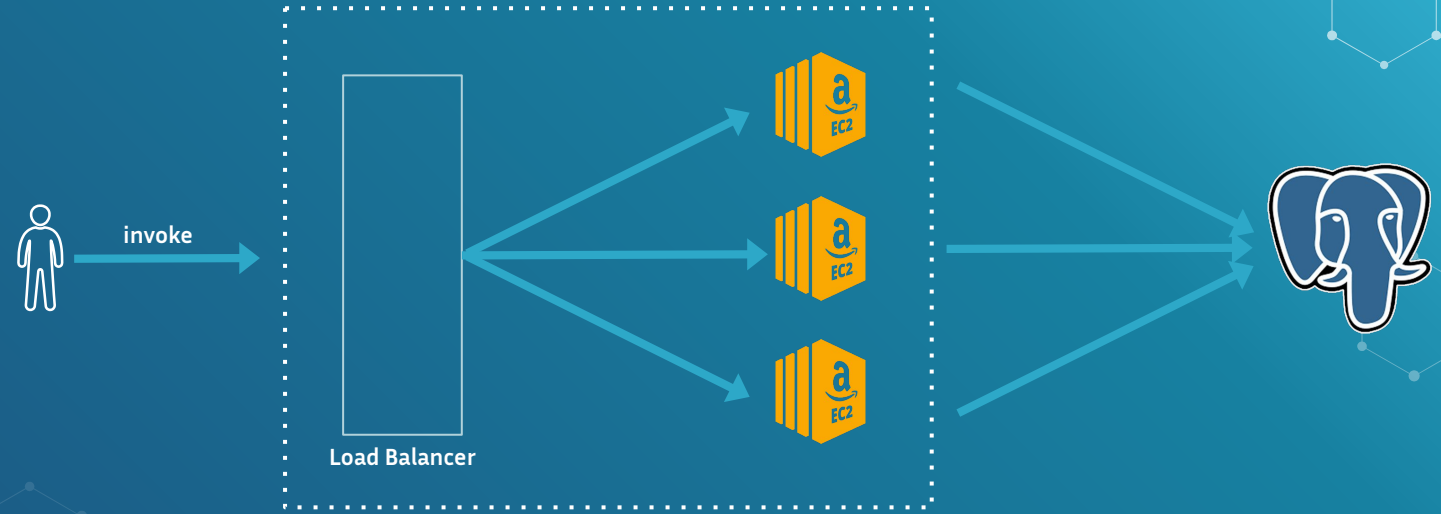
3. Once issue is fixed, transfer messages back to main queue

Best Practices

- Always use a DLQ for production applications
- Alarm on DLQ size > 0
- Be Proactive - have a process for re-driving messages

Advanced Concepts - RDS Proxy

Remember This?



Remember This?

RDS Proxy Feature

- Connection Pool Manager for RDS that integrates directly with Lambda
- Can be used with RDS MySQL, Postgres or Aurora
- If deciding to *not* use this feature, be careful with connection pool management

Good to Knows - Lambda Layers and Filesystem

Lambda Layers

- Allows you to easily add external libraries, data, config files to your function
- Zip the package contents and upload via console
- Only usable when uploading your function code as a .zip, and only 5 per function

Lambda Filesystem

- You can mount an Elastic File System volume to make its contents accessible to your function

Good to Knows - Lambda & Docker

Lambda and Docker

- As of December 2020, you can deploy your functions using a Docker image
- 10GB size limit
- Base images are provided

Good to Knows - Continuous Deployment

Lambda CI / CD

- For larger projects, you may want to consider CI / CD pipelines
- Integrate with Github/CodeCommit and Codebuild for an automatic build system
- Leverage CodePipeline to orchestrate the changes and update your function

Example CI/CD Pipeline

buildspec.yml

AWS SAM

AWS CDK

AWS CodeBuild

AWS CloudFormation

AWS Lambda

AWS CodePipeline

Practical Content - Overview

What's Next?

- Pre-requisites: AWS CLI, CDK, PIP
- **Console:**
  - Cloudwatch Event + Lambda
  - HTTP API + Lambda
- **CDK:**
  - SQS + Lambda