Serverless on AWS

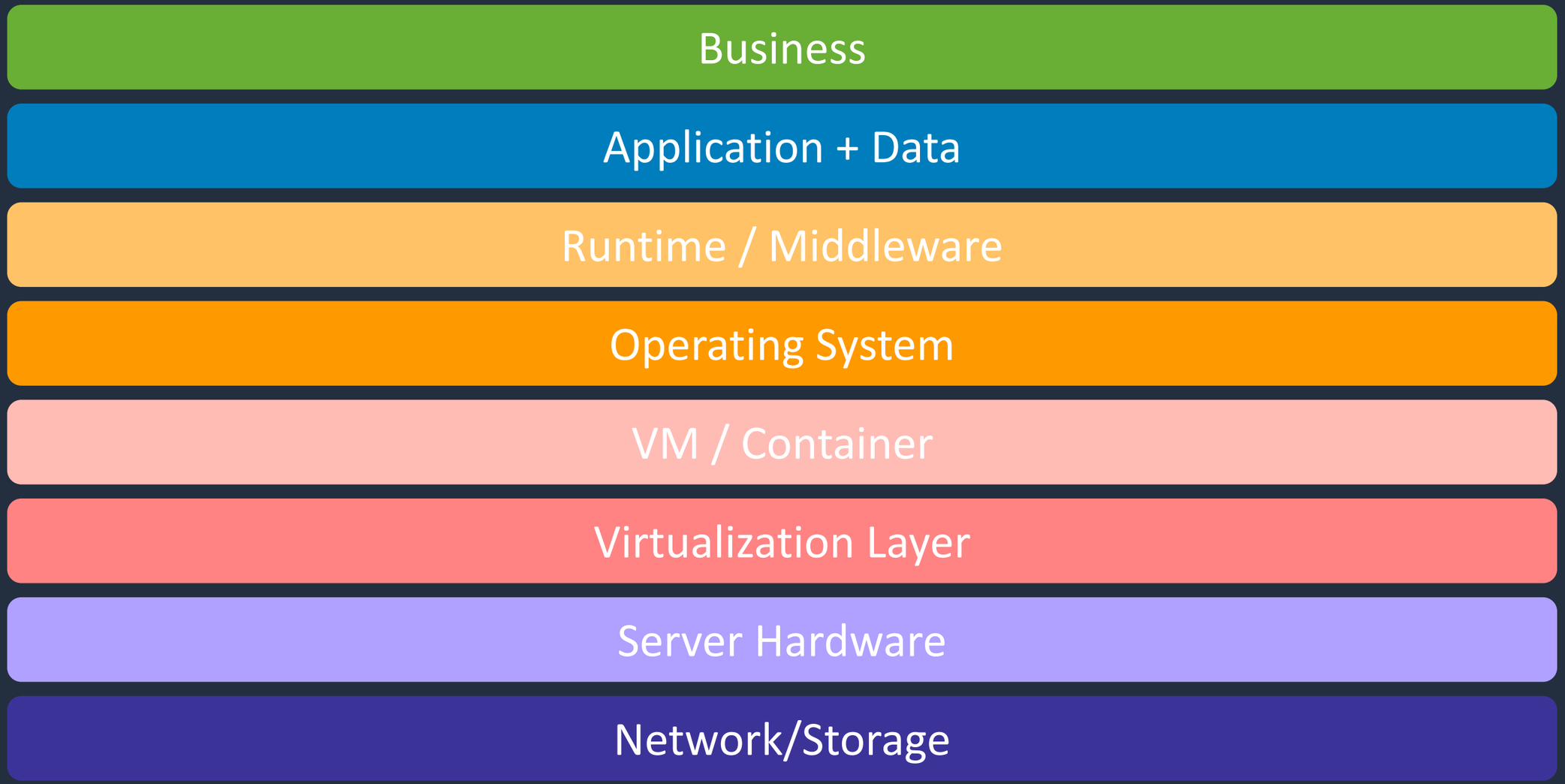
Immersion Day

Jason Hoog, AWS Solutions Architect

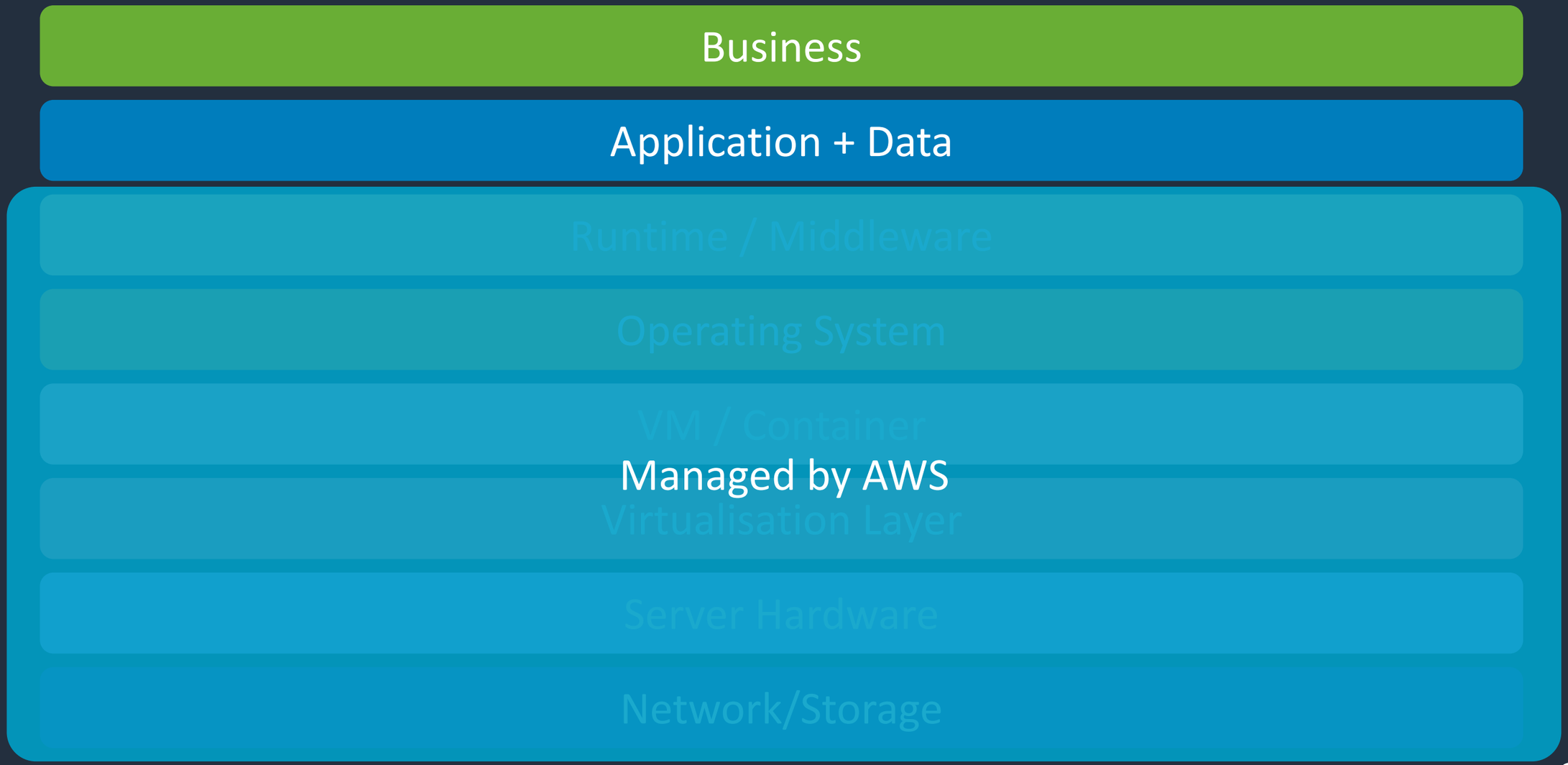
2021

Serverless Observability

Traditional monitoring layers



Serverless monitoring layers



Monitoring is more than watching for failures

Is it behaving as expected?

What is the usage?

What is the business impact?

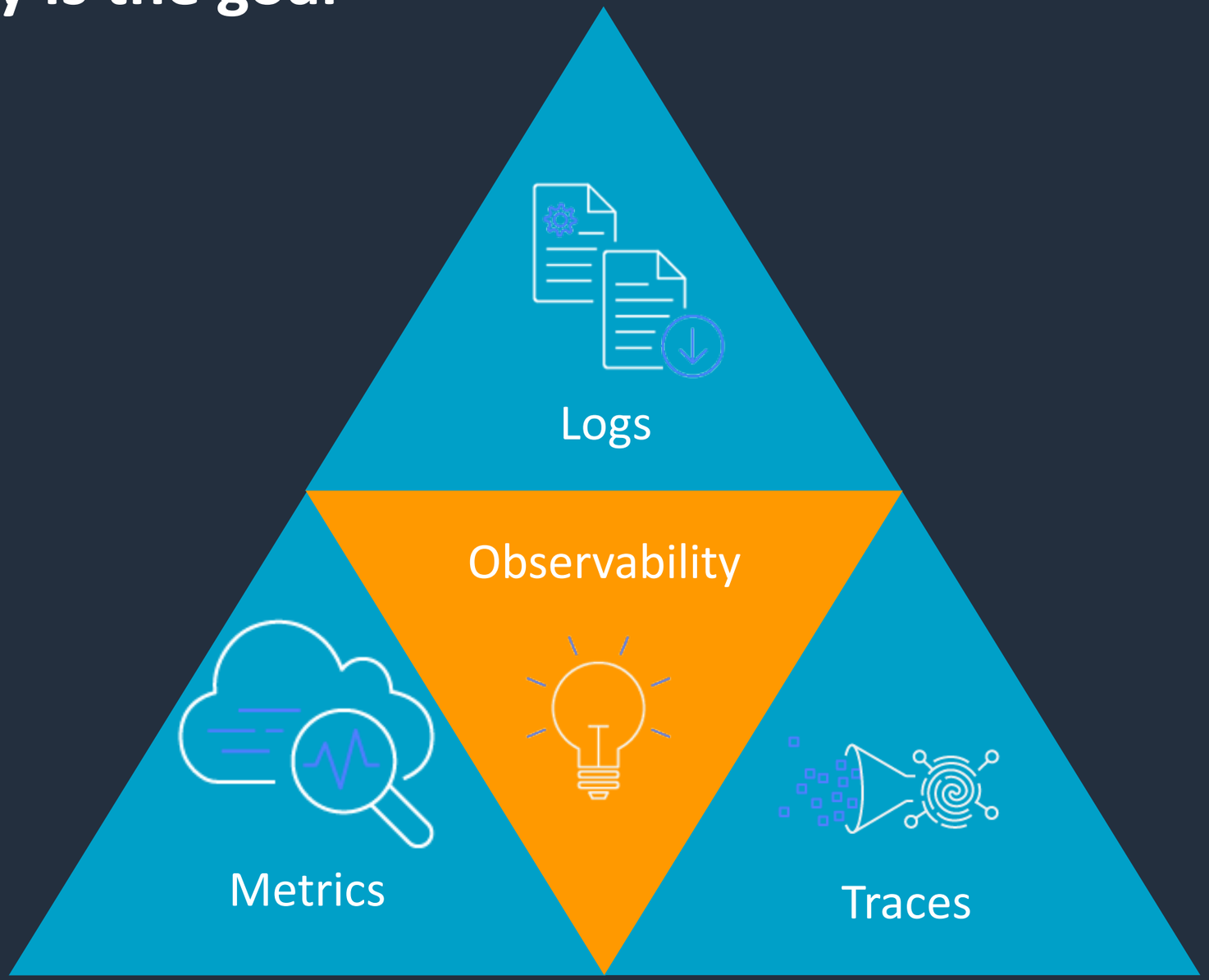
Three pillars of observability

Metrics	Logs	Traces
Numeric data measured at various time intervals (time series data); SLIs (request rate, error rate, duration, CPU%, etc.)	Timestamped records of discrete events that happened within an application or system, such as a failure, an error, or a state transformation	A trace represents a single user's journey across multiple applications and systems (usually microservices)

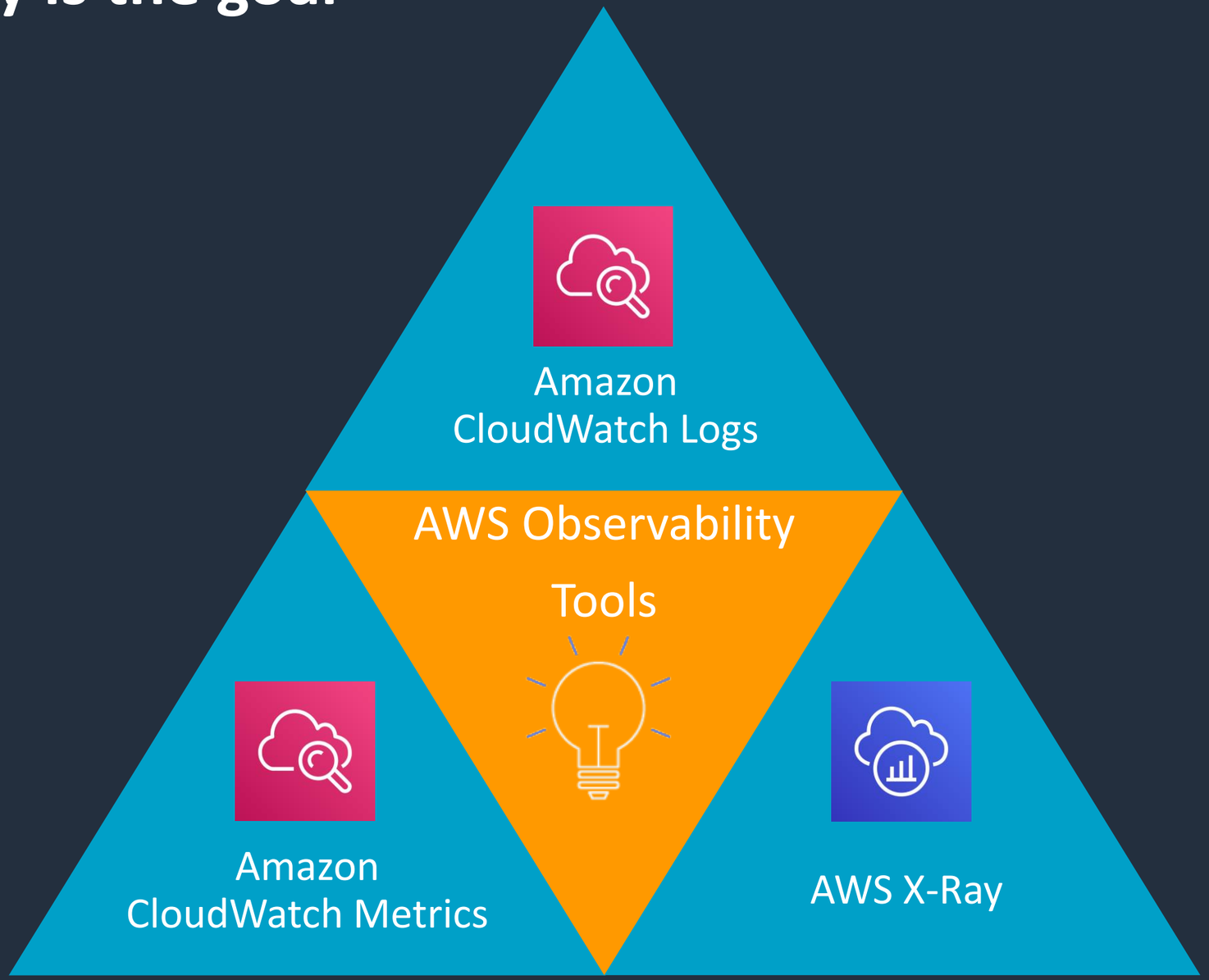
Definitions from: Distributed Systems Observability
https://www.oreilly.com/library/view/distributed-systems-observability/9781492033431/

aws

Observability is the goal



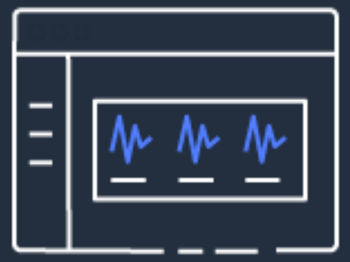
Observability is the goal



Breadth and depth of CloudWatch and X-Ray

Collect

- Embedded Metric Format
- Metric Filters
- StatsD & CollectD
- AWS PrivateLink



Monitor

- Cross-Account, Cross-Region Dashboards
- Automatic Dashboards
- Metric Math
- SQS and SNS add support for X-Ray

Act

- Synthetics
- Anomaly Detection
- Metric Math Alarms
- Search Expressions

Analyze

- ServiceLens
- Contributor Insights
- Container Insights
- Logs Insights
- X-Ray Analytics

aws

AWS Lambda-ready monitoring partners

aws

Metrics & Alarms

Serverless services publish metrics to CloudWatch

AWS Lambda

- Invocation metrics
  - Invocation Count, Errors
  - Throttles
  - Provisioned Concurrency Invocations + Spillover
- Performance metrics
  - Duration
  - Iterator age
- Concurrency metrics
  - Concurrent executions
  - Provisioned Concurrency utilization

Amazon API Gateway

- REST API / HTTP API
  - Request count
  - Latency, Integration Latency
  - Errors: 4xx, 5xx
  - Cache hit / miss (REST only)
  - Data processes (HTTP only)
- WebSocket API
  - Connect, message counts
  - Integration latency
  - Error: Client, Integration, Execution

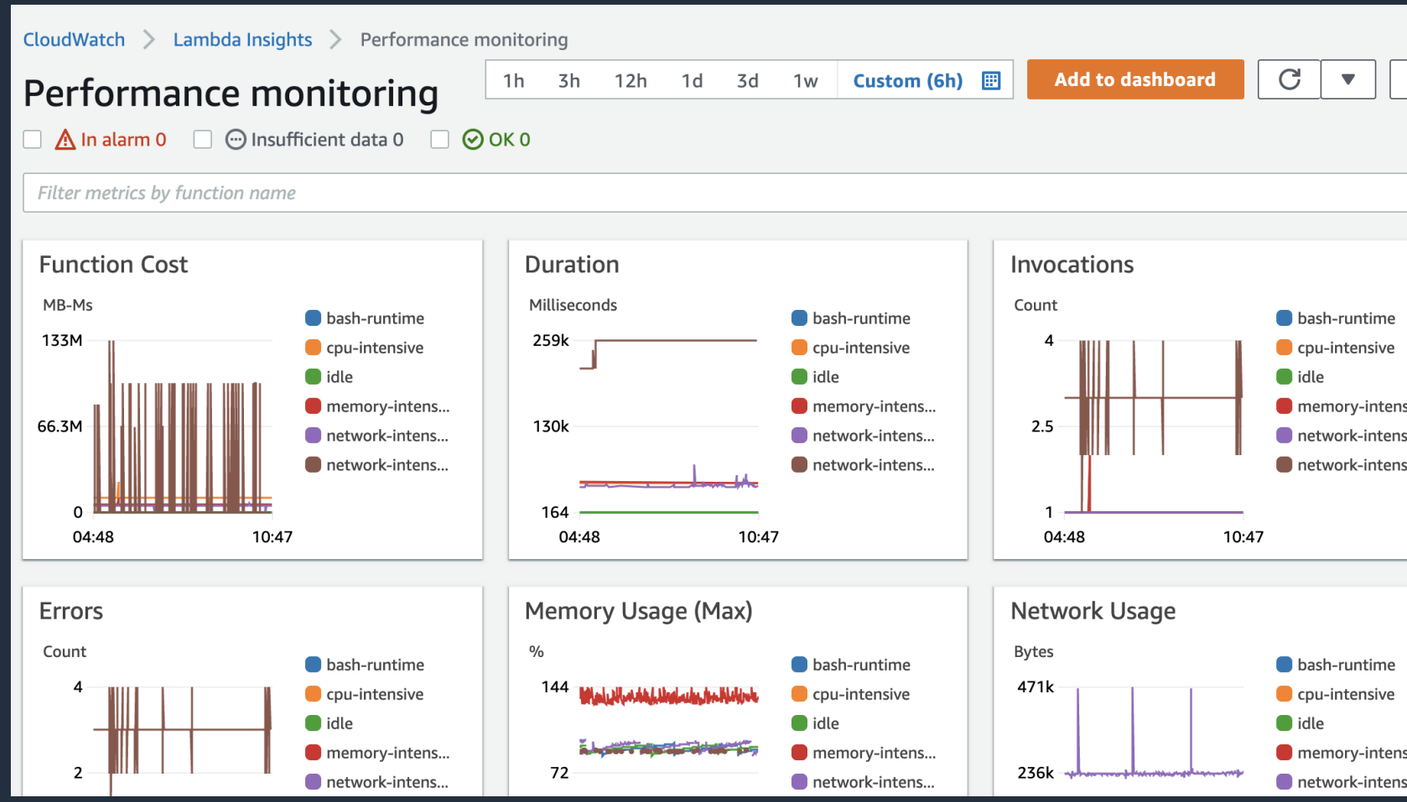
Not exhaustive, subject to change

aws

CloudWatch Lambda Insights adds deeper insight

Preview

- Collect and summarize performance metrics
- Drill down into metrics and errors for Lambda functions
- Review function cost, duration, memory usage, errors, etc.
- Easy to enable



Take action via CloudWatch Alarms

AWS Lambda

- Error rate
- Throttle count
- Dead Letter Queue count (async)
- Iterator age (e.g. Kinesis, SQS, DDB Streams)
- Concurrent executions (Regional, across all funcs)

Amazon API Gateway

- Success rate
  - Example: > 99% return 2xx
- Latency
  - Tail p90/p95/p99, threshold by SLA

Amazon SQS

- Message age
  - Message flow rate is a good indicator of health
  - Message in should equal message out

Structured Logging

```
[2021-02-23T19:59:07Z]  INFO    Request started
[2021-02-23T19:59:07Z]  ERROR   AccessDenied: Could not access resource
[2021-02-23T19:59:08Z]  INFO    Request finished
```


aws

Structured logging augments log messages with data

- Annotate log messages with additional data / context
- Useful in troubleshooting
- Easier for machine and human to parse, derive insights
- Often JSON, but key=value also an option

aws

Easily log to CloudWatch Logs

```
message = {
  "TimeStamp": "2023-02-24 13:08:22",
  "LogLevel": "INFO",
  "Message": "New item in cart",
  "QuantityInCart": 2,
  "ProductId": "123456",
  "CartId": "abcefege"
}

# Python
print(json.dumps(message))

// Node.js
console.log(JSON.stringify(message));

// Go
enc := json.NewEncoder(os.Stdout)
enc.Encode(message)

// Java
class JSONLoggerInitializerFactory...
```


aws

Custom Metrics

Use Embedded Metrics Format for custom metrics

- CloudWatch Embedded Metrics Format (EMF) generates metrics with a simple `print` statement
- Append metrics, dimensions, and annotations
- Faster, less expensive than `PutMetricData`
- Client libraries available for Python and Node.js

```
{
  "_aws": {
    "Timestamp": 1574109732004,
    "CloudWatchMetrics": [
      {
        "Namespace": "lambda-function-metrics",
        "Dimensions": [["functionVersion"]],
        "Metrics": [
          {
            "Name": "time",
            "Unit": "Milliseconds"
          }
        ]
      }
    ]
  },
  "functionVersion": "$LATEST",
  "time": 100,
  "requestId": "989ffbf8-9ace-4817-a57c-e4dd734019ee"
}
```

Publish custom metrics with EMF for…

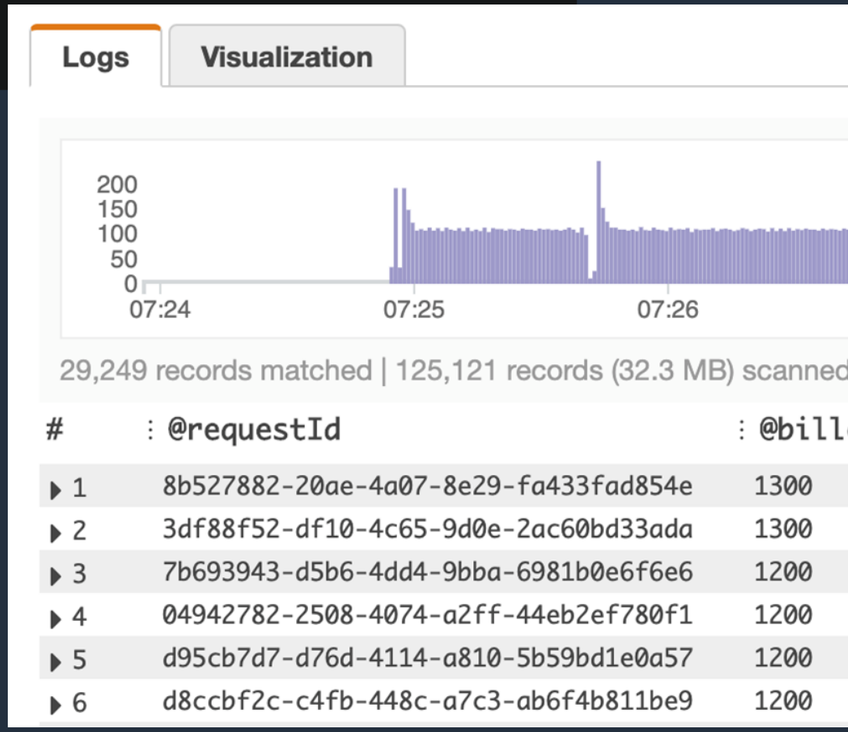
- Business / customer metrics
  - Revenue, sign-ups, page view, etc.
- Operational metrics
  - Deployment feedback time, on-call pages, etc.
- Error metrics
  - Exceptions thrown in Lambda functions, etc.
- Other information
  - Category, item, environment, other dimensions / context

aws

Search & analyze logs with CloudWatch Logs Insights

- Interactively query logs groups to filter and analyze logs
- Pre-built queries for common Lambda analysis, e.g.:
  - Latency statistics for trailing five minutes
  - Find overprovisioned memory
  - Most expensive requests

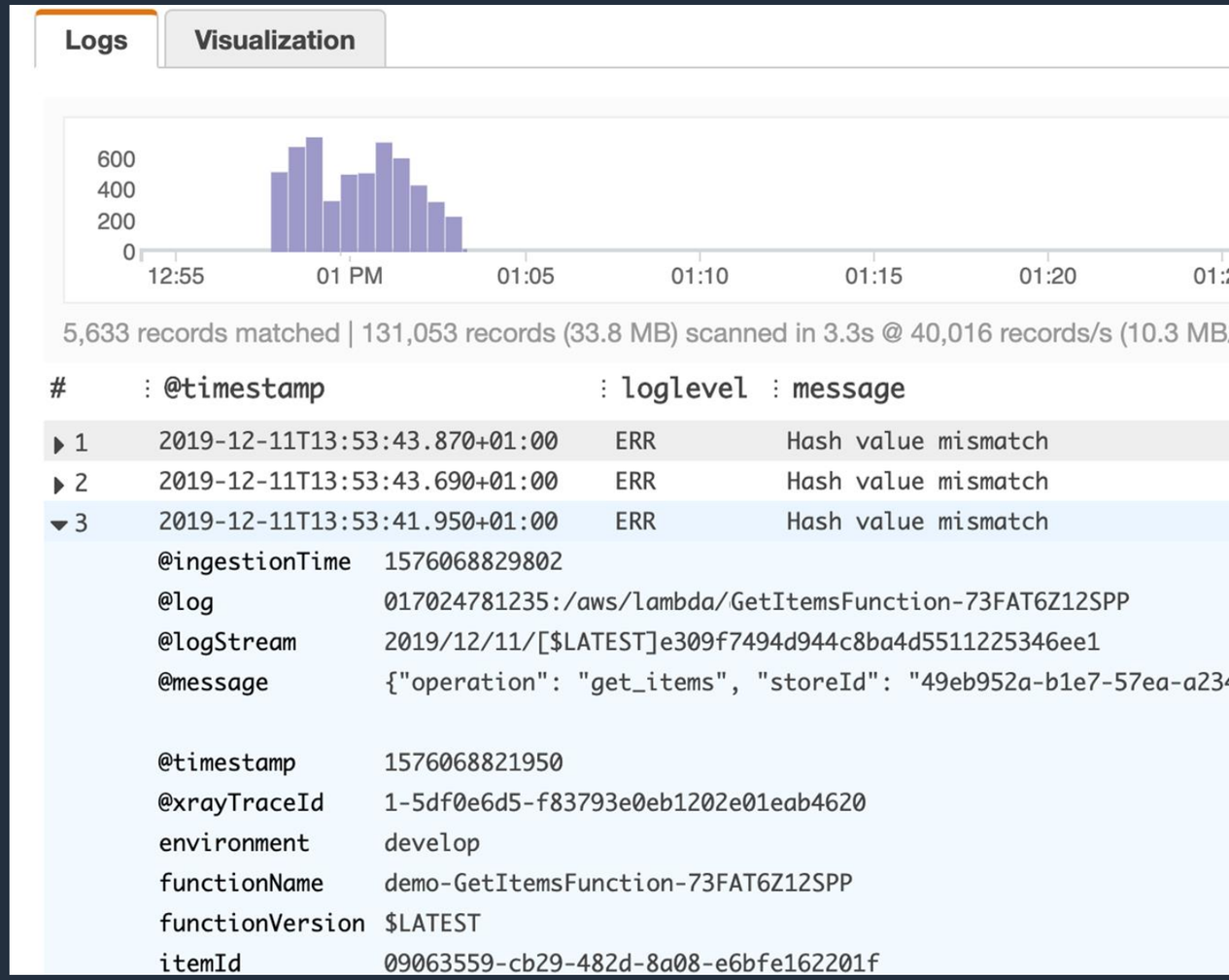
```
# Top 100 most expensive executions
filter @type = "REPORT"
  | fields @requestId, @billedDuration
  | sort by @billedDuration desc
  | limit 100
```



Quickly identify and diagnose errors

- CloudWatch Logs Insights can help quickly retrieve error messages across Log Groups
- Annotations provide additional context

```
# Get the last 10 error messages
fields @timestamp, loglevel, message
  | filter loglevel == "ERR"
  | sort @timestamp desc
  | limit 10
```



Tracing

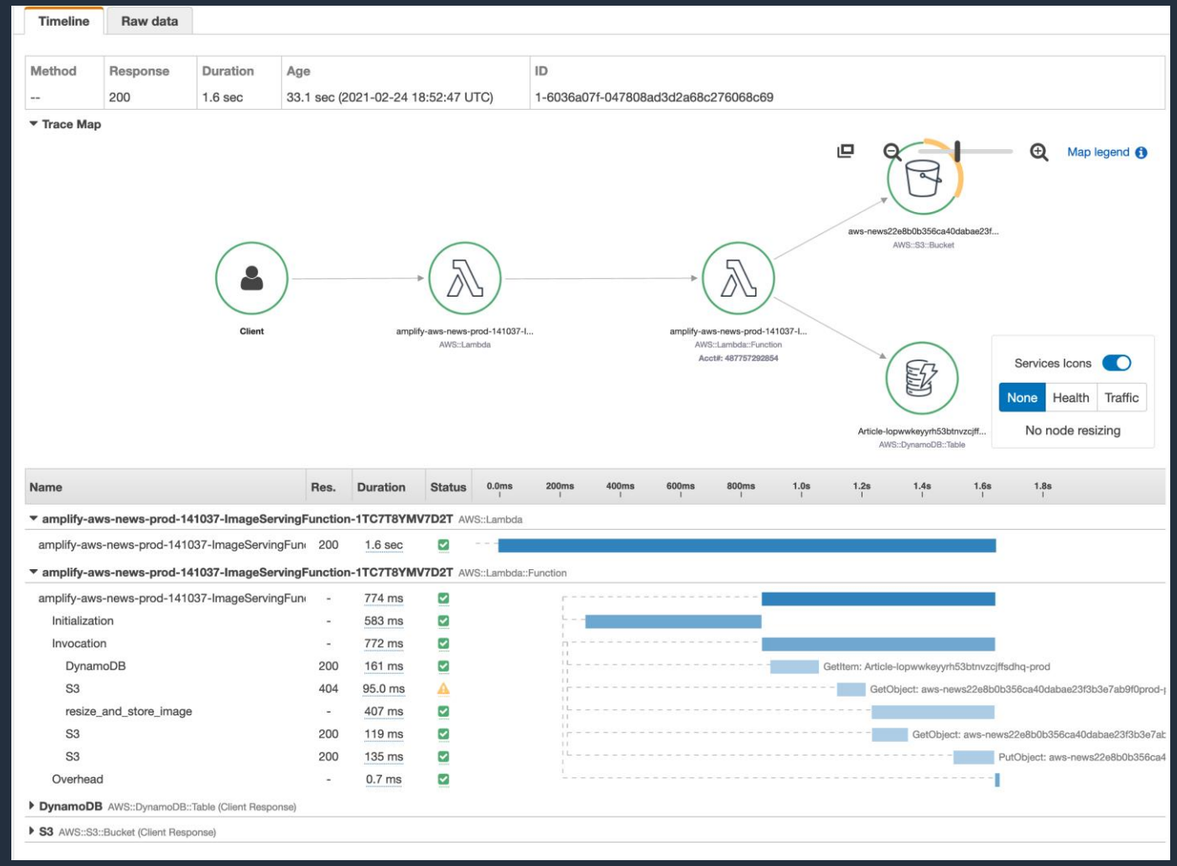
AWS X-Ray enables tracing of distributed applications

- Scales for microservice and serverless architectures
- Identify the root cause of performance issues and errors
- X-Ray provides a cross-service view of requests made to application
  - Aggregates data across services into a trace (via a passed trace identifier)
  - Support for Lambda, API Gateway, SNS, Step Functions, DynamoDB, S3, etc.

AWS X-Ray

AWS X-Ray enables tracing of distributed applications

- To use X-Ray with Lambda:
  - Enable X-Ray tracing in Console
  - Requires IAM permissions for X-Ray
- Instrument function using X-Ray SDK to extend trace
  - SDK for Python, Node.js, Go, Java, etc.




```
const AWSXRay = require('aws-xray-sdk-core');
const AWS = AWSXRay.captureAWS(require('aws-sdk'));

const s3Client = new AWS.S3();
```

Bring best practices with AWS Lambda Powertools

Suite of utilities for AWS Lambda functions to ease adoption of best practices:

Logging: output as structured JSON

Tracing: send traces to AWS X-Ray

Metrics: custom metrics with embedded metric format

Utilities: parameters, Amazon SQS batch processing, etc.

Python

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

Java

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

aws

Thank you!

https://aws.amazon.com/serverless/