Observability for your serverless applications

Julian Wood
Sr. Developer Advocate – AWS Serverless
@julian_wood

© 2020, Amazon Web Services, Inc. or its Affiliates.

What I’m covering today

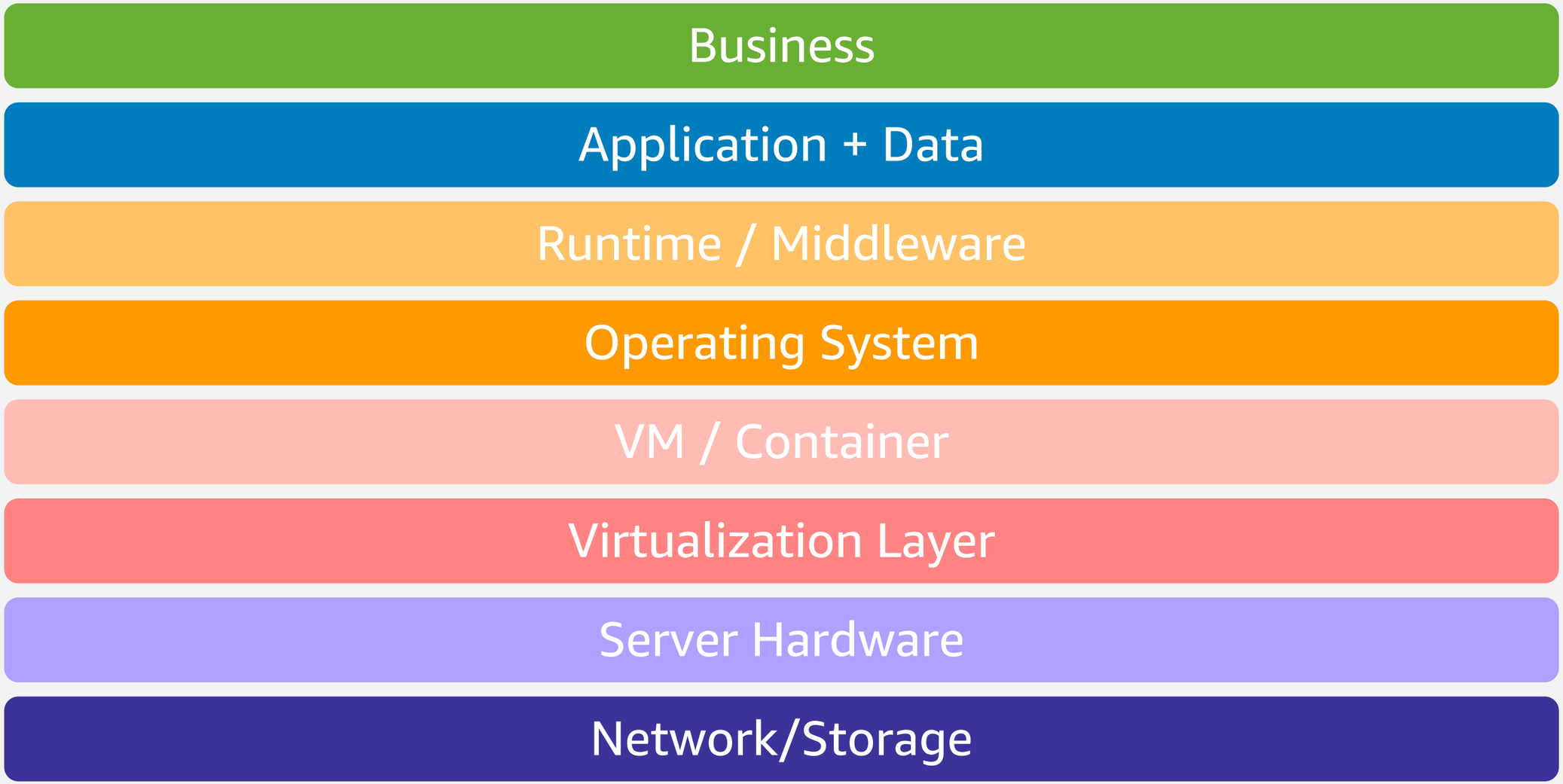
- What is observability
- Standard out-of-the-box metrics
- Structured logging
- Creating custom metrics
- Creating alerts
- Tracing
- Bringing metrics, logging, and tracing together

© 2020, Amazon Web Services, Inc. or its Affiliates.

"Everything fails, all the time."

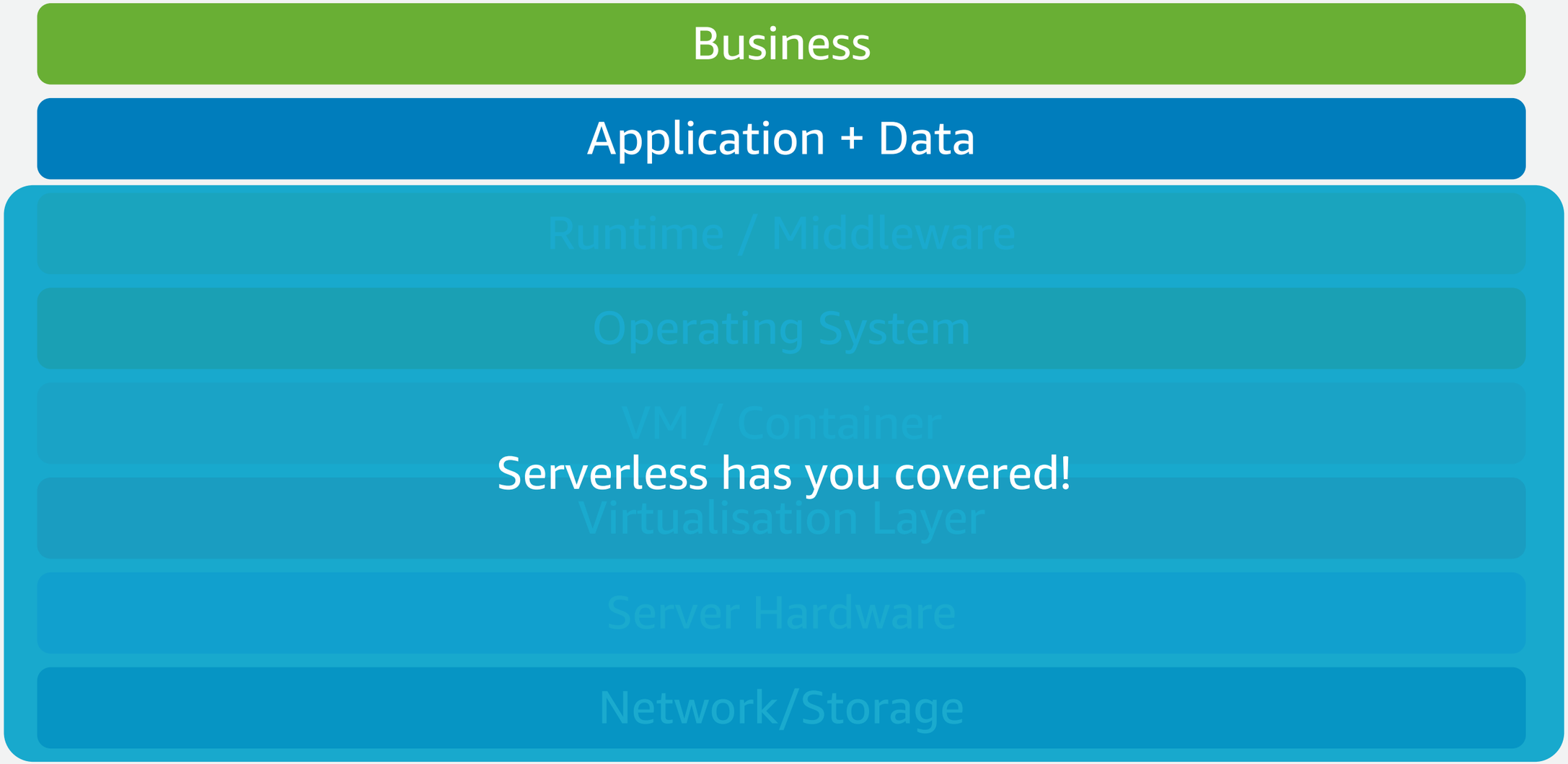
Dr. Werner Vogels, Amazon CTO

Traditional monitoring layers



© 2020, Amazon Web Services, Inc. or its Affiliates.

Traditional monitoring layers



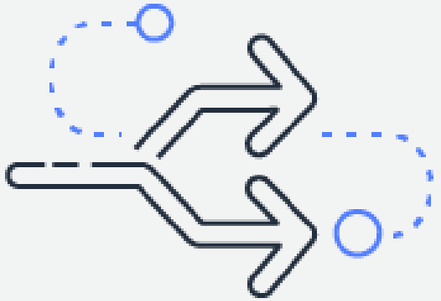
© 2020, Amazon Web Services, Inc. or its Affiliates.

Monitoring needs to evolve

User Experience

Short-lived resources

More devices, services, data



Faster release velocity

© 2020, Amazon Web Services, Inc. or its Affiliates.

Monitoring more than failures

Is it behaving as expected?

What is the usage?

What is the business impact?

© 2020, Amazon Web Services, Inc. or its Affiliates.

What is observability?

© 2020, Amazon Web Services, Inc. or its Affiliates.

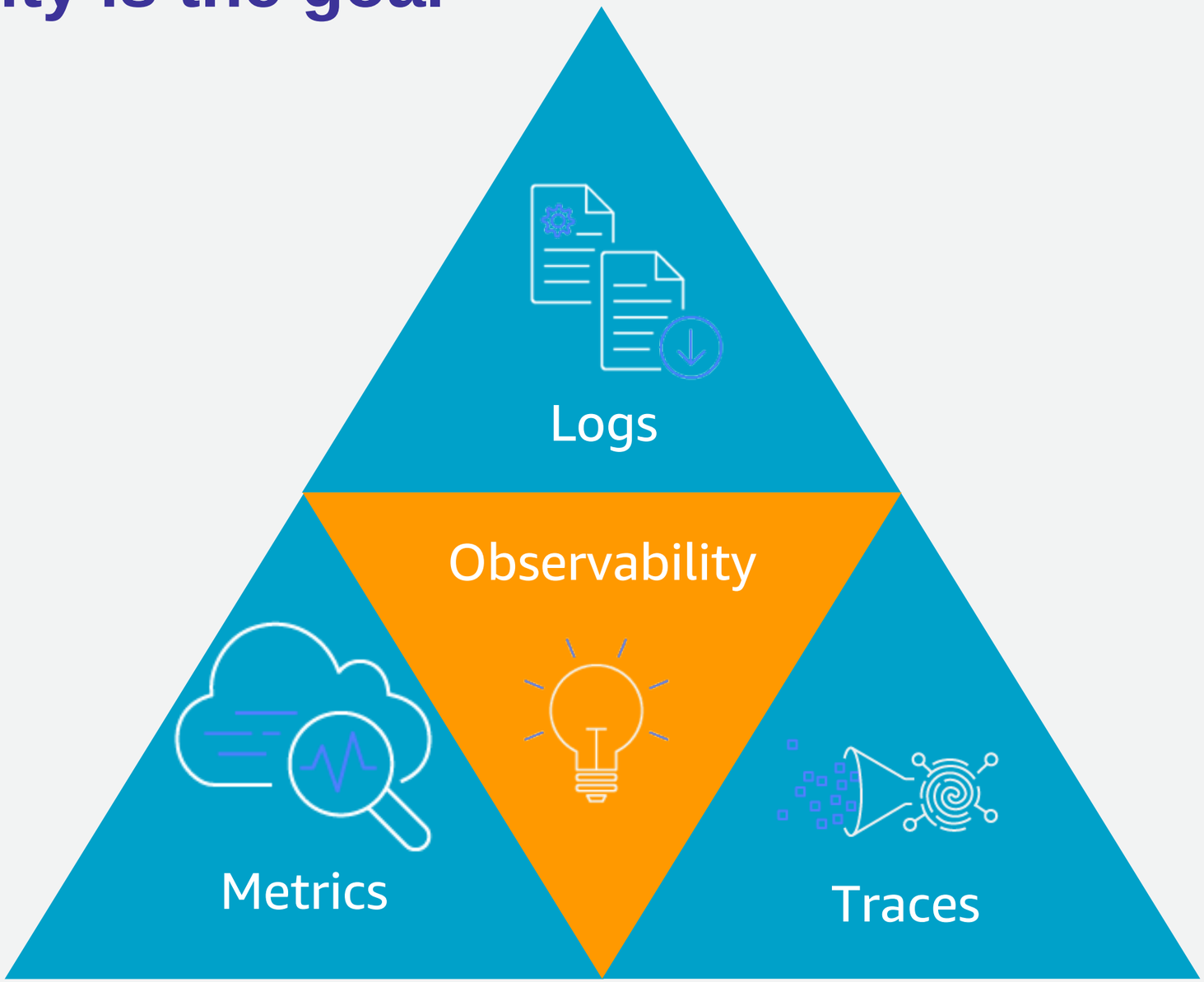
Three pillars of observability

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/

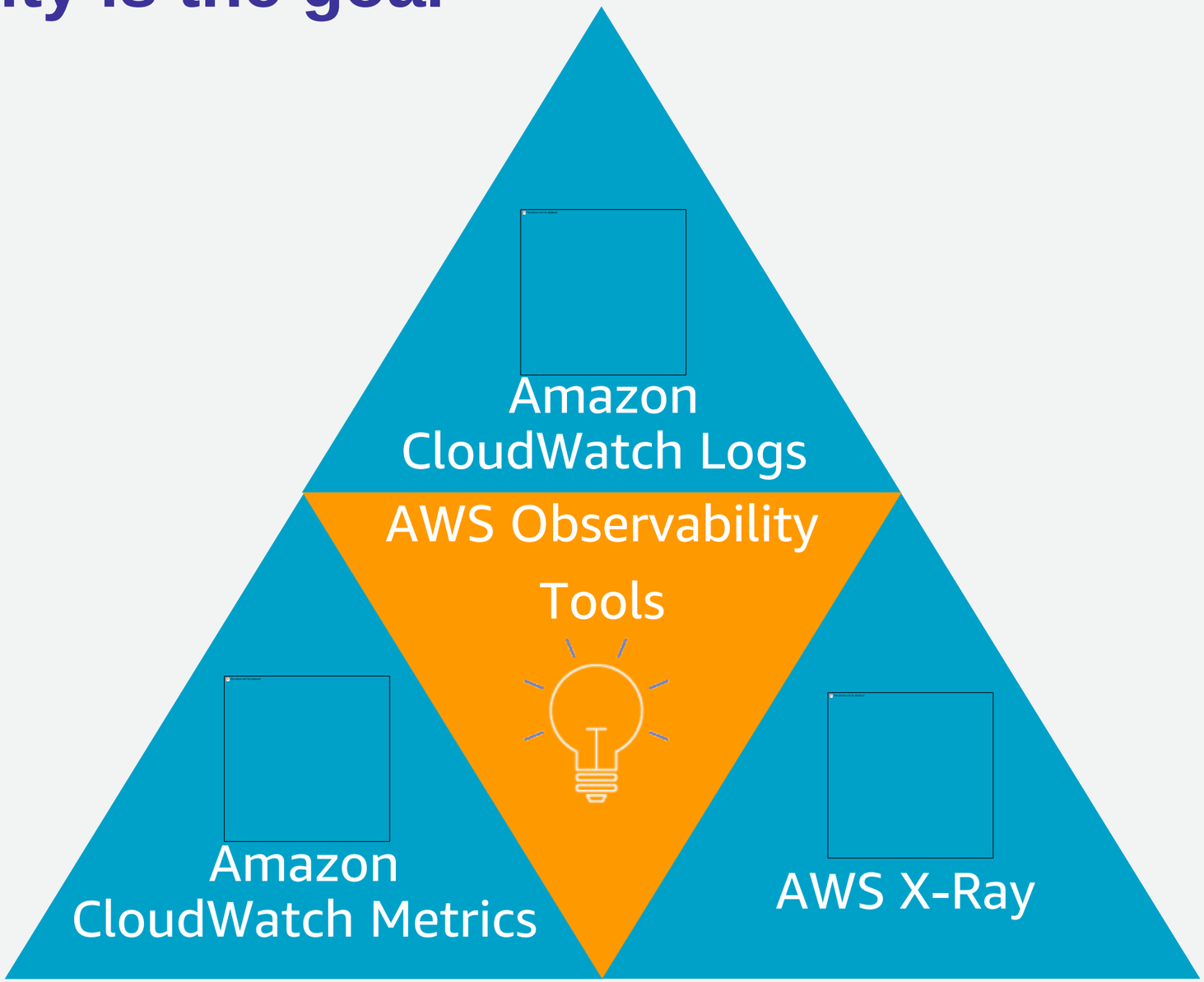
© 2020, Amazon Web Services, Inc. or its Affiliates.

Observability is the goal



© 2020, Amazon Web Services, Inc. or its Affiliates.

Observability is the goal

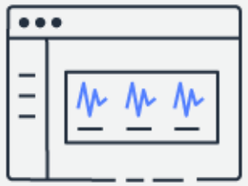


© 2020, Amazon Web Services, Inc. or its Affiliates.

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

© 2020, Amazon Web Services, Inc. or its Affiliates.

Amazon CloudWatch

1 Quadrillion +

(1,000,000,000,000,000 +)

Metric observations each month

Monitors entire infrastructure of AWS and Amazon.com

3.9 Trillion

Events each month

100 PB

Logs ingested each month

© 2020, Amazon Web Services, Inc. or its Affiliates.

AWS Lambda Ready Monitoring Partners

© 2020, Amazon Web Services, Inc. or its Affiliates.

Viewing standard metrics

© 2020, Amazon Web Services, Inc. or its Affiliates.

CloudWatch built-in metrics:

AWS Lambda

Invocation Metrics

Invocation Count, Invocation Errors, DeadLetterErrors, DestinationDeliveryFailures, Throttles, ProvisionedConcurrencyInvocations, ProvisionedConcurrencySpilloverInvocations

Performance Metrics

Duration, IteratorAge

Concurrency Metrics

ConcurrentExecutions, ProvisionedConcurrentExecutions, ProvisionedConcurrencyUtilization, UnreservedConcurrentExecutions

Amazon API Gateway

REST

API Calls Count, Latency, 4XXs, 5XXs, Integration Latency, Cache Hit Count, Cache Miss Count

HTTP

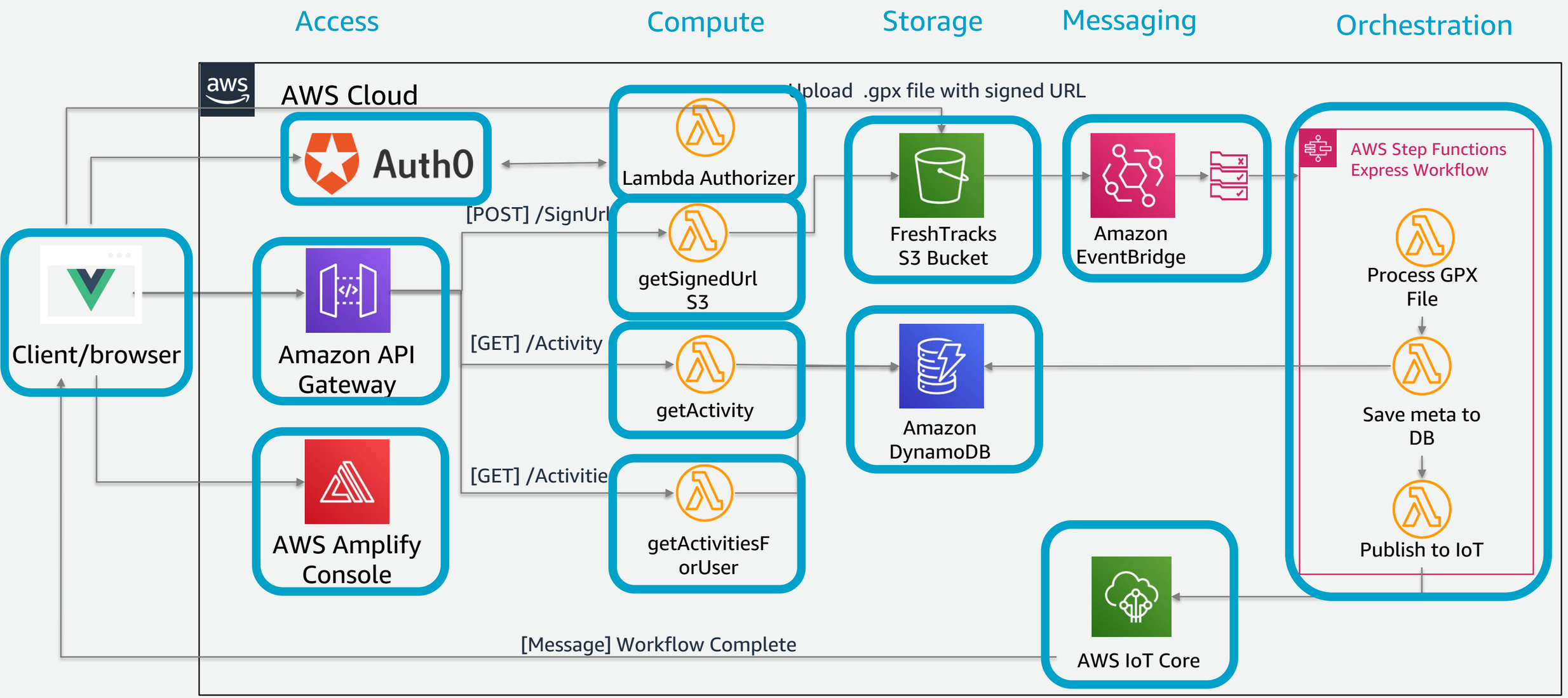
API Calls Count, Latency, 4XXs, 5XXs, Integration Latency, DataProcessed

WebSocket

Connect Count, Message Count, Integration Error, Client Error, Execution Error, Integration Latency

© 2020, Amazon Web Services, Inc. or its Affiliates.

Fresh Tracks Architecture



© 2020, Amazon Web Services, Inc. or its Affiliates.

AWS Management Console

My Fresh Tracks

https://master.d2wl5sw7t2m7g6.amplifyapp.com

70%

Home Activities

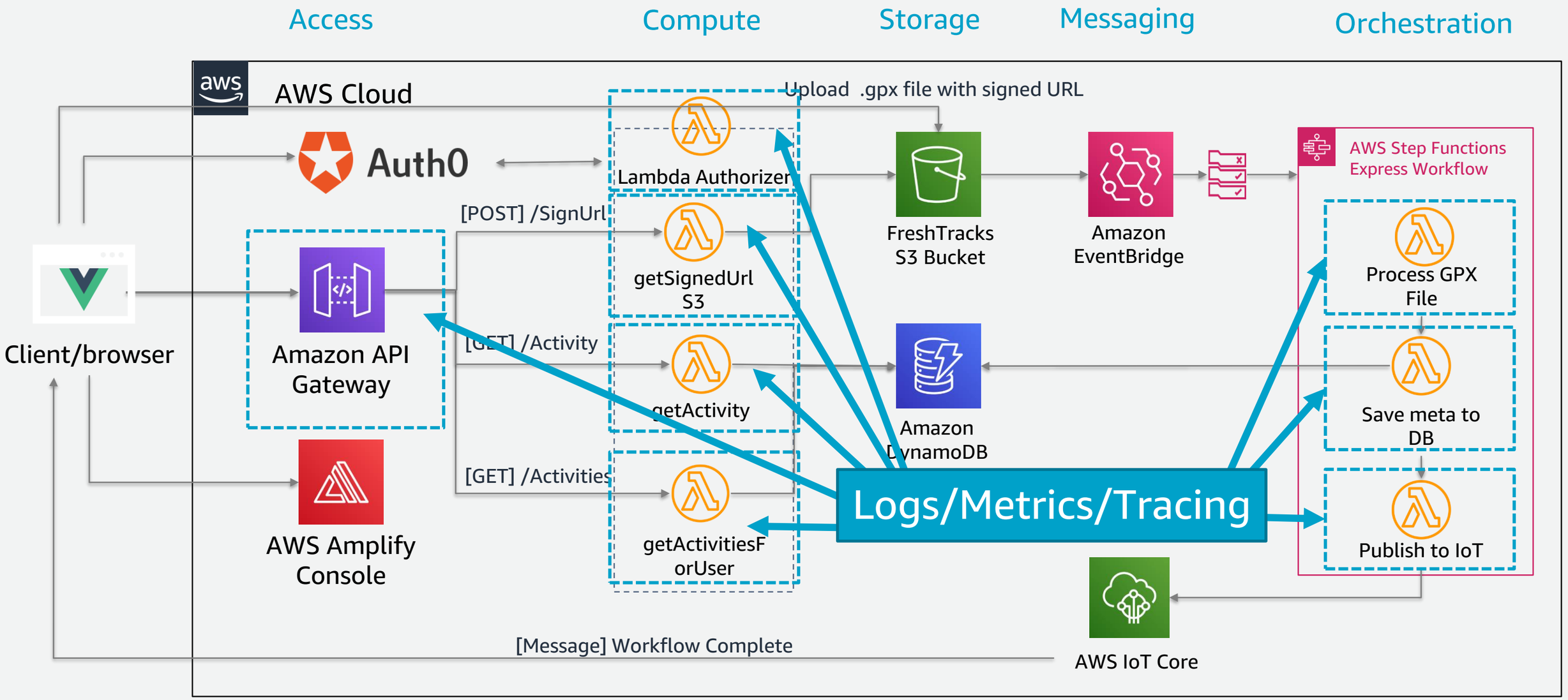
JR

Fresh Tracks

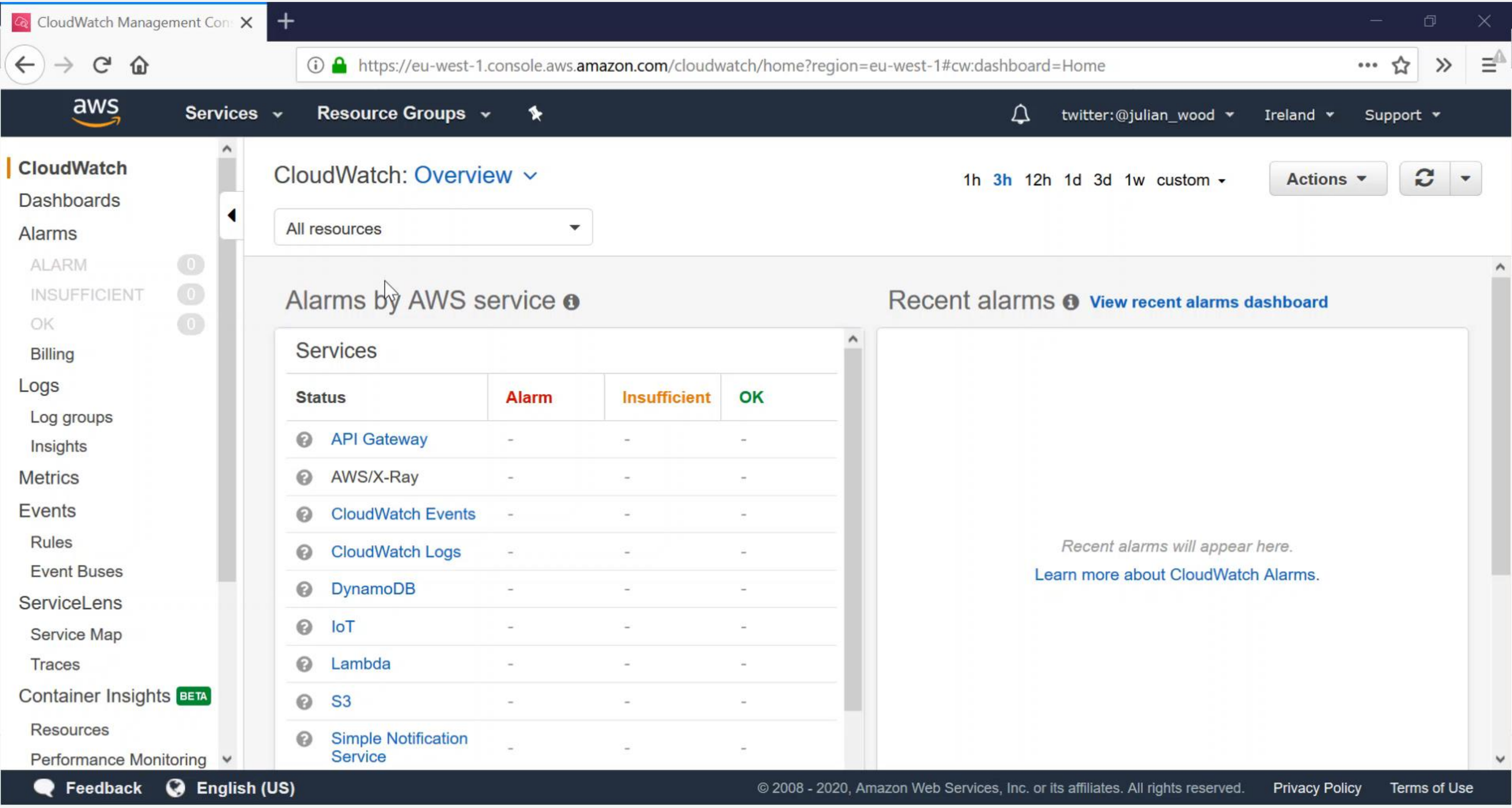
Fresh Tracks is a sample serverless web application that lets you visualize your ski and snowboarding data to keep a record of all your mountain

Support

Fresh Tracks Architecture



© 2020, Amazon Web Services, Inc. or its Affiliates.


CloudWatch Management Con
https://eu-west-1.console.aws.amazon.com/cloudwatch/home?region=eu-west-1#cw:dashboard=Home
aws
Services
Resource Groups
twitter:@julian_wood
Ireland
Support
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 BETA
Resources
Performance Monitoring
CloudWatch: Overview
1h 3h 12h 1d 3d 1w custom
Actions
All resources
Alarms by AWS service
Recent alarms
View recent alarms dashboard
Services
Status
Alarm
Insufficient
OK
API Gateway
AWS/X-Ray
CloudWatch Events
CloudWatch Logs
DynamoDB
IoT
Lambda
S3
Simple Notification Service
Recent alarms will appear here.
Learn more about CloudWatch Alarms.
Feedback
English (US)
© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

Structured Logging

© 2020, Amazon Web Services, Inc. or its Affiliates.

CloudWatch Logs

API Gateway REST API Logging

- 2 Levels of logging, ERROR and INFO
- Optionally log method request/body content
- Set globally in stage, or override per method

Lambda Logging

- Logging directly from your code with your language's equivalent of `console.log()` - basic request information included
- JSON structured logging via `PutMetricData` API or Embedded Metrics Format which includes invocation information

Metric Filters

- Build metrics based on log filters
- Jump to logs that generated metrics

Export logs to Amazon ElastiCache or S3

- Explore with Kibana or Athena/QuickSight

© 2020, Amazon Web Services, Inc. or its Affiliates.

Standard structured logging

```
message = {
  "PriceInCart": 100,
  "QuantityInCart": 2,
  "ProductId": "a23390f3",
  "CategoryId": "bca4cec1",
  "UserId": "31ba3930",
  "CartId": "58dd189f",
  "Environment": "prod",
  "LogLevel": "INFO",
  "Timestamp": "2019-12-11 12:44:40.300473",
  "Message": "Added 2 items 'a23390f3'
  to cart '58dd189f'"
}
```

```
# Python
print(json.dumps(message))

// Javascript
console.log(JSON.stringify(message));

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

// Java
class JSONLoggerInitializerFactory...
```

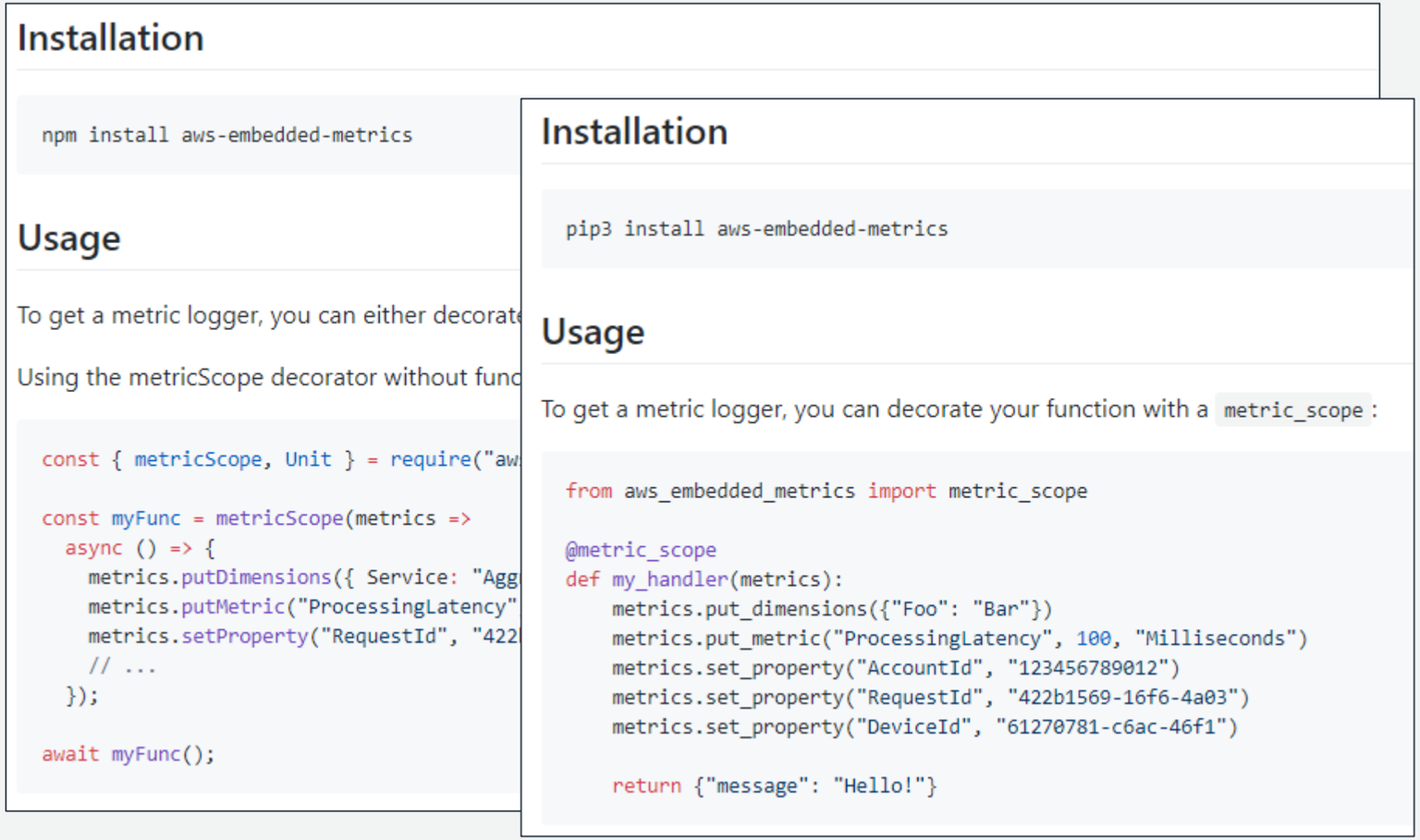
© 2020, Amazon Web Services, Inc. or its Affiliates.

CloudWatch Embedded Metrics Format

Embed custom metrics alongside detailed log event data.

Automatically generate metrics from structured CloudWatch Logs.

Open-source client libraries available for Node.js and Python



Installation

```
npm install aws-embedded-metrics
```

Usage

To get a metric logger, you can either decorat

Using the metricScope decorator without func

```
const { metricScope, Unit } = require("aw

const myFunc = metricScope(metrics =>
  async () => {
    metrics.putDimensions({ Service: "Agg
    metrics.putMetric("ProcessingLatency"
    metrics.setProperty("RequestId", "422
    // ...
  });

await myFunc();
```

Installation

```
pip3 install aws-embedded-metrics
```

Usage

To get a metric logger, you can decorate your function with a `metric_scope`:

```
from aws_embedded_metrics import metric_scope

@metric_scope
def my_handler(metrics):
    metrics.put_dimensions({"Foo": "Bar"})
    metrics.put_metric("ProcessingLatency", 100, "Milliseconds")
    metrics.set_property("AccountId", "123456789012")
    metrics.set_property("RequestId", "422b1569-16f6-4a03")
    metrics.set_property("DeviceId", "61270781-c6ac-46f1")

    return {"message": "Hello!"}
```

© 2020, Amazon Web Services, Inc. or its Affiliates.

CloudWatch Embedded Metrics Format

```
{
  "_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"
}
```

© 2020, Amazon Web Services, Inc. or its Affiliates.

Creating custom metrics

Application, business, and operations metrics

© 2020, Amazon Web Services, Inc. or its Affiliates.

Built-in metrics often not enough

What about business / customer metrics?

Measure application performance against business goals

Revenue, sign-ups, tracks uploaded, perceived latency, page views, etc.

How operationally stable is the application?

Continuous integration/deployment feedback time, mean time between failure/recovery, number of on-call pages and time to resolution, etc.

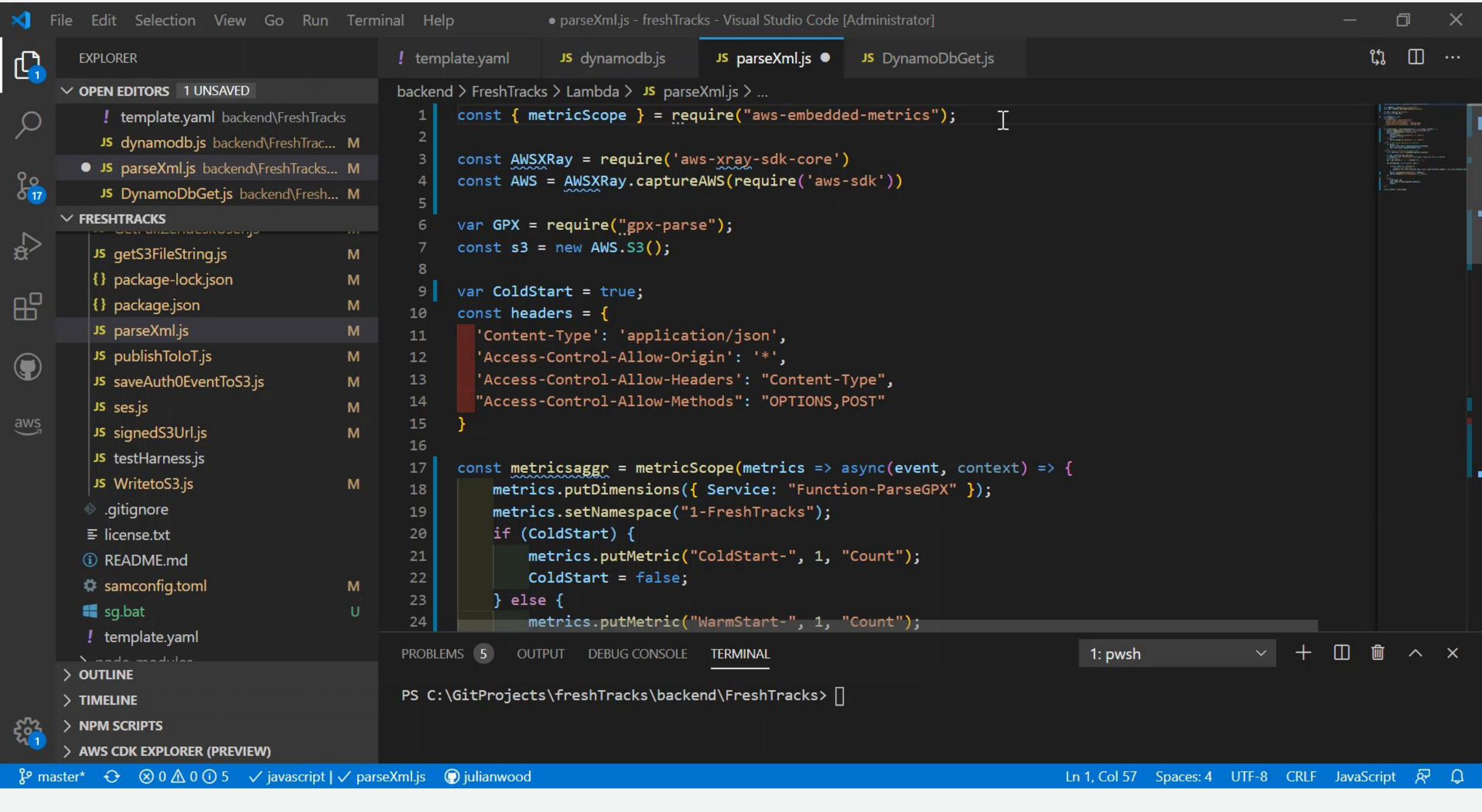
What about caught errors, warnings?

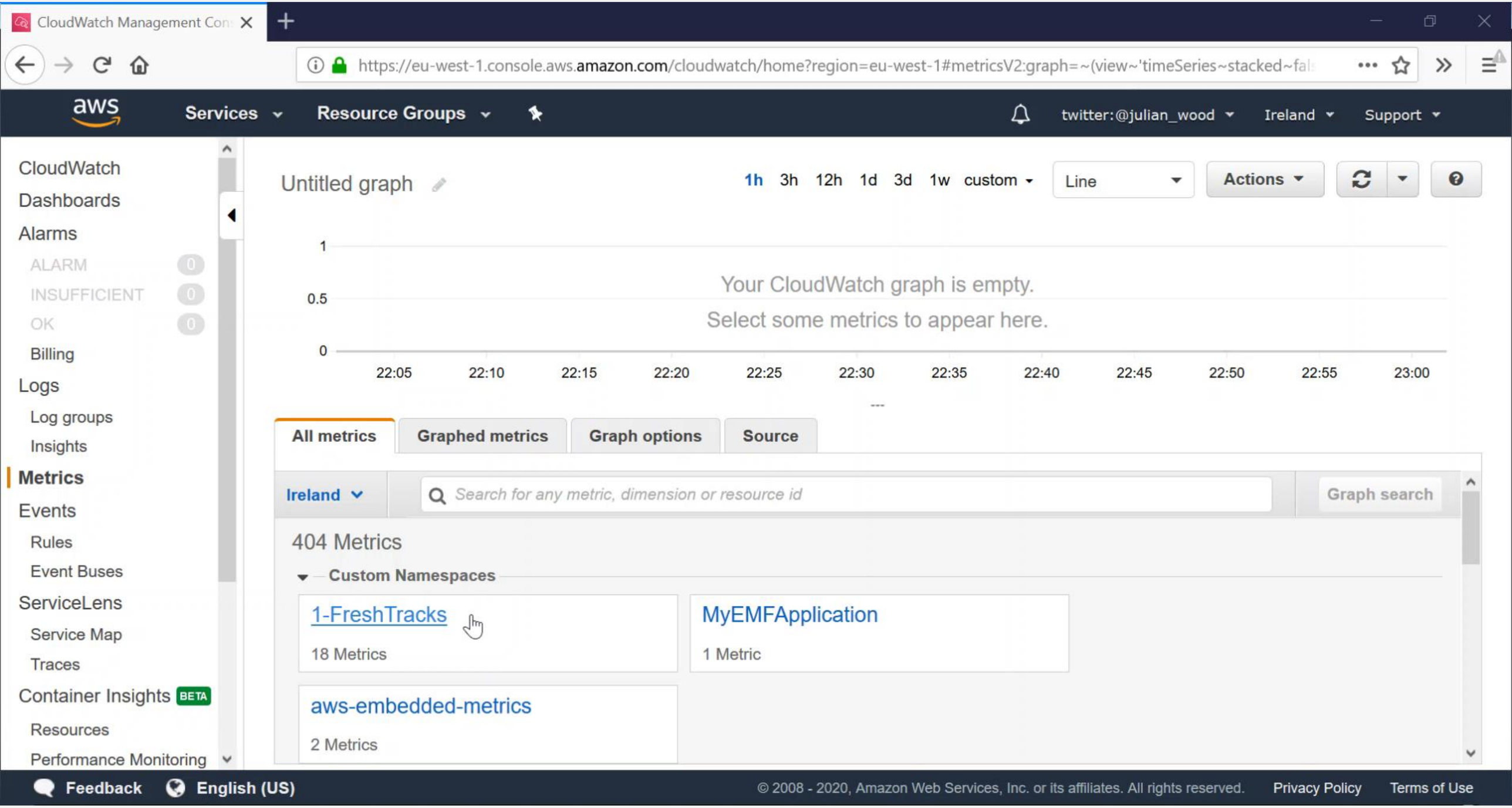
Caught exceptions are not counted as Errors on AWS Lambda.

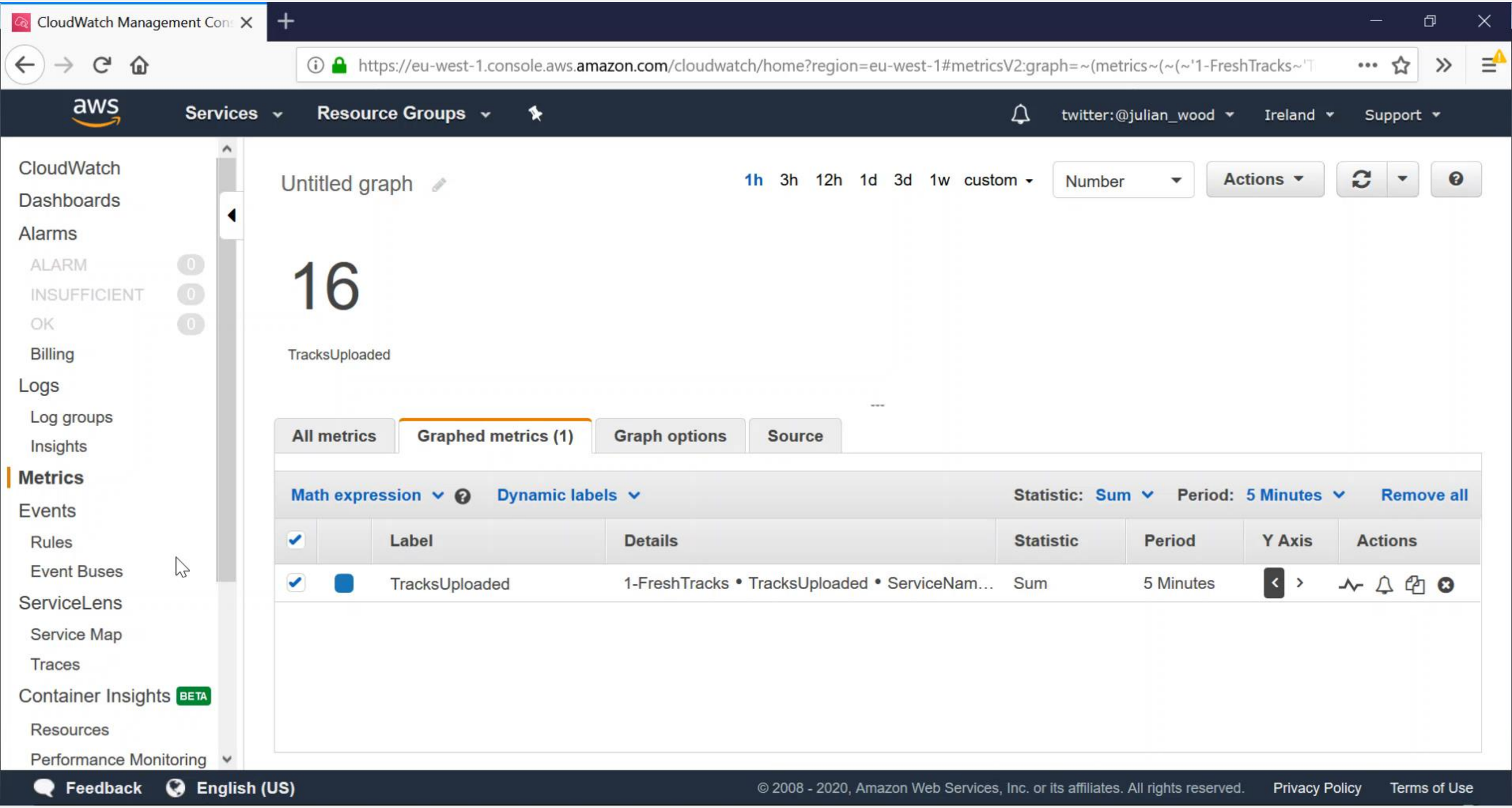
What if I want to use other dimensions?

User ID, category, item, tags, environment, etc.

© 2020, Amazon Web Services, Inc. or its Affiliates.


File Edit Selection View Go Run Terminal Help
• parseXml.js - freshTracks - Visual Studio Code [Administrator]
EXPLORER
OPEN EDITORS 1 UNSAVED
template.yaml backend\FreshTracks
dynamodb.js backend\FreshTrac... M
parseXml.js backend\FreshTracks... M
DynamoDbGet.js backend\Fresh... M
FRESHTRACKS
getS3FileString.js M
package-lock.json M
package.json M
parseXml.js M
publishToIoT.js M
saveAuth0EventToS3.js M
ses.js M
signedS3Url.js M
testHarness.js
WritetoS3.js M
.gitignore
license.txt
README.md
samconfig.toml M
sg.bat U
template.yaml
OUTLINE
TIMELINE
NPM SCRIPTS
AWS CDK EXPLORER (PREVIEW)
template.yaml
dynamodb.js
parseXml.js
DynamoDbGet.js
backend > FreshTracks > Lambda > parseXml.js > ...
1 const { metricScope } = require("aws-embedded-metrics");
2
3 const AWSXRay = require('aws-xray-sdk-core')
4 const AWS = AWSXRay.captureAWS(require('aws-sdk'))
5
6 var GPX = require("gpx-parse");
7 const s3 = new AWS.S3();
8
9 var ColdStart = true;
10 const headers = {
11 'Content-Type': 'application/json',
12 'Access-Control-Allow-Origin': '*',
13 'Access-Control-Allow-Headers': "Content-Type",
14 "Access-Control-Allow-Methods": "OPTIONS,POST"
15 }
16
17 const metricsaggr = metricScope(metrics => async(event, context) => {
18 metrics.putDimensions({ Service: "Function-ParseGPX" });
19 metrics.setNamespace("1-FreshTracks");
20 if (ColdStart) {
21 metrics.putMetric("ColdStart-", 1, "Count");
22 ColdStart = false;
23 } else {
24 metrics.putMetric("WarmStart-", 1, "Count");
PROBLEMS 5 OUTPUT DEBUG CONSOLE TERMINAL
1: pwsh
PS C:\GitProjects\freshTracks\backend\FreshTracks>
master* 0 0 5 javascript | parseXml.js julianwood
Ln 1, Col 57 Spaces: 4 UTF-8 CRLF JavaScript

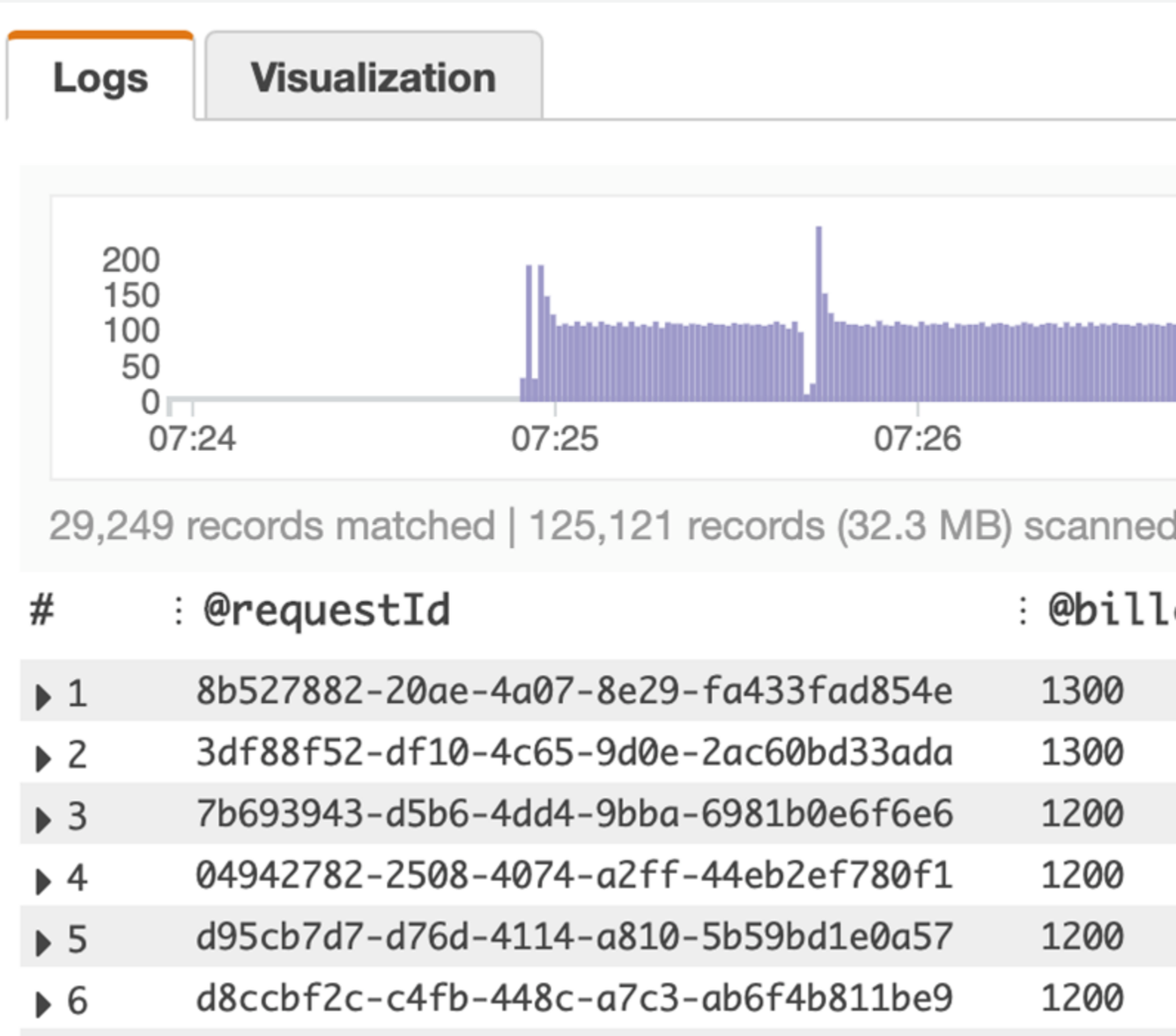

CloudWatch Management Con
https://eu-west-1.console.aws.amazon.com/cloudwatch/home?region=eu-west-1#metricsV2:graph=~(view~'timeSeries~stacked~fal
aws
Services
Resource Groups
twitter:@julian_wood
Ireland
Support
CloudWatch
Dashboards
Alarms
ALARM 0
INSUFFICIENT 0
OK 0
Billing
Logs
Log groups
Insights
Metrics
Events
Rules
Event Buses
ServiceLens
Service Map
Traces
Container Insights BETA
Resources
Performance Monitoring
Untitled graph
1h 3h 12h 1d 3d 1w custom
Line
Actions
1
0.5
0
Your CloudWatch graph is empty.
Select some metrics to appear here.
22:05 22:10 22:15 22:20 22:25 22:30 22:35 22:40 22:45 22:50 22:55 23:00
All metrics
Graphed metrics
Graph options
Source
Ireland
Search for any metric, dimension or resource id
Graph search
404 Metrics
Custom Namespaces
1-FreshTracks
18 Metrics
MyEMFApplication
1 Metric
aws-embedded-metrics
2 Metrics
Feedback
English (US)
© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use


CloudWatch Management Con
https://eu-west-1.console.aws.amazon.com/cloudwatch/home?region=eu-west-1#metricsV2:graph=~(metrics~(~('1-FreshTracks~'T
aws
Services
Resource Groups
twitter:@julian_wood
Ireland
Support
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 BETA
Resources
Performance Monitoring
Untitled graph
1h 3h 12h 1d 3d 1w custom
Number
Actions
16
TracksUploaded
All metrics
Graphed metrics (1)
Graph options
Source
Math expression
Dynamic labels
Statistic: Sum
Period: 5 Minutes
Remove all
Label
Details
Statistic
Period
Y Axis
Actions
TracksUploaded
1-FreshTracks • TracksUploaded • ServiceNam...
Sum
5 Minutes
Feedback
English (US)
© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

Query with CloudWatch Insights

Top 100 most expensive executions

```
filter @type = "REPORT"
| fields @requestId, @billedDuration
| sort by @billedDuration desc
| limit 100
```

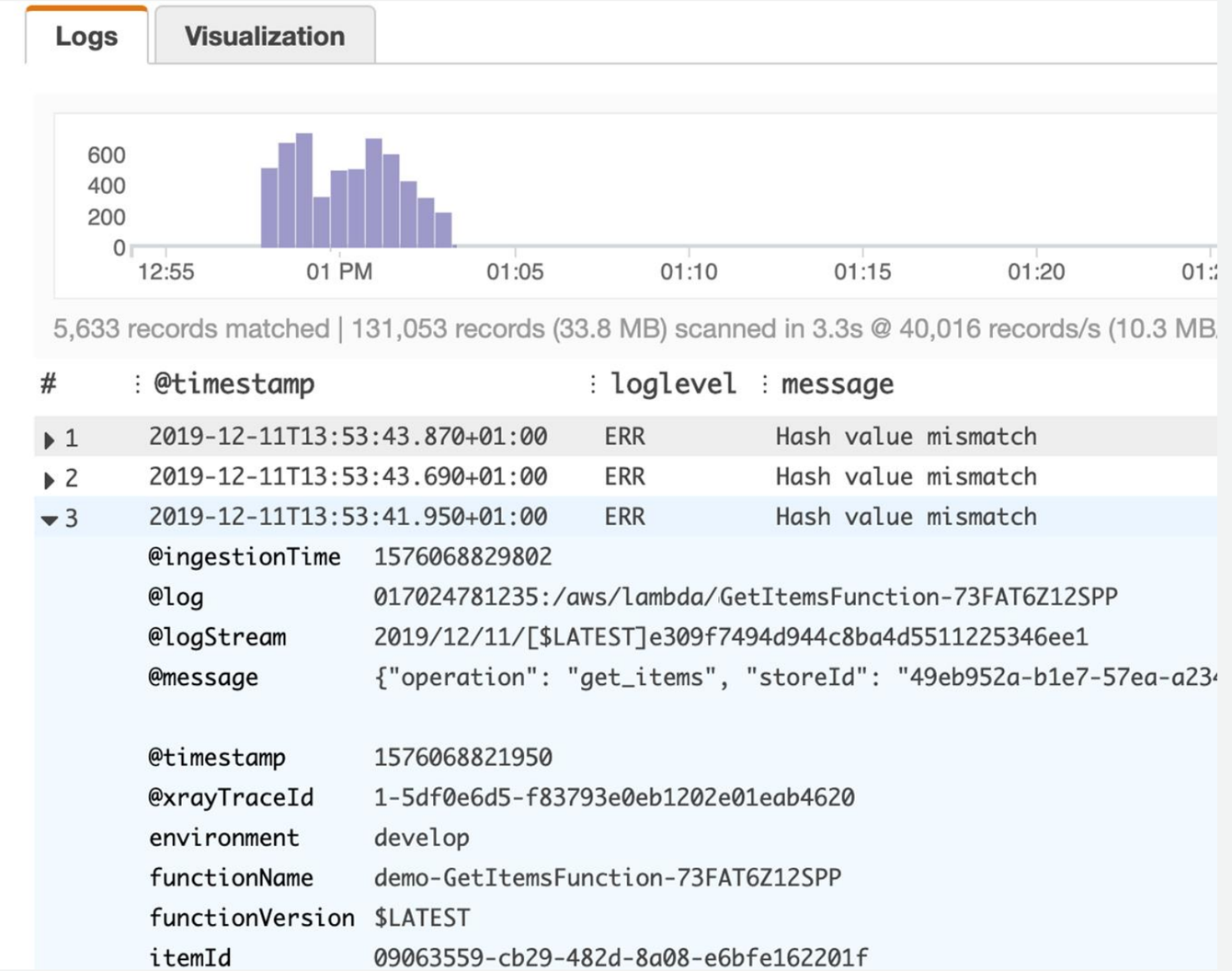


© 2020, Amazon Web Services, Inc. or its Affiliates.

Query with CloudWatch Insights

Get the last 100 error messages

```
fields Timestamp, LogLevel, Message
| filter LogLevel == "ERR"
| sort @timestamp desc
| limit 100
```



© 2020, Amazon Web Services, Inc. or its Affiliates.

Lambda Layers

Lets functions easily share code: Upload layer once, reference within any function

Layer can be anything: dependencies, training data, configuration files, etc

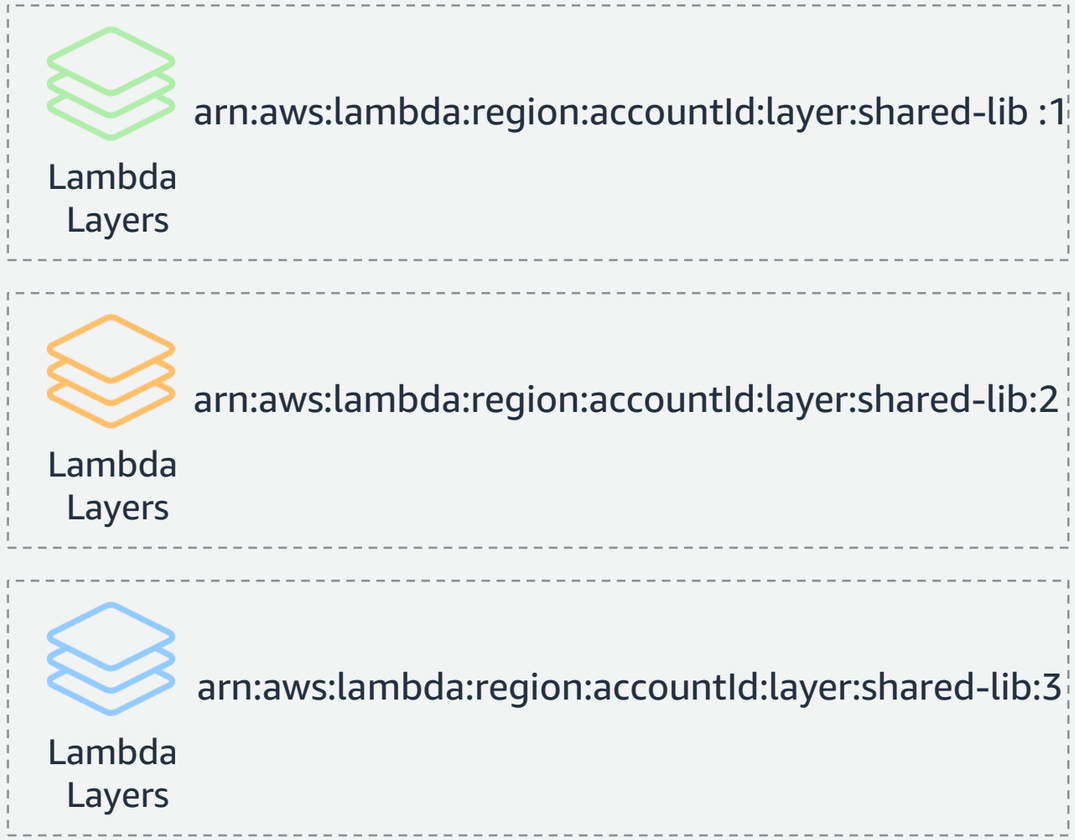
Promote separation of responsibilities, lets developers iterate faster on writing business logic

Built-in support for secure sharing by ecosystem

© 2020, Amazon Web Services, Inc. or its Affiliates.

Using Lambda Layers

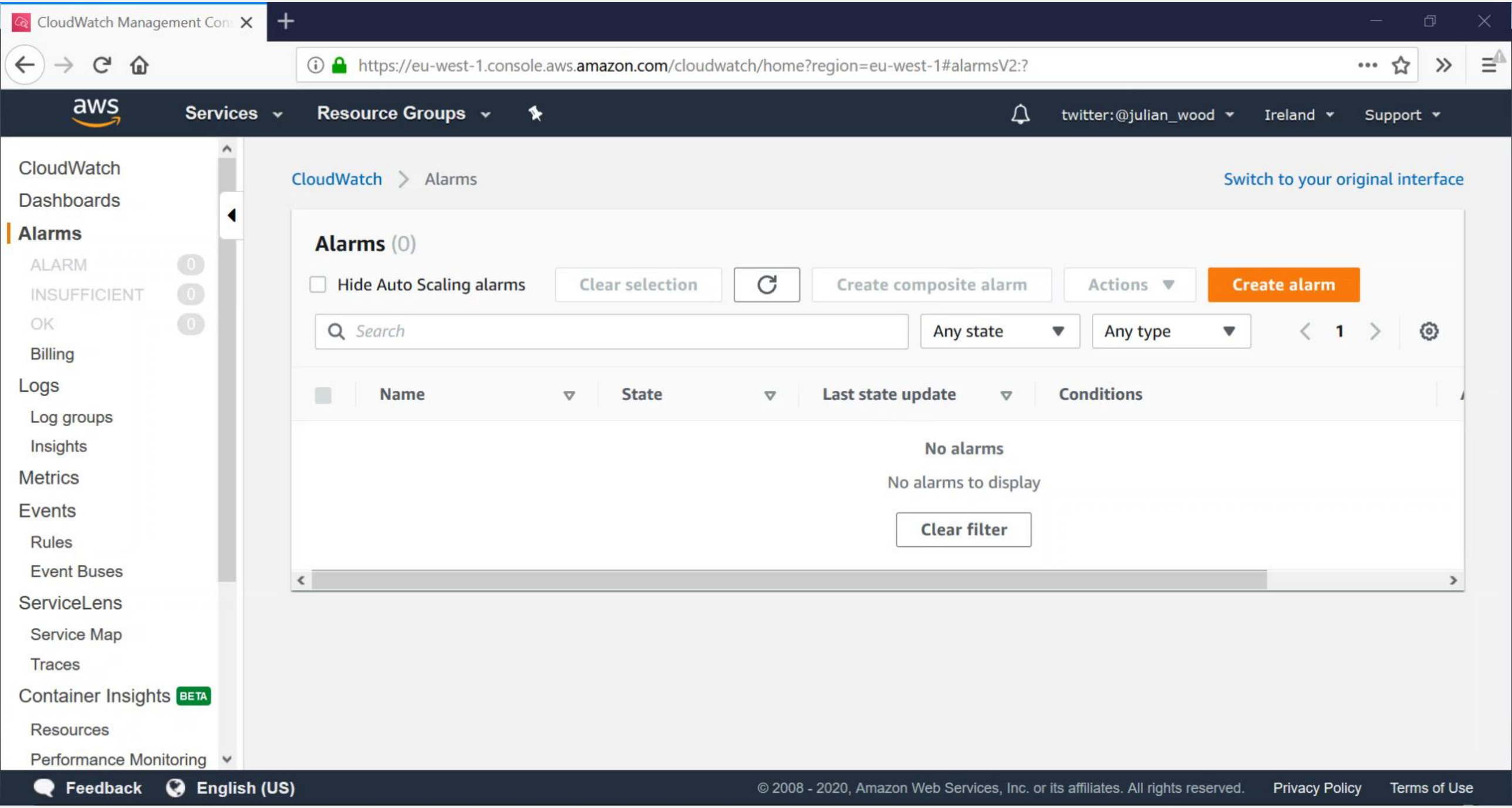
- Put common components in a ZIP file and upload it as a Lambda Layer
- Layers are immutable and can be versioned to manage updates
- When a version is deleted or permissions to use it are revoked, functions that used it previously will continue to work, but you won't be able to create new ones
- You can reference up to five layers, one of which can optionally be a custom runtime



© 2020, Amazon Web Services, Inc. or its Affiliates.

Creating alerts

© 2020, Amazon Web Services, Inc. or its Affiliates.

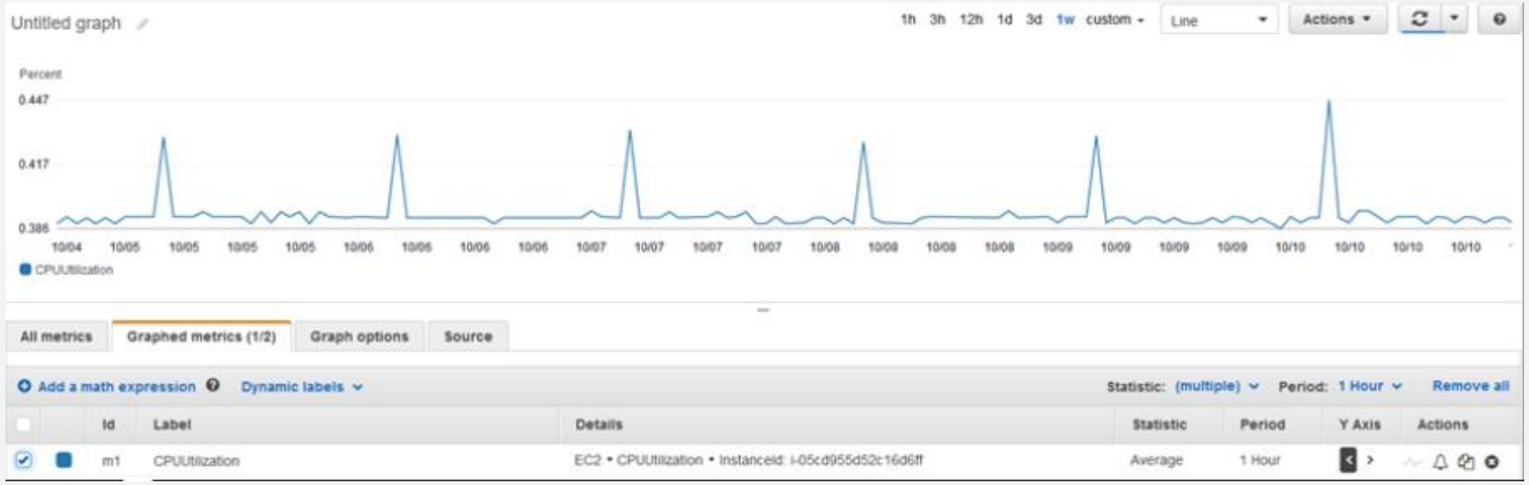

CloudWatch Management Con
https://eu-west-1.console.aws.amazon.com/cloudwatch/home?region=eu-west-1#alarmsV2:?
aws
Services
Resource Groups
twitter:@julian_wood
Ireland
Support
CloudWatch
Dashboards
Alarms
ALARM 0
INSUFFICIENT 0
OK 0
Billing
Logs
Log groups
Insights
Metrics
Events
Rules
Event Buses
ServiceLens
Service Map
Traces
Container Insights BETA
Resources
Performance Monitoring
CloudWatch > Alarms
Switch to your original interface
Alarms (0)
Hide Auto Scaling alarms
Clear selection
Create composite alarm
Actions
Create alarm
Search
Any state
Any type
1
Name
State
Last state update
Conditions
No alarms
No alarms to display
Clear filter
Feedback
English (US)
© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

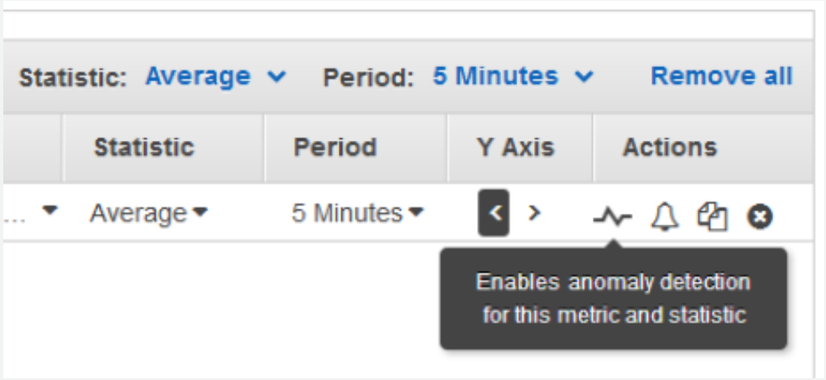
Configuring alerts via CloudFormation

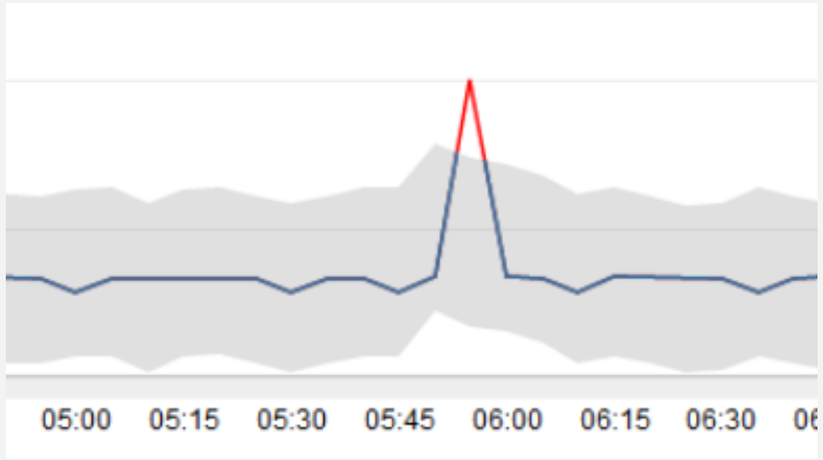
```
P99ApiGatewayLatencyAlarm:
  Type: AWS::CloudWatch::Alarm
  Properties:
    AlarmDescription: P99 Latency for API Gateway
    ComparisonOperator: GreaterThanOrEqualToThreshold
    Dimensions:
      - Name: ApiName
        Value: !Ref AWS::StackName
    EvaluationPeriods: 1
    ExtendedStatistic: "p99"
    MetricName: "Latency"
    Namespace: "AWS/ApiGateway"
    Period: 60
    Threshold: 1000
    TreatMissingData: notBreaching
    Unit: Milliseconds
```

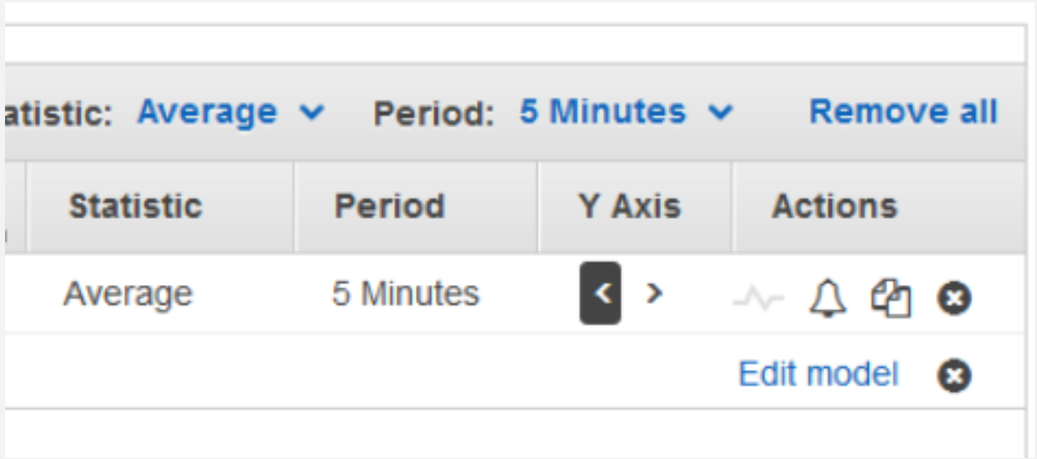
© 2020, Amazon Web Services, Inc. or its Affiliates.

Using CloudWatch Anomaly Detection Alarms









© 2020, Amazon Web Services, Inc. or its Affiliates.

Tracing

© 2020, Amazon Web Services, Inc. or its Affiliates.

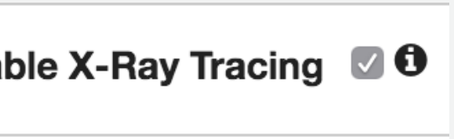
AWS X-Ray

End-to-end view of requests flowing through an application

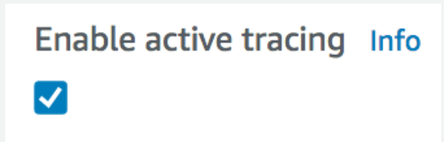
- Lambda: instruments incoming requests for all supported languages and can capture calls made in code

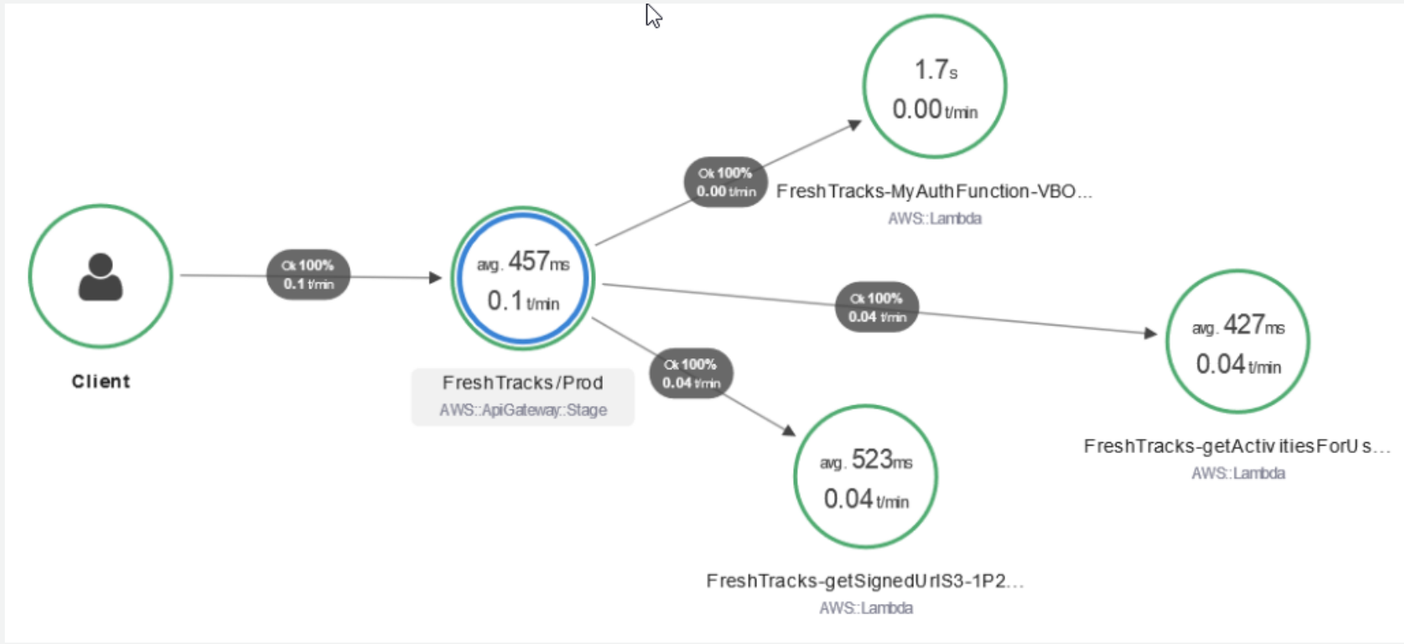
```
const AWSXRay = require('aws-xray-sdk-core');
const AWS = AWSXRay.captureAWS(require('aws-sdk'));

const documentClient = new AWS.DynamoDB.DocumentClient();
```



- API Gateway: inserts a tracing header into HTTP calls as well as reports data back to X-Ray itself

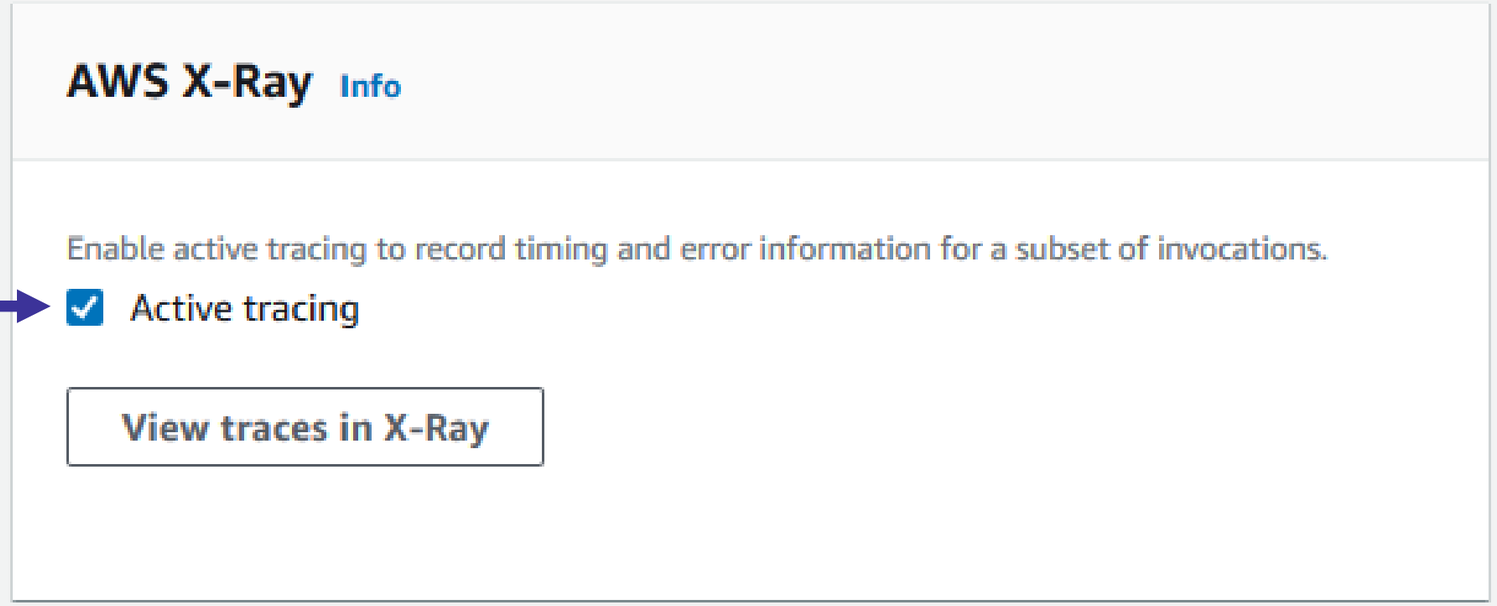




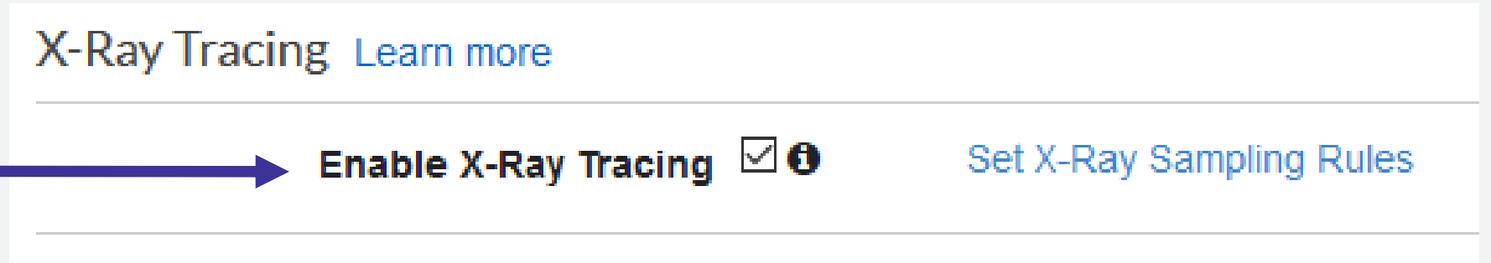
© 2020, Amazon Web Services, Inc. or its Affiliates.

Enabling X-Ray tracing: AWS Management Console

AWS Lambda Console
(per function)



Amazon API Gateway Console
(per stage)



Also need to add IAM permissions for X-Ray

© 2020, Amazon Web Services, Inc. or its Affiliates.

Enabling X-Ray tracing: AWS SAM

90% of the work in two lines!

Globals section

All Lambda functions

All API Gateway REST APIs

```
Globals:
  Function:
    Tracing: Active
  Api:
    TracingEnabled: True
```

© 2020, Amazon Web Services, Inc. or its Affiliates.

Enabling X-Ray tracing: AWS SAM

IAM policy statements

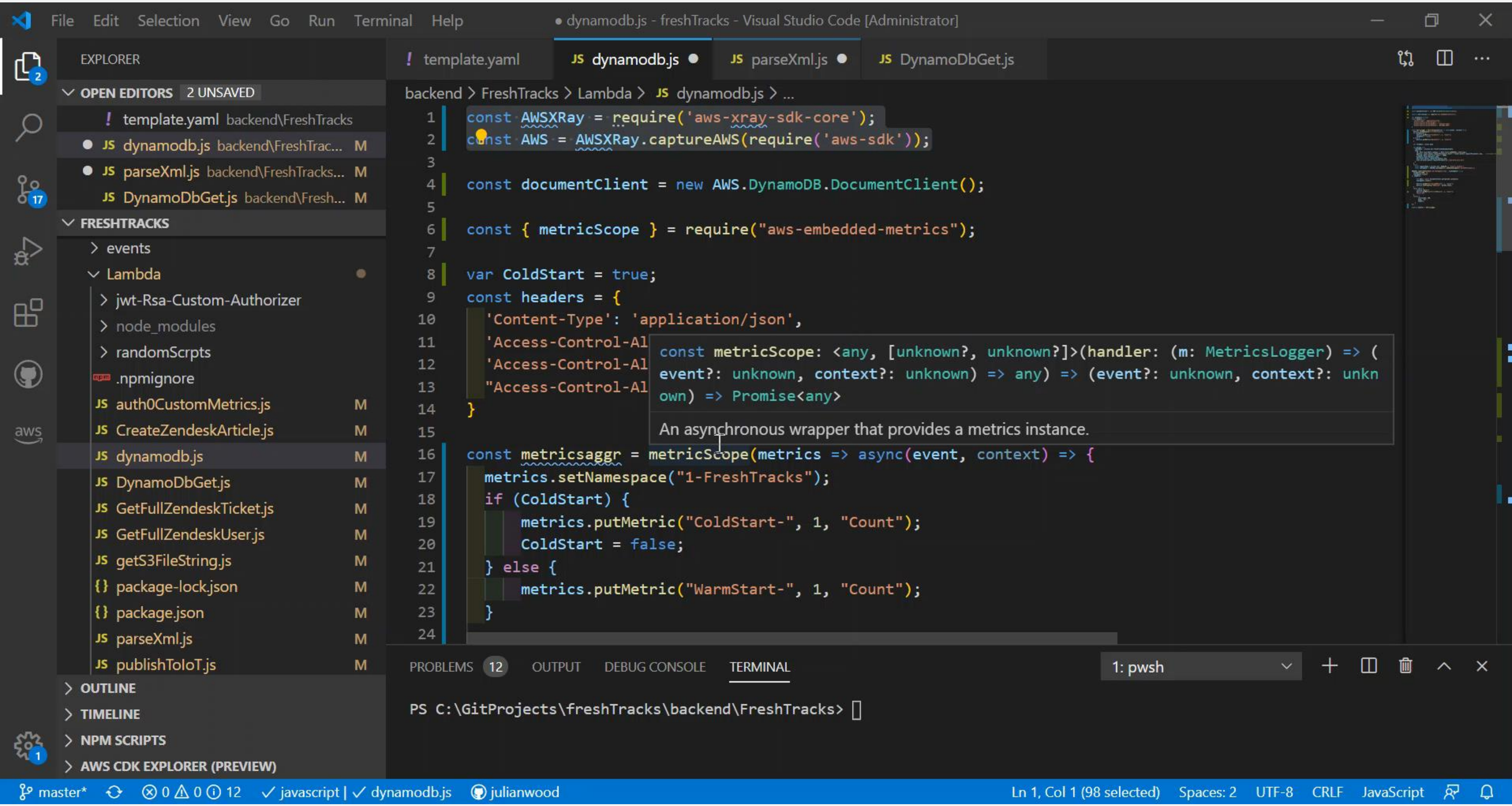
Lambda function

Execution Policy Role

```
MyLambdaFunction:
  Type: "AWS::Lambda::Function"
DependsOn:
  - roleLambdaExecutionPolicy

roleLambdaExecutionPolicy:
  Type: "AWS::IAM::Role"
  Properties:
    Policies:
        Statement:
        - Action:
          - "xray:PutTraceSegments"
          - "xray:PutTelemetryRecords"
          Effect: Allow
          Resource: "*"
```

© 2020, Amazon Web Services, Inc. or its Affiliates.


File Edit Selection View Go Run Terminal Help
• dynamodb.js - freshTracks - Visual Studio Code [Administrator]
EXPLORER
OPEN EDITORS 2 UNSAVED
template.yaml backend\FreshTracks
dynamodb.js backend\FreshTrac... M
parseXml.js backend\FreshTracks... M
DynamoDbGet.js backend\Fresh... M
FRESHTRACKS
events
Lambda
jwt-Rsa-Custom-Authorizer
node_modules
randomScrpts
.npmignore
auth0CustomMetrics.js M
CreateZendeskArticle.js M
dynamodb.js M
DynamoDbGet.js M
GetFullZendeskTicket.js M
GetFullZendeskUser.js M
getS3FileString.js M
package-lock.json M
package.json M
parseXml.js M
publishToIoT.js M
OUTLINE
TIMELINE
NPM SCRIPTS
AWS CDK EXPLORER (PREVIEW)
template.yaml
dynamodb.js
parseXml.js
DynamoDbGet.js
backend > FreshTracks > Lambda > dynamodb.js > ...
const AWSXRay = require('aws-xray-sdk-core');
const AWS = AWSXRay.captureAWS(require('aws-sdk'));
const documentClient = new AWS.DynamoDB.DocumentClient();
const { metricScope } = require("aws-embedded-metrics");
var ColdStart = true;
const headers = {
'Content-Type': 'application/json',
'Access-Control-Al
'Access-Control-Al
"Access-Control-Al
}
const metricsaggr = metricScope(metrics => async(event, context) => {
metrics.setNamespace("1-FreshTracks");
if (ColdStart) {
metrics.putMetric("ColdStart-", 1, "Count");
ColdStart = false;
} else {
metrics.putMetric("WarmStart-", 1, "Count");
}
const metricScope: <any, [unknown?, unknown?]>(handler: (m: MetricsLogger) => (event?: unknown, context?: unknown) => any) => (event?: unknown, context?: unknown) => Promise<any>
An asynchronous wrapper that provides a metrics instance.
PROBLEMS 12
OUTPUT
DEBUG CONSOLE
TERMINAL
1: pwsh
PS C:\GitProjects\freshTracks\backend\FreshTracks>
master*
0 0 12
javascript | dynamodb.js
julianwood
Ln 1, Col 1 (98 selected)
Spaces: 2
UTF-8
CRLF
JavaScript

AWS X-Ray
Getting started
Service map
Traces
Analytics
Configuration
Sampling
Encryption

Default | Enter service name, annotation, trace ID. Or click the Help icon for additional details. | Last 5 minutes

Service map

Updated on 2020-05-08 01:57:13 (UTC +01:00)

Enter a service name to find and select the node on map

Map legend

Services Icons

None | Health | Traffic

Resize nodes by traffic

© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

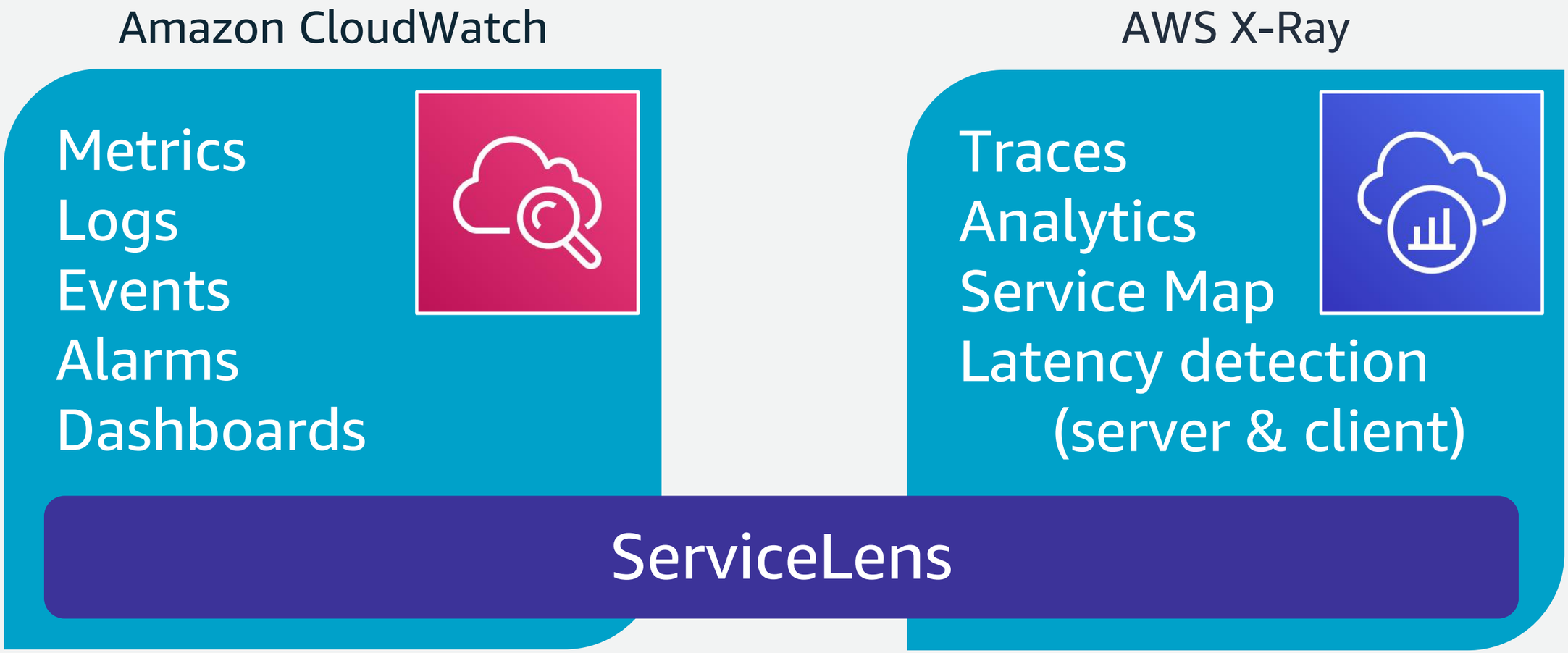
Bringing it all together

© 2020, Amazon Web Services, Inc. or its Affiliates.

CloudWatch ServiceLens

Unified access to metrics, logs, traces and canaries.

Enabling performance monitoring from end-user interaction to infrastructure layer insights



© 2020, Amazon Web Services, Inc. or its Affiliates.

CloudWatch Management Con

https://eu-west-1.console.aws.amazon.com/cloudwatch/home?region=eu-west-1#servicelens:map

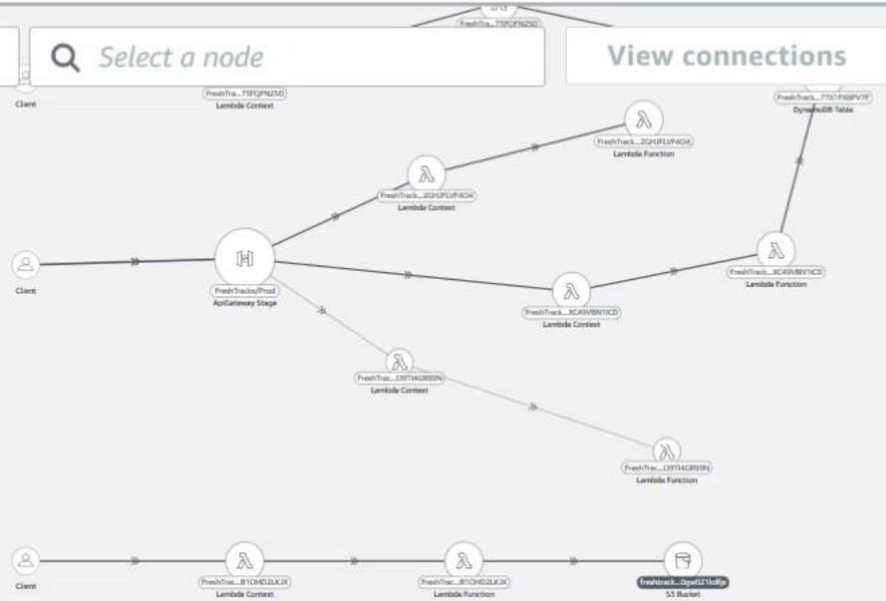
aws Services Resource Groups twitter:@julian_wood Ireland Support

- CloudWatch
- Dashboards
- Alarms
  - ALARM 0
  - INSUFFICIENT 1
  - OK 0
- Billing
- Logs
  - Log groups
  - Insights
- Metrics
- Events
  - Rules
  - Event Buses
- ServiceLens
  - Service Map
  - Traces
- Container Insights BETA
  - Resources
  - Performance Monitoring

CloudWatch > Service Map

5m 15m 30m 1h 3h 6h custom | Full screen | Map view | List view

Filter by X-Ray group | Select a node | View connections | Requests mode



Map legend

- Fault (5xx)
- Error (4xx)
- Throttle
- Current alarms
- AWS service: Service name
- AWS resource: Resource name
- Unknown node: Node name
- Size is requests

No node selected

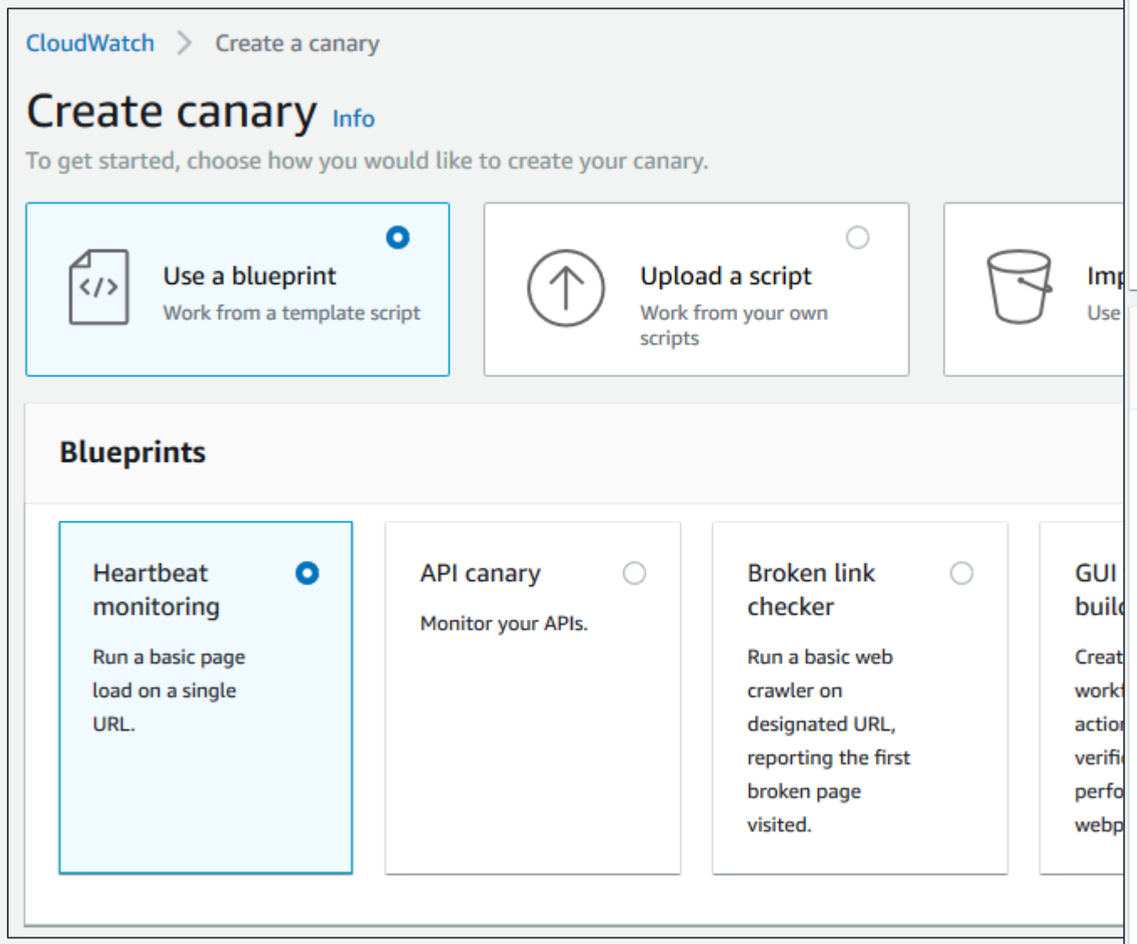
View logs | View traces | View dashboard

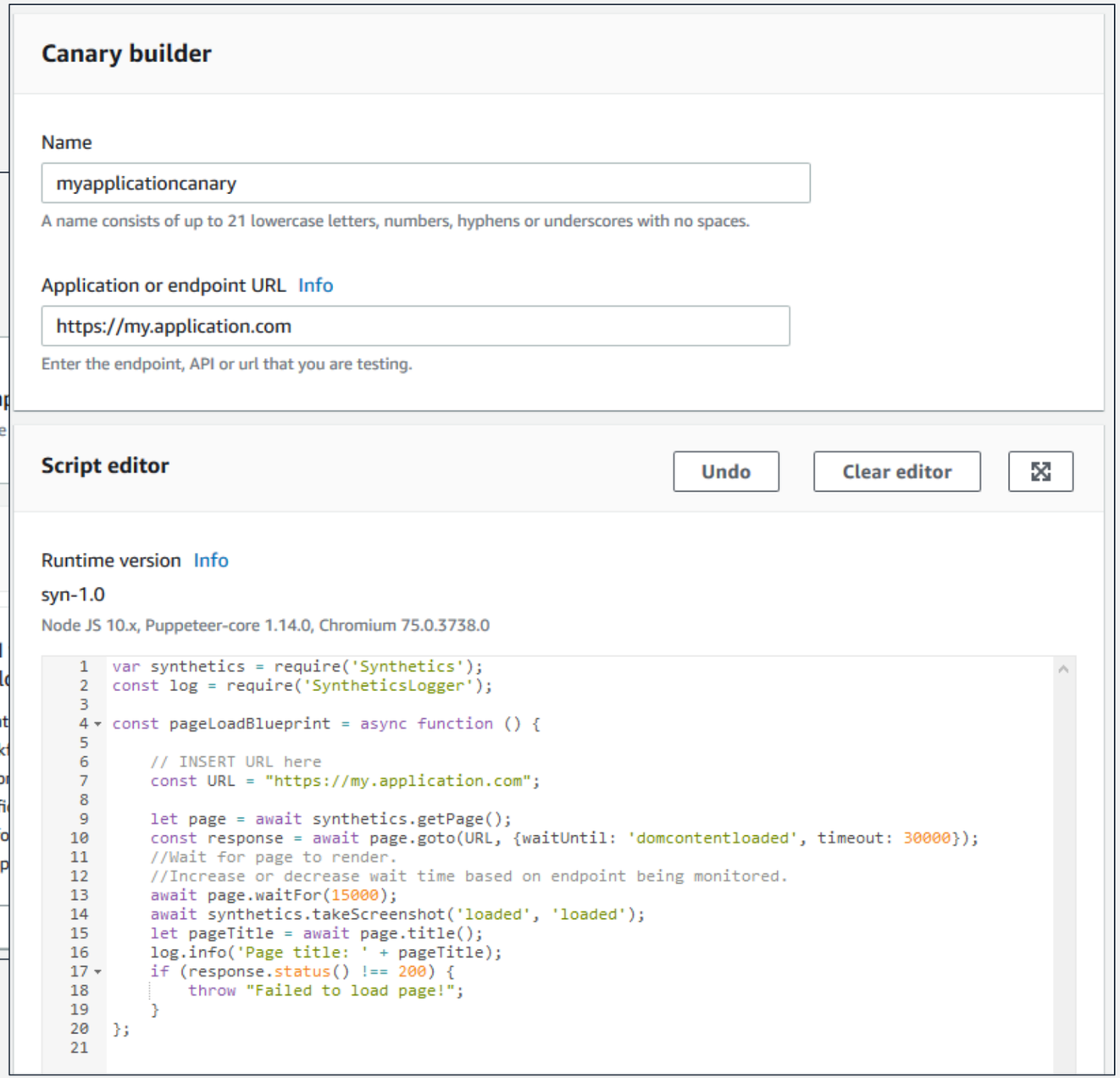
© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

https://eu-west-1.console.aws.amazon.com/cloudwatch/home?region=eu-west-1#servicelens:map

CloudWatch Synthetics

Monitor web applications using modular, light-weight canary tests





© 2020, Amazon Web Services, Inc. or its Affiliates.

Summary

- Serverless monitoring evolves: less infrastructure, more distributed services
- Observability brings together metrics, logs, traces for understanding workloads
- CloudWatch and X-Ray: collect, monitor, act, and analyze
- Many partners also available
- Standard metrics provided as default, add your own custom metrics too
- Instrument code with client libraries for structured logging and metrics
- Visualize with Metrics Graphs, query with CloudWatch Insights
- Create custom alerts
- X-Ray tracing for end-to-end view of application requests.
- ServiceLens brings together CloudWatch metrics and logs, and X-Ray traces

© 2020, Amazon Web Services, Inc. or its Affiliates.

Thanks!

Julian Wood
Sr. Developer Advocate - AWS Serverless
@julian_wood