

The background of the image features a dark blue gradient on the left, transitioning into a large, vibrant, abstract shape on the right. This shape is composed of overlapping curved segments in shades of orange, pink, and purple, creating a dynamic, modern aesthetic.

AWS re:Invent

NOV. 27 – DEC. 1, 2023 | LAS VEGAS, NV

COM307

Seamless observability with AWS Distro for OpenTelemetry

Liz Fong-Jones

(she/her)

Field CTO; AWS Community Hero

honeycomb.io



© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.



* PEACEFUL *



Debugging is a key requirement

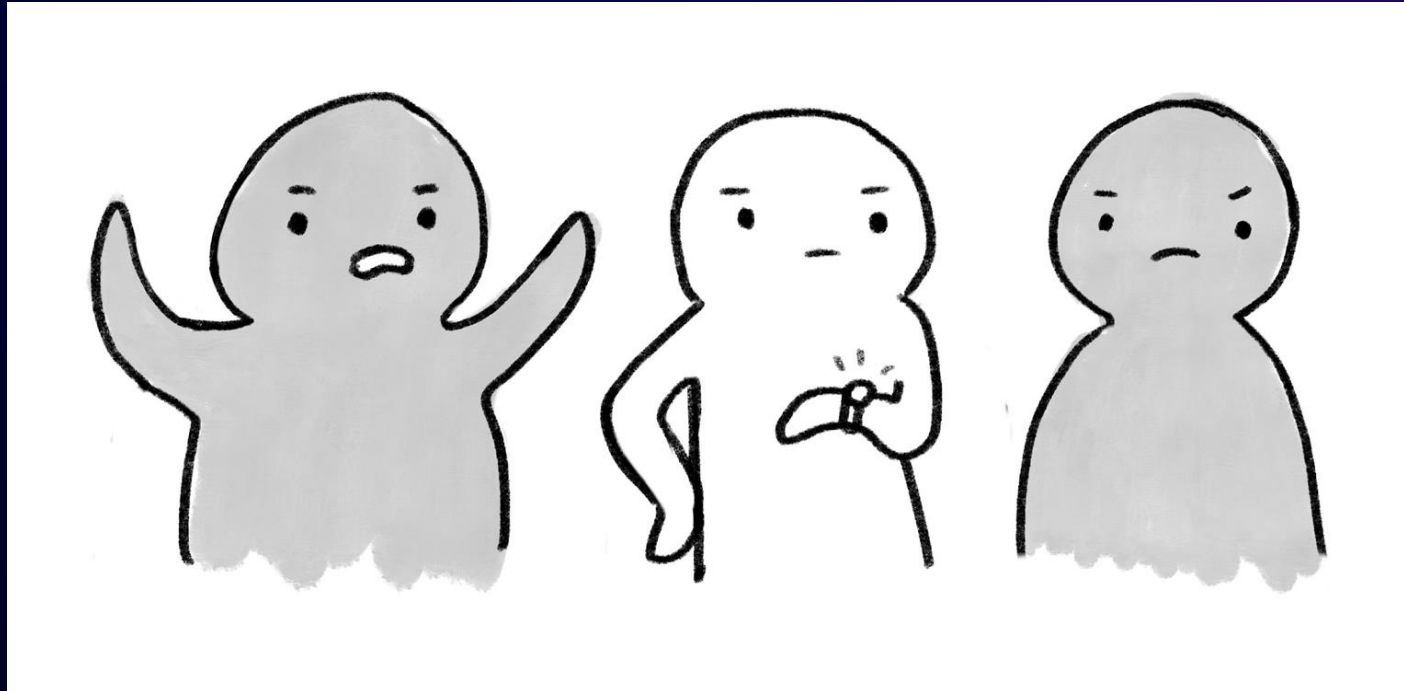
or else your services won't stay up!



What does **uptime** mean?



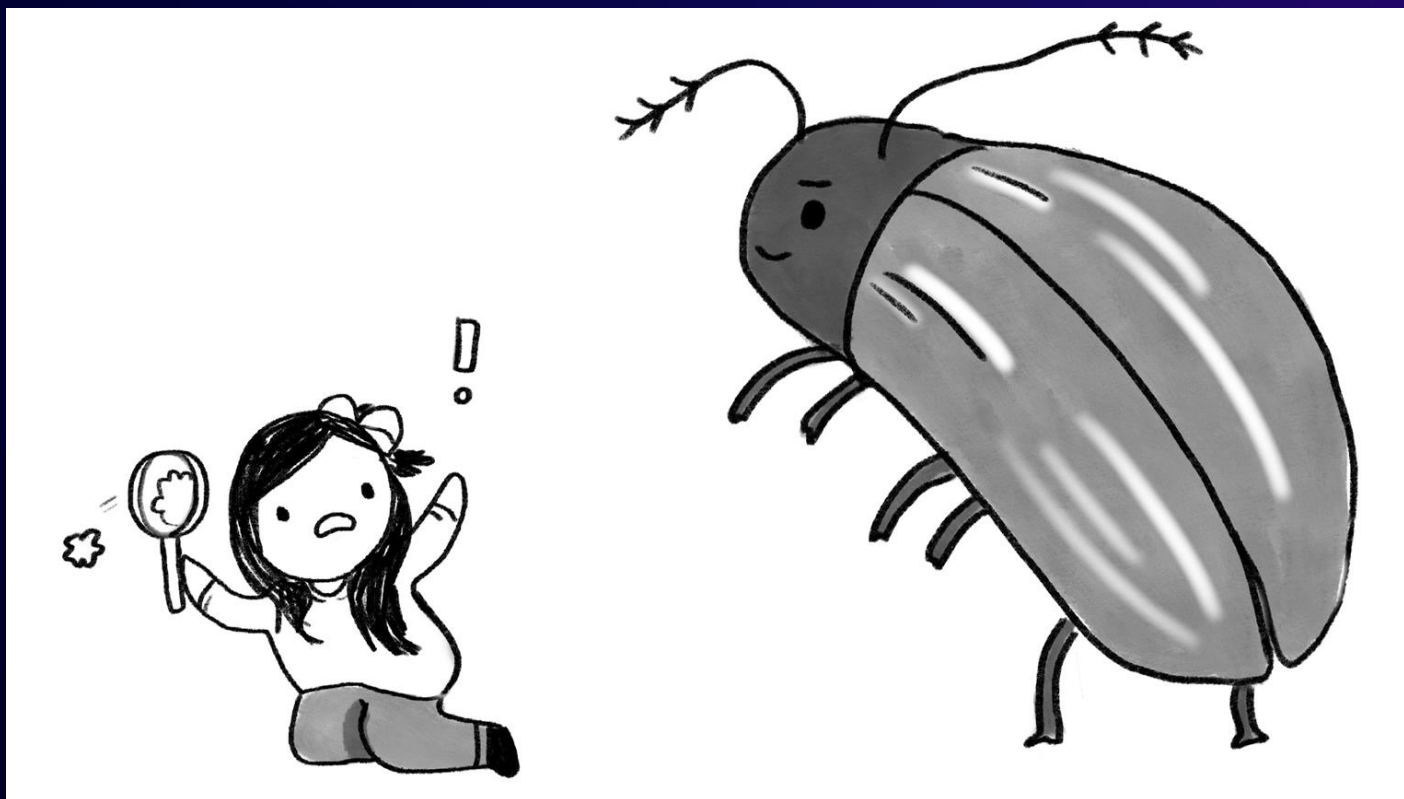
Is it **measured** in servers?



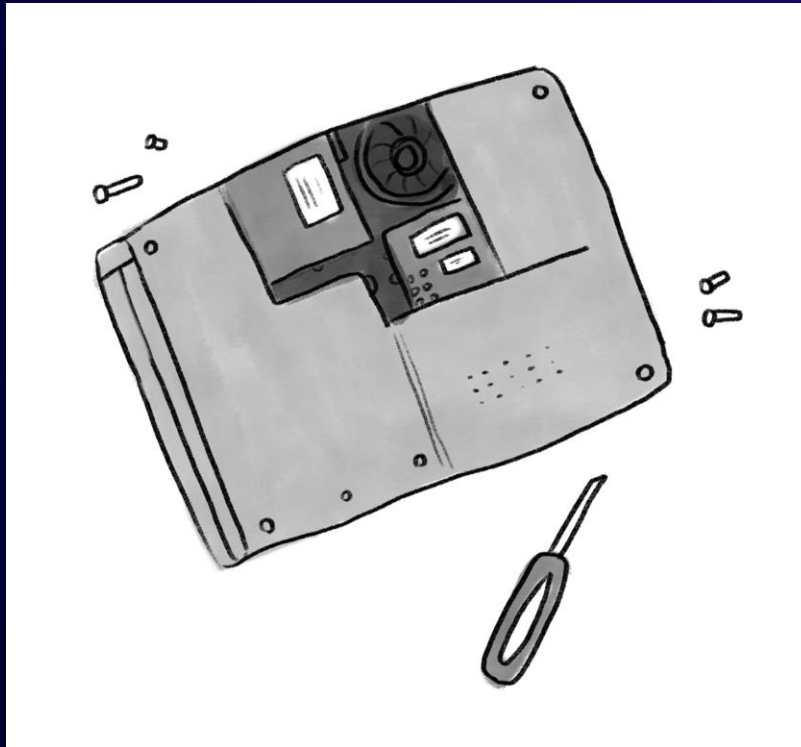
Is it **measured** in complaints?



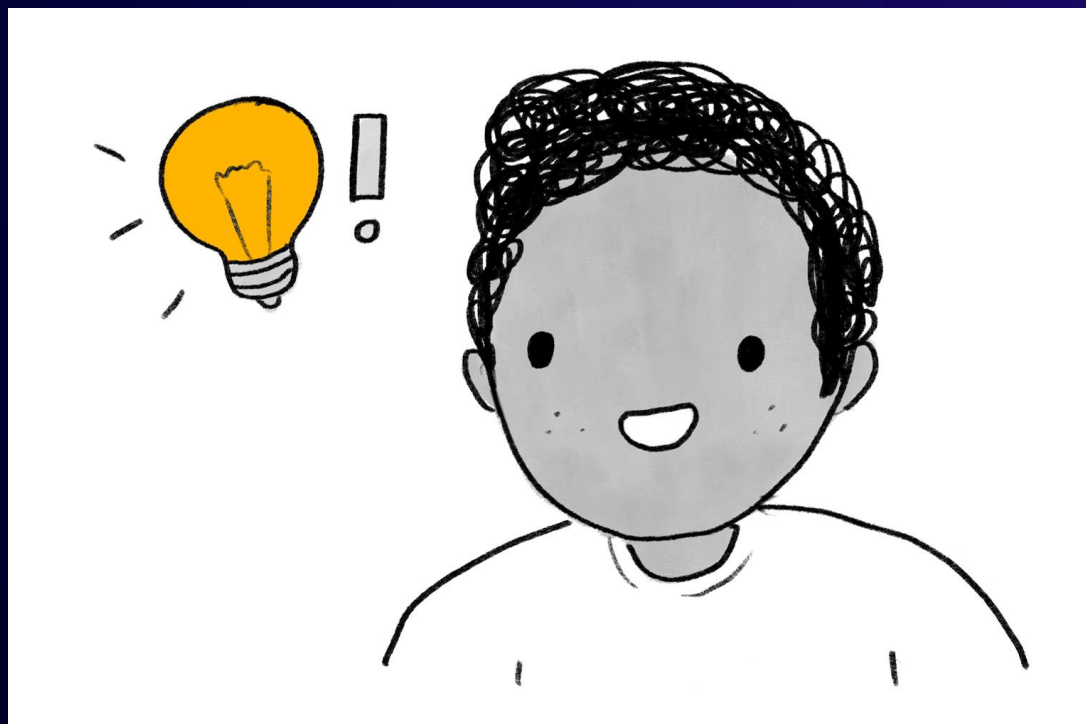
Our outages are **never identical**



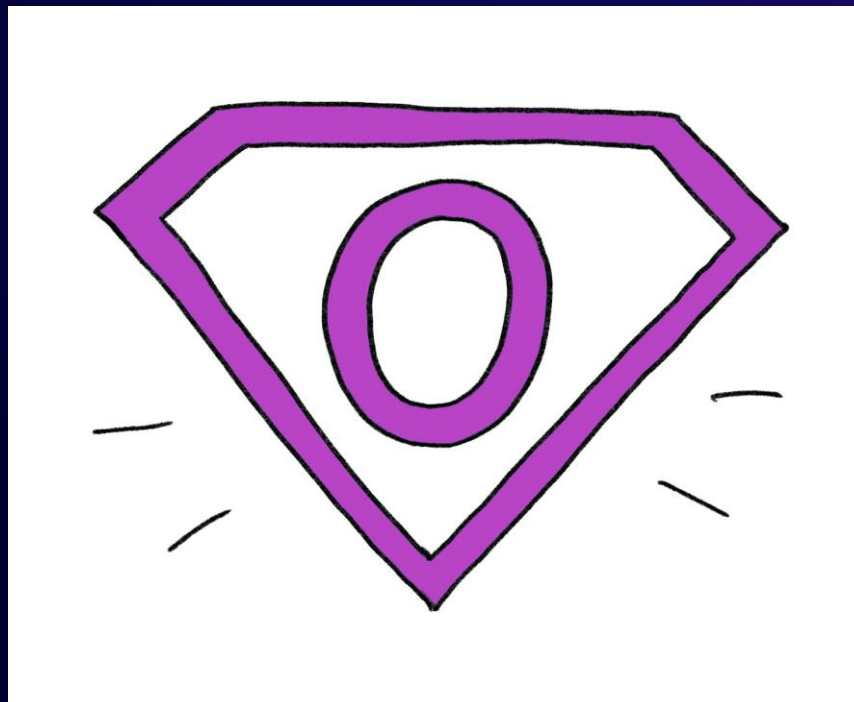
Failure modes **can't be predicted**



Support **debugging** novel cases
in production



Allow forming and testing **hypotheses**



Our services must be **observable**



AWS Well-Architected

AWS Architecture Blog

Implement observability

Implement observability in your workload so that you can understand its state and make data-driven decisions based on business requirements.

Observability goes beyond simple monitoring, providing a comprehensive understanding of a system's internal workings based on its external outputs. Rooted in metrics, logs, and traces, observability offers profound insights into system behavior and dynamics. With effective observability, teams can discern patterns, anomalies, and trends, allowing them to proactively address potential issues and maintain optimal system health.

Identifying key performance indicators (KPIs) is pivotal to ensure alignment between monitoring activities and business objectives. This alignment ensures that teams are making data-driven decisions using metrics that genuinely matter, optimizing both system performance and business outcomes.

Furthermore, observability empowers businesses to be proactive rather than reactive. Teams can understand the cause-and-effect relationships within their systems, predicting and preventing issues rather than just reacting to them. As workloads evolve, it's essential to revisit and refine the observability strategy, ensuring it remains relevant and effective.

Best practices

- [OPS04-BP01 Identify key performance indicators](#)
- [OPS04-BP02 Implement application telemetry](#)
- [OPS04-BP03 Implement user experience telemetry](#)
- [OPS04-BP04 Implement dependency telemetry](#)
- [OPS04-BP05 Implement distributed tracing](#)

[AWS Well-Architected Framework,](#)
[Operational Excellence Pillar](#)



© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Observability = outcomes through actions



We must answer questions about our systems

What characteristics did the queries that timed out at 500 ms have in common? Service versions? Client versions?

What does “healthy” generally look like?

...

Instrumenting produces data

TRACES

- End-to-end view (trace) of the lifecycle of a request in a system
- Lets you pinpoint failures and performance issues

METRICS

- Aggregated summary statistics
- Contextual information that complements traces

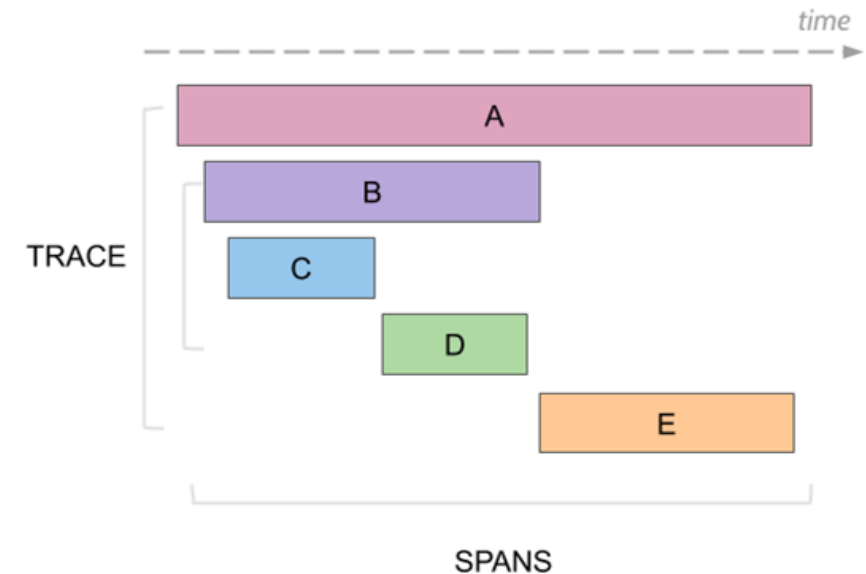
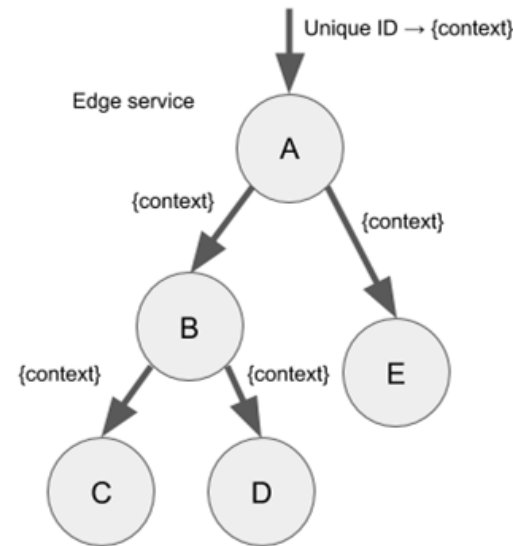
LOGS

- Detailed debugging information emitted by processes
- Ideal for when you've pinpointed a failure and want more information
... although spans in a trace are actually better for this

Why did I highlight traces?

They show how your systems are actually used by your users

- Traces are collections of spans
- Spans represent a unit of work of arbitrary length
- Spans contain structured metadata about that work (a log!)
- Spans are hierarchical



The best way to generate this data?



What is OpenTelemetry?

Also called “OTel”!

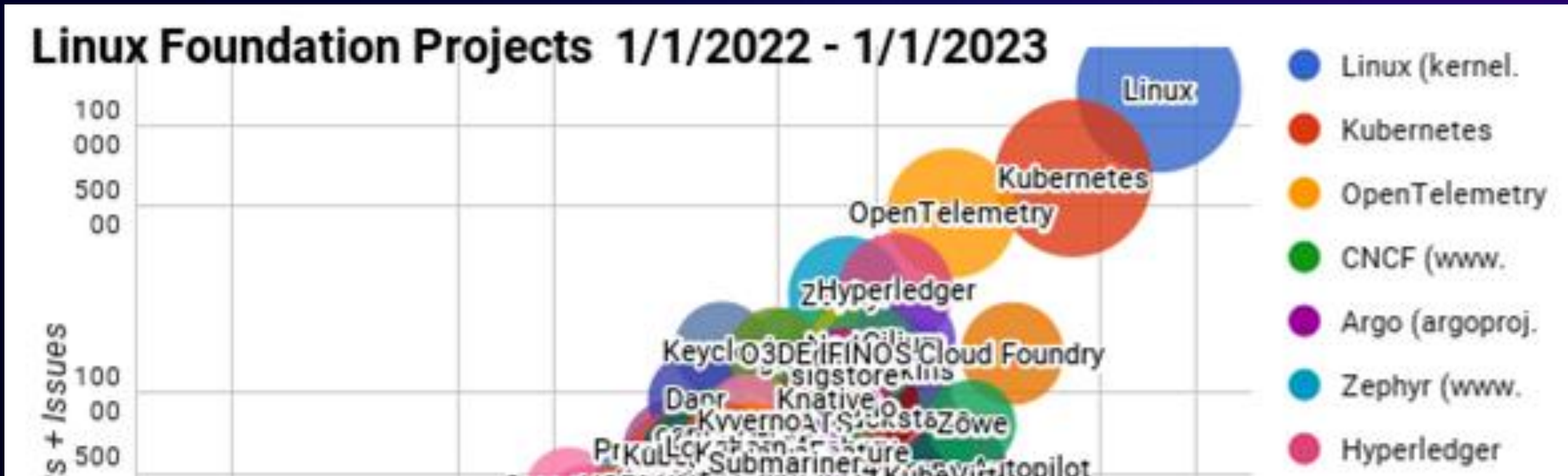


Open standard and tool set for telemetry

- Data format specification
- 11 language SDKs (Python, JS, Java, and more)
- Automatic instrumentation for libraries
- OpenTelemetry Collector
 - Receive data from different sources
 - Process data in flight
 - Export to different destinations
- Broad backend support
 - Datadog, New Relic, Splunk, Honeycomb, Lightstep, X-Ray, CloudWatch, and more



Third-largest Linux Foundation project



<https://github.com/cncf/velocity>

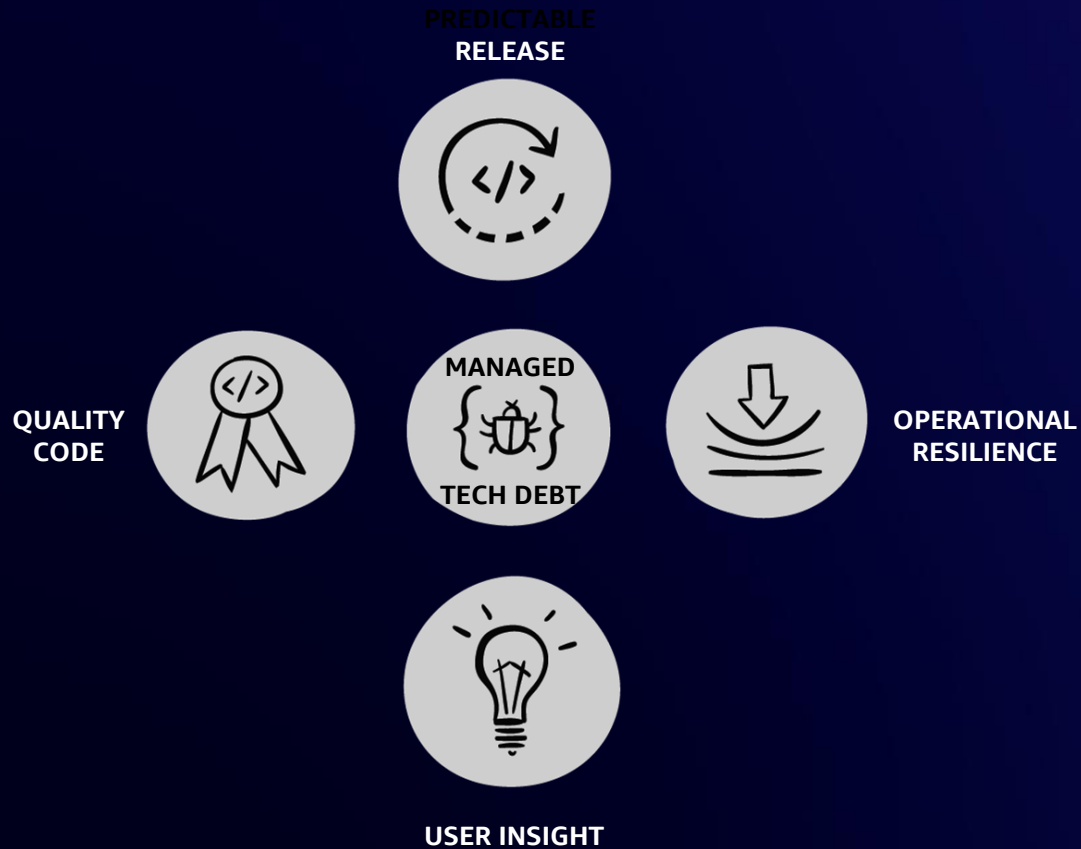
Time line

2016: OpenTracing founded
2017: OpenCensus founded
2018: OpenTelemetry formed
2019: OpenTelemetry alpha
2020: OpenTelemetry beta
2021: OpenTelemetry spec GA
2022: Metrics & SDK GAs
2023: Logs GA

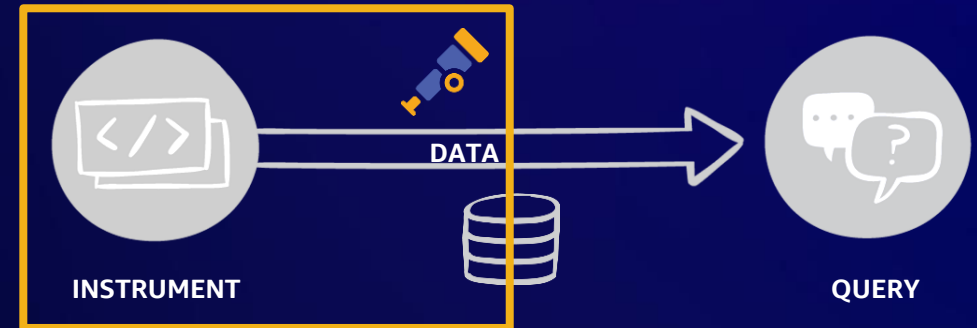


Observability = outcomes through actions

Outcomes



Actions





Liz Fong-Jones

Field CTO, Honeycomb
AWS Community Hero

Previous:
OpenTelemetry Governance Committee
OpenTelemetry Go SDK Approver

Roadmap

✓ What is O11y and why does it matter?

Pages, WAF, and OTel, oh my!

How to get started with OpenTelemetry

Just install the operator!

Telemetry routing and backends

AWS X-Ray, ISVs, and more, with some YAML

Advanced techniques

Getting the most out of your data

How to start with OTel on AWS

It's easier than you think!



Automatic instrumentation of apps

K8s & .NET, Go, Java, Node, Python: Start with the OTel Operator

Auto-injects instrumentation into runtimes

Automatically creates OTel Collectors for routing

AWS Distro for OpenTelemetry (ADOT) K8s add-on, creates collector that auto-routes to Amazon CloudWatch, Amazon Managed Service for Prometheus, and AWS X-Ray








Other languages or non-K8s:


Configure OpenTelemetry auto-instrumentation for your runtime (Java: Use ADOT version of Java instr)


Separately configure OTel Collector on each host (or pass env variable)

OTel Collector supports logs too

★ Some YAML assembly required

 Services [Alt+S]     N. Virginia  platform/lizf@honeycomb.io @ hound 


 Resource Groups & Tag Editor



AWS Distro for OpenTelemetry [Info](#)

Remove add-on

Listed by



Category


observability

Status


✓ Ready to install


Version


Select the version for this add-on.

v0.82.0-eksbuild.1 

Select IAM role

Select an IAM role to use with this add-on. To create a new role, follow the instructions in the [Amazon EKS User Guide](#) .

Inherit from node 




▼ Optional configuration settings ★

Add-on configuration schema

AWS Distro for OpenTelemetry (ADOT)

Amazon EKS supports open source APIs, libraries, and agents to collect distributed traces and metrics for application and cluster infrastructure monitoring with [AWS Distro for OpenTelemetry](#) (ADOT). Leveraging the ADOT EKS add-on enables a simplified experience for sending metrics and traces to multiple monitoring services including [AWS X-Ray](#), [Amazon Managed Service for Prometheus](#), and [Amazon CloudWatch](#). Installing cert-manager and granting EKS add-ons additional permissions are prerequisites.

Learn more 

[Amazon EKS add-on support for ADOT Operator](#)



Amazon Elastic Kubernetes Service



Clusters New

▼ Related services

Amazon ECR

AWS Batch

Documentation

Submit feedback

version before that date, it will automatically enter extended support. After the extended support preview ends, clusters on versions in extended support will be subject to additional fees. [Learn more](#)

▼ Cluster info Info

Status
✔ Active

Kubernetes version Info
1.25

Support type
✔ Standard support until
May 2024

Provider
EKS



Overview

Resources

Compute

Networking

Add-ons

Authentication

Logging

UI



Add-ons (0) Info

View details

Edit

Remove

Get more add-ons

Any category ▾

Any status ▾

< 1 >

No add-ons

No add-ons are configured for this cluster.

[Get more add-ons](#)

00:00



(Use IaC; ClickOps is for demos)

How to address Lambda

OpenTelemetry AWS Lambda Resource Detector for Golang

go reference license Apache 2.0

This module detects resource attributes available in AWS Lambda.

Installation

```
go get -u go.opentelemetry.io/contrib/detectors/aws/lambda
```

Usage

Create a sample Lambda Go application such as below.

```
package main

import (
    → "github.com/aws/aws-lambda-go/lambda"
    → sdktrace "go.opencensus.io/otel/sdk/trace"
    → lambdadetector "go.opentelemetry.io/contrib/detectors/aws/lambda"
)

func main() {
    → detector := lambdadetector.NewResourceDetector()
    → res, err := detector.Detect(context.Background())
    → if err != nil {
    →     → fmt.Printf("failed to detect lambda resources: %v\n", err)
    → }

    → tp := sdktrace.NewTracerProvider(
    →     → sdktrace.WithResource(res),
    → )
    → lambda.Start(<some lambda handler>)
}
```

Now your `TracerProvider` will have the following resource attributes and attach them to new spans:

Resource Attribute	Example Value
<code>cloud.provider</code>	aws
<code>cloud.region</code>	us-east-1
<code>faas.name</code>	MyLambdaFunction
<code>faas.version</code>	\$LATEST
<code>faas.instance</code>	2021/06/28/\$LATEST]2f399eb14537447da05ab2a2e39309de
<code>faas.max_memory</code>	128

Of note, `faas.id` and `cloud.account.id` are not set by the Lambda resource detector because they are not available outside a Lambda invocation. For this reason, when using the AWS Lambda Instrumentation these attributes are set as additional span attributes.



OpenTelemetry AWS Lambda Instrumentation for Golang

go reference license Apache 2.0

This module provides instrumentation for [AWS Lambda](#).

Installation

```
go get -u go.opentelemetry.io/contrib/instrumentation/github.com/aws/aws-lambda-go/otellambda
```

example

See [./example](#)

Usage

Create a sample Lambda Go application such as below.

```
package main

import (
    → "context"
    → "fmt"
    → "github.com/aws/aws-lambda-go/lambda"
)

type MyEvent struct {
    → Name string `json:"name"`
}

func HandleRequest(ctx context.Context, name MyEvent) (string, error) {
    → return fmt.Sprintf("Hello %s!", name.Name ), nil
}

func main() {
    → lambda.Start(HandleRequest)
}
```

Now use the provided wrapper to instrument your basic Lambda function:

```
// Add import
import "go.opentelemetry.io/contrib/instrumentation/github.com/aws/aws-lambda-go/otellambda"

// wrap lambda handler function
func main() {
    → lambda.Start(otellambda.InstrumentHandler(HandleRequest))
}
```

How to address Lambda

Wrap Lambda Invoke to get stats

Easiest way to emit telemetry spans for Lambda calling your function

Those spans still need to leave the process ...

(for Java, AWS has a one-stop shop extension that does both of these)

Add OTel Lambda layer for routing

Automatically creates an OTel Collector local to the Lambda; handles freeze/thaw

Layers continue executing after invoke returns

Lambda Layer

This layer includes a reduced version of the **AWS Distro for OpenTelemetry Collector (ADOT Collector)**, which runs as a Lambda extension.

Note: Lambda layers are a regionalized resource, meaning that they can only be used in the Region in which they are published. Make sure to use the layer in the same region as your Lambda functions.

Find the supported regions and amd64(x86_64)/arm64 layer ARN in the table below for the ARNs to consume.



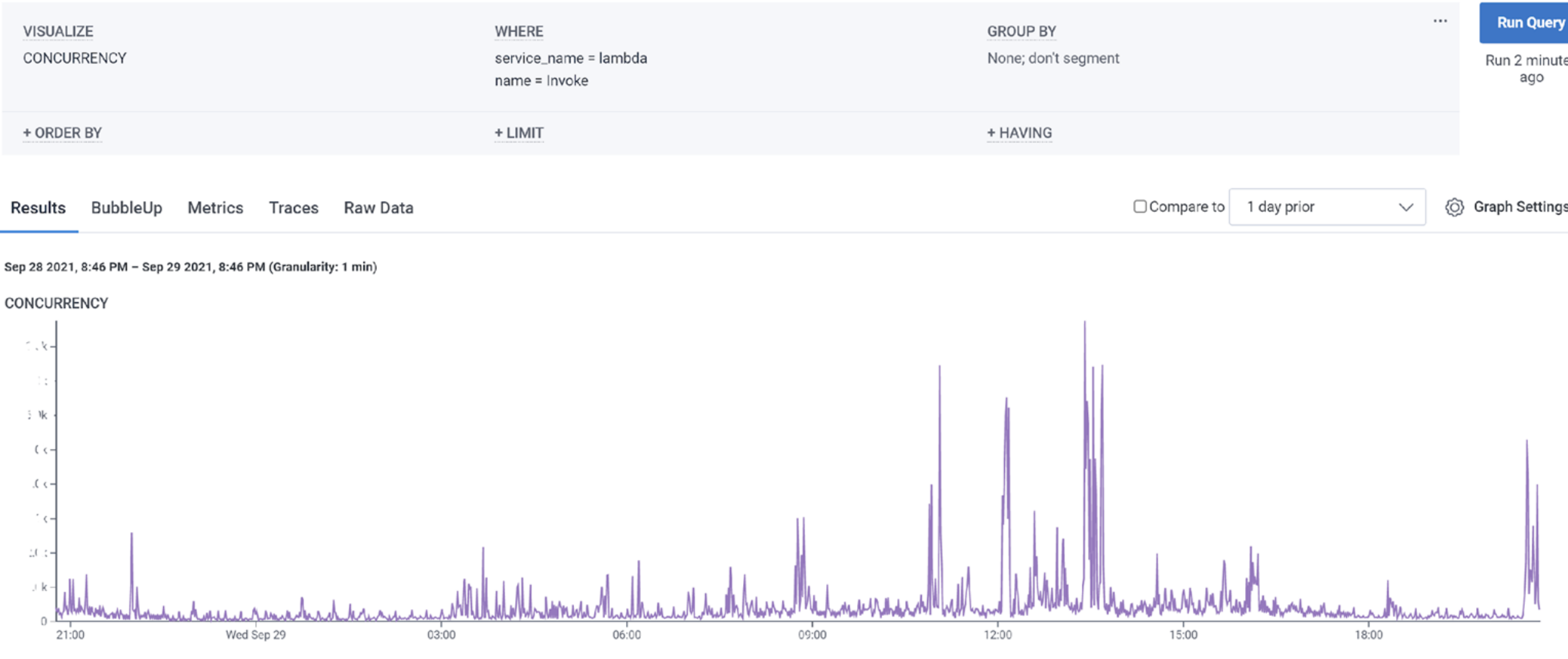
“

**Lots more compute
to play with, pretty
please! But only if I
want to play!**

– Retrievers



MOAR compute, on demand





VISUALIZE

COUNT, SUM(..., HEATMAP(...)

WHERE

attribute = value, attribute exists...

AND

GROUP BY

attribute(s)

Run Query

Clear | Cancel

+ ORDER BY

+ LIMIT

+ HAVING

Suggested Queries

Overall distribution of latencies

4:00 pm

HEATMAP(duration_ms)

2 hours

{..} Not grouped



No filters

Show the distribution of all events as a heatmap.

Run Query

Most time-consuming database queries

Shows the database queries your app is making, and the total time spent in each.

Run Query

Slowest traces

Show durations of the slowest traces over the past 2 hours.

By using a `MAX(durationMs)` calculation, we can identify the traces that took the longest overall. Click on a point in the graph to see the underlying traces.

Run Query

Trace volume

Show the number of distinct traces collected over time.

Run Query

Making the data useful

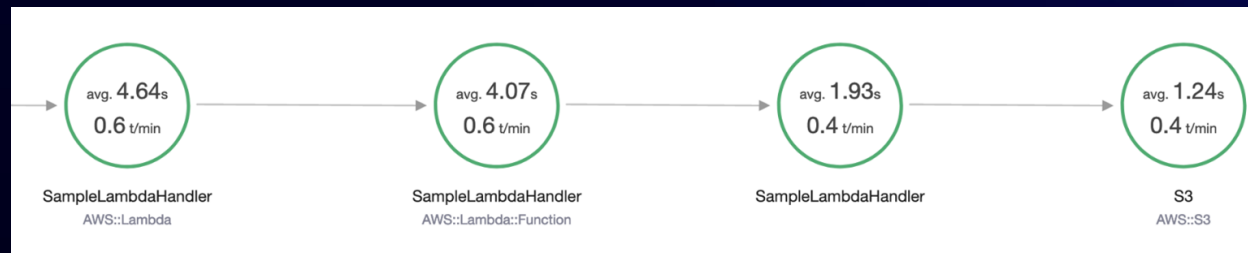
Routing the data somewhere useful

Try a variety of backends

X-Ray/Prom/CloudWatch is a start

Basic signal recording and interpretation

Supported by Amazon, lowest common denominator



ISVs can help you!

Try any OTel supporting ISV
(in no particular order):

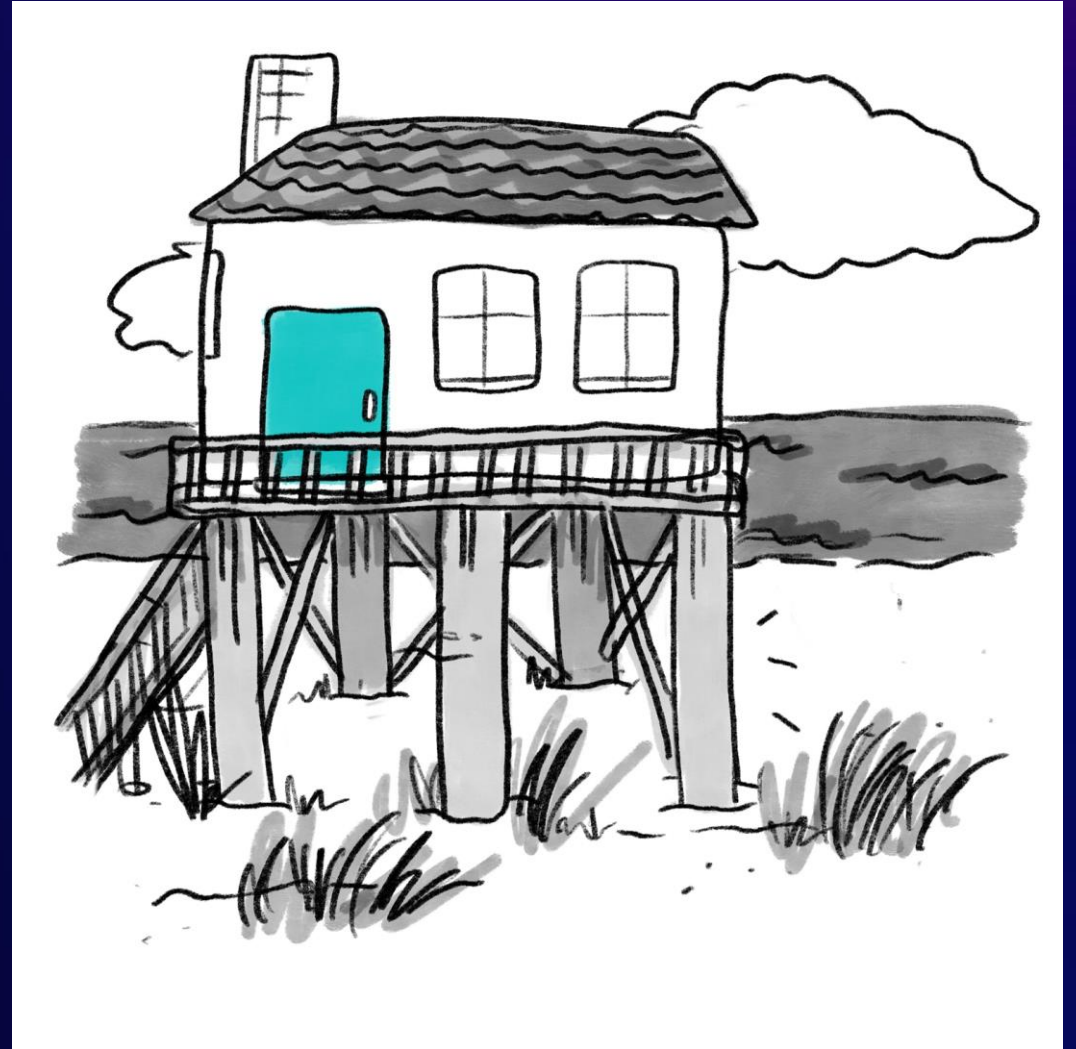
- Honeycomb
- Lightstep
- New Relic
- Splunk
- Aspecto
- ...

Getting metric data from CloudWatch

Configure Amazon Kinesis Data Firehose delivery stream export

It's buried in the CloudWatch options

You'll want to configure a compatible destination



Getting metric data from CloudWatch

aws

Services

Search

[Alt+S]

Resource Groups & Tag Editor

CloudWatch

Favorites and recents

Dashboards [New](#)

Alarms 0 1 0

Logs

Metrics [New](#)

All metrics

Explorer

Streams

X-Ray traces

Events

Application monitoring

Insights

Container Insights

Lambda Insights

Contributor Insights

CloudWatch > Metric Streams > aws2hny-production-metrics

aws2hny-production-metrics

EditStopDelete

Metric Stream Details [Info](#)

Metric Stream name

aws2hny-production-metrics

Metrics being streamed

All namespaces

Status

Running

Last updated time

11/18/2022 5:34 AM

Metric Stream ARN

arn:aws:cloudwatch:us-east-1:702835727665:metric-stream/aws2hny-production-metrics

Destination

Http

Destination details

https://api-dogfood.honeycomb.io/1/kinesis_events/cloudwatch-otlp

Kinesis Data Firehose

[aws2hny-production-metrics](#)

Service Role to write to Kinesis Data Firehose

[aws2hny-production-metrics20221117182620423300000002](#)

Output format

OpenTelemetry 0.7

aws

© 2023, Amazon

CloudShell

Feedback

Privacy

Terms

Cookie preferences

© 2023, Amazon Web Services, Inc. or its affiliates.



Amazon Kinesis



Dashboard

Data streams

Data Firehose

Managed Apache Flink [New](#)

▼ Resources

CloudFormation templates

AWS Glue Schema Registry

Analytics

Amazon Kinesis services

Collect, process, and analyze data streams in real time.

Get started

- ☒ **Kinesis Data Streams**
Collect streaming data with a data stream.
- ☐ **Kinesis Data Firehose**
Process and deliver streaming data with data delivery stream.
- ☐ **Managed Apache Flink**
Formerly Kinesis Data Analytics
Analyze streaming data with data analytics application.

[Create data stream](#)

Pricing (US East (N. Virginia))

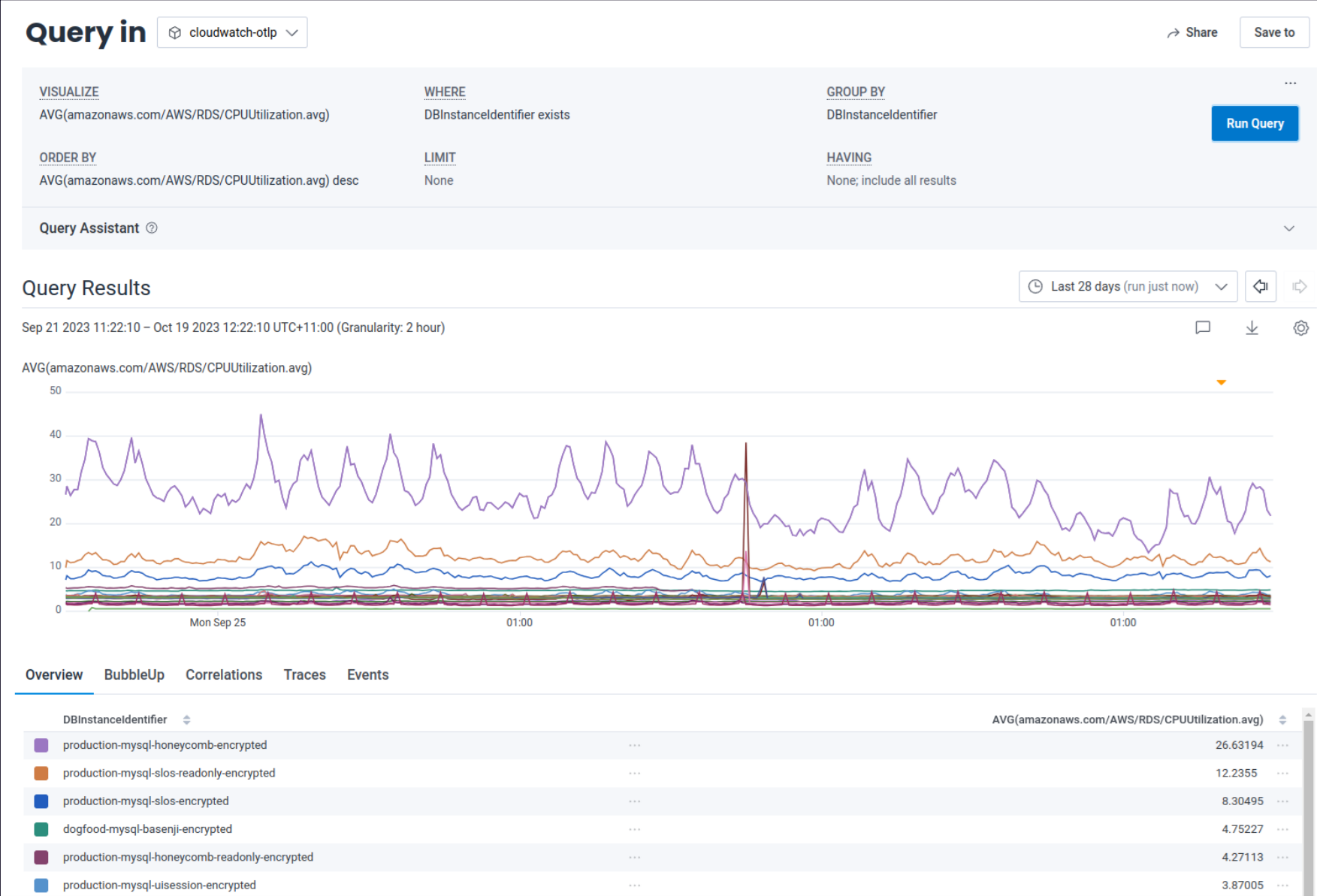
Amazon Kinesis Data Streams ▼

Shards \$0.015 per Hour

00:00



Getting metric data from CloudWatch



Query Results

Last 28 days (run just now)

Sep 21 2023 11:22:10 – Oct 19 2023 12:22:10 UTC+11:00 (Granularity: 2 hour)

AVG(amazonaws.com/AWS/RDS/CPUUtilization.avg)

Overview

BubbleUp

Correlations

Traces

Events

Advanced techniques

Make the most of OpenTelemetry



Add custom telemetry

Add custom attributes

Record data about important app layer context: UserID, CartID, CartAmount, language, etc.

And custom spans

Wrap time-consuming parts of your workload

Eliminate mystery unknown time in spans

Span attributes: Text

OwnerController.java

```
@GetMapping("/owners")
public String processFindForm(@RequestParam(defaultValue = "1") int page, Owner owner, BindingResult result,
    Model model) {
    // allow parameterless GET request for /owners to return all records
    if (owner.getLastName() == null) {
        owner.setLastName(""); // empty string signifies broadest possible search
    }
    // Grab Current Span and add attribute!
    Span span = Span.current();
    span.setAttribute("app.searchText", owner.getLastName());
}
```

Span attributes: Numbers

OwnerController.java

```
// find owners by last name
Page<Owner> ownersResults = findPaginatedForOwnersLastName(page, owner.getLastName());
if (ownersResults.isEmpty()) {
    // no owners found
    span.setAttribute("app.searchResults", 0);
    result.rejectValue("lastName", "notFound", "not found");
    return "owners/findOwners";
}
span.setAttribute("app.searchResults", ownersResults.getTotalElements());
```

You, 1 second

Wrapping a function the easy way in Java

```
import io.opentelemetry.instrumentation.annotations.WithSpan;
```

```
@WithSpan("name-this")
```

```
public void myFunction() {
```

```
    // app logic
```

```
}
```


Wrapping code with a Span (Java)

```
Span span = tracer.spanBuilder("name-this").startSpan();  
try (Scope scope = span.makeCurrent()) {  
  
    // app logic  
  
} catch (Throwable t) {  
    span.setStatus(StatusCode.ERROR);  
} finally {  
    span.end();  
}
```

Trace 50979ed8-39bb-4743-b16d-d8a33125100b

This trace contains 76,739 total spans, showing 32,000 closest to the root (and omitting the rest). [Learn more about large traces.](#)
Missing spans: Honeycomb can detect that this trace is incomplete. [Troubleshoot this](#)

76739 spans at Oct 13 2023 04:51:07 UTC+11:00

4 spans with errors

Fields

name	partition_id	0s	2s	4s	6s	8s	9.821s
MergeOSaurus	84	2.195s					
mergeLambdaResult	84		0.1186ms				
FetchPartialServeGroups	84		5.572s				
3 FetchPartial	50	7.802s					
concurrencyWait		19.2µs					
1 FetchPartialPhaseOne	50	2.711s					
12 local.getOverall	50	2.711s					
launchdarkly.NumberVariation	50	53.1µs					
6 GetSegments	50	75.86ms					
filterSegmentsForQuery	50	48.10ms					
launchdarkly.NumberVariation	50	46.1µs					
launchdarkly.NumberVariation	50	18.3µs					
63 ReadRows	50	325.7ms					
1 Process	50	325.8ms					
splitColdSegments	50	47.57ms					
207 fetchPartialsFromLambda	50	2.682s					
MergeOSaurus	50	2.682s					
MergeOSaurus	50	2.682s					
mergeLambdaResult	50		79.7µs				
FetchPartialServeGroups	50		5.091s				
3 FetchPartial	65	7.802s					
concurrencyWait		21.2µs					

FetchPartialPhaseOne

Latencies

Profile
Self

Profile
Children

Distribution of span duration



Fields

Filter fields and values in span

Timestamp	...
2023-10-12T17:51:07.909419605Z	
app.query-source	...
slo	
app.query_time_range_sec	...
86400	
dataset_id	...
64122	
duration_ms	...
2229.822069	
environment_id	...
19942	
global.availability_zone	...
us-east-1a	
global.build_id	...
978673	
global.commit_hash	...
d4a6ef52aad043877ab2d0783f8abb2d6201a2a5	
global.env	...

Add custom telemetry

AWS reports using its own semconv

OTel has basic semconv

AWS layers on its own names for containers, hosts, etc.

Set your own semconv too

Application- or organization-specific conventions prevent namespace pollution

Example: AWS detector semantic conventions

OpenTelemetry AWS Lambda Resource Detector for Golang [↗](#)

[reference](#) [license](#) [Apache 2.0](#)

This module detects resource attributes available in AWS Lambda.

Installation [↗](#)

```
go get -u go.opentelemetry.io/contrib/detectors/aws/lambda
```

Usage [↗](#)

Create a sample Lambda Go application such as below.

```
package main

import (
    → "github.com/aws/aws-lambda-go/lambda"
    → sdktrace "go.opencensus.io/otel/sdk/trace"
    → lambdadetector "go.opentelemetry.io/contrib/detectors/aws/lambda"
)

func main() {
    → detector := lambdadetector.NewResourceDetector()
    → res, err := detector.Detect(context.Background())
    → if err != nil {
    →     → fmt.Printf("failed to detect lambda resources: %v\n", err)
    → }

    → tp := sdktrace.NewTracerProvider(
    →     → sdktrace.WithResource(res),
    → )
    → lambda.Start(<some lambda handler>)
}
```

Now your `TracerProvider` will have the following resource attributes and attach them to new spans:

Resource Attribute	Example Value
<code>cloud.provider</code>	aws
<code>cloud.region</code>	us-east-1
<code>faas.name</code>	MyLambdaFunction
<code>faas.version</code>	\$LATEST
<code>faas.instance</code>	2021/06/28/[<code>\$LATEST</code>]2f399eb14537447da05ab2a2e39309de
<code>faas.max_memory</code>	128

Of note, `faas.id` and `cloud.account.id` are not set by the Lambda resource detector because they are not available outside a Lambda invocation. For this reason, when using the AWS Lambda Instrumentation these attributes are set as additional span attributes.



Configure X-Ray to OTel header prop

Sometimes you need to work with Amazon API Gateway

Use trace header compatibility to reuse TraceID

Then feed API Gateway logs from Amazon S3 in separately

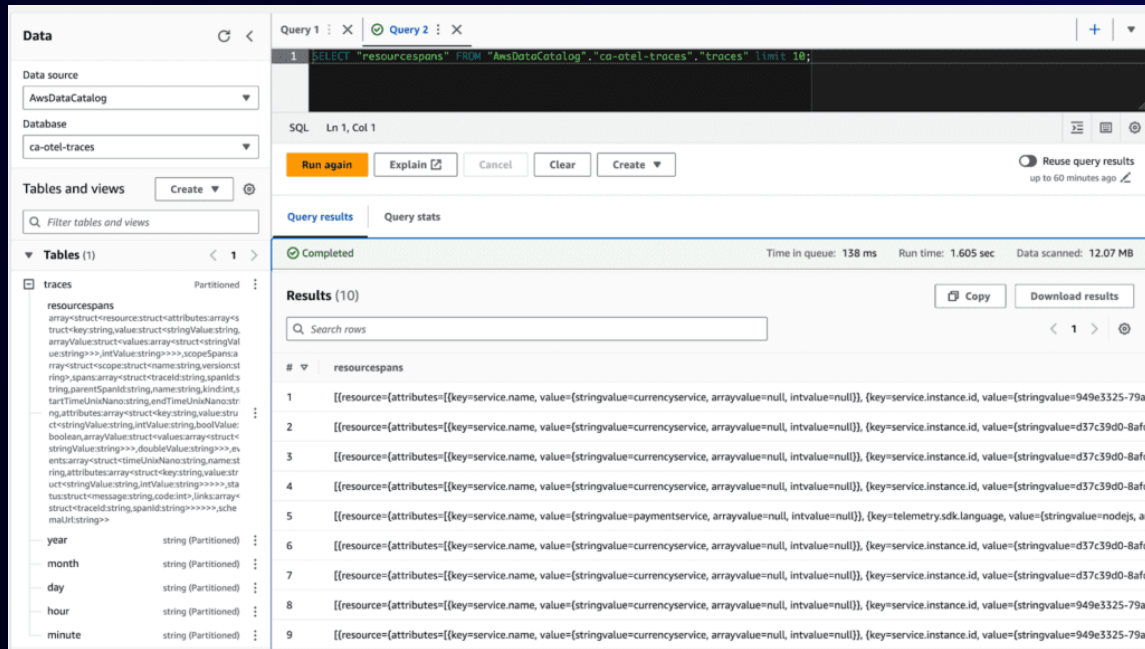
(Amazon, please support W3C TraceContext in API Gateway!)



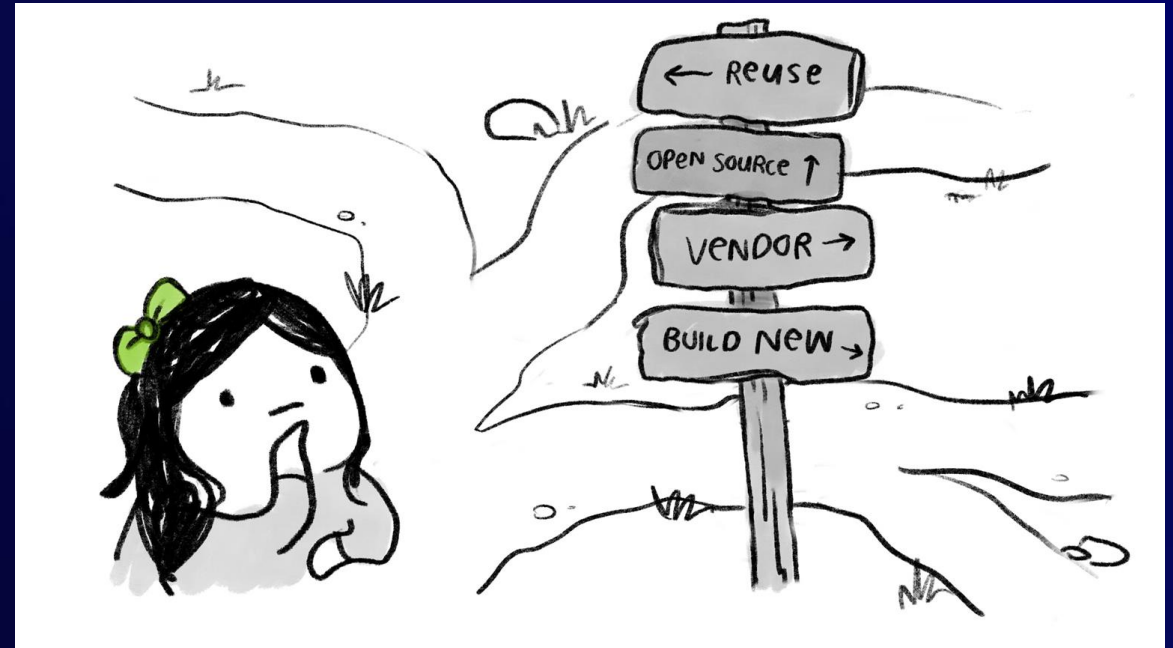
Tee data to multiple places

Infinite data storage is possible by teeing to S3 and using AWS Glue schemas and Amazon Athena

See our blog!



The screenshot shows the Amazon Athena console interface. On the left, the 'Data' pane shows the 'ca-otel-traces' database and a table named 'traces'. The 'traces' table schema is displayed, showing columns like 'resourcespans' (array<struct>), 'year' (string), 'month' (string), 'day' (string), 'hour' (string), and 'minute' (string). The main pane shows a SQL query: `SELECT "resourcespans" FROM "AwsDataCatalog"."ca-otel-traces"."traces" limit 10;`. The query is completed, and the results are displayed in a table with 10 rows. The results show JSON-like structures for 'resourcespans'.



Measure your CI/laC/console jobs



otel-cli by Equinix is handy
as is the OTel Chef and
Terraform integration
as is OTel support in your
favorite CI tool



Conclusion

- ✓ **What is O11y and why does it matter?**
Pages, WAF, and OTel, oh my!
- ✓ **How to get started with OpenTelemetry**
Just install the operator!
- ✓ **Telemetry routing and backends**
X-Ray, ISVs, and more, with some YAML
- ✓ **Advanced techniques**
Getting the most out of your data



* PEACEFUL *

Find me at the Modern Apps & Open Source Zone at 3pm, or at Honeycomb booth all week!



honeycomb.io/liz
@lizthegrey
linkedin.com/in/efong



Thank you!

Liz Fong-Jones

honeycomb.io/liz

[@lizthegrey](https://twitter.com/lizthegrey)

linkedin.com/in/efong



Please complete the session
survey in the mobile app