Spend Less Time Diagnosing Customer Issues & Optimize Engineering Efficiency with Observability

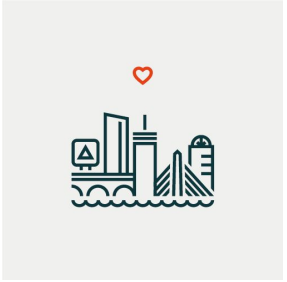
Jan Schulte, Solution Architect

Twitter: @_jan_schulte_

epsagon

Hello!

Jan Schulte
Solution Architect @ Epsagon
Living In Boston
From Berlin



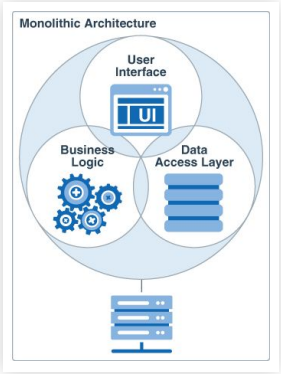
epsagon

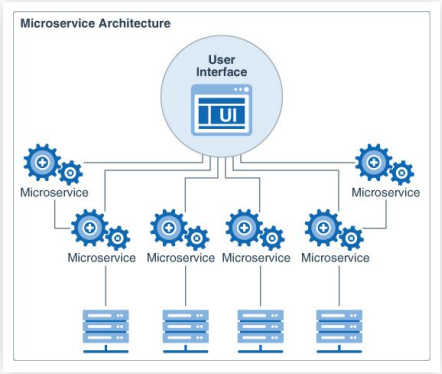
Agenda

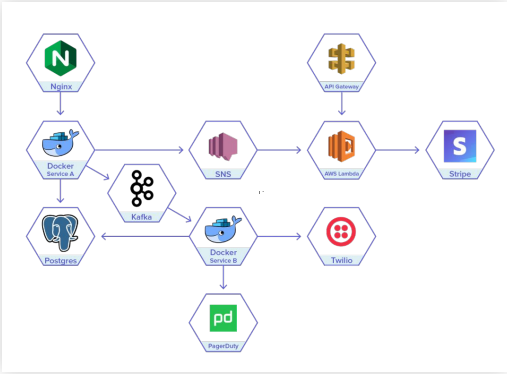
- Microservices: The New Normal
- Why Traditional Observability isn’t effective
- Efficient Observability for Microservices

Microservices: The New Normal

The Rise of Microservices



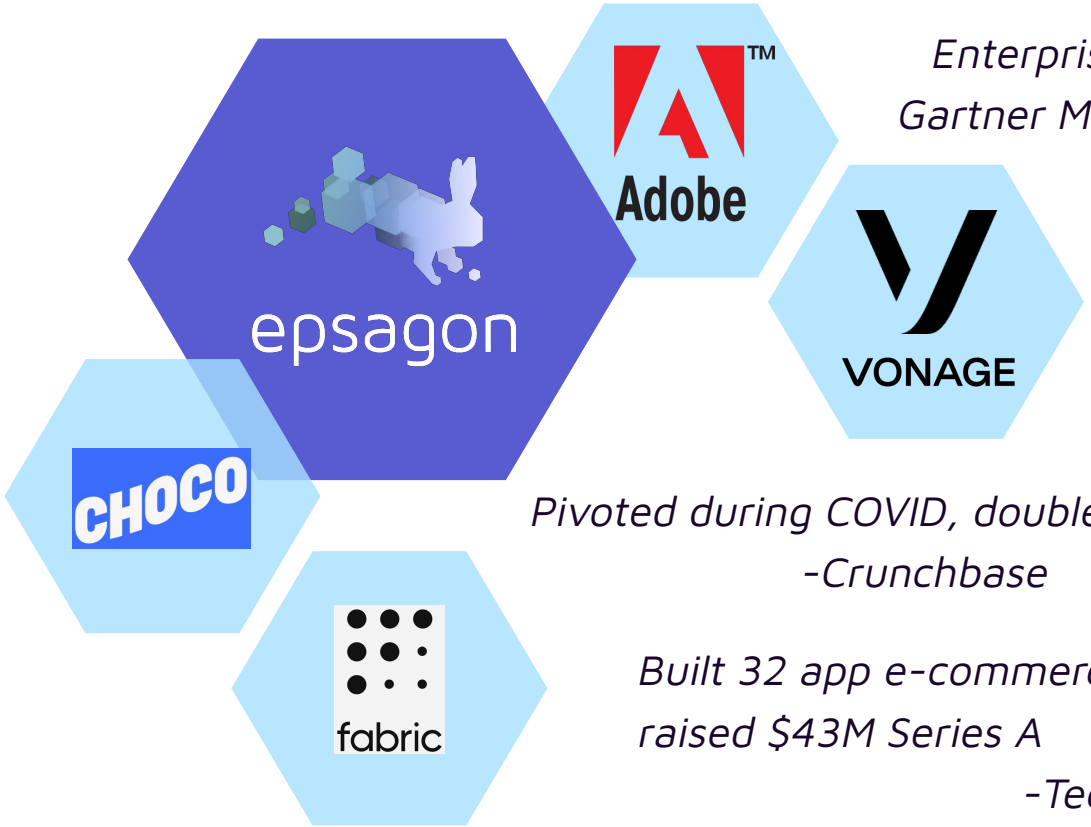




Host-based Monolithic → Host-based Distributed → Abstracted-host Highly Distributed

epsagon

Companies Winning with Microservices



Enterprise Cloud named "Leader" in seven Gartner Magic Quadrants and Forester Waves

Now releases features "Twice as Fast with Half the Effort"

Pivoted during COVID, doubled valuation
-Crunchbase

Built 32 app e-commerce platform in Seed stage, raised $43M Series A
-TechCrunch

“Everything fails, all the time”

- Werner Vogels, AWS CTO

So What's the Catch?

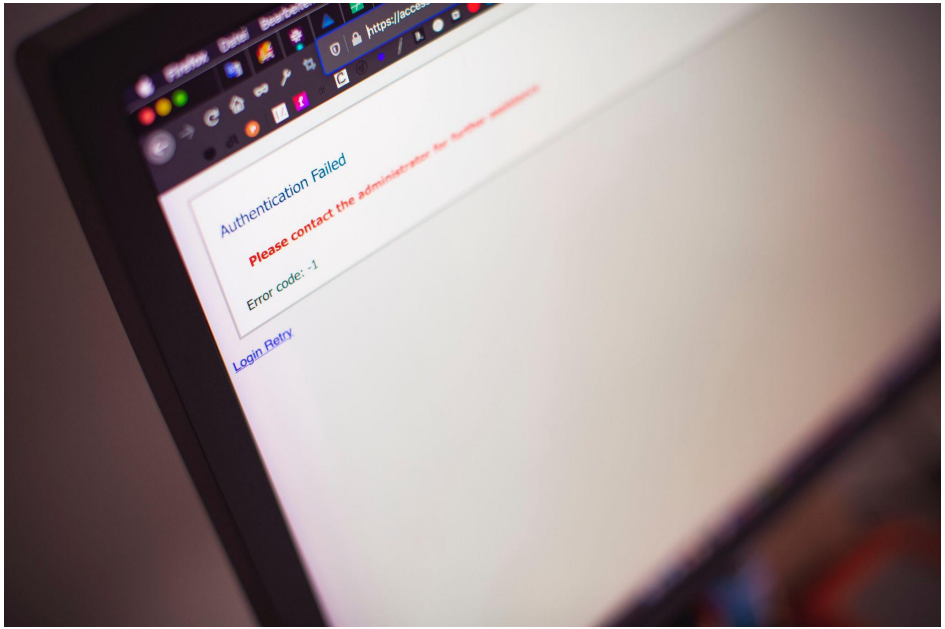
Cloud service + 3rd party APIs are difficult to troubleshoot

Thousands of containers, functions, and services with a wide variety of behaviors

Logic shifts from the code within a service to the calls between microservices

epsagon

The user finds an issue before you do



epsagon

Business Impacts

74%
of issues are found by customers before Ops

$7,900/min
Downtime costs on average

DETECT

IDENTIFY

Identifying the root cause is painful in distributed environments

RESOLVE

IMPROVE

Improve operational and engineering efficiencies

> “...when there’s a live problem **we can lose tens of thousands of dollars every hour**. Saving 45 minutes in troubleshooting a live issue is critical for the business.

VIA

epsagon

Achieving Observability in Microservices

Combining metrics, logs, and traces for observability is the only way to understand complex environments

Metrics tell us the **"what"**

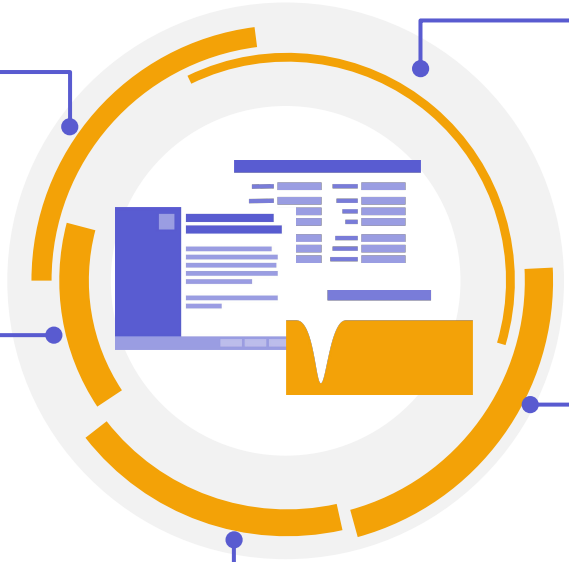
Logs tell us the **"why"**

Traces tell us the **"where"**

epsagon

The Need for a New Observability Outlook

Observability Workarounds



Traditional APM
sees within services, but not between services

Infrastructure Monitoring
alerts DevOps to issues, but does not give Devs context to remediate

Log Aggregators
hours of manually correlation that bottlenecks team knowledge in one SME

Manually Instrumented Distributed tracing
Requires heavy investment in time and FTE to setup, maintain and manage edge cases

APM with Microservice Add-ons
premium price to have partial workarounds on one platform

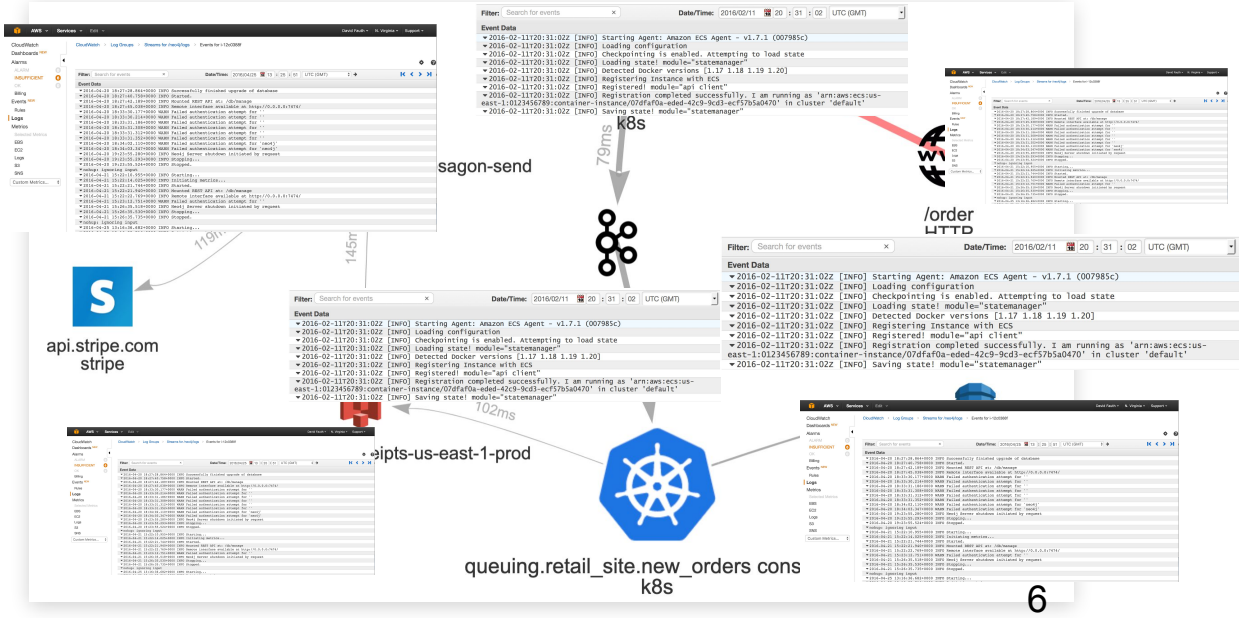
epsagon

Troubleshooting

Where do you start?

Manual Log Correlation?

Troubleshooting with Logs Doesn't Work



Finding specific logs out of billions of microservice executions can take hours, or even days.

Correlating these logs across dozens of microservices can be next to impossible.

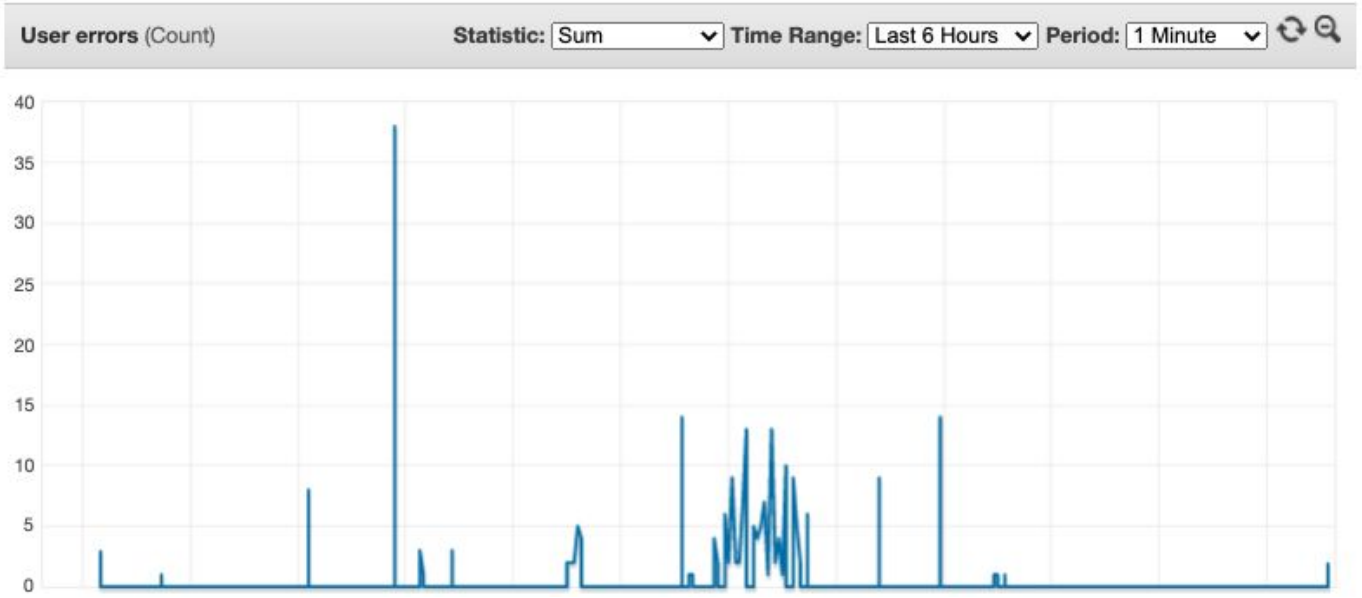
epsagon

Logs?

```
[http-nio-8080-exec-10] INFO io.jaegertracing.internal.reporters.LoggingReporter - Span reported:
615f47e4c32f589d:4e8220be4a768563:615f47e4c32f589d:1 - placeNewOrder

[http-nio-8080-exec-10] INFO io.jaegertracing.internal.reporters.LoggingReporter - Span reported:
615f47e4c32f589d:615f47e4c32f589d:0:1 - POST

[kafka-producer-network-thread | producer-1] INFO io.jaegertracing.internal.reporters.LoggingReporter - Span reported:
615f47e4c32f589d:9b14b78b08321244:4e8220be4a768563:1 - produce

09:26:16.894 [http-nio-8080-exec-27] INFO  com.epsagon.java.rest.OrdersService - placing new order {}

09:26:16.894 [http-nio-8080-exec-27] INFO  c.epsagon.java.kafka.producer.Sender - sending new order='NewOrder{itemId=0,
username='9a7ed47bfe21c01387fa3d93d3eacb',
discountCode='XMASSAVE30', quantity=4}' to topic='queuing.retail_site.new_orders'

09:26:17.242 [http-nio-8080-exec-27] ERROR Missing required parameter in input: "Key"
Unknown parameter in input: "Item", must be one of: TableName, Key, AttributeUpdates, Expected, ConditionalOperator,
ReturnValues, ReturnConsumedCapacity, ReturnItemCollectionMetrics, UpdateExpression, ConditionExpression,
ExpressionAttributeNames, ExpressionAttributeValues
```

epsagon

Metrics?



epsagon

Things missing?

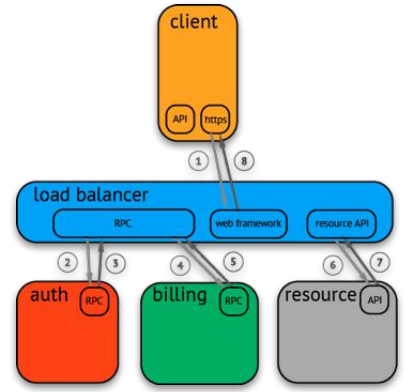
- How do we correlate between metrics and logs?
- How do we correlate data between different services?
- How do we find the **where** when something goes wrong?

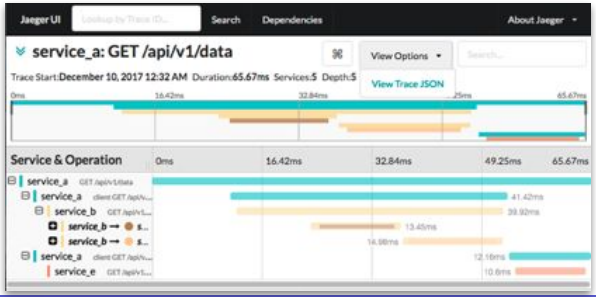
Observability using Distributed Tracing

What is Distributed Tracing?

*"A **trace** tells the story of a transaction or workflow as it propagates through a distributed system."*

Since distributed tracing connects every request in a transaction, it allows you to know and see what's happening to every service component and app in production



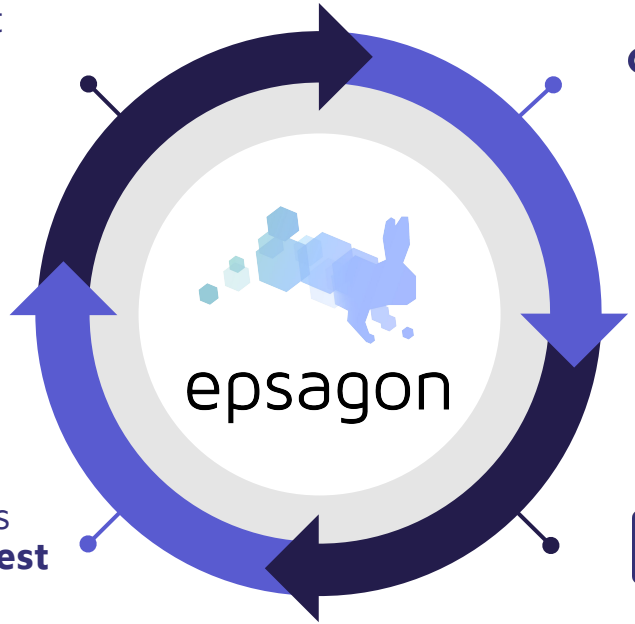


epsagon

Engineering Flywheel + Observability

Quickly troubleshoot API issues during **Production**

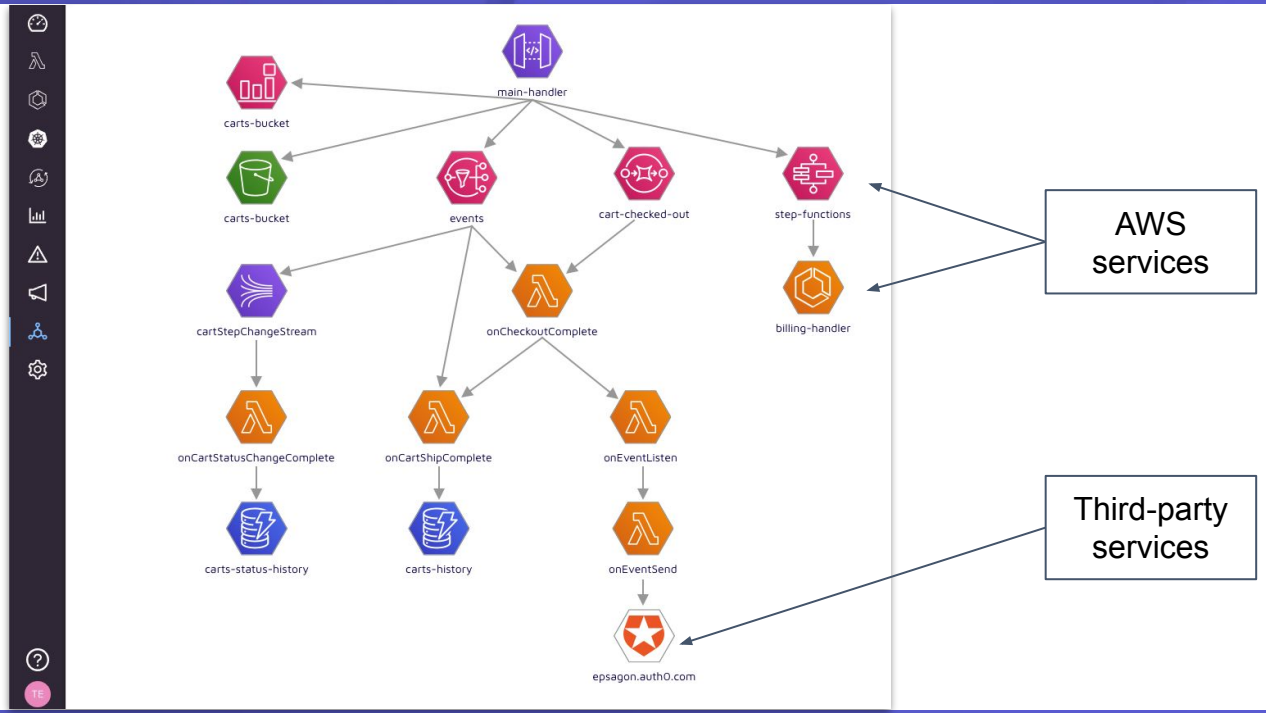
Accurately predict dependencies during **Design**



Easily catch business logic issues during **Test**

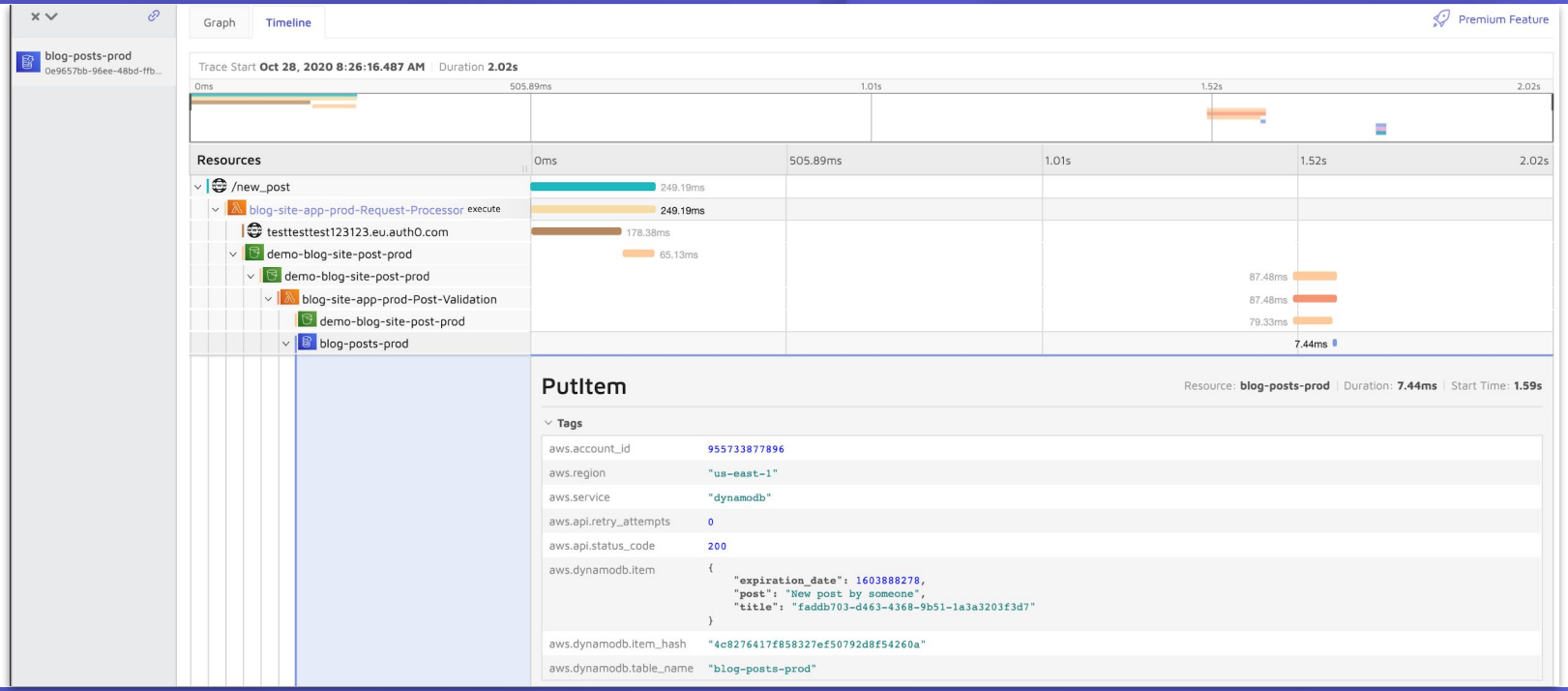
Confidently make changes to code during **Implementation**

Visualize and Understand



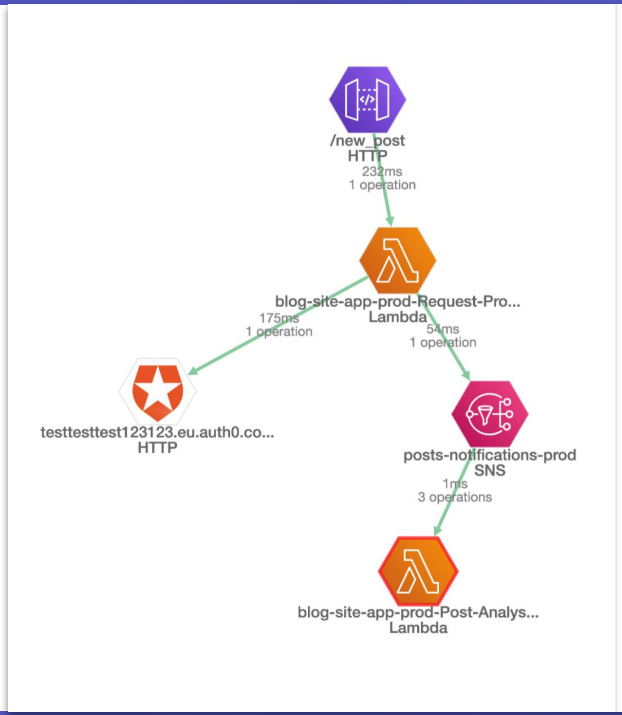
epsagon

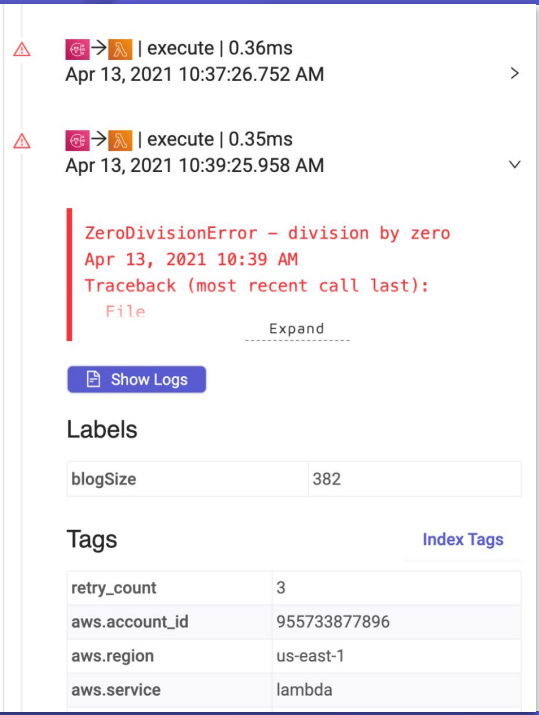
Where Does Our Code Spend Time?



epsagon
epsagon

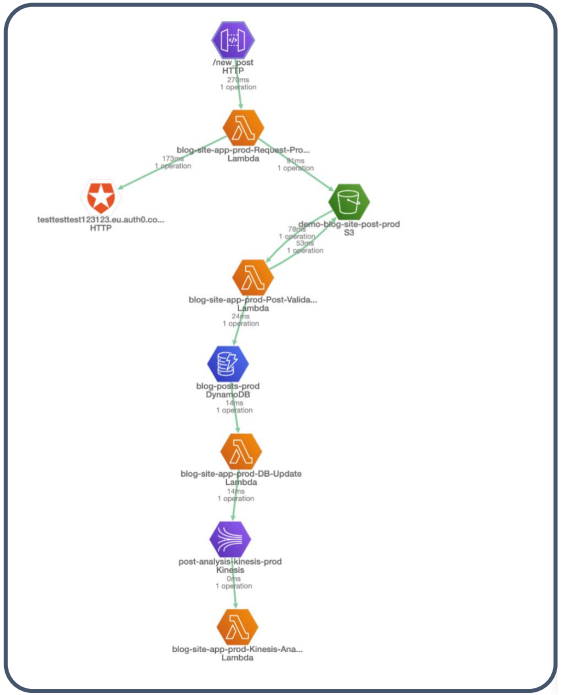
Bring Focus to the Problems





epsagon

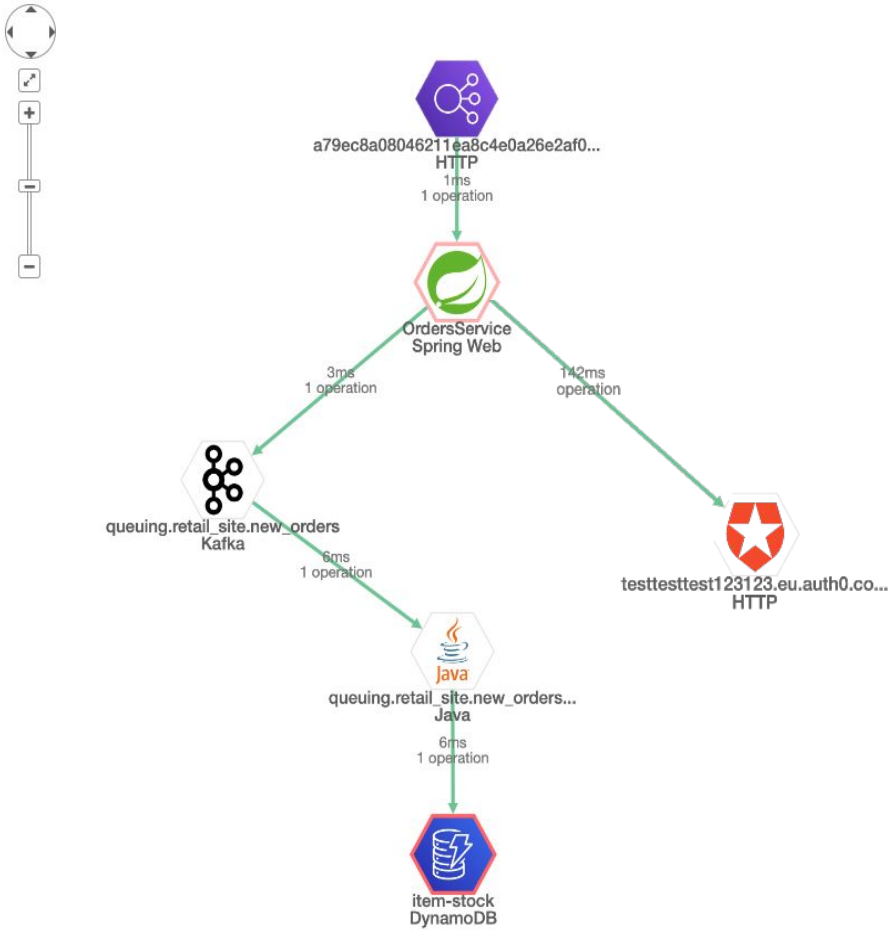
All the things mentioned + Payload information!



http.request.body

{ 4 items
"validate" : true
"title" : "Netflix event"
"post" :
"Netflix is excited to be heading back to Las Vegas for AWS re:Invent at the end of the month! Many Netflix engineers and recruiters will be in attendance, and were looking forward to meeting and reconnecting with cloud enthusiasts and Netflix OSS users. Were posting the schedule of Netflix talks here to make it a bit easier to find our speakers at re:Invent. Well also have a booth on the expo floor, so please stop by and say hello! At Netflix, we make explicit tradeoffs to balance our four key focus domains of innovation, reliability, security, and efficiency to ensure our

epsagon



| UpdateItem | 5.73ms

Sep 14, 2020 8:01:18.326 PM

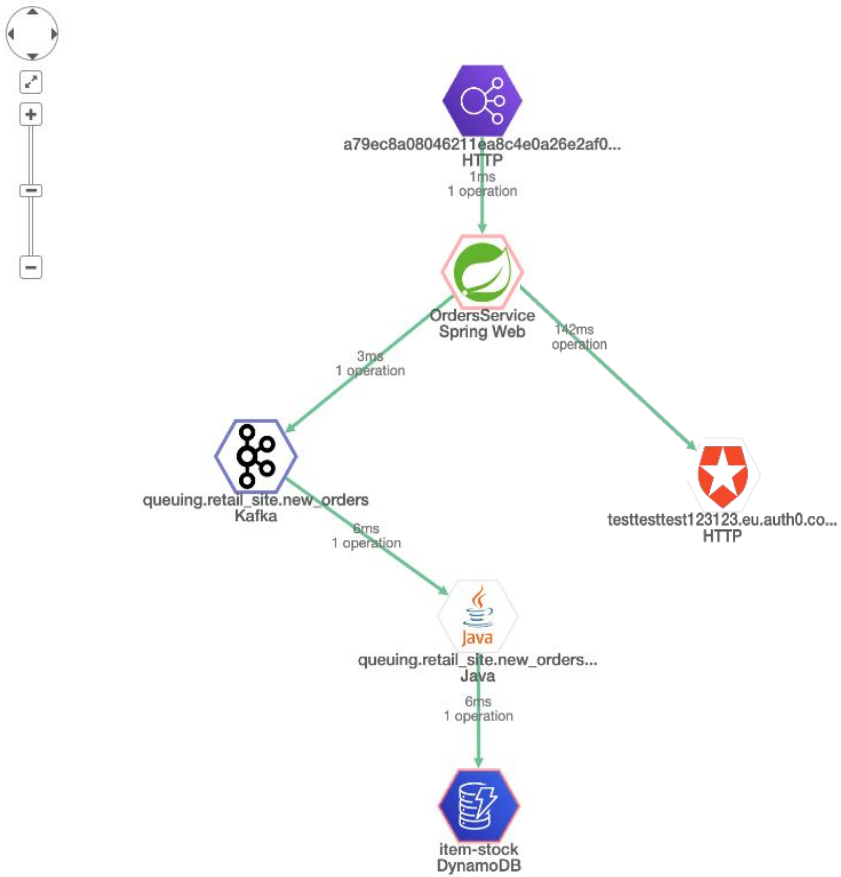
An error occurred (ValidationException) when calling the PutItem operation: One or more parameter values were invalid: Missing the key id in the item

Collapse

Tags

Index Tags

component	aws-sdk
error	True
hostname	stock-updater-856884bbd6-9t97s
ip	100.96.3.58
is_k8s	true
k8s_pod_name	stock-updater-856884bbd6-9t97s
aws.agent	aws-sdk
aws.agentVersion	>1.11.0
aws.endpoint	https://dynamodb.us-east-1.amazonaws.com
aws.operation	PutItemRequest
aws.region	us-east-1
aws.service	AmazonDynamoDBv2
env.runtime	opentracing-java
epsagon.version	Java-0.35.4
http.method	POST
http.url	https://dynamodb.us-east-1.amazonaws.com
span.kind	client
aws.dynamodb.table_n...	item-stock



queuing.retail_site.new_orders (1 operation)

Service Map

| produce | 2.66ms
Sep 14, 2020 8:01:17.260 PM

Tags

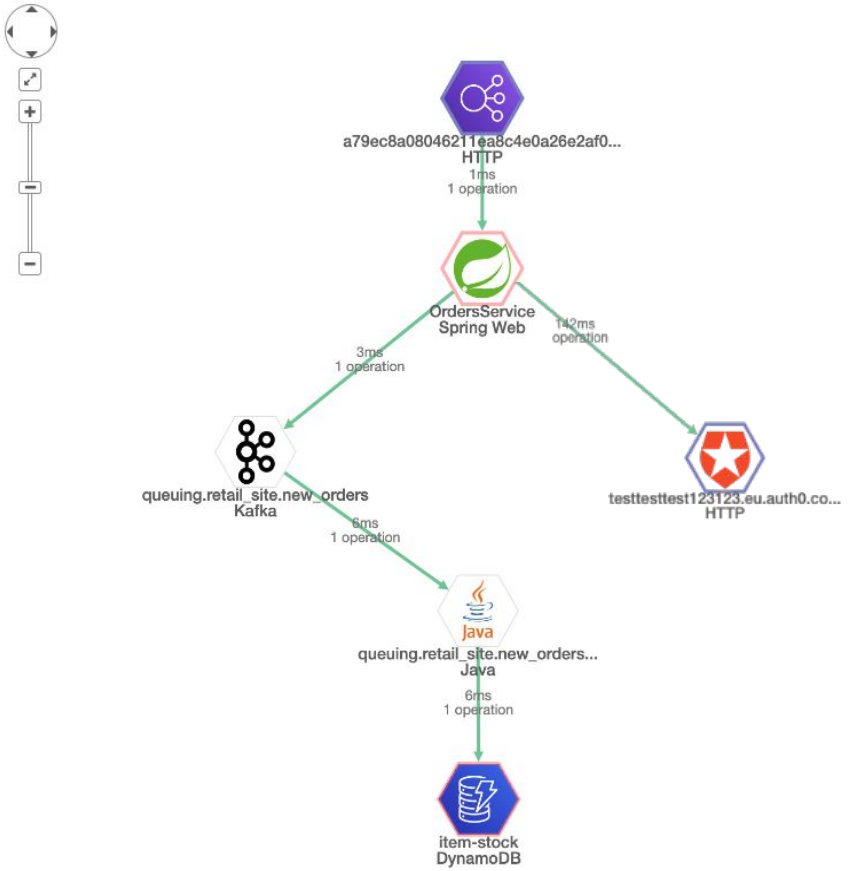
Index Tags

component	kafka-clients-0.11
hostname	orders-service-64cbdfcb5c-b2ghw
ip	100.96.1.35
is_k8s	true
k8s_pod_name	orders-service-64cbdfcb5c-b2ghw
env.runtime	opentracing-java
epsagon.version	Java-0.35.4
kafka.key	null
span.kind	producer

JSON Tags

kafka.value


```
{ 4 items
  "itemId" : 1
  "username" : ""
  "discountCode" : "DEL15"
  "quantity" : 122
}
```



Tags

Index Tags

http.host	testtesttest123123.eu.auth0.com
http.scheme	https
http.status_code	401
http.request.path	/api/v2/users/auth0%7C5ba1a9227dc7232e1aec4fd0

JSON Tags

http.request.headers

http.response.body

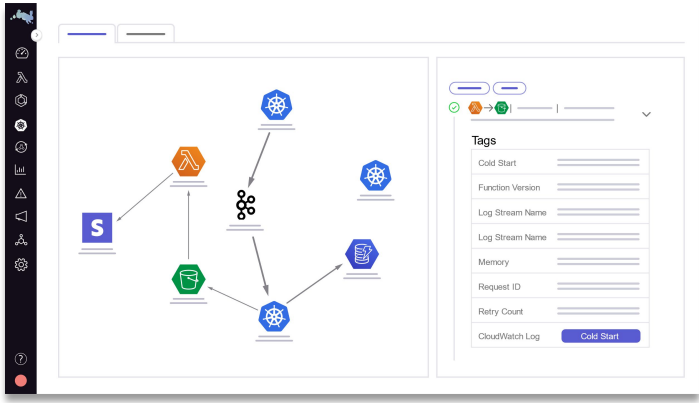

```
{ 4 items
  "message" : "Expired token received for JSON Web Token validation"
  "statusCode" : 401
  "error" : "Unauthorized"
  "attributes" : { 1 item
    "error" : "Expired token received for JSON Web Token validation"
  }
}
```


http.response.headers


```
{ 21 items
  "CF-Cache-Status" : "DYNAMIC"
```

Summary

- Microservices-based applications bring unique benefits and challenges
- Advantages of using Distributed Tracing approach for AWS and Third-party services
- Use Observability for
  - Keeping track of the architecture
  - Detecting performance issues and reduce MTTR



Be PROACTIVE not REACTIVE

epsagon

Thank you!

epsagon