The present and FUTURE of Serverless OBSERVABILITY

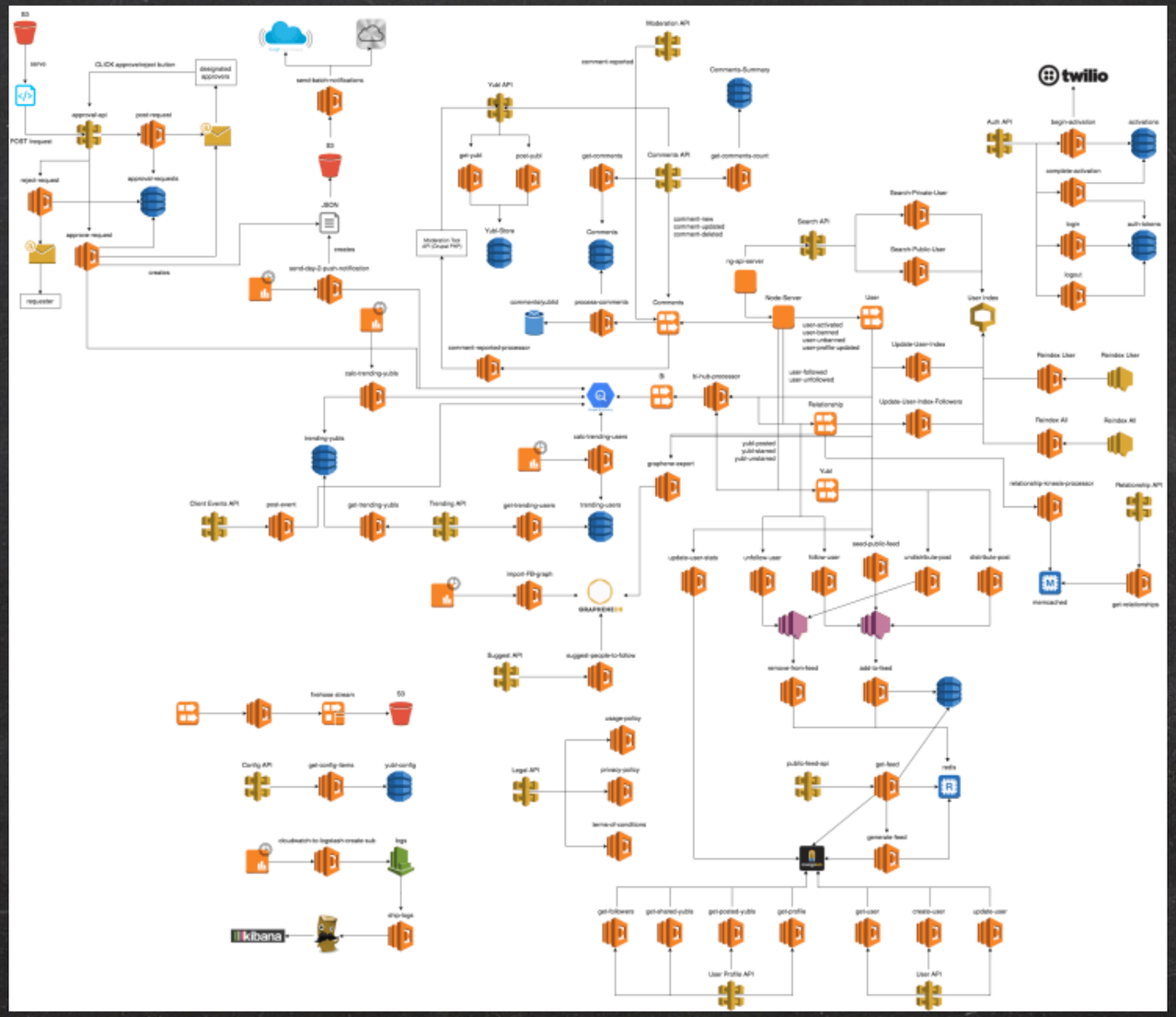
hi, my name is Yan.

hi, my name is Yan.

Space Ape™

hi, my name is Yan.

AWS user since 2009



http://bit.ly/yubl-serverless

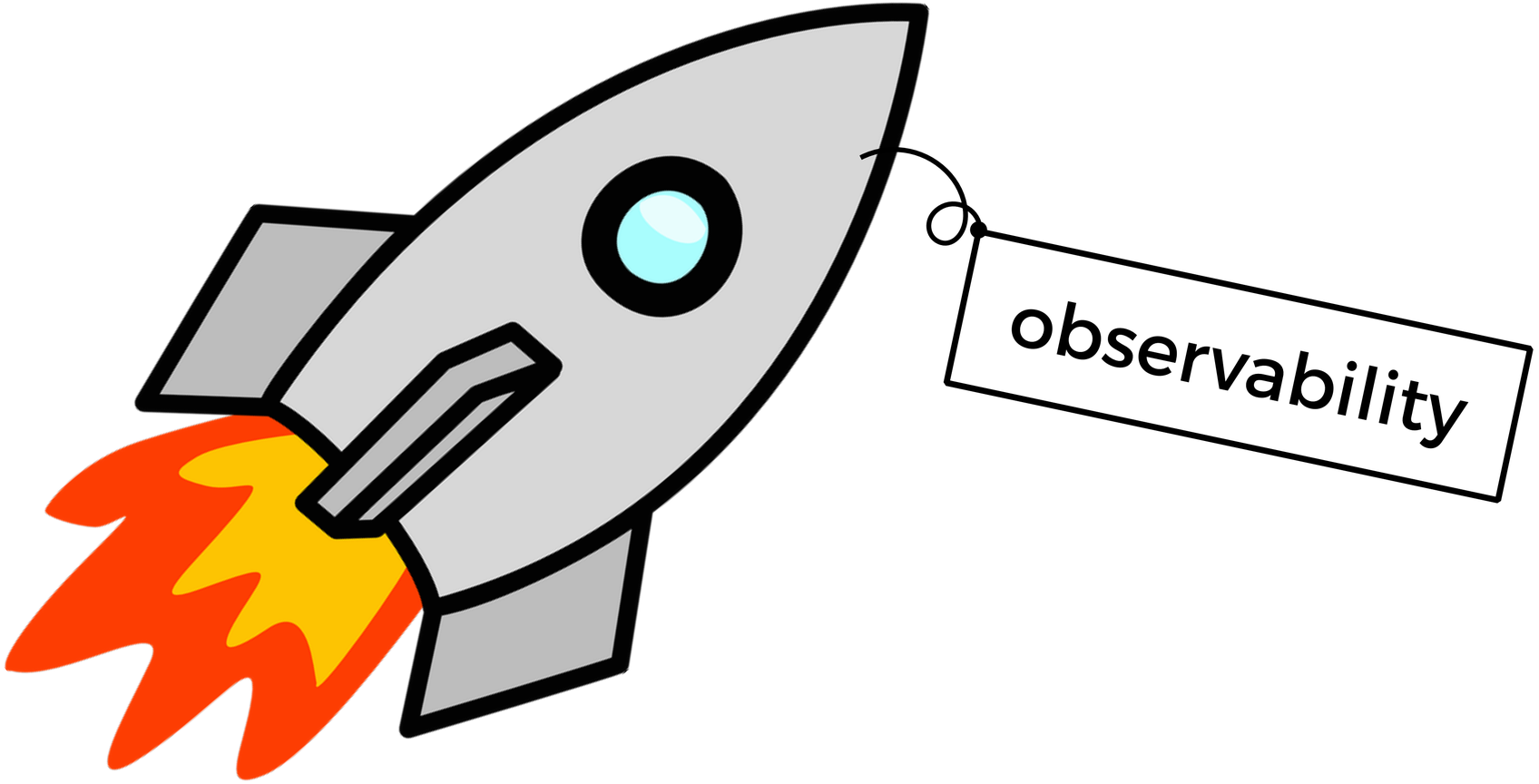
Serverless Applications Lens

AWS Well-Architected Framework

November 2017

http://bit.ly/2Cdsai5

🎉 2017 🎉



Cindy Sridharan
@copyconstruct on Twitter
Sep 5, 2017 · 12 min read

Monitoring and Observability

During lunch with a few friends in late July, the topic of Observability came up. I have a talk coming up at Velocity in less than a month called ***Monitoring in the time of Cloud Native***, so I've been speaking with friends about how they approach monitoring where they work. During this conversation, one of my friends mentioned:

He was only half joking. I've heard several variations of this zinger, some of them being:

http://bit.ly/2EXQZBj

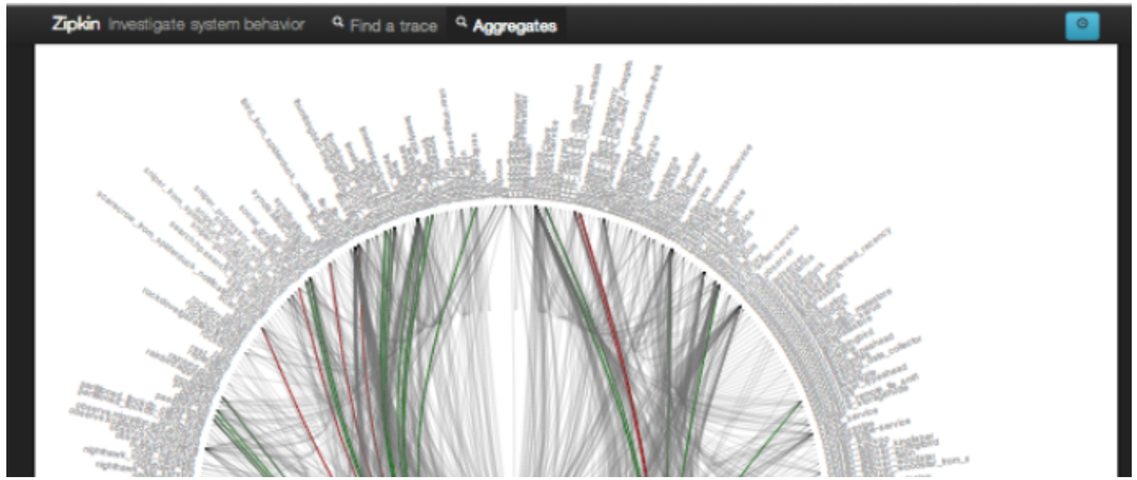
Observability at Twitter

By @gphat

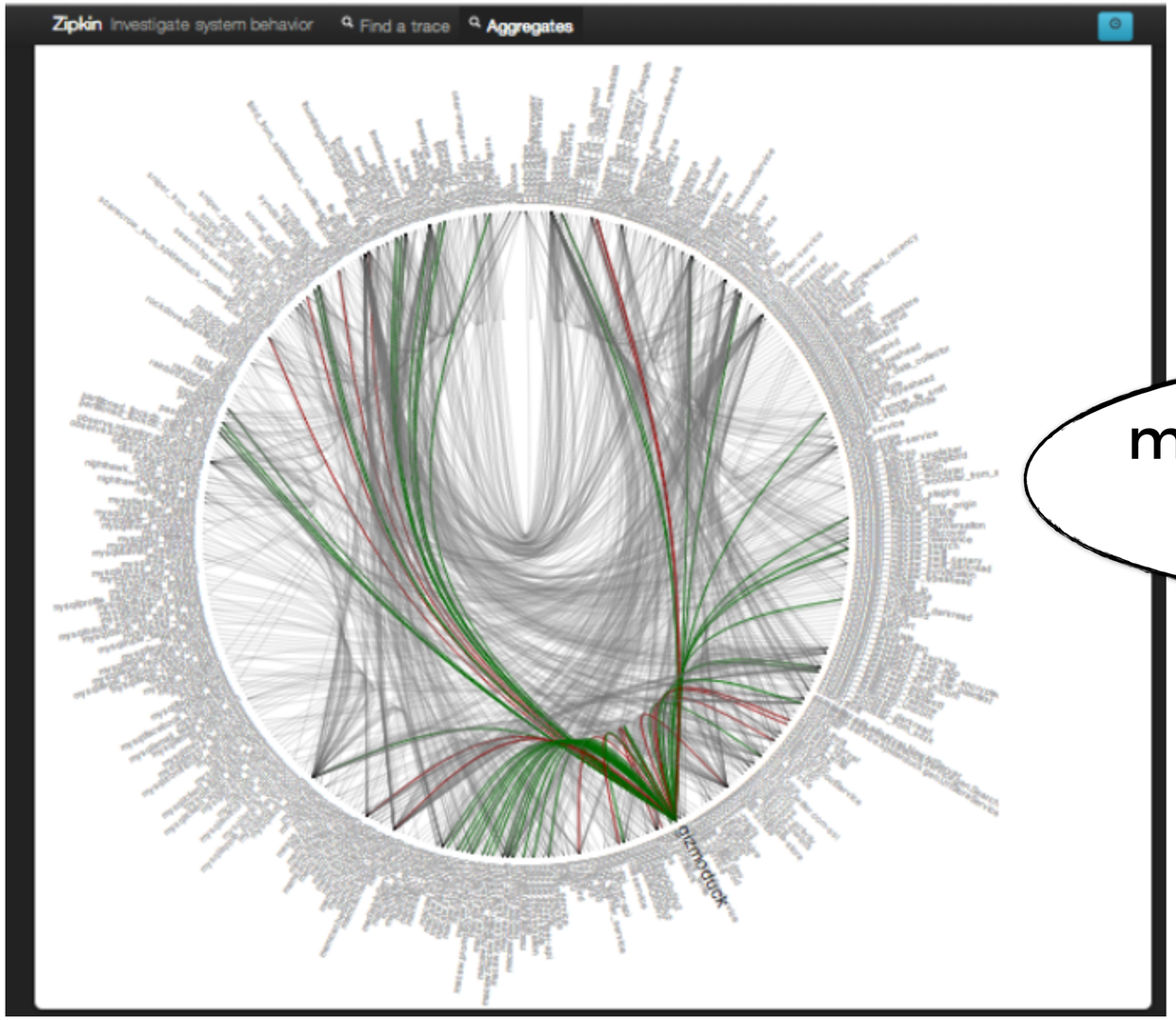
Monday, 9 September 2013

As Twitter has moved from a monolithic to a distributed architecture, our scalability has increased dramatically.

Because of this, the overall complexity of systems and their interactions has also escalated. This decomposition has led to Twitter managing hundreds of services across our datacenters. Visibility into the health and performance of our diverse service topology has become an important driver for quickly determining the root cause of issues, as well as increasing Twitter's overall reliability and efficiency. Debugging a complex program might involve instrumenting certain code paths or running special utilities; similarly Twitter needs a way to perform this sort of debugging for its distributed systems.



http://bit.ly/2EXKEFZ


Zipkin
Investigate system behavior
Find a trace
Aggregates

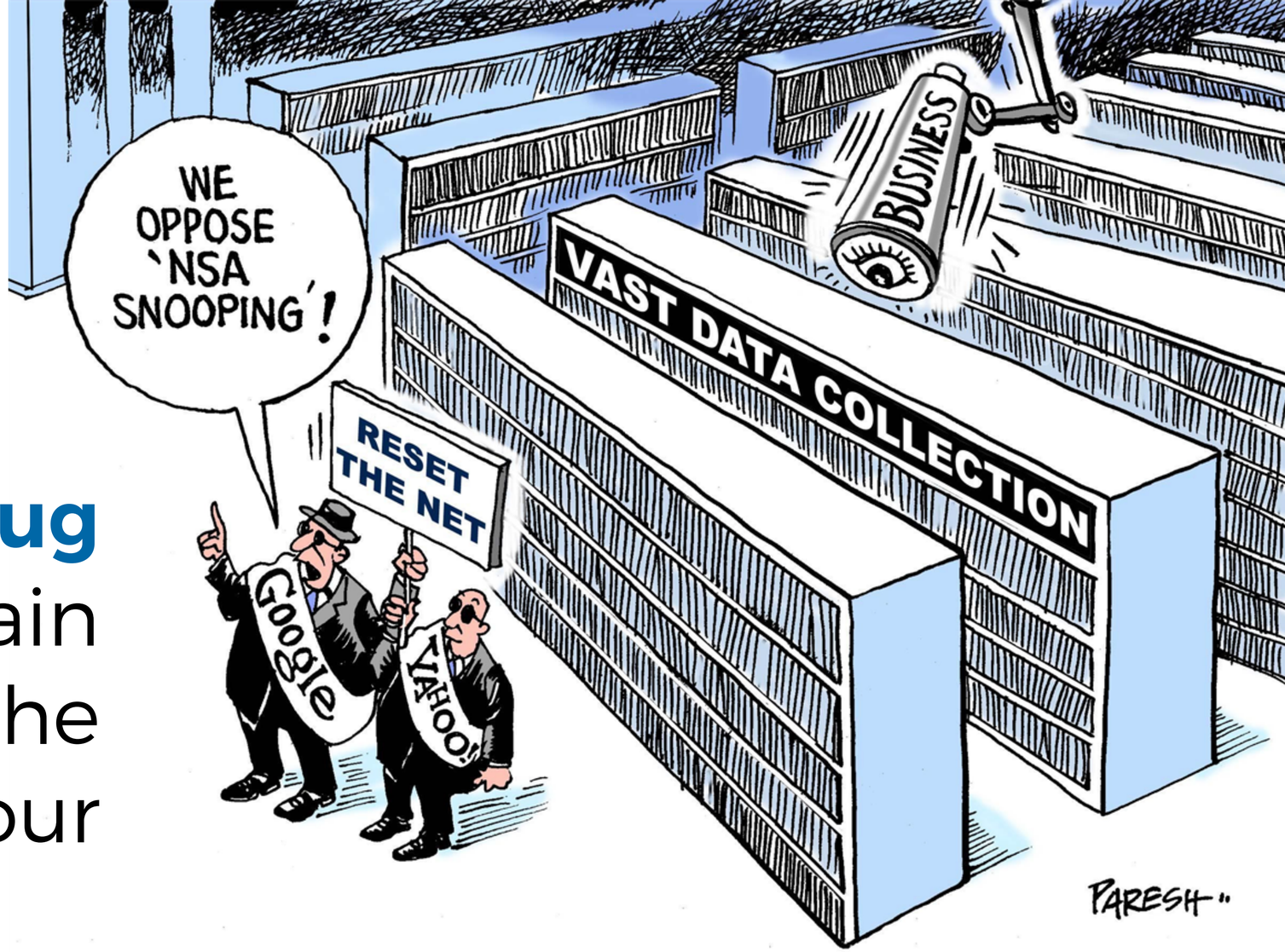

mm... I wonder what's going on here...

what is **observability**?
how is it different from monitoring?

Monitoring

watching out for
known failure modes
in the system,
e.g. network I/O, CPU,
memory usage, ...

being able to **debug** the system, and gain **insights** into the system's behaviour



Observability

> *However, I would argue that the health of the system no longer matters. We've entered an era where what matters is the health of each individual event, or each individual user's experience, or each shopping cart's experience (or other high cardinality dimensions).* ***With distributed systems you don't care about the health of the system, you care about the health of the event or the slice.***

- Charity Majors http://bit.ly/2E2QngU

> *However, I would argue that the health of the system no longer matters. We've entered an era where what matters is the health of each individual event, or each individual user's experience, or each shopping cart's experience (or other high cardinality dimensions).* ***With distributed systems you don't care about the health of the system, you care about the health of the event or the slice****.*

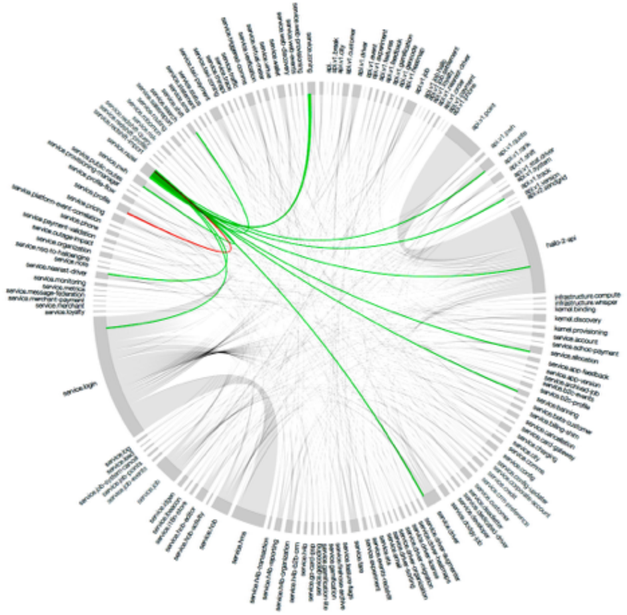
- Charity Majors http://bit.ly/2E2QngU

> *These are the four pillars of the Observability Engineering team's charter:*
>
> - *Monitoring*
> - *Alerting/visualization*
> - *Distributed systems tracing infrastructure*
> - *Log aggregation/analytics*

- Observability Engineering at Twitter http://bit.ly/2DnjyuW

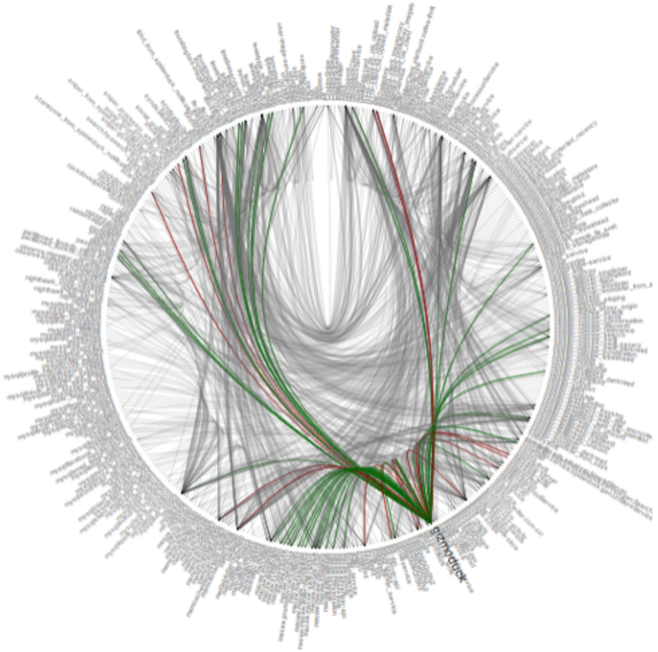
Observability is useful even outside of incidents and outages

450+ microservices



500+ microservices

500+ microservices

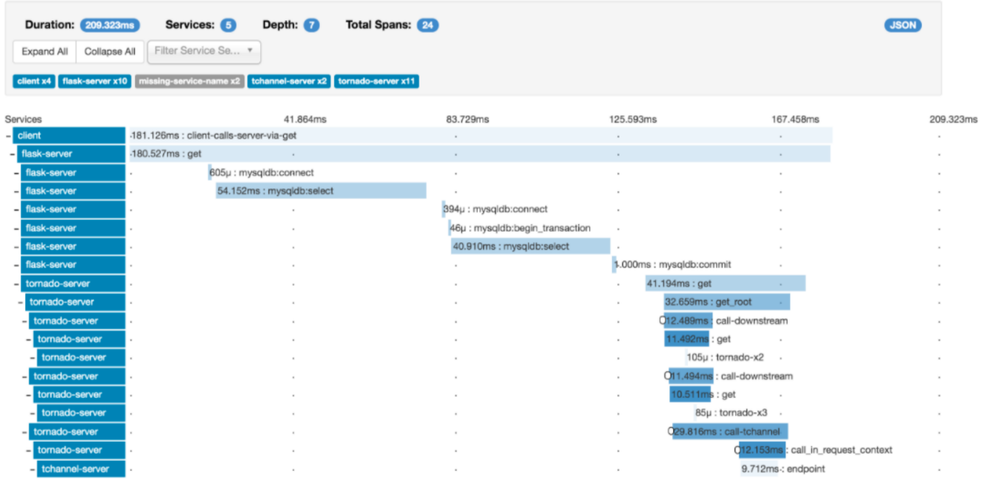


HAILO

NETFLIX

microservices death stars circa 2015

Zipkin

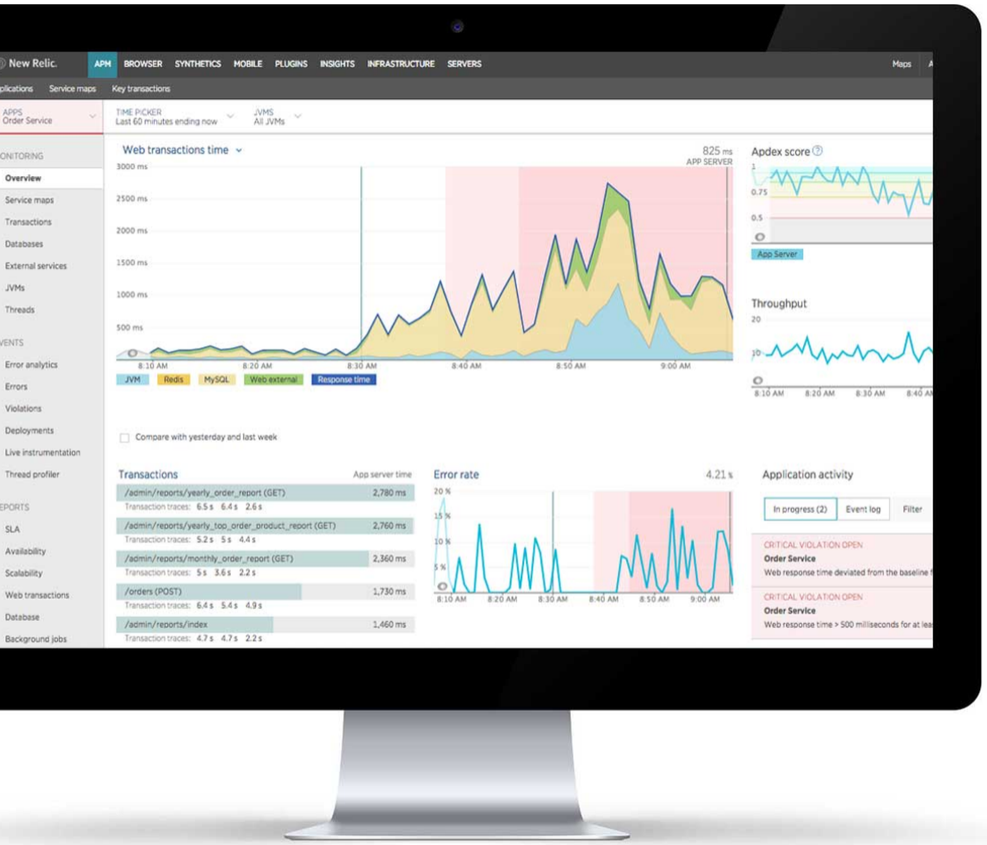


Zipkin is a distributed tracing system. It helps gather timing data needed to troubleshoot latency problems in microservice architectures. It manages both the collection and lookup of this data. Zipkin's design is based on the Google Dapper paper.

Applications are instrumented to report timing data to Zipkin. The Zipkin UI also presents a Dependency diagram showing how many traced requests went through each application. If you are troubleshooting latency problems or errors, you can filter or sort all traces based on the application, length of trace, annotation, or timestamp. Once you select a trace, you can see the percentage of the total trace time each span takes which allows you to identify the problem application.

Where to go next?

- To try out Zipkin, check out our Quickstart guide
- See if your platform has an existing instrumentation library
- Join the Zipkin Gitter chat channel
- The source code is on GitHub as openzipkin/zipkin
- Issues are also tracked on GitHub



Application performance magic.

Build, deploy, and maintain great software with New Relic APM.

Request a demo

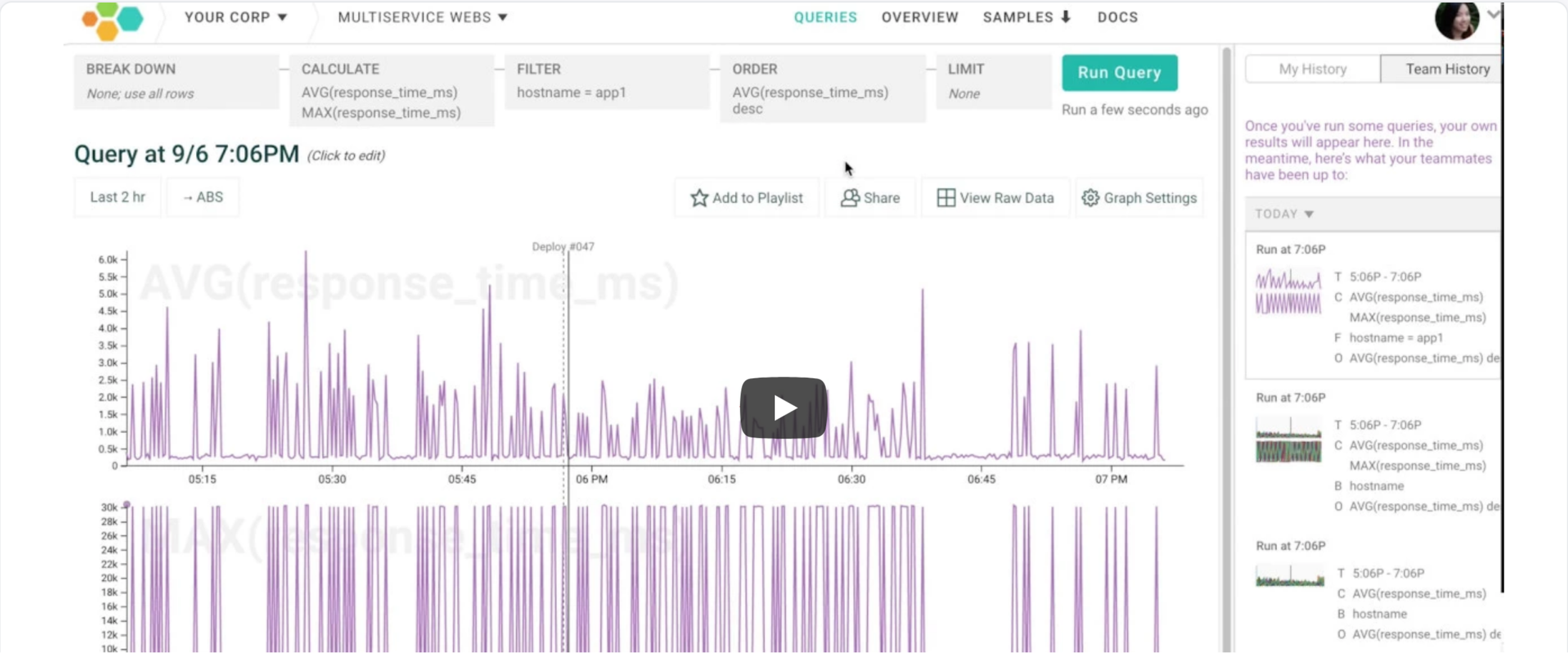
NEW Distributed Tracing

Distributed Tracing support will let you track a user request through

It's easy to stay up-to-date.

Your Email Address: *

Observability for a distributed world



TransAlta Teams With Splunk for Security and Operational Intelligence

Saves up to $1 million while reducing investigation times from days to minutes

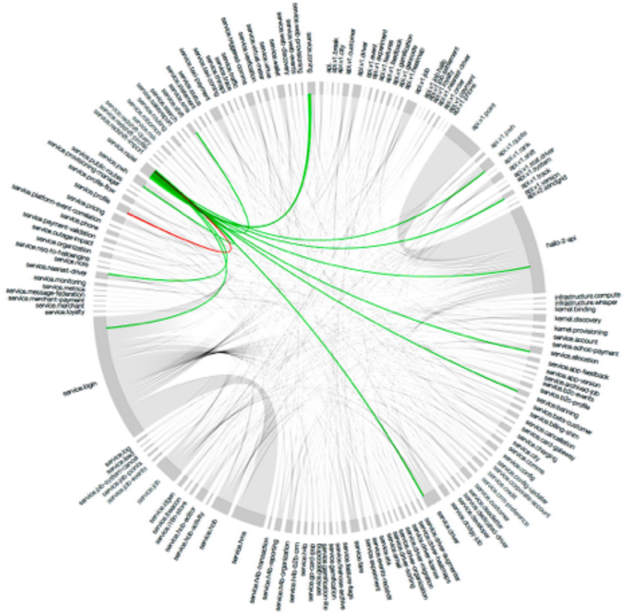
Read the Case Study

Gartner names Splunk a SIEM Magic Quadrant leader for the fifth year running!

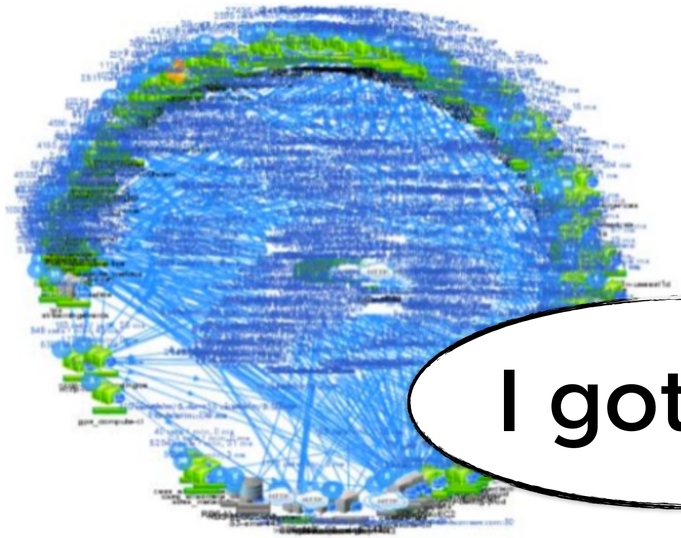
Read the Report

Turn Machine Data Into Answers

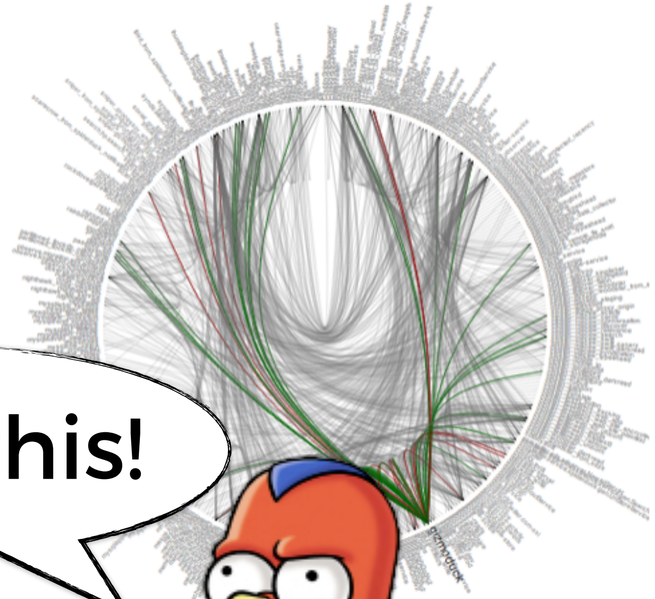
450+ microservices



500+ microservices



500+ microservices



HAIL O

NETFLIX

microservices death stars circa 201


Introducing AWS Lambda
An event-driven computing service for dynamic applications
amazon webservices


new
challenges

new challenges

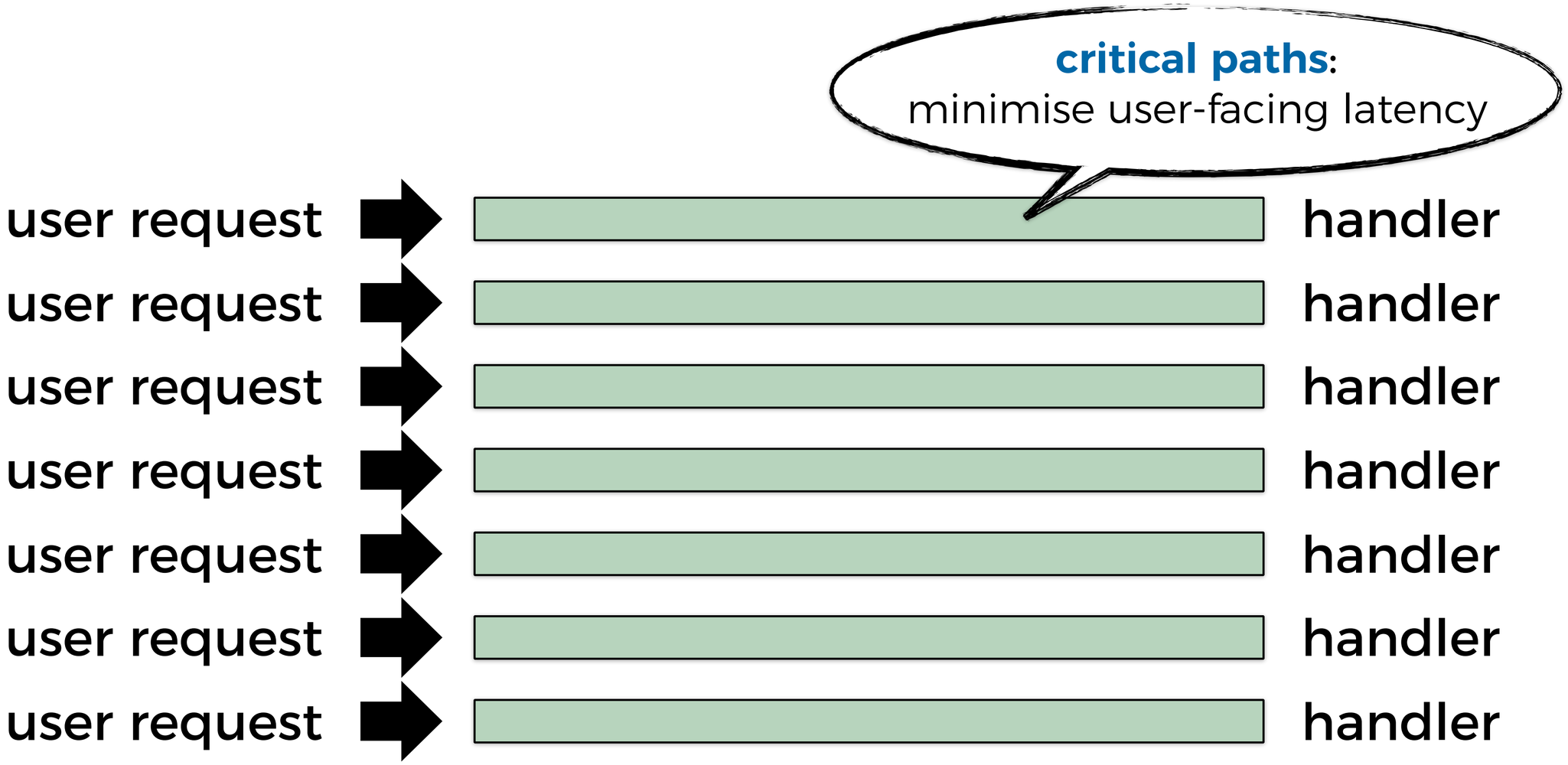
NO ACCESS
to underlying OS

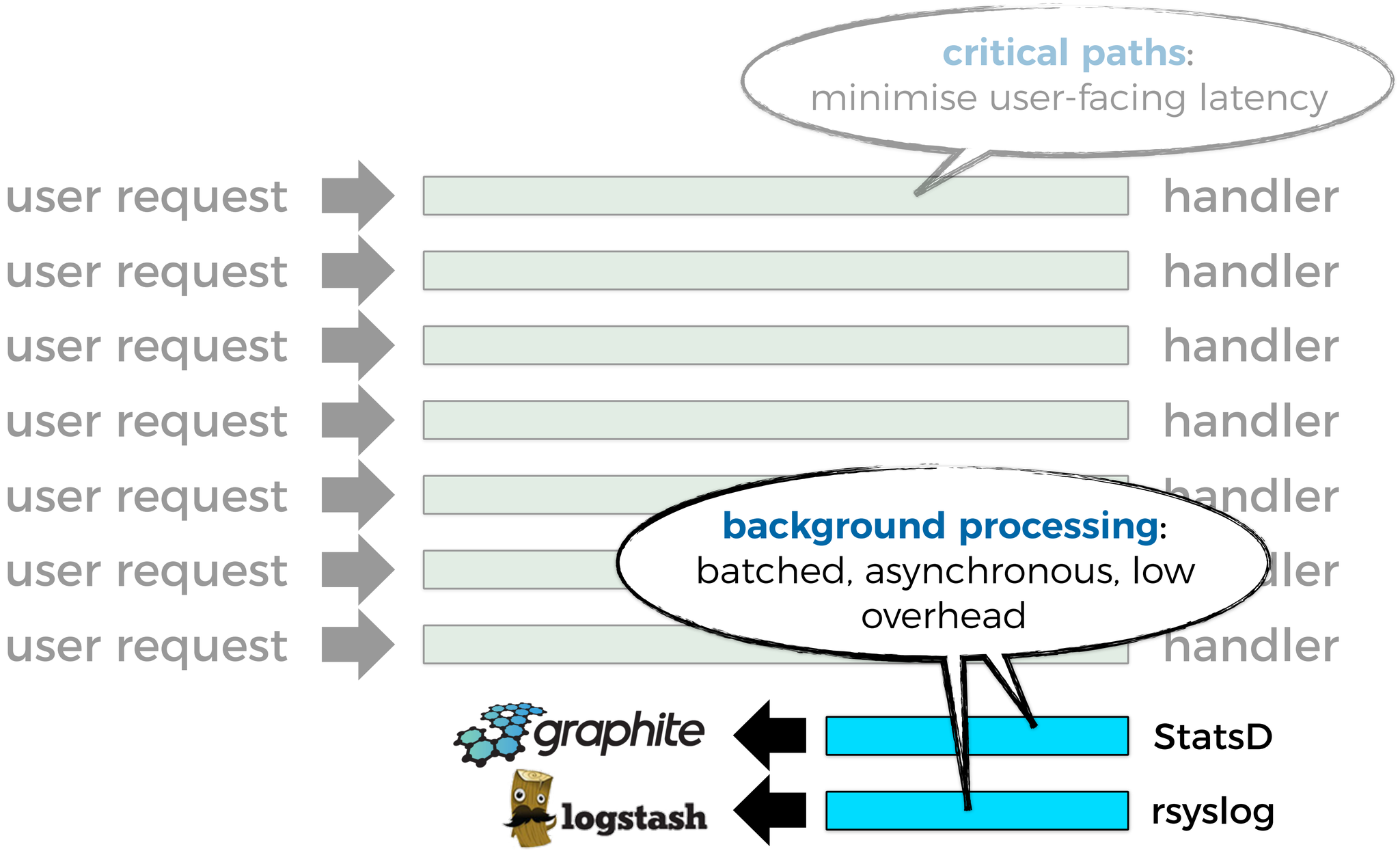
NOWHERE

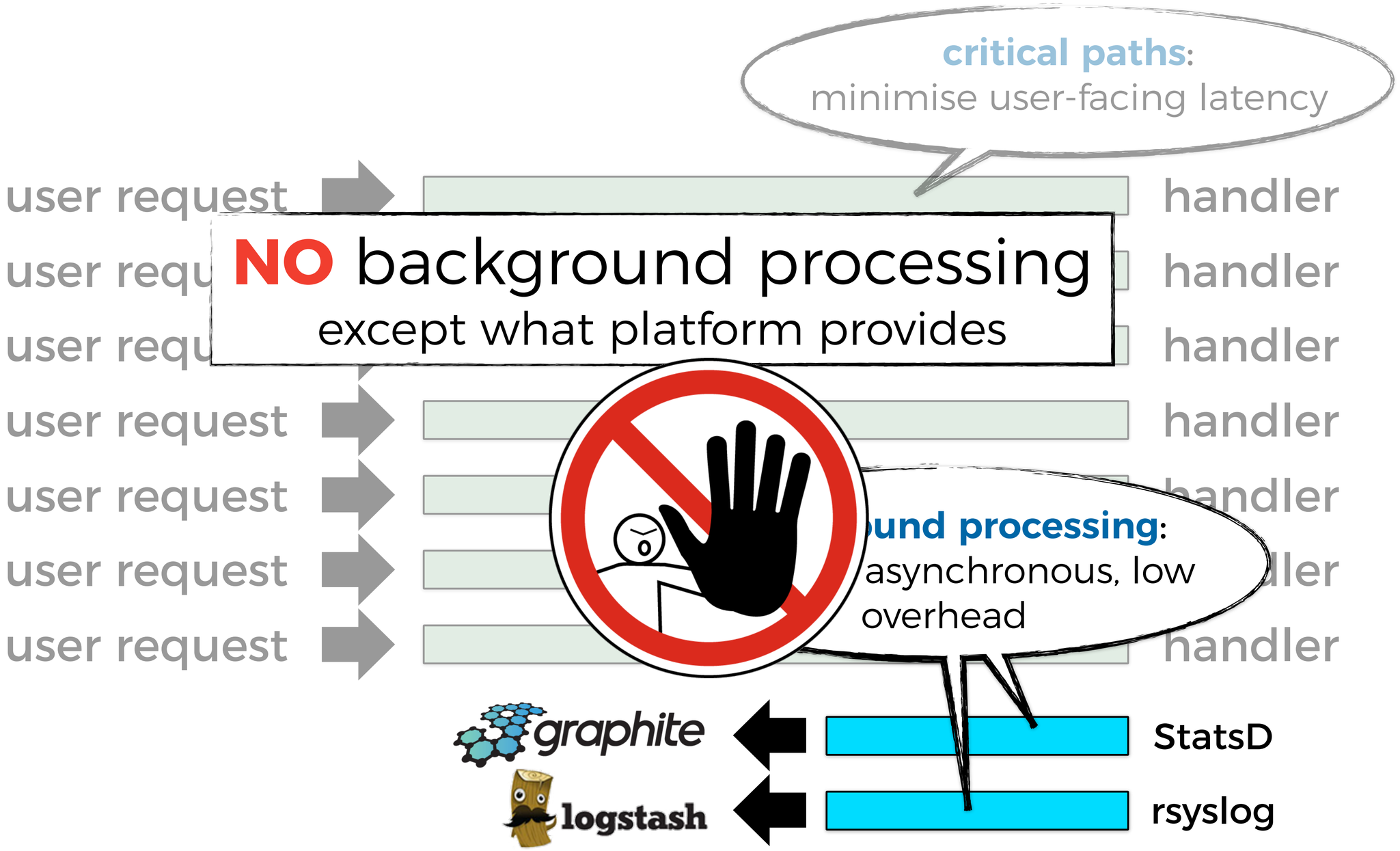
to install agents/daemons

new challenges

- **nowhere** to install agents/daemons

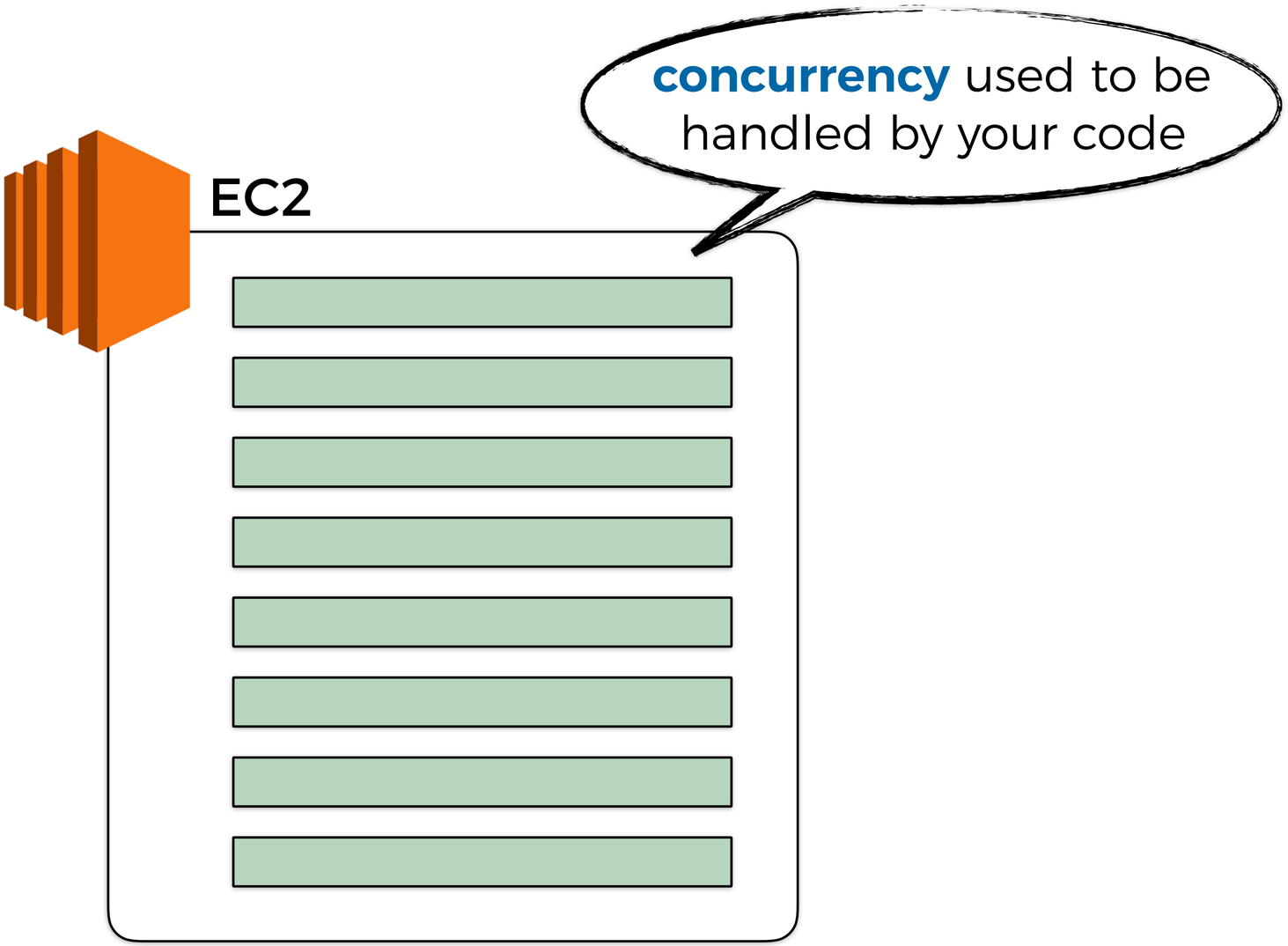

critical paths:
minimise user-facing latency
user request
handler
user request
handler
user request
handler
user request
handler
user request
handler
user request
handler
user request
handler

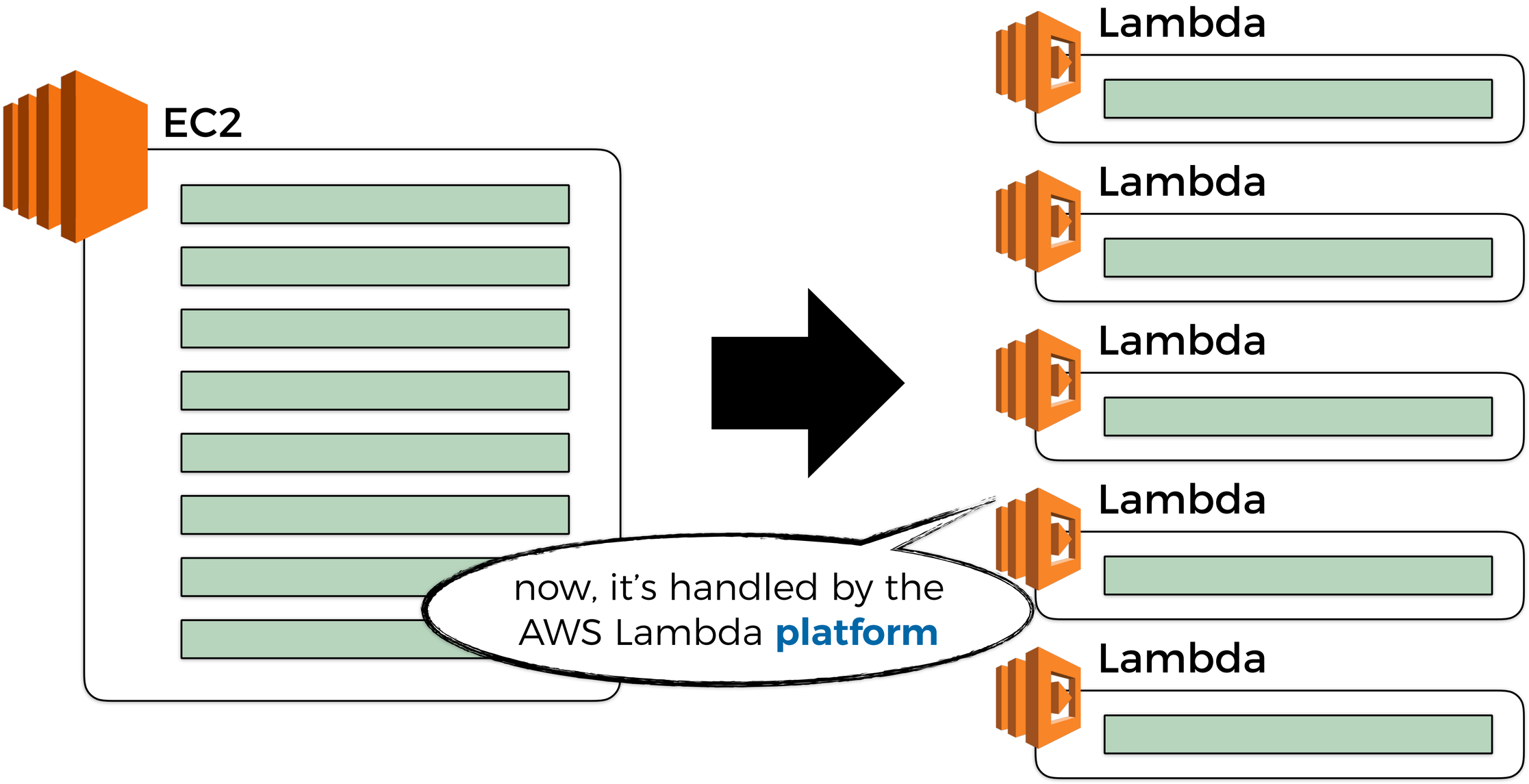

critical paths:
minimise user-facing latency
user request
handler
background processing:
batched, asynchronous, low overhead
graphite
logstash
StatsD
rsyslog

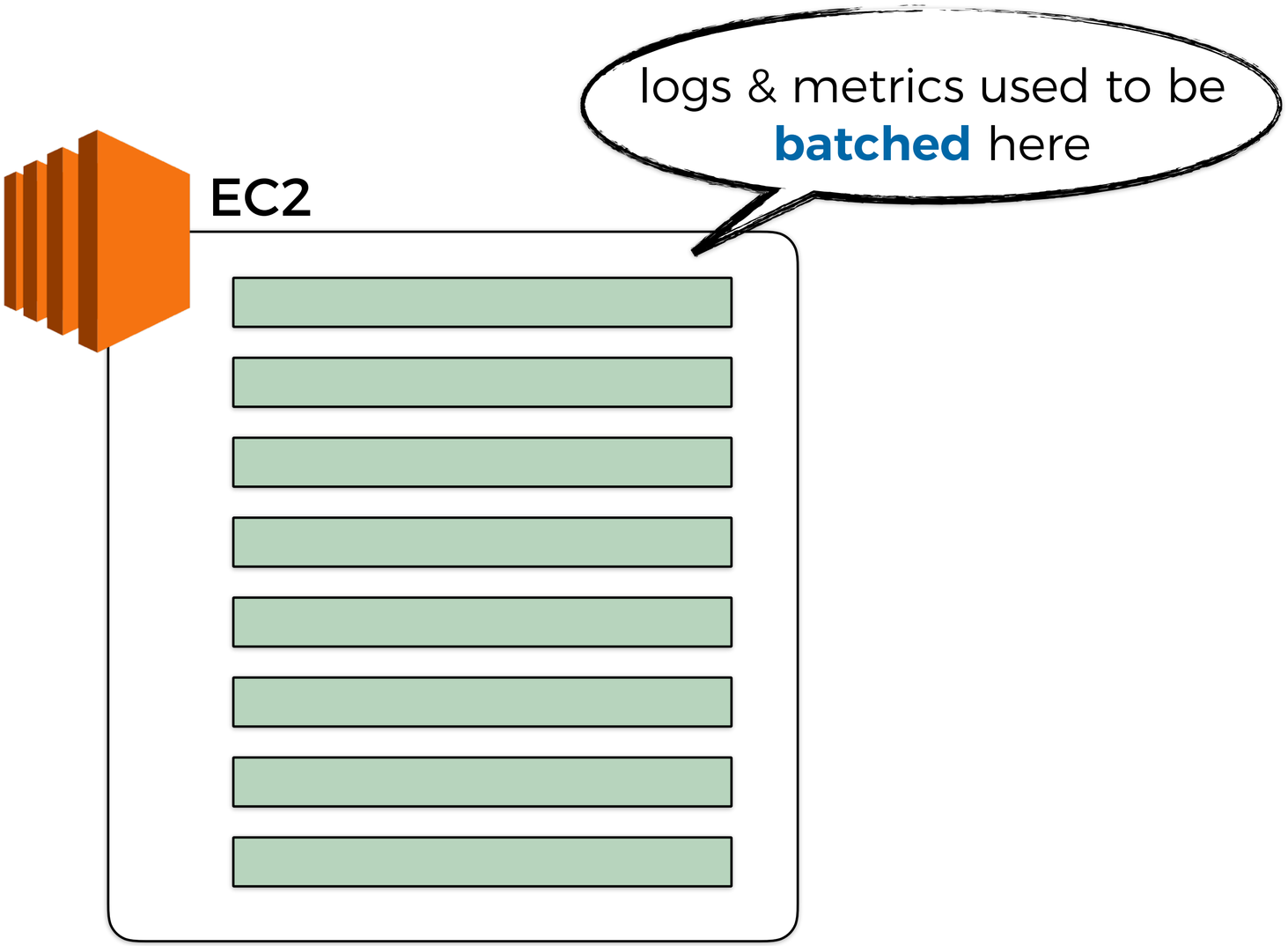

critical paths:
minimise user-facing latency
user request
handler
NO background processing
except what platform provides
ound processing:
asynchronous, low
overhead
graphite
logstash
StatsD
rsyslog

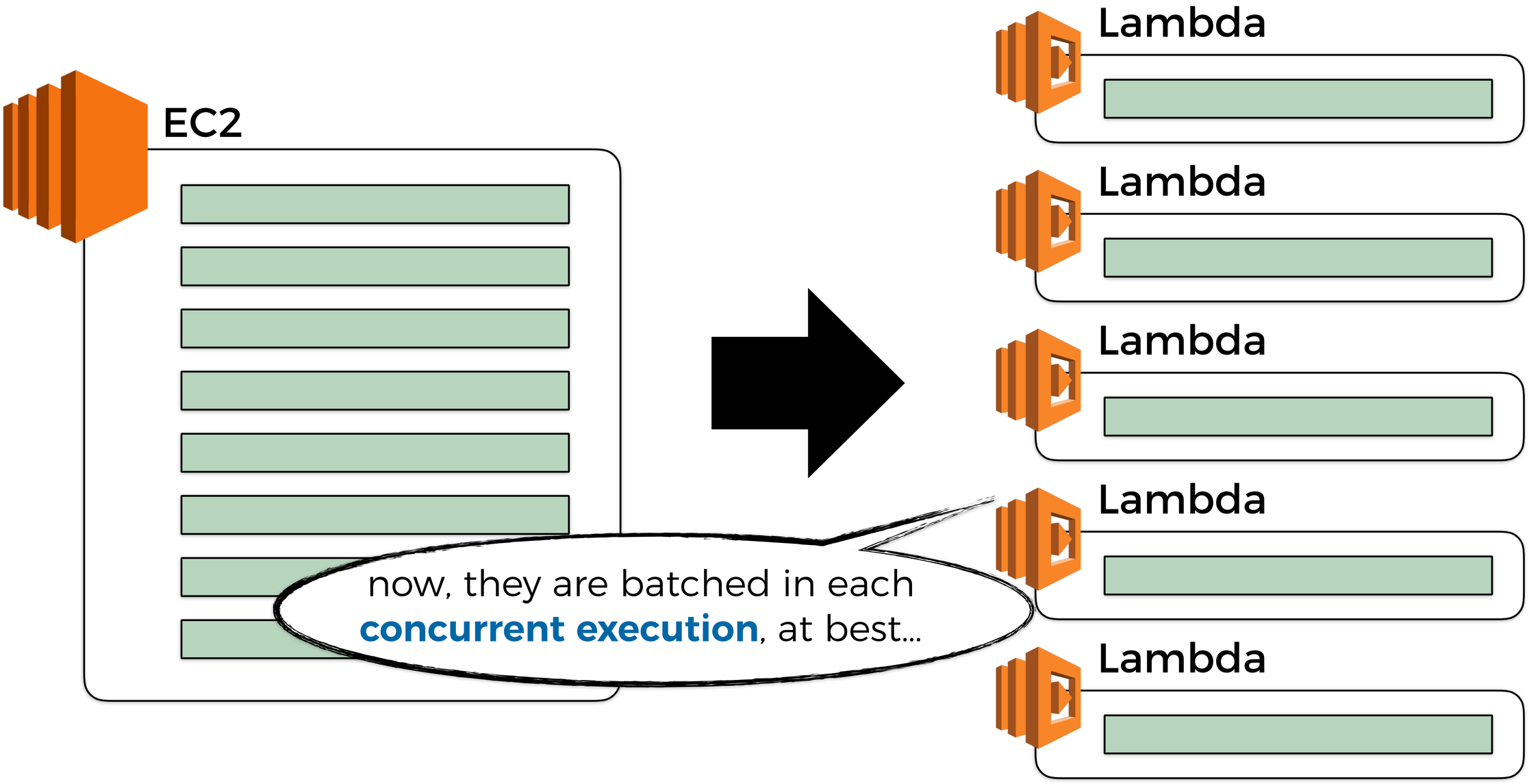
new challenges

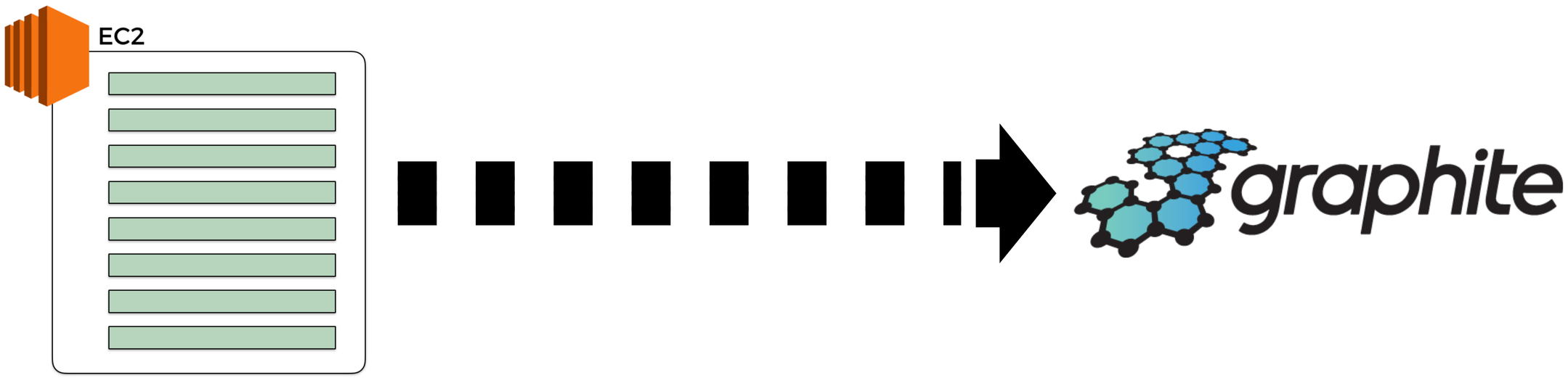
- **nowhere** to install agents/daemons
- **no** background processing

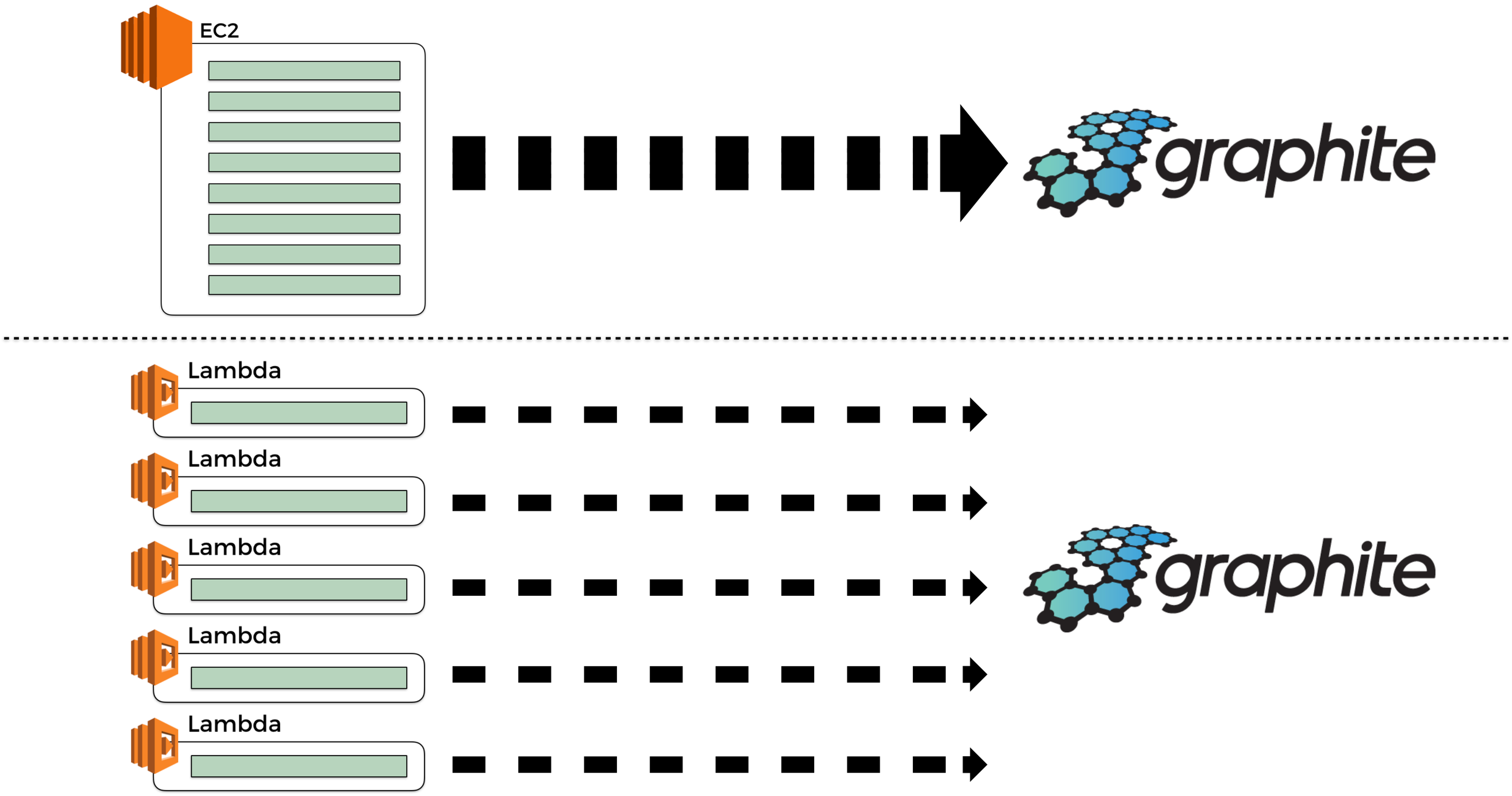

concurrency used to be handled by your code
EC2

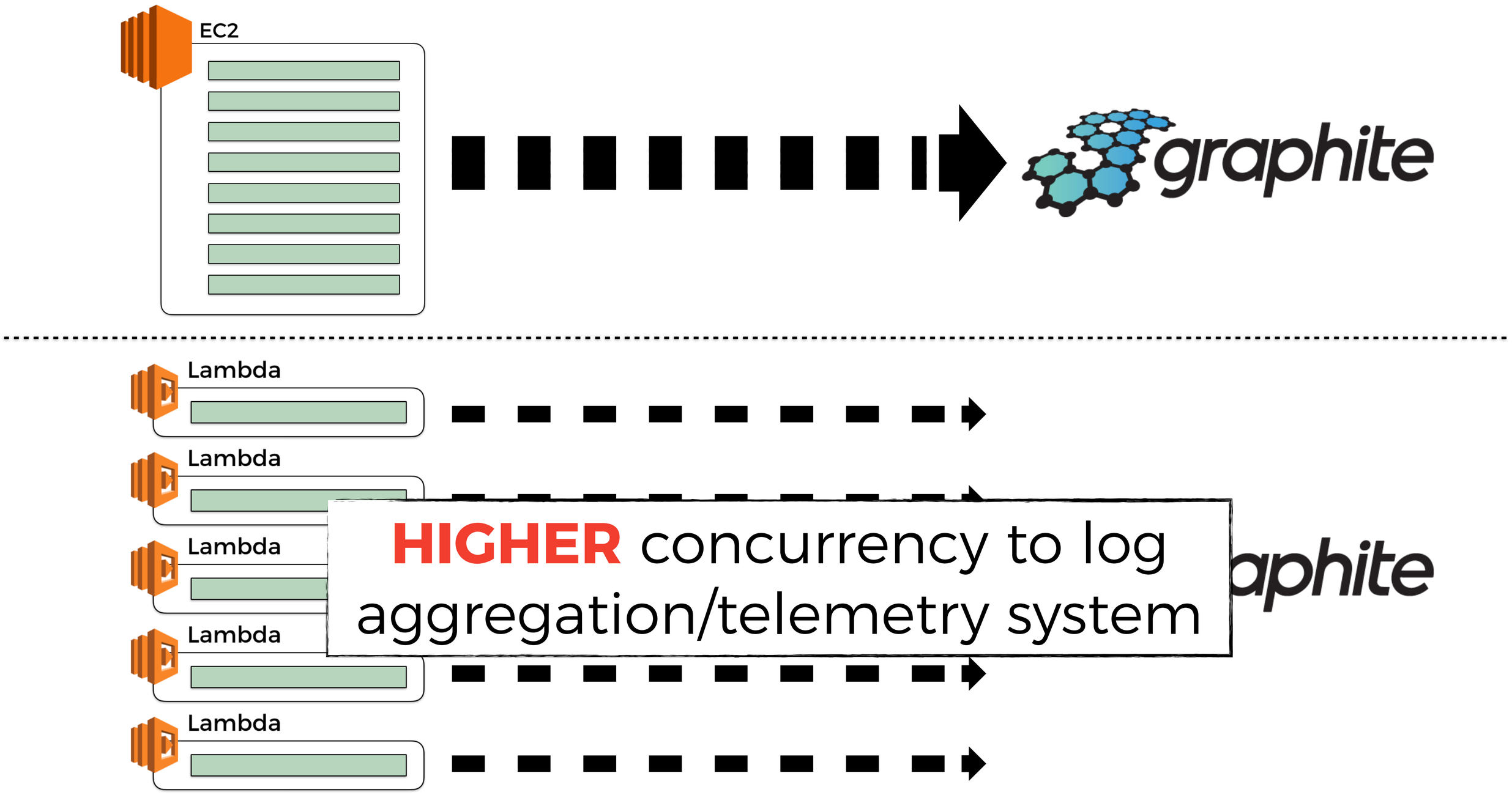

EC2
Lambda
Lambda
Lambda
Lambda
Lambda
now, it's handled by the
AWS Lambda **platform**


logs & metrics used to be
batched here
EC2


EC2
Lambda
Lambda
Lambda
Lambda
Lambda
now, they are batched in each
concurrent execution, at best...

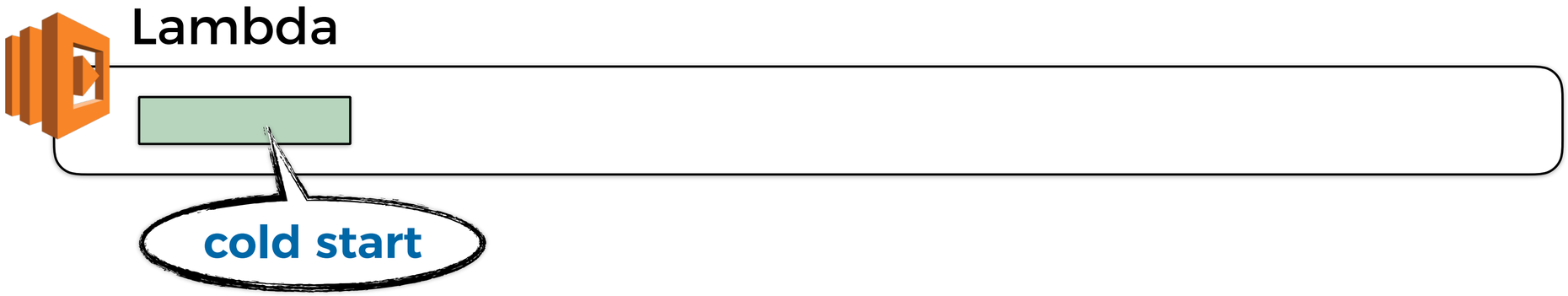

EC2
graphite

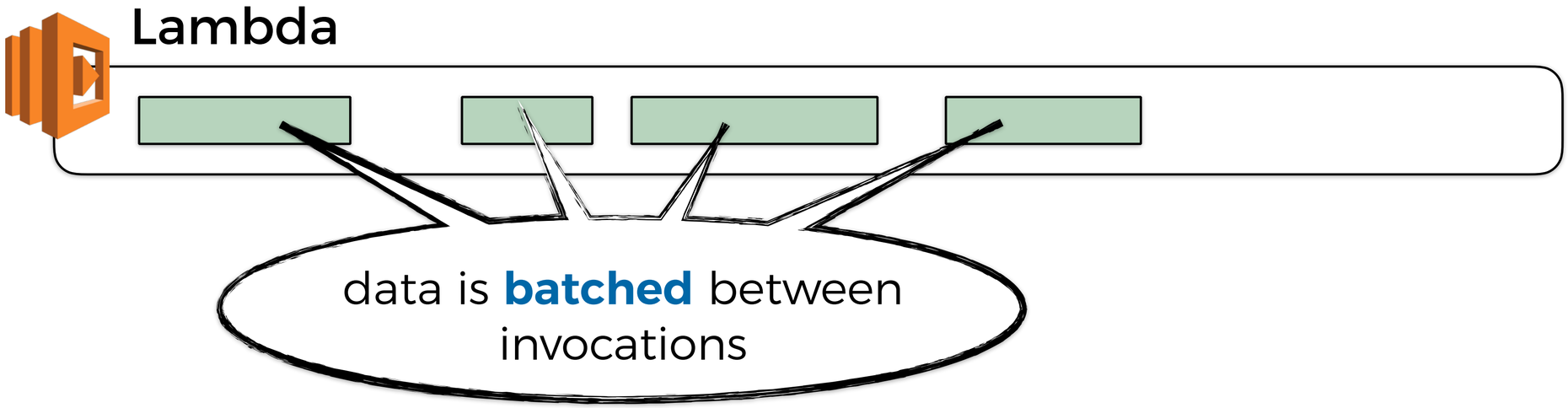

EC2
graphite
Lambda
Lambda
Lambda
Lambda
Lambda
graphite

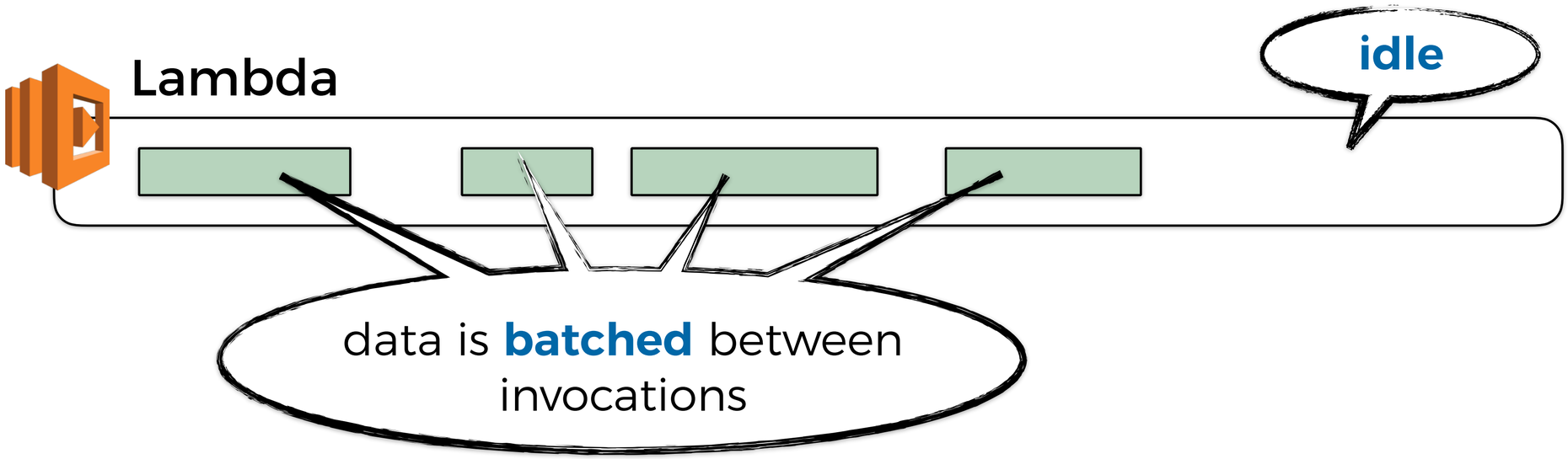

EC2
graphite
Lambda
Lambda
Lambda
Lambda
Lambda
HIGHER concurrency to log aggregation/telemetry system
aphite

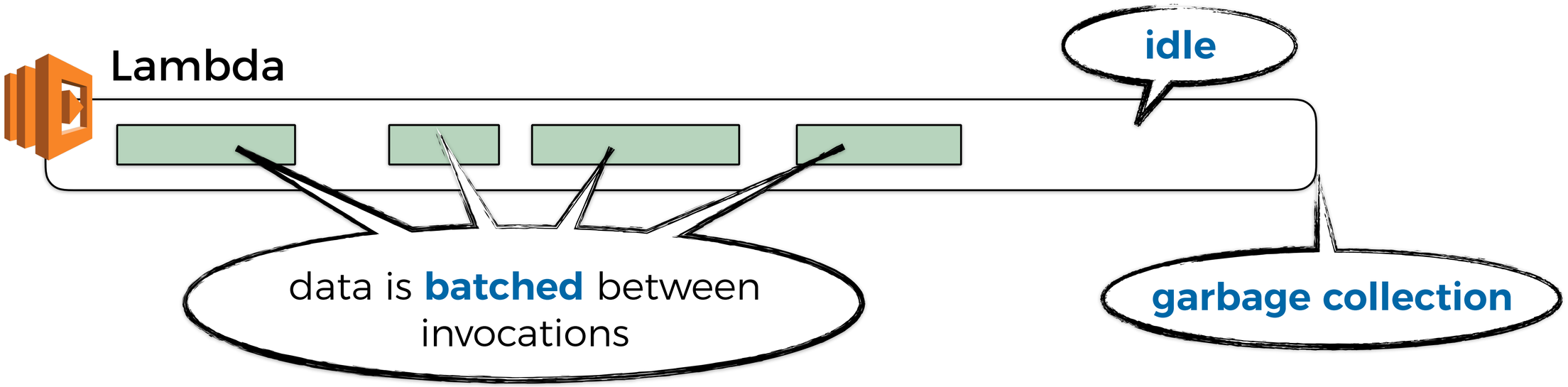
new challenges

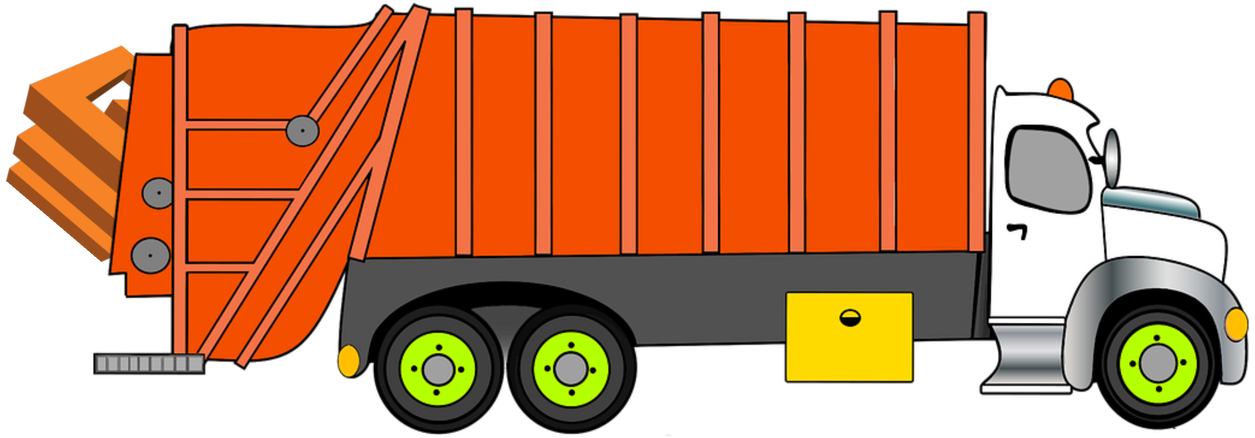
- **nowhere** to install agents/daemons
- **no** background processing
- **higher** concurrency to telemetry system

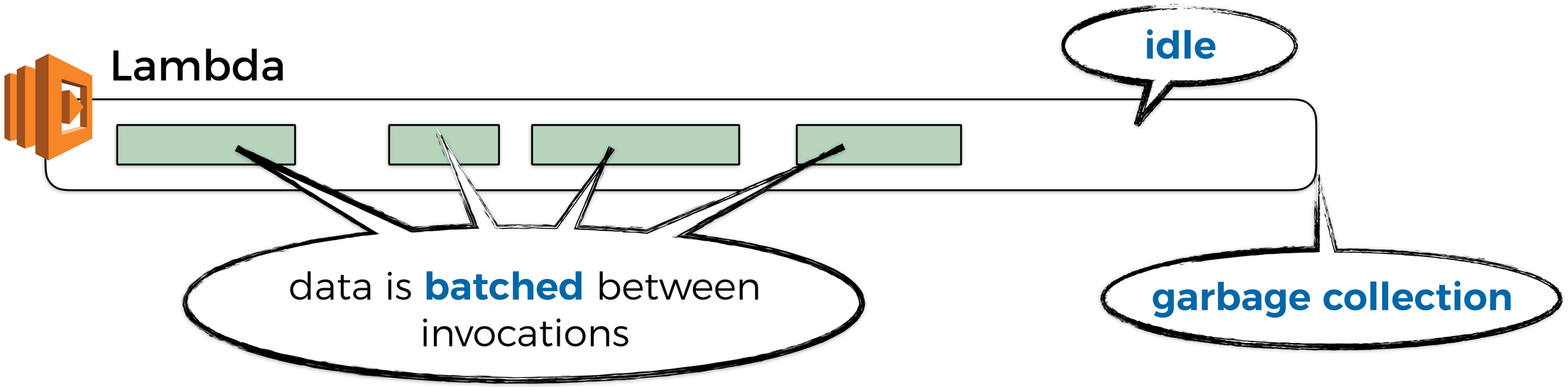

Lambda
cold start


Lambda
data is **batched** between invocations


Lambda
idle
data is **batched** between invocations


Lambda
idle
data is **batched** between invocations
garbage collection

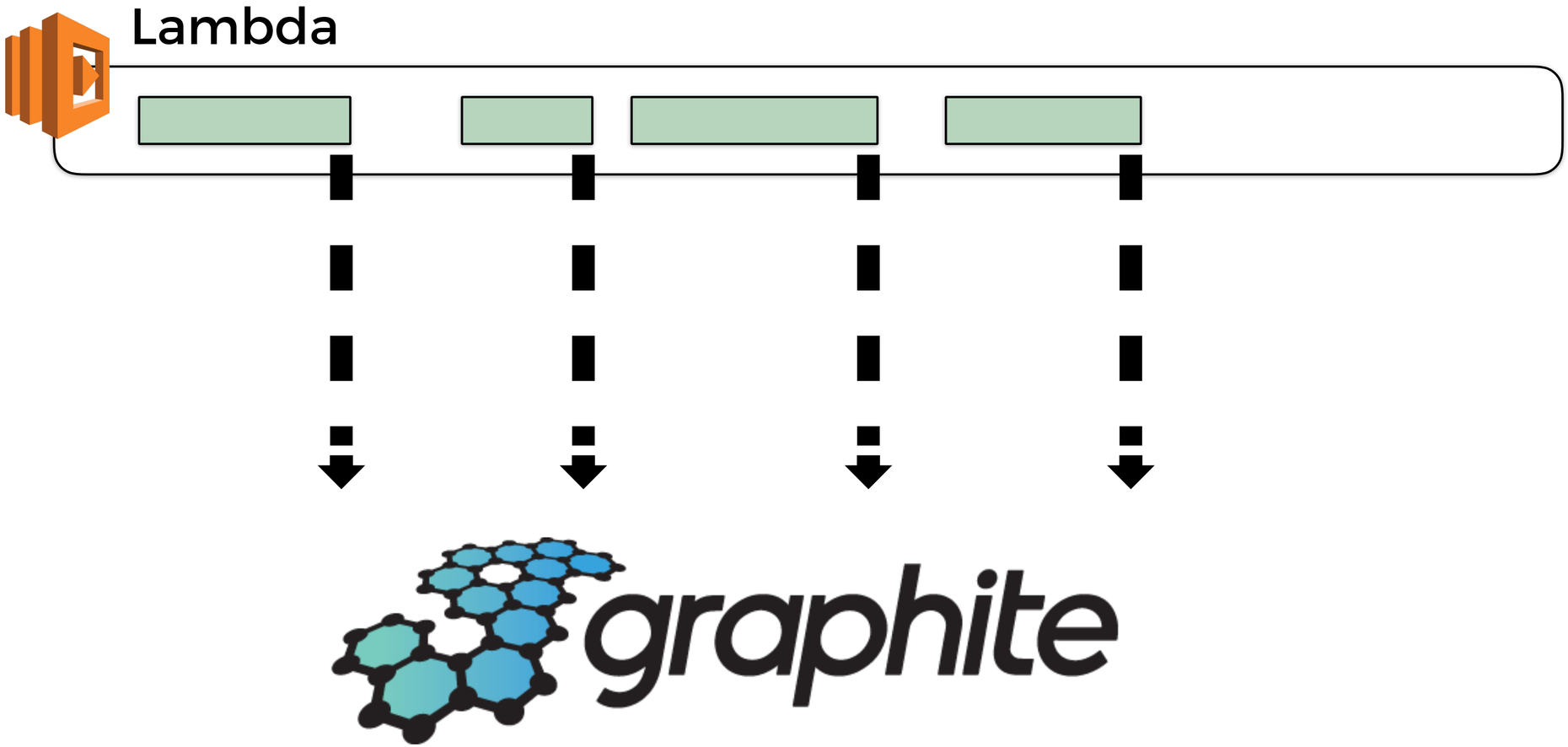


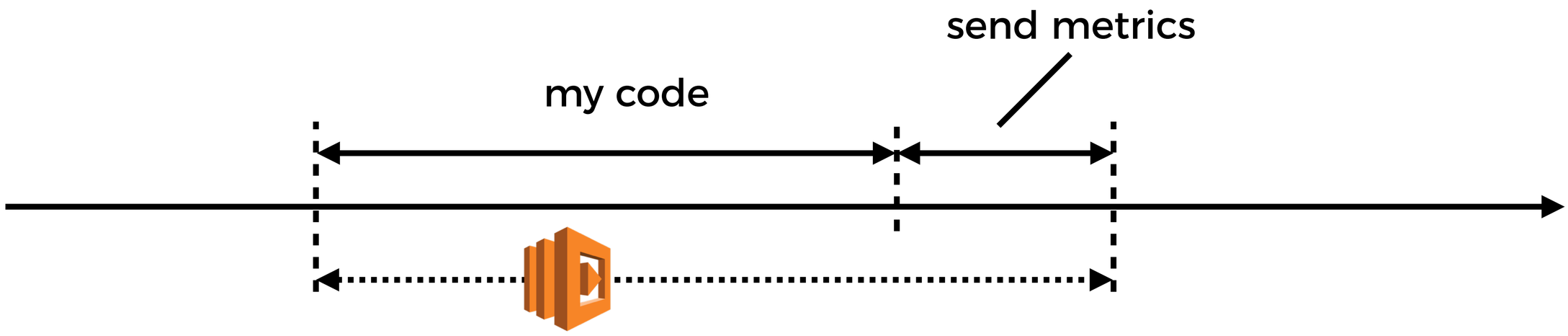


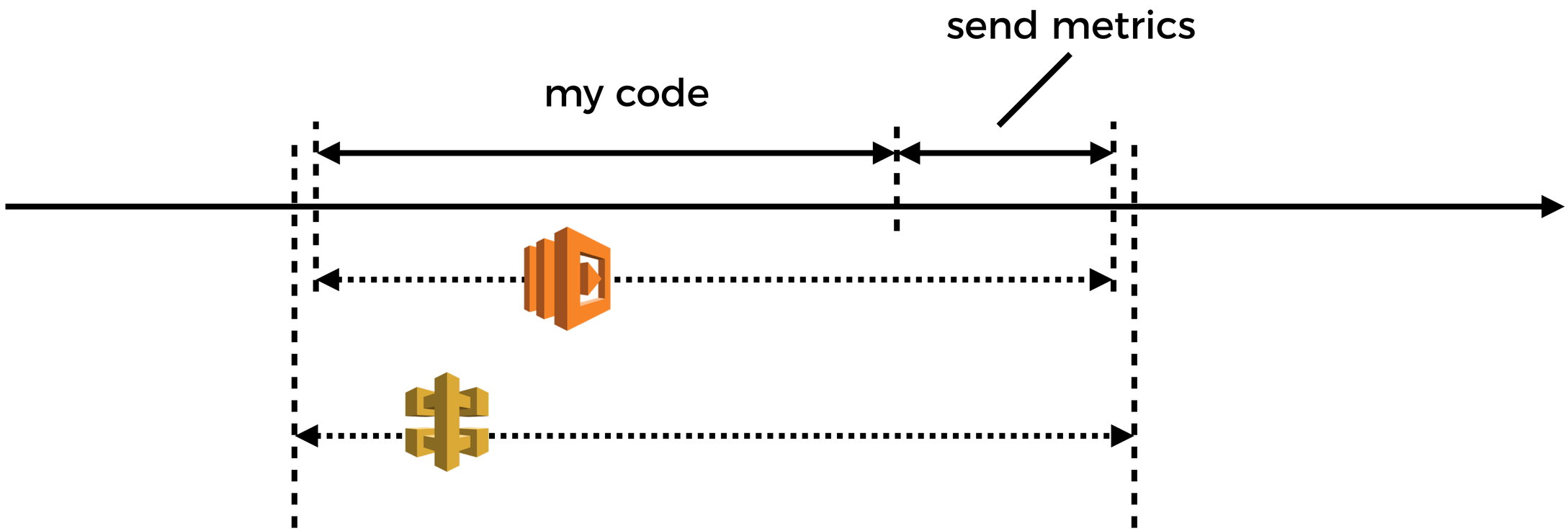
HIGH chance of data loss

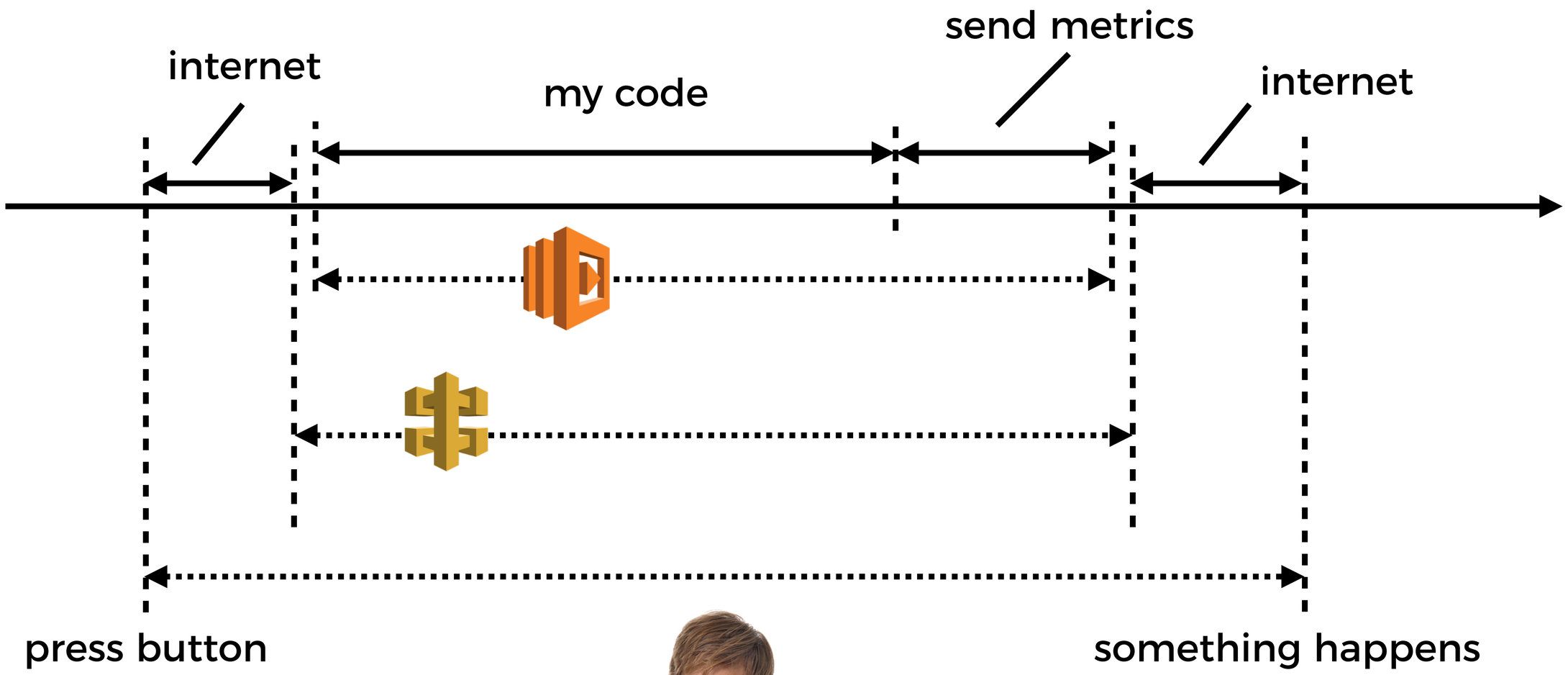
new challenges

- **nowhere** to install agents/daemons
- **no** background processing
- **higher** concurrency to telemetry system
- **high** chance of data loss (if batching)


Lambda
graphite


send metrics
my code


send metrics
my code


send metrics
internet
my code
internet
press button
something happens


A
HARD PLACE

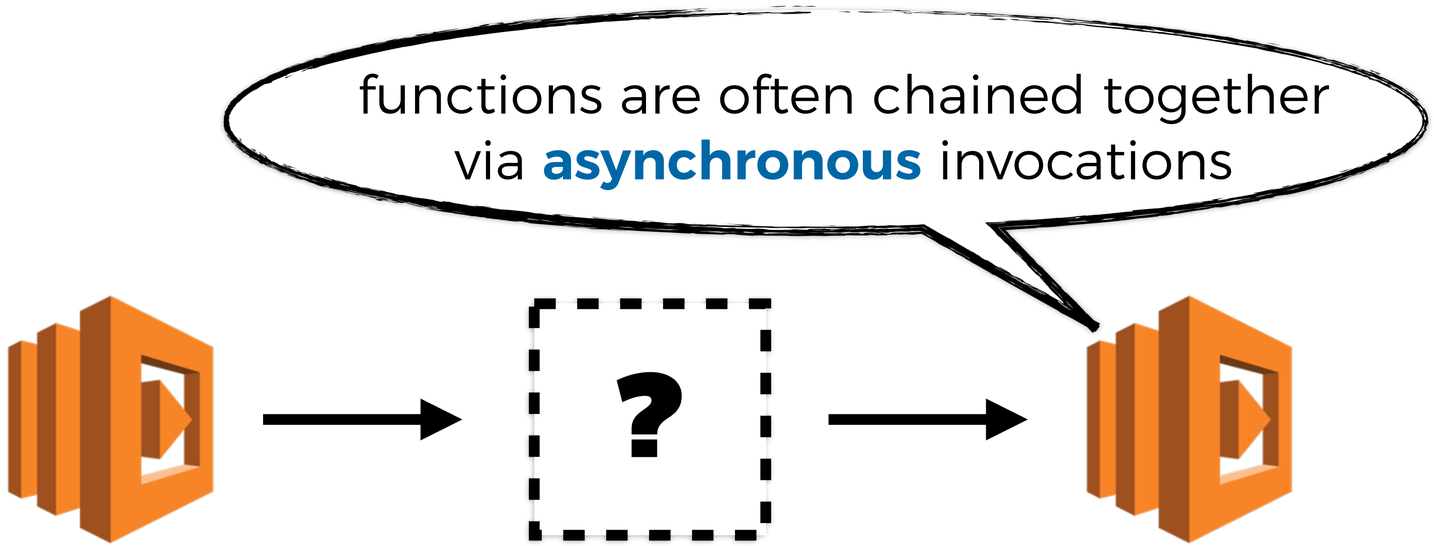
Randy Shoup Follow
Jan 5 · 8 min read

Image Source

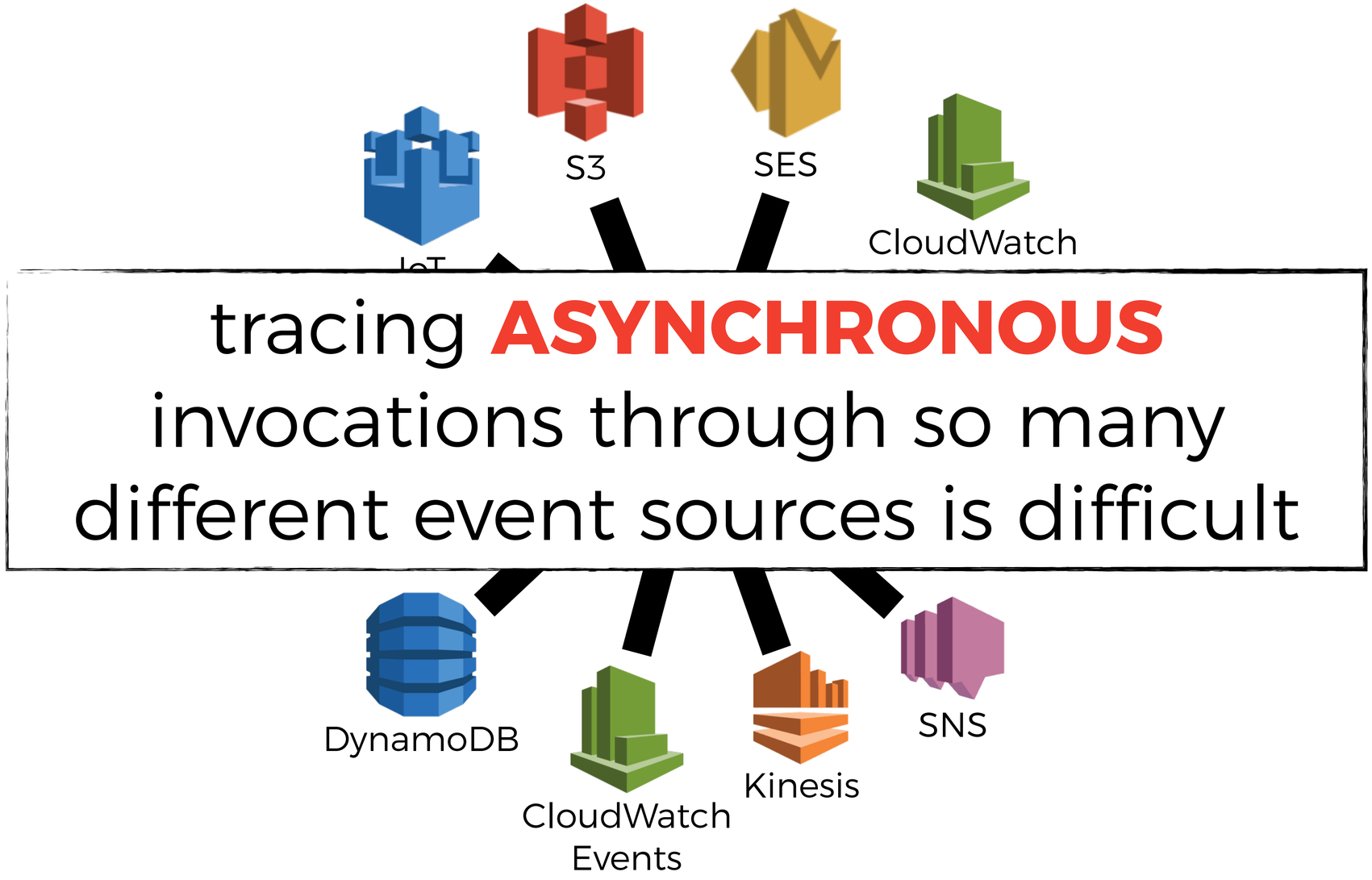
Events As First-Class Citizens

We've all heard of events and event-driven programming, but in my experience, events are not used nearly enough in our (software) lives. We often don't appreciate how powerful this tool can be in our toolbox, and consequently we don't take advantage of it when we really should.

http://bit.ly/2Dpidje


functions are often chained together
via **asynchronous** invocations
?


S3
SES
IoT
CloudWatch
Logs
?
DynamoDB
CloudWatch
Events
Kinesis
SNS


S3
SES
CloudWatch
tracing ASYNCHRONOUS invocations through so many different event sources is difficult
DynamoDB
CloudWatch Events
Kinesis
SNS

new challenges

- **nowhere** to install agents/daemons
- **no** background processing
- **higher** concurrency to telemetry system
- **high** chance of data loss (if batching)
- **asynchronous** invocations

the Present

> *These are the four pillars of the Observability Engineering team's charter:*
>
> - *Monitoring*
> - *Alerting/visualization*
> - *Distributed systems tracing infrastructure*
> - *Log aggregation/analytics*

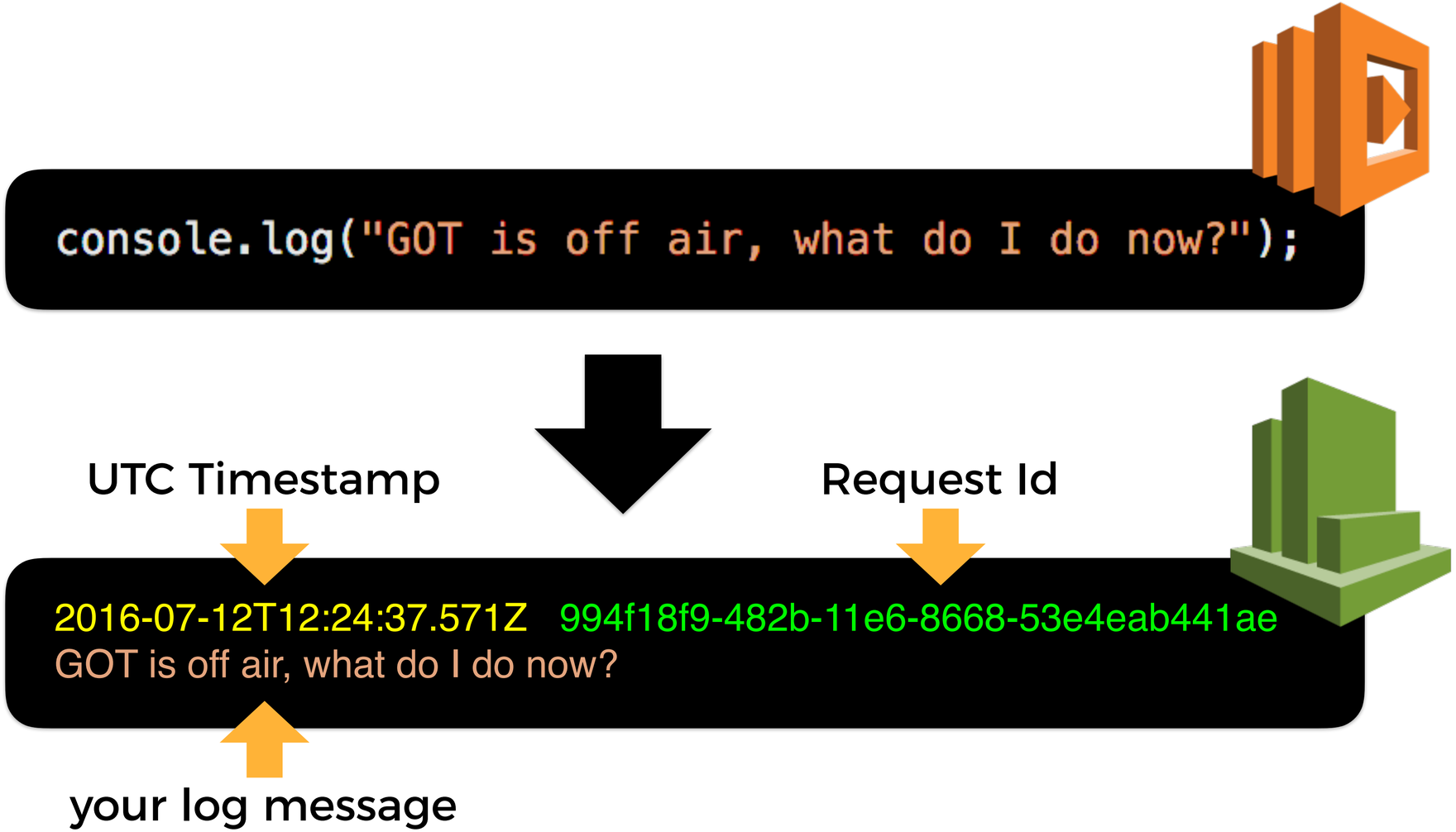
- Observability Engineering at Twitter http://bit.ly/2DnjyuW

aws

```
console.log("GOT is off air, what do I do now?");
```

```
console.log("GOT is off air, what do I do now?");
```

```
2016-07-12T12:24:37.571Z   994f18f9-482b-11e6-8668-53e4eab441ae
GOT is off air, what do I do now?
```


console.log("GOT is off air, what do I do now?");
UTC Timestamp
Request Id
2016-07-12T12:24:37.571Z 994f18f9-482b-11e6-8668-53e4eab441ae
GOT is off air, what do I do now?
your log message

CloudWatch > Log Groups > Streams for /aws/lambda/big-mouth-dev-get-index

Search Log Group | Create Log Stream | Delete Log Stream

Filter: Log Stream Name Prefix ×

Log Streams 1-50

Log Streams	Last Event Time
2018/01/13/[$LATEST]cd9779a0b20f44db8a661ca156dff8a1	2018-01-13 14:23 UTC
2018/01/13/[$LATEST]cb65dd255c9a426a8516ae38b06cd0bd	2018-01-13 13:55 UTC
2018/01/13/[$LATEST]064fbf40996847fd8a24b5b6cbd1e5af	2018-01-13 13:34 UTC
2018/01/07/[$LATEST]a60a31cdc718407bb7dd42b05f4735dc	2018-01-07 02:52 UTC
2018/01/07/[$LATEST]1c95f57d79624ef388349e71ce4ff34f	2018-01-07 02:48 UTC
2018/01/07/[$LATEST]bd9a05445760478b95d78afc916c4bd6	2018-01-07 01:23 UTC
2018/01/07/[$LATEST]00775bac85a04cb58bb421b87f6d57f5	2018-01-07 01:17 UTC
2018/01/06/[$LATEST]5c55f992924643718bfec831df799d3f	2018-01-06 22:27 UTC
2018/01/06/[$LATEST]45162e45baf64db697de04bb4af96aa1	2018-01-06 13:39 UTC
2018/01/06/[$LATEST]63b8533fec2845e8b65af4505984afa5	2018-01-06 03:57 UTC
2018/01/06/[$LATEST]63ff65ac546440889e944074f2b3a1e9	2018-01-06 01:24 UTC
2018/01/02/[$LATEST]f264f6c6cf344b4c8d6e9f549bdfc26c	2018-01-02 19:38 UTC
2018/01/02/[$LATEST]07308effe6dd4f549d23bd8adc5a8f93	2018-01-02 15:20 UTC
2017/12/24/[$LATEST]c3e00904081b4477ab254d823734c28c	2017-12-24 03:39 UTC
2017/12/24/[$LATEST]bdfbfe1fbf9d45a7a77eae39742d4165	2017-12-24 03:38 UTC
2017/12/24/[$LATEST]24249247d9864f8b954eea0554bd625d	2017-12-24 03:35 UTC
2017/12/24/[$LATEST]63851da58bcb4b078ceee2098e287c96	2017-12-24 03:31 UTC
2017/12/24/[$LATEST]16ca91740e524d44a20b90b86319573a	2017-12-24 03:24 UTC

CloudWatch > Log Groups > Streams for /aws/lambda/big-mouth-dev-get-index

Search Log Group | Create Log Stream | Delete Log Stream

one **log group** per function

Filter: Log Stream Name Prefix ×

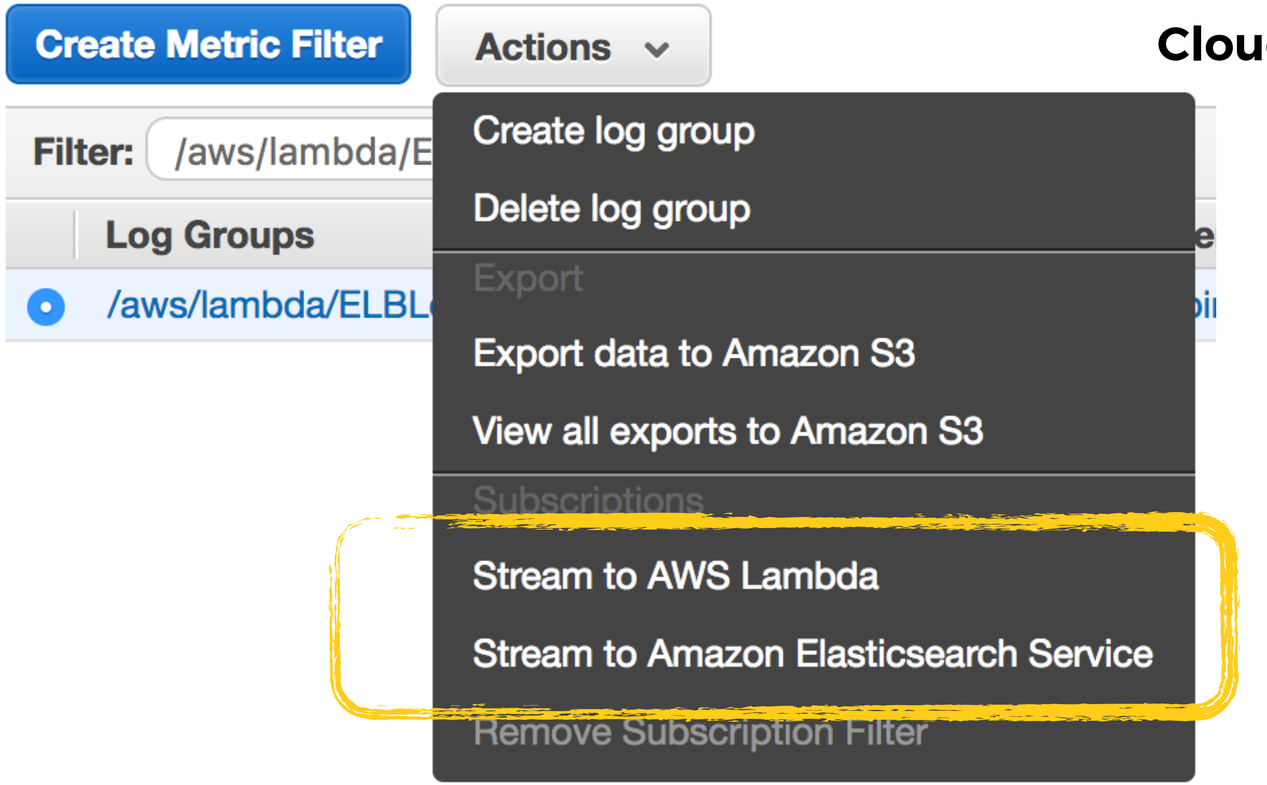
Log Streams 1-50

one **log stream** for each concurrent invocation

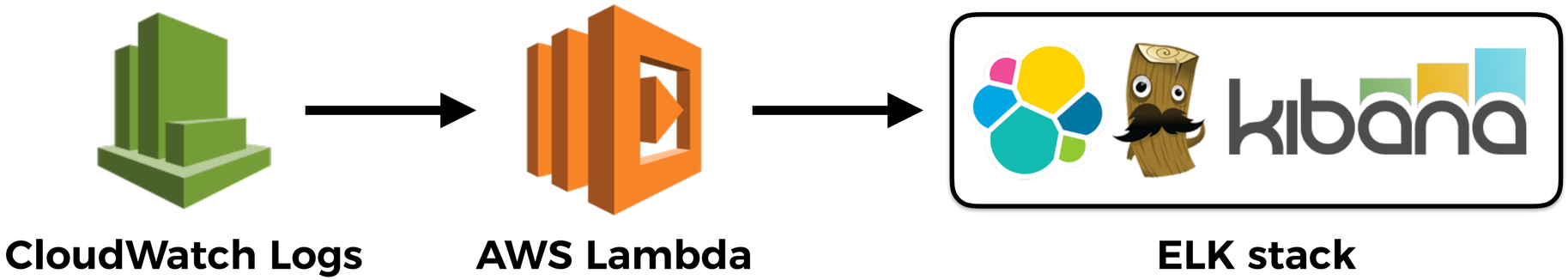
Log Streams	Last Event Time
2018/01/13/[$LATEST]cd9779a0b20f44db8a661ca156dff8a1	2018-01-13 14:23 UTC
2018/01/13/[$LATEST]cb65dd255c9a426a8516ae38b06cd0bd	2018-01-13 13:55 UTC
2018/01/13/[$LATEST]064fbf40996847fd8a24b5b6cbd1e5af	2018-01-13 13:34 UTC
2018/01/07/[$LATEST]a60a31cdc718407bb7dd42b05f4735dc	2018-01-07 02:52 UTC
2018/01/07/[$LATEST]1c95f57d79624ef388349e71ce4ff34f	
2018/01/07/[$LATEST]bd9a05445760478b95d78afc916c4bd6	
2018/01/07/[$LATEST]00775bac85a04cb58bb421b87f6d57f5	
2018/01/06/[$LATEST]5c55f992924643718bfec831df799d3f	
2018/01/06/[$LATEST]45162e45baf64db697de04bb4af96aa1	
2018/01/06/[$LATEST]63b8533fec2845e8b65af4505984afa5	2018-01-06 03:57 UTC
2018/01/06/[$LATEST]63ff65ac546440889e944074f2b3a1e9	2018-01-06 01:24 UTC
2018/01/02/[$LATEST]f264f6c6cf344b4c8d6e9f549bdfc26c	2018-01-02 19:38 UTC
2018/01/02/[$LATEST]07308effe6dd4f549d23bd8adc5a8f93	2018-01-02 15:20 UTC
2017/12/24/[$LATEST]c3e00904081b4477ab254d823734c28c	2017-12-24 03:39 UTC
2017/12/24/[$LATEST]bdfbfe1fbf9d45a7a77eae39742d4165	2017-12-24 03:38 UTC
2017/12/24/[$LATEST]24249247d9864f8b954eea0554bd625d	2017-12-24 03:35 UTC
2017/12/24/[$LATEST]63851da58bcb4b078ceee2098e287c96	2017-12-24 03:31 UTC
2017/12/24/[$LATEST]16ca91740e524d44a20b90b86319573a	2017-12-24 03:24 UTC

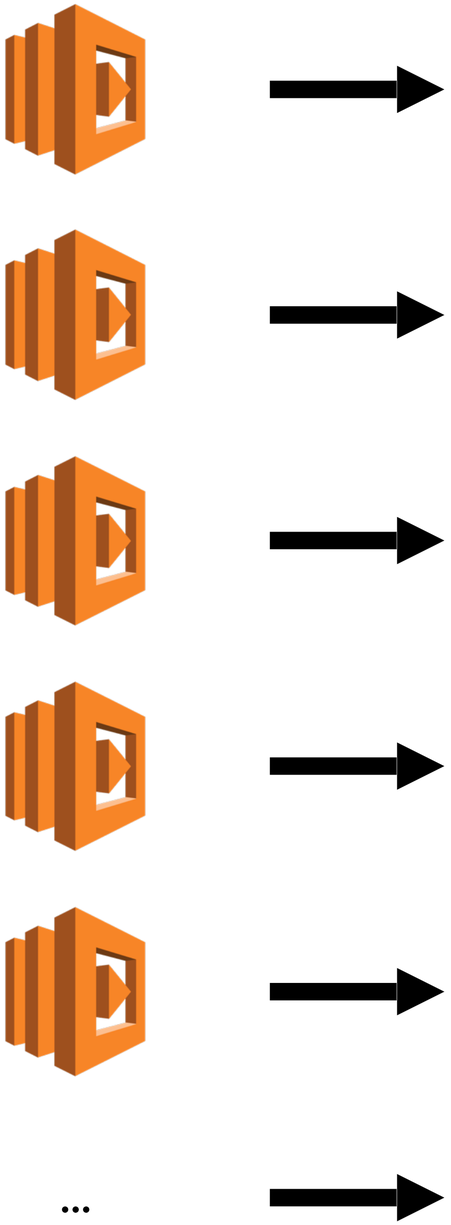
logs are not easily searchable in CloudWatch Logs

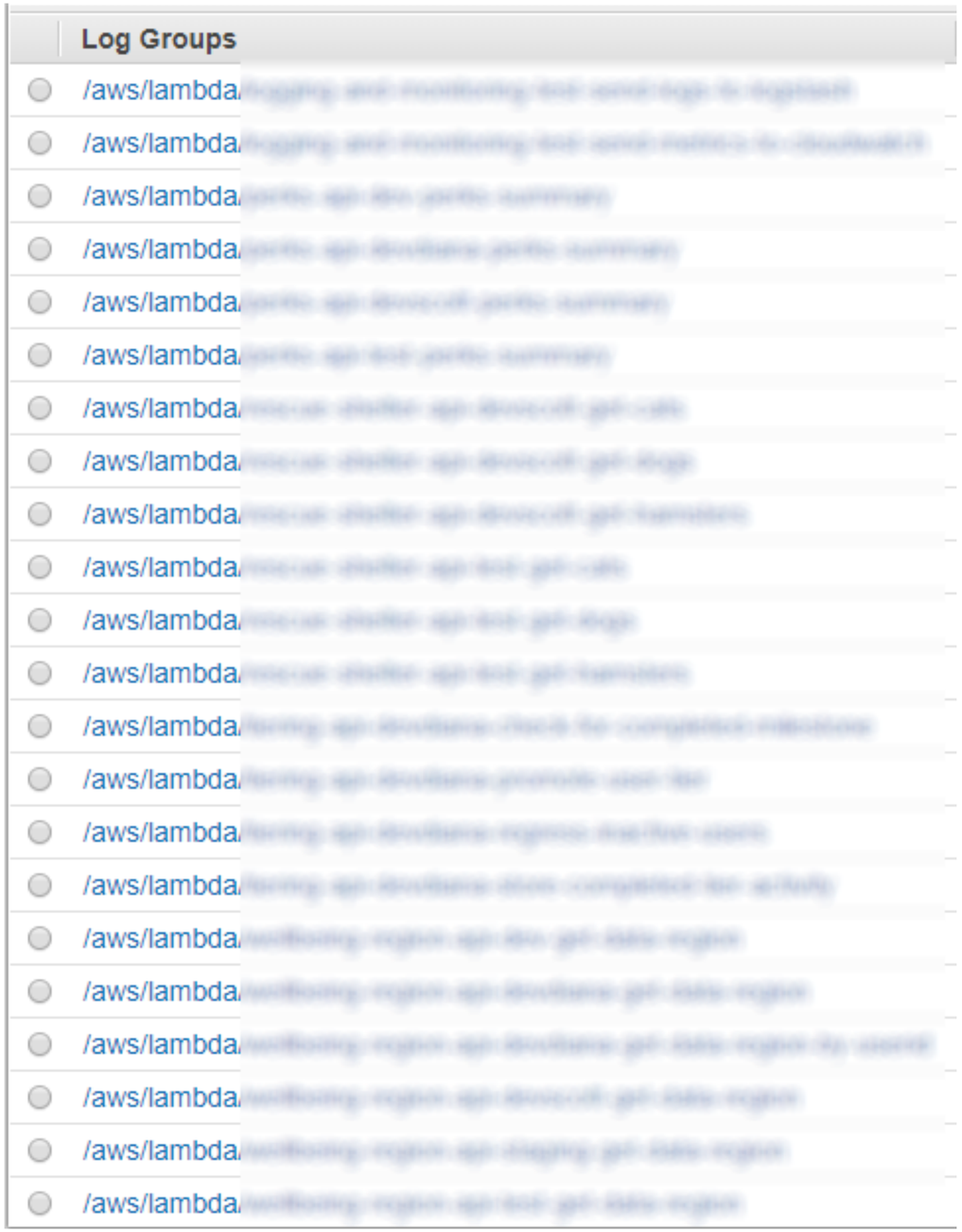
me



CloudWatch Logs


kibana
CloudWatch Logs
AWS Lambda
ELK stack


...


Log Groups
/aws/lambda
/aws/lambda
/aws/lambda
/aws/lambda
/aws/lambda
/aws/lambda
/aws/lambda
/aws/lambda
/aws/lambda
/aws/lambda
/aws/lambda
/aws/lambda
/aws/lambda
/aws/lambda
/aws/lambda
/aws/lambda
/aws/lambda
/aws/lambda
/aws/lambda
/aws/lambda
/aws/lambda
/aws/lambda

Event Source

Build or customize an Event Pattern or set a Schedule to invoke Targets.

Event Pattern Schedule

Build event pattern to match events by service

Service Name: CloudWatch Logs

Event Type: AWS API Call via CloudTrail

Any operation Specific operation(s)

× CreateLogGroup

Event Pattern Preview Copy to clipboard Edit

```
{
 "source": [
  "aws.logs"
 ],
 "detail-type": [
  "AWS API Call via CloudTrail"
 ],
 "detail": {
  "eventSource": [
   "logs.amazonaws.com"
  ],
  "eventName": [
   "CreateLogGroup"
  ]
```

Targets

Select Target to invoke when an event matches your Event Pattern or when schedule is triggered.

Lambda function

Function: subscribe-log-group

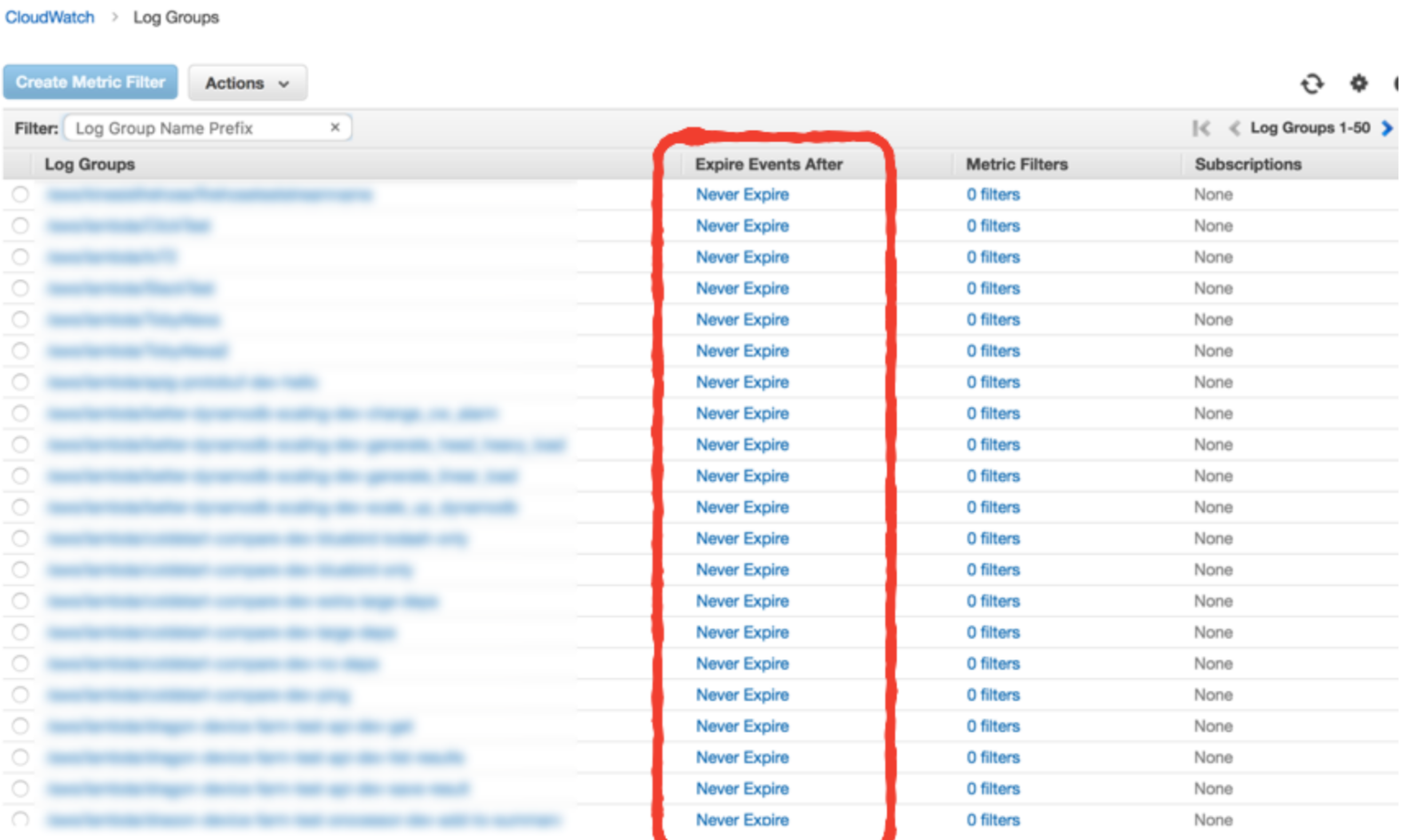
Configure version/alias

Configure input

Add target*

CloudWatch Logs

```yaml
subscribe:
  handler: functions/subscribe/handler.handler
  description: Subscribe logs to the ship-log function
  environment:
    DEST_FUNC: "arn:aws:lambda:#{AWS::Region}:#{AWS::AccountId}:function:${self:service}-${se
  events:
    - cloudwatchEvent:
        event:
          source:
            - aws.logs
          detail-type:
            - AWS API Call via CloudTrail
          detail:
            eventSource:
              - logs.amazonaws.com
            eventName:
              - CreateLogGroup
```



	Log Groups	Expire Events After	Metric Filters	Subscriptions
○	[redacted]	Never Expire	0 filters	None
○	[redacted]	Never Expire	0 filters	None
○	[redacted]	Never Expire	0 filters	None
○	[redacted]	Never Expire	0 filters	None
○	[redacted]	Never Expire	0 filters	None
○	[redacted]	Never Expire	0 filters	None
○	[redacted]	Never Expire	0 filters	None
○	[redacted]	Never Expire	0 filters	None
○	[redacted]	Never Expire	0 filters	None
○	[redacted]	Never Expire	0 filters	None
○	[redacted]	Never Expire	0 filters	None
○	[redacted]	Never Expire	0 filters	None
○	[redacted]	Never Expire	0 filters	None
○	[redacted]	Never Expire	0 filters	None
○	[redacted]	Never Expire	0 filters	None
○	[redacted]	Never Expire	0 filters	None
○	[redacted]	Never Expire	0 filters	None
○	[redacted]	Never Expire	0 filters	None
○	[redacted]	Never Expire	0 filters	None
○	[redacted]	Never Expire	0 filters	None
○	[redacted]	Never Expire	0 filters	None

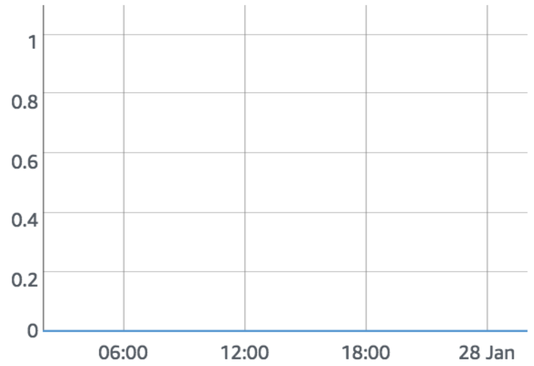
apig-iscoldstart-dev-hello

Qualifiers Actions Select a test event.. Test Save

Invocation count

Last 24 hours

Jump to Metrics Jump to Logs

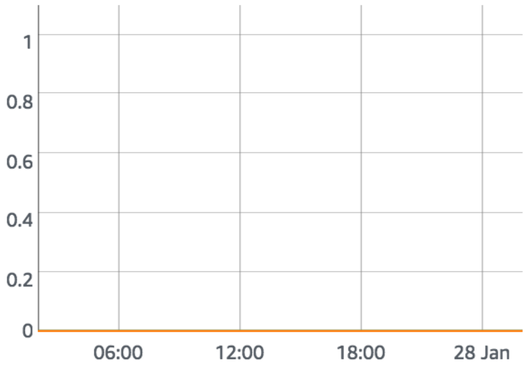


Count

Invocation duration

Last 24 hours

Jump to Metrics Jump to Logs

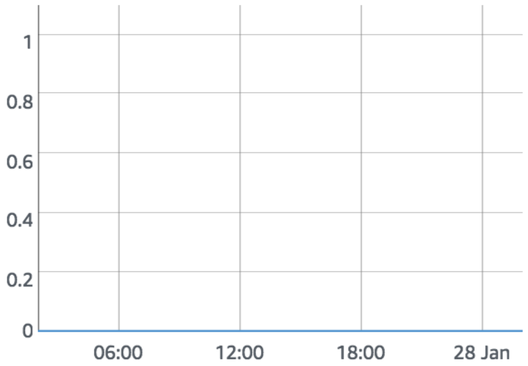


Max Milliseconds
Avg Milliseconds
Min Milliseconds

Invocation errors

Last 24 hours

Jump to Metrics Jump to Logs

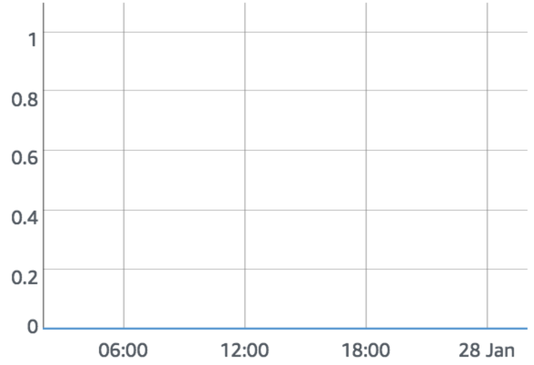


Count

Throttled invocations

Last 24 hours

Jump to Metrics Jump to Logs

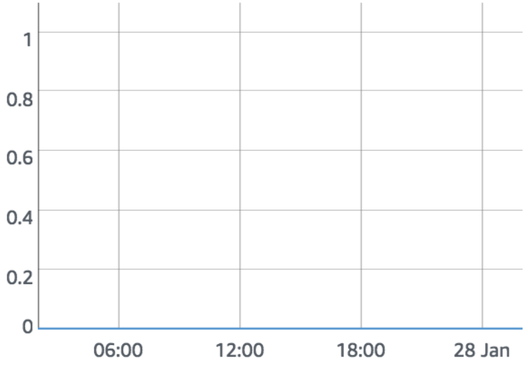


Count

Iterator age

Last 24 hours

Jump to Metrics Jump to Logs

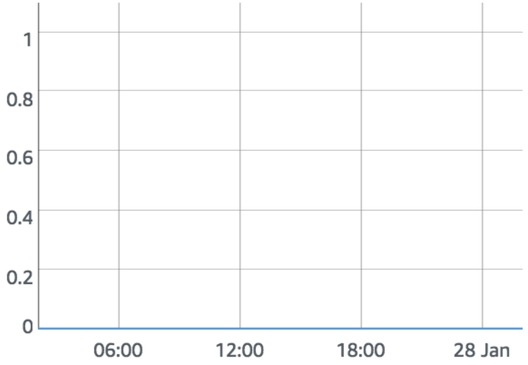


Max Milliseconds

DLQ errors

Last 24 hours

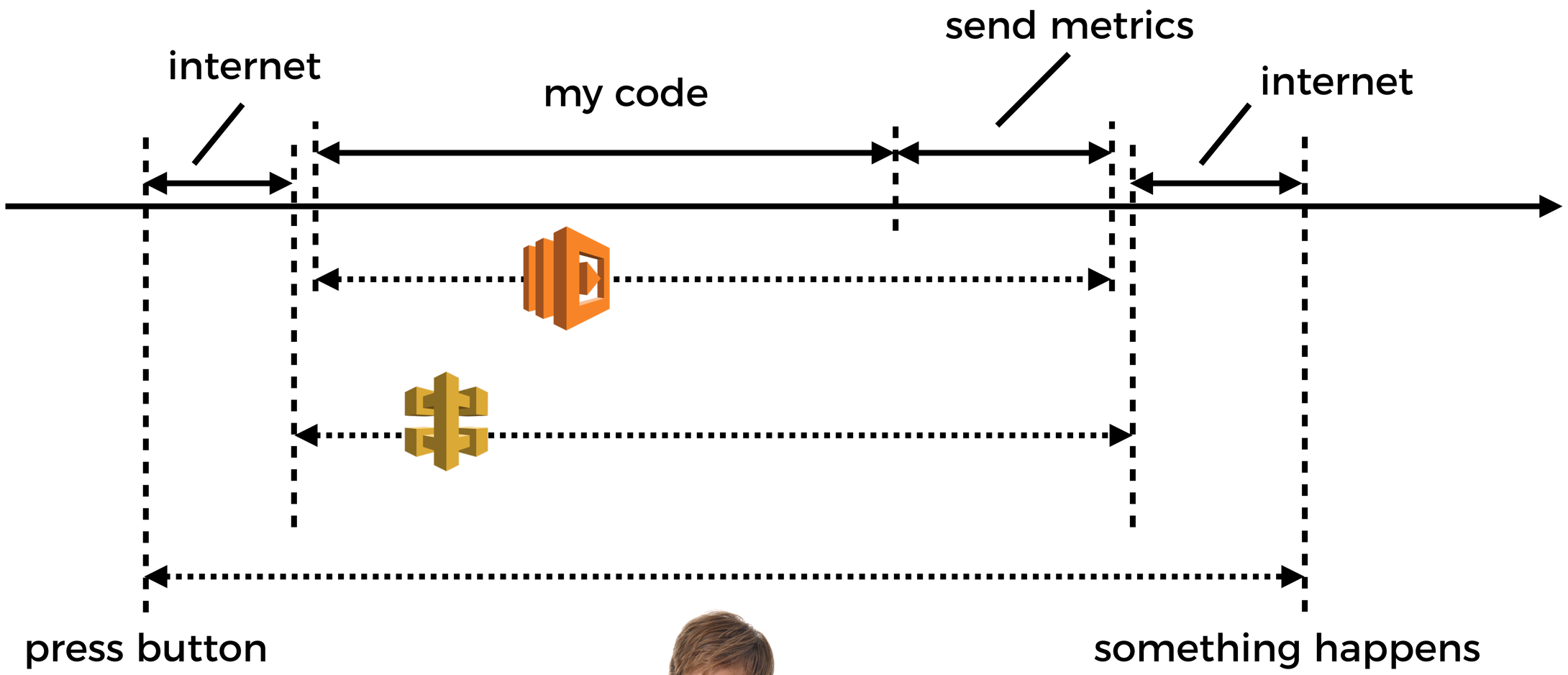
Jump to Metrics Jump to Logs

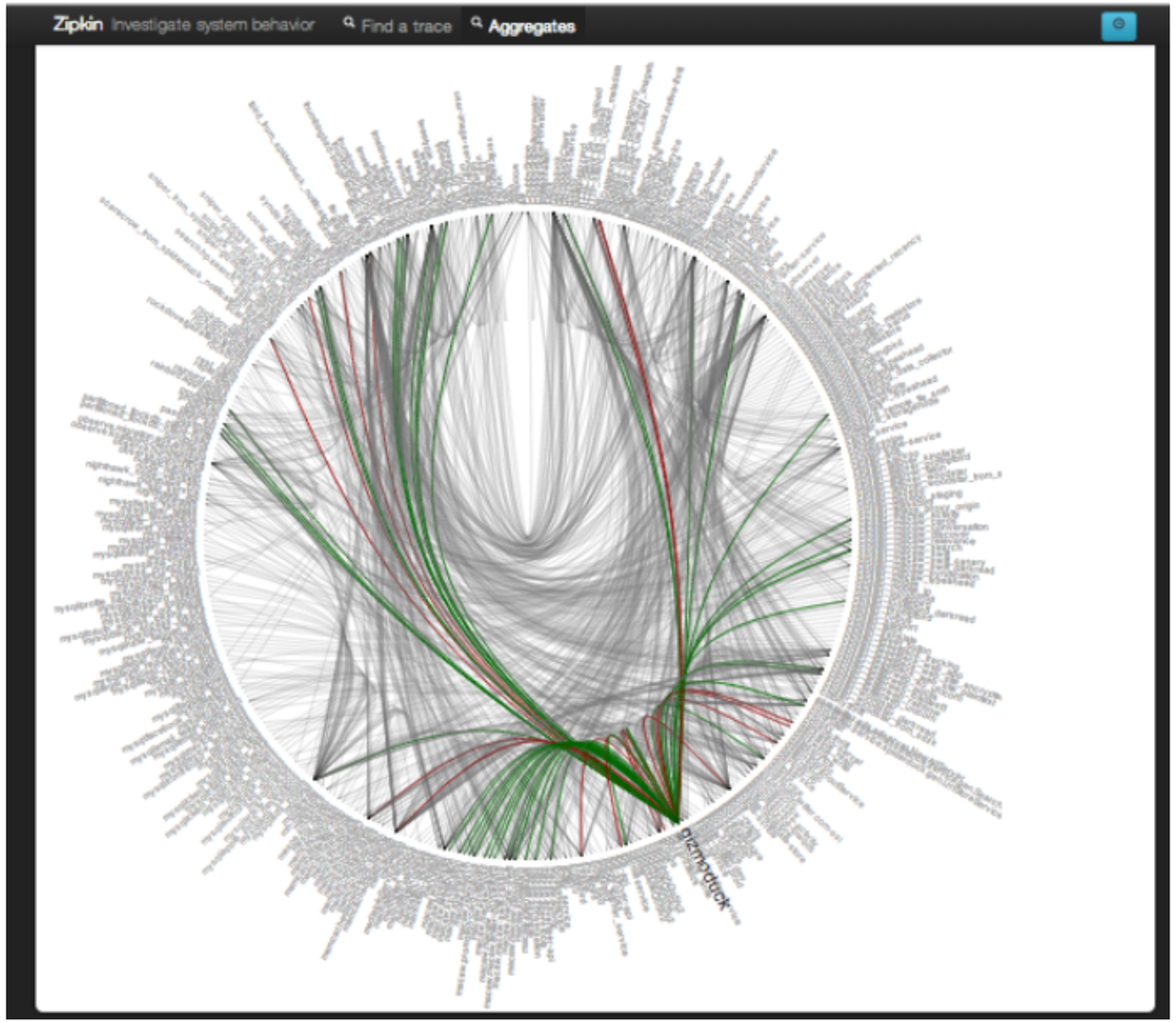


Count

new challenges

- **nowhere** to install agents/daemons
- **no** background processing


send metrics
internet
my code
internet
press button
something happens



those extra 10-20ms for sending custom metrics would **compound** when you have microservices and multiple APIs are called within one slice of user event

Amazon found every **100ms** of latency **cost them 1% in sales**.

http://bit.ly/2EXPfbA

metrics

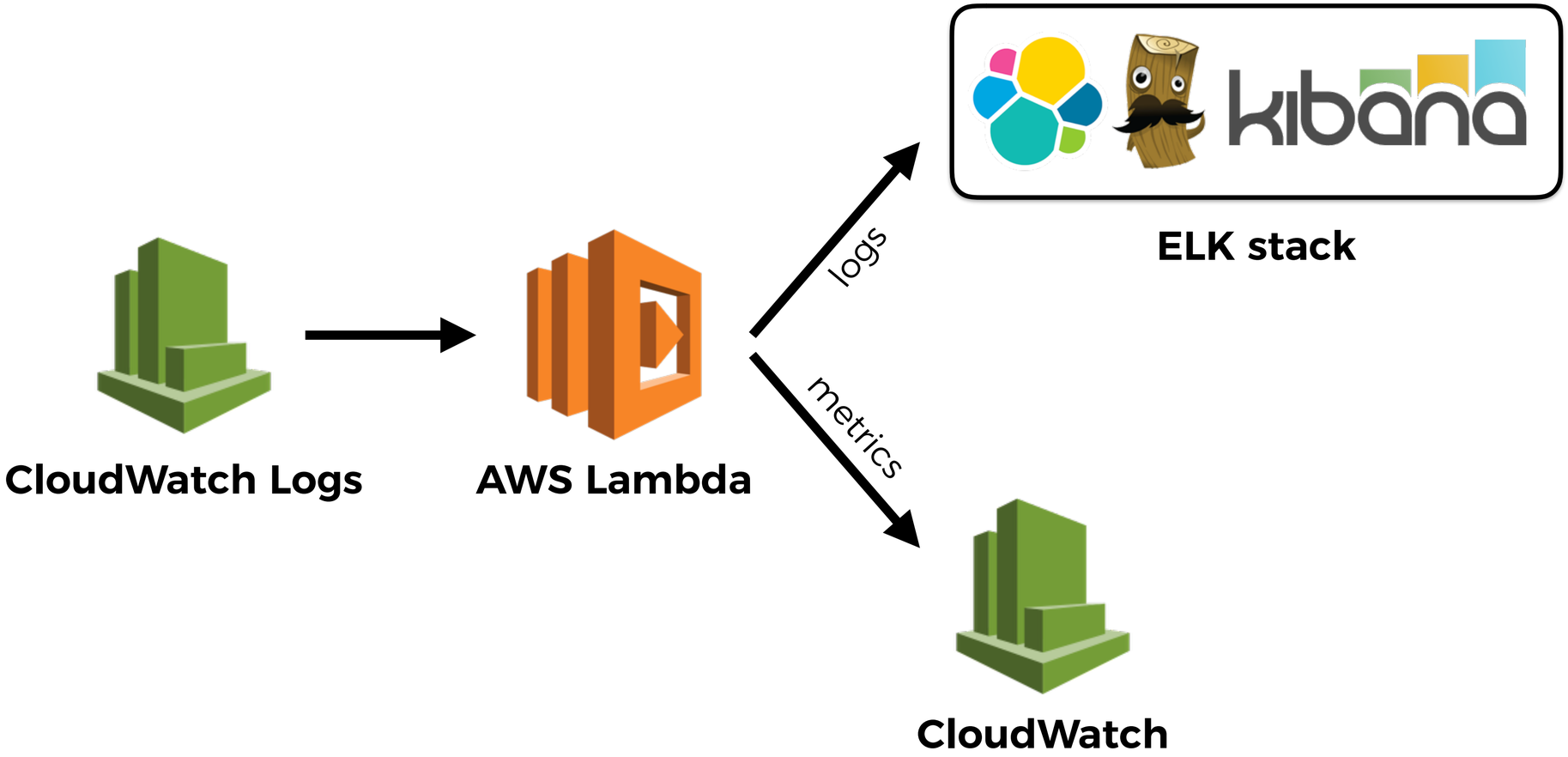
timestamp | metric value | metric name

```
console.log("MONITORING|1489795335|27.4|latency|user-api-latency");
console.log("MONITORING|1489795335|8|count|yubls-served");
```

metric type

logs

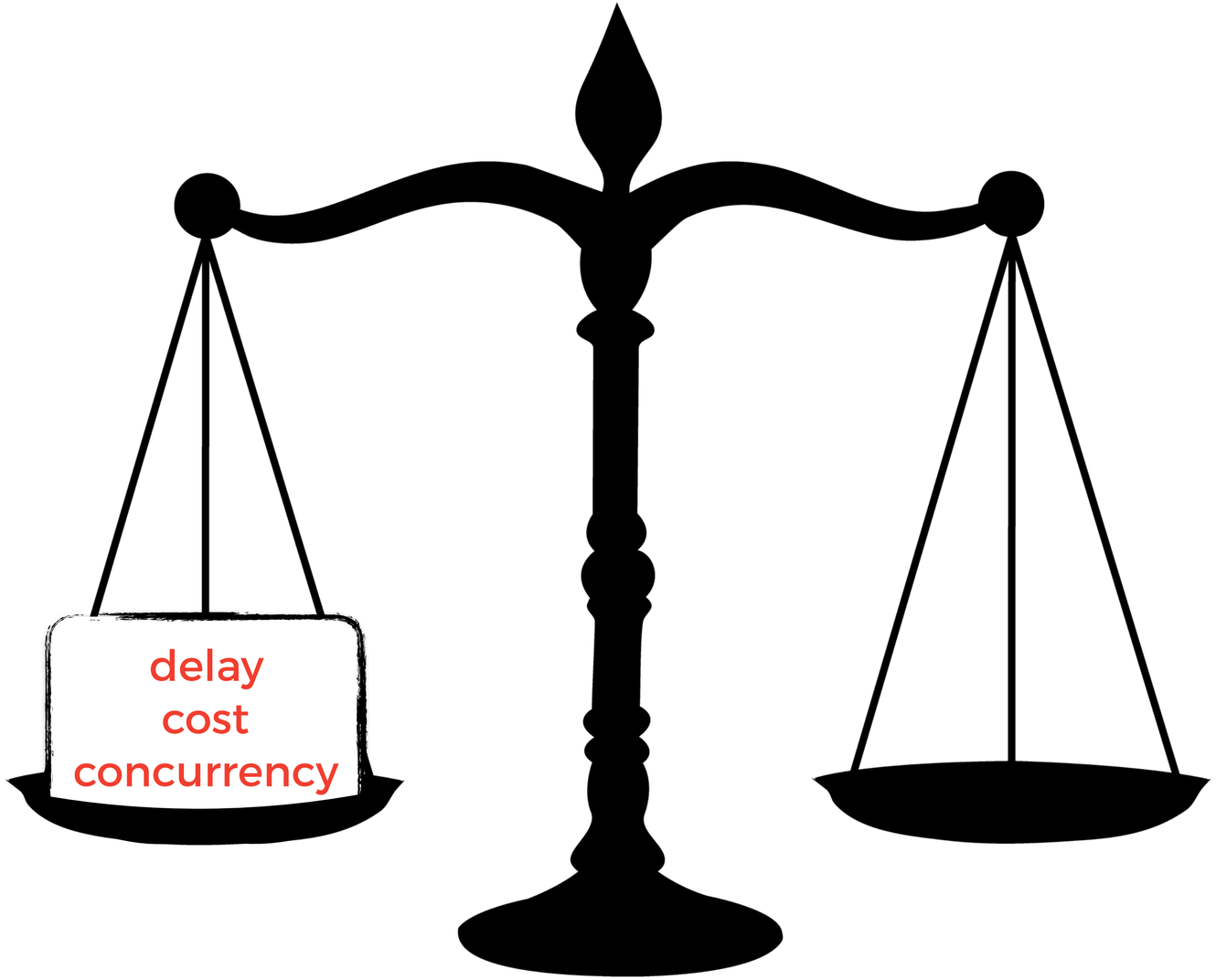
```
console.log("hydrating yubls from db…");
console.log("fetching user info from user-api");
```

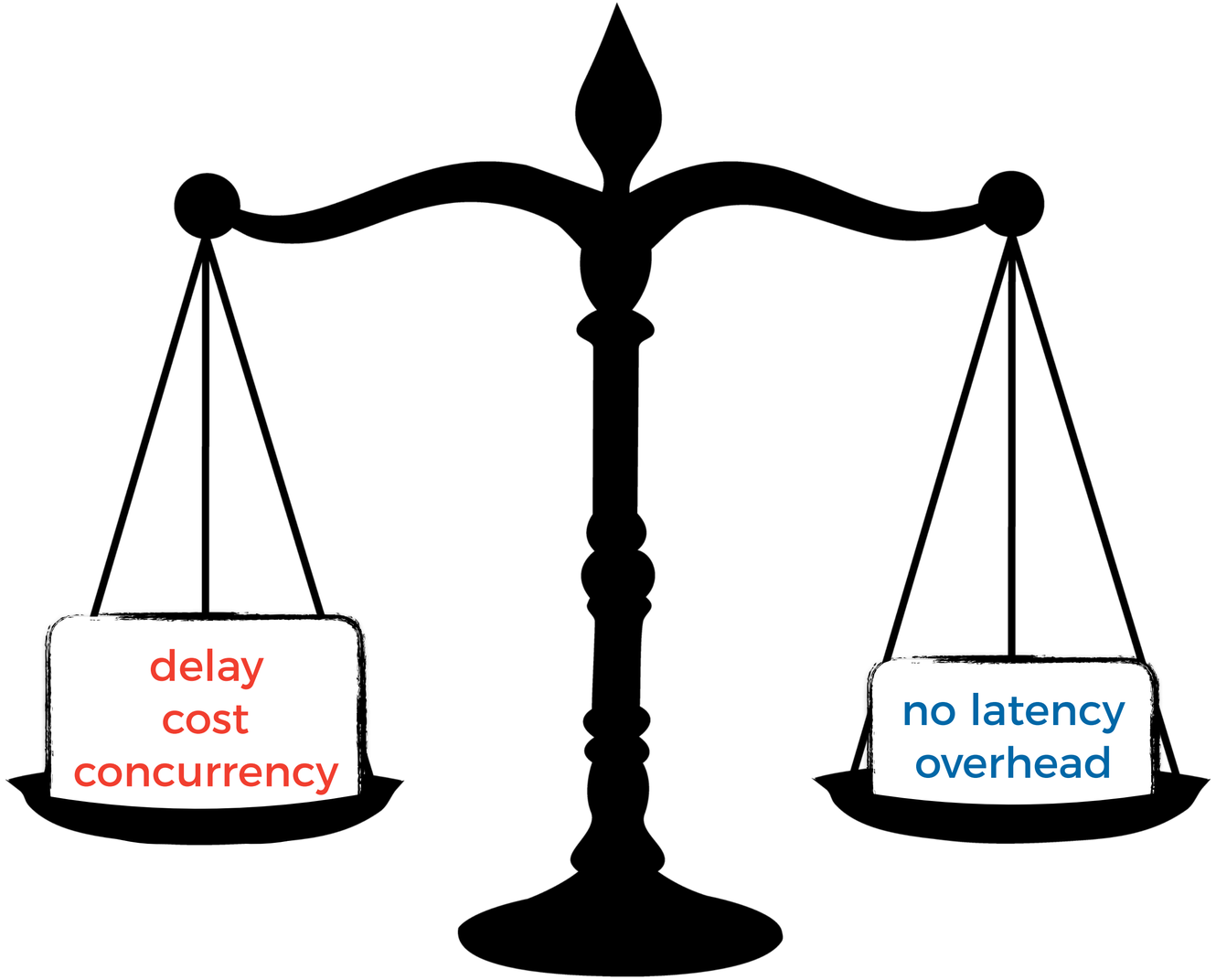

kibana
ELK stack
logs
metrics
CloudWatch Logs
AWS Lambda
CloudWatch

THUNDRA

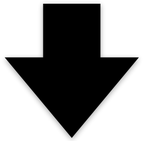
PURESEC

DASHBIRD


delay
cost
concurrency

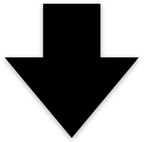

delay
cost
concurrency
no latency
overhead

API Gateway



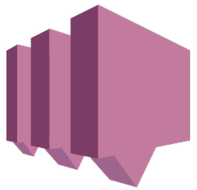
send custom metrics
asynchronously

API Gateway



send custom metrics asynchronously

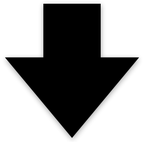
S3



SNS

Kinesis

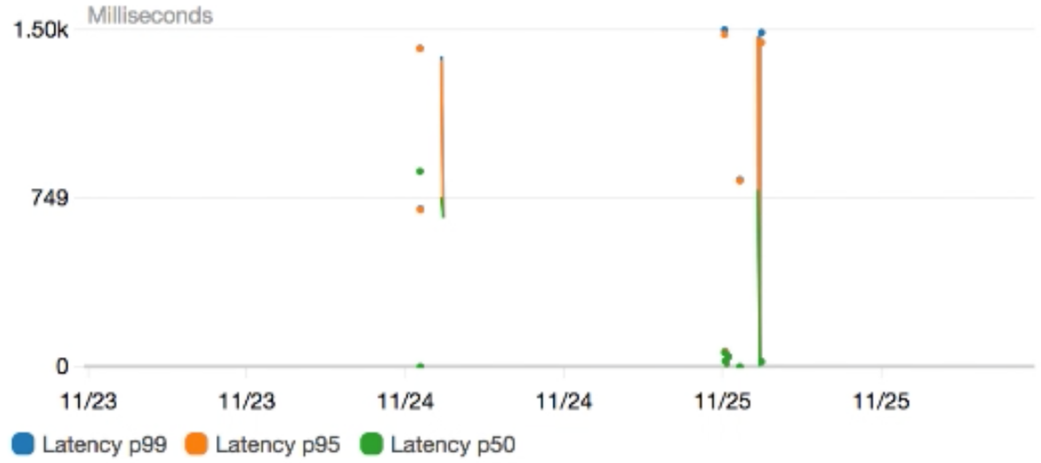
...



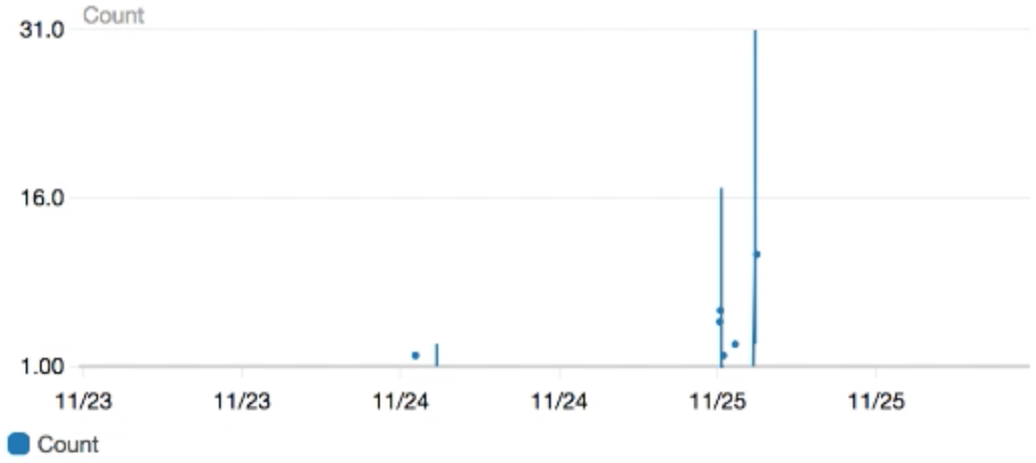
send custom metrics as part of function invocation

big-mouth | Add widget | Actions | Save dashboard | 1h 3h 12h 1d 3d 1w custom

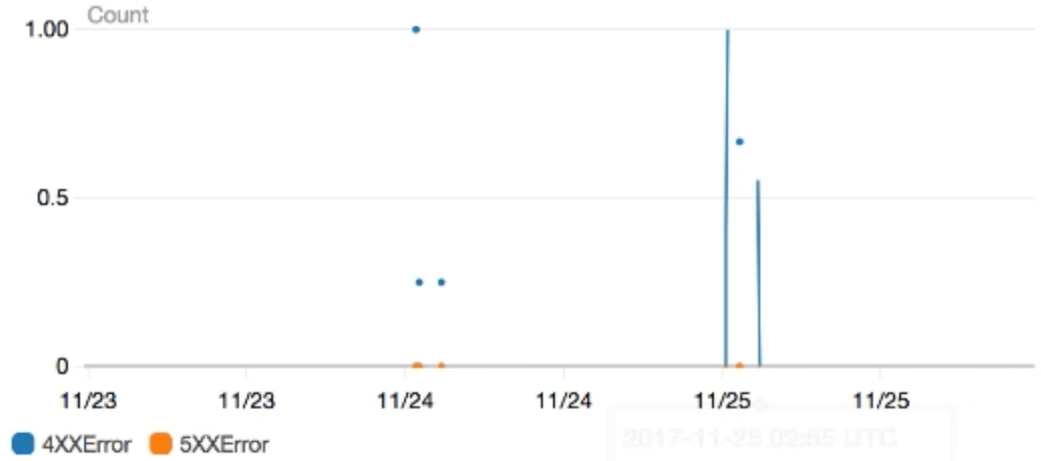
Latency



Count



4XXError, 5XXError



X-Ray

AWS X-Ray helps developers analyze and debug production, distributed applications, such as those built using a microservices architecture. With X-Ray, you can understand how your application and its underlying services are performing to identify and troubleshoot the root cause of performance issues and errors. X-Ray provides an end-to-end view of requests as they travel through your application, and shows a map of your application's underlying components. You can use X-Ray to analyze both applications in development and in production, from simple three-tier applications to complex microservices applications consisting of thousands of services.

How It Works

TRACE REQUESTS

AWS X-Ray traces requests made to your application.

X-Ray collects data about the request from each of the underlying application services it passes through.

RECORD TRACES

X-Ray combines the data gathered from each service into singular units called traces.

VIEW SERVICE MAP

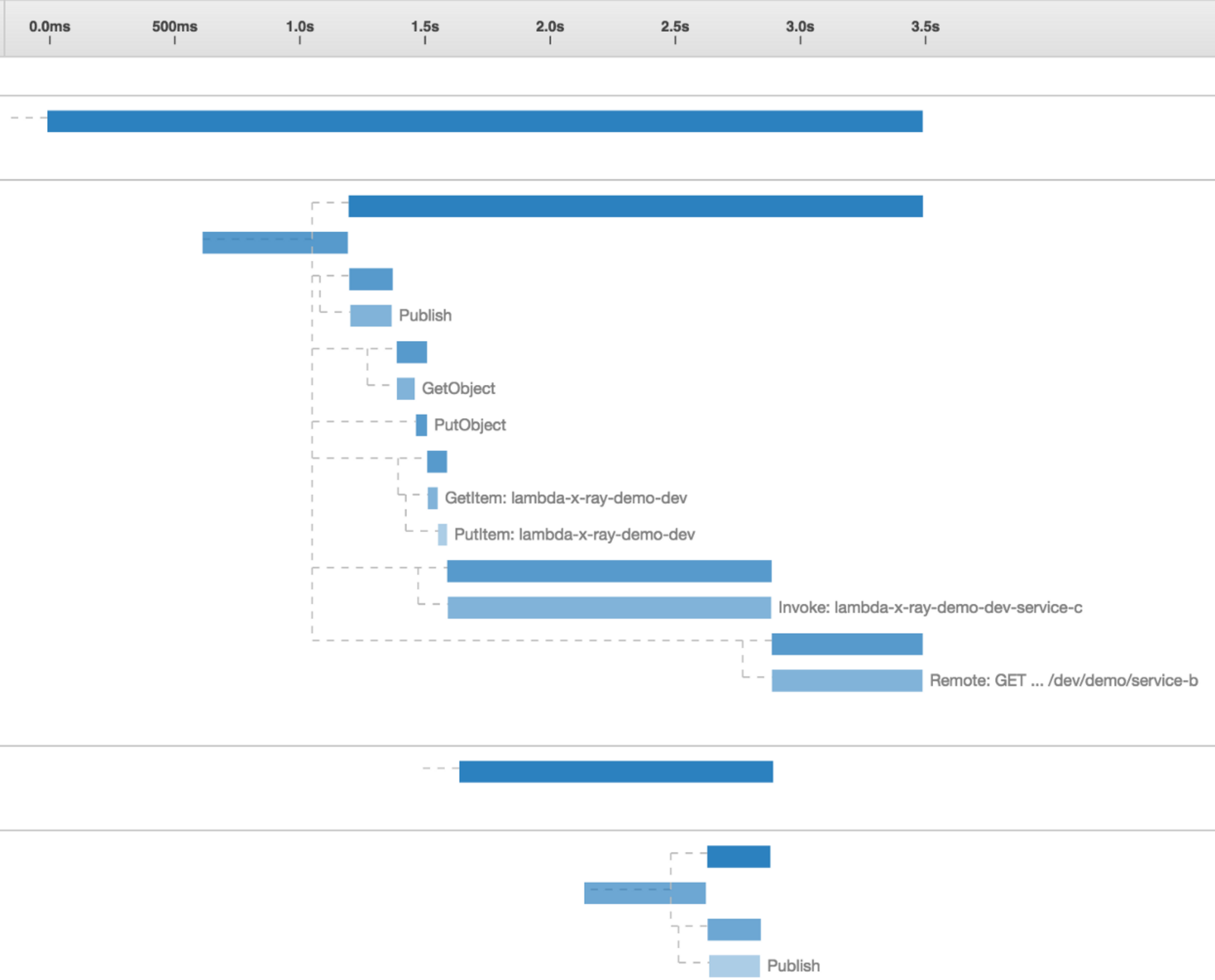
View the service map to see trace data such as latencies, HTTP statuses, and metadata for each service.

ANALYZE ISSUES

Drill into the service showing unusual behavior to identify the root issue.

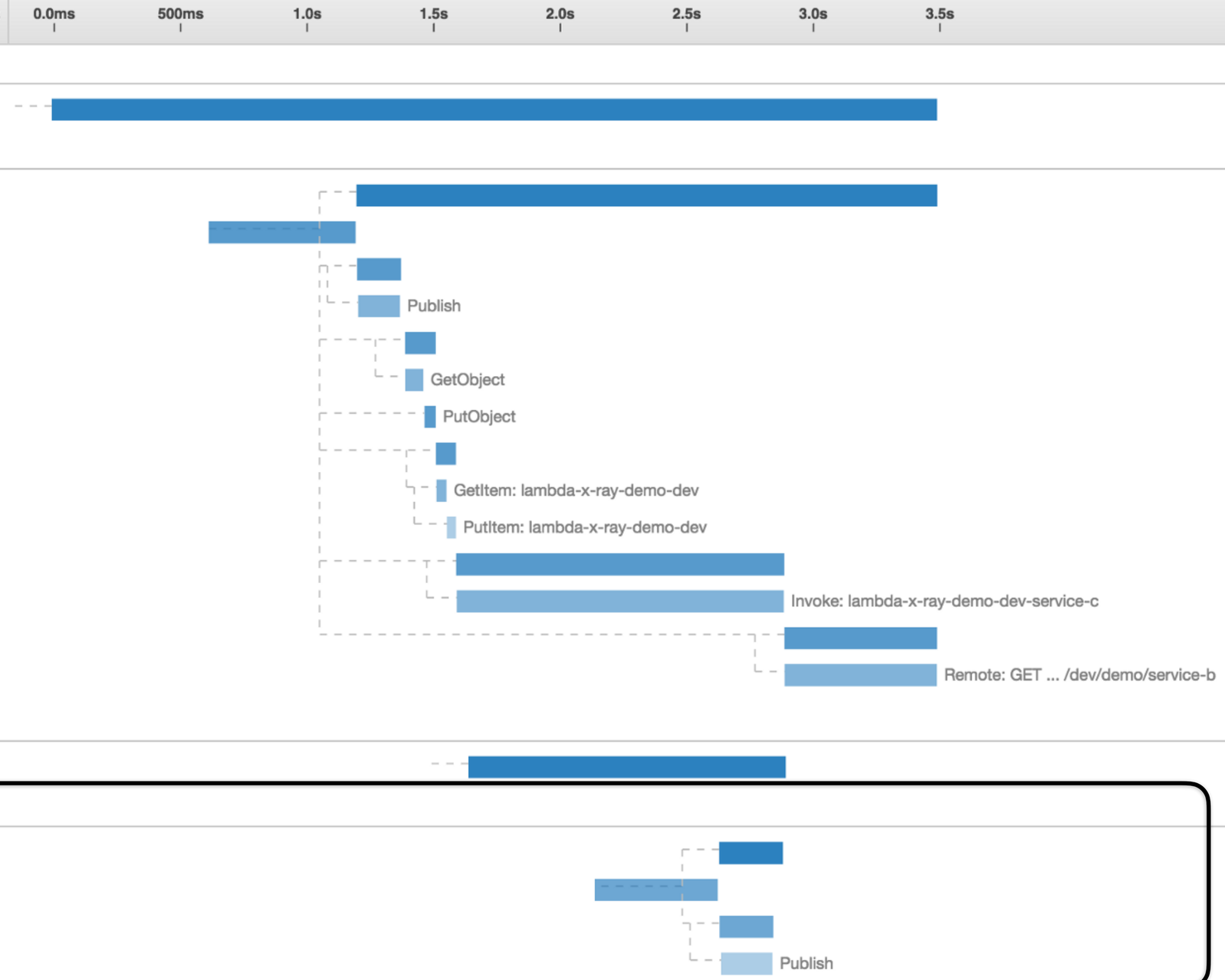
Timeline | Raw data

Name	Res.	Duration	Status
lambda-x-ray-demo-dev-service-a AWS::Lambda			
lambda-x-ray-demo-dev-service-a	200	3.2 sec	✓
lambda-x-ray-demo-dev-service-a AWS::Lambda::Function			
lambda-x-ray-demo-dev-service-a	-	2.1 sec	✓
Initialization	-	529 ms	✓
## publishing to SNS	-	158 ms	✓
SNS	200	151 ms	✓
## accessing S3	-	110 ms	✓
S3	404	65.0 ms	!
S3	200	40.0 ms	✓
## accessing DynamoDB	-	72.0 ms	✓
DynamoDB	200	36.0 ms	✓
DynamoDB	200	32.0 ms	✓
## invoking service-c	-	1.2 sec	✓
Lambda	200	1.2 sec	✓
## calling service b	-	550 ms	✓
1g7cabdok5.execute-api.us-east-1.amazonav	200	548 ms	✓
lambda-x-ray-demo-dev-service-c AWS::Lambda			
lambda-x-ray-demo-dev-service-c	200	1.1 sec	✓
lambda-x-ray-demo-dev-service-c AWS::Lambda::Function			
lambda-x-ray-demo-dev-service-c	-	230 ms	✓
Initialization	-	443 ms	✓
## publishing to SNS	-	193 ms	✓
SNS	200	185 ms	✓



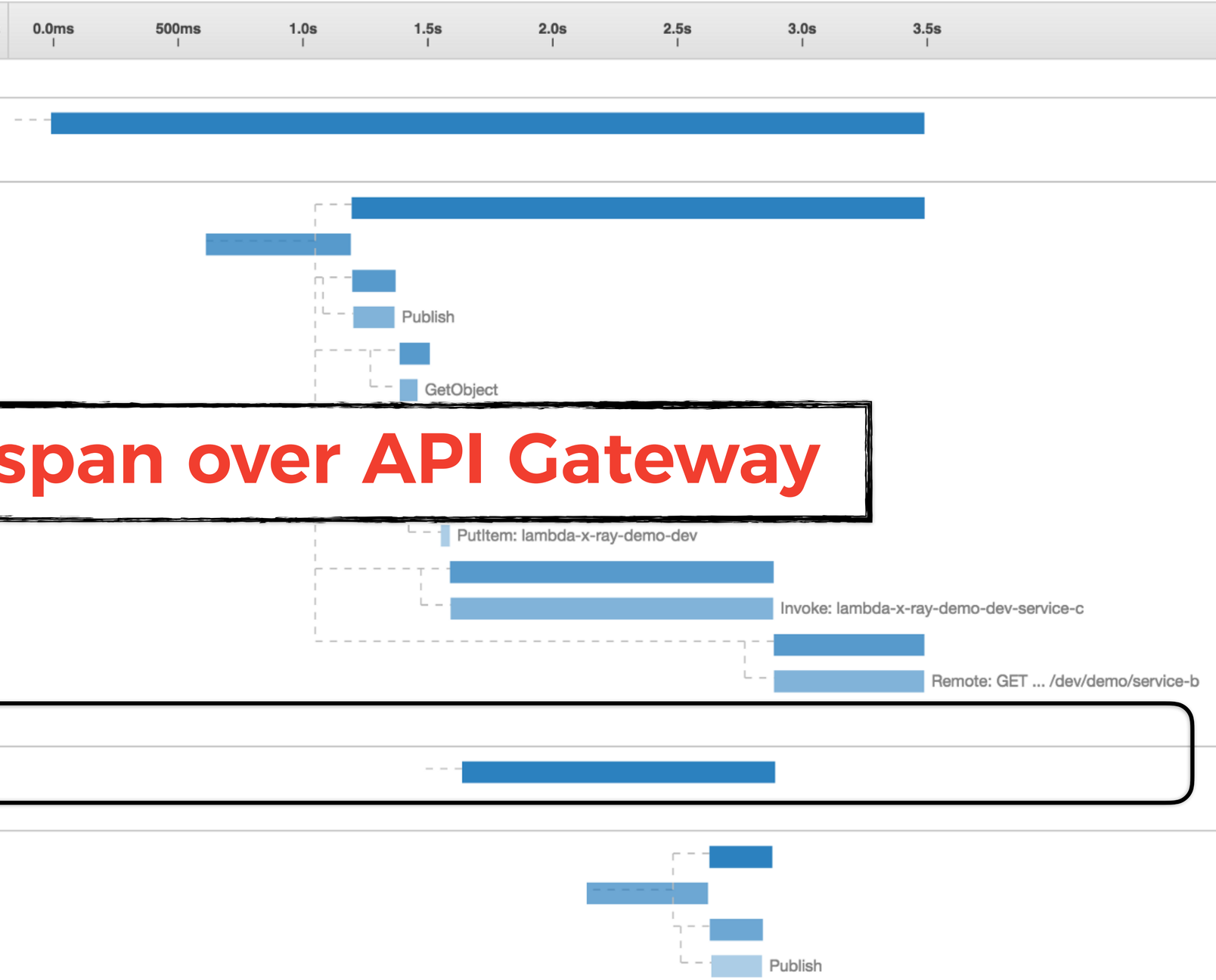
Timeline | Raw data

Name	Res.	Duration	Status
lambda-x-ray-demo-dev-service-a AWS::Lambda			
lambda-x-ray-demo-dev-service-a	200	3.2 sec	✓
lambda-x-ray-demo-dev-service-a AWS::Lambda::Function			
lambda-x-ray-demo-dev-service-a	-	2.1 sec	✓
Initialization	-	529 ms	✓
## publishing to SNS	-	158 ms	✓
SNS	200	151 ms	✓
## accessing S3	-	110 ms	✓
S3	404	65.0 ms	!
S3	200	40.0 ms	✓
## accessing DynamoDB	-	72.0 ms	✓
DynamoDB	200	36.0 ms	✓
DynamoDB	200	32.0 ms	✓
## invoking service-c	-	1.2 sec	✓
Lambda	200	1.2 sec	✓
## calling service b	-	550 ms	✓
1g7cabdok5.execute-api.us-east-1.ar lav	200	548 ms	✓
lambda-x-ray-demo-dev-service-c A bda			
lambda-x-ray-demo-dev-service-c	200	1.1 sec	✓
lambda-x-ray-demo-dev-service-c AWS::Lambda::Function			
lambda-x-ray-demo-dev-service-c	-	230 ms	✓
Initialization	-	443 ms	✓
## publishing to SNS	-	193 ms	✓
SNS	200	185 ms	✓



Timeline | Raw data

Name	Res.	Duration	Status
lambda-x-ray-demo-dev-service-a AWS::Lambda			
lambda-x-ray-demo-dev-service-a	200	3.2 sec	✓
lambda-x-ray-demo-dev-service-a AWS::Lambda::Function			
lambda-x-ray-demo-dev-service-a	-	2.1 sec	✓
Initialization	-	529 ms	✓
## publishing to SNS	-	158 ms	✓
SNS	200	151 ms	✓
## accessing S3	-	110 ms	✓
S3	404	65.0 ms	!
S3			
## accessing DynamoDB			
DynamoDB			
DynamoDB	200	32.0 ms	✓
## invoking service-c	-	1.2 sec	✓
Lambda	200	1.2 sec	✓
## calling service b	-	550 ms	✓
1g7cabdok5.execute-api.us-east-1.amazonav	200	548 ms	✓
lambda-x-ray-demo-dev-service-c AWS::Lambda			
lambda-x-ray-demo-dev-service-c	200	1.1 sec	✓
lambda-x-ray-demo-dev-service-c AWS::Lambda::Function			
lambda-x-ray-demo-dev-service-c	-	230 ms	✓
Initialization	-	443 ms	✓
## publishing to SNS	-	193 ms	✓
SNS	200	185 ms	✓



do not span over API Gateway

Timeline | Raw data

Name	Res.	Duration	Status
lambda-x-ray-demo-dev-service-a AWS::Lambda			
lambda-x-ray-demo-dev-service-a	200	3.2 sec	✓
lambda-x-ray-demo-dev-service-a AWS::Lambda::Function			
lambda-x-ray-demo-dev-service-a	-	2.1 sec	✓
Initialization			
## publishing to SNS			
SNS			
## accessing S3			
S3			
S3			
## accessing DynamoDB			
DynamoDB			
DynamoDB			
## invoking service-c			
Lambda			
## calling service b			
1g7cabdok5.execute-api.us-east-1.amazonav			
lambda-x-ray-demo-dev-service-c AWS::Lambda			
lambda-x-ray-demo-dev-service-c	200	1.1 sec	✓
lambda-x-ray-demo-dev-service-c AWS::Lambda::Function			
lambda-x-ray-demo-dev-service-c	-	230 ms	✓
Initialization	-	443 ms	✓
## publishing to SNS	-	193 ms	✓
SNS	200	185 ms	✓

0.0ms 500ms 1.0s 1.5s 2.0s 2.5s 3.0s 3.5s

e: lambda-x-ray-demo-dev-service-c

Remote: GET ... /dev/demo/service-b

Publish

narrow focus on a function

good for homing in on performance issues for a particular function, but offers little to help you build **intuition** about how your system operates as a whole.

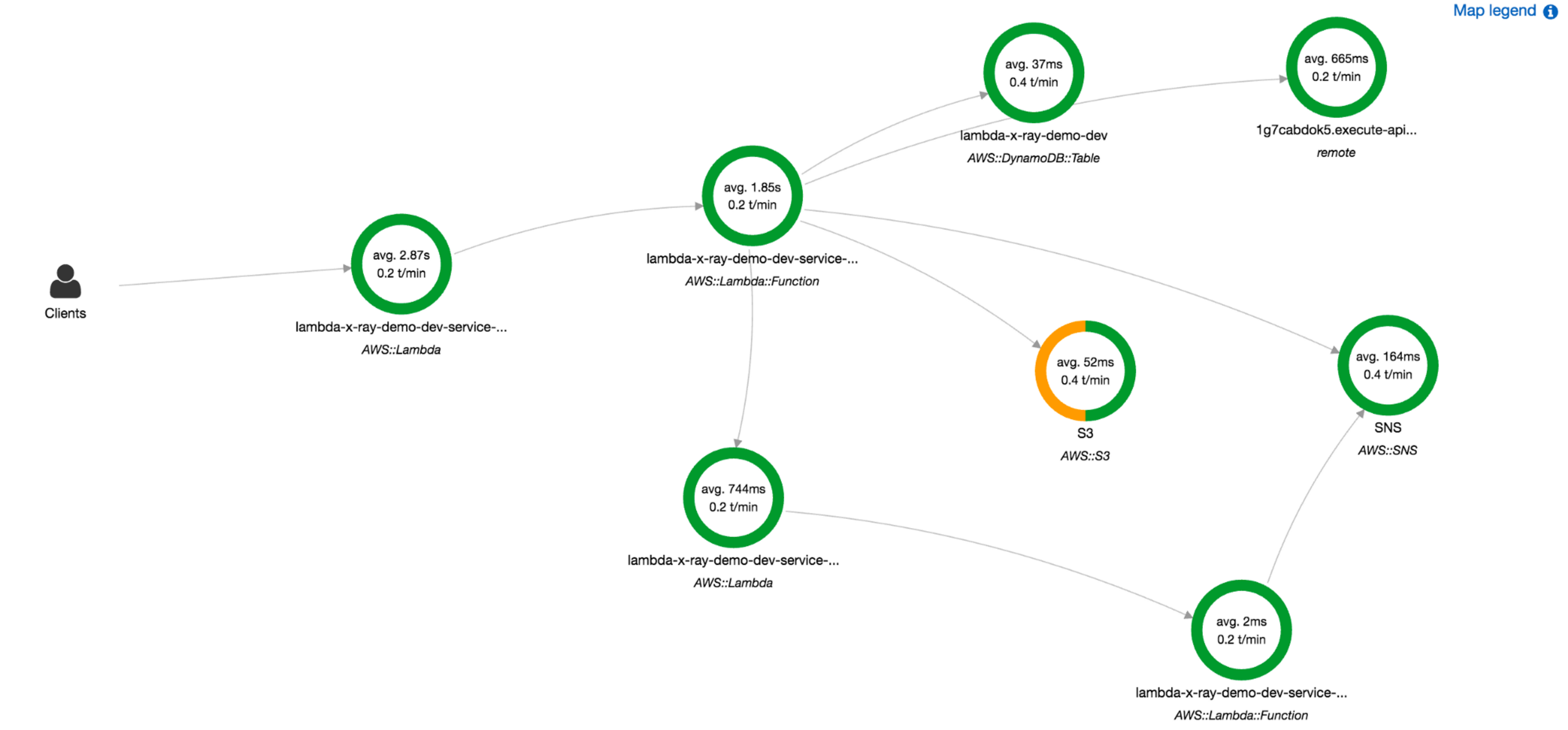
> *However, I would argue that the health of the system no longer matters. We've entered an era where what matters is the health of each individual event, or each individual user's experience, or each shopping cart's experience (or other high cardinality dimensions). With distributed systems you don't care about the health of the system, you care about the* ***health of the event or the slice****.*

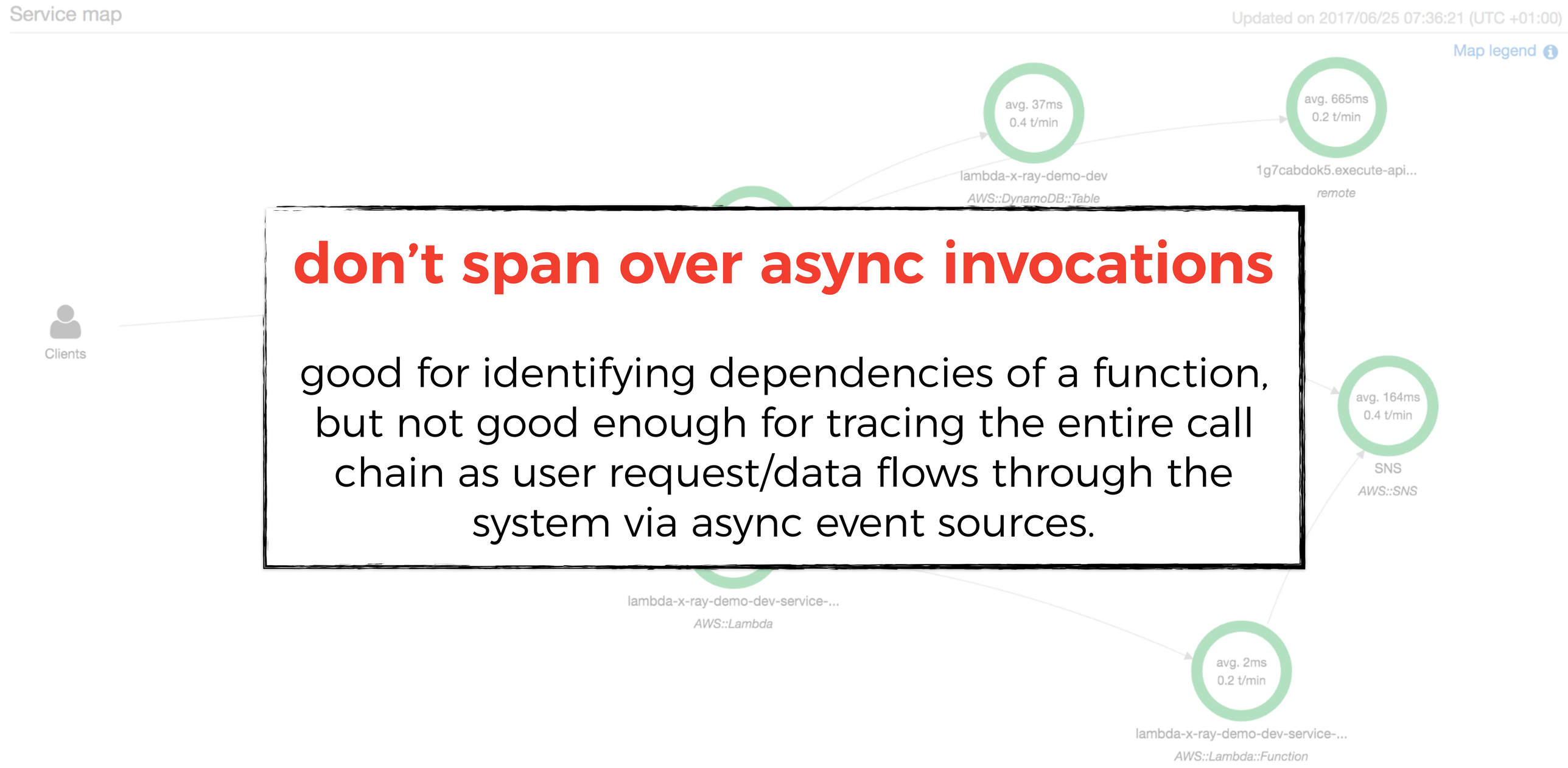
- Charity Majors http://bit.ly/2E2QngU


follow the data

Service map

Updated on 2017/06/25 07:36:21 (UTC +01:00)





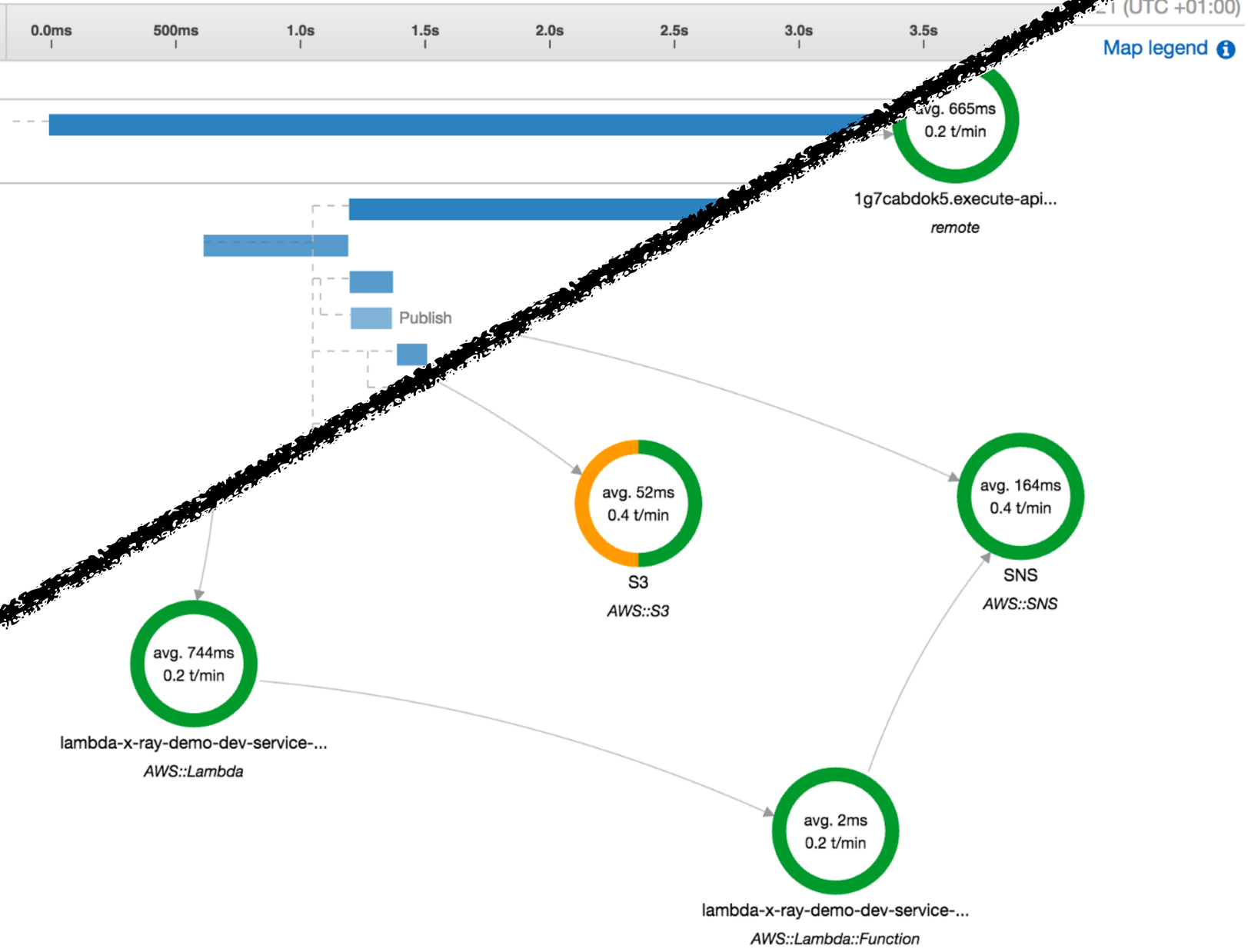
don’t span over async invocations

good for identifying dependencies of a function, but not good enough for tracing the entire call chain as user request/data flows through the system via async event sources.

don’t span over non-AWS services

Timeline | Raw data

Name	Res.	Duration	Status
lambda-x-ray-demo-dev-service-a AWS::Lambda			
lambda-x-ray-demo-dev-service-a	200	3.2 sec	✓
lambda-x-ray-demo-dev-service-a AWS::Lambda::Function			
lambda-x-ray-demo-dev-service-a	-	2.1 sec	✓
Initialization	-	529 ms	✓
## publishing to SNS	-	158 ms	✓
SNS	200	151 ms	✓
## accessing S3	-	110 ms	✓
S3	404	65.0 ms	!
S3	200	40.0 ms	✓
## accessing DynamoDB	-	72.0 ms	✓
DynamoDB	200	36.0 ms	✓
DynamoDB	200	32.0 ms	✓
## invoking service-c	-	1.2 sec	✓
Lambda	200	1.2 sec	✓
## calling service b	-	550 ms	✓
1g7cabdok5.execute-api.us-east-1.amazonav	200	54[illegible]	✓
lambda-x-ray-demo-dev-service-c AWS::Lambda			
lambda-x-ray-demo-dev-service-c			
lambda-x-ray-demo-dev-service-c A[illegible]			
lambda-x-ray-demo-dev-servic[illegible]			
Initialization			
## publishing[illegible]			
SN[illegible]			



Timeline | Raw data

Name	Res.	Duration	Status
lambda-x-ray-demo-dev-service-a AWS::Lambda			
lambda-x-ray-demo-dev-service-a	200	3.2 sec	✓
lambda-x-ray-demo-dev-service-a AWS::Lambda::Function			
lambda-x-ray-demo-dev-service-a	-	2.1 sec	✓
Initialization	-	529 ms	✓
## publishing to SNS	-	158 ms	✓
SNS	200	151 ms	✓
## accessing S3	-	110 ms	✓
S3	404	65.0 ms	!
S3	200	40.0 ms	✓
## accessing DynamoDB	-	72.0 ms	✓
DynamoDB	200	36.0 ms	✓
DynamoDB	200	32.0 ms	✓
## invoking service-c	-	1.2 sec	✓
Lambda	200	1.2 sec	✓
## calling service b	-	550 ms	✓
1g7cabdok5.execute-api.us-east-1.amazonav	200	548 ms	
lambda-x-ray-demo-dev-service-c AWS::Lambda			
lambda-x-ray-demo-dev-service-c			
lambda-x-ray-demo-dev-service-c AW			
lambda-x-ray-demo-dev-service-c			
Initialization			
## publishing to			
SNS			

0.0ms 500ms 1.0s 1.5s 2.0s 2.5s 3.0s 3.5s

21 (UTC +01:00)

Map legend

avg. 665ms
0.2 t/min
1g7cabdok5.execute-api...
remote

Publish

avg. 52ms
0.4 t/min
S3
AWS::S3

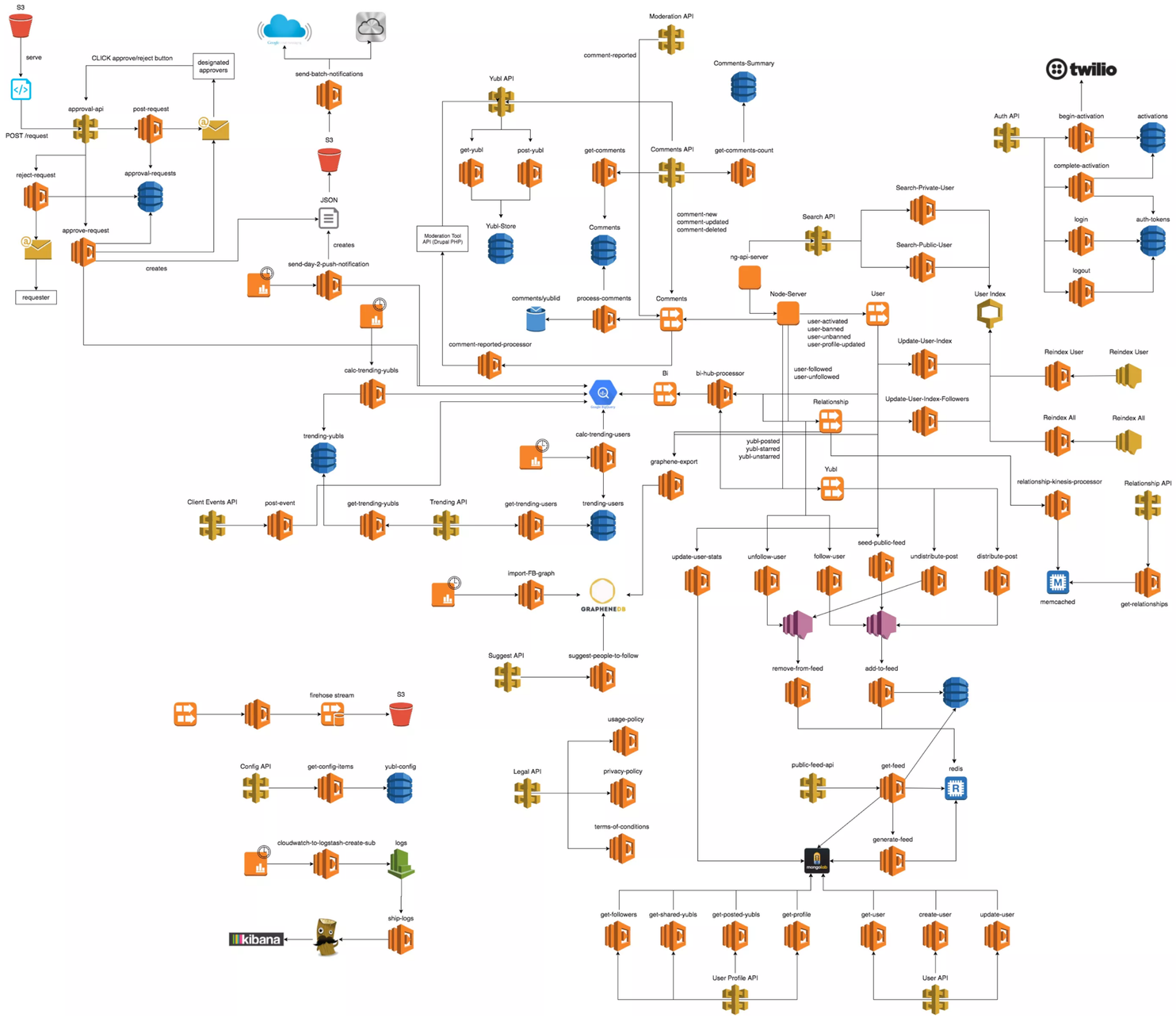
avg. 164ms
0.4 t/min
SNS
AWS::SNS

avg. 744ms
0.2 t/min
lambda-x-ray-demo-dev-service-...
AWS::Lambda

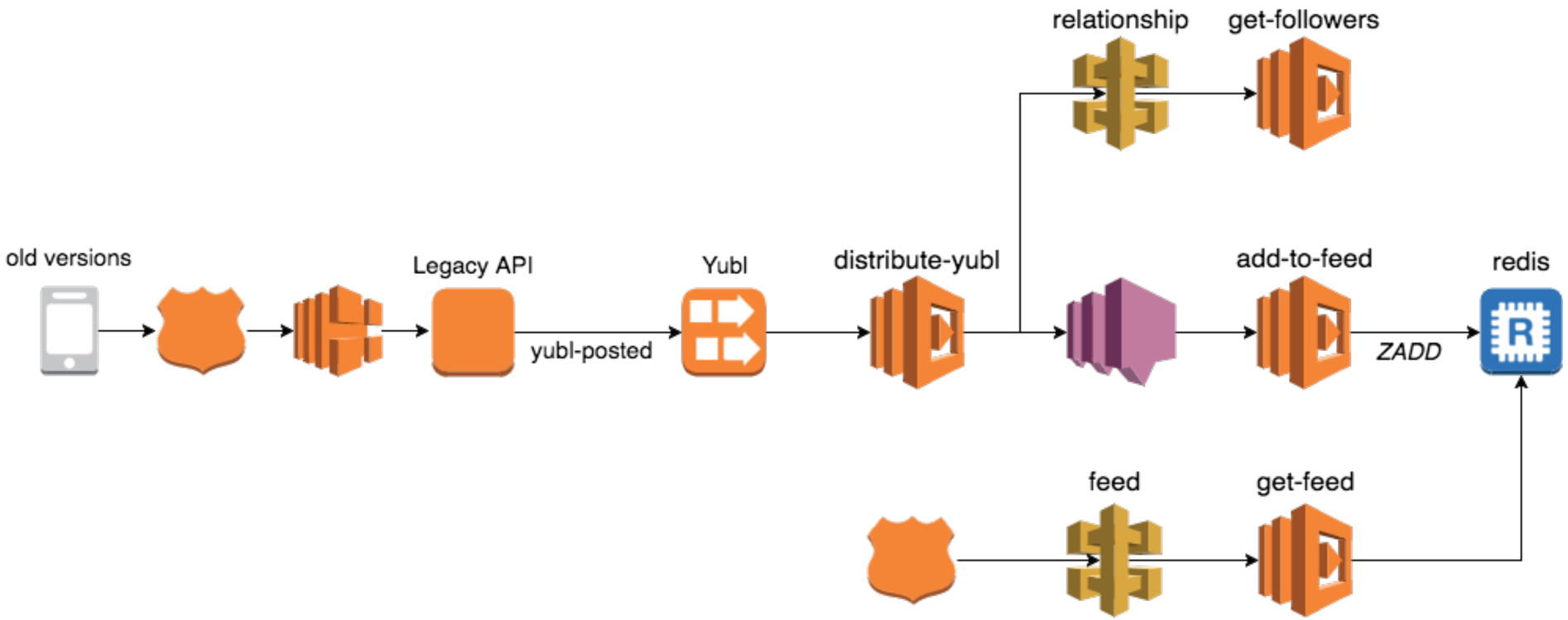
avg. 2ms
0.2 t/min
lambda-x-ray-demo-dev-service-...
AWS::Lambda::Function

static view

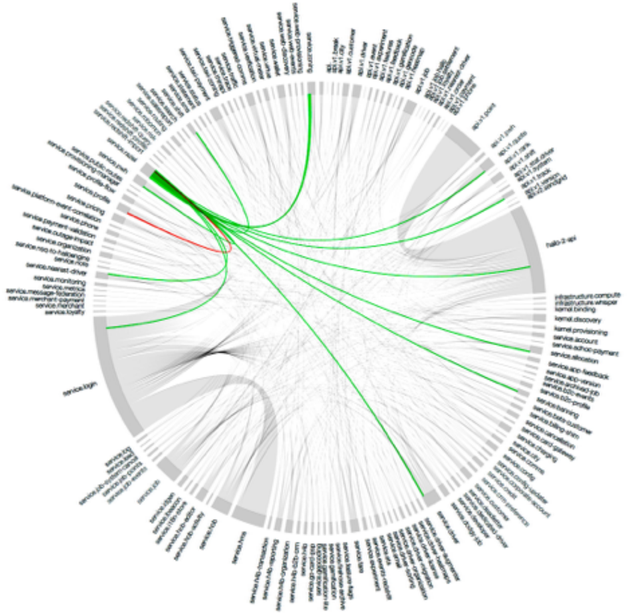
our tools need to do more to help us with understanding & debugging our distributed system, not just what happens inside one function


S3
serve
POST /request
CLICK approve/reject button
designated approvers
approval-api
post-request
reject-request
approval-requests
approve-request
requester
creates
send-batch-notifications
S3
JSON
creates
send-day-2-push-notification
Moderation API
comment-reported
Comments-Summary
Yubl API
get-yubl
post-yubl
Yubl-Store
get-comments
Comments API
get-comments-count
Comments
comment-new
comment-updated
comment-deleted
Moderation Tool API (Drupal PHP)
comments/yublid
process-comments
Comments
comment-reported-processor
calc-trending-yubls
trending-yubls
Client Events API
post-event
get-trending-yubls
Trending API
get-trending-users
trending-users
calc-trending-users
import-FB-graph
GRAPHENEDB
Suggest API
suggest-people-to-follow
firehose stream
S3
Config API
get-config-items
yubl-config
cloudwatch-to-logstash-create-sub
logs
ship-logs
kibana
Search API
Search-Private-User
Search-Public-User
ng-api-server
Node-Server
User
User Index
user-activated
user-banned
user-unbanned
user-profile-updated
user-followed
user-unfollowed
Bi
bi-hub-processor
Relationship
yubl-posted
yubl-starred
yubl-unstarred
graphene-export
Yubl
Update-User-Index
Update-User-Index-Followers
Reindex User
Reindex User
Reindex All
Reindex All
relationship-kinesis-processor
Relationship API
memcached
get-relationships
update-user-stats
unfollow-user
follow-user
seed-public-feed
undistribute-post
distribute-post
remove-from-feed
add-to-feed
usage-policy
Legal API
privacy-policy
terms-of-conditions
public-feed-api
get-feed
redis
generate-feed
mongolab
get-followers
get-shared-yubls
get-posted-yubls
get-profile
get-user
create-user
update-user
User Profile API
User API
twilio
Auth API
begin-activation
activations
complete-activation
login
auth-tokens
logout

“one user action/vertical slice through the system”

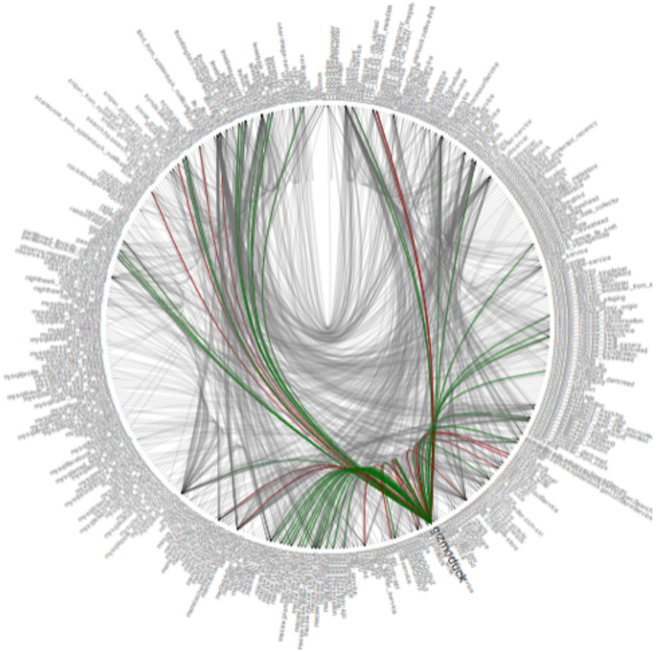


450+ microservices



500+ microservices

500+ microservices

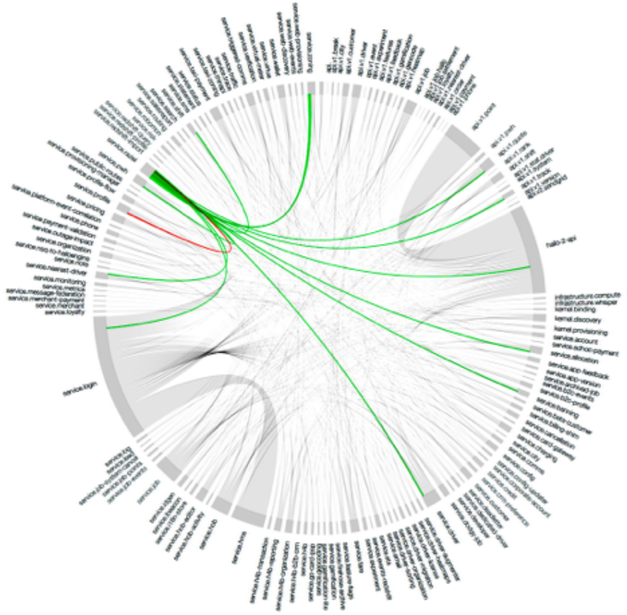


HAILO

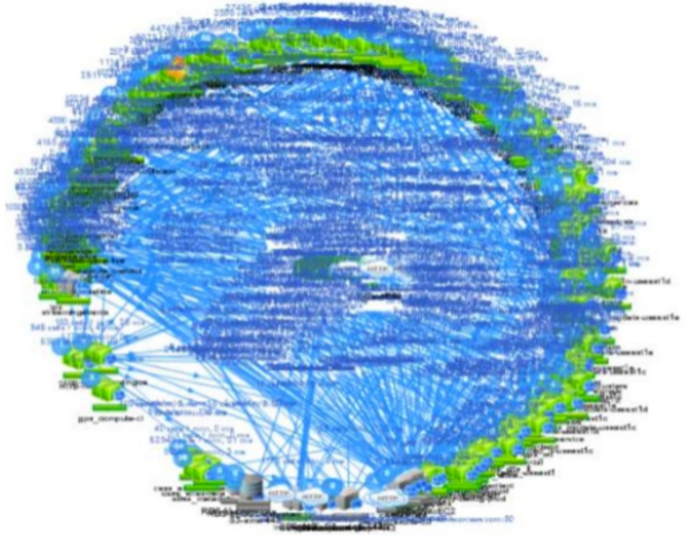
NETFLIX

microservices death stars circa 2015

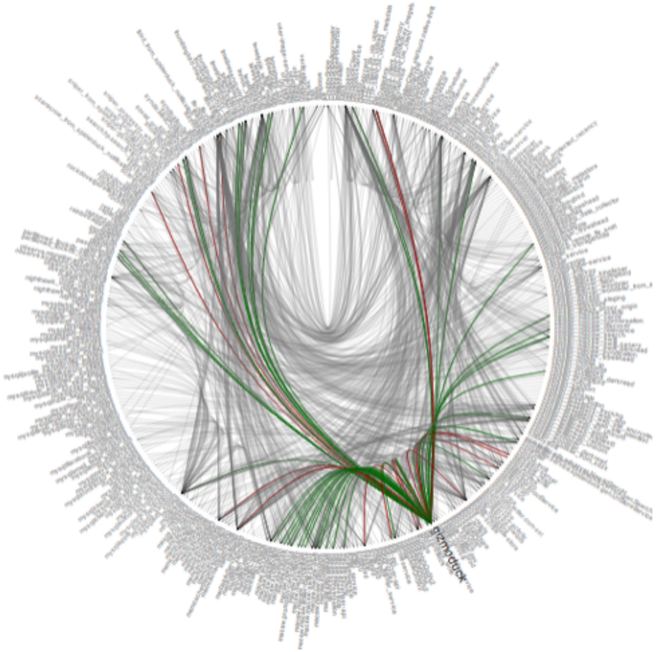
450+ microservices



500+ microservices



500+ microservices



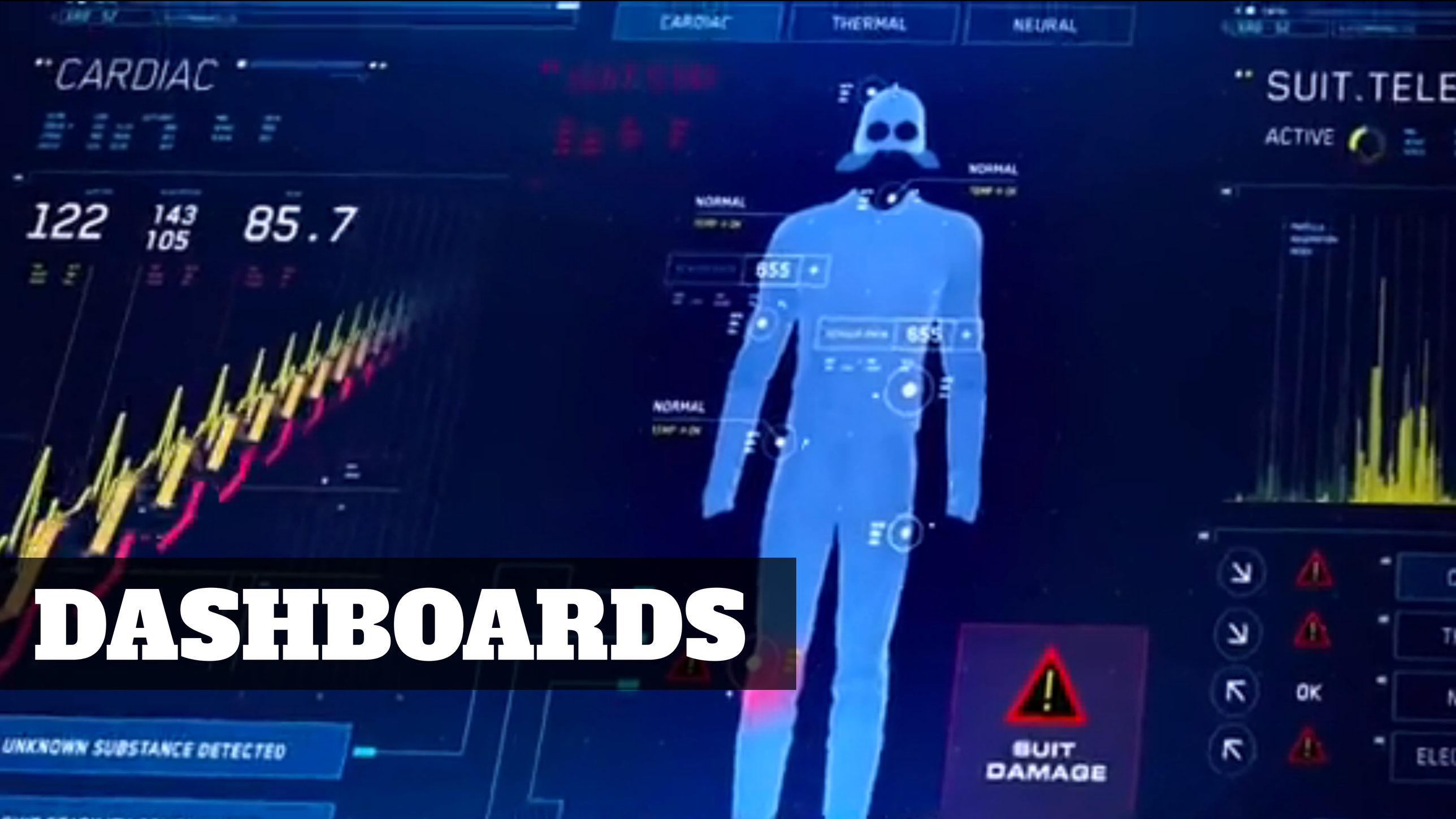
HAIL O

NETFLIX

microservices death stars


OUTATIME
THE FUTURE IS NOW


WARNING: this is part fiction, part inspired by new tools


CARDIAC
THERMAL
NEURAL
CARDIAC
122
143
105
85.7
NORMAL
NORMAL
NORMAL
655
655
SUIT.TELE
ACTIVE
OK
SUIT
DAMAGE
UNKNOWN SUBSTANCE DETECTED

DASHBOARDS

The Trade Desk — Performance Management

Dashboard
Reports
Team
Configure
History

All teams - quarter-to-quarter

+14%

(QTD)

Monthly Trend
(last 3 months)

Last updated 4 days ago

Product team - year-over-year

-6%

(QTD)

Yearly Trend
(last 3 years)

Last updated 3 days ago

Top and bottom 3 teams - percentage of goals (YTD)

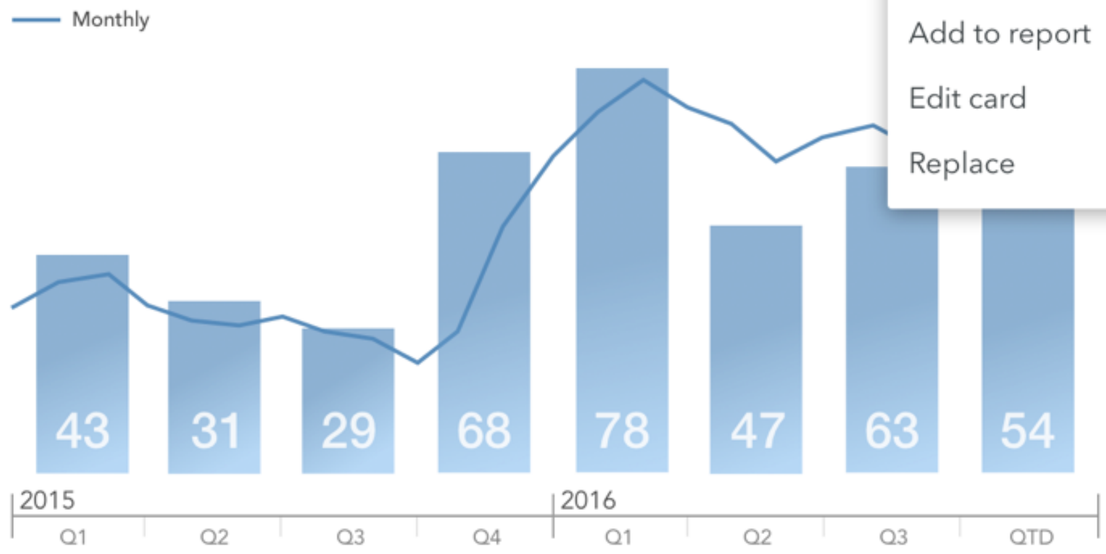
Front-end	91%	Product	65%
UX design	84%	Big Data	53%
QA	72%	Analytics	47%

Last updated 4 days ago

Quarterly team performance - last 2 years (percentage of goal)

PRODUCT UX DESIGN ENGINEERING

See more detail
Add to report
Edit card
Replace



Last updated 2 days ago

Meetings attended by team (QTD)

PRODUCT UX DESIGN ENGINEERING

Last updated 1 day ago

Engineering performance (QTD)

NEW YORK PUNE SYDNEY

Last updated 2 days ago

New Relic - Synthetics

$monitor_name *

New Relic SYNTHETICS

Overview

Success rate 1d

(No data)

Average size 2d

1.42 MiB

Load time 1d

2.49 s

Availability

Last 30 min... 30m

100 %

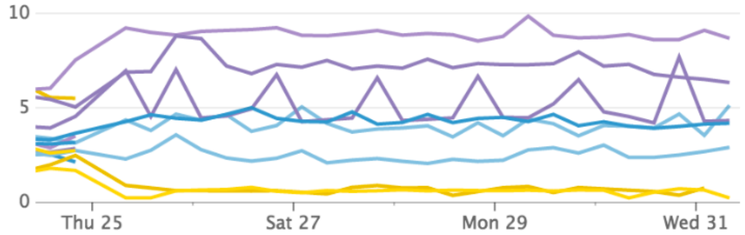
Last 7 days 1w

100 %

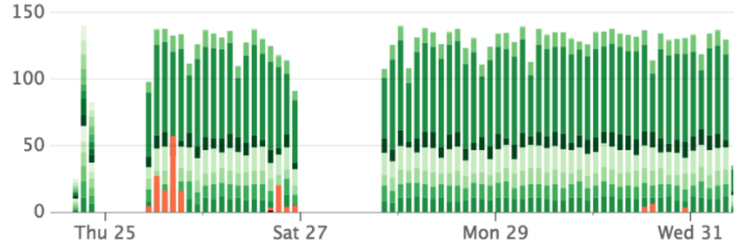
Last 30 days 1mo

100 %

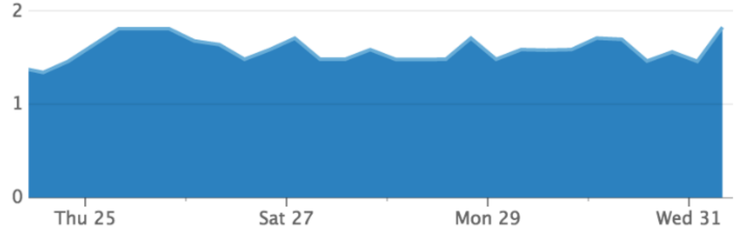
Load time 1w



Checks runs and failures 1w



Average load size 1w

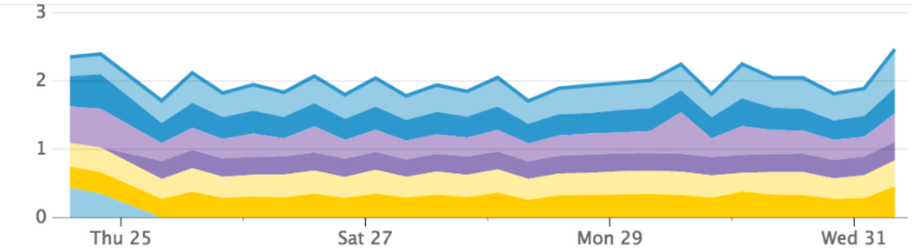


Slowest results 1w

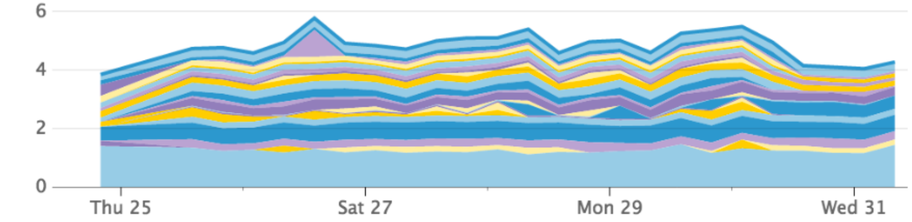
54.05	tokyo_jp
50.84	frankfurt_de
46.53	sydney_au
43.77	fremont_ca_usa
38.95	newark_nj_usa
38.73	singapore_sg
32.3	dallas_tx_usa
28.87	são_paulo_br
18.47	san_francisco_ca_usa
16.35	dublin_ie

Resources

Resources load time 1w



Time spent by third parties 1w



Slowest resources 1w

569	kissmetrics.com
480	datadoghq.com

Network

Network timings 1w

HTTP Response codes 1w

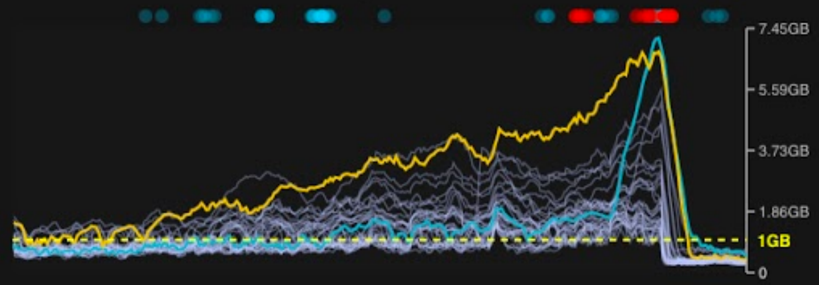
Google Cloud Monitoring powered by stackdriver | Infrastructure | Services | Alerting | Groups | Dashboards | All Resources | gstackdriver | Help | Dan

Overall Data Processing Health

TIME 1h 6h 1d 1w 1m 6w custom

Mon 25 Wed 27 Fri 29 Jan 31February Wed 03 Fri 05

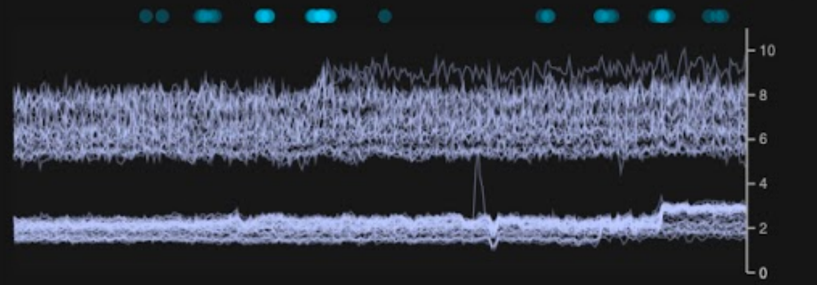
Archiver Buffer Size (Should be < 1GB)



637MB archiver-102:archiver.total_buffered_bytes Feb 4, 2016 1:32:10 PM [UTC -0500]

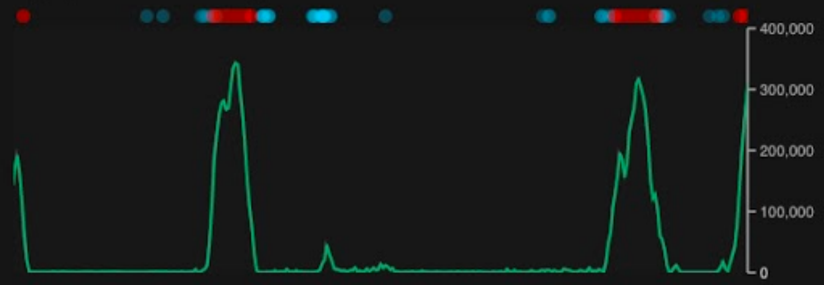
VALUE	NAME	AVERAGE	PEAK	VOLATILITY
637MB	archiver-102:archiver.total_buffered_bytes			
544MB	archiver-155:archiver.total_buffered_bytes			
680MB	archiver-146:archiver.total_buffered_bytes			
758MB	archiver-158:archiver.total_buffered_bytes			

GW Cell 5min Load Avg (Should be < 2)



VALUE	NAME	AVERAGE	PEAK	VOLATILITY
2.4	alerting-indexer-41			
1.8	alerting-indexer-43			
1.8	alerting-indexer-44			
1.9	alerting-indexer-45			

Aggregation Jobs Pending



31 prod.aggregation.packages_outstanding Feb 4, 2016 1:32:10 PM [UTC -0500]

VALUE	NAME	AVERAGE	PEAK	VOLATILITY
31	prod.aggregation.packages_outstanding			

gateway Load Average (past 5m) [Average]

VALUE	NAME	AVERAGE	PEAK	VOLATILITY
5.89	gateway-11			
6.67	gateway-12			

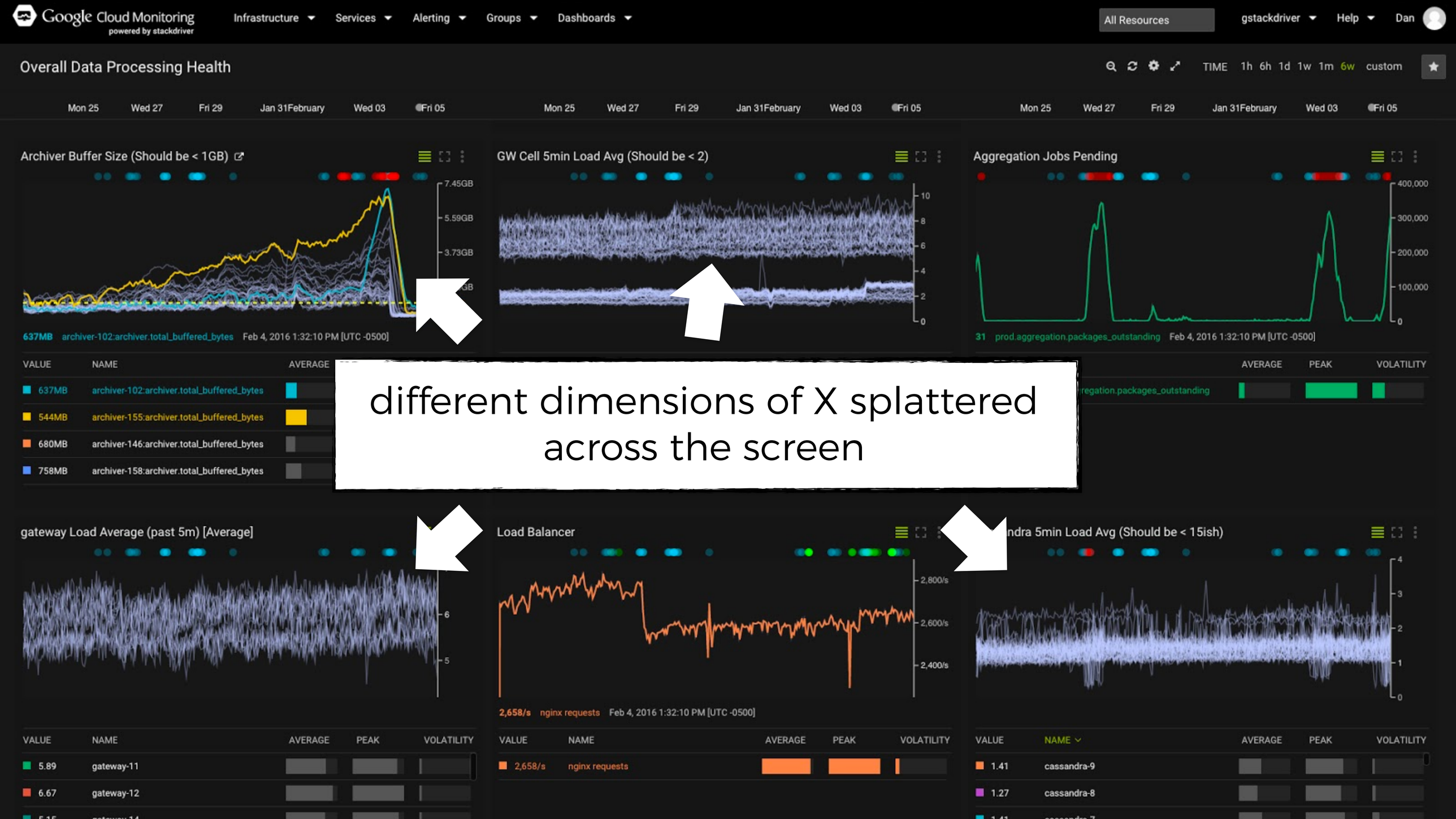
Load Balancer

2,658/s nginx requests Feb 4, 2016 1:32:10 PM [UTC -0500]

VALUE	NAME	AVERAGE	PEAK	VOLATILITY
2,658/s	nginx requests			

Cassandra 5min Load Avg (Should be < 15ish)

VALUE	NAME	AVERAGE	PEAK	VOLATILITY
1.41	cassandra-9			
1.27	cassandra-8			


Google Cloud Monitoring
powered by stackdriver
Infrastructure
Services
Alerting
Groups
Dashboards
All Resources
gstackdriver
Help
Dan
Overall Data Processing Health
TIME 1h 6h 1d 1w 1m 6w custom
Mon 25 Wed 27 Fri 29 Jan 31February Wed 03 Fri 05
Archiver Buffer Size (Should be < 1GB)
7.45GB
5.59GB
3.73GB
637MB archiver-102:archiver.total_buffered_bytes Feb 4, 2016 1:32:10 PM [UTC -0500]
VALUE NAME AVERAGE
637MB archiver-102:archiver.total_buffered_bytes
544MB archiver-155:archiver.total_buffered_bytes
680MB archiver-146:archiver.total_buffered_bytes
758MB archiver-158:archiver.total_buffered_bytes
GW Cell 5min Load Avg (Should be < 2)
10
8
6
4
2
0
Aggregation Jobs Pending
400,000
300,000
200,000
100,000
0
31 prod.aggregation.packages_outstanding Feb 4, 2016 1:32:10 PM [UTC -0500]
AVERAGE PEAK VOLATILITY
different dimensions of X splattered across the screen
gateway Load Average (past 5m) [Average]
6
5
VALUE NAME AVERAGE PEAK VOLATILITY
5.89 gateway-11
6.67 gateway-12
Load Balancer
2,800/s
2,600/s
2,400/s
2,658/s nginx requests Feb 4, 2016 1:32:10 PM [UTC -0500]
VALUE NAME AVERAGE PEAK VOLATILITY
2,658/s nginx requests
ndra 5min Load Avg (Should be < 15ish)
4
3
2
1
0
VALUE NAME AVERAGE PEAK VOLATILITY
1.41 cassandra-9
1.27 cassandra-8

Circuit Breakers Sort: Error then Volume | Alphabetical | Volume | Error | Mean | Median | 90 | 99 | 99.5 Success | Latent | Short-Circuited | Timeout | Rejected | Failure | Error %

VideoMetadataGetEpisode
32,054,367 | 0 | 0.0 %
0
Host: 5,673.3/s
Cluster: 3,205,436.7/s
Circuit Closed

Hosts	565	90th	0ms
Median	0ms	99th	0ms
Mean	0ms	99.5th	0ms

DeviceTypeServiceGetType
226,431 | 0 | 0.0 %
0 | 162
0
Host: 40.1/s
Cluster: 22,659.3/s
Circuit Closed

Hosts	565	90th	0ms
Median	0ms	99th	1ms
Mean	0ms	99.5th	2ms

SubscriberGetAccount
220,152 | 12 | 0.0 %
0 | 24
0
Host: 39.0/s
Cluster: 22,018.8/s
Circuit Closed

Hosts	565	90th	12ms
Median	2ms	99th	60ms
Mean	5ms	99.5th	90ms

IdentityCookieAuth
189,301 | 0 | 0.0 %
0 | 55
0
Host: 33.5/s
Cluster: 18,935.6/s
Circuit Closed

Hosts	565	90th	1ms
Median	0ms	99th	49ms
Mean	1ms	99.5th	75ms

ABCallServiceInternal
188,808 | 0 | 0.0 %
0 | 0
0
Host: 33.4/s
Cluster: 18,880.8/s
Circuit Closed

Hosts	565	90th	17ms
Median	7ms	99th	63ms
Mean	9ms	99.5th	97ms

QTGetQTVGenres
123,538 | 0 | 0.0 %
0 | 0
0
Host: 21.9/s
Cluster: 12,353.8/s
Circuit Closed

Hosts	565	90th	0ms
Median	0ms	99th	1ms
Mean	0ms	99.5th	1ms

CinematchGetPredictions
78,992 | 7 | 0.0 %
0 | 0
0
Host: 14.0/s
Cluster: 7,899.9/s
Circuit Closed

Hosts	565	90th	24ms
Median	3ms	99th	131ms
Mean	12ms	99.5th	372ms

ABTestGetAllocationMap
76,945 | 0 | 0.0 %
0 | 0
0
Host: 13.6/s
Cluster: 7,694.5/s
Circuit Closed

Hosts	565	90th	11ms
Median	1ms	99th	52ms
Mean	4ms	99.5th	81ms

CryptexDecipher
72,614 | 9 | 0.0 %
0 | 0
65
Host: 12.9/s
Cluster: 7,268.8/s
Circuit Closed

Hosts	565	90th	12ms
Median	3ms	99th	118ms
Mean	13ms	99.5th	498ms

CinematchGetMovieRatings
59,788 | 5 | 0.0 %
0 | 0
0
Host: 10.6/s
Cluster: 5,979.3/s
Circuit Closed

Hosts	565	90th	44ms
Median	14ms	99th	145ms
Mean	21ms	99.5th	204ms

CinematchGetNumRatings
57,563 | 0 | 0.0 %
0 | 0
0
Host: 10.2/s
Cluster: 5,756.3/s
Circuit Closed

Hosts	563	90th	0ms
Median	0ms	99th	0ms
Mean	0ms	99.5th	0ms

VideoHistoryGetBookmarks
47,175 | 189 | 1.9 %
243 | 0
0
Host: 8.4/s
Cluster: 4,760.7/s
Circuit (Closed:558,Open:7))

Hosts	565	90th	73ms
Median	31ms	99th	159ms
Mean	37ms	99.5th	207ms

VideoMetadataGetEpisodes
47,398 | 0 | 0.0 %
0
Host: 8.4/s
Cluster: 4,739.8/s
Circuit Closed

Hosts	562	90th	0ms
Median	0ms	99th	0ms
Mean	0ms	99.5th	0ms

GPSGetGroup
40,695 | 0 | 0.0 %
0 | 0
0
Host: 7.2/s
Cluster: 4,069.5/s
Circuit Closed

Hosts	565	90th	145ms
Median	11ms	99th	386ms
Mean	44ms	99.5th	476ms

GeoLookupCommand
27,564 | 0 | 0.1 %
0 | 0
14
Host: 4.9/s
Cluster: 2,757.8/s
Circuit Closed

Hosts	565	90th	9ms
Median	1ms	99th	57ms
Mean	3ms	99.5th	102ms

L10nClientCommand
6,507 | 0 | 0.0 %
0 | 0
0
Host: 1.2/s
Cluster: 650.7/s
Circuit Closed

Hosts	565	90th	15ms
Median	4ms	99th	50ms
Mean	6ms	99.5th	56ms

ViewingHistoryGetViewingHistory
4,700 | 0 | 0.0 %
0 | 0
0
Host: 0.8/s
Cluster: 470.0/s
Circuit Closed

Hosts	564	90th	39ms
Median	12ms	99th	177ms
Mean	19ms	99.5th	177ms

SocialGetTitleContext
4,334 | 8 | 0.3 %
0 | 8
0
Host: 0.8/s
Cluster: 435.0/s
Circuit Closed

Hosts	565	90th	254ms
Median	51ms	99th	1129ms
Mean	110ms	99.5th	1129ms

ABTestGetAllocationForTest
4,218 | 5 | 0.1 %
0 | 0
0
Host: 0.8/s

Hosts			
Median			
Mean			

VideoMetadataGetEpisodeId
2,770 | 0 | 0.0 %
0
Host: 0.5/s

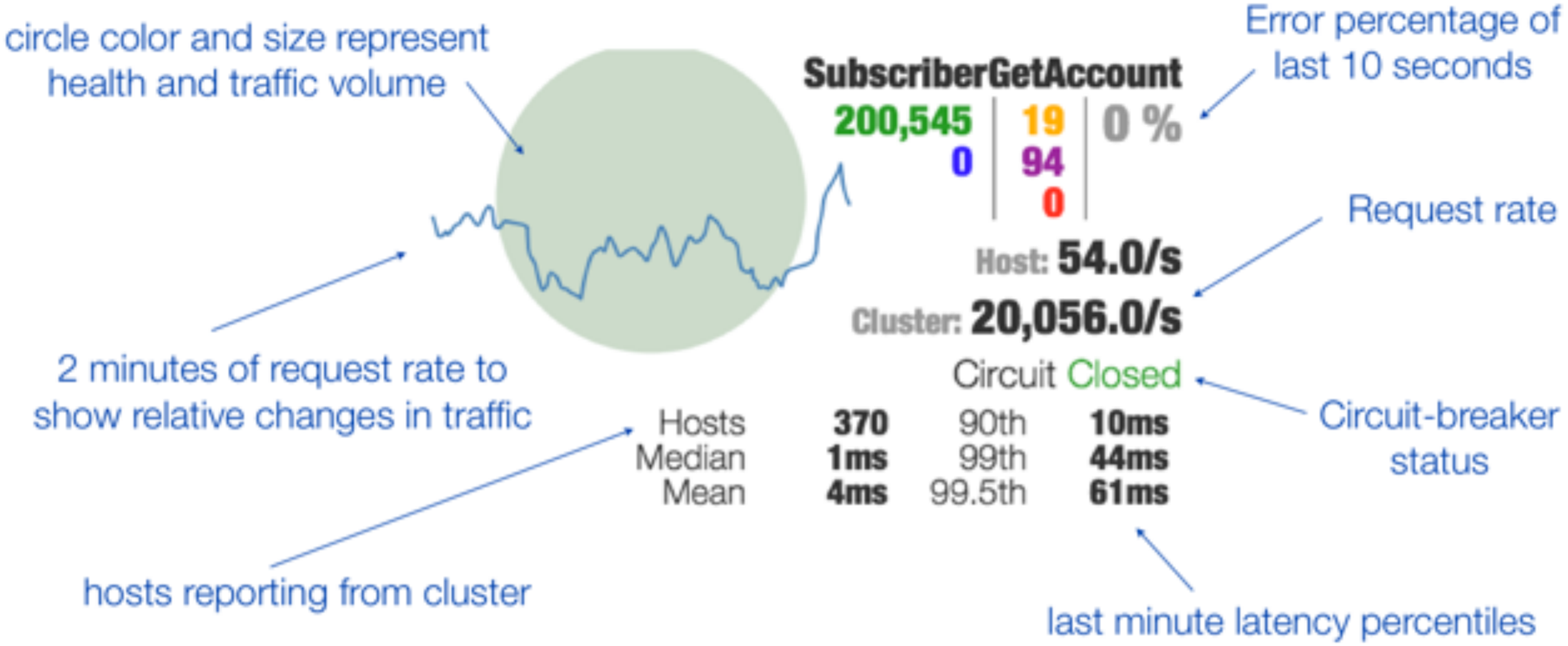
SocialGetHiddenStatus
1,666 | 6 | 0.9 %
0 | 0
0

SearchAutocomplete
393 | 7 | 0.7 %
0 | 0
0

P13NGetPostPlayRecommendations
434 | 1 | 0.2 %
0 | 0
0

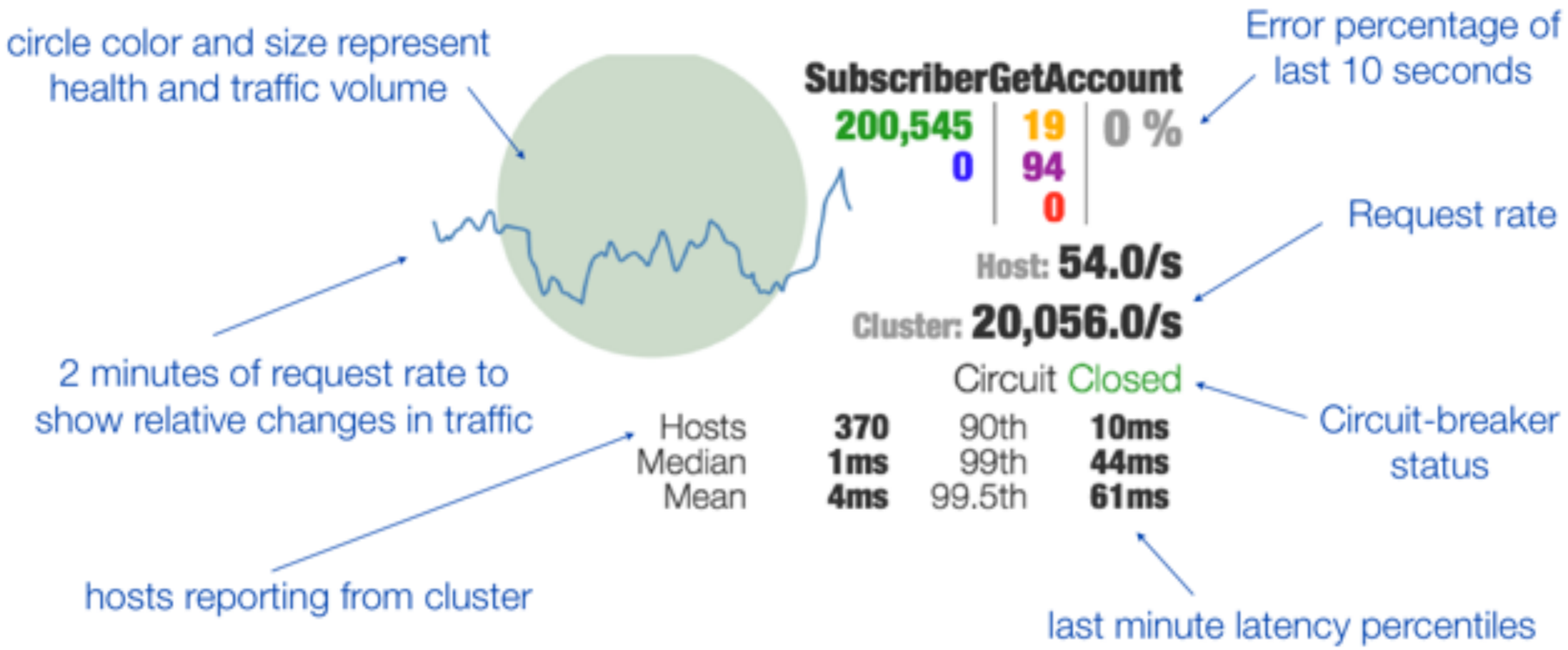
GETUS

HYSTRIX
DEFEND YOUR APP



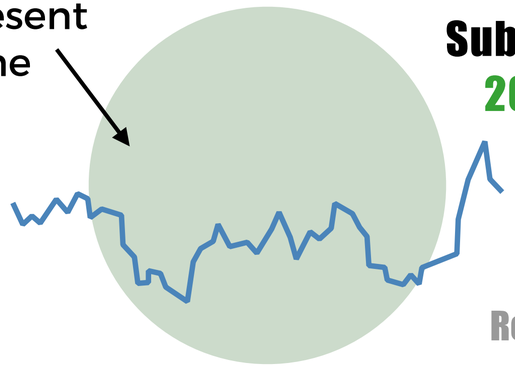
Rolling 10 second counters
with 1 second granularity

Successes	200,545	19	Thread timeouts
Short-circuited (rejected)	0	94	Thread-pool Rejections
		0	Failures/Exceptions



+ cold starts
+ throttled invocations
+ concurrent executions
+ estimated cost ($)

circle colour and size represent health and traffic volume



2 minutes of request rate to show relative changes in traffic

SubscriberGetAccount

200,545	19	0 %
0	94	0 %
	0	

Est Cost: **$54.0/s**

Req Rate: **20,056.0/s**

Error percentage of last 10 seconds

Cold start percentage last 10 seconds

Estimated cost

Request rate

Concurrency	370	90th	10ms
Median	1ms	99th	44ms
Mean	4ms	99.5th	61ms

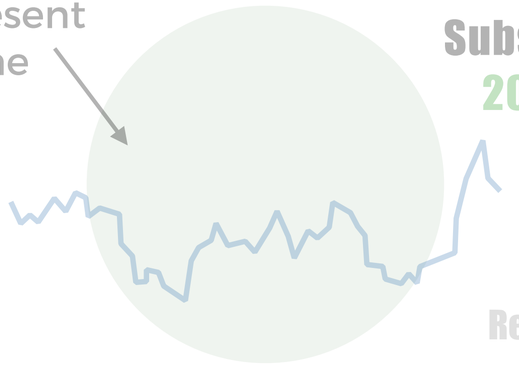
no. of concurrent executions of this function

last minute latency percentiles

Rolling 10 second counters with 1 second granularity

Successes	200,545	19	Timeouts
Cold starts	0	94	Errors
		0	Throttled Invocations

circle colour and size represent health and traffic volume



2 minutes of request rate to show relative changes in traffic

SubscriberGetAccount

200,545	19	0 %
0	94	0 %
	0	

Est Cost: **$54.0/s**

Req Rate: **20,056.0/s**

Error percentage of last 10 seconds

Cold start percentage last 10 seconds

Estimated cost

Request rate

Concurrency	370	90th	10ms
Median	1ms	99th	44ms
Mean	4ms	99.5th	61ms

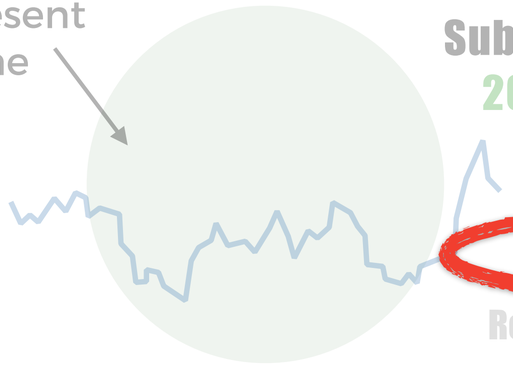
no. of concurrent executions of this function

last minute latency percentiles

Rolling 10 second counters with 1 second granularity

Successes	200,545	19	Timeouts
Cold starts	0	94	Errors
		0	Throttled Invocations

circle colour and size represent health and traffic volume



2 minutes of request rate to show relative changes in traffic

SubscriberGetAccount

200,545	19	0 %
0	94	0 %
	0	

Est Cost: **$54.0/s**

Req Rate: **20,056.0/s**

Error percentage of last 10 seconds

Cold start percentage last 10 seconds

Estimated cost

Request rate

Concurrency	370	90th	10ms
Median	1ms	99th	44ms
Mean	4ms	99.5th	61ms

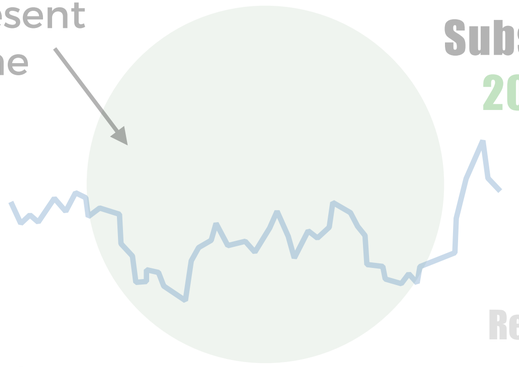
no. of concurrent executions of this function

last minute latency percentiles

Rolling 10 second counters with 1 second granularity

Successes	200,545	19	Timeouts
Cold starts	0	94	Errors
		0	Throttled Invocations

circle colour and size represent health and traffic volume



2 minutes of request rate to show relative changes in traffic

SubscriberGetAccount

200,545	19	0 %
0	94	0 %
	0	

Est Cost: **$54.0/s**

Req Rate: **20,056.0/s**

Error percentage of last 10 seconds

Cold start percentage last 10 seconds

Estimated cost

Request rate

Concurrency	370	90th	10ms
Median	1ms	99th	44ms
Mean	4ms	99.5th	61ms

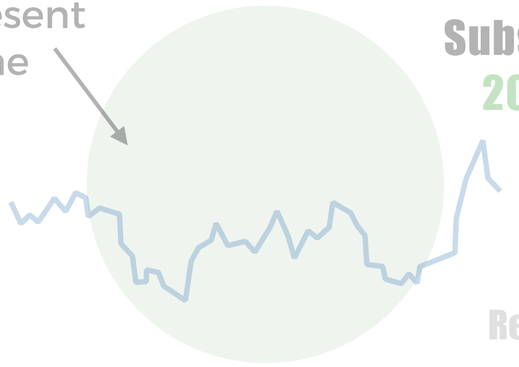
no. of concurrent executions of this function

last minute latency percentiles

Rolling 10 second counters with 1 second granularity

Successes	200,545	19	Timeouts
Cold starts	0	94	Errors
		0	Throttled Invocations

circle colour and size represent health and traffic volume



2 minutes of request rate to show relative changes in traffic

SubscriberGetAccount

200,545	19	0	%
0	94	0	%
	0		

Est Cost: **$54.0/s**

Req Rate: **20,056.0/s**

Error percentage of last 10 seconds

Cold start percentage last 10 seconds

Estimated cost

Request rate

Concurrency	370	90th	10ms
Median	1ms	99th	44ms
Mean	4ms	99.5th	61ms

no. of concurrent executions of this function

last minute latency percentiles

Rolling 10 second counters with 1 second granularity

Successes	200,545	19	Timeouts
Cold starts	0	94	Errors
		0	Throttled Invocations

Circuit Breakers Sort: Error then Volume | Alphabetical | Volume | Error | Mean | Median | 90 | 99 | 99.5 Success | Latent | Short-Circuited | Timeout | Rejected | Failure | Error %

VideoMetadataGetEpisode
32,054,367 | 0 | 0.0 %
0
Host: 5,673.3/s
Cluster: 3,205,436.7/s
Circuit Closed

Hosts	565	90th	0ms
Median	0ms	99th	0ms
Mean	0ms	99.5th	0ms

DeviceTypeServiceGetType
226,431 | 0 | 0.0 %
0 | 162
0
Host: 40.1/s
Cluster: 22,659.3/s
Circuit Closed

Hosts	565	90th	0ms
Median	0ms	99th	1ms
Mean	0ms	99.5th	2ms

SubscriberGetAccount
220,152 | 12 | 0.0 %
0 | 24
0
Host: 39.0/s
Cluster: 22,018.8/s
Circuit Closed

Hosts	565	90th	12ms
Median	2ms	99th	60ms
Mean	5ms	99.5th	90ms

IdentityCookieAuth
189,301 | 0 | 0.0 %
0 | 55
0
Host: 33.5/s
Cluster: 18,935.6/s
Circuit Closed

Hosts	565	90th	1ms
Median	0ms	99th	49ms
Mean	1ms	99.5th	75ms

ABCallServiceInternal
188,808 | 0 | 0.0 %
0 | 0
0
Host: 33.4/s
Cluster: 18,880.8/s
Circuit Closed

Hosts	565	90th	17ms
Median	7ms	99th	63ms
Mean	9ms	99.5th	97ms

QTGetQTVGenres
123,538 | 0 | 0.0 %
0 | 0
0
Host: 21.9/s
Cluster: 12,353.8/s
Circuit Closed

Hosts	565	90th	0ms
Median	0ms	99th	1ms
Mean	0ms	99.5th	1ms

CinematchGetPredictions
78,992 | 7 | 0.0 %
0 | 0
0
Host: 14.0/s
Cluster: 7,899.9/s
Circuit Closed

Hosts	565	90th	24ms
Median	3ms	99th	131ms
Mean	12ms	99.5th	372ms

ABTestGetAllocationMap
76,945 | 0 | 0.0 %
0 | 0
0
Host: 13.6/s
Cluster: 7,694.5/s
Circuit Closed

Hosts	565	90th	11ms
Median	1ms	99th	52ms
Mean	4ms	99.5th	81ms

CryptexDecipher
72,614 | 9 | 0.0 %
0 | 0
65
Host: 12.9/s
Cluster: 7,268.8/s
Circuit Closed

Hosts	565	90th	12ms
Median	3ms	99th	118ms
Mean	13ms	99.5th	498ms

CinematchGetMovieRatings
59,788 | 5 | 0.0 %
0 | 0
0
Host: 10.6/s
Cluster: 5,979.3/s
Circuit Closed

Hosts	565	90th	44ms
Median	14ms	99th	145ms
Mean	21ms	99.5th	204ms

CinematchGetNumRatings
57,563 | 0 | 0.0 %
0 | 0
0
Host: 10.2/s
Cluster: 5,756.3/s
Circuit Closed

Hosts	563	90th	0ms
Median	0ms	99th	0ms
Mean	0ms	99.5th	0ms

VideoHistoryGetBookmarks
47,175 | 189 | 1.9 %
243 | 0
0
Host: 8.4/s
Cluster: 4,760.7/s
Circuit (Closed:558,Open:7))

Hosts	565	90th	73ms
Median	31ms	99th	159ms
Mean	37ms	99.5th	207ms

VideoMetadataGetEpisodes
47,398 | 0 | 0.0 %
0
Host: 8.4/s
Cluster: 4,739.8/s
Circuit Closed

Hosts	562	90th	0ms
Median	0ms	99th	0ms
Mean	0ms	99.5th	0ms

GPSGetGroup
40,695 | 0 | 0.0 %
0 | 0
0
Host: 7.2/s
Cluster: 4,069.5/s
Circuit Closed

Hosts	565	90th	145ms
Median	11ms	99th	386ms
Mean	44ms	99.5th	476ms

GeoLookupCommand
27,564 | 0 | 0.1 %
0 | 0
14
Host: 4.9/s
Cluster: 2,757.8/s
Circuit Closed

Hosts	565	90th	9ms
Median	1ms	99th	57ms
Mean	3ms	99.5th	102ms

L10nClientCommand
6,507 | 0 | 0.0 %
0 | 0
0
Host: 1.2/s
Cluster: 650.7/s
Circuit Closed

Hosts	565	90th	15ms
Median	4ms	99th	50ms
Mean	6ms	99.5th	56ms

ViewingHistoryGetViewingHistory
4,700 | 0 | 0.0 %
0 | 0
0
Host: 0.8/s
Cluster: 470.0/s
Circuit Closed

Hosts	564	90th	39ms
Median	12ms	99th	177ms
Mean	19ms	99.5th	177ms

SocialGetTitleContext
4,334 | 8 | 0.3 %
0 | 8
0
Host: 0.8/s
Cluster: 435.0/s
Circuit Closed

Hosts	565	90th	254ms
Median	51ms	99th	1129ms
Mean	110ms	99.5th	1129ms

ABTestGetAllocationForTest
4,218 | 5 | 0.1 %
0 | 0
0
Host: 0.8/s

Hosts			
Median			
Mean			

VideoMetadataGetEpisodeId
2,770 | 0 | 0.0 %
0
Host: 0.5/s

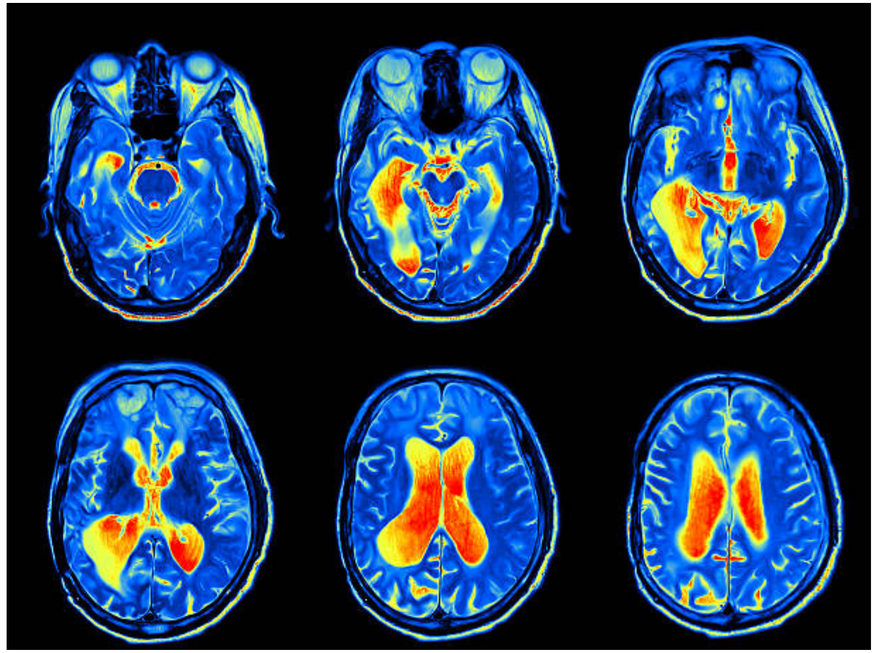
SocialGetHiddenStatus
1,666 | 6 | 0.9 %
0 | 0
0

SearchAutocomplete
393 | 7 | 0.7 %
0 | 0
0

P13NGetPostPlayRecommendations
434 | 1 | 0.2 %
0 | 0
0

GETUS

HYSTRIX
DEFEND YOUR APP



VIZCERAL

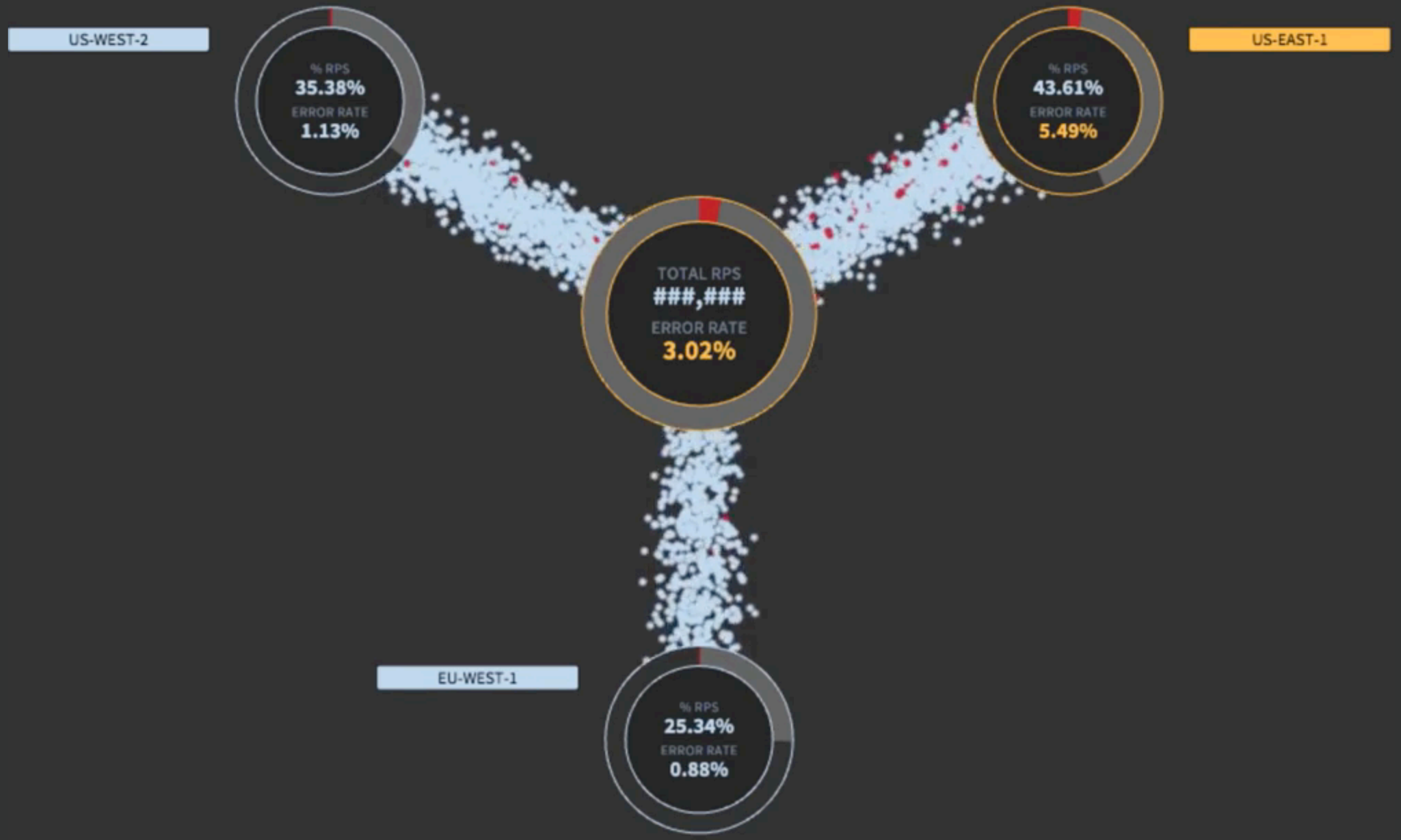
Service Traffic Map / us-east-1

200 services / 116 filtered (show) Locate Service Filters Display

VIZCERAL

Service Traffic Map

Filters Display

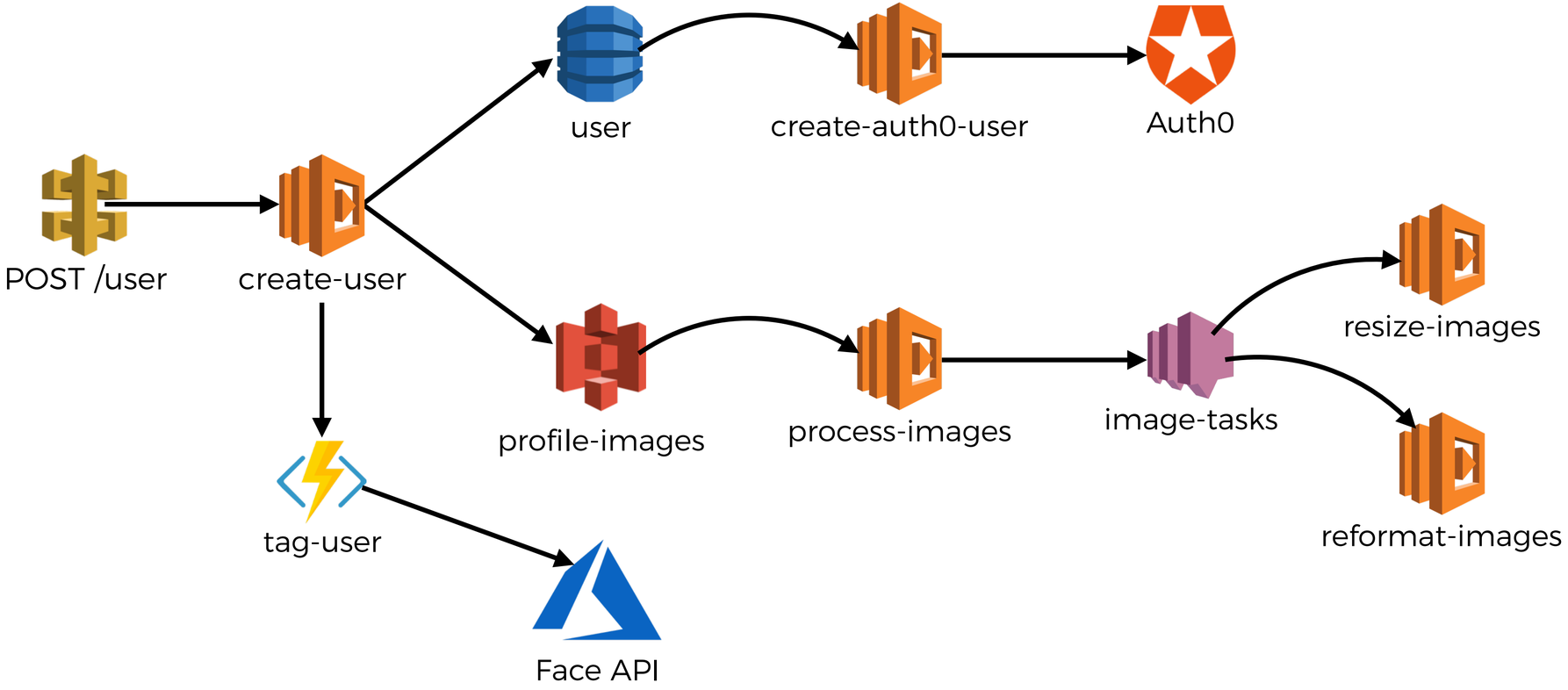


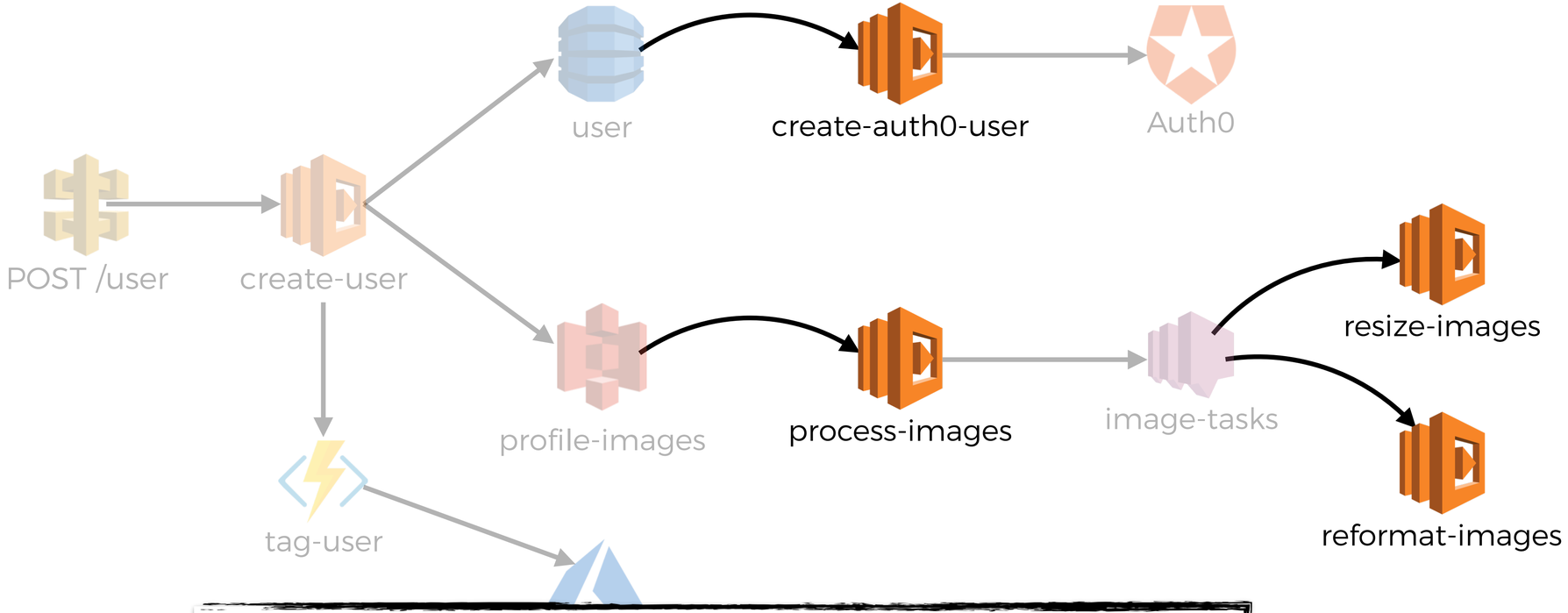

birds-eye view of our system as it lives and breathes

VIZCERAL

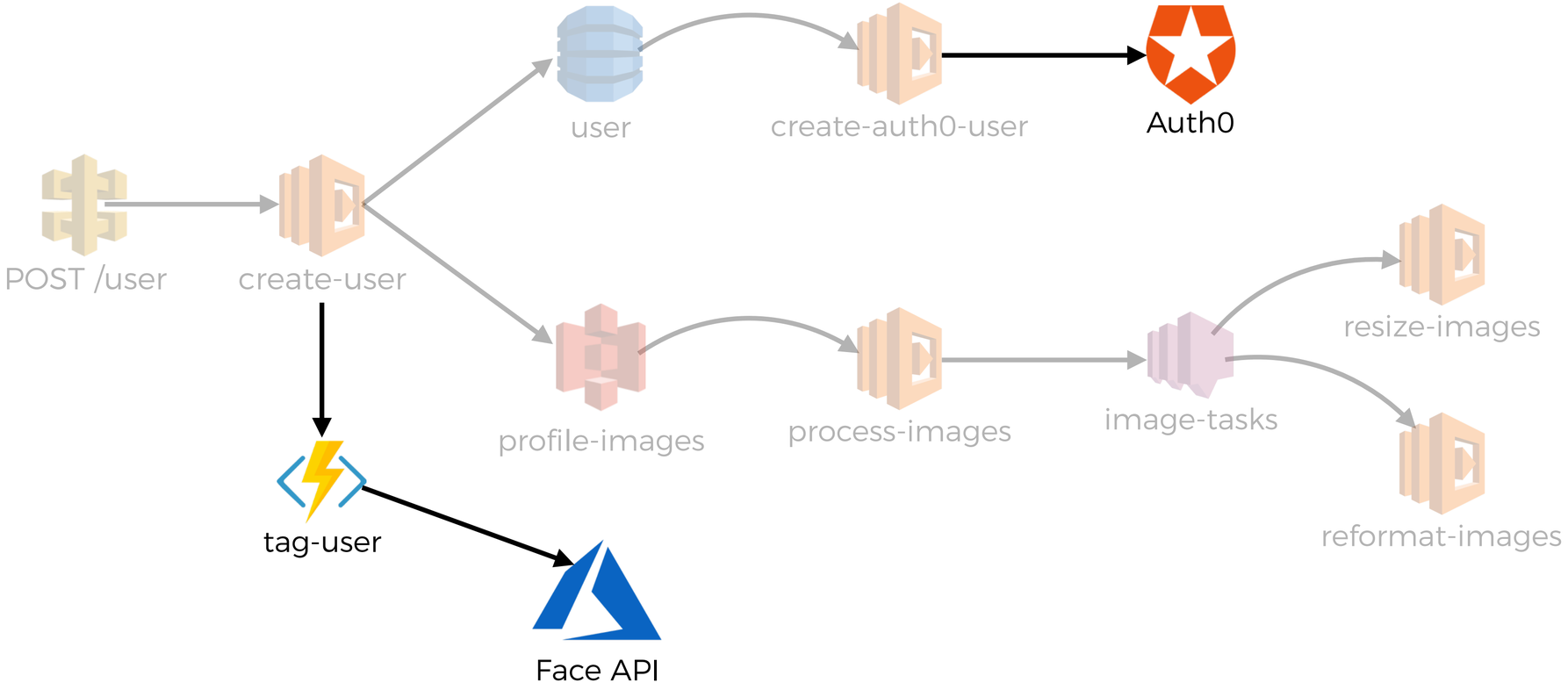
Service Traffic Map / us-east-1

200 services / 116 filtered (show) Locate Service Filters Display

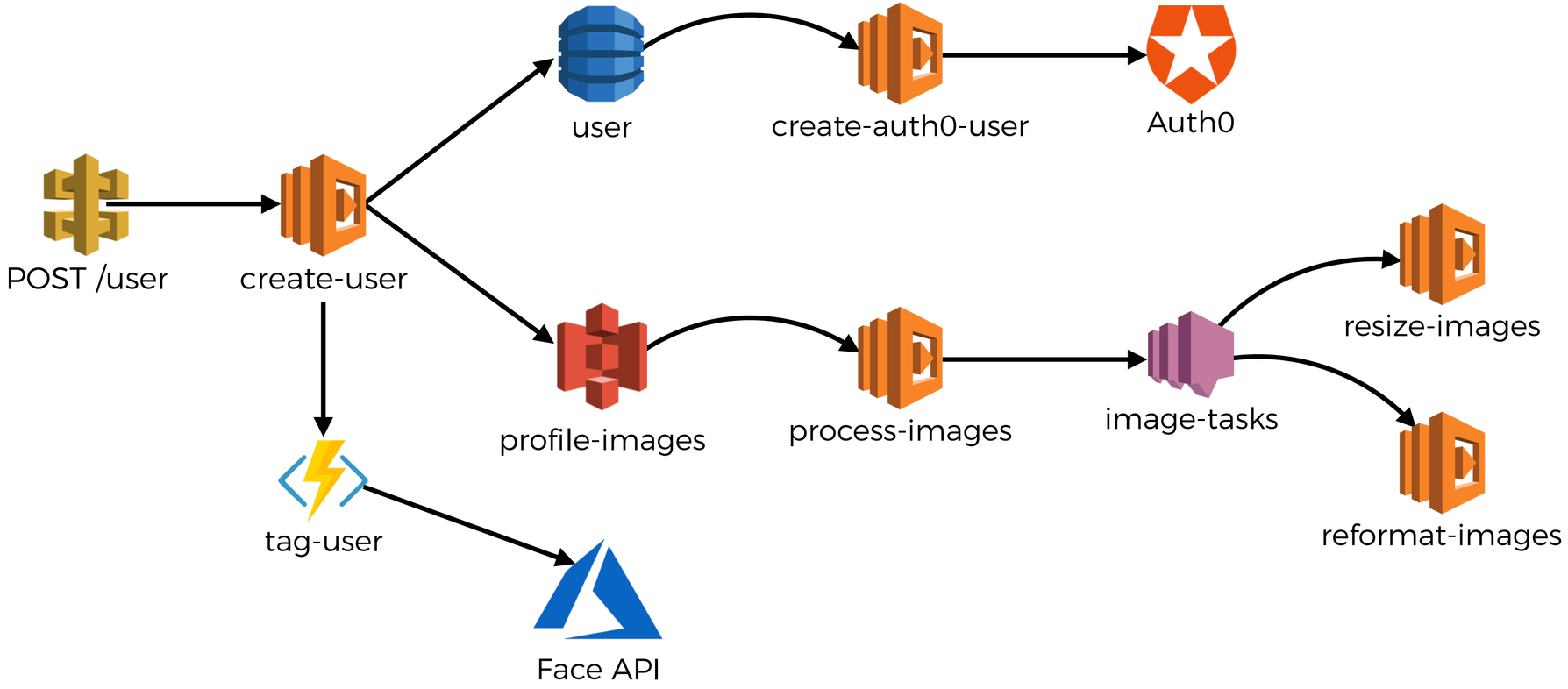

user
create-auth0-user
Auth0
POST /user
create-user
profile-images
process-images
image-tasks
resize-images
reformat-images
tag-user
Face API



trace **async** invocations

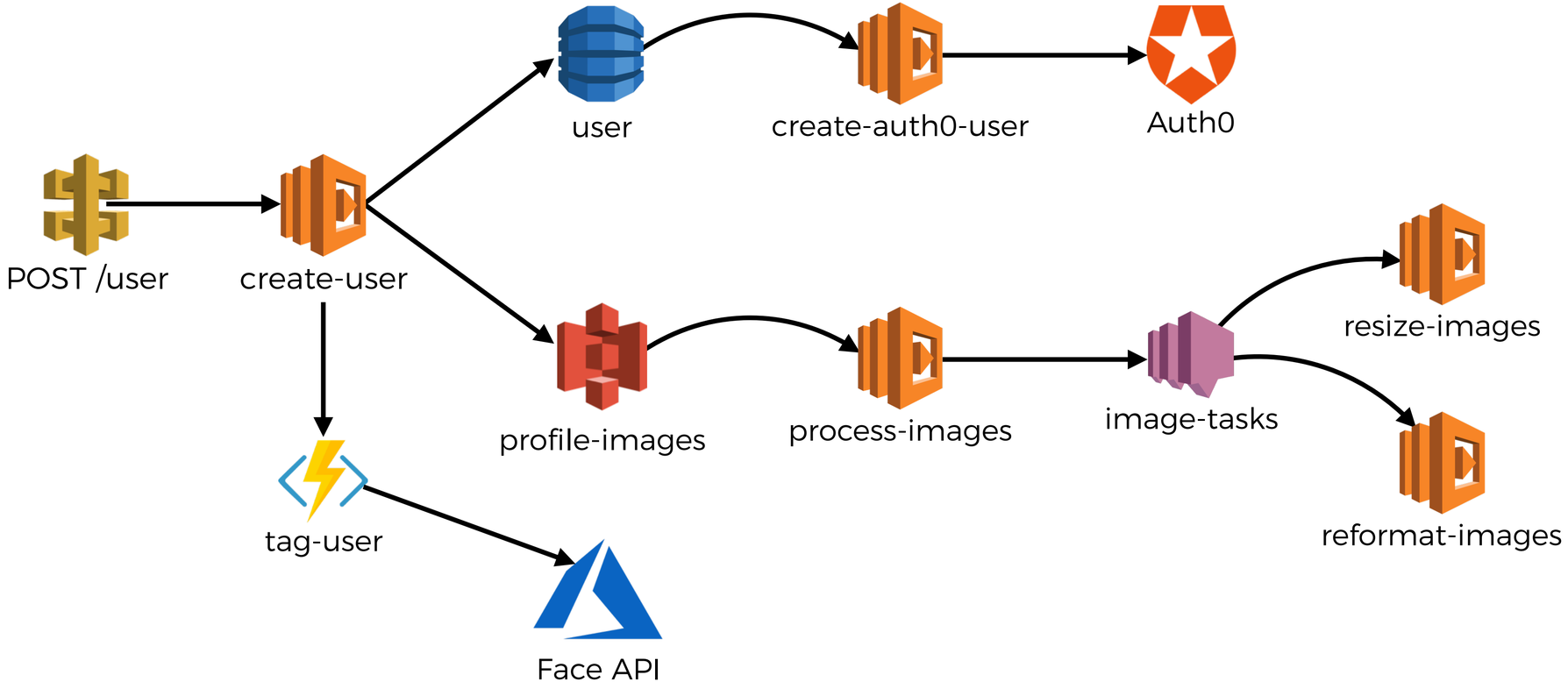


trace **non-AWS** resources



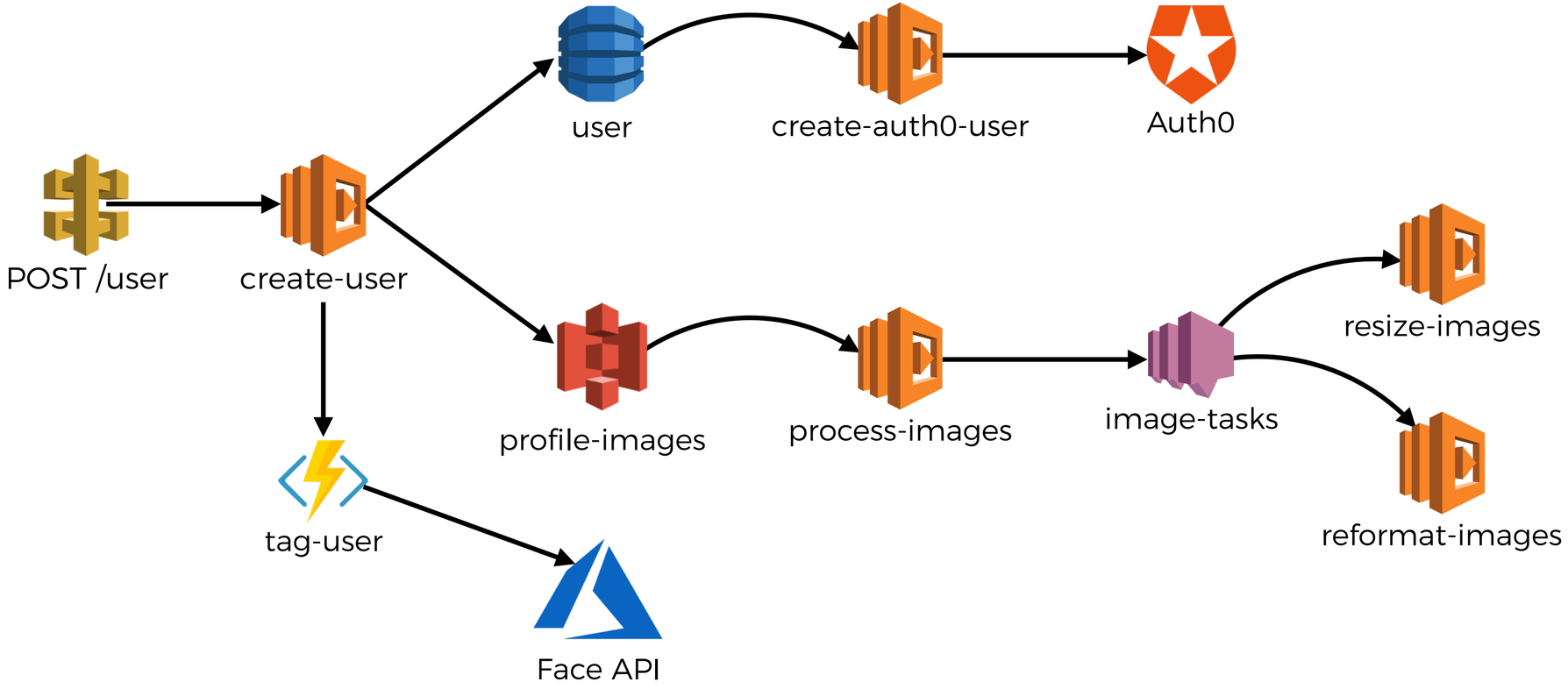
Logs

timestamp	component	level	message
2018/01/25 20:51:23.188	POST /user	debug	incoming request...
2018/01/25 20:51:23.201	create-user	debug	saving user [theburningmonk] in the [user] table...
2018/01/25 20:51:23.215	create-user	debug	saved user [theburningmonk] in the [user] table
2018/01/25 20:51:23.521	tag-user	debug	tagging user [theburningmonk] with Azure Face API...



Logs

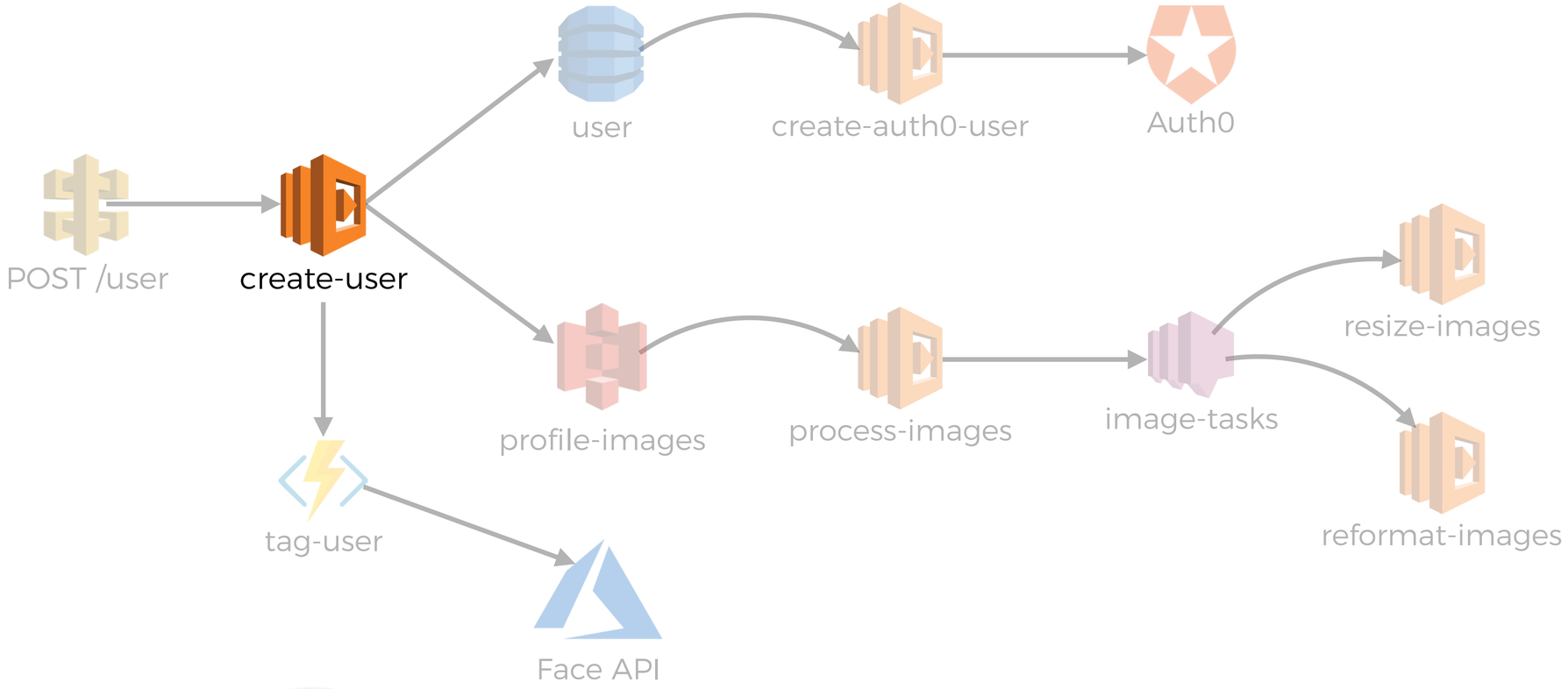
timestamp	component	level	message
2018/01/25 20:51:23.188	POST /user	debug	incoming request...
2018/01/25 20:51:23.201	create-user	debug	saving user [theburningmonk] in the [user] table...
2018/01/25 20:51:23.215	create-user	debug	saved user [theburningmonk] in the [user] table
2018/01/25 20:51:23.521	tag-user	debug	tagging user [theburningmonk] with Azure Face API...



Logs

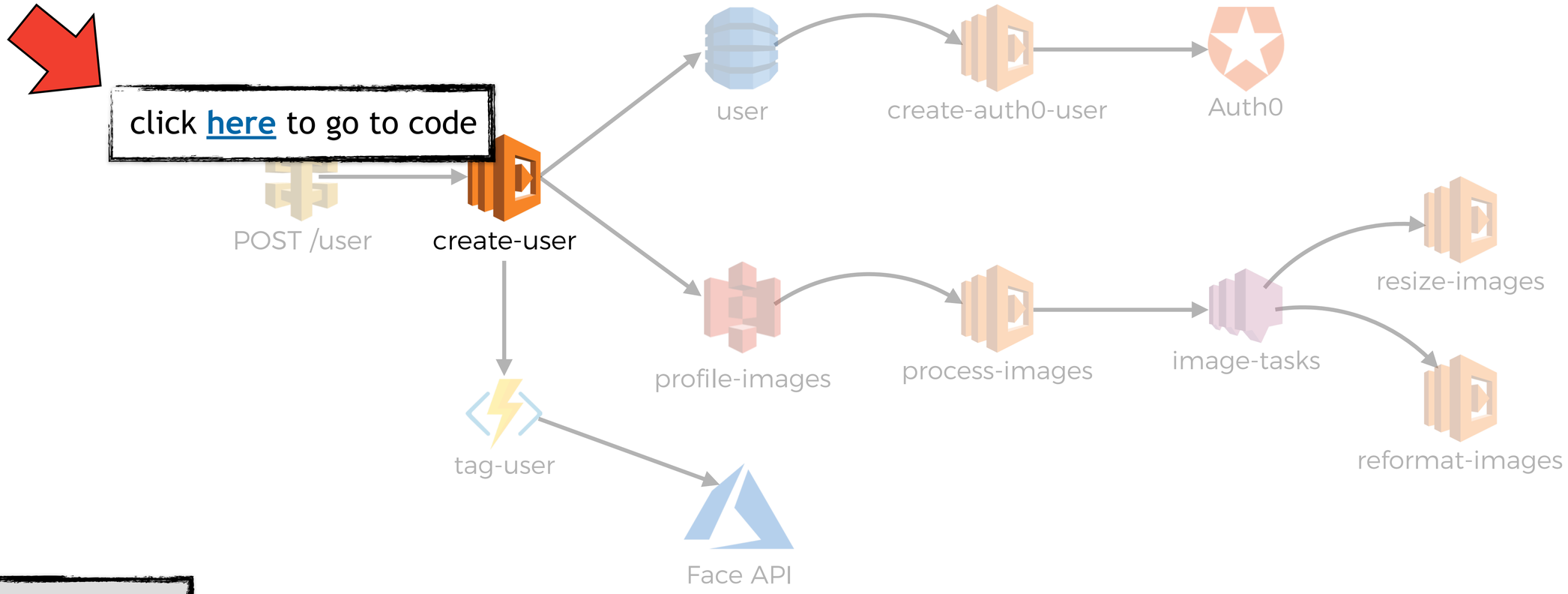
timestamp	component	level	message
2018/01/25 20:51:23.188	POST /user	debug	incoming request...

request-id	0ae4ba5d-dab1-4f9e-9de7-eace27ebfbc2
start-time	2018/01/25 20:51:23.188
method	POST



Logs

timestamp	component	level	message
2018/01/25 20:51:23.201	create-user	debug	saving user [theburningmonk] in the [user] table...
2018/01/25 20:51:23.215	create-user	debug	saved user [theburningmonk] in the [user] table
2018/01/25 20:51:23.585	create-user	debug	tagged user [theburningmonk] with Azure Face API...
2018/01/25 20:51:23.587	create-user	debug	uploading profile image...



Logs

timestamp	component	level	message
2018/01/25 20:51:23.201	create-user	debug	saving user [theburningmonk] in the [user] table...
2018/01/25 20:51:23.215	create-user	debug	saved user [theburningmonk] in the [user] table
2018/01/25 20:51:23.585	create-user	debug	tagged user [theburningmonk] with Azure Face API...
2018/01/25 20:51:23.587	create-user	debug	uploading profile image...

AWS Cloud9 File Edit Find View Goto Tools Window

Environment

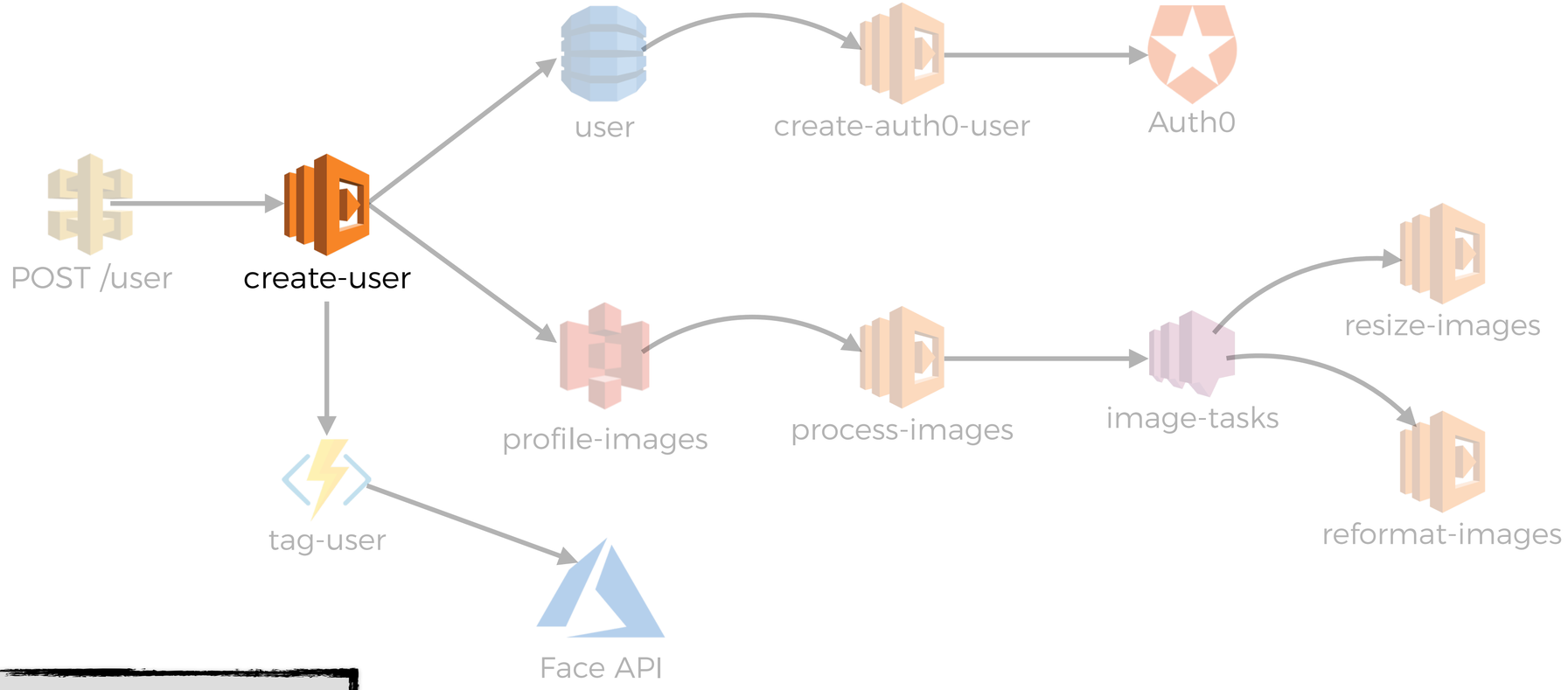
big-mouth-dev-get-restaura

- functions
- build.sh
- buildspec.yml
- package-lock.json
- package.json
- README.md
- seed-restaurants.js
- template.yml

get-restaurants.j

```javascript
'use strict';

const co = require('co');
const AWS = require('aws-sdk');
const dynamodb = new AWS.DynamoDB.DocumentClient();

const defaultResults = process.env.defaultResults || 8;
const tableName = process.env.restaurants_table;

function* getRestaurants(count) {
  let req = {
    TableName: tableName,
    Limit: count
  };

  let resp = yield dynamodb.scan(req).promise();
  return resp.Items;
}

module.exports.handler = co.wrap(function* (event, context, cb) {
  let restaurants = yield getRestaurants(defaultResults);
  let response = {
    statusCode: 200,
    body: JSON.stringify(restaurants)
  }

  cb(null, response);
});
```

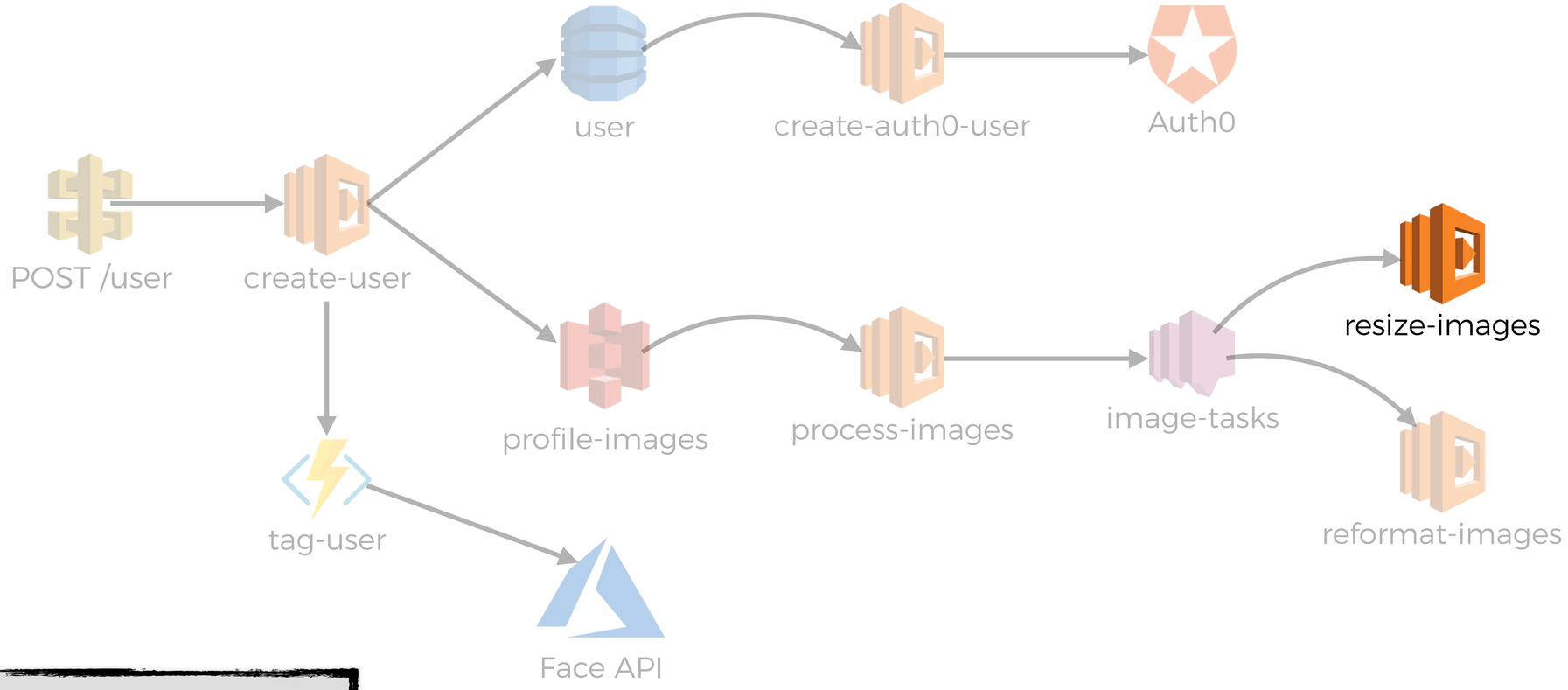
1:1 JavaScript Spaces: 2



Logs | **Input/Output**

input	output

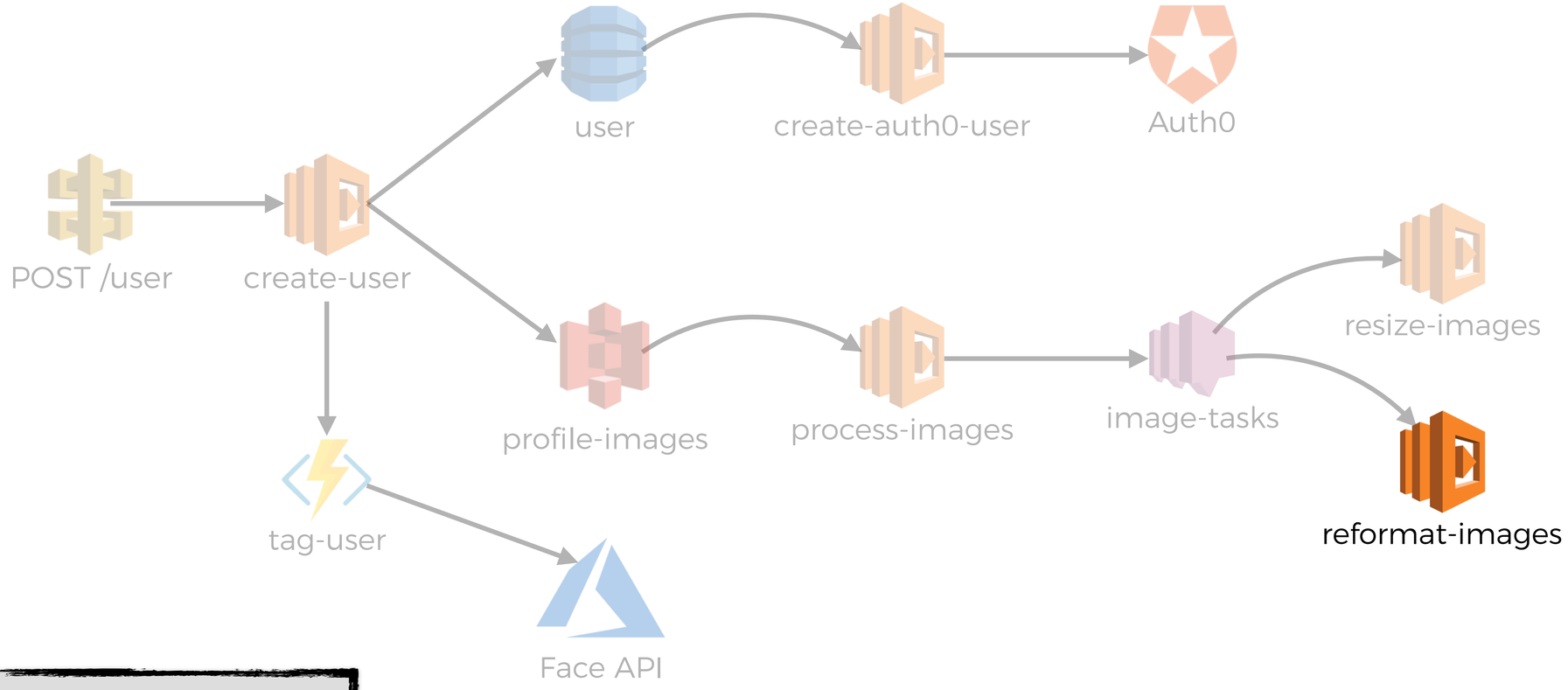
{
"body": "{ \"username\":\"theburningmonk\"}",
"resource": "/user",
"requestContext": {
"resourceId": "123456",
"apiId": “1234567890",
"resourcePath": "/user",
	{
	"statusCode": 200
	}



Logs | **Input/Output**

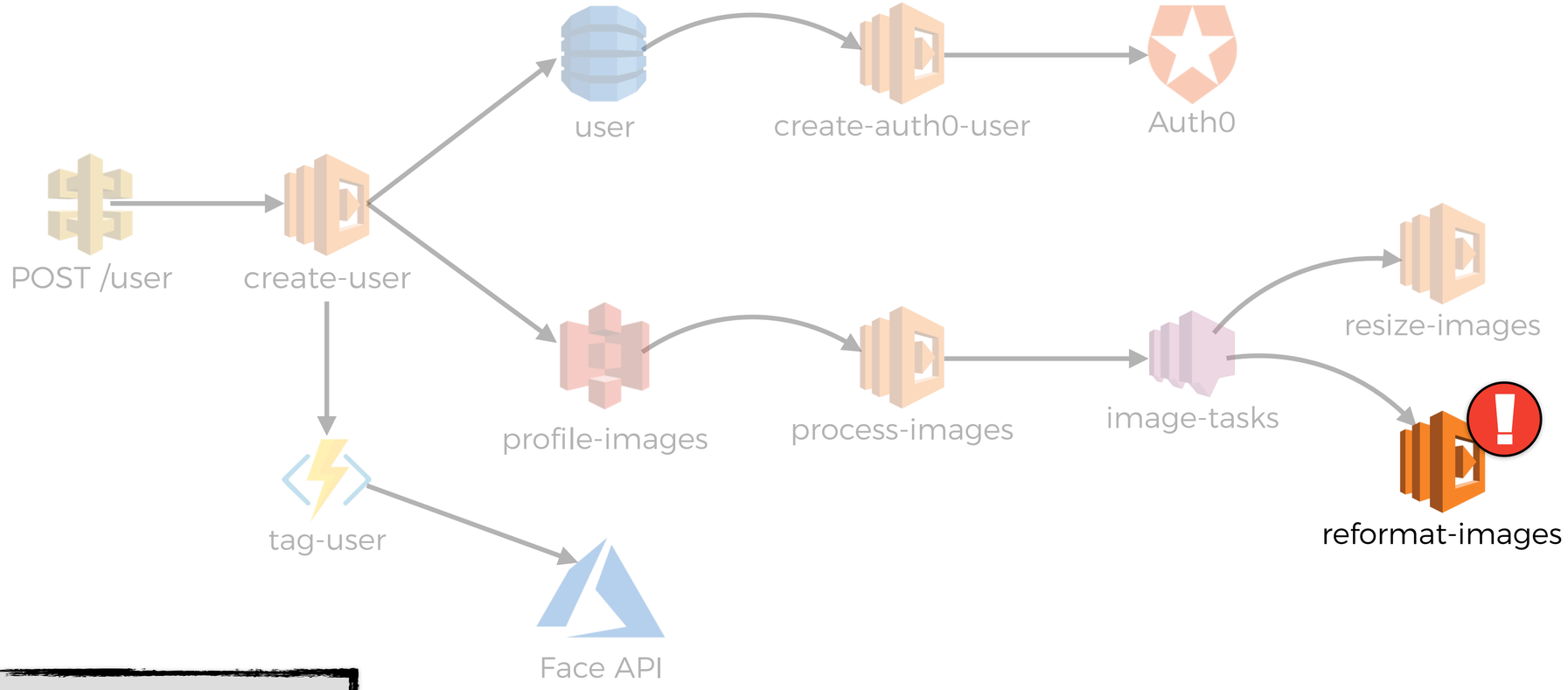
input	output

{ "Records": [
{ "Sns": {
"Type": "Notification",
"MessageId": "...",
"TopicArn": "...",
"Message": "...",
"Timestamp": "2018/01/25 20:51:24.215",
	{
	"error": null,
	"result": "OK"
	}



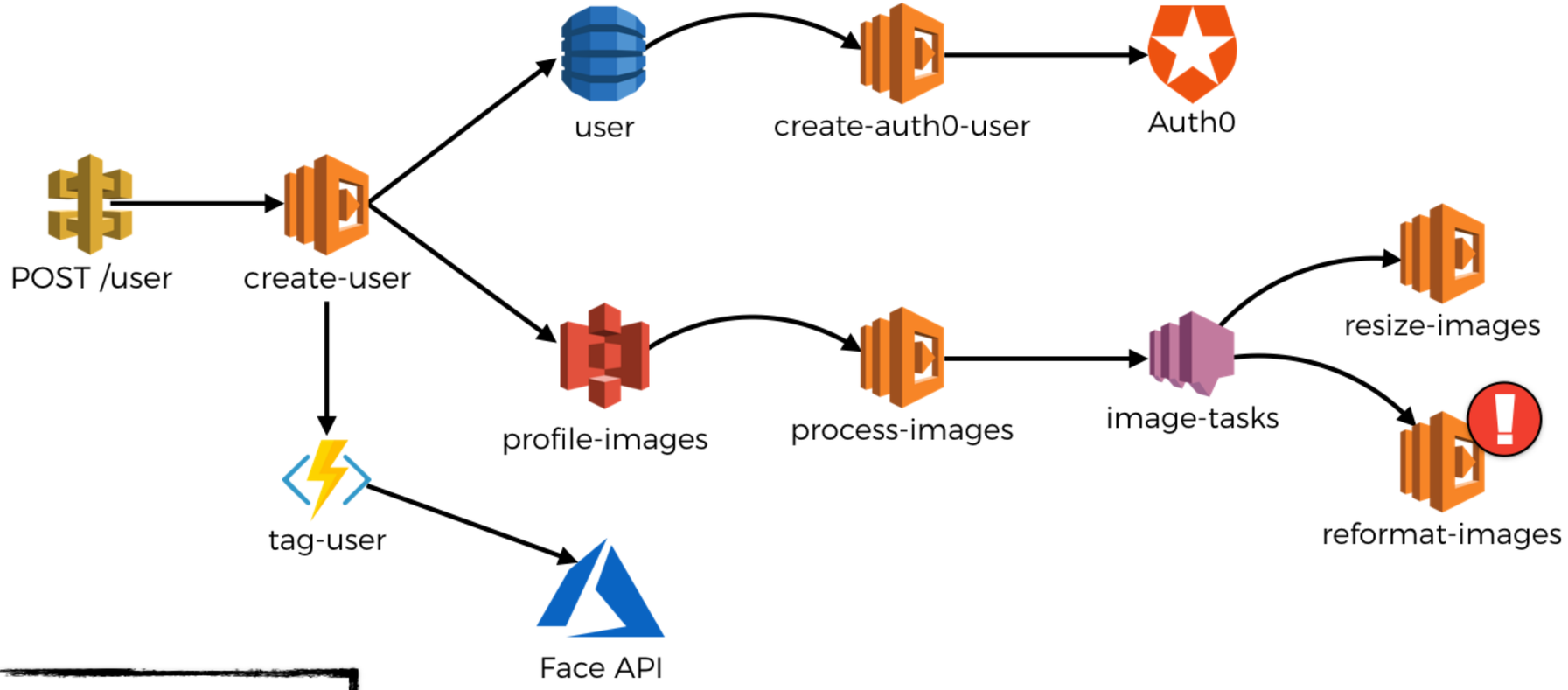
Logs | **Input/Output**

input	error
{ "Records": [{ "Sns": { "Type": "Notification", "MessageId": "...", "TopicArn": "...", "Message": "...", "Timestamp": "2018/01/25 20:51:24.215",	[com.spaceape.dragon.handler.ReformatProfileImageHandler] Null reference exception *java.lang.NullReferenceException: ... * at ... * at ... * at ...



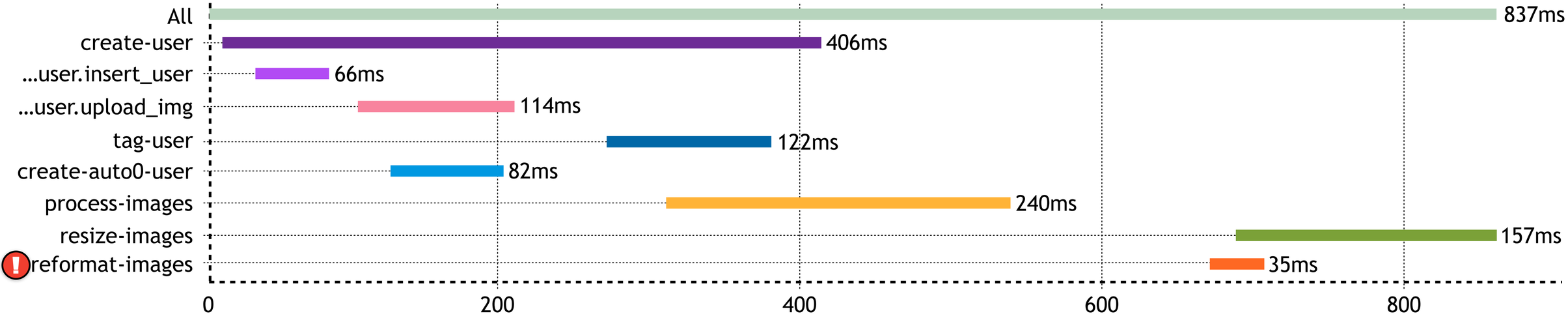
Logs | **Input/Output**

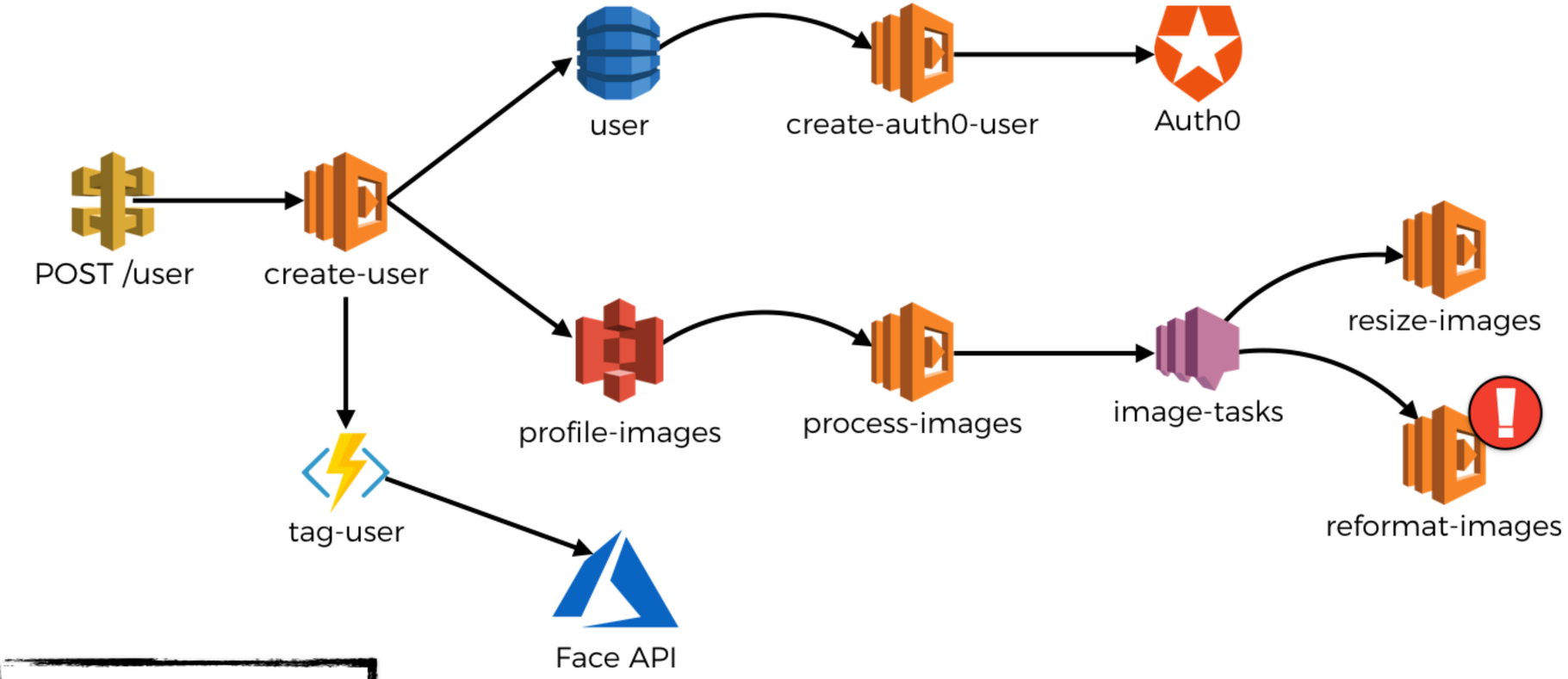
input	error
{ "Records": [{ "Sns": { "Type": "Notification", "MessageId": "...", "TopicArn": "...", "Message": "...", "Timestamp": "2018/01/25 20:51:24.215",	[com.spaceape.dragon.handler.ReformatProfileImageHandler] Null reference exception *java.lang.NullReferenceException: ... * at ... * at ... * at ...



Logs | **Input/Output**

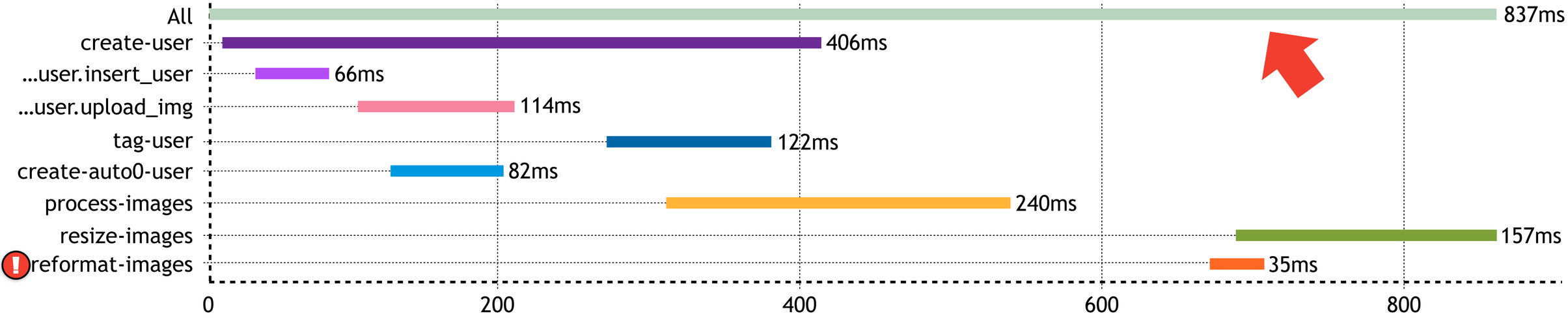
timestamp	component	level	message
2018/01/25 20:51:23.188	POST /user	debug	incoming request...
2018/01/25 20:51:23.201	create-user	debug	saving user [theburningmonk] in the [user] table...
2018/01/25 20:51:23.215	create-user	debug	saved user [theburningmonk] in the [user] table
2018/01/25 20:51:23.521	tag-user	debug	tagging user [theburningmonk] with Azure Face API...

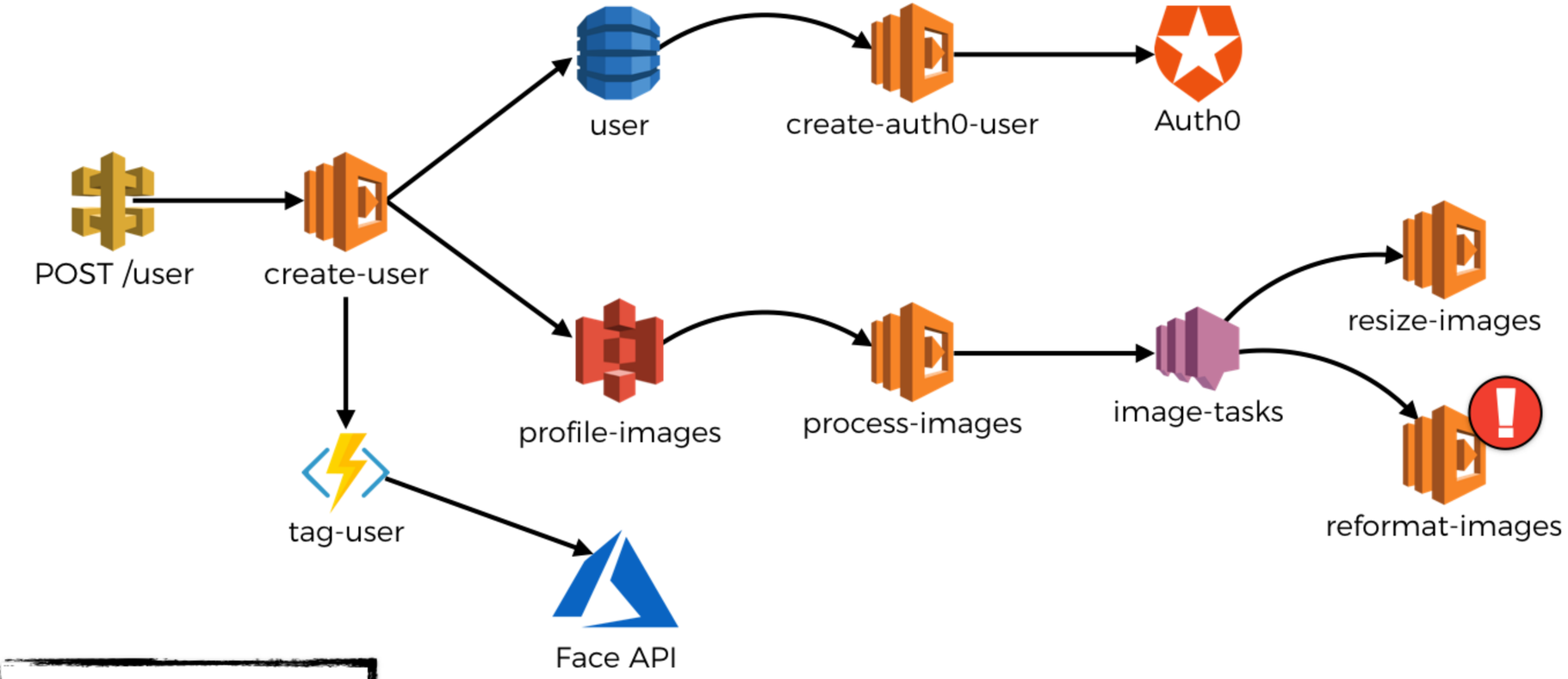




Logs | **Input/Output**

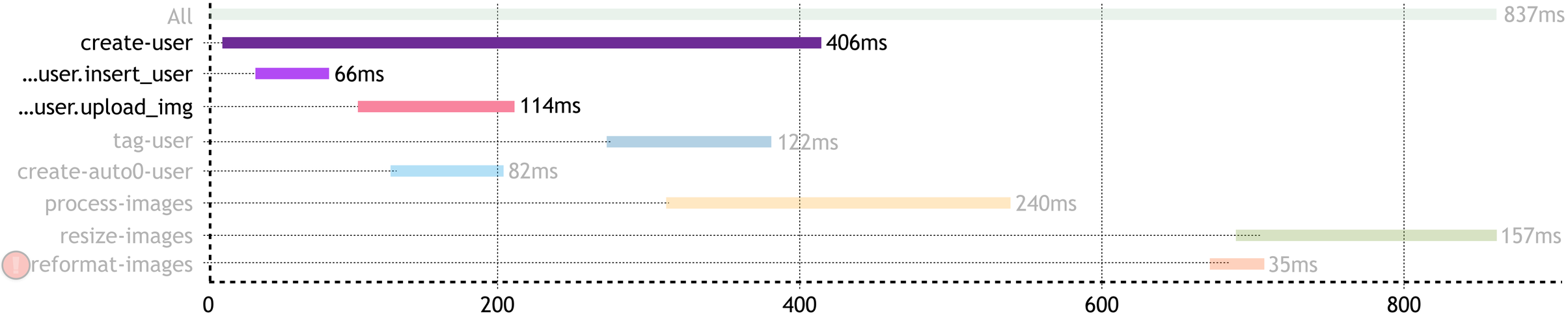
timestamp	component	level	message

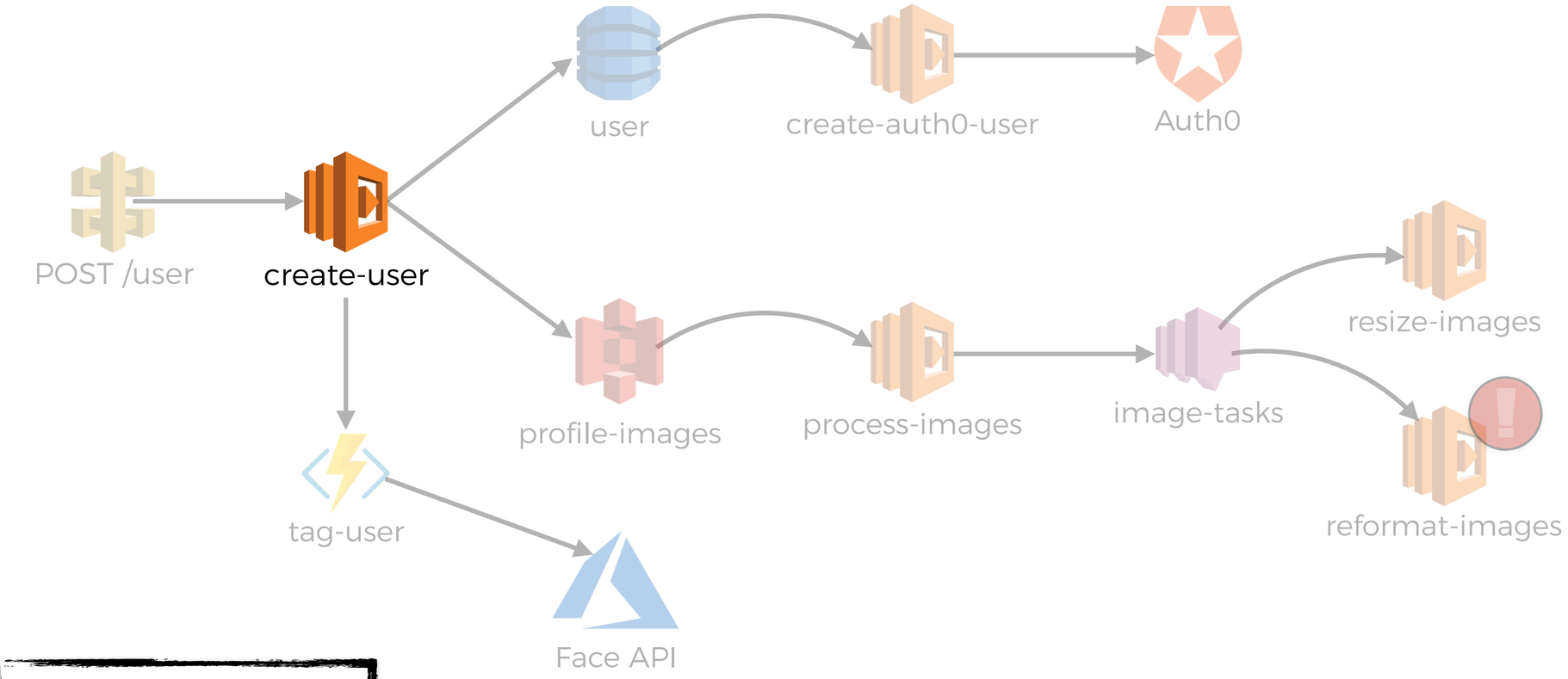




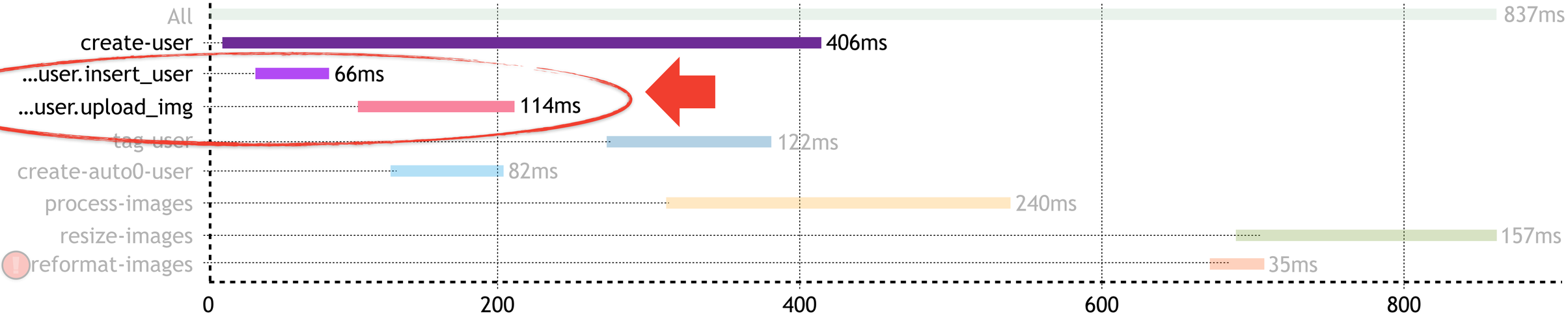
Logs | **Input/Output**

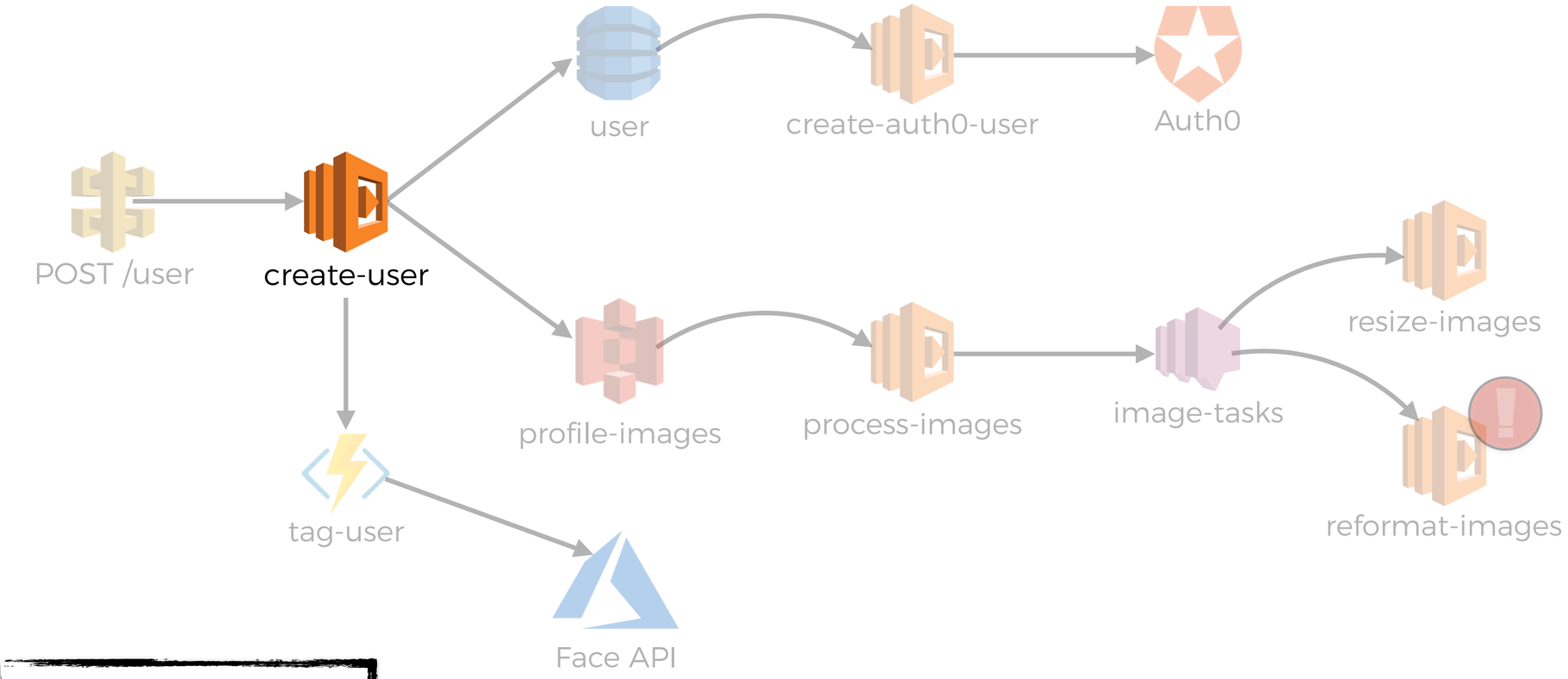
timestamp	component	level	message





Logs | **Input/Output**

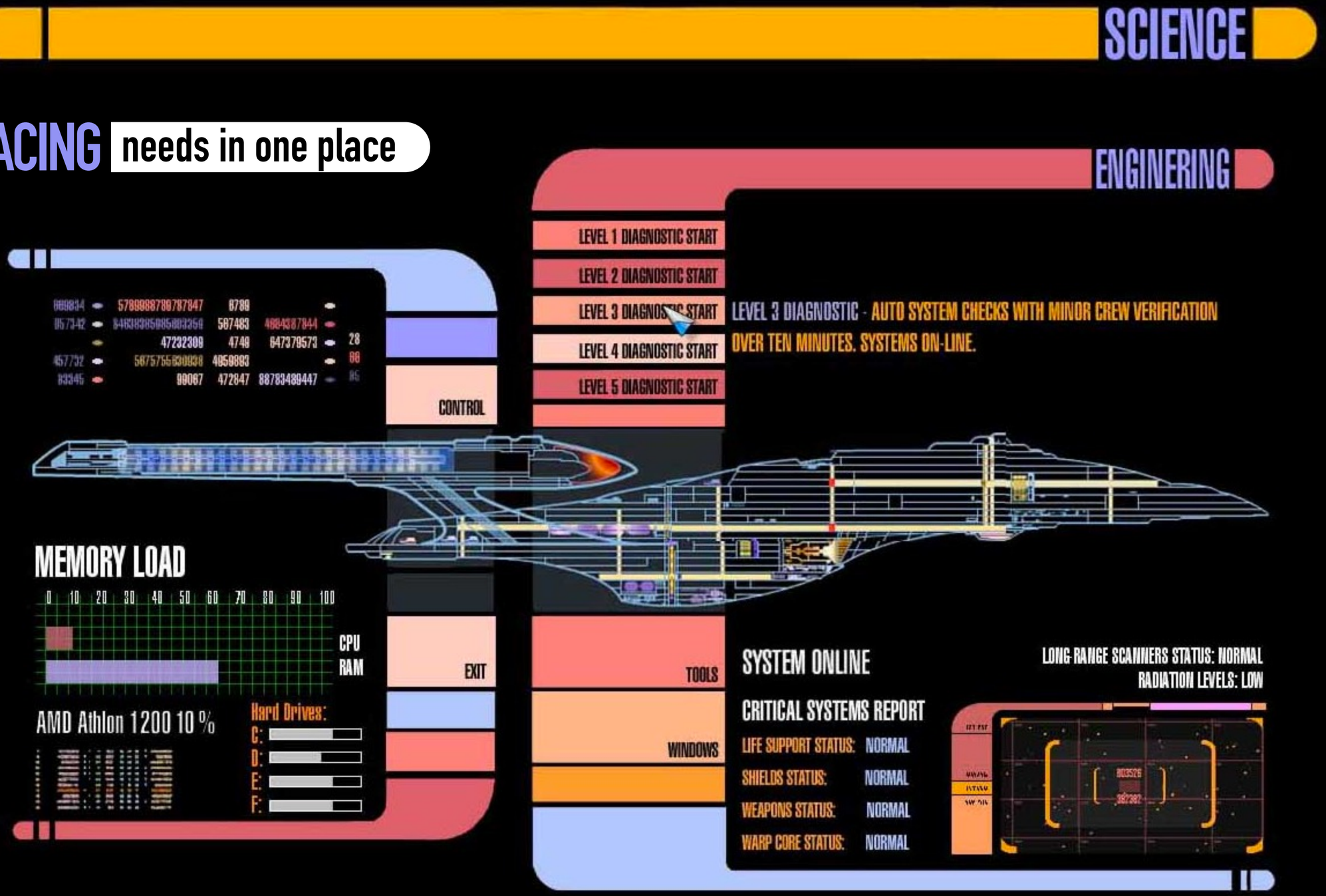




Logs | **Input/Output**

timestamp	request_id	level	message

all your TRACING needs in one place



VIZCERAL

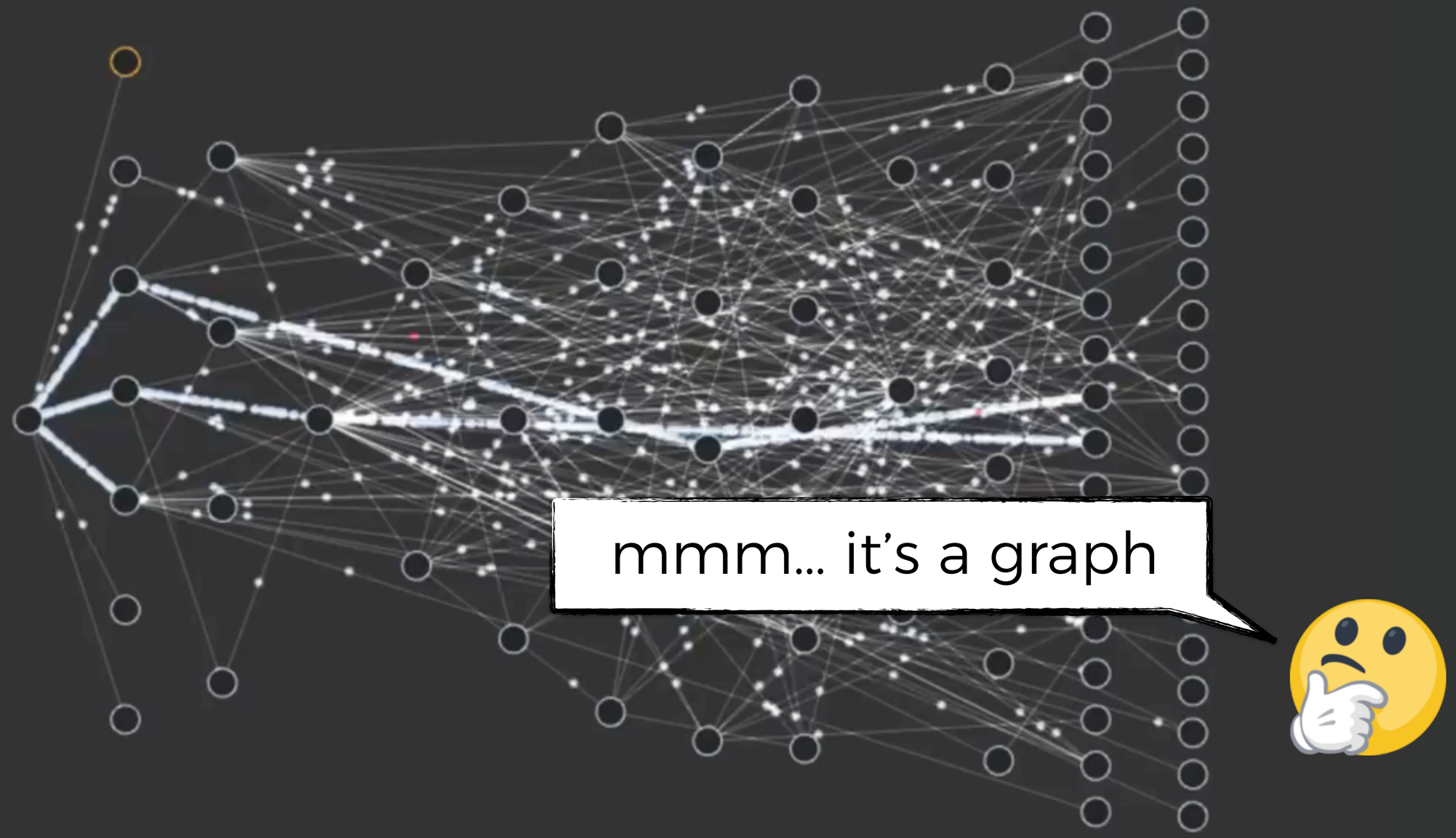
Service Traffic Map / us-east-1

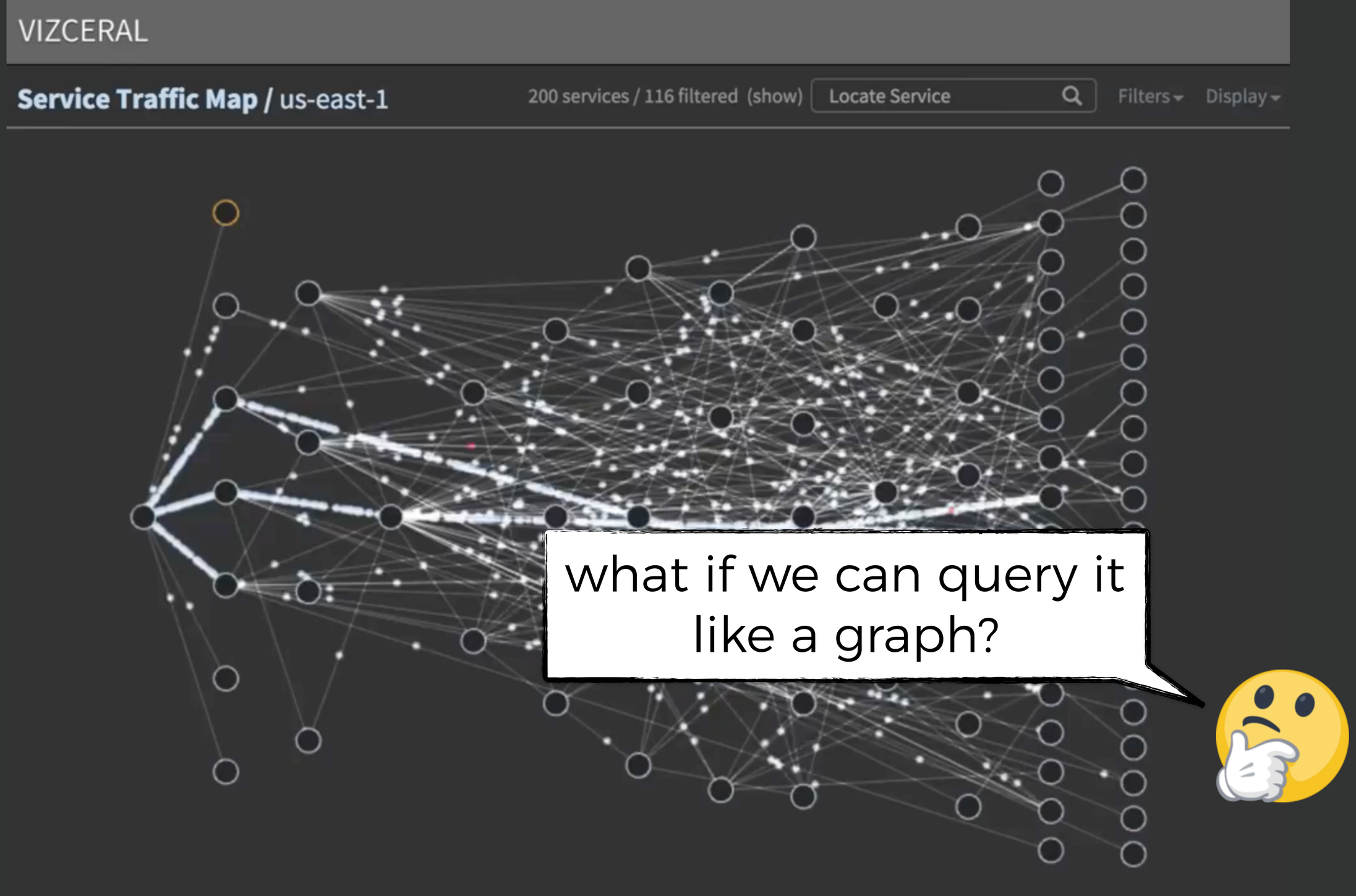
200 services / 116 filtered (show) Locate Service Filters Display

VIZCERAL

Service Traffic Map / us-east-1

200 services / 116 filtered (show) Locate Service Filters Display




VIZCERAL
Service Traffic Map / us-east-1
200 services / 116 filtered (show)
Locate Service
Filters
Display
what if we can query it like a graph?

neo4j

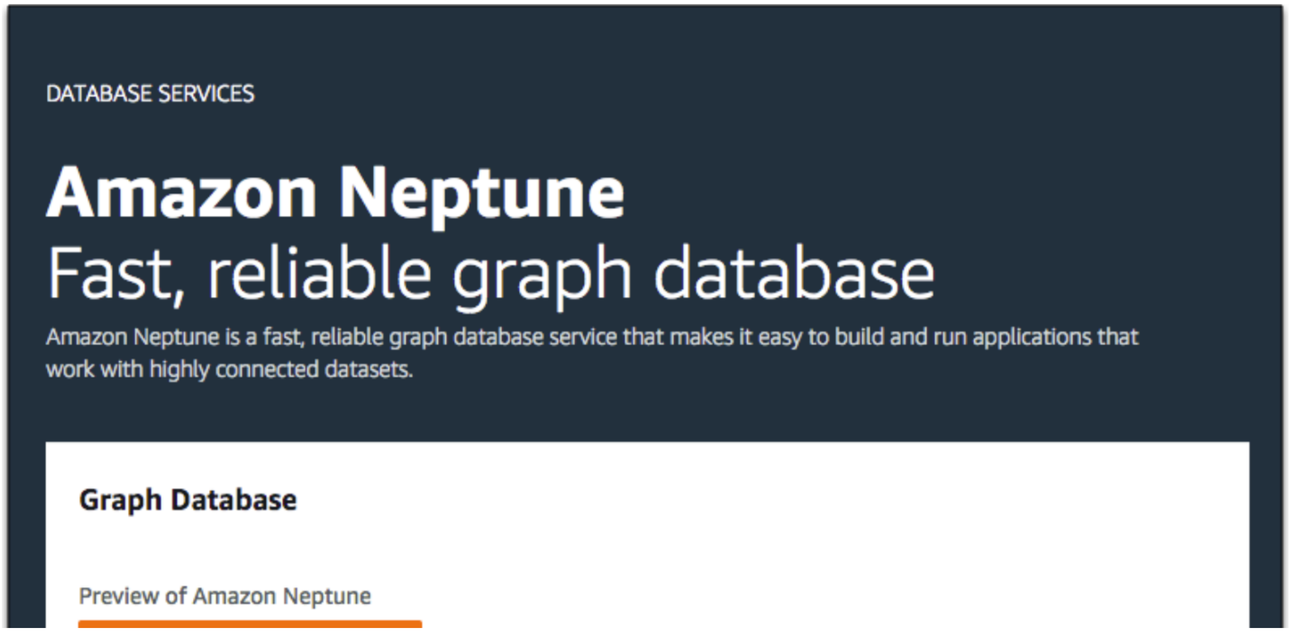
Amazon Neptune – A Fully Managed Graph Database Service

by Randall Hunt | on 29 NOV 2017 | in Amazon Neptune, Amazon RDS*, AWS Re:Invent*, Database*, Launch*, News* | Permalink | Comments | Share

Of all the data structures and algorithms we use to enable our modern lives, **graphs** are changing the world everyday. Businesses continuously create and ingest rich data with complex relationships. Yet developers are still forced to model these complex relationships in traditional databases. This leads to frustratingly complex queries with high costs and increasingly poor performance as you add relationships. We want to make it easy for you to deal with these modern and increasingly complex datasets, relationships, and patterns.

Hello, Amazon Neptune

Today we're launching a limited preview (sign up here) of Amazon Neptune, a fast and reliable graph database service that makes it easy to gain insights from relationships among your highly connected datasets. The core of Amazon Neptune is a purpose-built, high-performance graph database engine optimized for storing billions of relationships and querying the graph with milliseconds of latency. Delivered as a fully managed database, Amazon Neptune frees customers to focus on their applications rather than tedious undifferentiated operations like maintenance, patching, backups, and restores. The service supports fast-failover, point-in-time recovery, and Multi-AZ deployments for high availability. With support for up to 15 read replicas you can scale query throughput to 100s of thousands of queries per second. Amazon Neptune runs within your Amazon Virtual Private Cloud and allows you to encrypt your data at rest, giving you complete control over your data integrity in transit and at rest.



http://amzn.to/2nk7uiW

ability to query based on the **relationship** between observed components

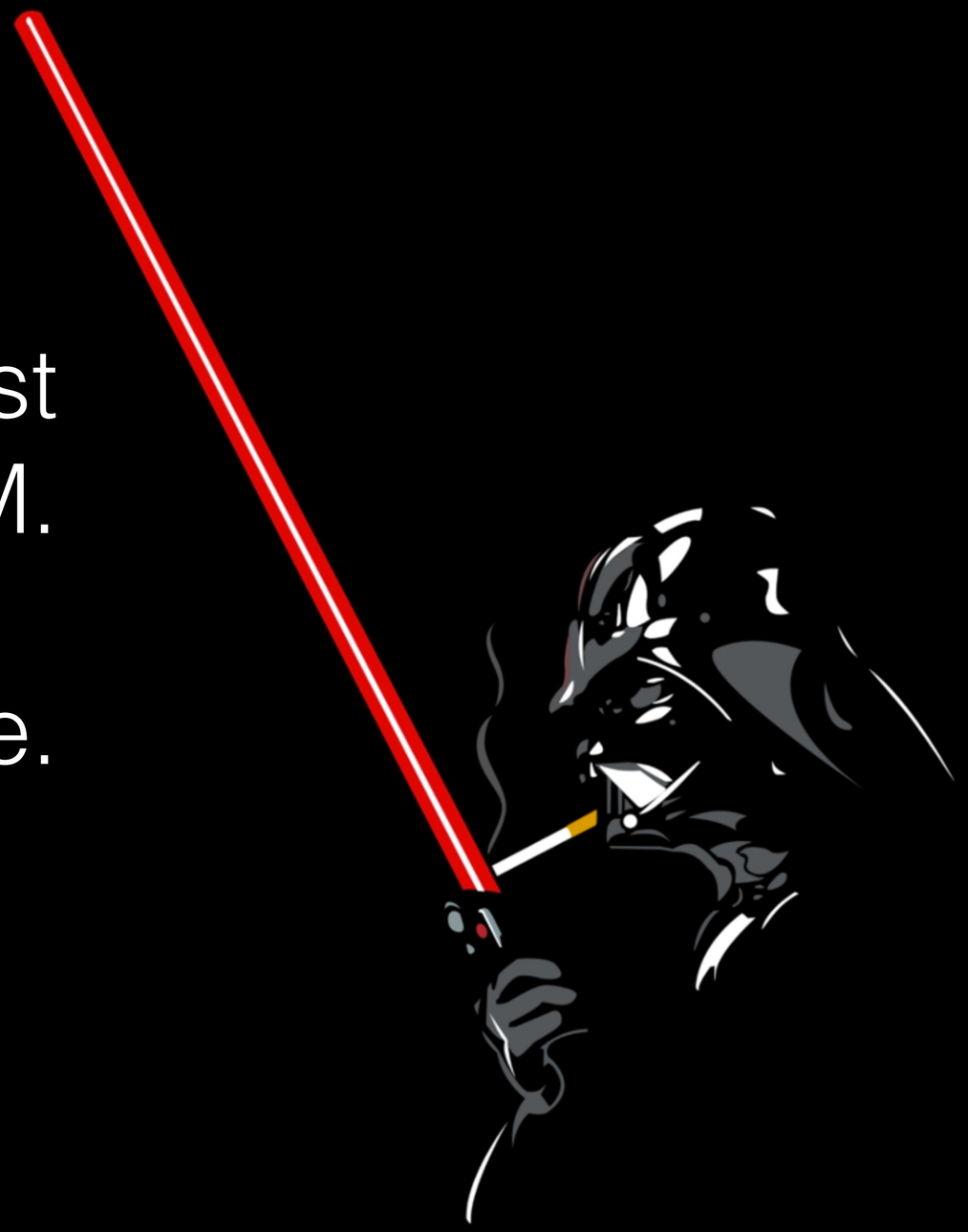
(as well as the components themselves)

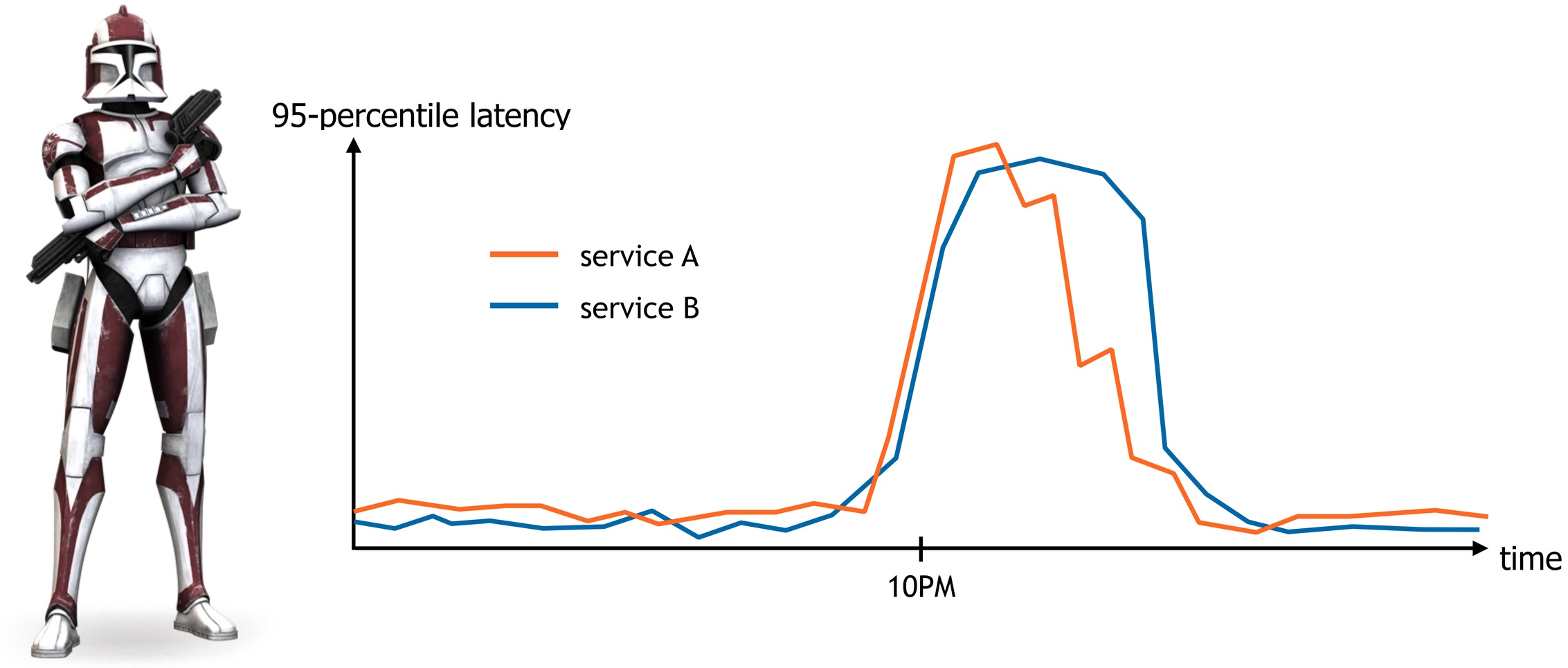
root **cause** analysis

> *the elevated error rate in service X was **caused** by DynamoDB table throttling.*

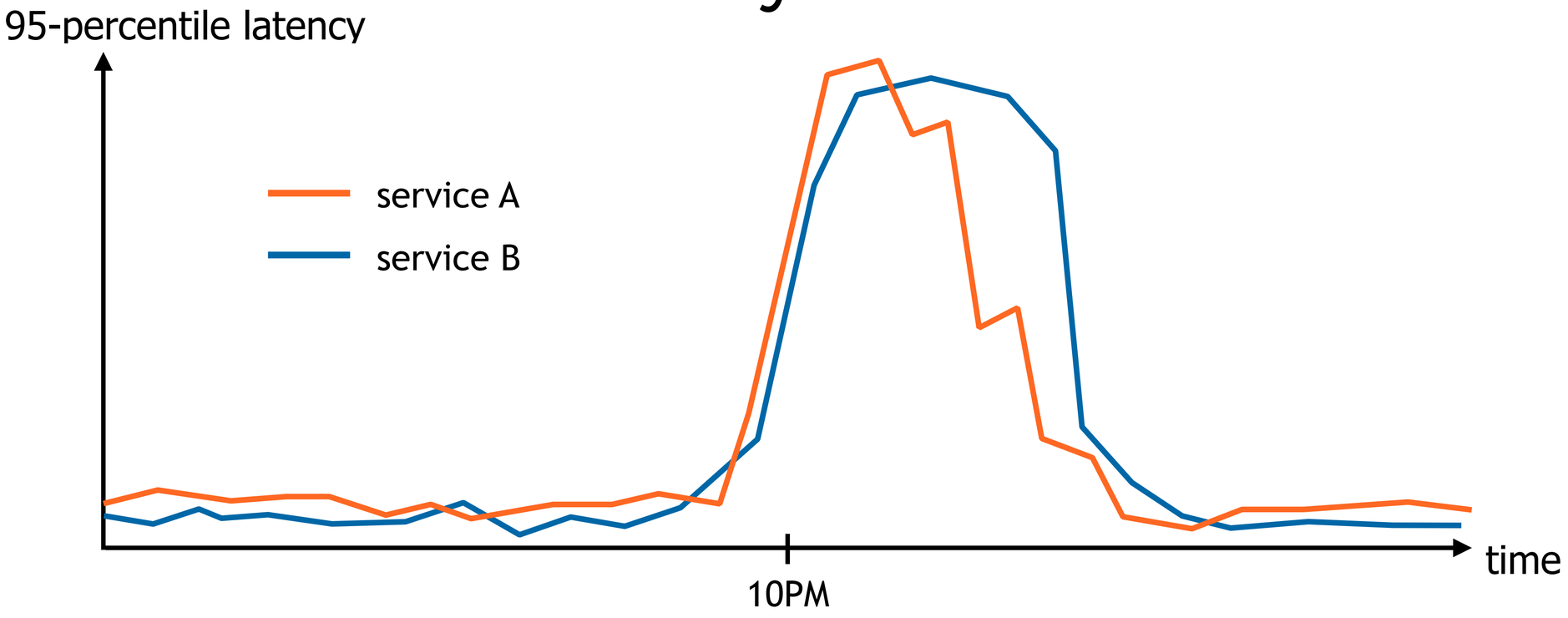
payment was slow last night around 10PM.

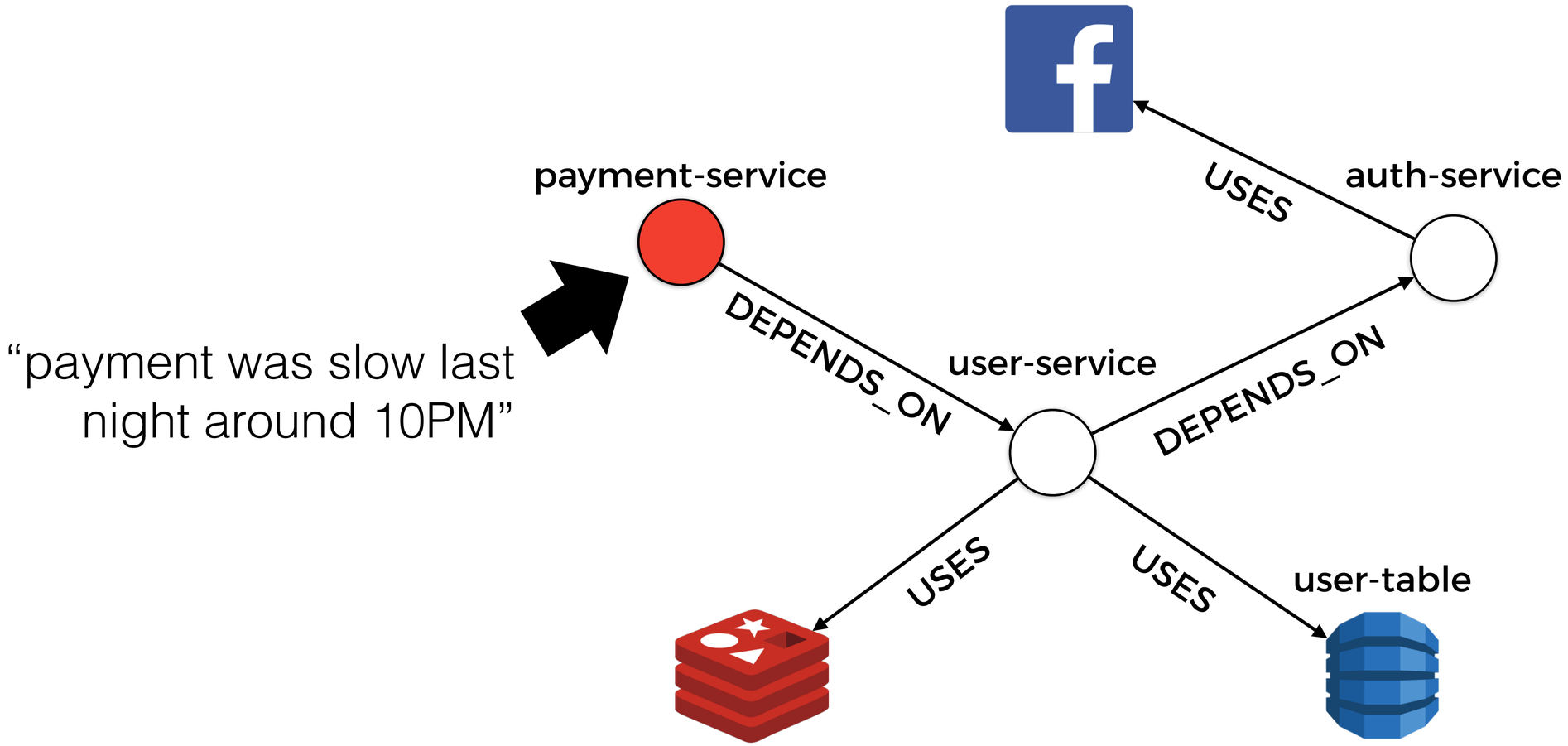
investigate.

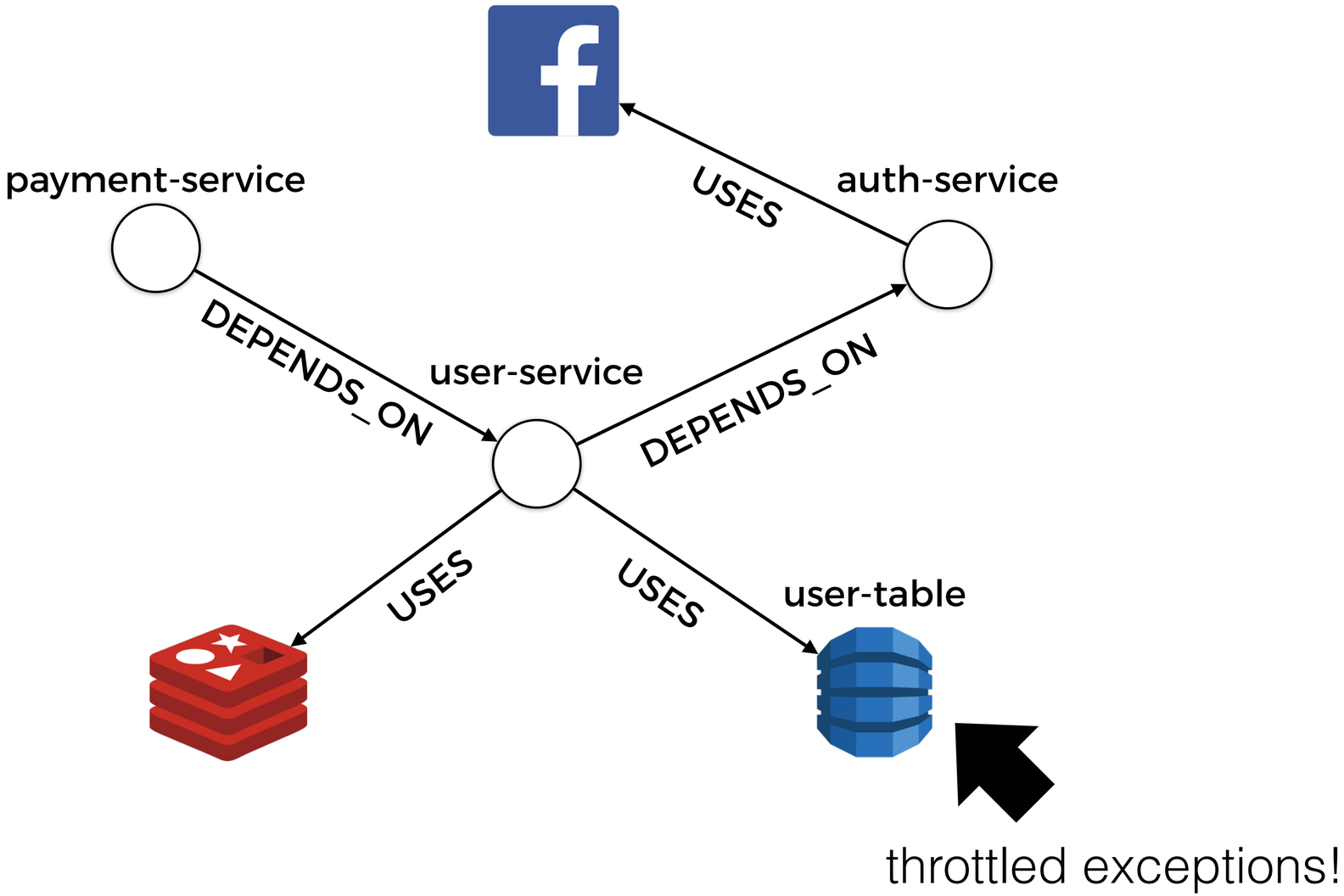



95-percentile latency
service A
service B
10PM
time

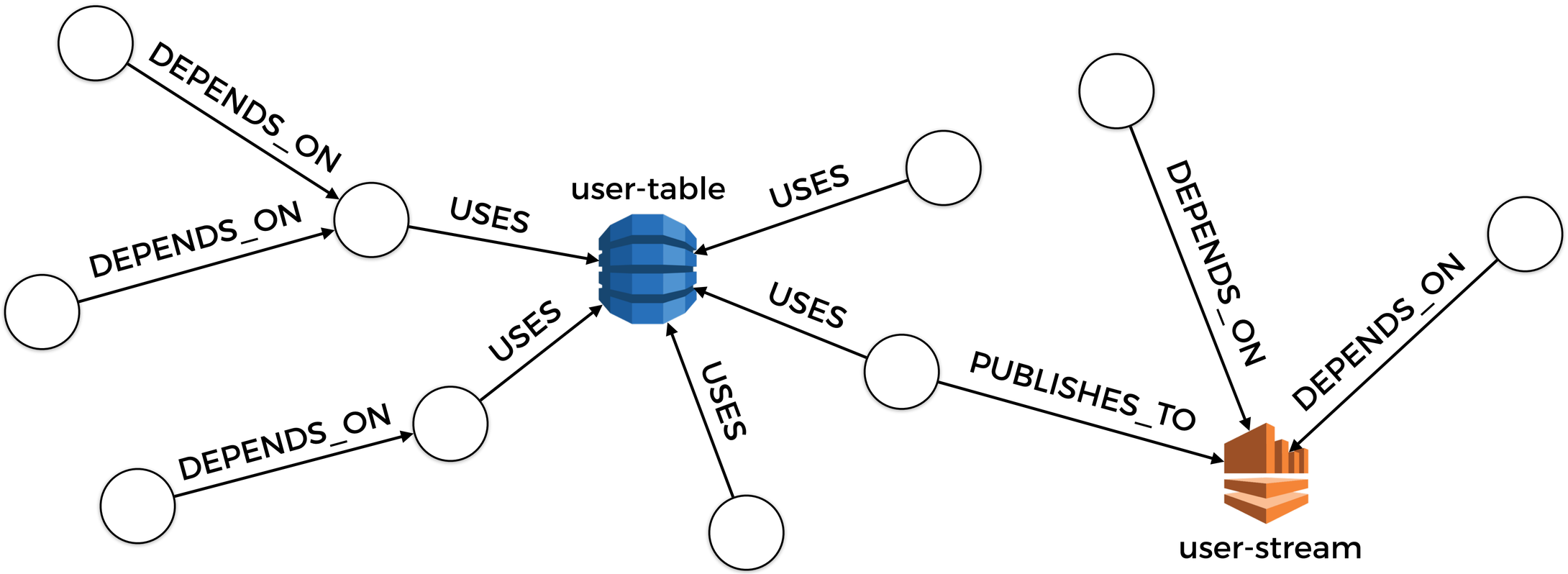
causality? or correlation?



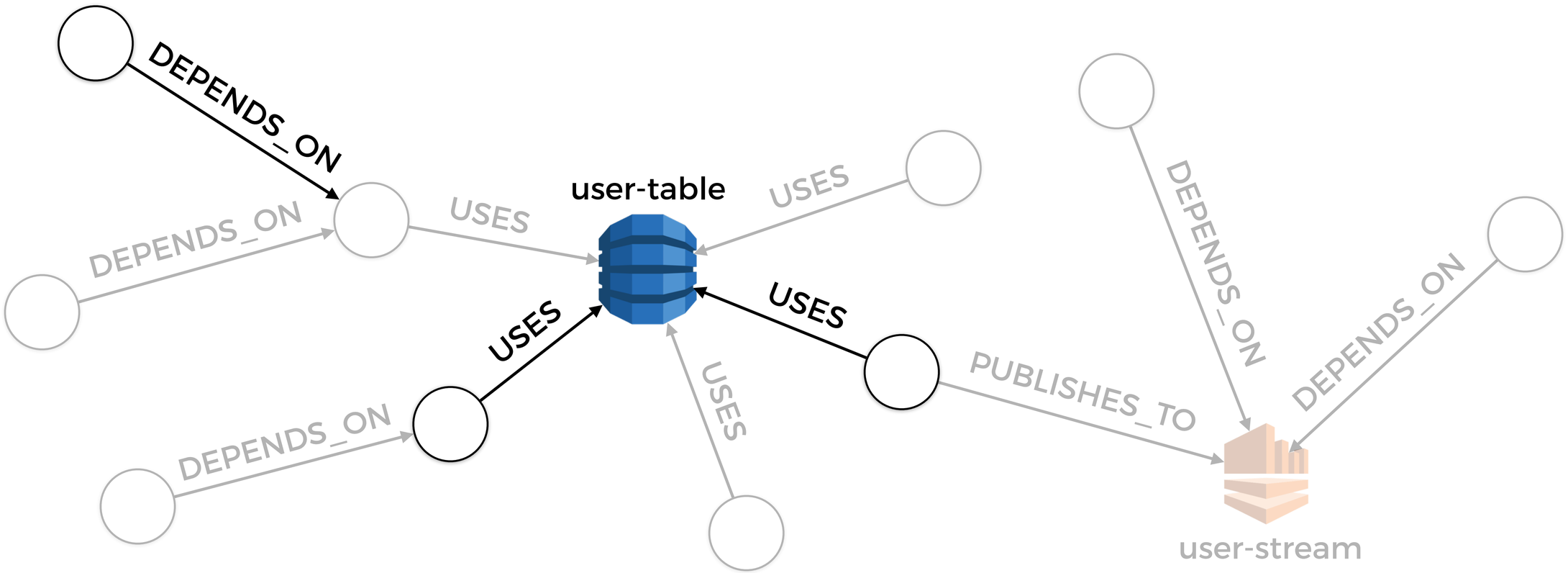

payment-service
"payment was slow last night around 10PM"
DEPENDS_ON
user-service
USES
auth-service
DEPENDS_ON
USES
USES
user-table


payment-service
USES
auth-service
DEPENDS_ON
user-service
DEPENDS_ON
USES
USES
user-table
throttled exceptions!

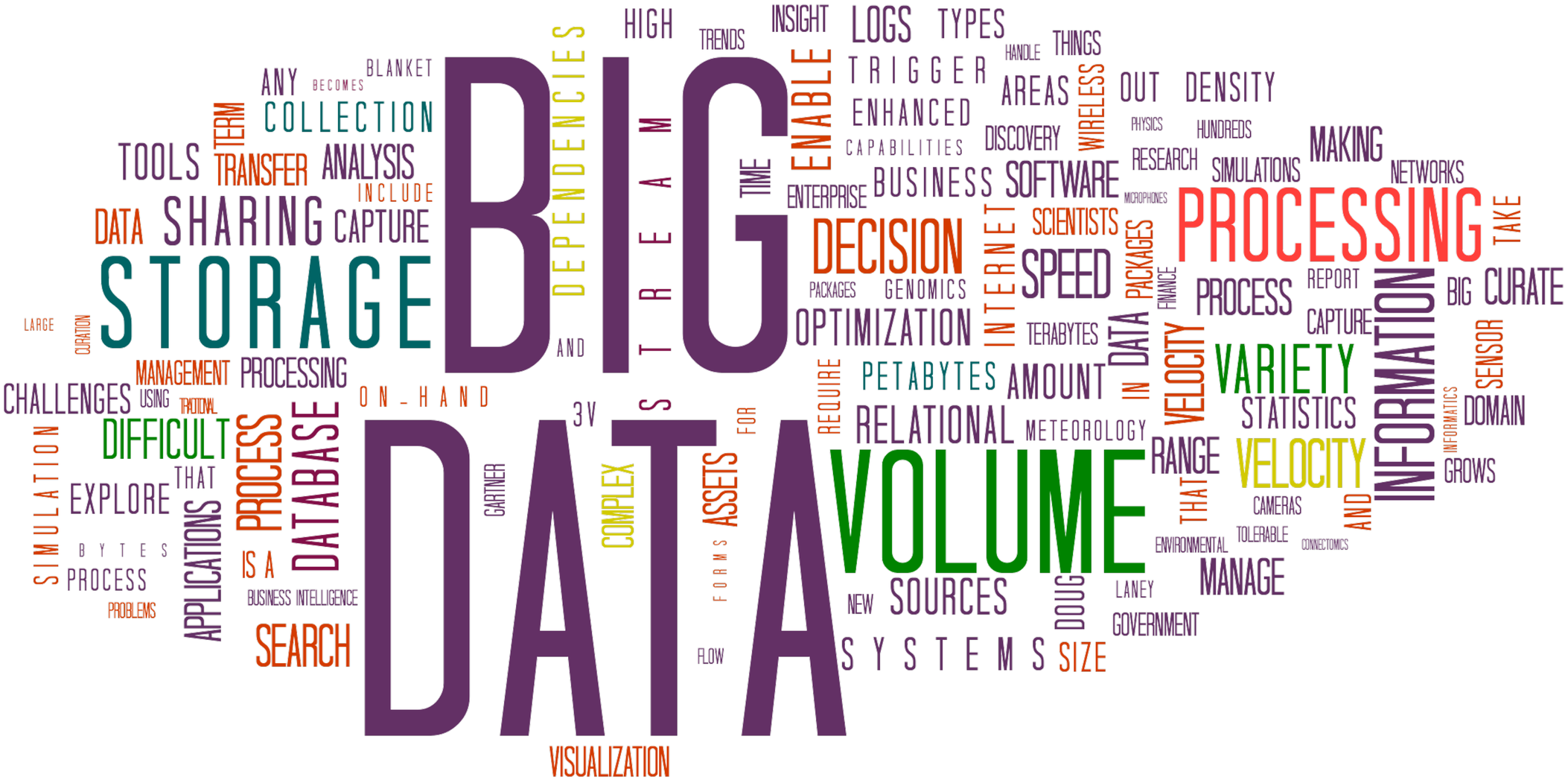
“what else is impacted by the throttled exceptions on user-table?”



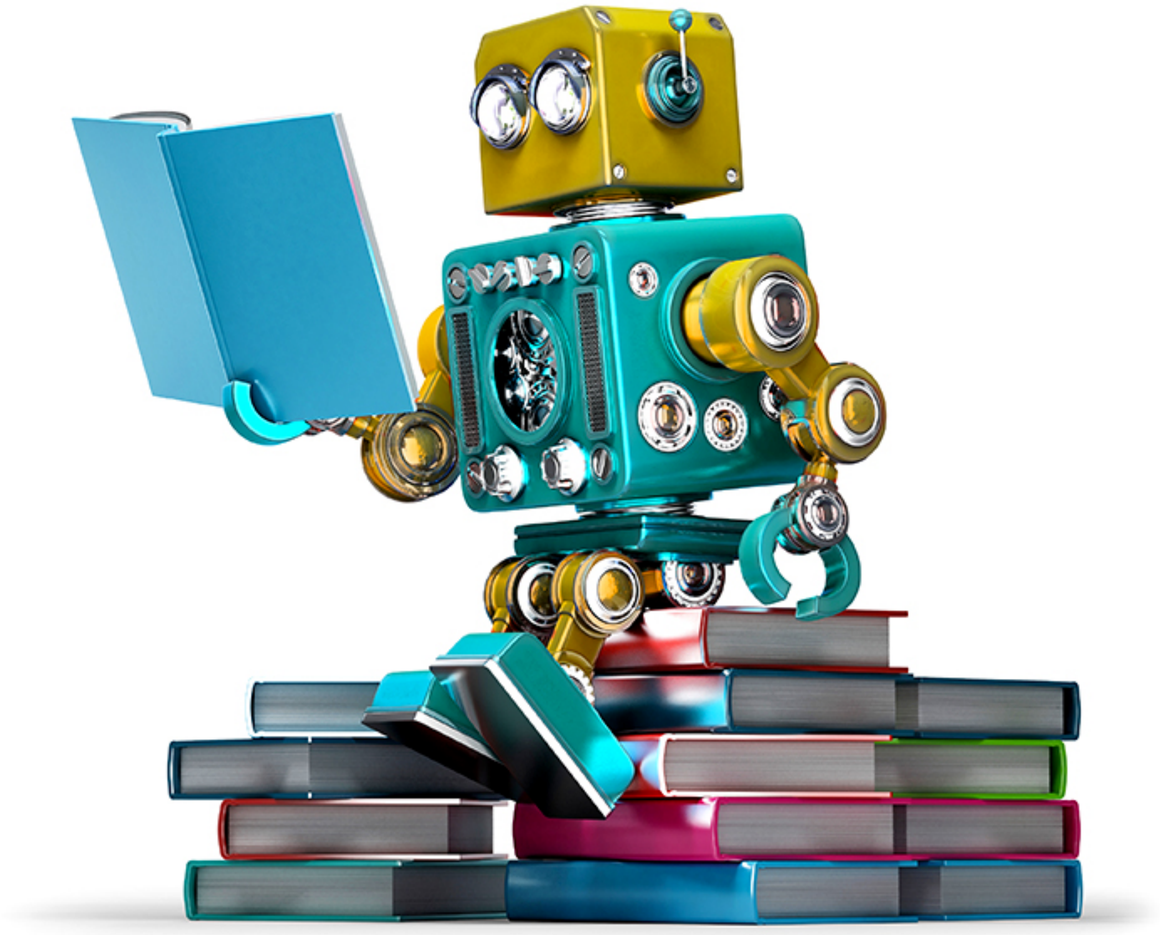
“what else is impacted by the throttled exceptions on user-table?”



wouldn’t that be nice?


BIG
DATA
VOLUME
STORAGE
PROCESSING
INFORMATION
VARIETY
VELOCITY
DECISION
SPEED
SOFTWARE
BUSINESS
ENABLE
STREAM
DEPENDENCIES
COLLECTION
TOOLS
TRANSFER
ANALYSIS
SHARING
CAPTURE
DATA
DIFFICULT
CHALLENGES
MANAGEMENT
PROCESSING
DATABASE
PROCESS
APPLICATIONS
SIMULATION
EXPLORE
SEARCH
VISUALIZATION
SYSTEMS
SOURCES
RELATIONAL
OPTIMIZATION
GENOMICS
PETABYTES
AMOUNT
STATISTICS
RANGE
MANAGE
CURATE
DOMAIN
SENSOR
GROWS
MAKING
NETWORKS
SIMULATIONS
RESEARCH
DISCOVERY
ENHANCED
TRIGGER
LOGS
TYPES
INSIGHT
HIGH
TRENDS
THINGS
OUT
DENSITY
AREAS
WIRELESS
INTERNET
SCIENTISTS
PACKAGES
TERABYTES
METEOROLOGY
CAMERAS
TOLERABLE
ENVIRONMENTAL
GOVERNMENT
LANEY
DOUG
SIZE
NEW
FLOW
ASSETS
COMPLEX
GARTNER
3V
ON-HAND
BLANKET
BECOMES
ANY
TERM
INCLUDE
LARGE
CURATION
USING
TRADITIONAL
THAT
BYTES
PROBLEMS
IS A
BUSINESS INTELLIGENCE
FORMS
FOR
REQUIRE
AND
TIME
ENTERPRISE
CAPABILITIES
HANDLE
PHYSICS
HUNDREDS
MICROPHONES
REPORT
CAPTURE
TAKE
BIG
INFORMATICS
CONNECTOMICS

MACHINE LEARNING



use ML to auto-detect **erroneous** or **suspicious** behaviours, or to suggest possible **improvements**

VIZCERAL

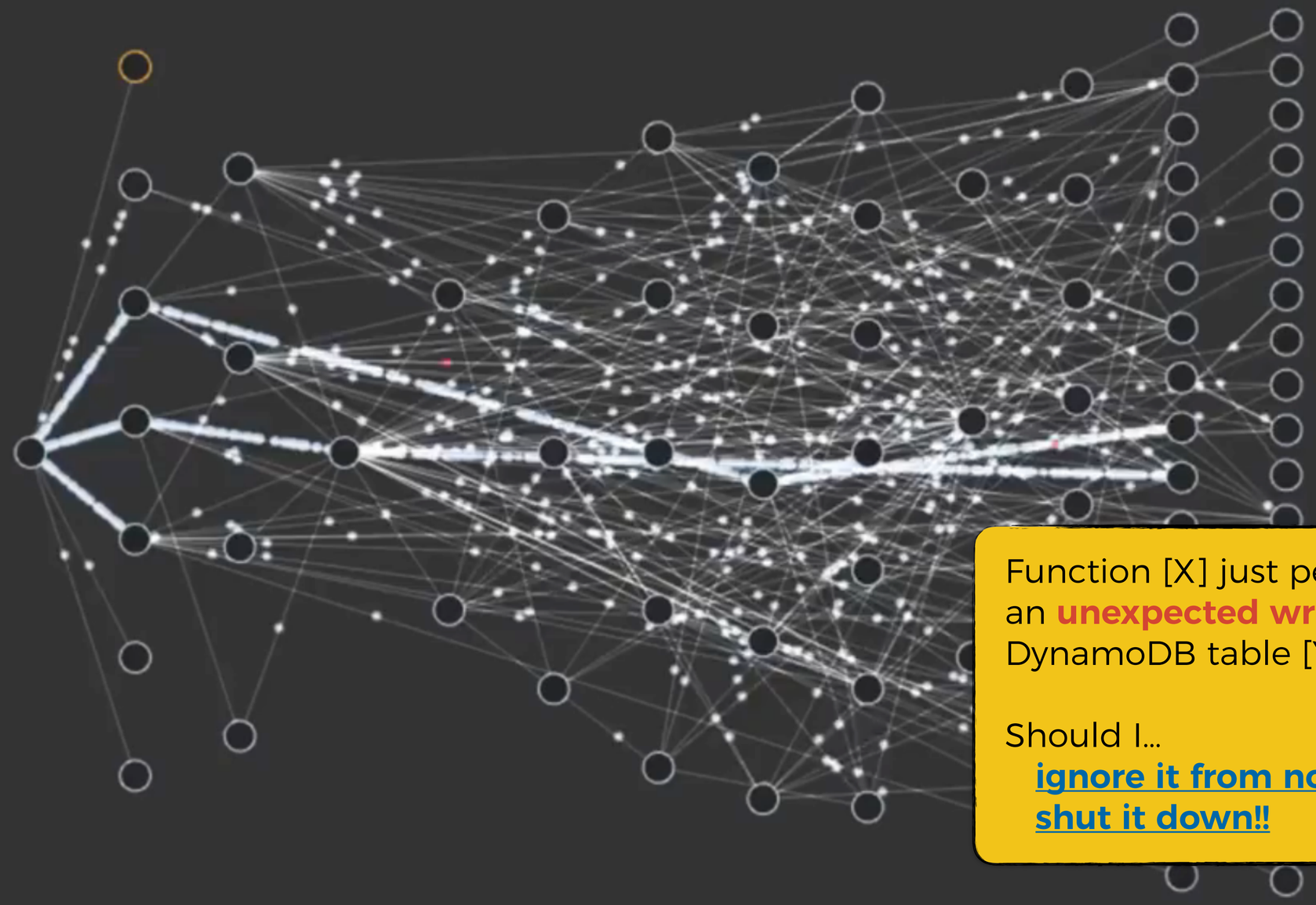
Service Traffic Map / us-east-1

200 services / 116 filtered (show) Locate Service Filters Display

VIZCERAL

Service Traffic Map / us-east-1

200 services / 116 filtered (show) | Locate Service | Filters | Display



Function [X] just performed an **unexpected write** against DynamoDB table [Y].

Should I...

ignore it from now on

shut it down!!

!

Google

Gmail

More

1 of many

COMPOSE

Inbox (805)
Starred
Important
Sent Mail
Drafts (10)

Categories

Deleted Messages
Job Agencies
Notes
Personal
Receipts
Sent Messages
Travel
Twitter
Work

[production] Suspicious behaviour detected!

Inbox x

Observability Bot <bot@bestobservability.com> 1:18 AM (0 minutes ago)

to me

Hello Yan,

The function [**X**] has performed an **unexpected write** against DynamoDB table [**Y**], it has never done that before, is this intended?

Should I...

ignore it from now on
shut it down!!!

Click here to Reply or Forward

Google

Gmail

More

1 of many

COMPOSE

Inbox (805)
Starred
Important
Sent Mail
Drafts (10)

Categories

Deleted Messages
Job Agencies
Notes
Personal
Receipts
Sent Messages
Travel
Twitter
Work

5

[production] Suspicious behaviour detected!

Inbox x

Observability Bot <bot@bestobservability.com> 1:18 AM (0 minutes ago)

to me

Hello Yan,

The function [**X**] has performed an **unexpected write** against DynamoDB table [**Y**], it has never done that before, is this intended?

Should I...
ignore it from now on
shut it down!!!

don't bother me about this again

Click here to Reply or Forward

Google

Gmail

More

1 of many

COMPOSE

Inbox (805)
Starred
Important
Sent Mail
Drafts (10)

Categories

Deleted Messages
Job Agencies
Notes
Personal
Receipts
Sent Messages
Travel
Twitter
Work

[production] Suspicious behaviour detected!

Inbox x

Observability Bot <bot@bestobservability.com> 1:18 AM (0 minutes ago)

to me

Hello Yan,

The function [**X**] has performed an **unexpected write** against DynamoDB table [**Y**], it has never done that before, is this intended?

Should I...

ignore it from now on
shut it down!!!

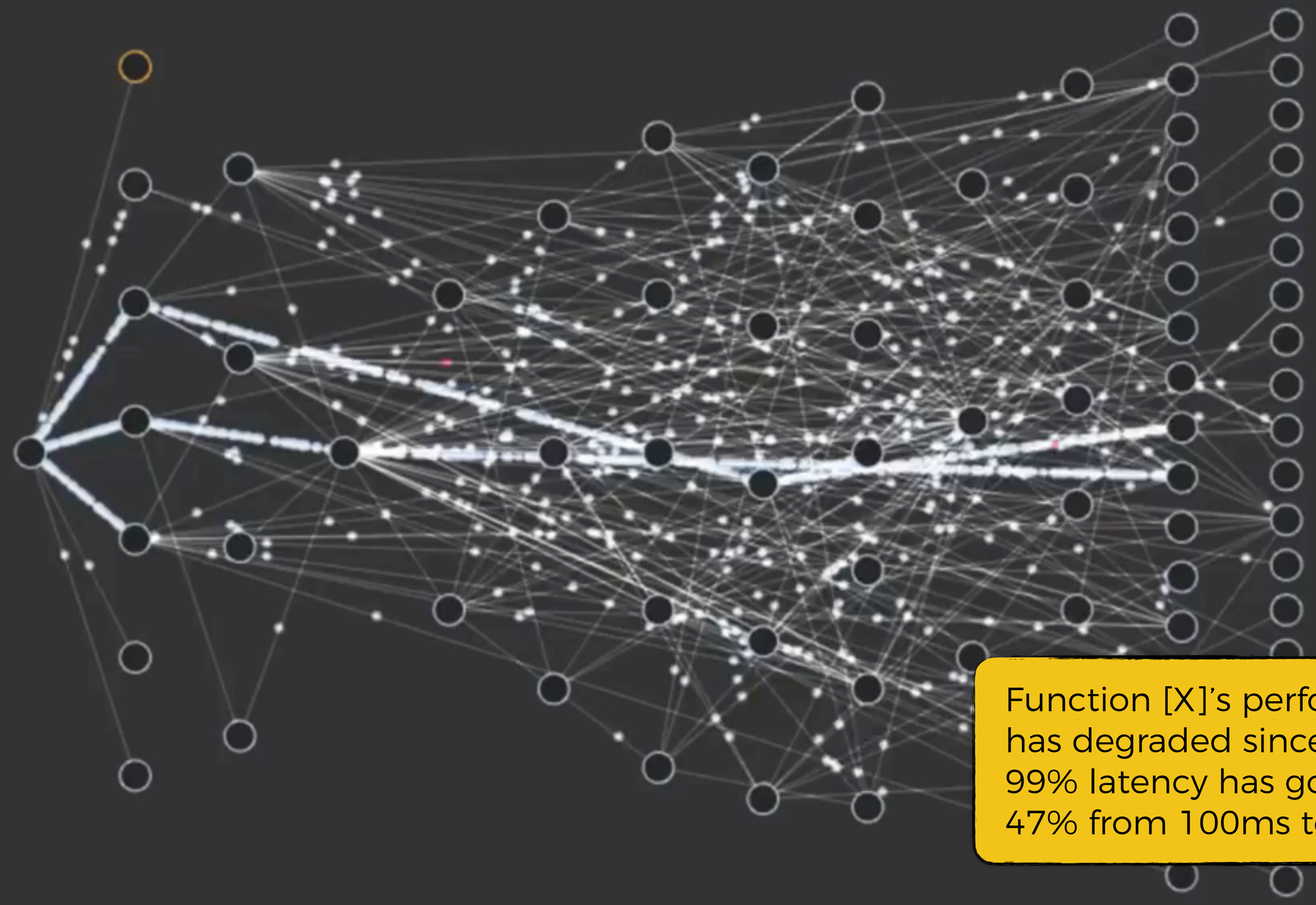
auto-modify IAM role with DENY rule

Click here to Reply or Forward

VIZCERAL

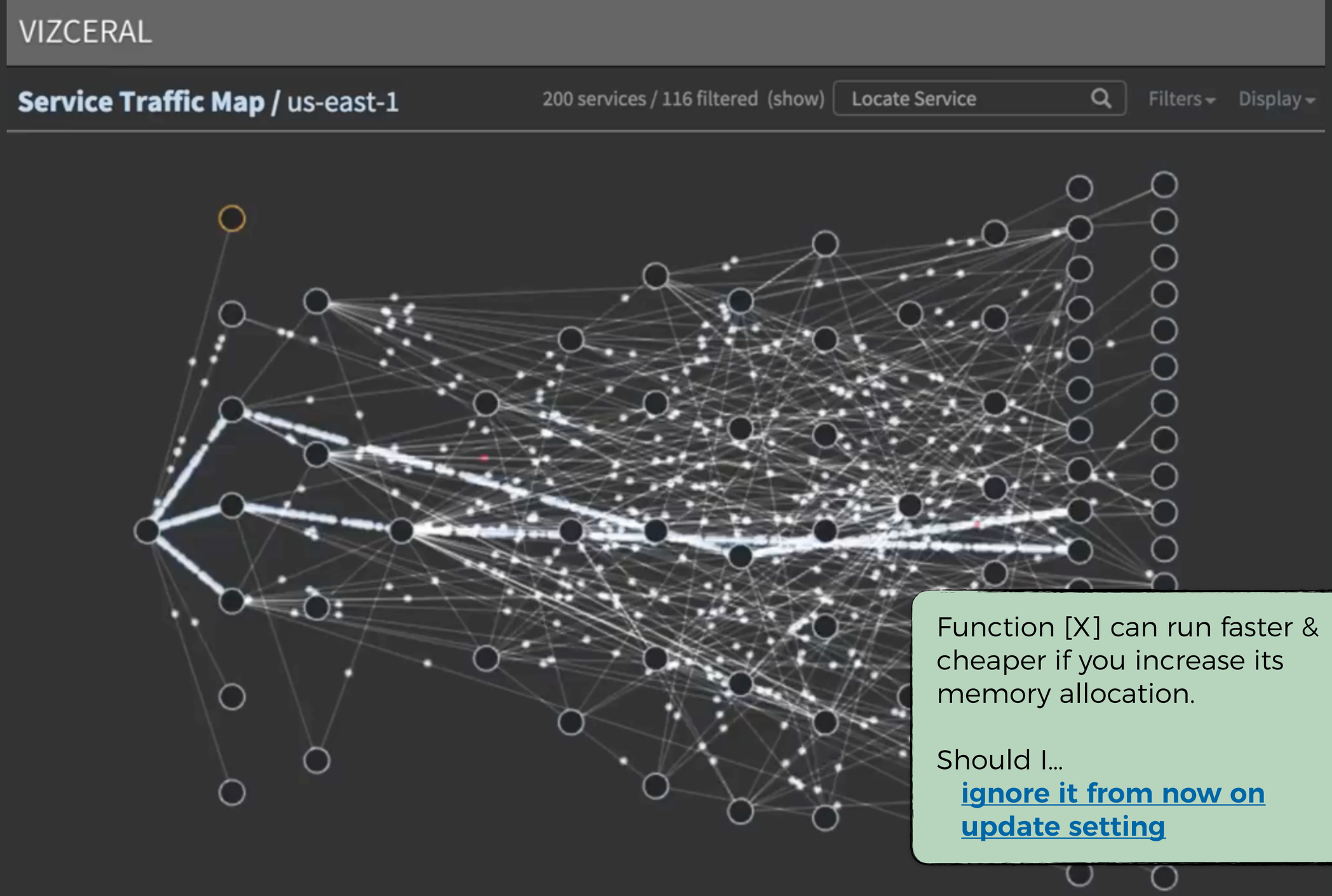
Service Traffic Map / us-east-1

200 services / 116 filtered (show) | Locate Service | Filters | Display



Function [X]'s performance has degraded since yesterday - 99% latency has gone up by 47% from 100ms to 147ms.

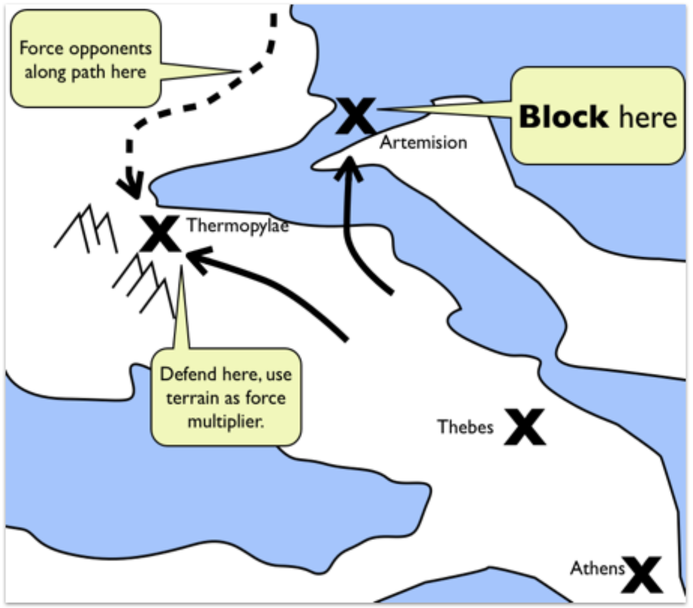
!


VIZCERAL
Service Traffic Map / us-east-1
200 services / 116 filtered (show)
Locate Service
Filters
Display
Function [X] can run faster & cheaper if you increase its memory allocation.
Should I...
ignore it from now on
update setting
!


zzz... the future of... zzz ...
serverless observability... zzz

Simon Wardley

Which is MORE useful in determining strategy?



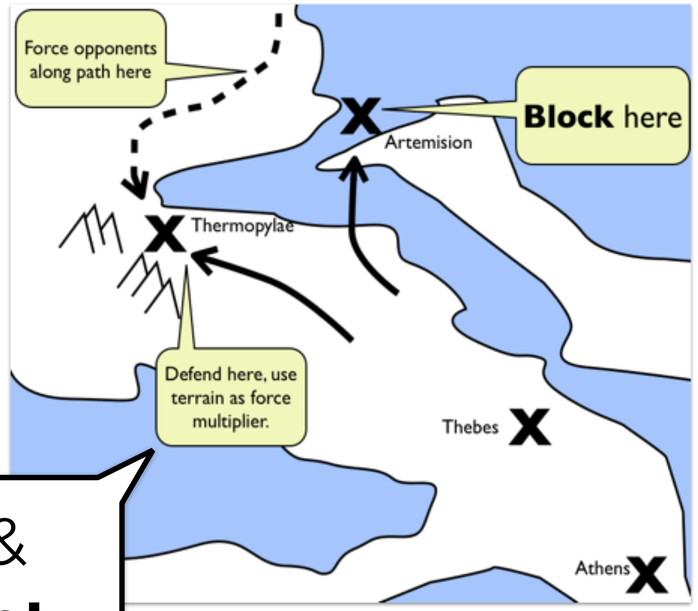
OR

Which one do we use in business?

Which is MORE useful in determining strategy?

Simon Wardley

context & movement



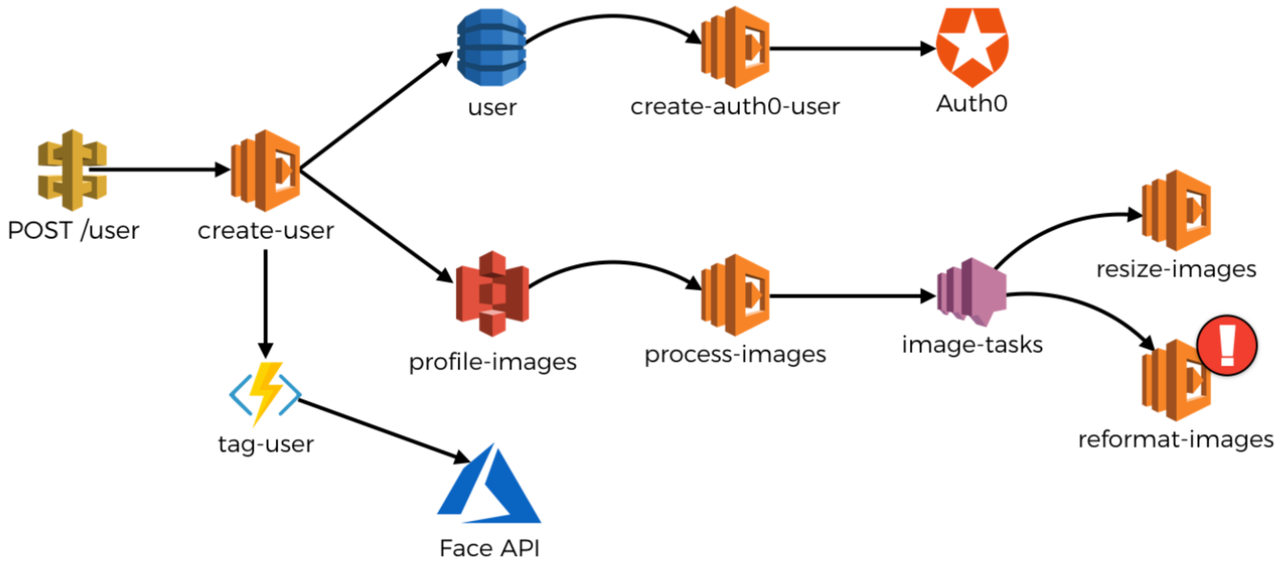
OR

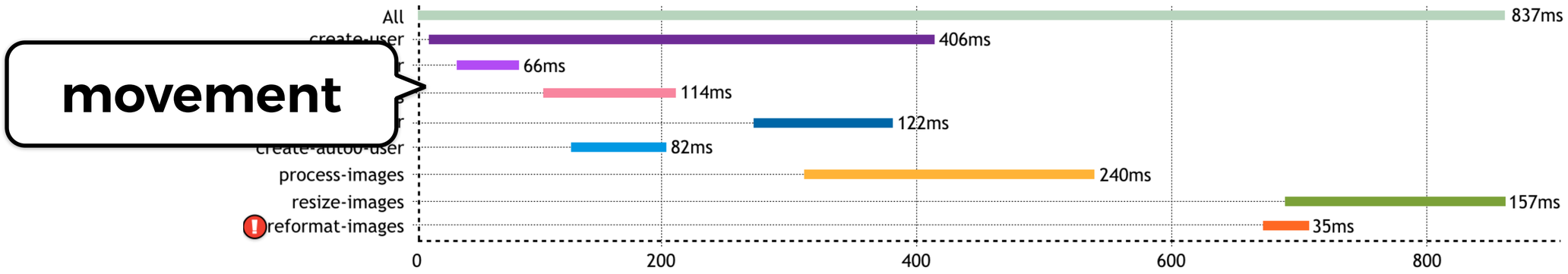
Which one do we use in business?

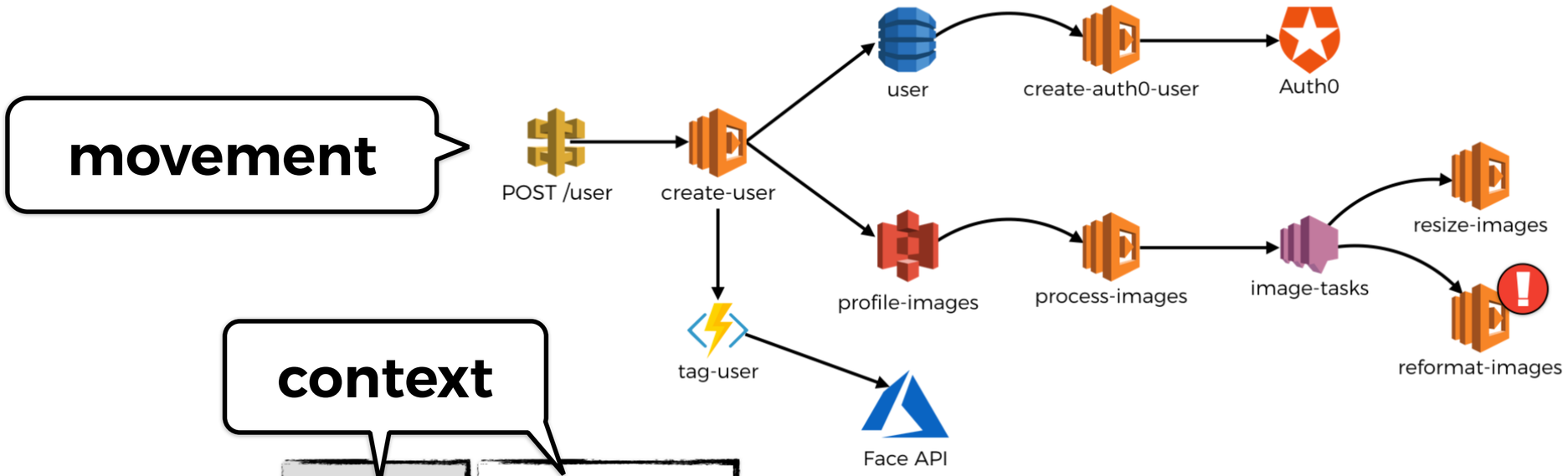
> *However, I would argue that the health of the system no longer matters. We've entered an era where what matters is the health of each individual event, or each individual user's experience, or each shopping cart's experience (or other high cardinality dimensions). With distributed systems you don't care about the health of the system, you care about the* ***health of the event or the slice****.*

- Charity Majors http://bit.ly/2E2QngU

“one user action/vertical slice through the system”



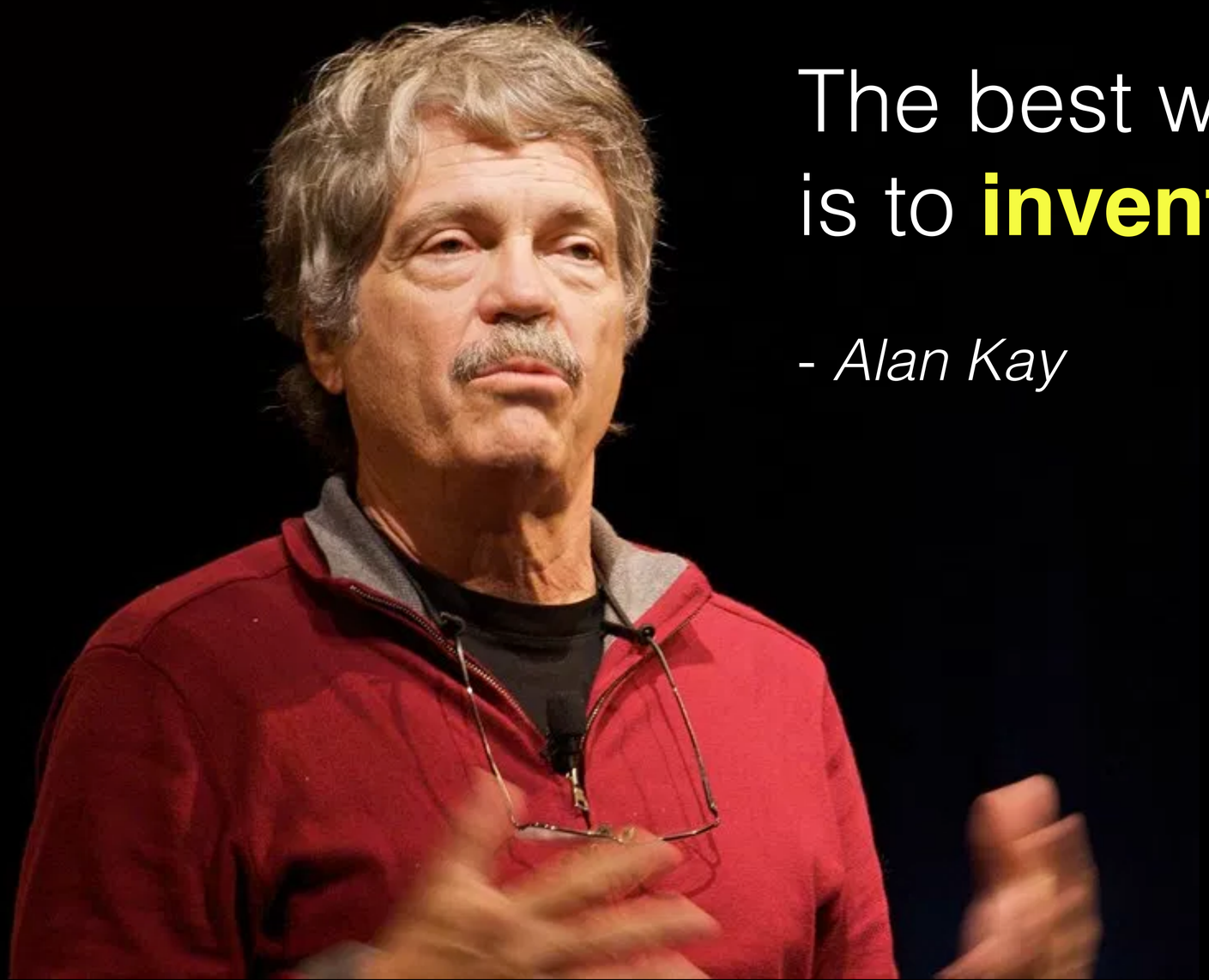




context

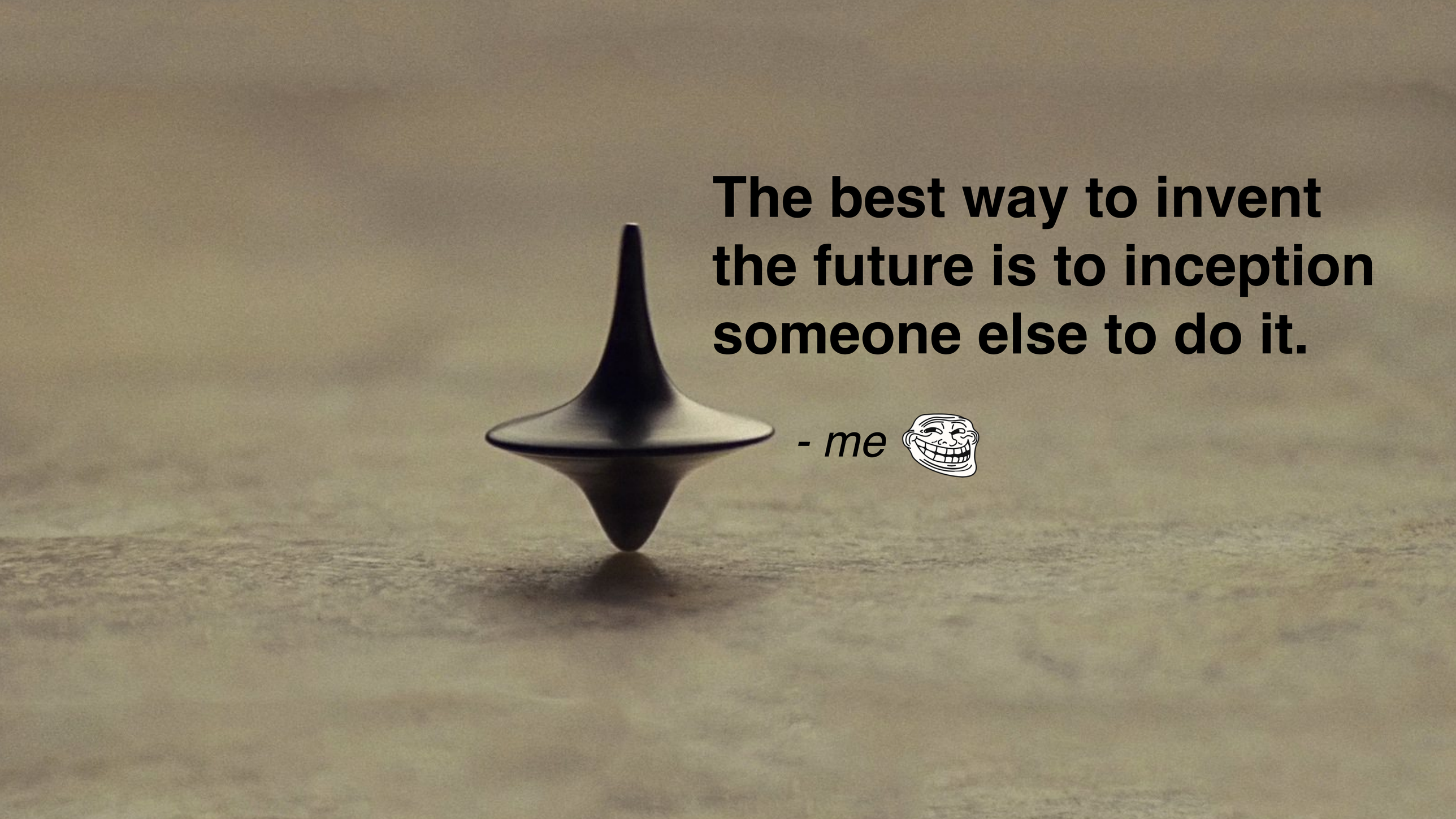
Logs | **Input/Output**

timestamp	component	level	message
2018/01/25 20:51:23.188	POST /user	debug	incoming request...
2018/01/25 20:51:23.201	create-user	debug	saving user [theburningmonk] in the [user] table...
2018/01/25 20:51:23.215	create-user	debug	saved user [theburningmonk] in the [user] table
2018/01/25 20:51:23.521	tag-user	debug	tagging user [theburningmonk] with Azure Face API...



The best way to predict the future is to **invent it**.

- *Alan Kay*



The best way to invent the future is to inception someone else to do it.

- me

FULL OBSERVABILITY FOR AWS LAMBDA

Instrument and profile your functions with zero overhead.
Gain visibility to identify and resolve issues faster.

your e-mail address | Sign up for early access

twitter · blog · community

Serkan Özal
@serkan_ozal

Nitzan Shapira
@nitzanshapira

Ran Ribenzaft
@ranrib

Epsagon

IO|

PRODUCT BLOG GET STARTED SIGN UP SLACK DASHBOARD

SEE INSIDE YOUR LAMBDA FUNCTIONS

Try it Free

Scroll Down to Find Out How

Adam Johnson
@adjohn

Erica Windisch
@ewindisch

Charity Majors
@mipsytipsy

Cindy Sridharan
@copyconstruct

Liz Fong-Jones
@lizthegrey

JBD
@rakyll

Erica Windisch
@ewindisch

Production-Ready Serverless MEAP

Operational Best Practices

API Gateway and Kinesis
Authentication & authorisation (IAM, Cognito)
Testing
Running & Debugging functions locally
Log aggregation
Monitoring & Alerting
X-Ray
Correlation IDs
CI/CD
Performance and Cost optimisation
Error Handling
Configuration management
VPC
Security
Leading practices (API Gateway, Kinesis, Lambda)
Canary deployments

get 40% off with code: ytcui

http://bit.ly/production-ready-serverless