

# Analyze Root Cause and End-User Impact using AWS X-Ray

Abhishek Singh  
General Manager, AWS X-Ray  
June 19<sup>th</sup>, 2019

# Agenda

- Monitoring and debugging modern applications
- AWS X-Ray overview
- Root cause and end-user impact analysis using AWS X-Ray
- Demo
- Q&A

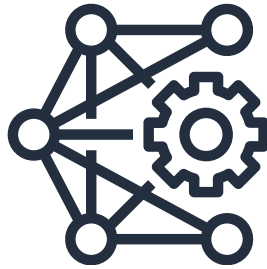


# Monitoring and Debugging Modern Applications

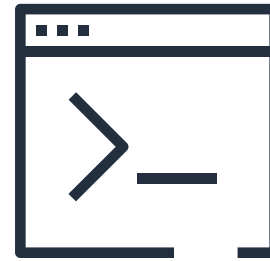
# Monitoring and Debugging Modern Applications



Built on Containers  
and Serverless



Distributed  
Microservices  
Architecture

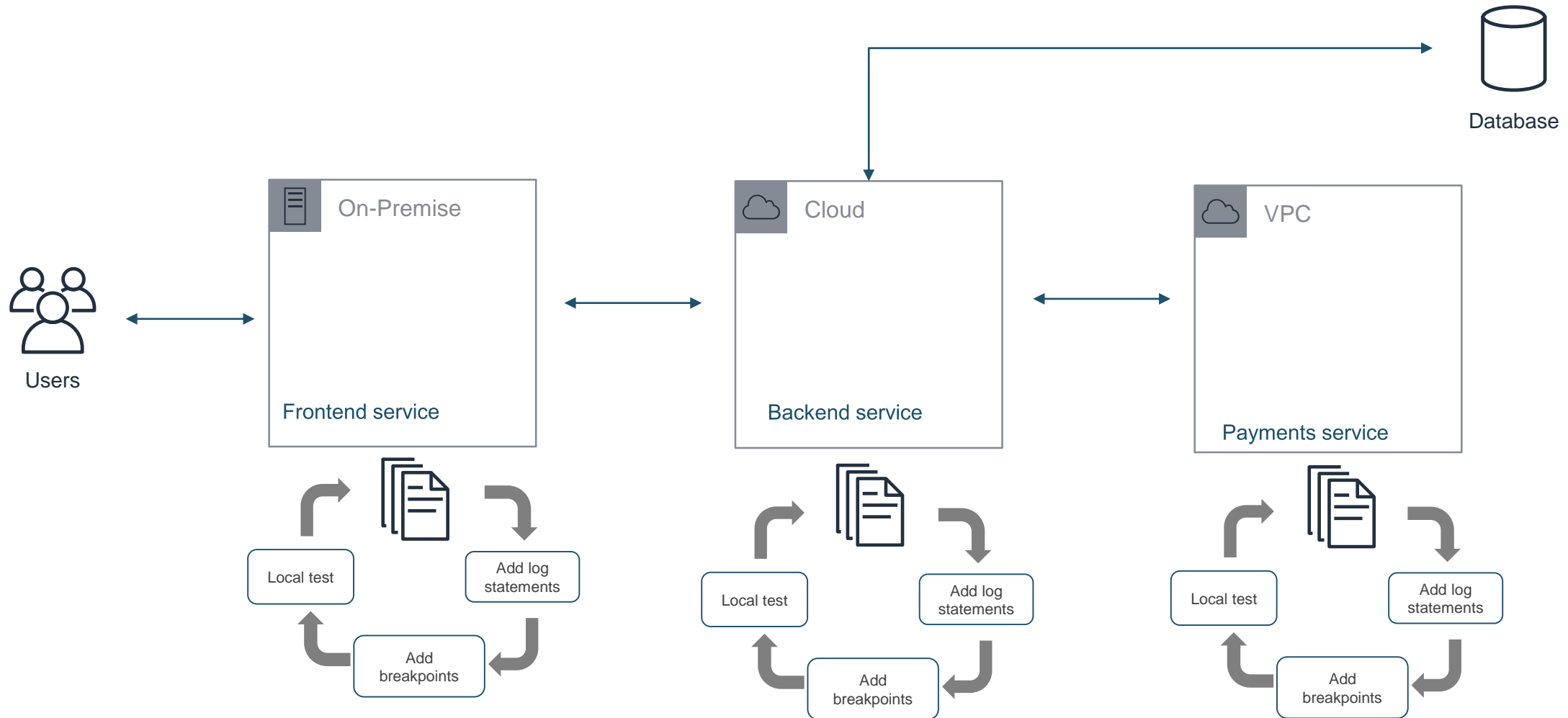


Autonomous and  
Independent  
Components

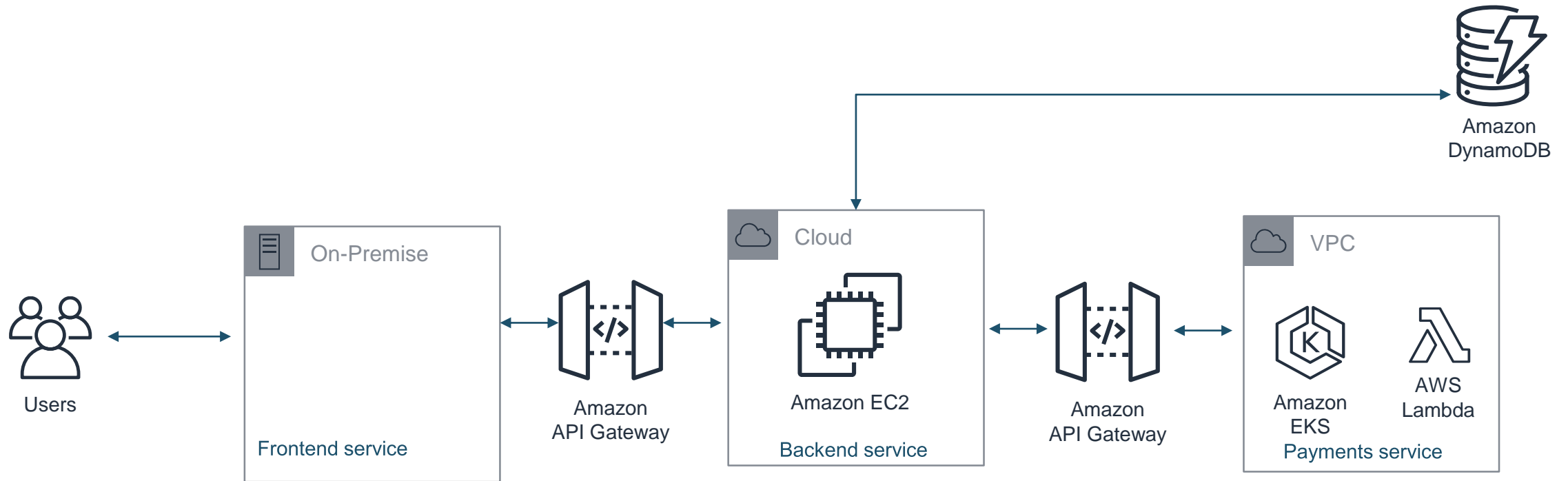


Analyze Impact of  
Isolated Errors

# Traditional Debugging Does Not Scale



# Tracing For Your Modern Applications

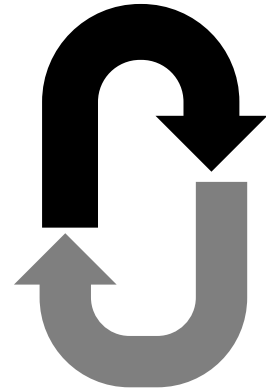


# AWS X-Ray Overview

# AWS X-Ray Overview



Analyze and Debug  
Issues Quickly



End-to-End View of  
Individual Services



Identify Customer  
Impact



Cloud Agnostic

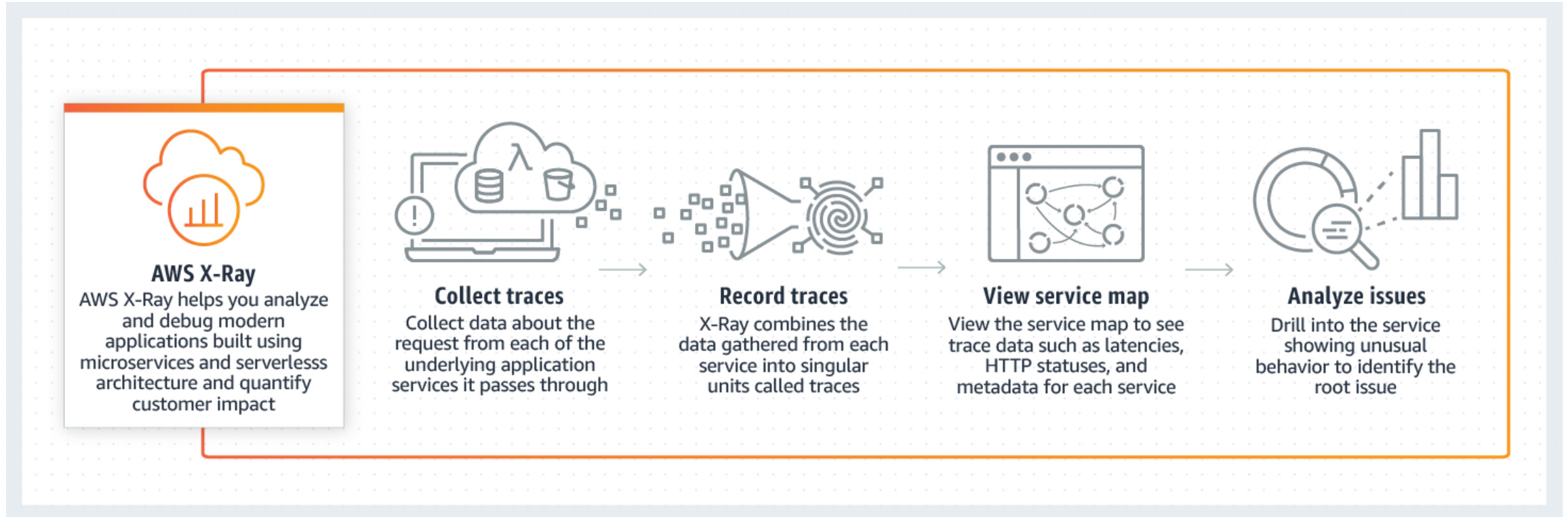


# AWS X-Ray Overview

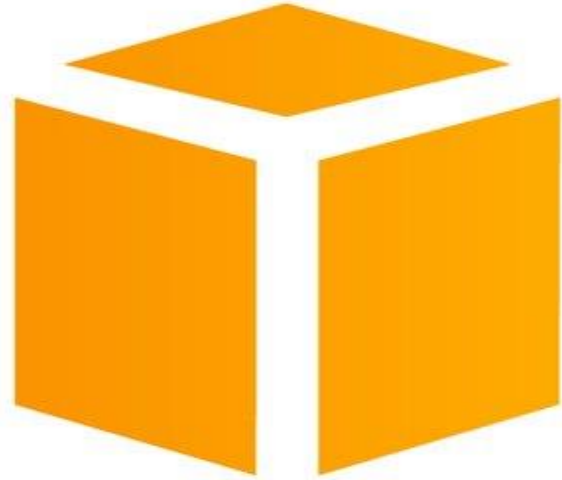


AWS X-Ray helps developers **analyze and debug** applications built using **microservices** architecture to identify the **root cause** and **end-user impact**.

# How AWS X-Ray Works



# AWS X-Ray Instrumentation



Strong Open Source  
Community



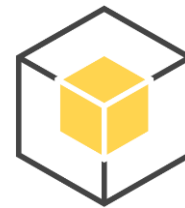
.NET  
(with .NET Core 2.0)



Go (beta)



Java



Python

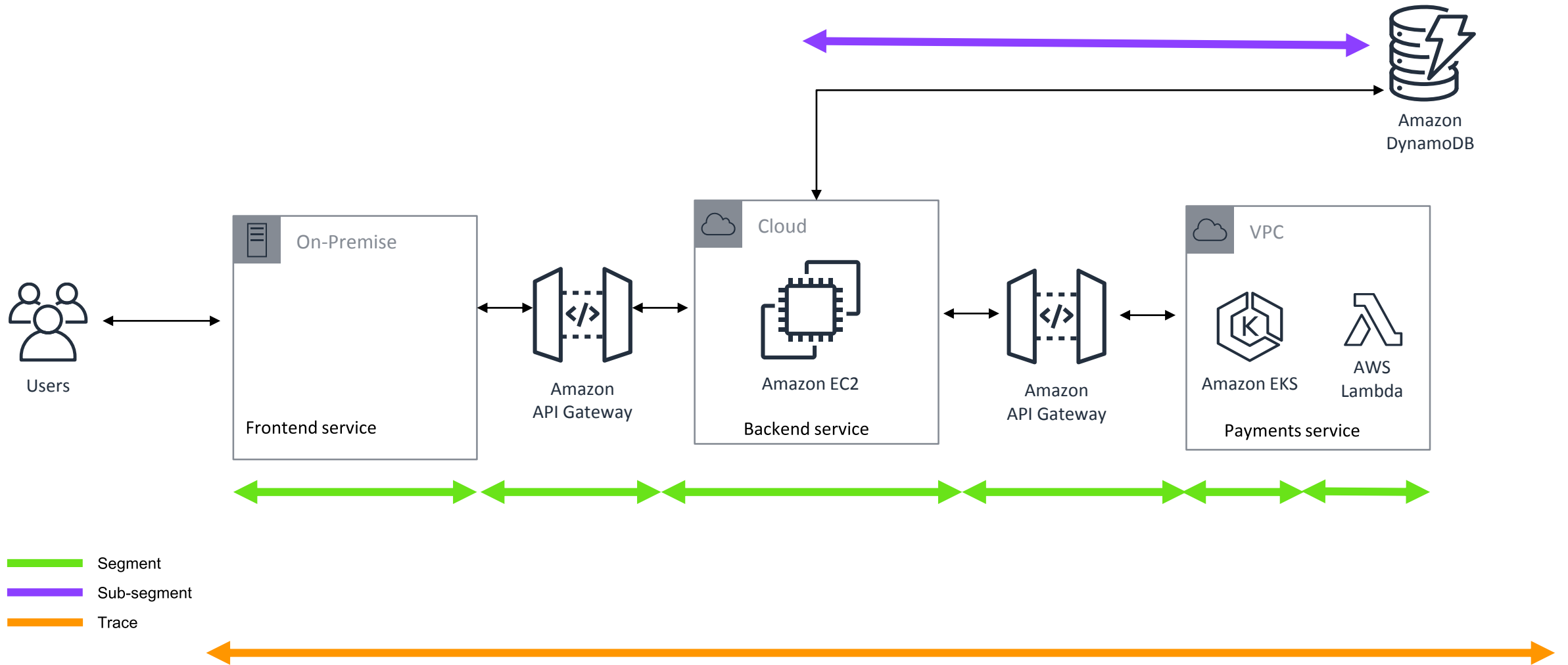


Ruby (beta)

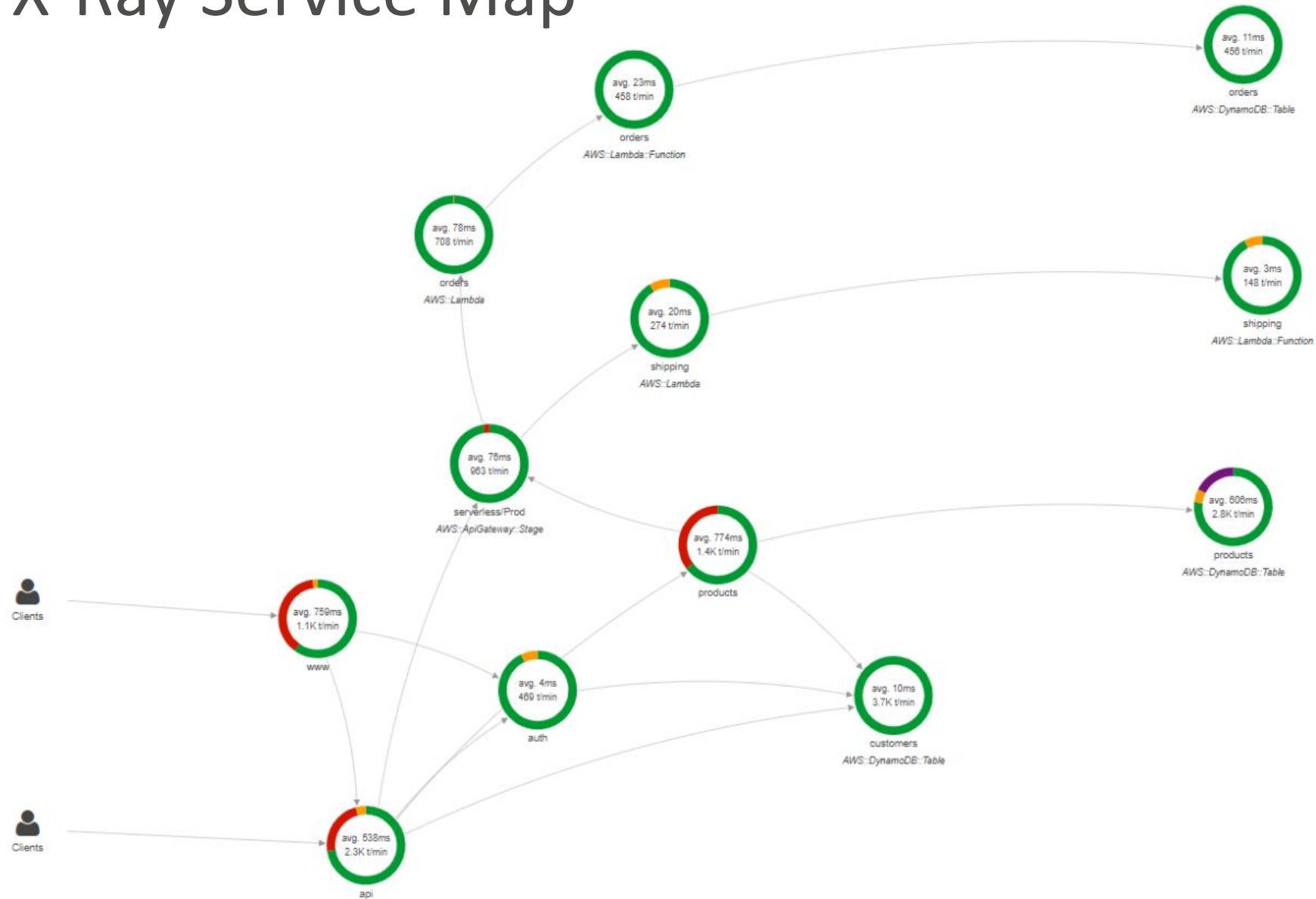


Node.js

# AWS X-Ray for your Modern Applications

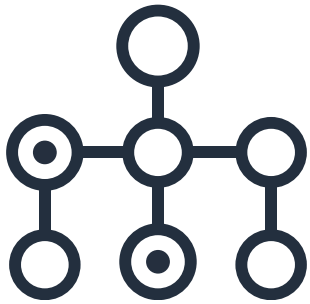


# AWS X-Ray Service Map



# Root Cause and End-User Impact Analysis using AWS X-Ray

# AWS X-Ray Groups



Focus on selected workflows

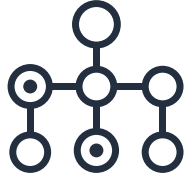


Get alerted on increased error/fault rates



Increase visibility on selected services/end-users

# AWS X-Ray Groups

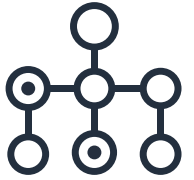


Focus on selected workflows





# AWS X-Ray Groups



Focus on selected workflows



# AWS X-Ray Groups



Get alerted on increased error/fault rates

Group detail ✕

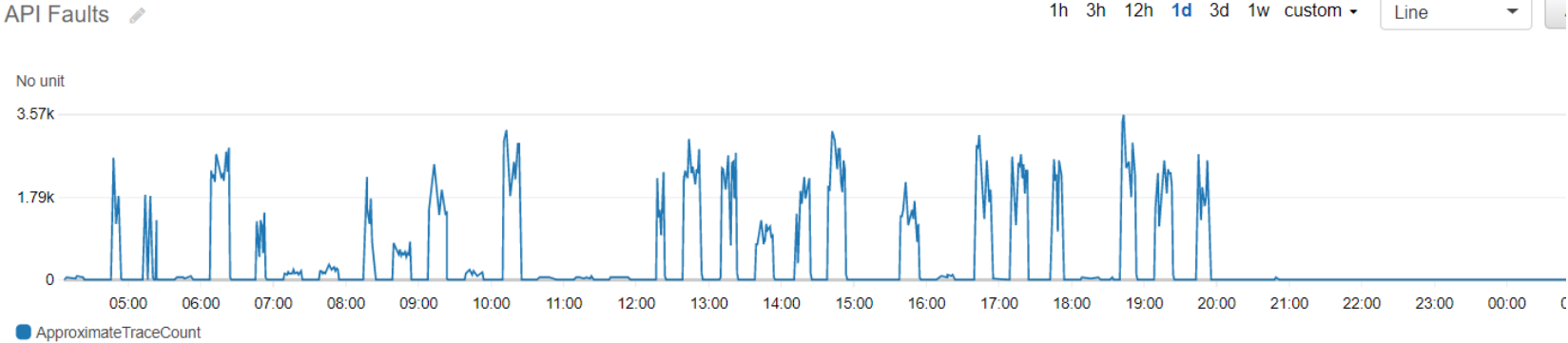
Name fault

ARN `arn:aws:xray:us-east-1:776750115777:xray-group/fault/PQEJKLWH5M7H...`

Filter expression

**Delete**

- CloudWatch
- Dashboards
- Alarms
  - ALARM** 3
  - INSUFFICIENT 0
  - OK 13
- Billing
- Events
- Rules
- Event Buses
- Logs
- Insights
- Metrics**
- Alpine
- Favorites
- [+ Add a dashboard](#)



All metrics		Graphed metrics (1)		Graph options	Source	
<a href="#">+ Add a math expression</a> <a href="#">Dynamic labels</a> <span>Statistic: Sum</span> <span>Period</span>						
<input checked="" type="checkbox"/>	Label	Details			Statistic	Period
<input checked="" type="checkbox"/>	ApproximateTraceCount	X-Ray • ApproximateTraceCount • GroupName: fault			Sum	1 Minute



# AWS X-Ray Groups



Increase visibility on selected services/end-user

Group detail

Name slow

ARN arn:aws:xray:us-east-1:779168132807:group/slow/W3ULUGNWUD4N...

Filter expr

AWS X-Ray

Getting started

Service map

Traces

Analytics new

Configuration

Sampling

Encryption

Delete

slow

service("serverless/Prod") { responsetime > 1.5}

2019/06/04 01:23 - 201

Trace overview

Group by:

URL

Done 10

URL	Avg response time	% of Traces	Response
http://storefront.us-east-1.elasticbeanstalk.com/	1.7 sec	50.00%	2 OK, 0 Throttled, 0 Errors, 0 Faul
http://storefront.us-east-1.elasticbeanstalk.com/error/	3.0 sec	25.00%	0 OK, 0 Throttled, 0 Errors, 1 Faul
http://storefront.us-east-1.elasticbeanstalk.com/slow/	3.0 sec	25.00%	0 OK, 0 Throttled, 0 Errors, 1 Faul

Trace list

ID	Age	Method	Response	Response time	URL	Client IP
...f357dc91	2.4 hr	GET	200	1.6 sec	http://storefront.us-east-1.elasticbeanst...	18.213.102.12
...f1c1c029	2.5 hr	GET	200	1.8 sec	http://storefront.us-east-1.elasticbeanst...	18.213.102.12
...f6add96	2.8 hr	GET	500	3.0 sec	http://storefront.us-east-1.elasticbeanst...	18.213.102.12
...108fbe28	2.8 hr	GET	500	3.0 sec	http://storefront.us-east-1.elasticbeanst...	18.213.102.12

# AWS X-Ray Analytics



Understand End-User Impact



Analyze Root Cause



Compare Trends

# AWS X-Ray Analytics



Default  Last 5 minutes

All traces in group 9.7K traces in the group. [Show in charts](#)

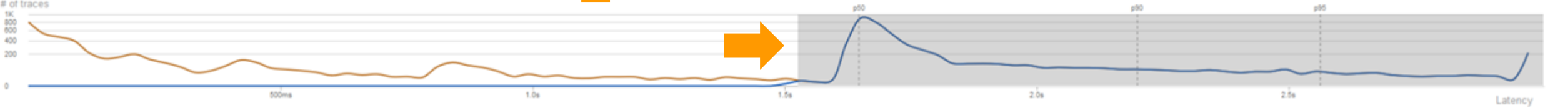
Retrieved traces 10.2K traces.

Filtered trace set A 5358 traces (52.76% of retrieved traces)  
Filters: **responsetime** [Refine](#)

[+ Compare](#)  
(Copy filter trace set A)

## Response time distribution

Click and drag to filter the traces by response time



## Time series activity

Click and drag to filter the traces by time



Select rows from below tables to filter traces. Tables below are configurable. Click on the cog icons to explore options

USER	COUNT	%
Emma	2070	38.63%
Olivia	1092	20.38%
Ava	370	6.91%
William	601	11.22%
Mason	360	6.72%
Sophia	357	6.66%
Liam	371	6.92%
Mark	137	2.56%

HTTP STATUS CODE	COUNT	%
200	791	14.76%
500	4542	84.77%
502	25	0.47%



# AWS X-Ray Analytics



Analyze Root Cause

RESPONSE TIME ROOT CAUSE	COUNT	%
api → forward → products.us-east-1.elasticbeanstalk.com ⇒ products → fetch → DynamoDB → wait	1673	36.83%
www → products → store-api.us-east-1.elasticbeanstalk.com ⇒ api → forward → products.us-east-1.elasticbeanstalk.com ⇒ products → fetch → DynamoDB → wait	2566	56.49%
api → forward → products.us-east-1.elasticbeanstalk.com	75	1.65%
www → products → store-api.us-east-1.elasticbeanstalk.com	108	2.38%
www → products → store-api.us-east-1.elasticbeanstalk.com → response	41	0.90%
www → products → store-api.us-east-1.elasticbeanstalk.com ⇒ api → forward → products.us-east-1.elasticbeanstalk.com	2	0.04%
www → products → store-api.us-east-1.elasticbeanstalk.com ⇒ api → forward → products.us-east-1.elasticbeanstalk.com ⇒ products → fetch → DynamoDB	7	0.15%
www → products → store-api.us-east-1.elasticbeanstalk.com ⇒ api → forward → products.us-east-1.elasticbeanstalk.com → response	18	0.40%

Trace List	USERS	RESPONSE	RESPONSE TIME
http://store-api.us-east-1.elasticbeanstalk.com/products/	Emma	500	1.729
http://store-api.us-east-1.elasticbeanstalk.com/products/	William	500	1.69
http://storefront.us-east-1.elasticbeanstalk.com/products/	Mason	500	1.757
http://storefront.us-east-1.elasticbeanstalk.com/	Olivia	500	1.665
http://store-api.us-east-1.elasticbeanstalk.com/products/	William	500	1.706

FAULT ROOT CAUSE	COUNT	%
api → error ⇒ products → error → error → error → error ⇒ products (AWS::DynamoDB::Table)	1297	
api → error ⇒ products → error → error → error → error → error ⇒ products (AWS::DynamoDB::Table)	225	
www → error → error ⇒ api → error ⇒ products → error → error → error → error ⇒ products (AWS::DynamoDB::Table)	1996	

FAULT ROOT CAUSE MESSAGE	COUNT	%
ProvisionedThroughputExceededException: The level of configured provisioned throughput for the table was exceeded. Consider increasing your provisioning level with the UpdateTable API. status code: 4	4437	97.69%
http request to products failed with status code: 500	37	0.81%
http request to api failed with status code: 502	14	0.31%



# AWS X-Ray Analytics



Compare Trends

Retrieved traces ⓘ

10.3K traces.

Filtered trace set A ⓘ

1899 traces.(18.45% of retrieved traces)

Filters: timerange

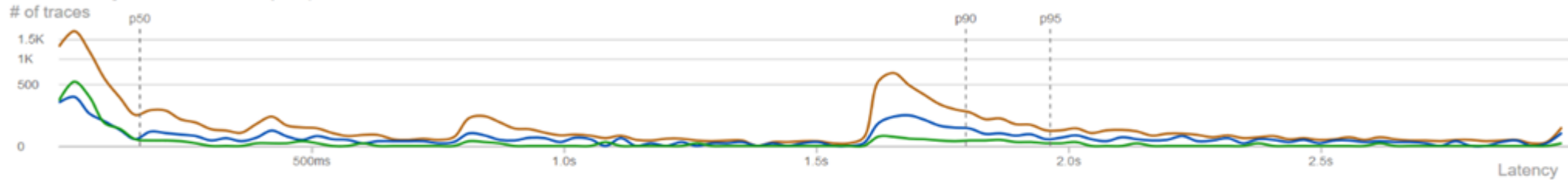
Filtered trace set B ⓘ

1354 traces.(13.15% of retrieved traces)

Filters: timerange

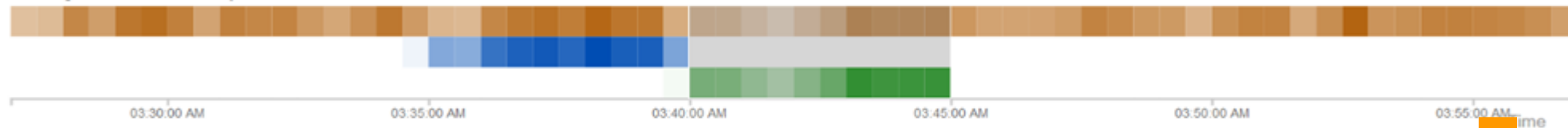
## Response time distribution ⓘ

Click and drag to filter the traces by response time



## Time series activity ⓘ

Click and drag to filter the traces by time



USER	COUNT	%	Δ
William	189	13.96%	0.00%
Olivia	279	20.61%	0.17%
Sophia	88	6.50%	0.23%
Mason	98	7.24%	0.76%
Emma	433	31.98%	-0.41%
Noah	74	5.47%	0.62%

HTTP STATUS CODE	COUNT	%	Δ
200	1218	89.96%	32.87%
400	16	1.18%	0.39%
401	22	1.62%	-0.22%
500	97	7.16%	-32.01%
-	1	0.07%	-0.19%

# Demo





# Thank you!