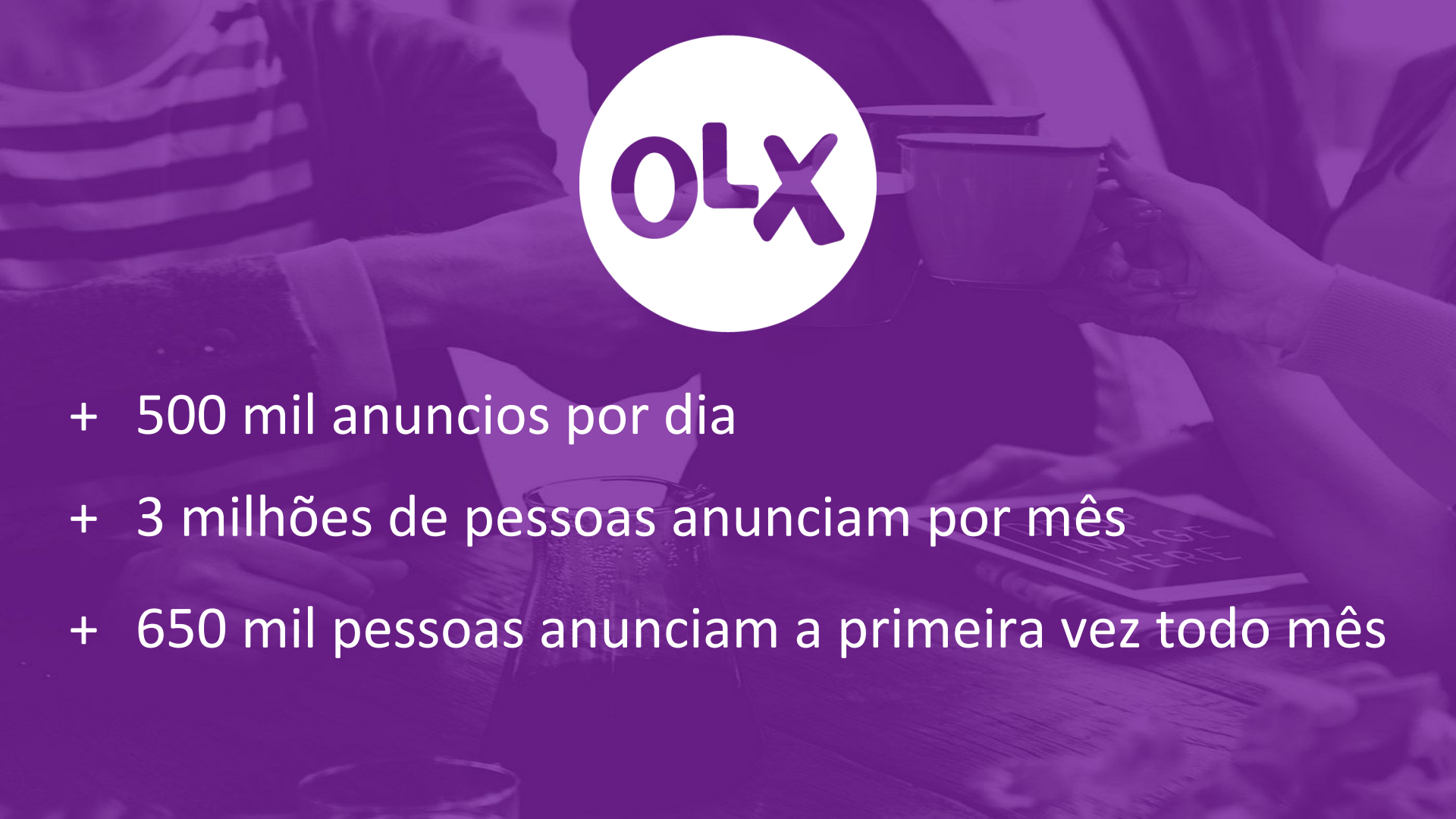
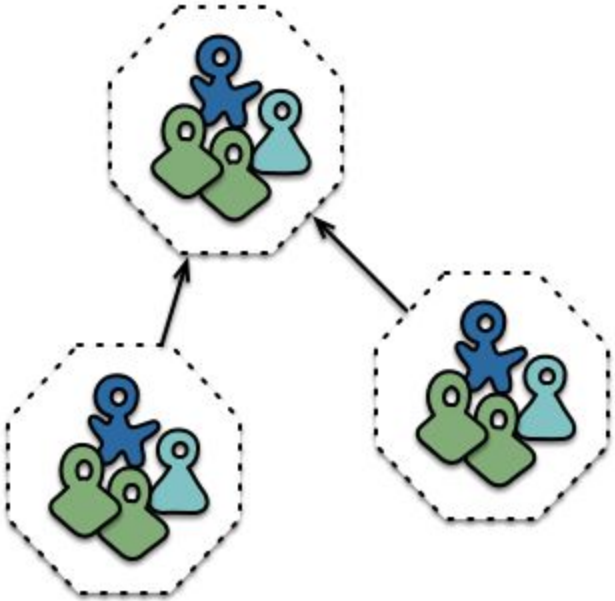
Observability: Você sabe exatamente o que seus Microservices andam aprontando?

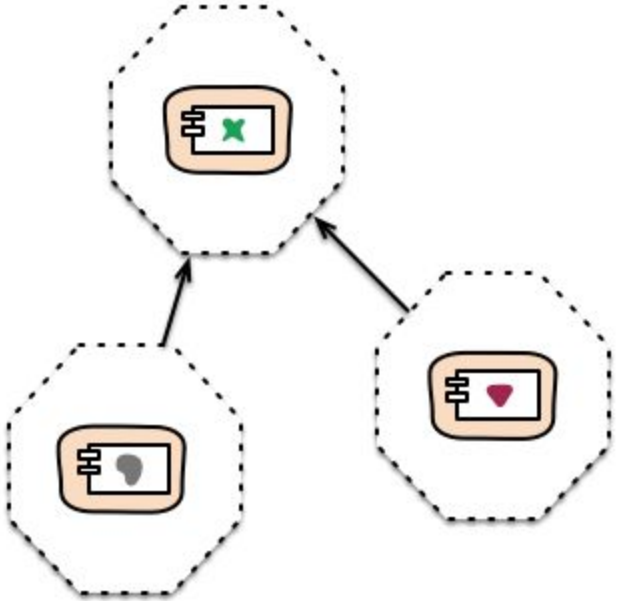
Sinval Vieira



+ 500 mil anuncios por dia
+ 3 milhões de pessoas anunciam por mês
+ 650 mil pessoas anunciam a primeira vez todo mês



Cross-functional teams...



... organised around capabilities
Because Conway's Law

Arquitetura

"Microservices must be small enough to fit in my head"

- James Lewis


OBSERVABILITY
COM
SEM

Contexto

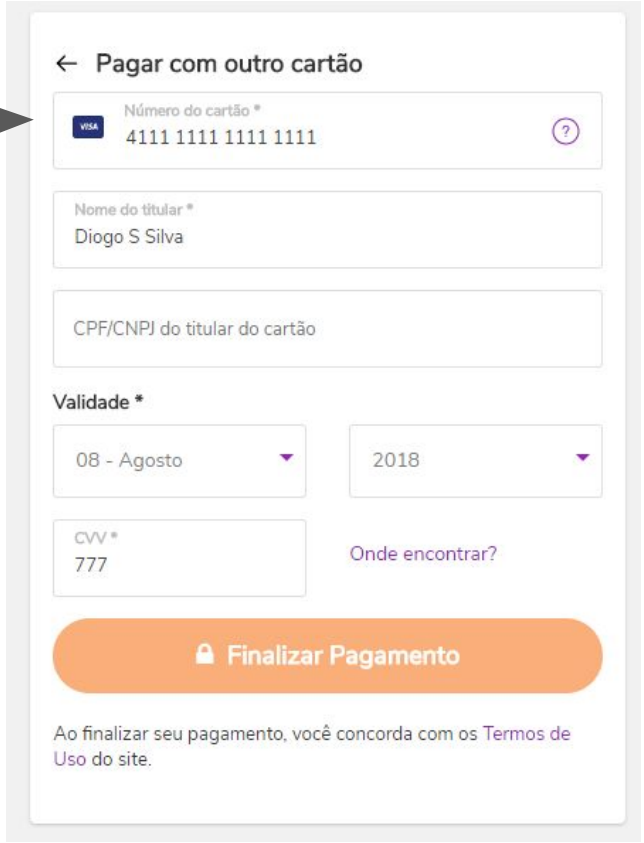
3 Devs

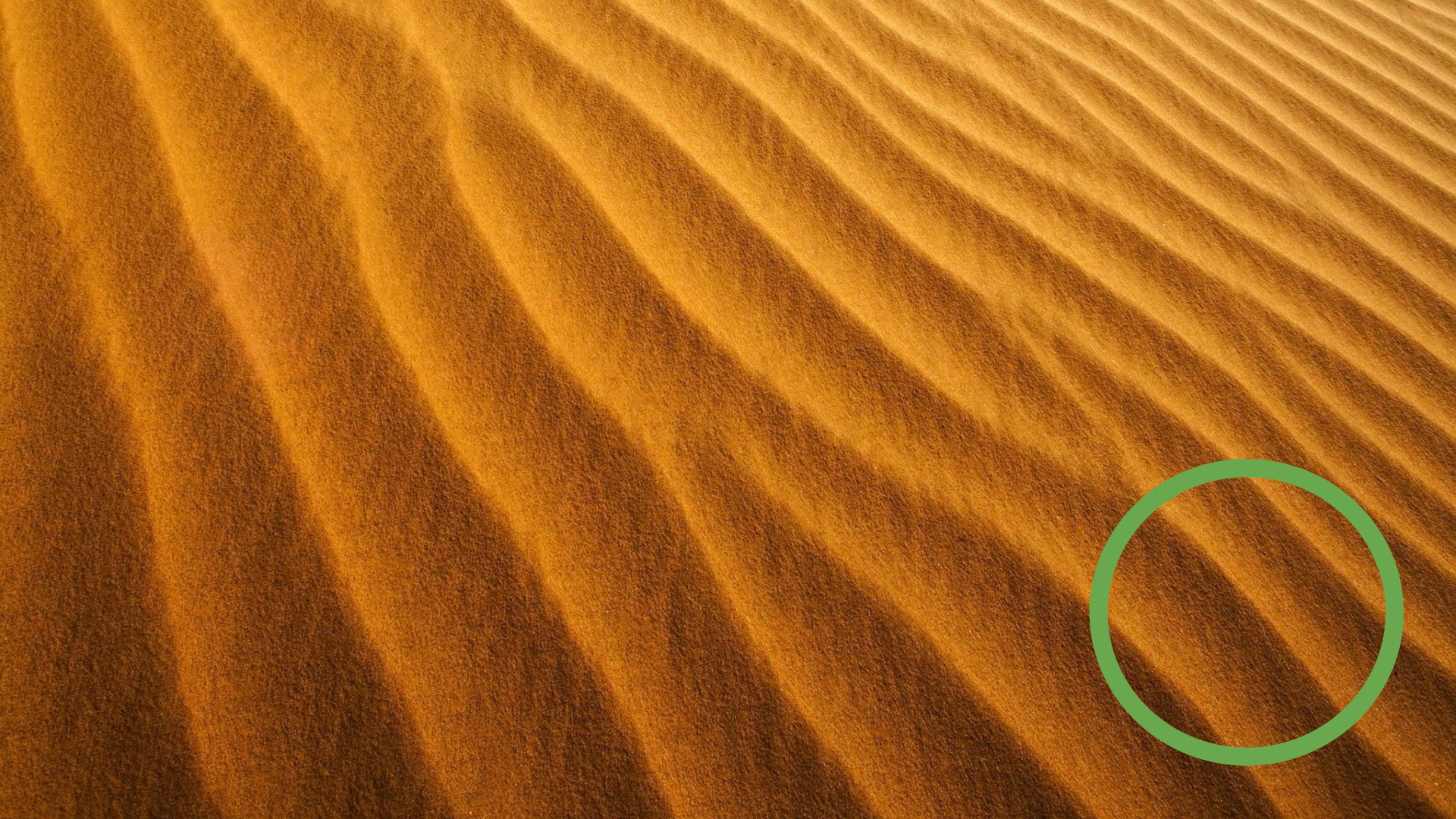
Product Owner

Gerente de Engenharia
(Vegan)

Contexto


MINHAS COMPRAS
1 Destaque Ouro
TOTAL
R$ 26,99
Finalizar
Até 16x mais visualizações!
PUBLICADO
Bolo de casamento
3 visitas - 18/07/2018 . 16:44
Editar
Já vendi
Excluir
DESTAQUE BÁSICO
R$ 8,99
Como funciona?
Turbinar
DESTAQUE PRATA
R$ 17,99
Como funciona?
Turbinar
DESTAQUE OURO
R$ 26,99
Como funciona?
Selecionado
DESTAQUE DIAMANTE
R$ 57,99
Como funciona?
Turbinar


Pagar com outro cartão
Número do cartão *
4111 1111 1111 1111
Nome do titular *
Diogo S Silva
CPF/CNPJ do titular do cartão
Validade *
08 - Agosto
2018
CVV *
777
Onde encontrar?
Finalizar Pagamento
Ao finalizar seu pagamento, você concorda com os Termos de Uso do site.



Payments

- Entregas ágeis
- Independente
- DevOps
- Desacoplado do monolito
- Microservices
- Serverless

Desafios

OBSERVABILITY

"In control theory, observability is a measure of how well internal states of a system can be inferred from knowledge of its external." - WIKIPEDIA

Tornar sistemas complexos transparentes o suficiente para todos que os operam e os desenvolvem.

OBSERVABILITY ← Logging + Monitoring + Tracing + Visualization

Observability at Twitter: technical overview, part I

By @anthonyjasta
Friday, 18 March 2016

The Observability Engineering team at Twitter provides full-stack libraries and multiple services to our internal engineering teams to monitor service health, alert on issues, support root cause investigation by providing distributed systems call traces, and support diagnosis by creating a searchable index of aggregated application/system logs. These are the four pillars of the Observability Engineering team's charter:

- Monitoring
- Alerting/visualization
- Distributed systems tracing infrastructure
- Log aggregation/analytics

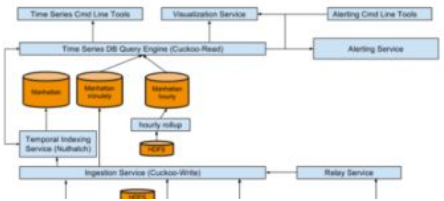
We fulfill this charter for all applications and services running in our owned and operated data center as well as acquired properties that are deployed in external public clouds, such as Amazon AWS and Google Compute.

Twitter services handle a large number of tweets every day. Responsible for providing visibility into the services powering Twitter, our observability services are some of the largest scale internal systems. The request volume and data size on disk have grown tremendously since we last discussed this in a blog post two years ago. Today, our time series metric ingestion service handles more than 2.8 billion write requests per minute, stores 4.5 petabytes of time series data, and handles 25,000 query requests per minute.

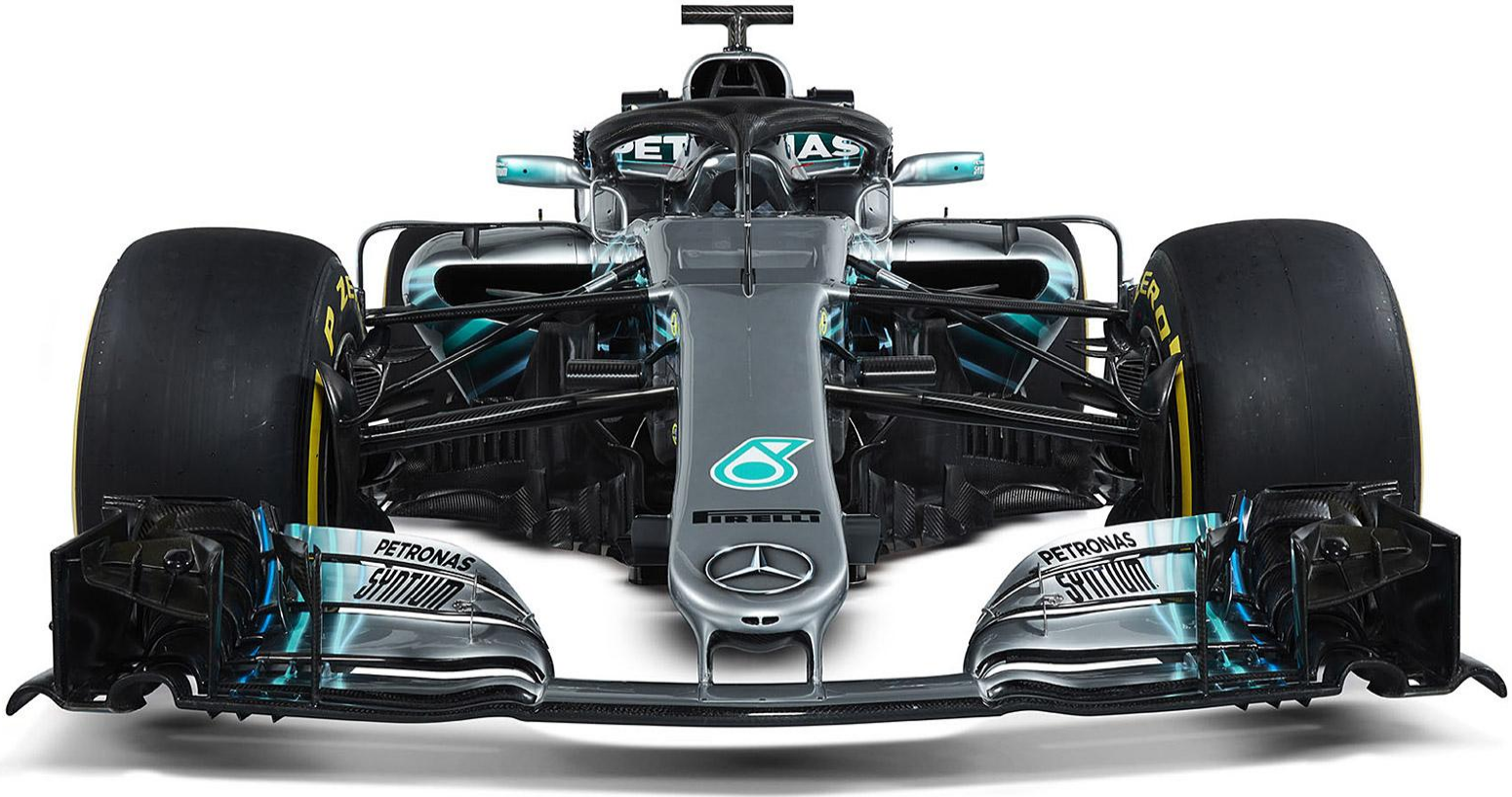
This blog is the first in a two-part series. It covers our architecture, metrics ingestion, time series database, and indexing services.

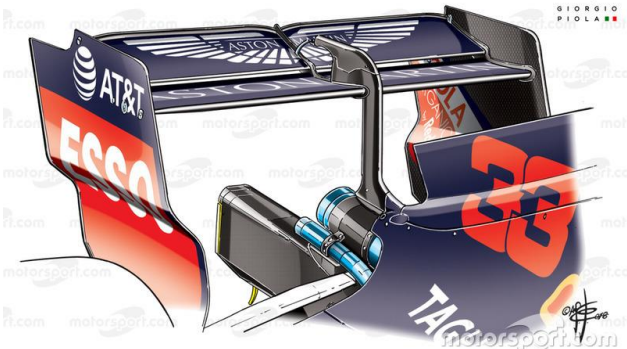
Architecture

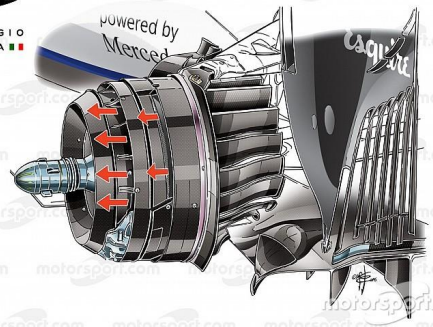
Our system architecture has also changed quite a bit over the past two years . Here is our current architecture:

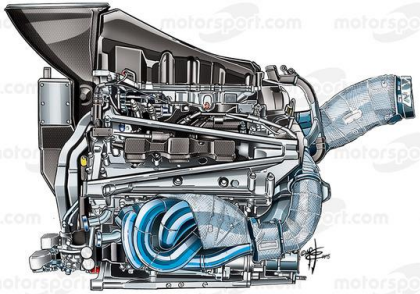


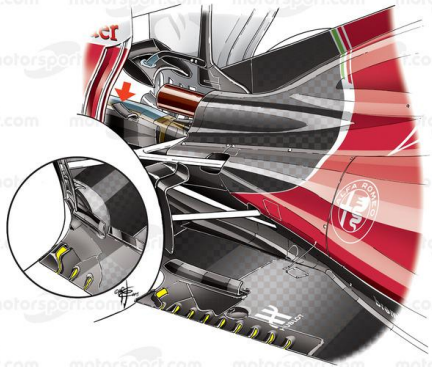
1 EXEMPLO

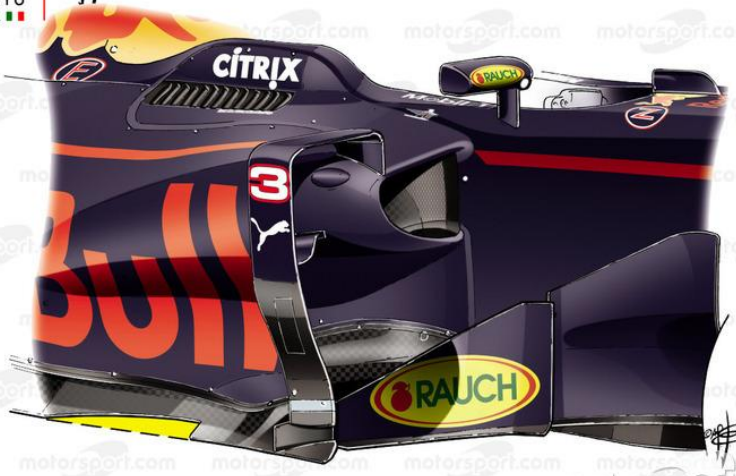

PETRONAS
SYNTIUM
PETRONAS
SYNTIUM
PIRELLI


AT&T
ESSO
33


Powered by
Merced






CITRIX
RAUCH
3
RAUCH

OBSERVABILITY ← Logging + Monitoring + Tracing + Visualization

Logging

O primeiro passo: 5W's

- **W**HO
- **W**HAT
- **W**HEN
- **W**HERE
- **W**HY

O segundo passo:

INFO

DEBUG

WARNING

ERROR

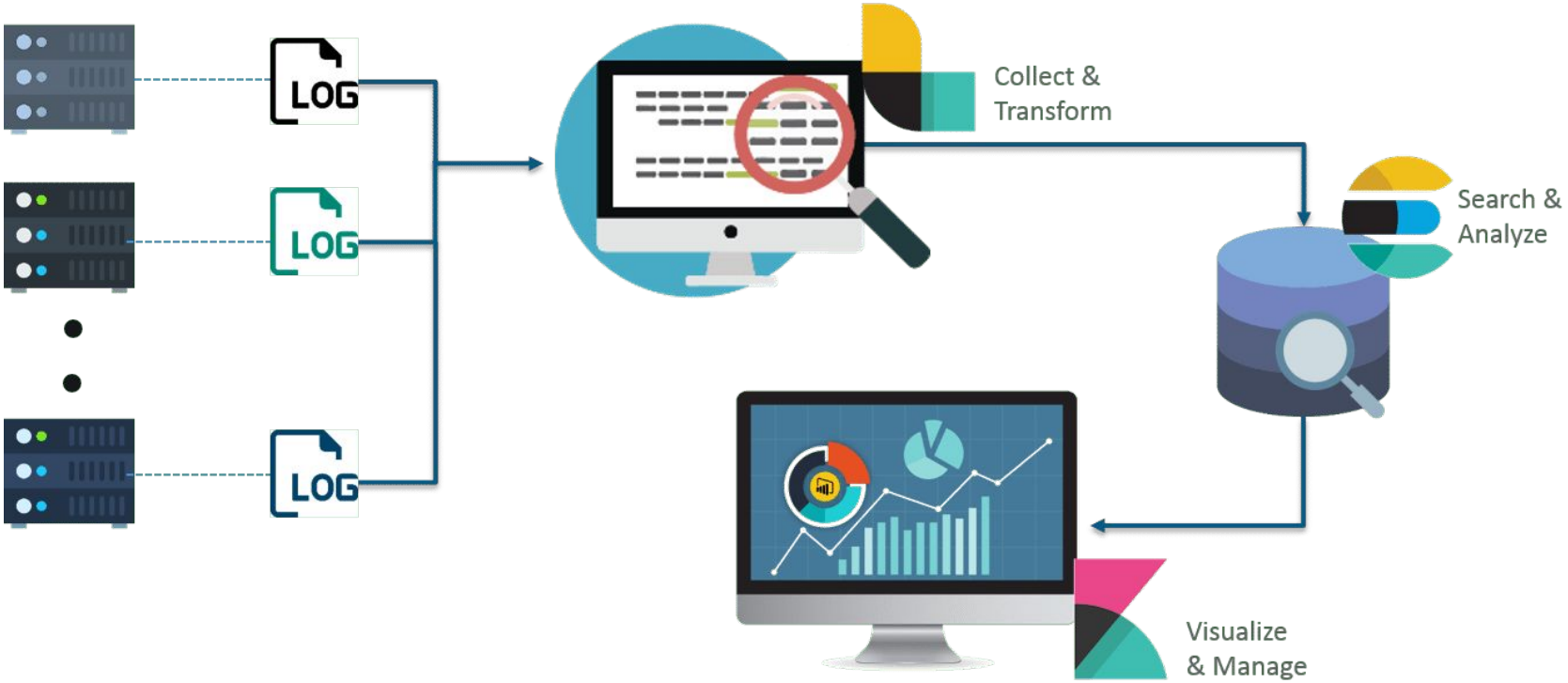
Logs

Sinval Vieira
Nov 21 · 14 min read

Log é uma mensagem primordialmente utilizada para registro de um acontecimento na execução de um sistema. O principal uso de logs é depuração de comportamento para identificar problemas e suas causas. Muitos engenheiros de software não sabem como escrever logs eficazes e também não têm noção do que deve ser colocado numa mensagem. O objetivo deste texto é descrever boas práticas para extrair o máximo de valor do log. Vamos discutir desde o entendimento da anatomia de uma mensagem, ao processo de escrita, descrição, e compreensão de suas diferentes aplicações no cenário moderno de desenvolvimento, operação e monitoria de software.

Por que logar?

O principal objetivo de uma mensagem de log é informar a um terceiro (na maioria dos casos será desenvolvedor ou operador) sobre informações detalhadas da execução de um programa. Quando a aplicação está em


LOG
LOG
LOG
Collect &
Transform
Search &
Analyze
Visualize
& Manage

fluentd

Monitoring

Fontes de Dados Valiosas

- Logs
- Métricas
- Application Performance Management (APM)


New
Relic.

Tracing

Dapper, a Large-Scale Distributed Systems Tracing Infrastructure

Venue

Google, Inc. (2010)

Authors

Benjamin H. Sigelman

Luiz André Barroso

Mike Burrows

Pat Stephenson

Manoj Plakal

Donald Beaver

Saul Jaspan

Chandan Shanbhag

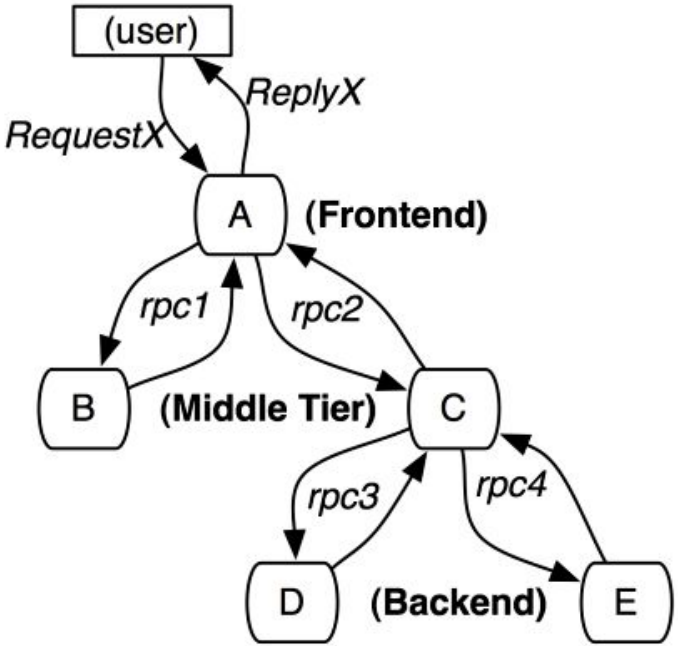
Research Areas

Distributed Systems and Parallel Computing

Abstract

Modern Internet services are often implemented as complex, large-scale distributed systems. These applications are constructed from collections of software modules that may be developed by different teams, perhaps in different programming languages, and could span many thousands of machines across multiple physical facili- ties. Tools that aid in understanding system behavior and reasoning about performance issues are invaluable in such an environment.

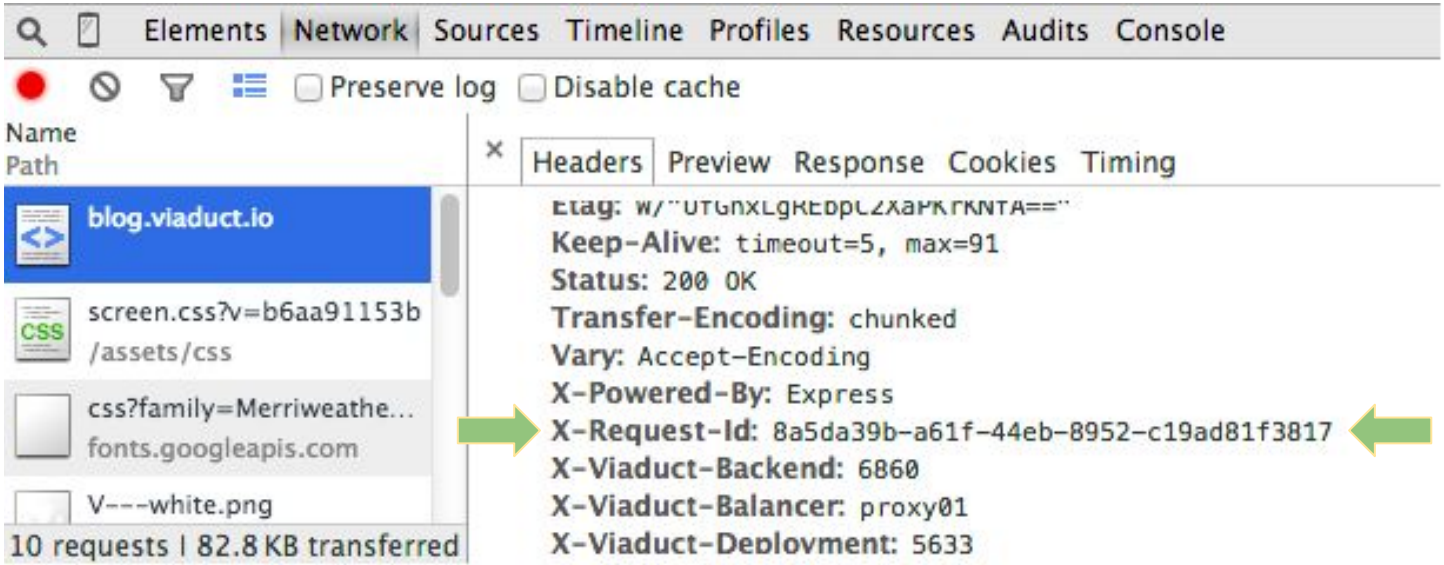
Here we introduce the design of Dapper, Google's production distributed systems tracing infrastructure, and describe how our design goals of low overhead, application-level transparency, and ubiquitous deployment on a very large scale system were met. Dapper shares conceptual similarities with other tracing systems, particularly Magpie [3] and X-Trace [12], but certain design choices were made that have been key to its success in our environment, such as the use of sampling and restricting the instrumentation to a rather small number of common libraries.


(user)
ReplyX
RequestX
A
(Frontend)
rpc1
rpc2
B
(Middle Tier)
C
rpc3
rpc4
D
(Backend)
E

+


New
Relic.

X-Request-ID


Elements
Network
Sources
Timeline
Profiles
Resources
Audits
Console
Preserve log
Disable cache
Name
Path
blog.viaduct.io
screen.css?v=b6aa91153b
/assets/css
css?family=Merriweathe...
fonts.googleapis.com
V---white.png
10 requests | 82.8 KB transferred
Headers
Preview
Response
Cookies
Timing
Keep-Alive: timeout=5, max=91
Status: 200 OK
Transfer-Encoding: chunked
Vary: Accept-Encoding
X-Powered-By: Express
X-Request-Id: 8a5da39b-a61f-44eb-8952-c19ad81f3817
X-Viaduct-Backend: 6860
X-Viaduct-Balancer: proxy01
X-Viaduct-Deployment: 5633

Microservice A

[1257016e-9ec5-41bf-9f6e-02a3aa611ed5]

[2017-09-02 00:11:57.557] [INFO] MENSAGEM DO **MICROSERVICE A**

Microservice B

[1257016e-9ec5-41bf-9f6e-02a3aa611ed5,49bc7875-e678-4351-b245-34c88fa9b54f]

[2017-09-02 00:11:57.604] [INFO] MENSAGEM DO **MICROSERVICE B**

Microservice C

[1257016e-9ec5-41bf-9f6e-02a3aa611ed5,49bc7875-e678-4351-b245-34c88fa9b54f,
343e68df-fad9-4cee-8eb8-5349f5c304f9]

[2017-09-02 00:11:57.698] [INFO] MENSAGEM DO **MICROSERVICE C**

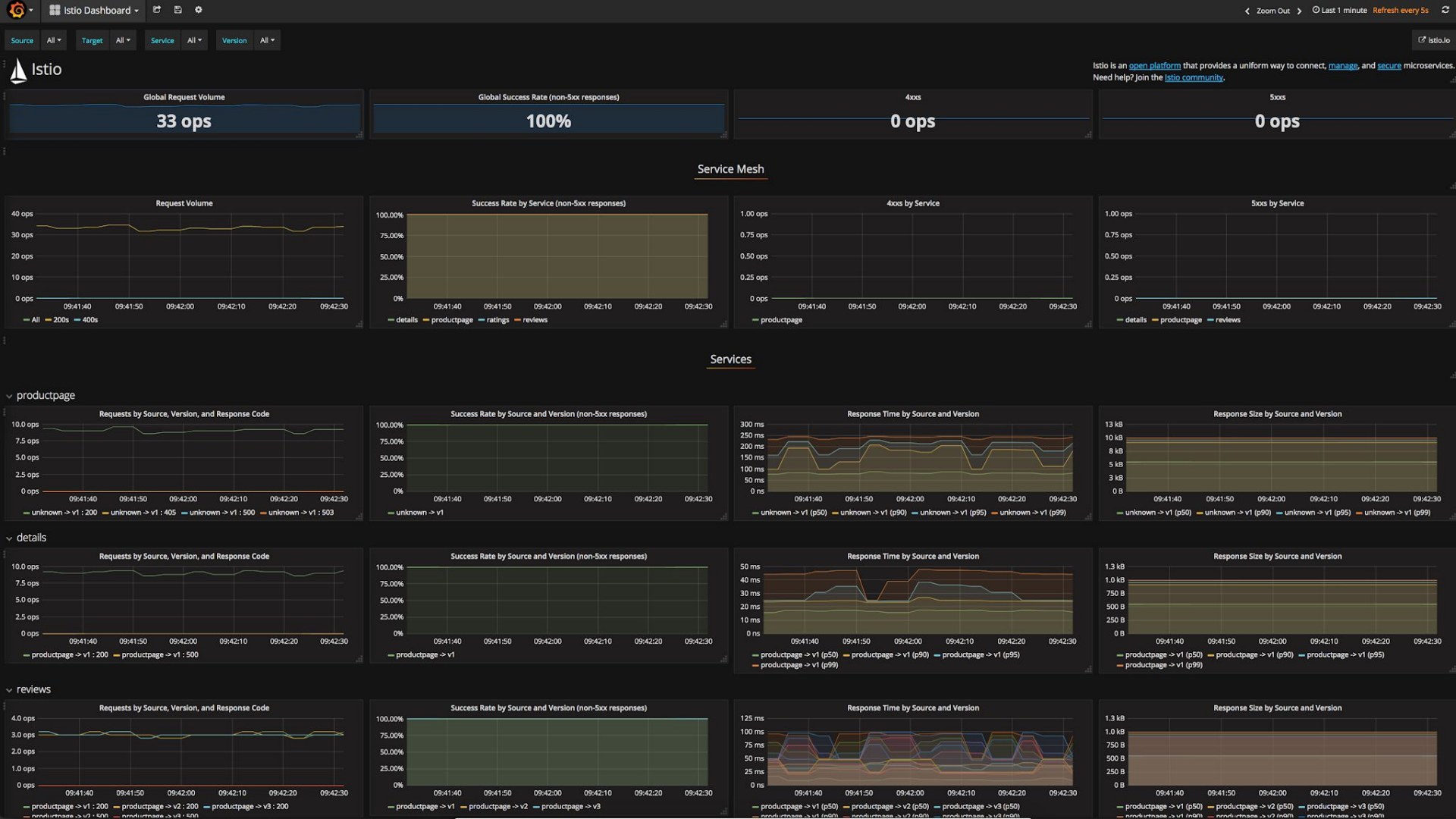
Visualization

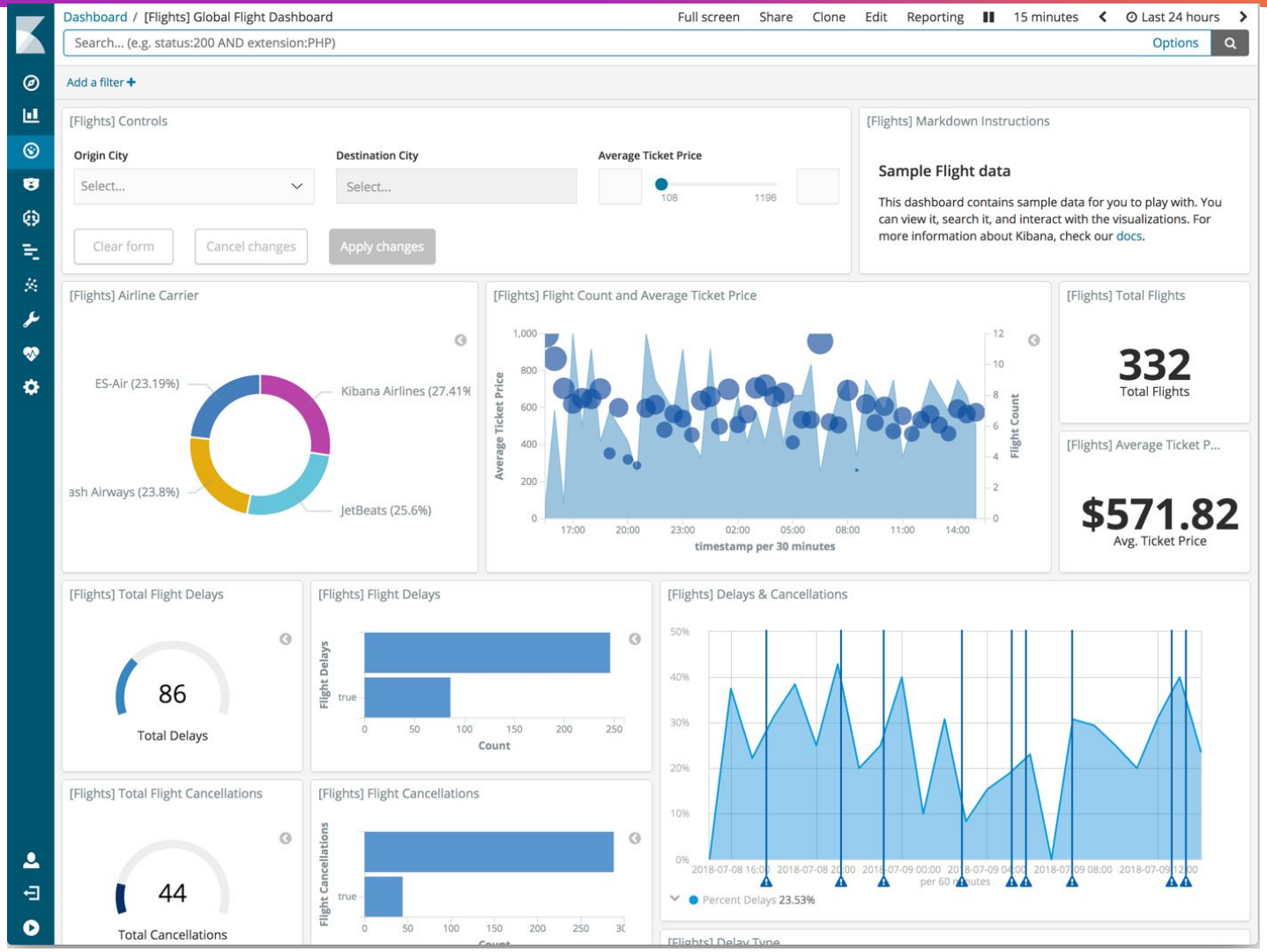

Red Bull
FINITI

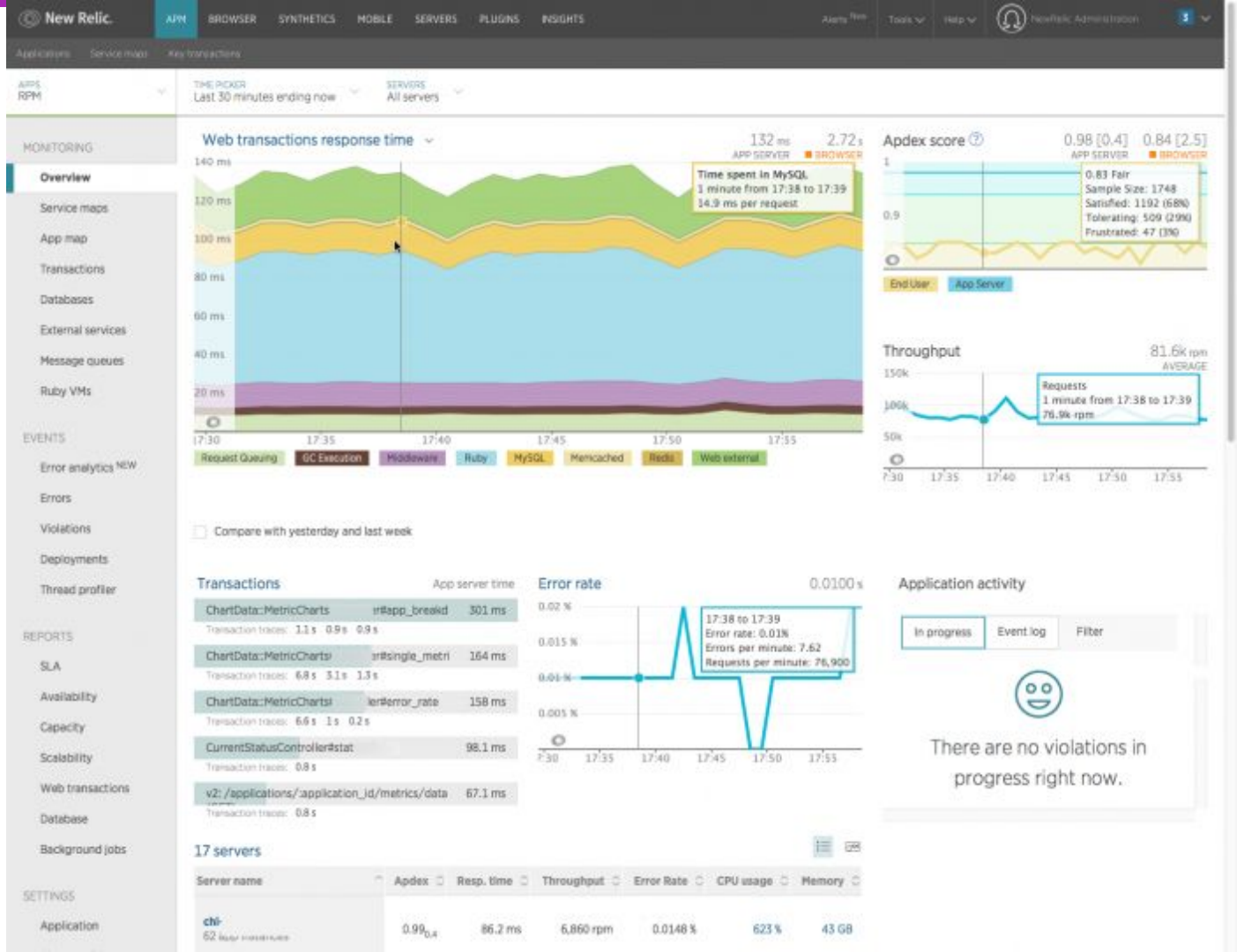

Silanna
Emil Frey
9
motorsport.com

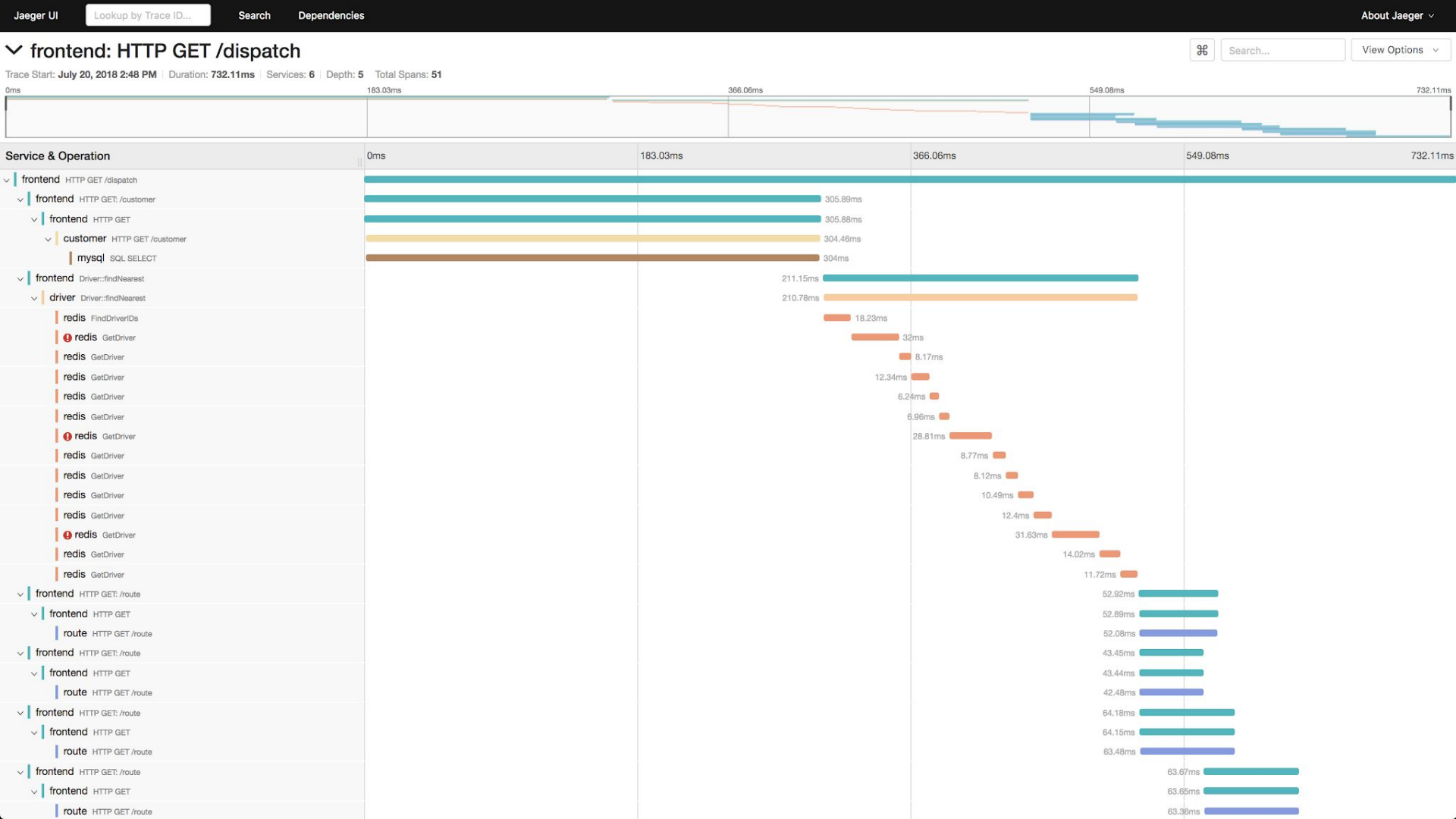

HHONORS
motorsport.com

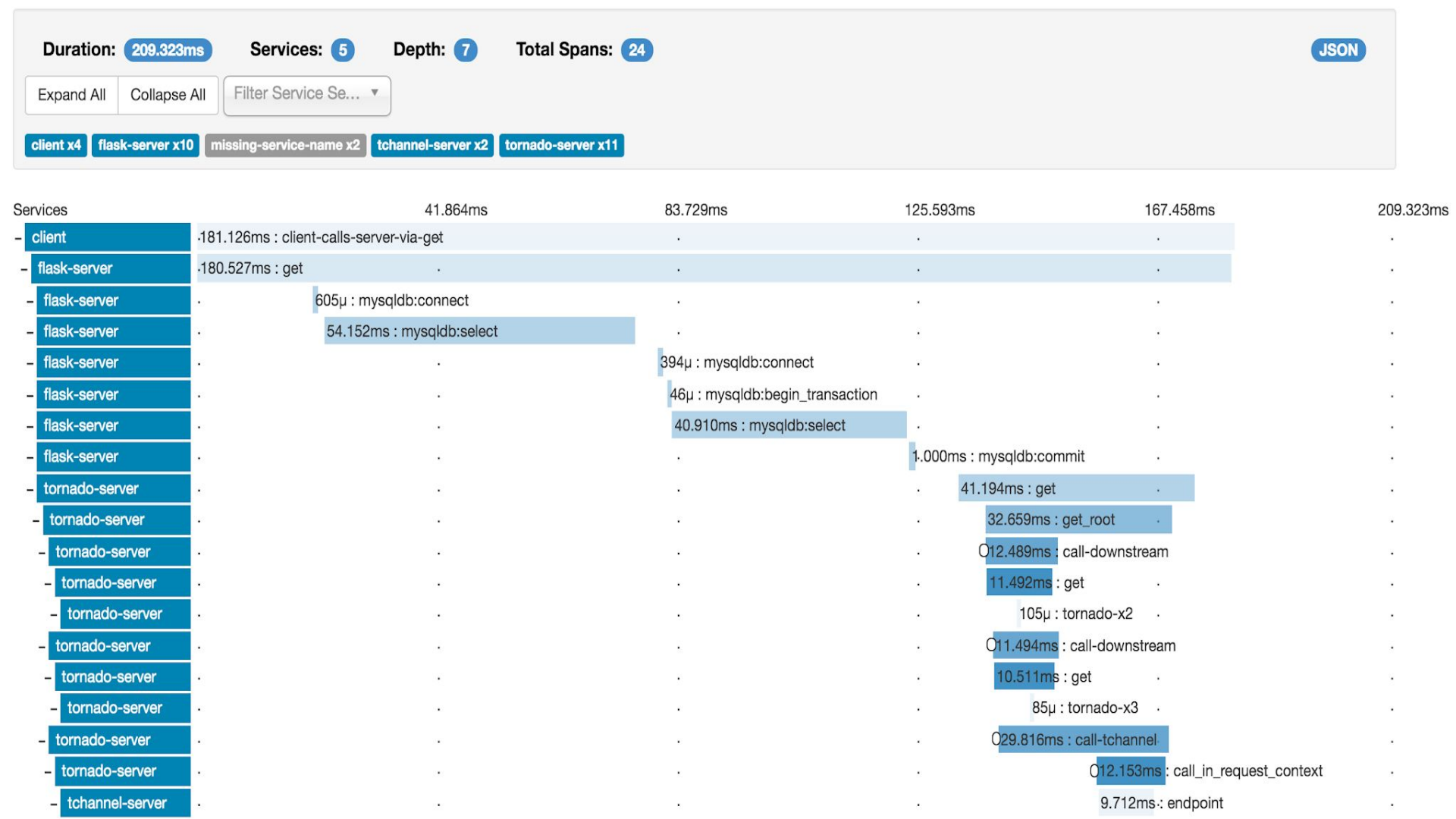

JCB


Istio Dashboard
Zoom Out
Last 1 minute
Refresh every 5s
Source All
Target All
Service All
Version All
Istio.io
Istio
Istio is an open platform that provides a uniform way to connect, manage, and secure microservices.
Need help? Join the Istio community.
Global Request Volume
33 ops
Global Success Rate (non-5xx responses)
100%
4xxs
0 ops
5xxs
0 ops
Service Mesh
Request Volume
Success Rate by Service (non-5xx responses)
4xxs by Service
5xxs by Service
Services
productpage
Requests by Source, Version, and Response Code
Success Rate by Source and Version (non-5xx responses)
Response Time by Source and Version
Response Size by Source and Version
details
Requests by Source, Version, and Response Code
Success Rate by Source and Version (non-5xx responses)
Response Time by Source and Version
Response Size by Source and Version
reviews
Requests by Source, Version, and Response Code
Success Rate by Source and Version (non-5xx responses)
Response Time by Source and Version
Response Size by Source and Version


Dashboard / [Flights] Global Flight Dashboard
Full screen
Share
Clone
Edit
Reporting
15 minutes
Last 24 hours
Search... (e.g. status:200 AND extension:PHP)
Options
Add a filter +
[Flights] Controls
Origin City
Select...
Destination City
Select...
Average Ticket Price
108
1196
Clear form
Cancel changes
Apply changes
[Flights] Markdown Instructions
Sample Flight data
This dashboard contains sample data for you to play with. You can view it, search it, and interact with the visualizations. For more information about Kibana, check our docs.
[Flights] Airline Carrier
ES-Air (23.19%)
Kibana Airlines (27.41%
ash Airways (23.8%)
JetBeats (25.6%)
[Flights] Flight Count and Average Ticket Price
Average Ticket Price
Flight Count
timestamp per 30 minutes
[Flights] Total Flights
332
Total Flights
[Flights] Average Ticket P...
$571.82
Avg. Ticket Price
[Flights] Total Flight Delays
86
Total Delays
[Flights] Flight Delays
Flight Delays
true
Count
[Flights] Delays & Cancellations
Percent Delays 23.53%
[Flights] Total Flight Cancellations
44
Total Cancellations
[Flights] Flight Cancellations
Flight Cancellations
true
[Flights] Delay Type


New Relic.
APM
BROWSER
SYNTHETICS
MOBILE
SERVERS
PLUGINS
INSIGHTS
APPS
RPM
TIME PICKER
Last 30 minutes ending now
SERVERS
All servers
MONITORING
Overview
Service maps
App map
Transactions
Databases
External services
Message queues
Ruby VMs
EVENTS
Error analytics NEW
Errors
Violations
Deployments
Thread profiler
REPORTS
SLA
Availability
Capacity
Scalability
Web transactions
Database
Background jobs
SETTINGS
Application
Web transactions response time
132 ms
APP SERVER
2.72 s
BROWSER
Time spent in MySQL
1 minute from 17:38 to 17:39
14.9 ms per request
Request Queuing
GC Execution
Middleware
Ruby
MySQL
Memcached
Redis
Web external
Apdex score
0.98 [0.4]
APP SERVER
0.84 [2.5]
BROWSER
0.83 Fair
Sample Size: 1748
Satisfied: 1192 (68%)
Tolerating: 509 (29%)
Frustrated: 47 (3%)
End User
App Server
Throughput
81.6k rpm
AVERAGE
Requests
1 minute from 17:38 to 17:39
76.9k rpm
Compare with yesterday and last week
Transactions
App server time
ChartData::MetricCharts
301 ms
ChartData::MetricCharts
164 ms
ChartData::MetricCharts
158 ms
CurrentStatusController#stat
98.1 ms
v2: /applications/:application_id/metrics/data
67.1 ms
Error rate
0.0100 s
17:38 to 17:39
Error rate: 0.01%
Errors per minute: 7.62
Requests per minute: 76,900
Application activity
In progress
Event log
Filter
There are no violations in progress right now.
17 servers
Server name
Apdex
Resp. time
Throughput
Error Rate
CPU usage
Memory
86.2 ms
6,860 rpm
0.0148 %
623 %
43 GB


Jaeger UI
Lookup by Trace ID...
Search
Dependencies
About Jaeger
frontend: HTTP GET /dispatch
Search...
View Options
Trace Start: July 20, 2018 2:48 PM | Duration: 732.11ms | Services: 6 | Depth: 5 | Total Spans: 51
0ms
183.03ms
366.06ms
549.08ms
732.11ms
Service & Operation
frontend HTTP GET /dispatch
frontend HTTP GET: /customer
frontend HTTP GET
customer HTTP GET /customer
mysql SQL SELECT
frontend Driver::findNearest
driver Driver::findNearest
redis FindDriverIDs
redis GetDriver
redis GetDriver
redis GetDriver
redis GetDriver
redis GetDriver
redis GetDriver
redis GetDriver
redis GetDriver
redis GetDriver
redis GetDriver
redis GetDriver
redis GetDriver
frontend HTTP GET: /route
frontend HTTP GET
route HTTP GET /route
frontend HTTP GET: /route
frontend HTTP GET
route HTTP GET /route
frontend HTTP GET: /route
frontend HTTP GET
route HTTP GET /route
frontend HTTP GET: /route
frontend HTTP GET
route HTTP GET /route
305.89ms
305.88ms
304.46ms
304ms
211.15ms
210.78ms
18.23ms
32ms
8.17ms
12.34ms
6.24ms
6.96ms
28.81ms
8.77ms
8.12ms
10.49ms
12.4ms
31.63ms
14.02ms
11.72ms
52.92ms
52.89ms
52.08ms
43.45ms
43.44ms
42.48ms
64.18ms
64.15ms
63.48ms
63.67ms
63.65ms
63.36ms


Duration: 209.323ms
Services: 5
Depth: 7
Total Spans: 24
JSON
Expand All
Collapse All
Filter Service Se...
client x4
flask-server x10
missing-service-name x2
tchannel-server x2
tornado-server x11
Services
41.864ms
83.729ms
125.593ms
167.458ms
209.323ms
client
181.126ms : client-calls-server-via-get
flask-server
180.527ms : get
flask-server
605µ : mysqldb:connect
flask-server
54.152ms : mysqldb:select
flask-server
394µ : mysqldb:connect
flask-server
46µ : mysqldb:begin_transaction
flask-server
40.910ms : mysqldb:select
flask-server
1.000ms : mysqldb:commit
tornado-server
41.194ms : get
tornado-server
32.659ms : get_root
tornado-server
12.489ms : call-downstream
tornado-server
11.492ms : get
tornado-server
105µ : tornado-x2
tornado-server
11.494ms : call-downstream
tornado-server
10.511ms : get
tornado-server
85µ : tornado-x3
tornado-server
29.816ms : call-tchannel
tornado-server
12.153ms : call_in_request_context
tchannel-server
9.712ms : endpoint


New
Relic.


Grafana

JAEGER

ZIPKIN

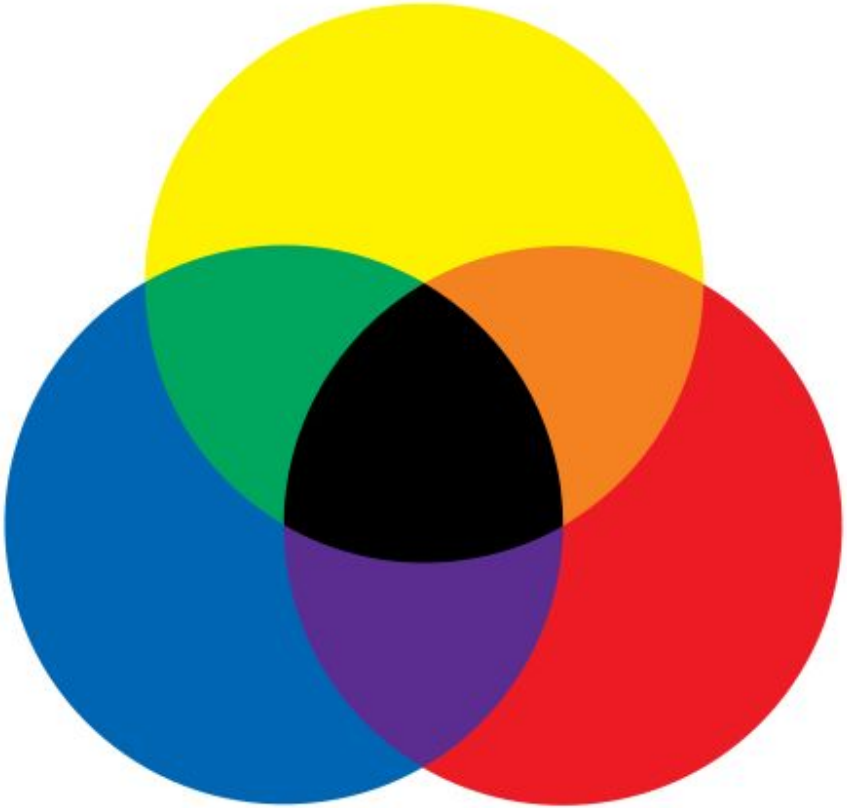
Conclusão

OBSERVABILITY ← Logging + Monitoring + Tracing + Visualization

OBSERVABILITY = CUSTO

CUSTO

- Adicionar sensores
- Criar e gerenciar alertas e os planos de ação e contenção
- Coletar, processar e analisar dados de métricas de hardware e software



Opiniões do Sinval

1. Service Mesh é uma ótima alternativa!

- Consul? Istio? Linkerd? Conduit? -> QUAL DELES USAR? NÃO SEI VEIO :(

2. OpenTracing -> FUNDAMENTAL

3. Istio + Kiali -> PARECE LINDO

* Logs com x-request-id já ajudam bastante, com uma boa centralização de logs já é possível fazer um tracing distribuído decente.

* Não acredito que começar com as novidades do momento seja obrigatório, mas recomendo fortemente a avaliação.

Opiniões do Sinval

4. Tá afim do AWS X-ray? -> CAI PRA DENTRO

5. É mais conservador? Quer ficar no ELK + NewRelic + Grafana + Prometheus….

- **#tamojunto**

OBSERVABILITY


DO IT. JUST DO IT.
imgflip.com

OBRIGADO