TAPSI

Observability
In Modern Software Architectures

Mohammad Ali Tofighi
ali.tofighi@tapsi.cab
linkedin.com/in/alitou

Feb. 2021

Table of Contents

TAPSI

World Gets Complicated

Evolution of Software Applications

TAPSI

- Simple Applications
  - Observing is trivial
- Complex Applications
  - Different Components
    - Hardware
    - Software
    - Network
    - Load

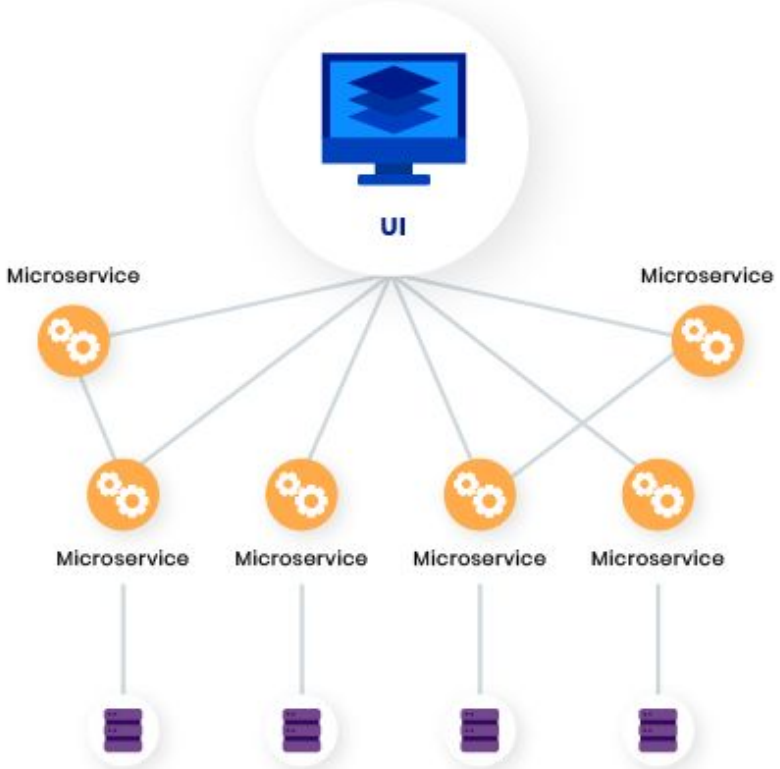
TAPSI

- Sample Application: Find First N Prime Numbers
  - Starting from simple structures
    - Processing stdin input and print result to stdout
      - The only present factor is our implementation.
  - Getting into more complex ones
    - Getting input from 3rd. party API and store result to database
      - Network overhead and database usage come to play.
    - Calling 1,000 times per second
      - Load affects performance.
- What will you do if you ***feel*** that your application is **slow**?

Revolution of Software Architectures

TAPSI

Monolithic Architecture



Microservice Architecture

Revolution of Software Architectures – Continued

TAPSI

- Distribution
  - Introduction of SOA (Service-Oriented Architecture) in 2009
- Abstraction (Cloud)
  - Reduced visibility of application
- Agile Development
  - Resulting in application instability and more living bugs
- Outsourcing
  - Relying on 3rd. party black-box services

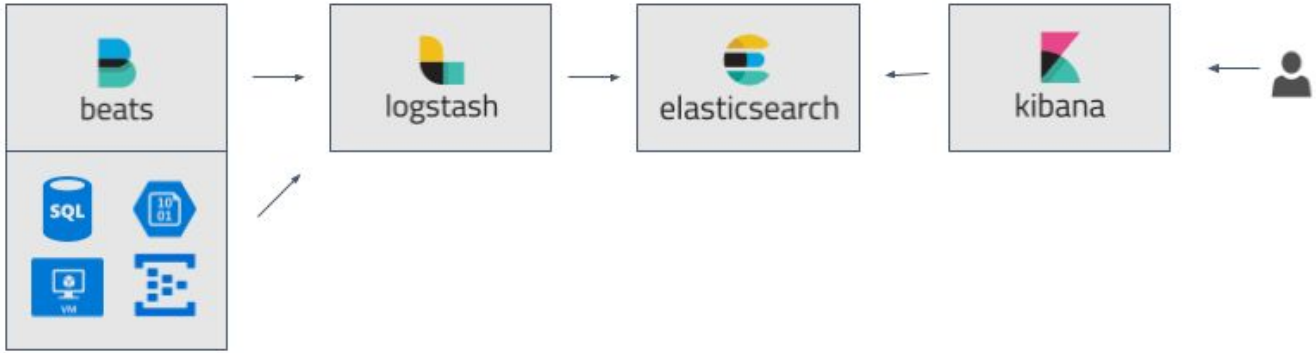
Three Generations of Application Monitoring

TAPSI

- 1st. Generation – Watching Infrastructure (1990s and early 2000s)
  - Focus on servers uptime, load, network, and storage
  - Dump important logs into text files
  - Analyzing data manually to detect the cause of an issue

Time	Level	Thread	Message
10:48:06....	INFO	Thread_13	Transaction placed successfully.
10:48:06....	INFO	Thread_3	Client '.Alice' transaction complete.
10:48:11....	INFO	Thread_11	Verifying network connection.
10:48:12....	INFO	Thread_7	Client '.Alice' transaction initiated.
10:48:12....	TRACE	Thread_13	Process start client initiated.
10:48:12....	TRACE	Thread_6	ESENT database transaction completed.
10:48:12....	INFO	Thread_1	Network connection established.
10:48:12....	TRACE	Thread_13	Executing SQL statements.
10:48:12....	WARN	Thread_10	Transaction field 698 contains invalid contract details.
10:48:13....	WARN	Thread_8	Transaction time delayed significantly. This may effect available pricing.
10:48:13....	INFO	Thread_1	Transaction details:
10:48:14....	ERROR	Thread_2	Invalid transaction details detected. Verifying with server.
10:48:14....	INFO	Thread_5	Client '.Alice' transaction complete.
10:48:19....	INFO	Thread_14	Verifying network connection.
10:48:20....	INFO	Thread_1	Client '.Alice' transaction initiated.

- 2nd. Generation – Gathering Information (2000s and 2010s)
  - Introduction of log management tools, error aggregation tools, notification tools, etc.
  - Getting a lot more information, faster
  - Costing lots of money and time to buy and maintain these tools
  - Still, hard to correlate problems and causes



TAPSI

What do you guess
about the 3rd. generation? 🤔

TAPSI

- 3rd. Generation – Contextual Intelligence
  - Appearance of AIOps in 2016
  - Unify all those tools into an integrated and more ‘aware’ platform
  - Fully automated, from deployment to discovery, problem identification and root cause.
  - NoOps is coming!

What Does Observability Mean?

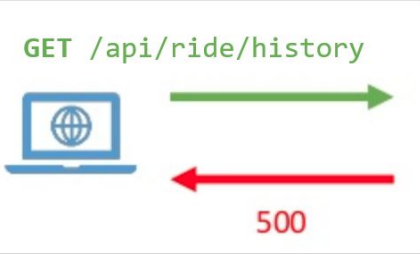
TAPSI

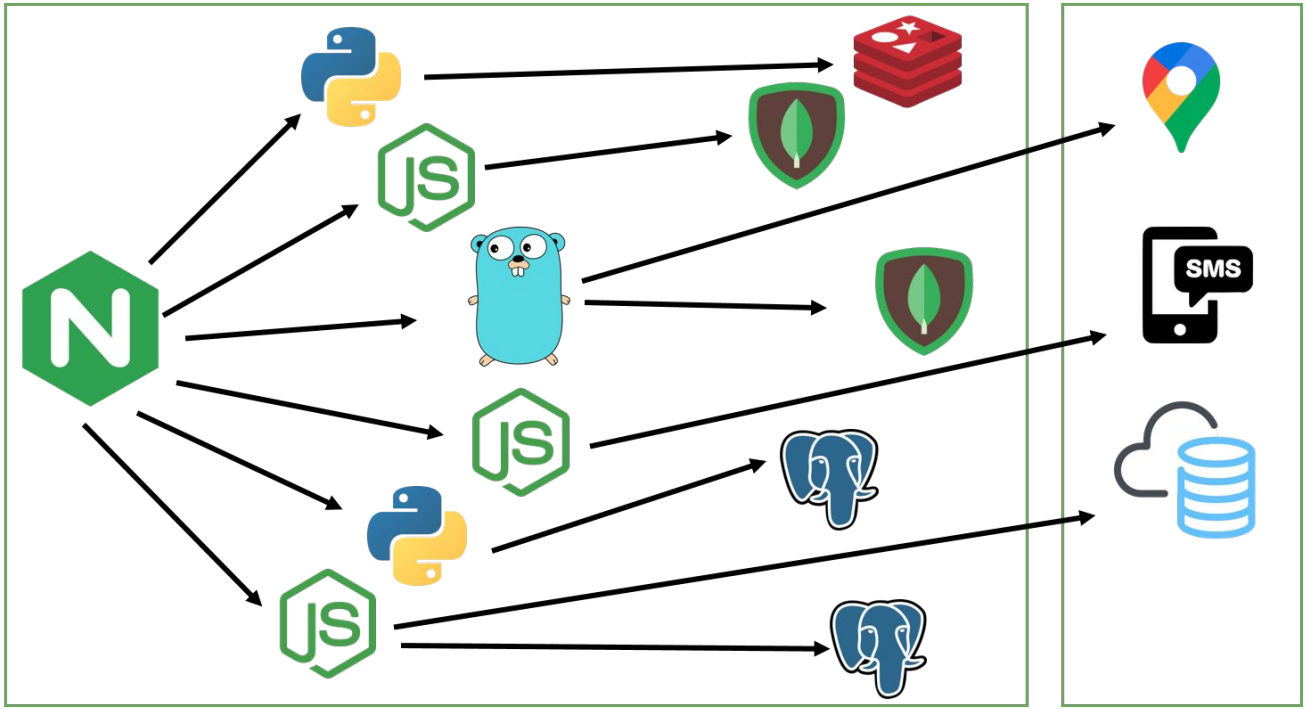
- Originated From Control Theory
  - Observability is a measure of how well **internal states** of a system can be inferred from knowledge of its **external outputs**.

TAPSI

A sample microservice-based application

What it really is:

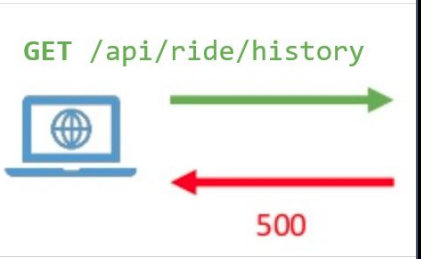




TAPSI

A sample microservice-based application

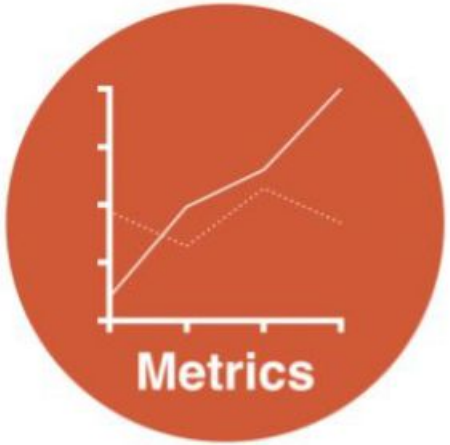
What it feels like:

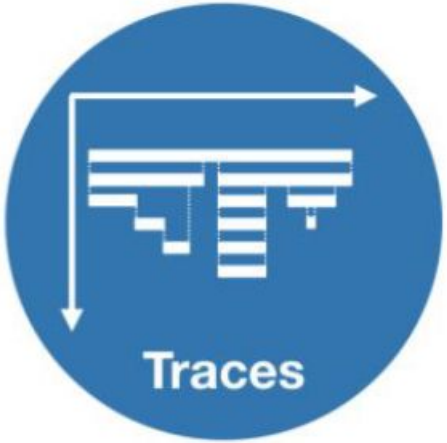


Three Pillars of Observability

TAPSI

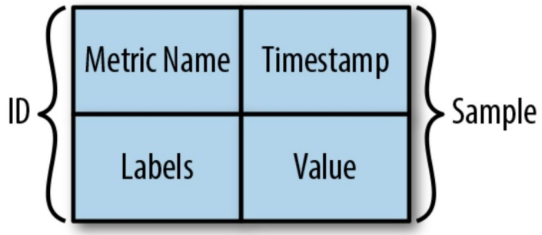
I would like to call them "Three Lenses of Observability"





Observability is a measure of how well **internal states** of a system can be inferred from knowledge of its **external outputs**.

- Metrics
  - Reflect the state of the system as a time series of numbers
  - Can be used to trigger an alert when the metric exceeds specified thresholds for a predefined length of time.

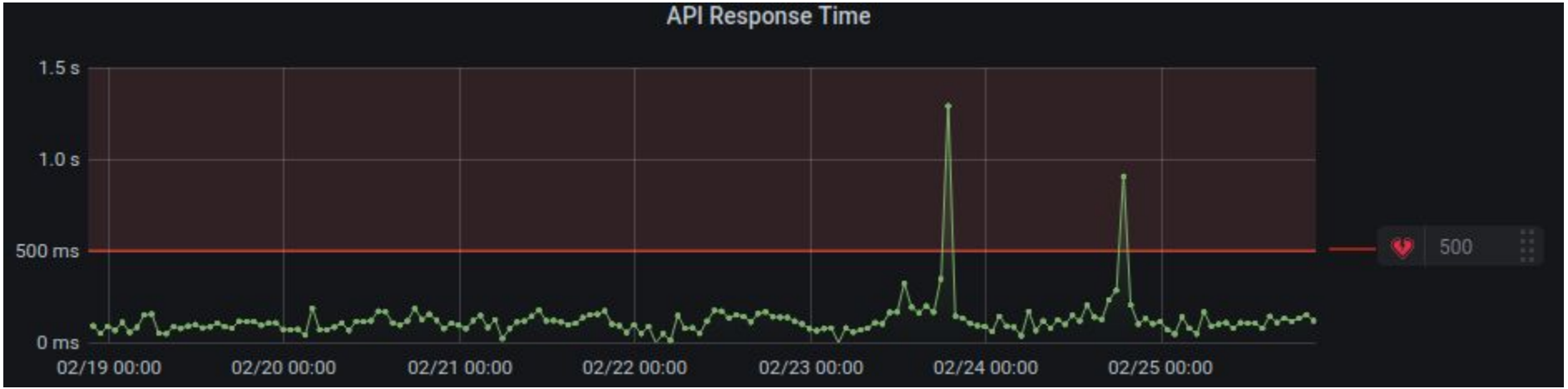


Observability is a measure of how well **internal states** of a system can be inferred from knowledge of its **external outputs**.

Three Pillars of Observability – Continued

TAPSI

- Metrics
  - *Something* happened during the two peaks below



Observability is a measure of how well **internal states** of a system can be inferred from knowledge of its **external outputs**.

Three Pillars of Observability – Continued

TAPSI

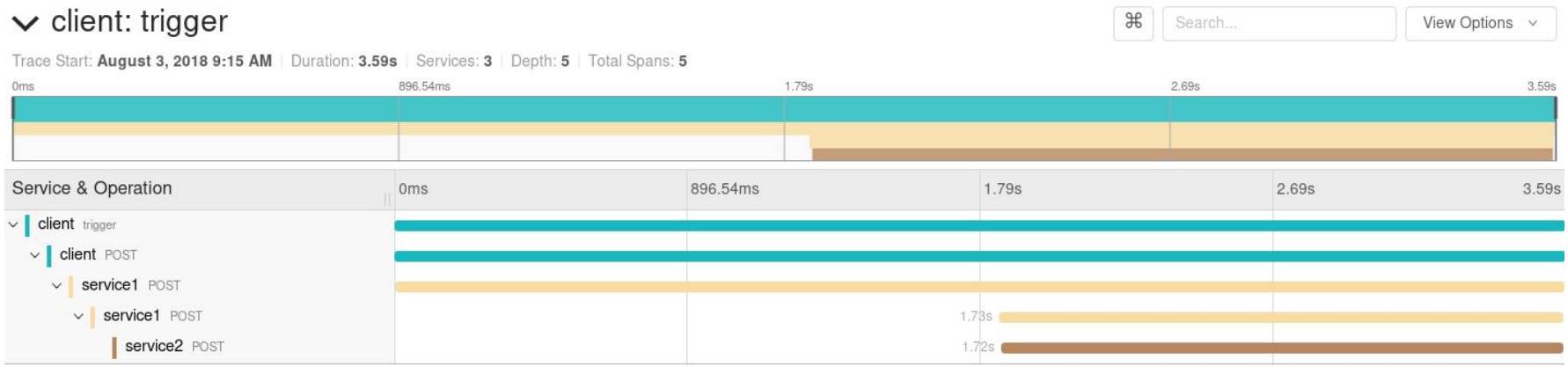
- Traces
  - Show the activity for a transaction within the entire application — all the way from the browser or mobile device down through to the database and back
  - Are the only way to understand the relationships between microservices

Observability is a measure of how well **internal states** of a system can be inferred from knowledge of its **external outputs**.

Three Pillars of Observability – Continued

TAPSI

- Traces
  - *Something* slowed down *Service 1*'s work before calling service 2



Observability is a measure of how well **internal states** of a system can be inferred from knowledge of its **external outputs**.

- Logs
  - Help us examine what really happened at the system or software level

Observability is a measure of how well **internal states** of a system can be inferred from knowledge of its **external outputs**.

Three Pillars of Observability – Continued

TAPSI

- Logs
  - A special query in the database didn't use any indexes!

```
message
[redacted] mongodb 2021-02-12T00:26:01.949+0330 I COMMAND  [conn26675] command [redacted] command: find { find: [redacted]
[redacted] }, projection: {}, returnKey: false, showRecordId: false, lsid: { id: UUID("9cec27de-af2f-4385-a6
e5-5b4225cfd3fb") }, $clusterTime: { clusterTime: Timestamp(1613076960, 59), signature: { hash: BinData(0, BBF15FDC70B9233F77C2FDC878314650913995A6), keyId: 6869354269267460097
} }, $db: [redacted] } planSummary: COLLSCAN keysExamined:0 docsExamined:41561 hasSortStage:1 cursorExhausted:1 numYields:324 nreturned:0 reslen:244 locks:{ G
lobal: { acquireCount: { r: 650 } }, Database: { acquireCount: { r: 325 } }, Collection: { acquireCount: { r: 325 } } } protocol:op_msg 115ms
```

Observability is a measure of how well **internal states** of a system can be inferred from knowledge of its **external outputs**.

TAPSI

- Metrics + Traces + Logs: The Greater Sum of the Parts
  - In order to see the full picture, we need to use all the data.

Observability is a measure of how well **internal states** of a system can be inferred from knowledge of its **external outputs**.

Fatal Flaws and Challenges

Fatal Flaws and Challenges

TAPSI

- Dependency of "Pillars"
  - Metrics, Traces, and Logs are just pieces, with low benefits on their own.
  - None of them directly addresses a particular pain point, use case, or business need.

Fatal Flaws and Challenges – Continued

TAPSI

- "Metrics are enough. Tracing and log aggregation are hard to implement!"
  - If `service="foo"` has problems, we can check metrics with the same label
  - Usually, there are more than *thousands* of metrics for `service="foo"`
  - Metrics aren't enough!

Fatal Flaws and Challenges – Continued

TAPSI

- "We use log aggregators because devs shouldn't have access to servers!"
  - No. We use them because of:
    - Centralization
    - Searchability
    - Accessibility
    - Monitoring & Alerting

TAPSI

- The Huge Amount of Logs
  - If we want to use logs to account for individual transactions:
    - Transaction rate ×
    - Number of microservice instances ×
    - Network and storage cost ×
    - Weeks of data retention
      - Too much cost!
  - Logs are less useful than monolithic applications
    - Due to distributed and concurrent nature of microservices

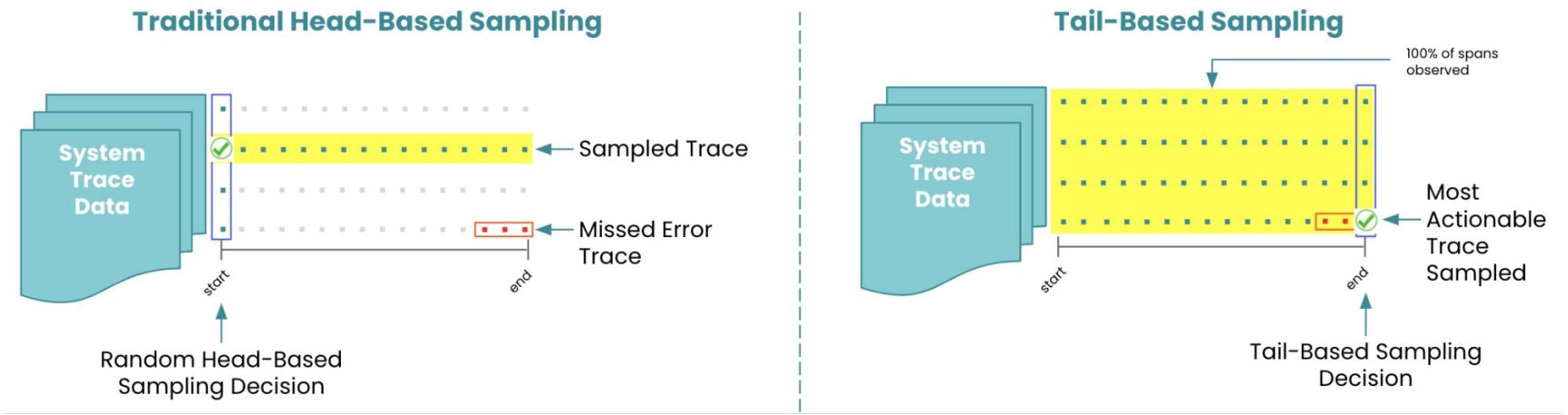
Fatal Flaws and Challenges – Continued

TAPSI

- What To Monitor?
  - Google's Golden Signals: https://sre.google/sre-book/monitoring-distributed-systems
  - RED
    - Rate
    - Errors
    - Duration
  - USE
    - Utilization
    - Saturation
    - Errors

TAPSI

- What To Trace? How Often Do The Tracing?
  - Head-based sampling
  - Tail-based sampling



Fatal Flaws and Challenges – Continued

TAPSI

- Yellow = Green + Red
  - Requires experience or specific knowledge
  - Correlating metrics, traces and logs is really an extraordinary complicated task
  - We may make mistakes during correlating them, and confuse ourselves
  - We will need machines. Remember the 3rd. generation of monitoring

Open-Source Tools

- Prometheus
  - Is an open-source systems monitoring and alerting toolkit originally built at SoundCloud in 2012;
  - Joined the Cloud Native Computing Foundation in 2016;
  - Is the second CNCF graduated project, after Kubernetes;
  - Has a multi-dimensional data model with time series data identified by metric name and key/value pair of labels.

TAPSI

- Prometheus

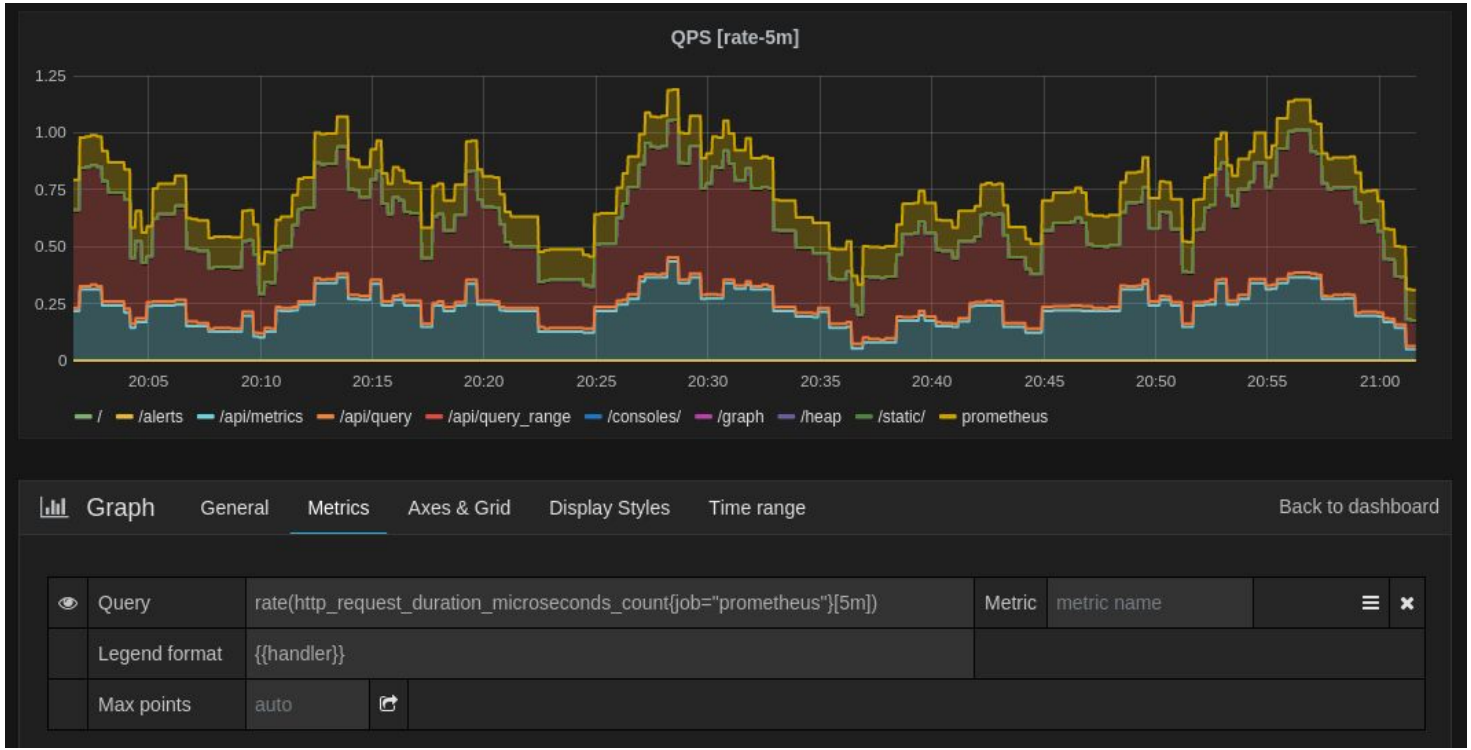
localhost:3000/metrics

```
# TYPE http_server_requests_total counter
# HELP http_server_requests_total The total number of HTTP requests handled by the Rack application.
http_server_requests_total{code="200",method="get",path="/"} 1.0
# TYPE http_server_request_duration_seconds histogram
# HELP http_server_request_duration_seconds The HTTP response duration of the Rack application.
http_server_request_duration_seconds_bucket{method="get",path="/",le="0.005"} 0.0
http_server_request_duration_seconds_bucket{method="get",path="/",le="0.01"} 0.0
http_server_request_duration_seconds_bucket{method="get",path="/",le="0.025"} 0.0
http_server_request_duration_seconds_bucket{method="get",path="/",le="0.05"} 0.0
http_server_request_duration_seconds_bucket{method="get",path="/",le="0.1"} 0.0
http_server_request_duration_seconds_bucket{method="get",path="/",le="0.25"} 0.0
http_server_request_duration_seconds_bucket{method="get",path="/",le="0.5"} 1.0
http_server_request_duration_seconds_bucket{method="get",path="/",le="1"} 1.0
http_server_request_duration_seconds_bucket{method="get",path="/",le="2.5"} 1.0
http_server_request_duration_seconds_bucket{method="get",path="/",le="5"} 1.0
http_server_request_duration_seconds_bucket{method="get",path="/",le="10"} 1.0
http_server_request_duration_seconds_bucket{method="get",path="/",le="+Inf"} 1.0
http_server_request_duration_seconds_sum{method="get",path="/"} 0.251396
http_server_request_duration_seconds_count{method="get",path="/"} 1.0
# TYPE http_server_exceptions_total counter
# HELP http_server_exceptions_total The total number of exceptions raised by the Rack application.
```

- Grafana
  - Is a multi-platform open source analytics and interactive visualization tool, started in 2014;
  - Allowing to query, visualize, alert on and understand metrics, no matter where they are stored.

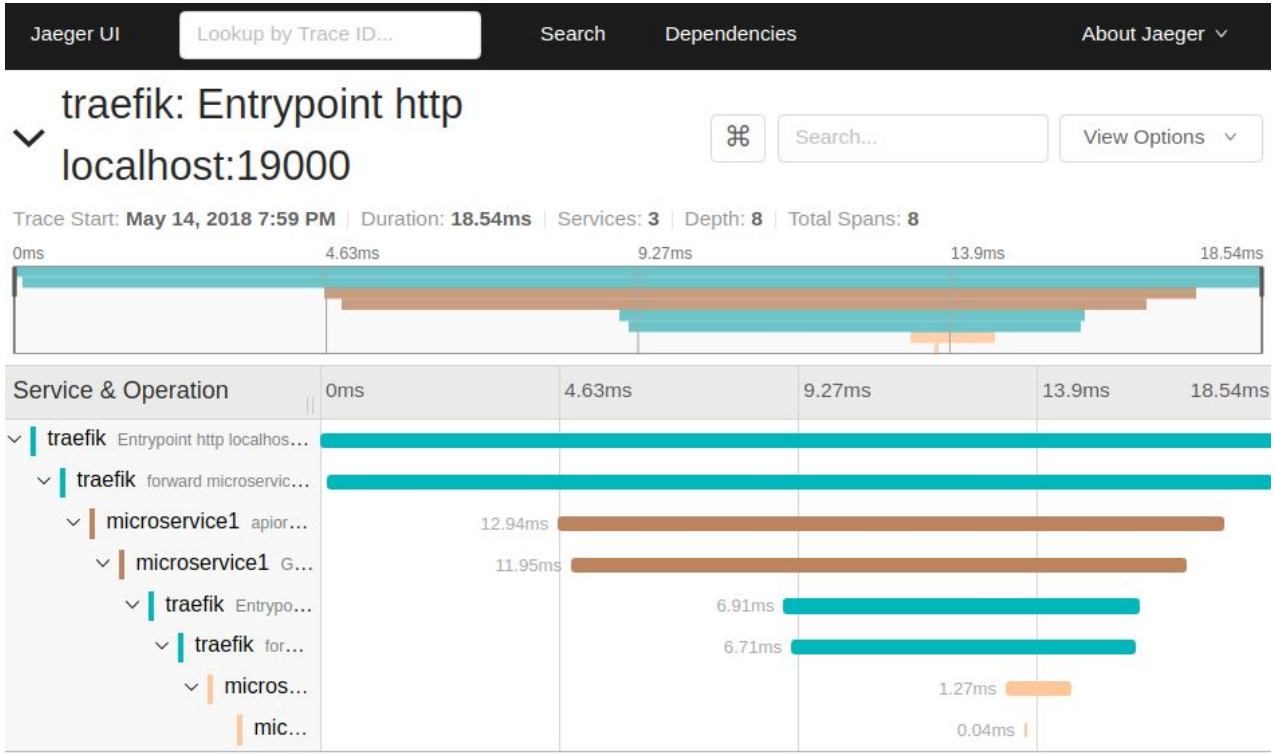
TAPSI

- Grafana



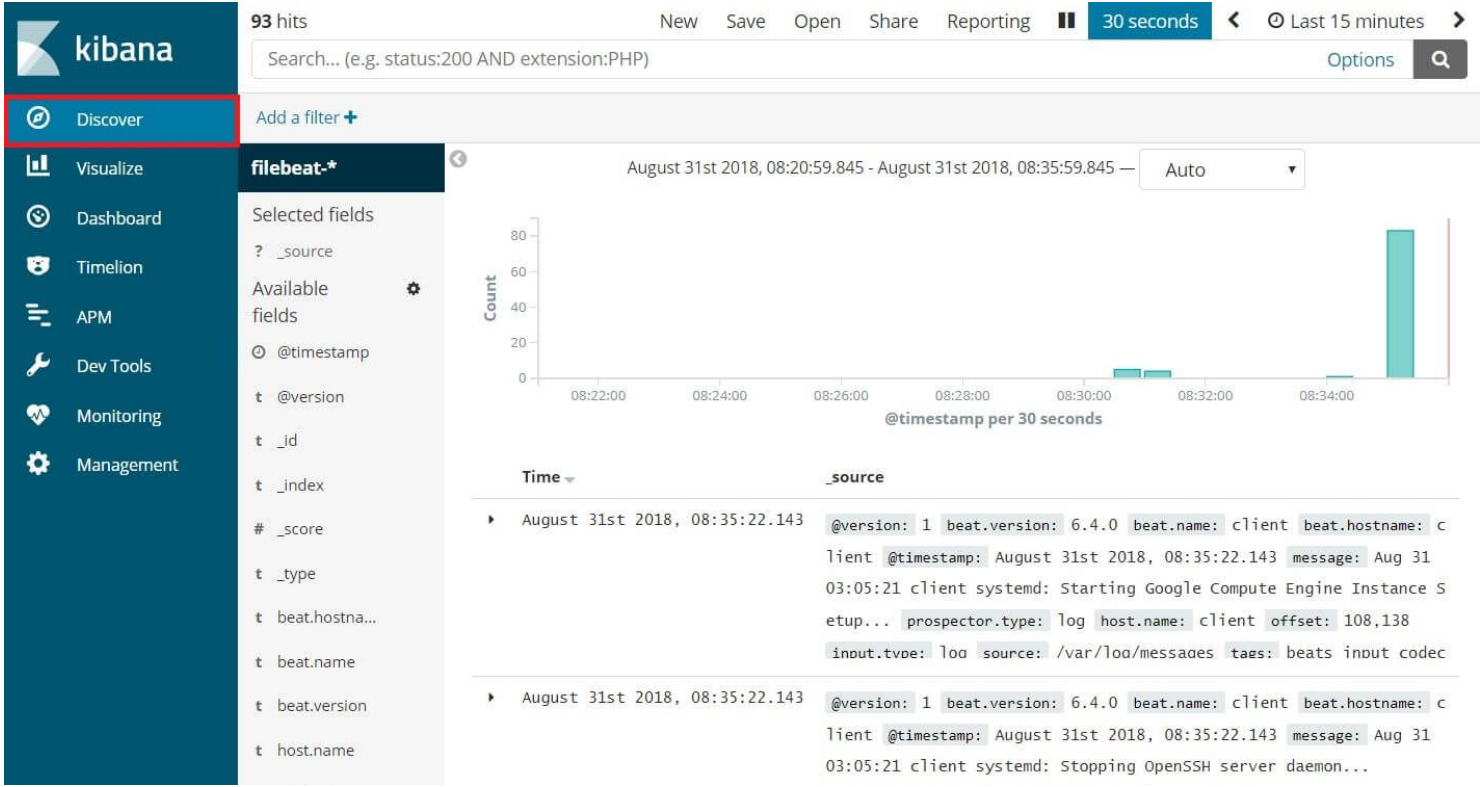
- Jaeger
  - Is an open source, end-to-end distributed tracing tool initially developed at Uber in 2017;
  - Is the 7th. graduated projects from CNCF.

- Jaeger


Jaeger UI
Lookup by Trace ID...
Search
Dependencies
About Jaeger
traefik: Entrypoint http localhost:19000
Search...
View Options
Trace Start: May 14, 2018 7:59 PM | Duration: 18.54ms | Services: 3 | Depth: 8 | Total Spans: 8
0ms
4.63ms
9.27ms
13.9ms
18.54ms
Service & Operation
traefik Entrypoint http localhos...
traefik forward microservic...
microservice1 apior...
12.94ms
microservice1 G...
11.95ms
traefik Entrypo...
6.91ms
traefik for...
6.71ms
micros...
1.27ms
mic...
0.04ms

- ELK Stack
  - "ELK" is the acronym for three open source projects: Elasticsearch, Logstash, and Kibana;
  - Elasticsearch is a search and analytics engine;
  - Logstash is a data processing pipeline that ingests data from multiple sources and then sends it to Elasticsearch;
  - Kibana lets users visualize data with charts and graphs in Elasticsearch.

- ELK Stack


kibana
93 hits
New
Save
Open
Share
Reporting
30 seconds
Last 15 minutes
Search... (e.g. status:200 AND extension:PHP)
Options
Discover
Visualize
Dashboard
Timelion
APM
Dev Tools
Monitoring
Management
Add a filter +
filebeat-*
Selected fields
? _source
Available fields
@timestamp
t @version
t _id
t _index
_score
t _type
t beat.hostna...
t beat.name
t beat.version
t host.name
August 31st 2018, 08:20:59.845 - August 31st 2018, 08:35:59.845 —
Auto
Count
80
60
40
20
0
08:22:00
08:24:00
08:26:00
08:28:00
08:30:00
08:32:00
08:34:00
@timestamp per 30 seconds
Time
_source
August 31st 2018, 08:35:22.143
@version: 1 beat.version: 6.4.0 beat.name: client beat.hostname: client @timestamp: August 31st 2018, 08:35:22.143 message: Aug 31 03:05:21 client systemd: Starting Google Compute Engine Instance Setup... prospector.type: log host.name: client offset: 108,138 input.type: log source: /var/log/messages tags: beats_input_codec
August 31st 2018, 08:35:22.143
@version: 1 beat.version: 6.4.0 beat.name: client beat.hostname: client @timestamp: August 31st 2018, 08:35:22.143 message: Aug 31 03:05:21 client systemd: Stopping OpenSSH server daemon...

TAPSI

- There are more!

fluentbit

ZABBIX

Nagios®

fluentd

Thanks!

ali.tofighi@tapsi.cab