

# Osservabilità di applicazioni in cloud con Elastic

---

Enrico Zimuel, Principal Software Engineer

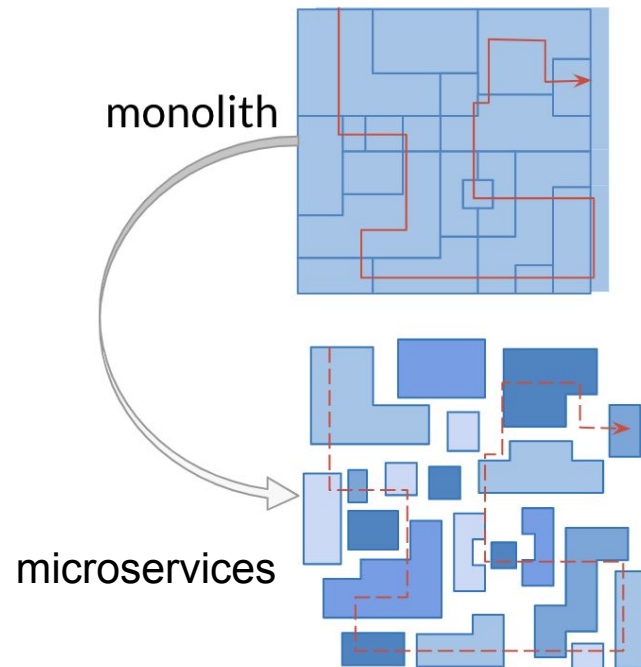


# Osservabilità (observability)

Nella teoria del controllo, l'**osservabilità** è una proprietà di un **sistema dinamico** che indica la possibilità di risalire allo stato del sistema a partire dalla conoscenza delle sue uscite. (*Wikipedia*)

# Evoluzione dei sistemi software

- Big Data
  - Come gestire grandi quantità di dati?
- Digital Transformation
  - Come le applicazioni influenzano il mio business?
- Cloud
  - Come monitorare un'applicazione in cloud?
- Serverless
  - Come monitorare un servizio FaaS?
- Containers, Microservizi
  - Come monitorare un container/microservizio?

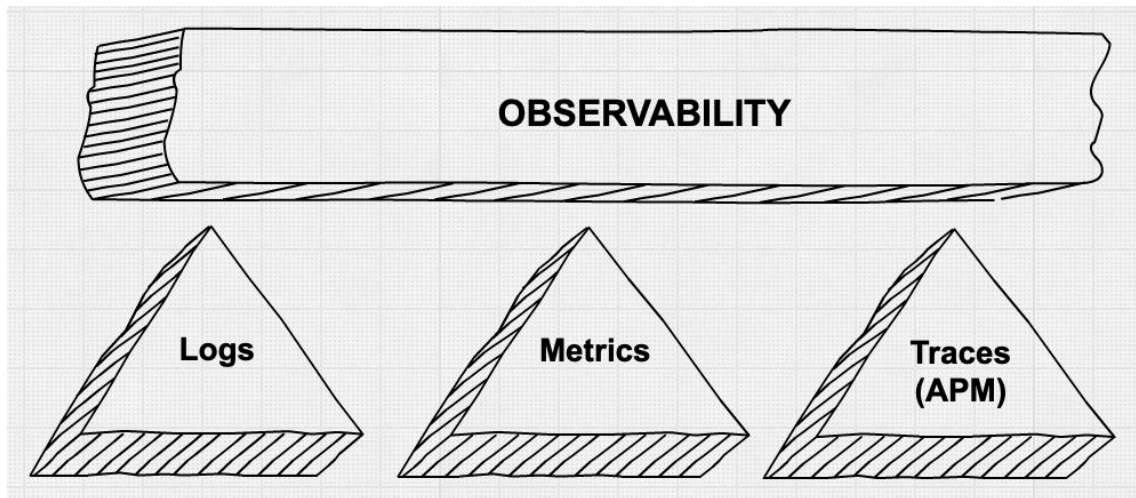


# I sistemi software sono un sistema complesso

Possiamo solo cercare di mitigare la complessità

- I sistemi software sono intrinsecamente **dinamici** e **instabili**
- È impossibile prevedere la miriade di stati di parziale fallimento in cui potrebbero finire varie parti del sistema
- I possibili errori devono essere gestiti in ogni fase, dalla progettazione del sistema all'implementazione, al collaudo, alla distribuzione e, infine, al funzionamento

# I tre pilastri dell'osservabilità



APM = Application Performance Monitoring

# Logs

I log vengono generati per ogni evento (es. richieste HTTP)

```
64.242.88.10 - - [07/Mar/2017:16:10:02 -0800] "GET /mailman/listinfo/hsdivision HTTP/1.1" 200 6291
64.242.88.10 - - [07/Mar/2017:16:11:58 -0800] "POST /twiki/bin/view/TWiki/WikiSyntax HTTP/1.1" 404 7352
64.242.88.10 - - [07/Mar/2017:16:20:55 -0800] "GET /twiki/bin/view/Main/DCCAndPostFix HTTP/1.1" 200 5253
```

***Ogni evento è memorizzato in una o più righe in un file di testo***

Vantaggi:

- Generalmente disponibile nell'host dove viene eseguito il servizio
- Facile lettura e personalizzabile (per applicazioni in-house)

Svantaggi:

- Logs sono a livello di componente, non a livello applicativo
- Devono essere inseriti esplicitamente nel codice del programma
- La formattazione è importante



# Metriche (metrics)

Le metriche sono misurazioni periodiche di diversi valori

07/Mar/2017	16:10:00	all	2.58	0.00	0.70	1.12	0.05	95.55	server1	containerX	regionA
07/Mar/2017	16:20:00	all	2.56	0.00	0.69	1.05	0.04	95.66	server2	containerY	regionB
07/Mar/2017	16:30:00	all	2.64	0.00	0.65	1.15	0.05	95.50	server2	containerZ	regionC

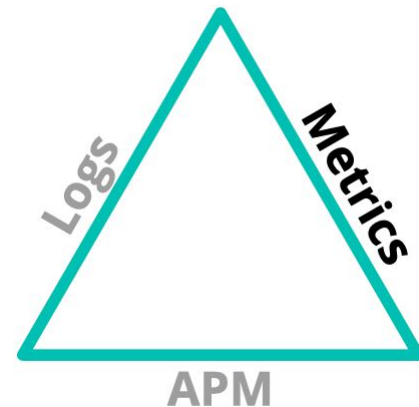
***Ogni x secondi, misura il carico della CPU e stampa il risultato***

Vantaggi:

- Mettono in evidenza i trend, sintetizzano lo storico di un sistema
- Possono essere utilizzati con sistemi di alert come incidenti e anomalie

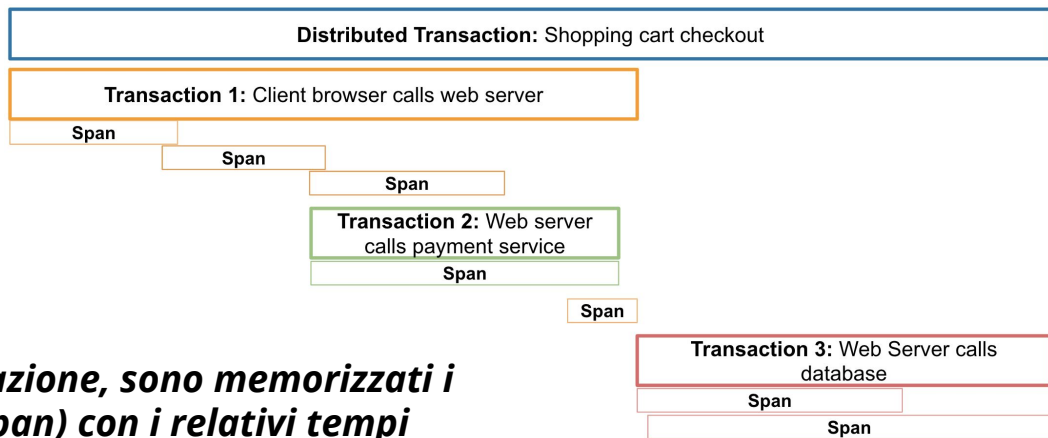
Svantaggi:

- La maggior parte delle metriche sono a livello di componente, non a livello applicativo
- I valori anomali in un certo intervallo possono essere livellati dalla media
- I container possono distorcere i risultati se provenienti da host o network



# Traces

Traces (tracce) sono attività specifiche all'interno dell'applicazione



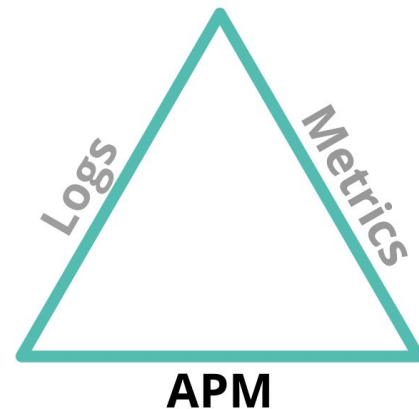
***Per ogni transazione, sono memorizzati i componenti (span) con i relativi tempi***

Vantaggi:

- Informazioni molto dettagliate per analizzare problemi in profondità

Svantaggi:

- I traces possono contenere “troppe” informazioni
- Necessità di aggiungere del codice nelle applicazioni (instrumenting)

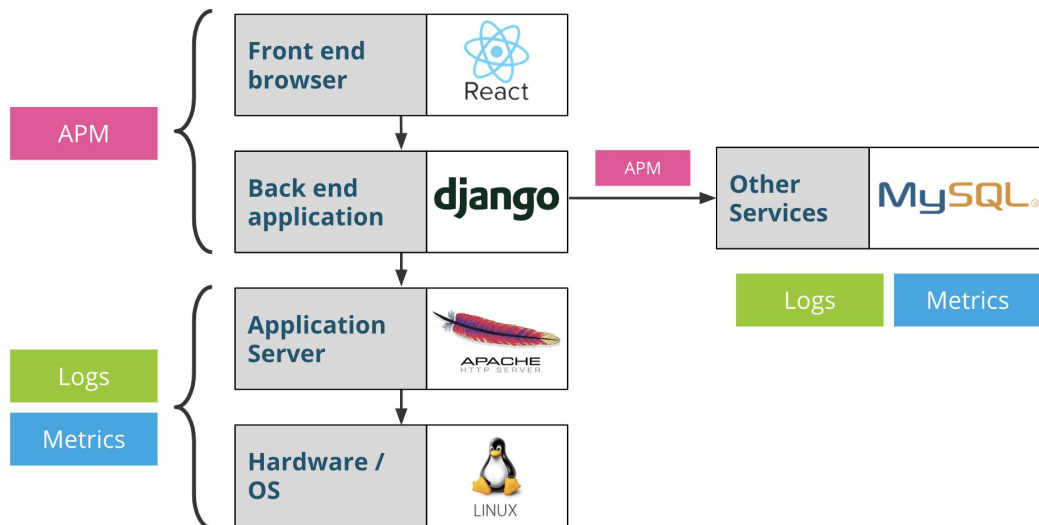




# Logs, Metriche e Traces (APM)

## Riassumendo: un tipico caso

- Richieste HTTP e errori in log di un server web
- CPU e RAM come metriche misurate dalle API del sistema operativo
- Tempi dell'applicazione e dei componenti catturate dai traces



# Logs + Metriche + Traces ≠ Osservabilità

- L'utilizzo di un sistema di log, delle metriche e delle funzionalità di traces non sono sufficienti per rendere un sistema osservabile
- E' necessario integrare e unificare i dati in un sistema che agevoli la ricerca delle informazioni evidenziando le criticità
- C'è bisogno di una cultura dell'osservabilità, gli strumenti da soli non sono sufficienti

# Osservabilità ≠ Monitoraggio (Monitoring)

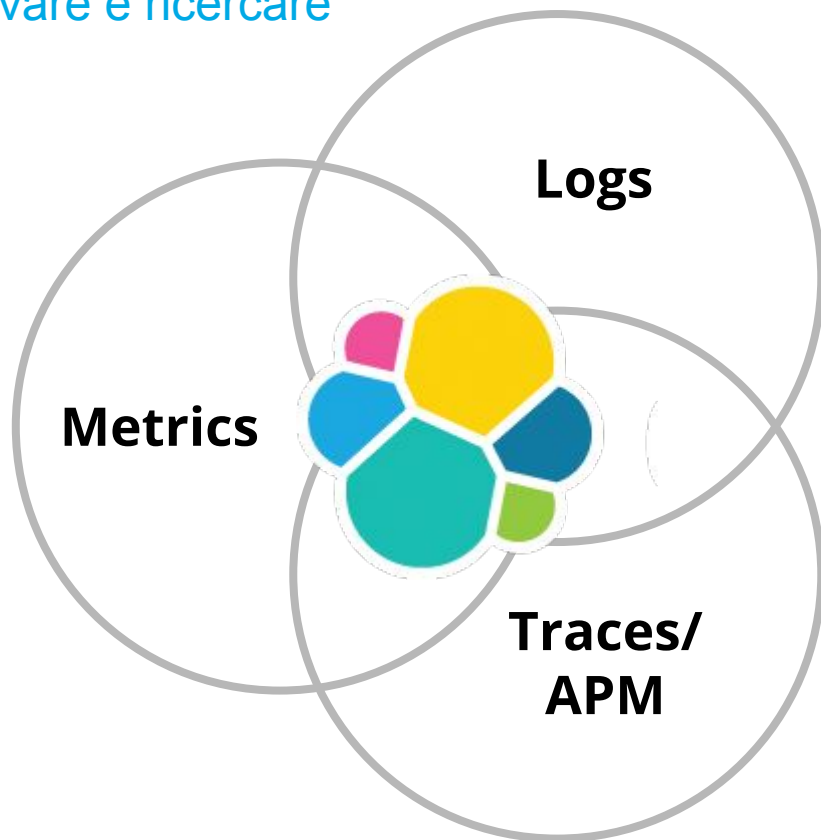
“Il monitoraggio indica se il sistema funziona.  
L'osservabilità consente di rispondere sul  
perché non funziona”

Baron Schwartz (@xaprb)

**L'osservabilità è un caso specifico  
di "search"**

# Lo stack open source di Elastic

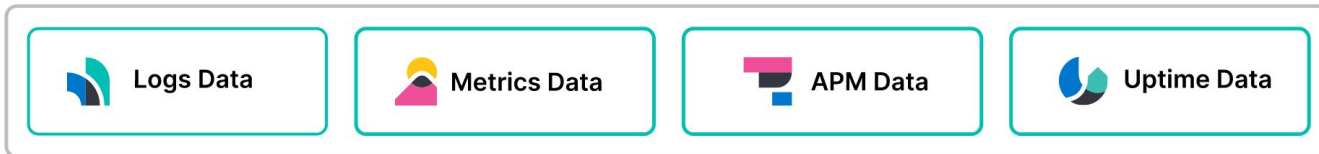
Osservare è ricercare



- ✓ Discovery
- ✓ Filtering & Correlation
- ✓ Anomaly Detection
- ✓ Alerting
- ✓ Dashboards
- ✓ Reporting
- ✓ ***Integrated***

# Elastic Observability

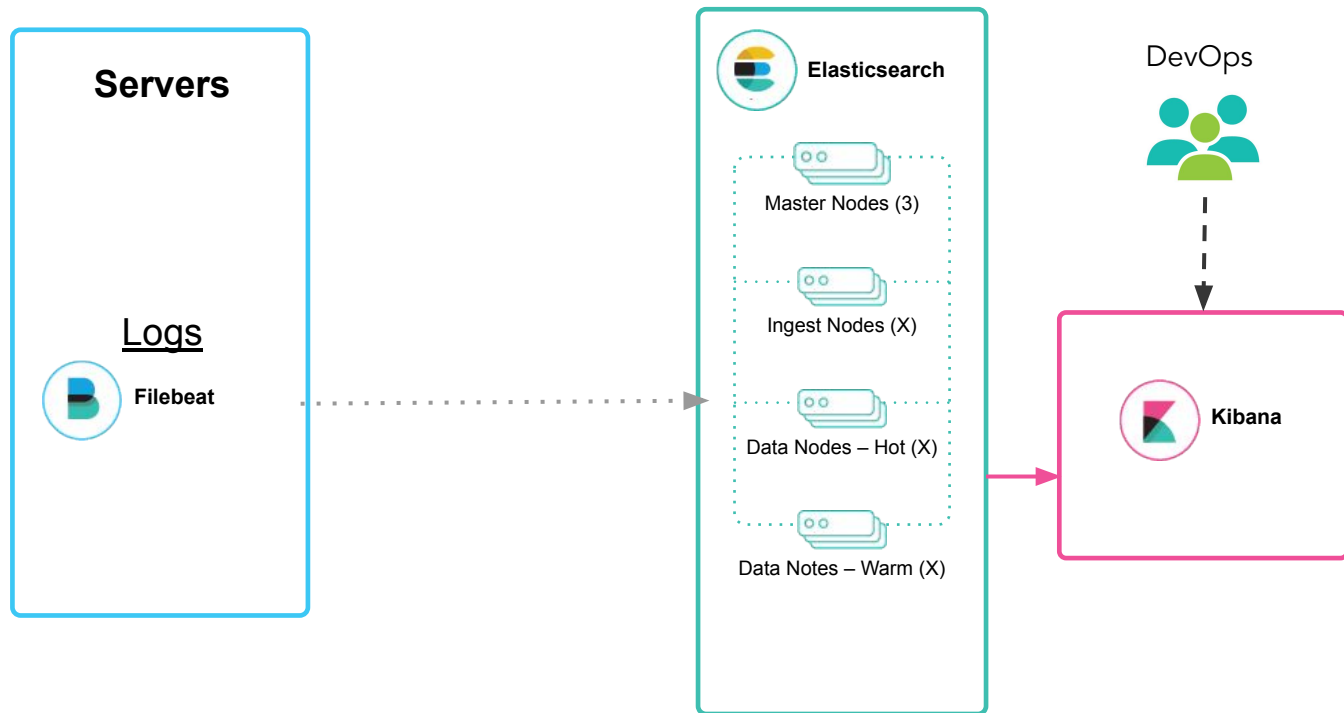
## Dev & Ops Teams



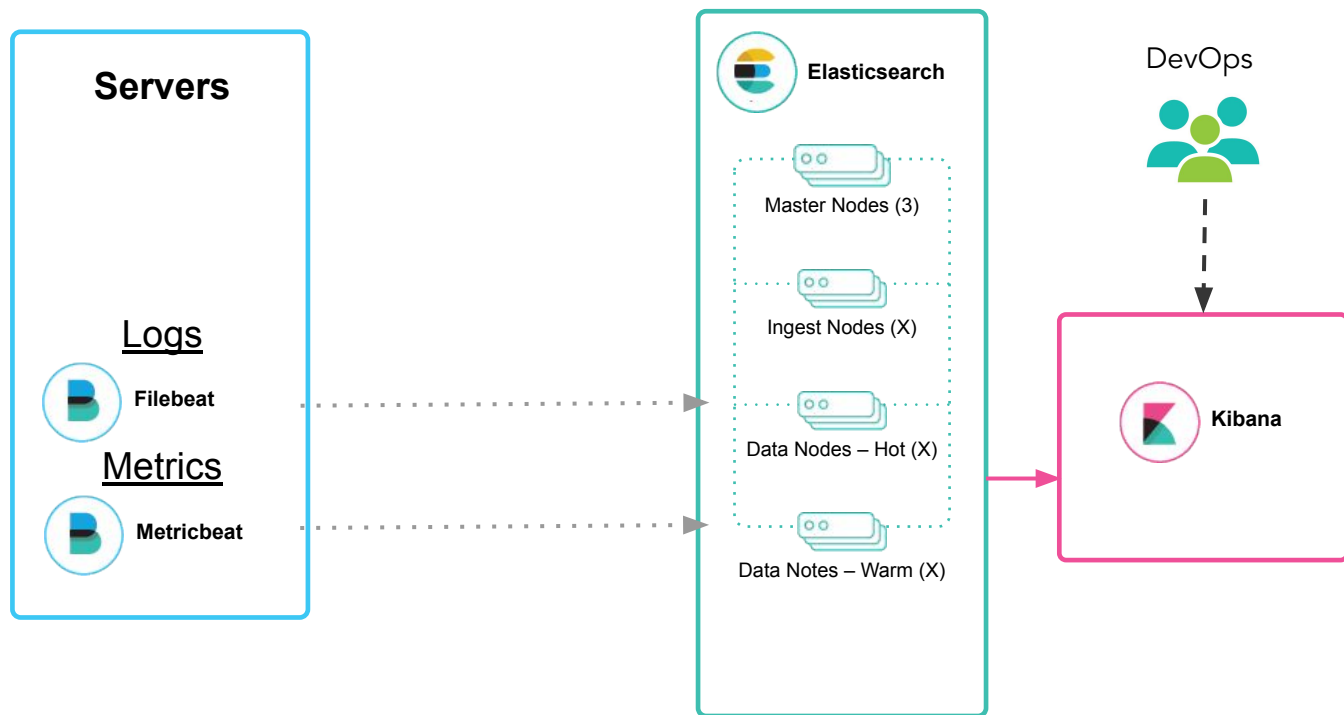
Web Logs	Host/Container Metrics	Real User Monitoring	Uptime
App Logs	Database Metrics	Transaction Monitoring	Response
Database Logs	Network Metrics	Distributed Tracing	Correctness
Container Logs	Storage Metrics	Dependency Mapping	Certificate Validation

 Elasticsearch +  Kibana

# Elastic e i 3 pilastri dell'osservabilità

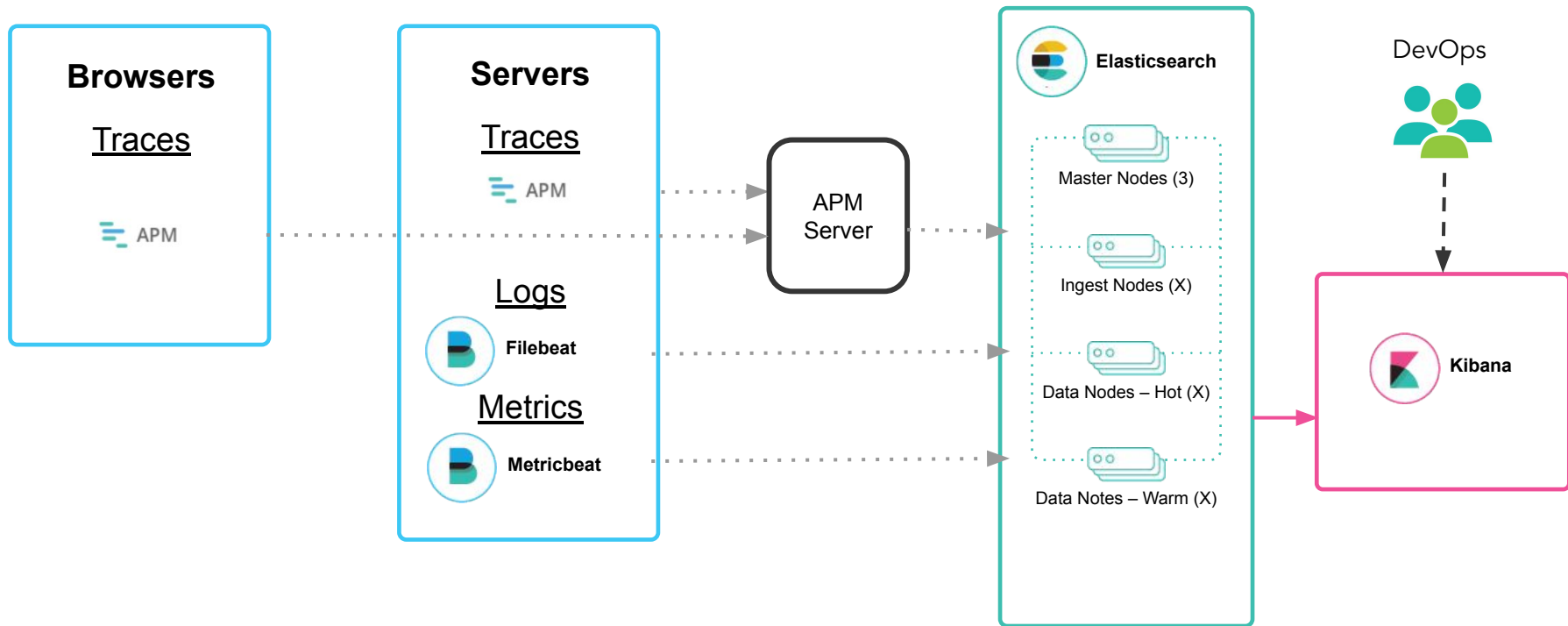


# Elastic e i 3 pilastri dell'osservabilità

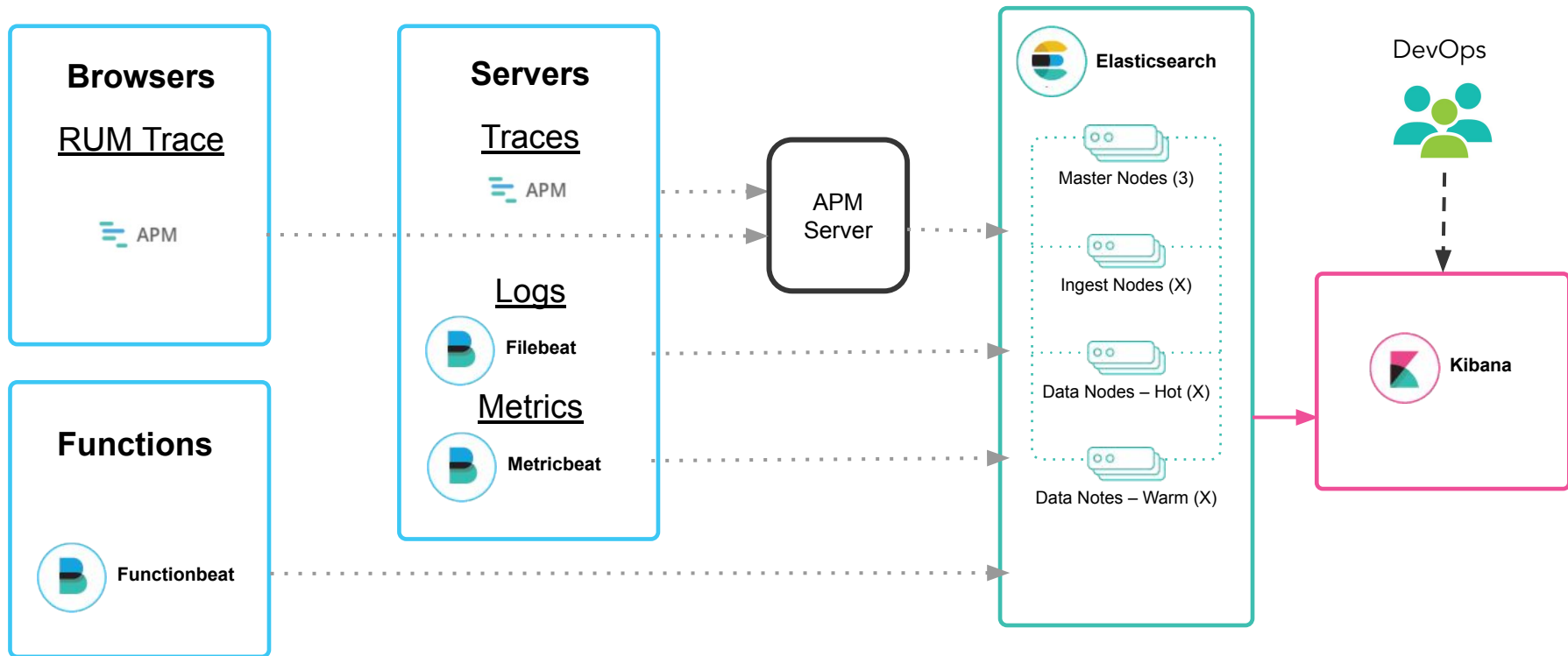




# Elastic e i 3 pilastri dell'osservabilità



# Elastic e i 3 pilastri dell'osservabilità



# Logs

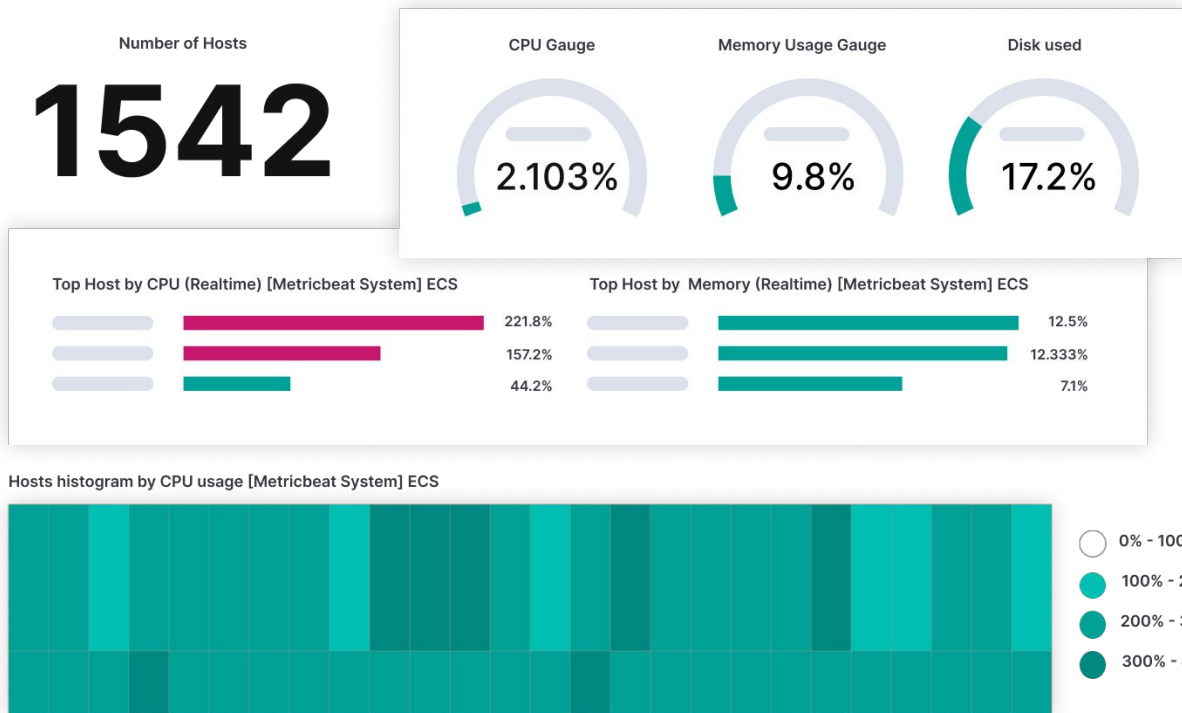
The screenshot displays the Elastic Logs interface. At the top, there is a search bar with the placeholder text "Search for log entries... (e.g. host.name:host-1)". To the right of the search bar are buttons for "Configuration", "Customize", "Highlights", and a date/time filter set to "07/17/2019 10:24:55 AM". A "Stream live" button is also present.

The main area contains a table of log entries. The table has two columns: "Timestamp" and "Message". The entries are sorted by time, showing a sequence of events from 10:24:55.497 to 10:24:55.514. The messages include error details, application exceptions, and status reports.

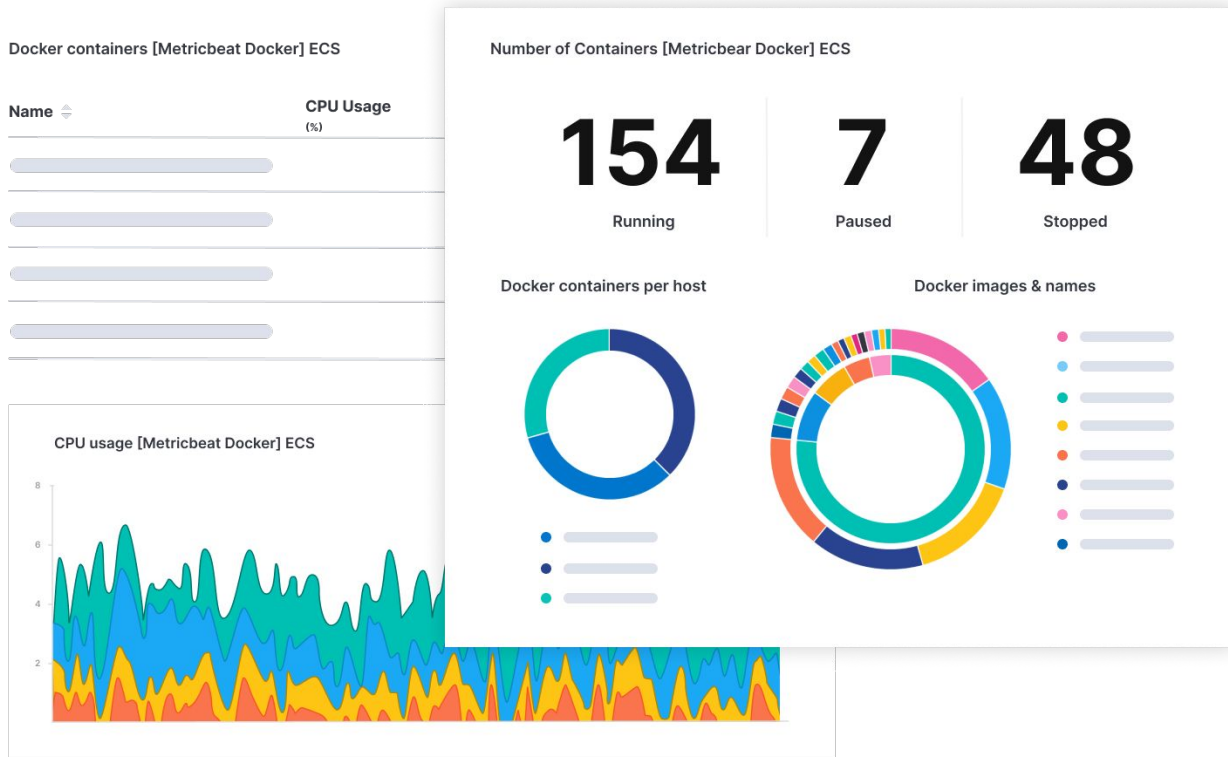
Timestamp	Message
Jul 17, 2019 @ 10:24:55.497	<pre>{\"level\":50,\"time\":1563373495496,\"pid\":786,\"hostname\":\"f559ddf5cb6b\",\"msg\":\"Application encountered an uncaught exception. Flushing Elastic APM queue and exiting...\", \"v\":1}</pre>
Jul 17, 2019 @ 10:24:55.497	<pre>{\"level\":50,\"time\":1563373495496,\"pid\":786,\"hostname\":\"f559ddf5cb6b\",\"code\":\"ECONNRESET\",\"msg\":\"socket hang up\", \"stack\":\"Error: socket hang up\\n    at createHangUpError (_http_client.js:342:15)\\n    at Socket.socketCloseListener (_http_client.js:377:23)\\n    at emitOne (events.js:121:20)\\n    at Socket.emit (events.js:211:7)\\n    at TCP._handle.close [as _onclose] (net.js:561:12)\", \"type\":\"Error\", \"v\":1}</pre>
Jul 17, 2019 @ 10:24:55.497	<pre>{\"level\":50,\"time\":1563373495496,\"pid\":786,\"hostname\":\"f559ddf5cb6b\",\"msg\":\"Application encountered an uncaught exception. Flushing Elastic APM queue and exiting...\", \"v\":1}</pre>
Jul 17, 2019 @ 10:24:55.497	<pre>{\"level\":50,\"time\":1563373495496,\"pid\":786,\"hostname\":\"f559ddf5cb6b\",\"msg\":\"Elastic APM queue flushed!\", \"v\":1}</pre>
2019-07-17 14:24:55.507 UTC [10307] LOG:	could not receive data from client: Connection reset by peer
Jul 17, 2019 @ 10:24:55.507	2019-07-17T14:24:55: PM2 log: App [server:1] exited with code [1] via signal [SIGINT]
Jul 17, 2019 @ 10:24:55.509	2019-07-17T14:24:55: PM2 log: App [server:1] starting in -fork mode-
Jul 17, 2019 @ 10:24:55.512	14:24:55 dotnet.1   SUCCESSES: 0   FAILURES: 12083   WORKERS: 1
Jul 17, 2019 @ 10:24:55.513	14:24:55 dotnet.1   SUCCESSES: 0   FAILURES: 12083   WORKERS: 1
Jul 17, 2019 @ 10:24:55.513	14:24:55 dotnet.1   SUCCESSES: 0   FAILURES: 12083   WORKERS: 1
Jul 17, 2019 @ 10:24:55.513	14:24:55 dotnet.1   ClientConnectorError(111, 'Connection refused')
Jul 17, 2019 @ 10:24:55.513	14:24:55 dotnet.1   File \"/usr/local/lib/python3.7/site-packages/molotov/worker.py\", line 286, in step
Jul 17, 2019 @ 10:24:55.513	14:24:55 dotnet.1   **scenario['kw']
Jul 17, 2019 @ 10:24:55.513	14:24:55 dotnet.1   File \"molotov_scenarios.py\", line 34, in scenario_products_top
Jul 17, 2019 @ 10:24:55.513	14:24:55 dotnet.1   async with session.get(join(SERVER_URL, 'api', 'products', 'top')) as resp:
Jul 17, 2019 @ 10:24:55.513	14:24:55 dotnet.1   File \"/usr/local/lib/python3.7/site-packages/aiohttp/client.py\", line 843, in __aenter__
Jul 17, 2019 @ 10:24:55.514	14:24:55 dotnet.1   self._resp = await self._coro

# Metriche d'infrastruttura (Server)

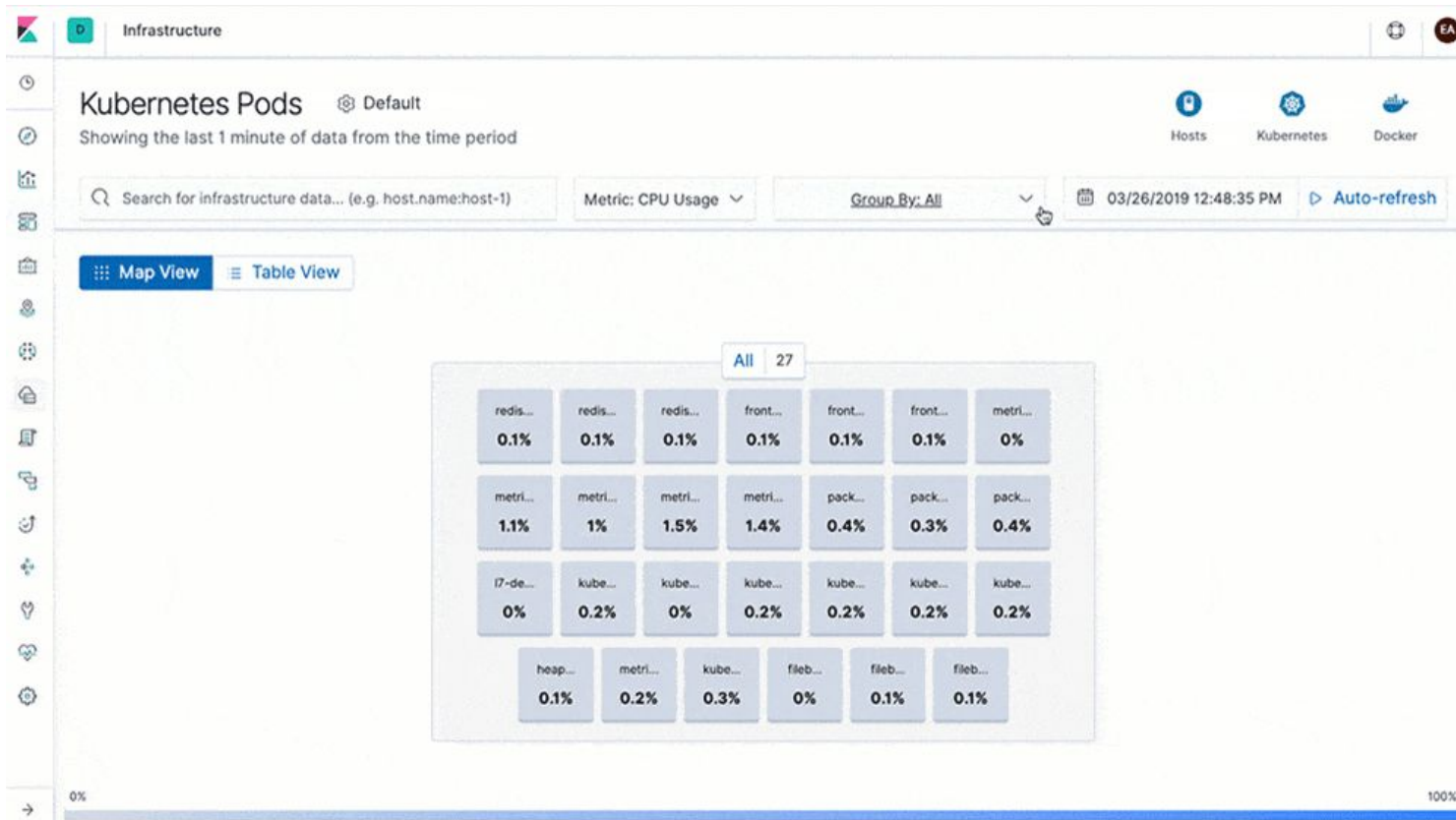
System Overview | Host Overview | Containers Overviews



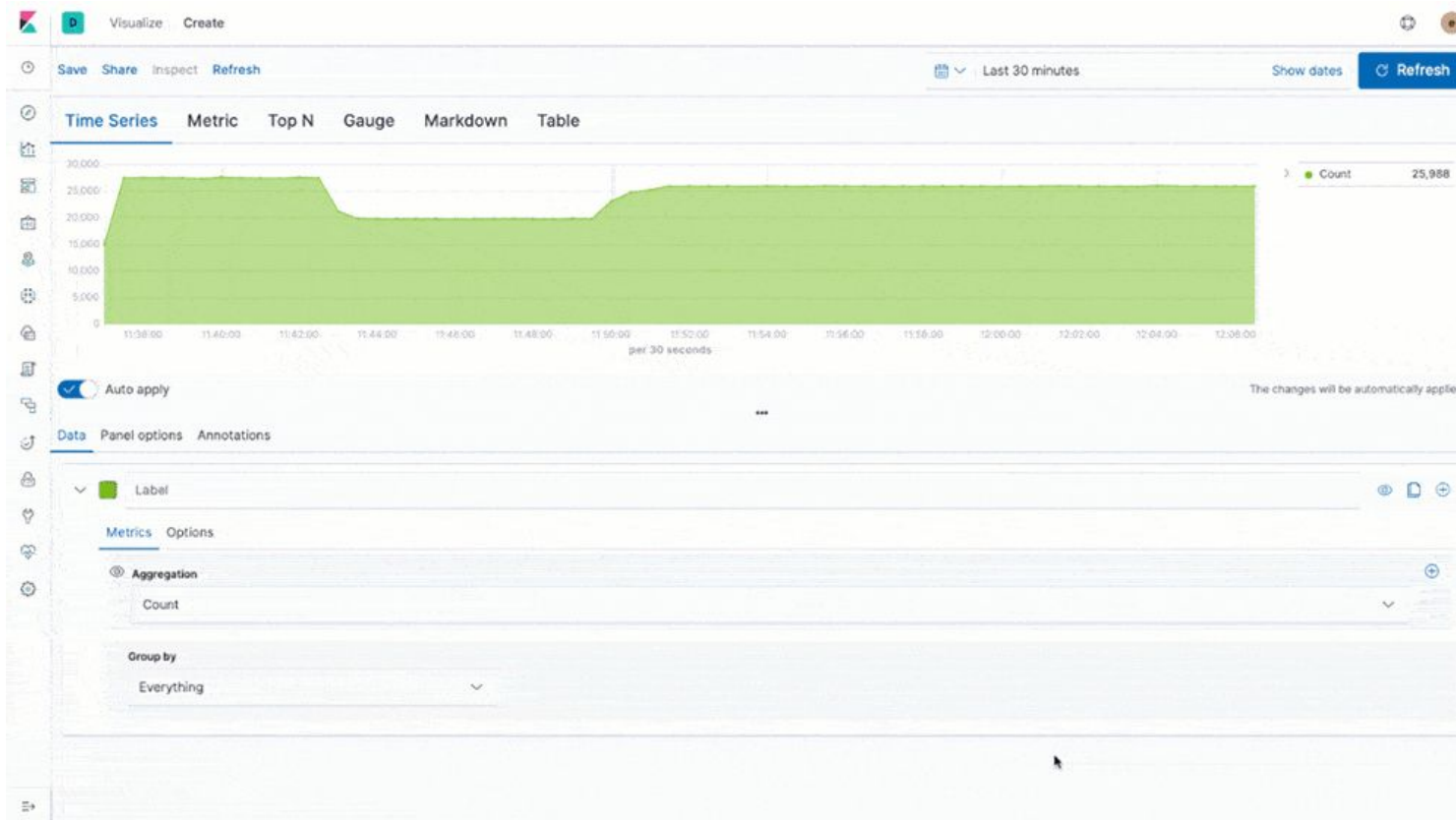
# Metriche d'infrastruttura (Docker)



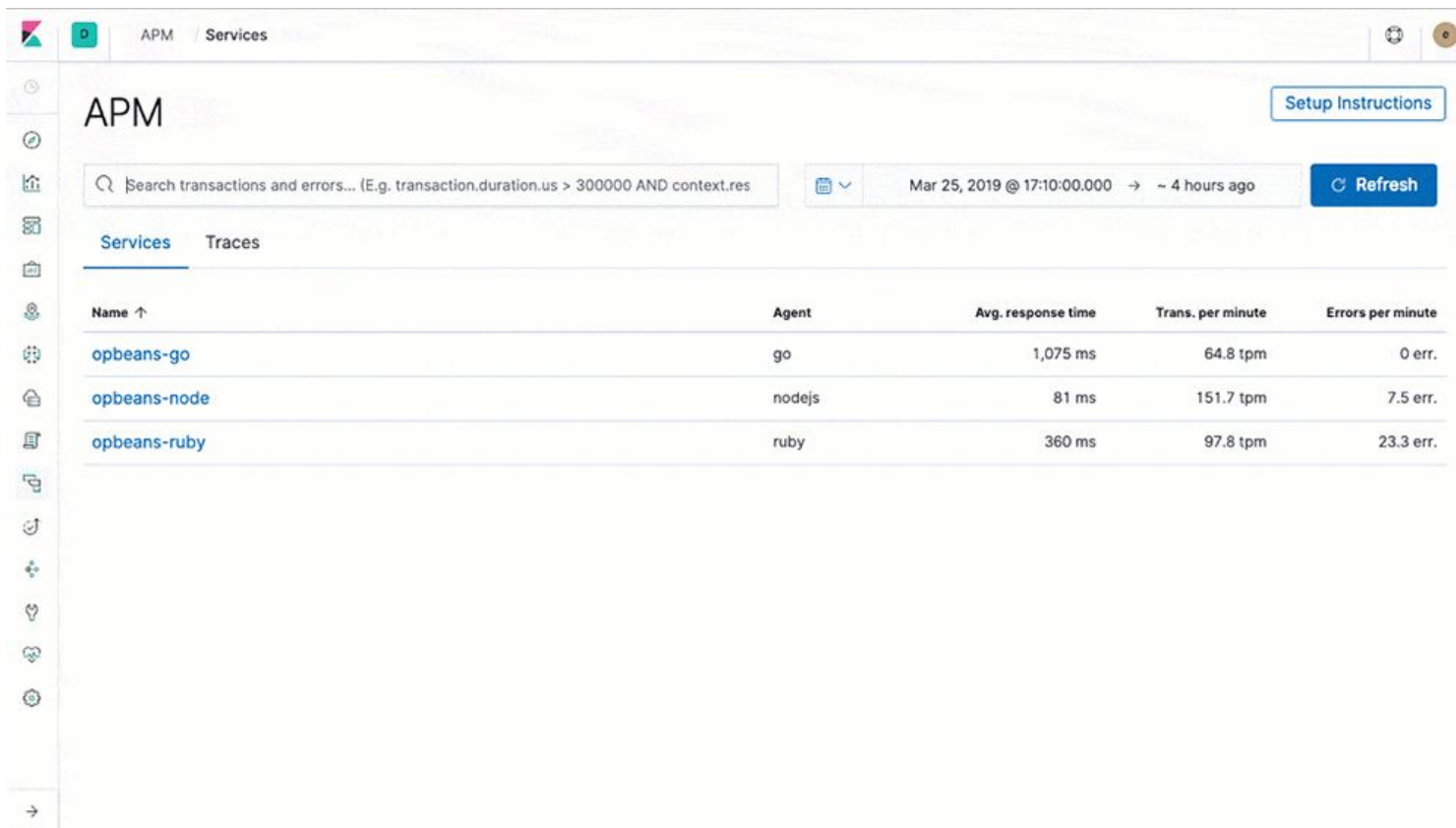
# Metriche d'infrastruttura (Kubernetes)



# Metriche d'infrastruttura (Phrometus)



# APM

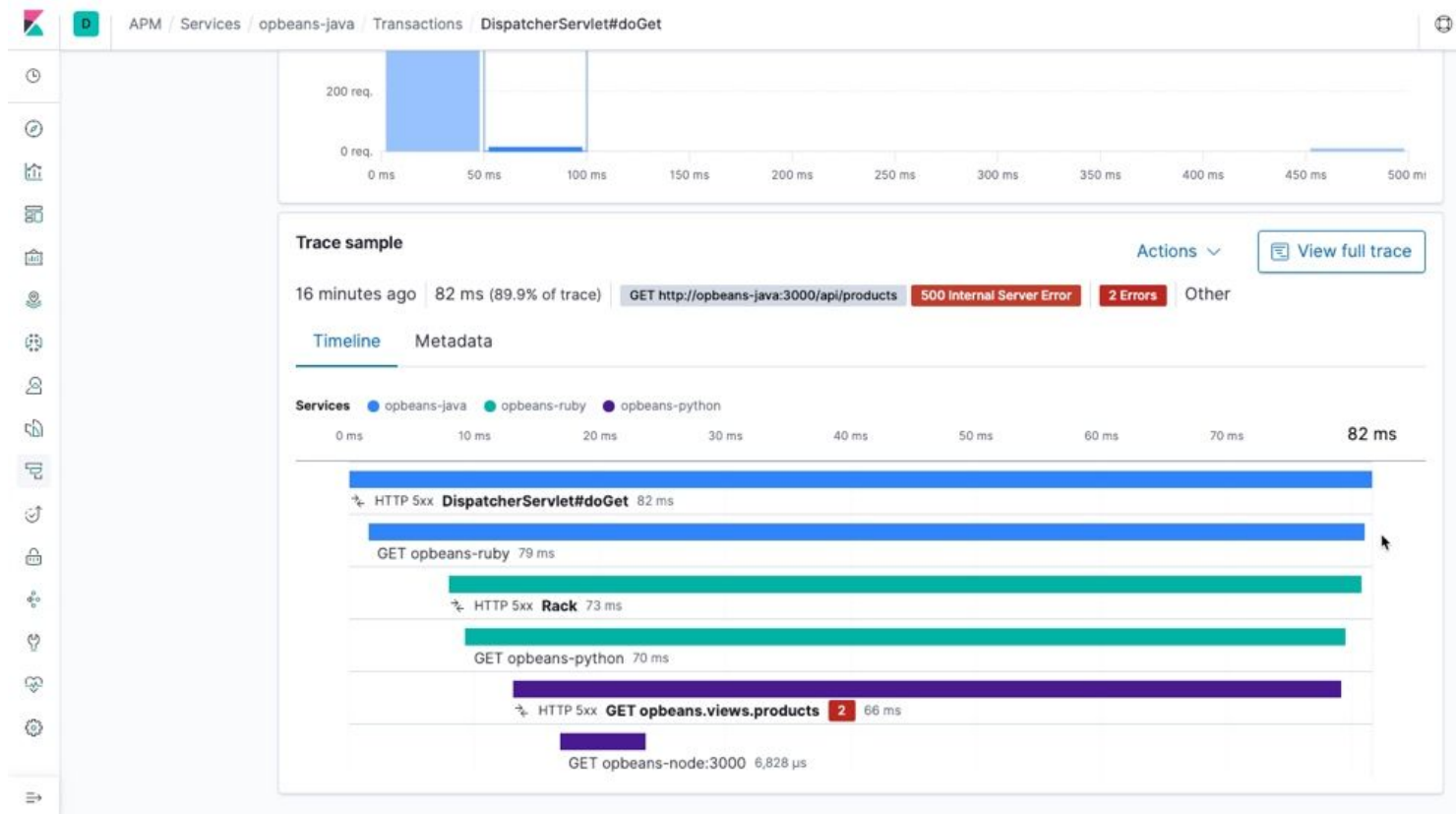


The screenshot shows the Elastic APM interface. At the top, there's a navigation bar with 'APM / Services' and a 'Setup Instructions' button. Below that, a search bar contains the text 'Search transactions and errors... (E.g. transaction.duration.us > 300000 AND context.res'. To the right of the search bar, there's a date range selector set to 'Mar 25, 2019 @ 17:10:00.000' and a 'Refresh' button. The main content area has two tabs: 'Services' (selected) and 'Traces'. Below the tabs is a table with the following data:

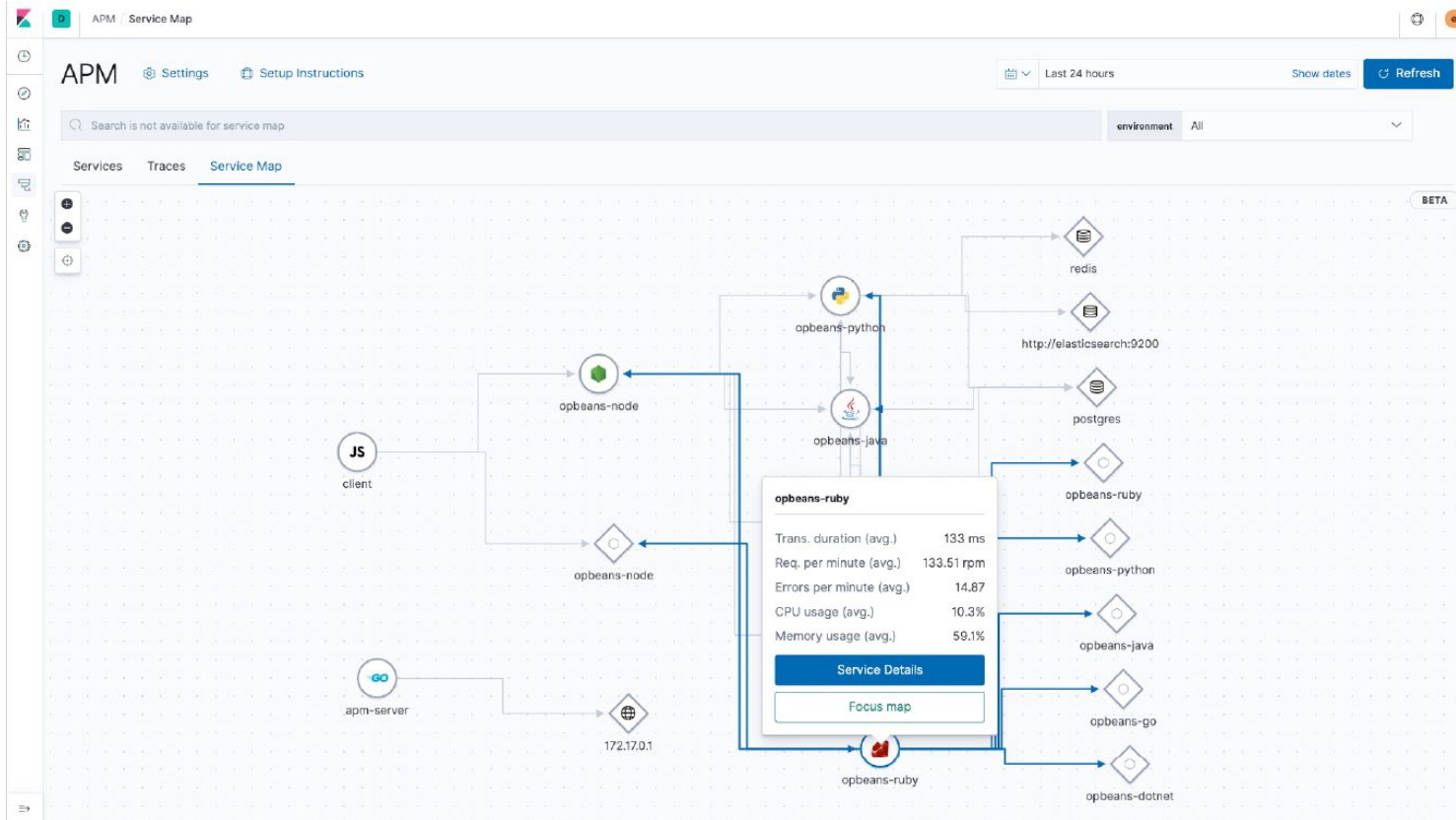
Name ↑	Agent	Avg. response time	Trans. per minute	Errors per minute
<a href="#">opbeans-go</a>	go	1,075 ms	64.8 tpm	0 err.
<a href="#">opbeans-node</a>	nodejs	81 ms	151.7 tpm	7.5 err.
<a href="#">opbeans-ruby</a>	ruby	360 ms	97.8 tpm	23.3 err.



# APM (transaction & span)



# APM (service map)



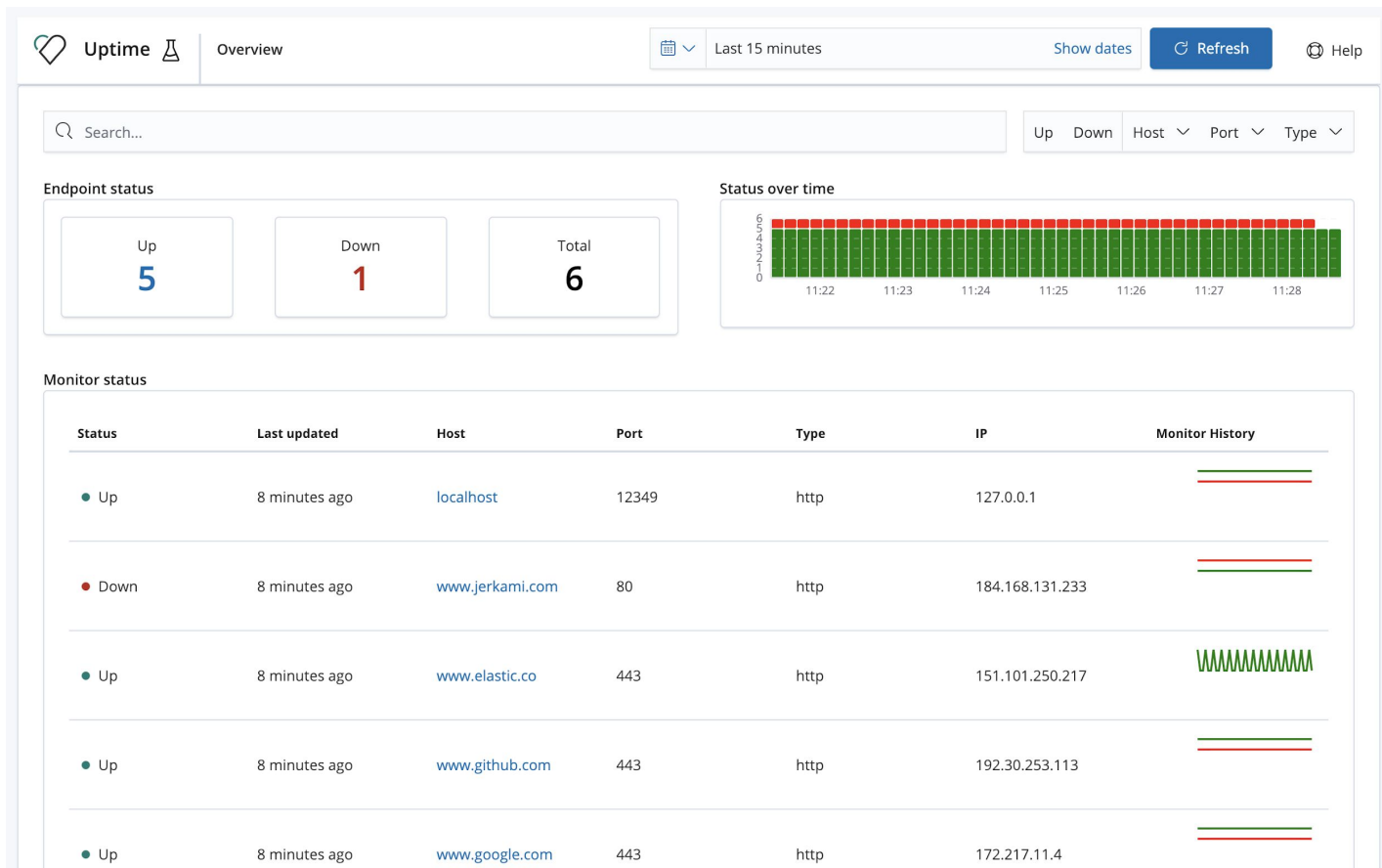
# APM agents

- Java, Go, Node.js, Python, Ruby, .NET
- Real User Monitoring (JavaScript)
- PHP (in lavorazione)



Maggiori info: <https://www.elastic.co/apm>

# Uptime



# Alert automatici con machine learning

Con l'utilizzo di algoritmi di **machine learning** lo stack di Elastic è in grado di evidenziare **automaticamente anomalie** su tutte le serie storiche dei dati



# Riferimenti

- Tanya Bragin, [Observability with the Elastic Stack](#), Elastic blog post, 28 Feb. 2019
- R. I. Cook, [How Complex Systems Fail](#), in Web Operations: O'Reilly, 2010
- Aris Papadopoulos, [Introducing the new alerting framework for Elastic Observability, Elastic Security, and the Elastic Stack](#), Elastic blog post, 13 May 2020
- C. Perrow, *Normal Accidents: Living with High-Risk Technologies*, Princeton University Press, 1999
- Nicolas Rufin, [Why Observability loves the Elastic Common Schema](#), Elastic blog post, 16 May 2019
- Dan Roscigno, [Elasticsearch Observability: Embracing Prometheus and OpenMetrics Standards for Metrics](#), Elastic blog post, 3 Apr. 2019
- Baron Schwartz, [Monitoring Isn't Observability](#), OrangeMatter blog post, 14 Sept. 2017
- Jamie Smith, [Monitoring Applications with Elasticsearch and Elastic APM](#), Elastic blog post, 30 Jan. 2019
- Cindy Sridharan, [Distributed Systems Observability](#), O'Reilly Media, 2018



# Grazie!

Domande?

<https://discuss.elastic.co>

<https://github.com/elastic>

