



I assume you're here because you want to...

# Build an **Incident Response** stack using **OpenTelemetry**



**Annanay Agarwal**  
Senior Software Engineer

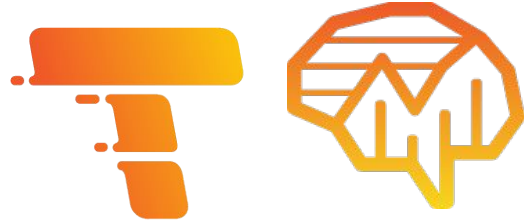
March 16, 2024

# Agenda

Part 1: What is Incident Response Management?

Part 2: OpenTelemetry and its principles

Part 3: Query patterns and building the stack



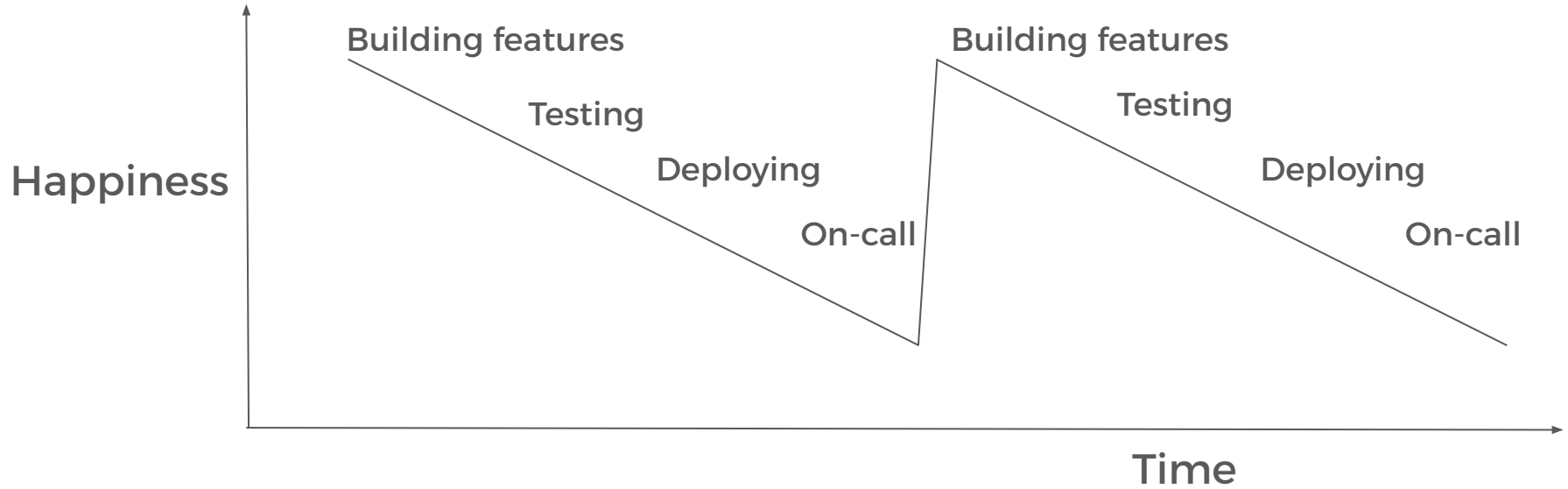
# Part 1: What is Incident Response Management?



**How many of us have been on call?**



# Software Development Life Cycle?





*Current generation internet-facing technology platforms are complex and prone to brittle failure. Without the continuous effort of engineers to keep them running they would stop working -- **many in days, most in weeks, all within a year.***

**Stella report**

*<https://snafucatchers.github.io>*



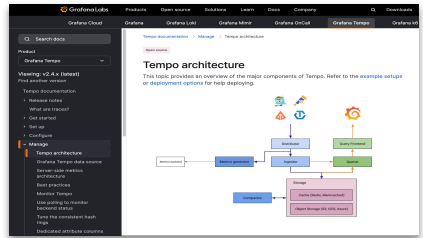
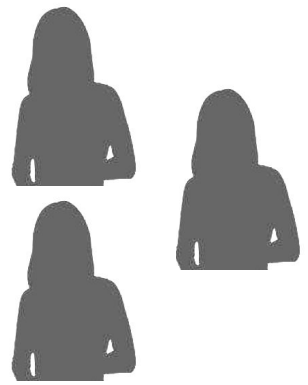
# Things go wrong all the time

- Nodes die
- Processes leak memory
- DNS resolution fails
- Backward incompatible release
- Disks out of space





# Debug.. and coordinate!



# Stress of incident response impacts the human body and mind

Fine motor skills go out the window.  
Field of vision narrows.  
Short term memory is often shot.  
Bias to make decisions faster and with incomplete data.

HUMAN FACTORS, 1995, 37(1), 32-64

## Toward a Theory of Situation Awareness in Dynamic Systems

MICA R. ENDSLEY,<sup>1</sup> *Texas Tech University, Lubbock, Texas*

This paper presents a theoretical model of situation awareness based on its role in dynamic human decision making in a variety of domains. Situation awareness is presented as a predominant concern in system operation, based on a descriptive view of decision making. The relationship between situation awareness and numerous individual and environmental factors is explored. Among these factors, attention and working memory are presented as critical factors limiting operators from acquiring and interpreting information from the environment to form situation awareness, and mental models and goal-directed behavior are hypothesized as important mechanisms for overcoming these limits. The impact of design features, workload, stress, system complexity, and automation on operator situation awareness is addressed, and a taxonomy of errors in situation awareness is introduced, based on the model presented. The model is used to generate design implications for enhancing operator situation awareness and future directions for situation awareness research.

### INTRODUCTION

The range of problems confronting human factors practitioners has continued to grow over the past 50 years. Practitioners must deal with human performance in tasks that are primarily physical or perceptual, as well as consider human behavior involving highly complex cognitive tasks with increasing frequency. As technology has evolved, many complex, dynamic systems have been created that tax the abilities of humans to act as effective, timely decision makers when operating these systems. The operator's situation awareness (SA) will be presented as a crucial construct on which decision making and performance in such systems hinge.

In this paper I strive to show (a) the importance of SA in decision making in dynamic en-

vironments and the utility of using a model of decision making that takes SA into account, and (b) a theory of SA that expands on prior work in this area (Endsley, 1988a, 1990c, 1993b). True SA, it will be shown, involves far more than merely being aware of numerous pieces of data. It also requires a much more advanced level of situation understanding and a projection of future system states in light of the operator's pertinent goals. As such, SA presents a level of focus that goes beyond traditional information-processing approaches in attempting to explain human behavior in operating complex systems.

SA can be shown to be important in a variety of contexts that confront human factors practitioners.

*Aircraft.* In the area with perhaps the longest history, SA was recognized as a crucial commodity for crews of military aircraft as far back as World War I (Press, 1986). SA has grown in importance as a major design goal for civil,

<sup>1</sup> Requests for reprints should be sent to Mica R. Endsley, Department of Industrial Engineering, Texas Tech University, Lubbock, TX 79409.



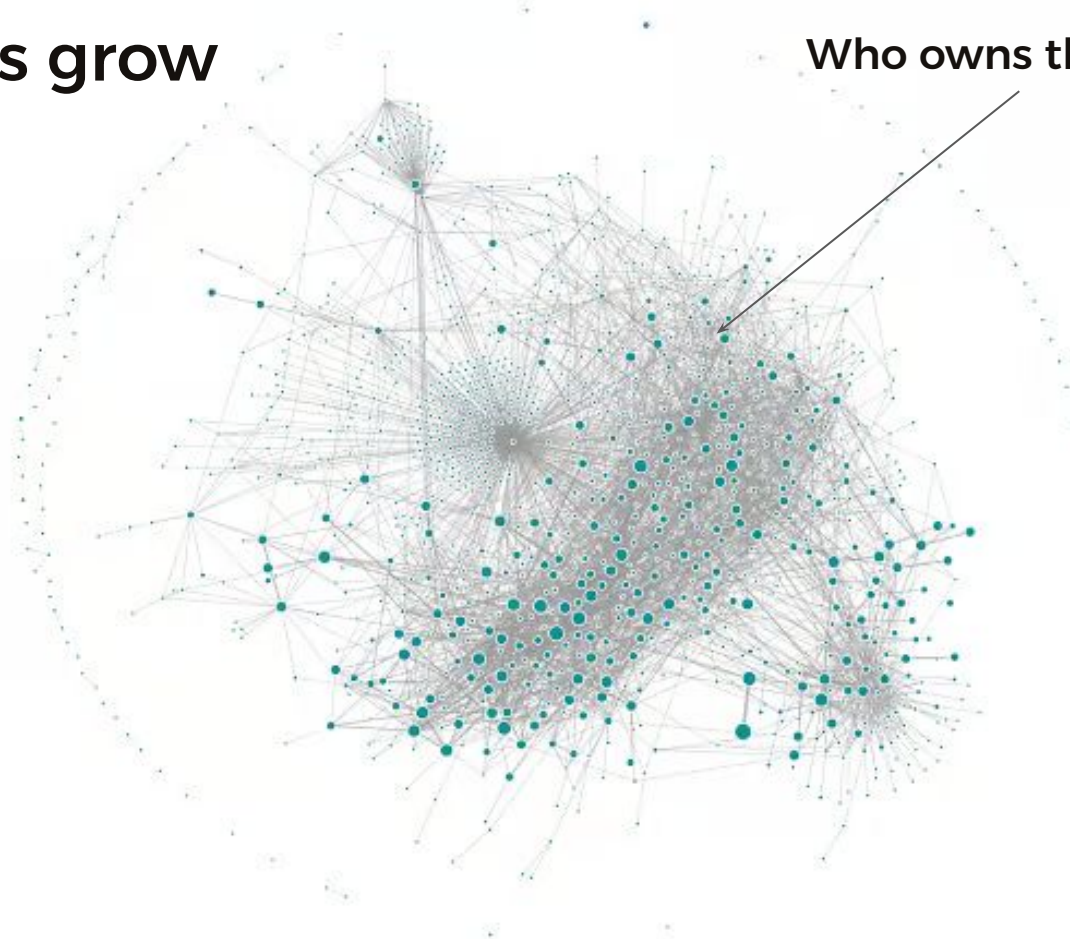
# When you're a smaller company its easier to coordinate

Everyone uses open floor plans these days anyway.



# But companies grow

Who owns this service?



[Uber's microservice architecture circa mid-2018 from Jaeger](#)



<https://fera.com.my/fire-drill-training-malaysia/>

# As a company grows it needs an incident response playbook

## What is an incident?

An incident is an issue with a production system where any of the following is true:

- There may be visible impact for customers
- You need to involve a second squad to fix the problem
- The issue is unresolved after an hour of concentrated analysis

## What is incident response?

A broad term that describes the processes we follow when something unexpected happens. It covers:

- Roles and responsibilities
- Bringing the right people together
- Keeping track of what's going on
- Communicating internally and externally
- Creating an understanding of why incidents occur so we can avoid or minimise them in future



# Roles

## Commander

The commander keeps the incident moving towards a resolution by ensuring we are coordinating, communicating, and documenting.

- Shield and support the investigator
  - Communicates for them
  - Helps with prioritization
  - Provides escalation and resourcing support

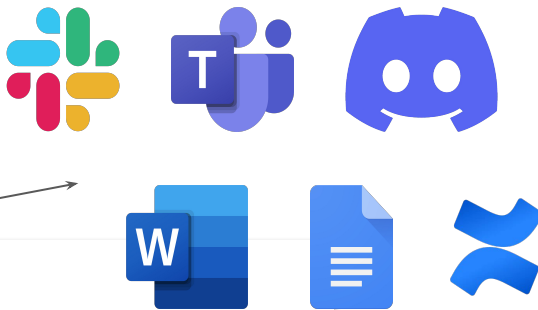
## Investigator

The Investigator diagnoses and resolves the incident.

- Determine impact
- Identify temporary or permanent fixes to stop the bleeding
  - “Stop the bleeding” is a term used by paramedics to resolve the immediate impact of heavy trauma first, rather than to seek a perfect solution



# Tools



## Commander

The commander keeps the incident moving towards a resolution by ensuring we are coordinating, communicating, and documenting.

- Shield and support the investigator
  - Communicates for them
  - Helps with prioritization
  - Provides escalation and resourcing support

## Investigator

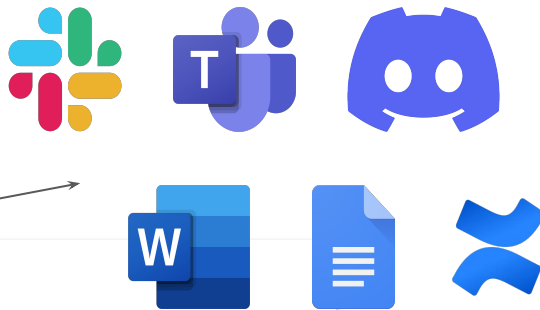
The Investigator diagnoses and resolves the incident.

- Determine impact
- Identify temporary or permanent fixes to stop the bleeding
  - “Stop the bleeding” is a term used by paramedics to resolve the immediate impact of heavy trauma first, rather than to seek a perfect solution





# Tools



## Commander

The commander keeps the incident moving towards a resolution by ensuring we are coordinating, communicating, and documenting.

- Shield and support the investigator
  - Communicates for them
  - Helps with prioritization
  - Provides escalation and resourcing support

## Investigator

The Investigator diagnoses and resolves the incident.

- Determine impact
- Identify temporary or permanent fixes to stop the bleeding
  - “Stop the bleeding” is a term used by paramedics to resolve the immediate impact of heavy trauma first, rather than to seek a perfect solution



Part 2:  
OpenTelemetry  
and its  
principles



# Observability in IRM



[CEO of Zomato tweets about New Year eve's war room](#)

# Where do I start?

## Core resources first

CPU, Memory, Networks – k8s nodes, database services, etc

Basic services – what does everything else depend on?

CI/CD – when was the last deployment? What versions are out there?



Lou Gold, CC BY-NC-SA 2.0 Deed

# Where do I start?

## Core resources first

CPU, Memory, Networks – k8s nodes, database services, etc

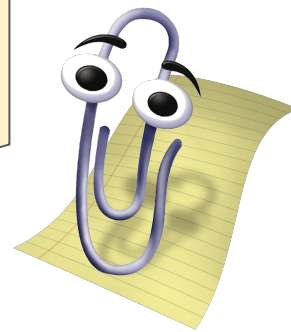
Basic services – what does everything else depend on?

CI/CD – when was the last deployment? What versions are out there?

Did you know: the prometheus community has written exporters for many commonly used software already



<https://prometheus.io/docs/instrumenting/exporters/>



# Where do I start?

## Core resources first

CPU, Memory, Networks – k8s nodes, database services, etc

Basic services – what does everything else depend on?

CI/CD – when was the last deployment? What versions are out there?

## Applications next

Services implementing business logic





OpenTelemetry is a collection of APIs, SDKs, and tools. Use it to instrument, generate, collect, and export telemetry data (metrics, logs, and traces) to help you analyze your software's performance and behavior.

OpenTelemetry is generally available across several languages and is suitable for use.



## Instrumentation points



OpenTelemetry is a collection of **APIs, SDKs, and tools**. Use it to instrument, generate, collect, and export telemetry data (**metrics, logs, and traces**) to help you analyze your software's performance and behavior.

OpenTelemetry is **generally available** across **several languages** and is suitable for use.



# Instrumentation points

```
import (  
    "context"  
    "fmt"  
    "log"  
    "os"  
    "os/signal"  
    "time"  
  
    "google.golang.org/grpc"  
    "google.golang.org/grpc/credentials/insecure"  
  
    "go.opentelemetry.io/otel"  
    "go.opentelemetry.io/otel/attribute"  
    "go.opentelemetry.io/otel/exporters/otlp/otlptrace/otlptracegrpc"  
    "go.opentelemetry.io/otel/propagation"  
    "go.opentelemetry.io/otel/sdk/resource"  
    sdktrace "go.opentelemetry.io/otel/sdk/trace"  
    semconv "go.opentelemetry.io/otel/semconv/v1.24.0"  
    "go.opentelemetry.io/otel/trace"  
)
```

SDK

```
tracer := otel.Tracer("test-tracer")  
  
// Attributes represent additional key-value descriptors that can be bound  
// to a metric observer or recorder.  
commonAttrs := []attribute.KeyValue{  
    attribute.String("attrA", "chocolate"),  
    attribute.String("attrB", "raspberry"),  
    attribute.String("attrC", "vanilla"),  
}  
  
// work begins  
ctx, span := tracer.Start(  
    ctx,  
    "CollectorExporter-Example",  
    trace.WithAttributes(commonAttrs...))  
defer span.End()  
for i := 0; i < 10; i++ {  
    _, iSpan := tracer.Start(ctx, fmt.Sprintf("Sample-%d", i))  
    log.Printf("Doing really hard work (%d / 10)\n", i+1)  
  
    <-time.After(time.Second)  
    iSpan.End()  
}
```

API



Language	Traces	Metrics	Logs
<a href="#">C++</a>	Stable	Stable	Stable
<a href="#">C#/.NET</a>	Stable	Stable	Stable
<a href="#">Erlang/Elixir</a>	Stable	Experimental	Experimental
<a href="#">Go</a>	Stable	Stable	In development
<a href="#">Java</a>	Stable	Stable	Stable
<a href="#">JavaScript</a>	Stable	Stable	Experimental
<a href="#">PHP</a>	Stable	Stable	Stable
<a href="#">Python</a>	Stable	Stable	Experimental
<a href="#">Ruby</a>	Stable	In development	In development
<a href="#">Rust</a>	Beta	Alpha	Alpha
<a href="#">Swift</a>	Stable	Experimental	In development



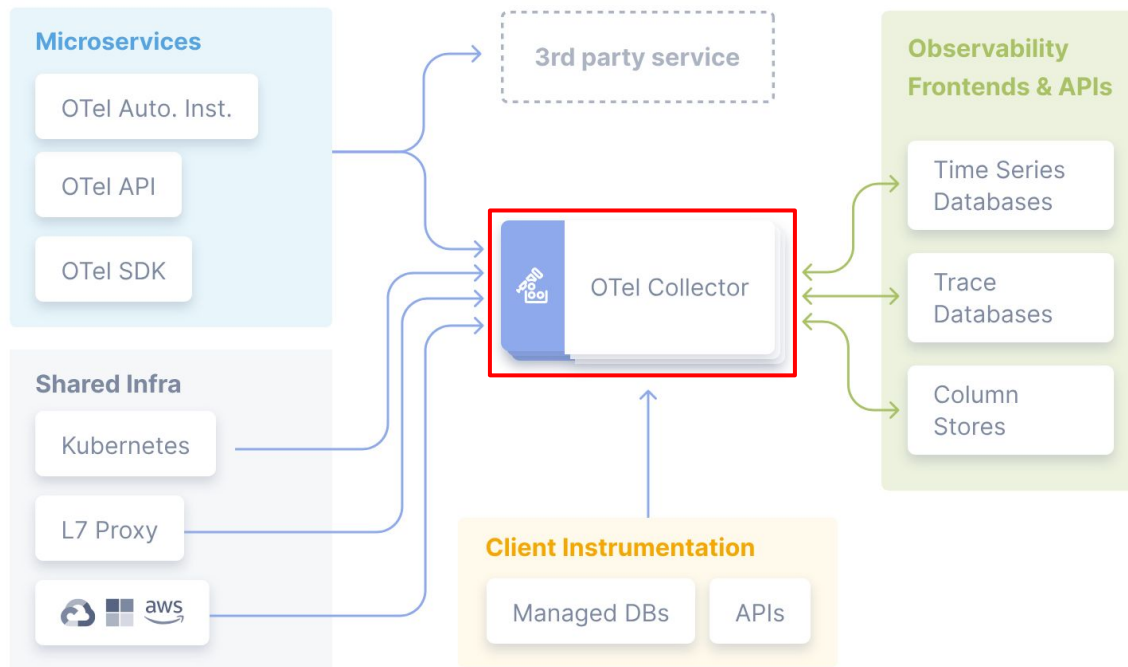
Instrumentation points

Ex: Collector (data pipeline)

OpenTelemetry is a collection of **APIs, SDKs, and tools**. Use it to instrument, generate, collect, and export telemetry data (**metrics, logs, and traces**) to help you analyze your software's performance and behavior.

OpenTelemetry is **generally available** across **several languages** and is suitable for use.

# Collector (data pipeline)



<https://opentelemetry.io/docs/>



Instrumentation points

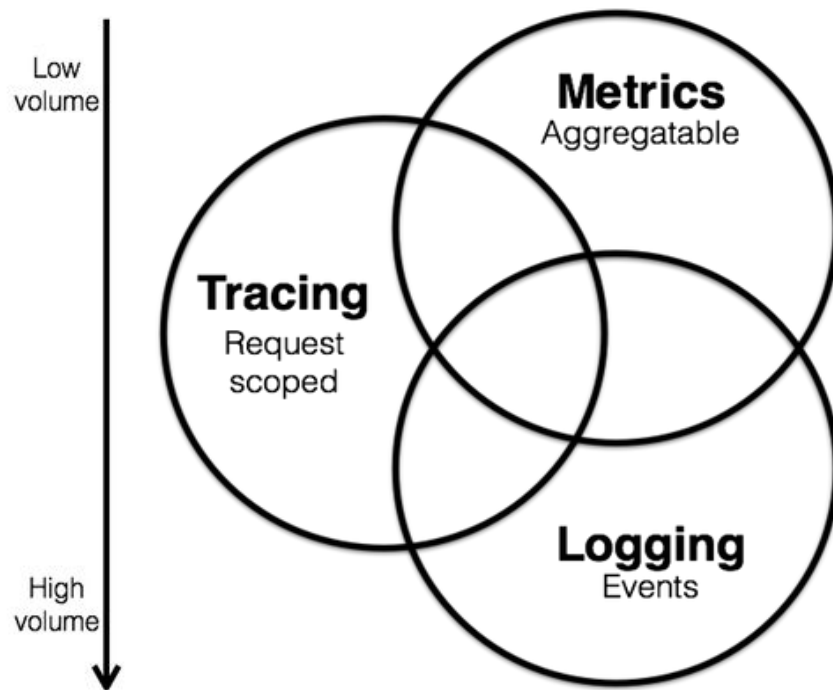
Ex: Collector (data pipeline)

signals

OpenTelemetry is a collection of **APIs, SDKs, and tools**. Use it to instrument, generate, collect, and export telemetry data (**metrics, logs, and traces**) to help you analyze your software's performance and behavior.

OpenTelemetry is **generally available** across **several languages** and is suitable for use.

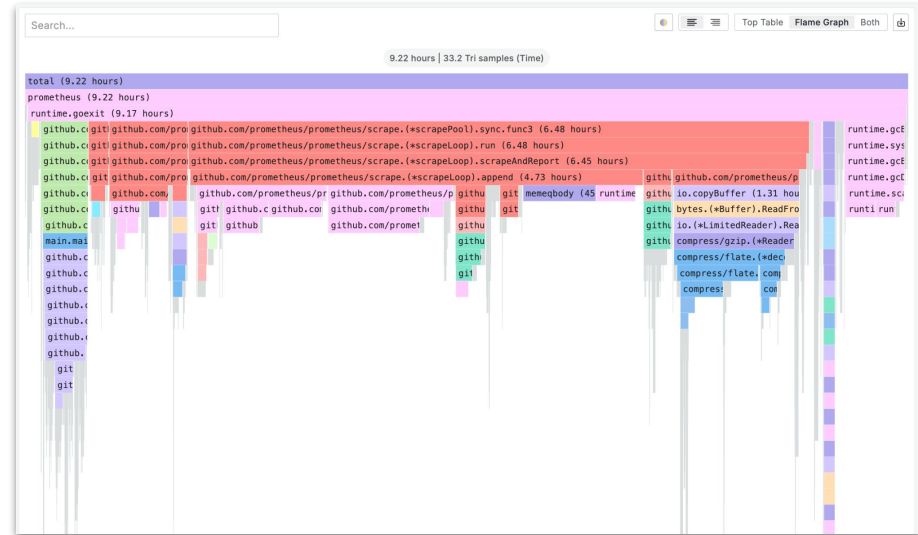
# Signals



<https://peter.bourgon.org/blog/2017/02/21/metrics-tracing-and-logging.html>

# There can be more signals...

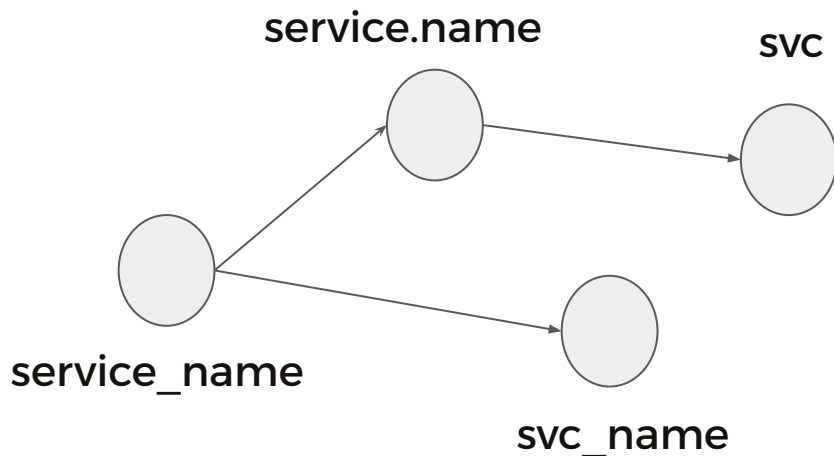
Continuous profiling helps finding and debugging painful performance issues down to the function and line of code.



# Semantic conventions

Defines a common set of attributes which provide meaning to data when collecting, producing and consuming it.

```
db.connection_string  
db.instance.id  
k8s.cluster.name  
k8s.namespace.name  
... •
```





# Part 3: Query patterns and building the stack.



# Choosing storage

So far we've seen instrumentation and the data pipeline. That's where OTel specifications and implementations end.

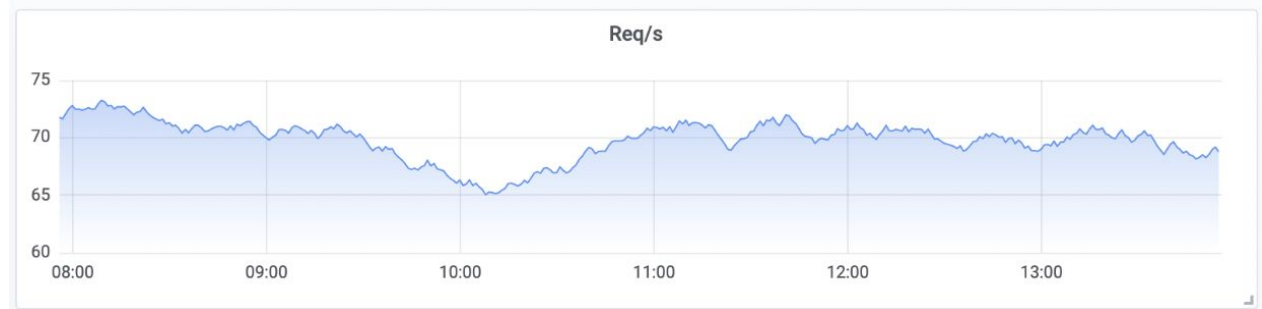
BYO-Storage.

Let's see some examples from PromQL, LogQL and TraceQL.



# Query Patterns - Metrics

```
rate(  
  tempo_request_duration_seconds_count{  
    job="query-frontend",  
    route=~"tempo_api_.*"  
  }[1m]  
)
```



# Query Patterns - Metrics

Aggregate based on different labels

```
sum by (cluster, namespace) (  
  rate(  
    tempo_request_duration_seconds_count{  
      job="query-frontend",  
      route=~"tempo_api_.*"  
    } [1m]  
  )  
)
```

# Query Patterns - Metrics

Percentiles

```
histogram_quantile(  
  0.95,  
  sum(  
    rate(  
      tempo_request_duration_seconds_bucket{  
        job="query-frontend",  
        route=~"tempo_api_.*"  
      } [1m])  
    )  
  )  
)
```

# Query Patterns - Logs

Search for keyword – spot errors

```
{cluster="ops-us-east-0", namespace="loki-ops"} |= "error"
```

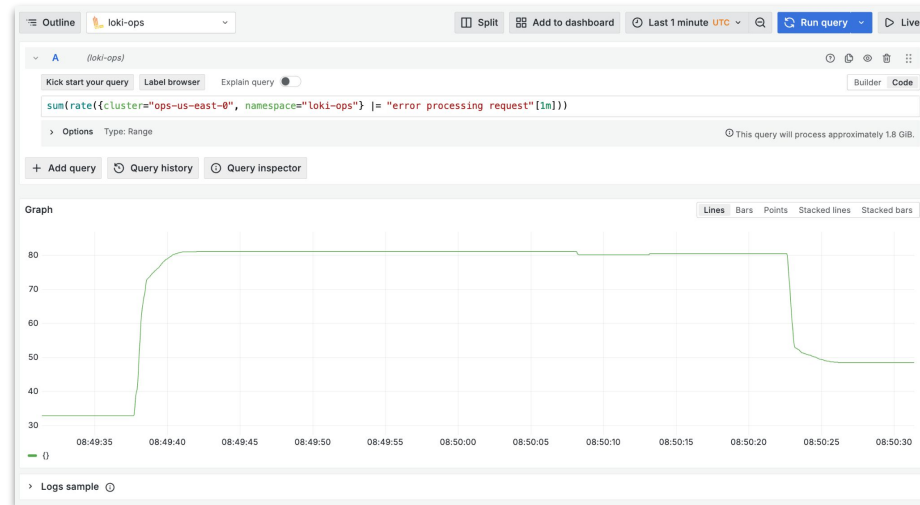
The screenshot shows the Grafana Loki query interface. At the top, the query editor contains the query: `{cluster="ops-us-east-0", namespace="loki-ops"} |= "error"`. Below the query editor is a bar chart titled "Logs volume" showing the number of logs over time. The x-axis represents time from 08:51:50 to 08:52:45, and the y-axis represents the number of logs, ranging from 0 to 60. There are two prominent bars: one at approximately 08:52:20 with a volume of about 60, and another at approximately 08:52:25 with a volume of about 30. Below the chart is a "Logs" section with various filters and options. The "Display results" dropdown is set to "Newest first". The log entries are displayed in a list, with the error message `error processing request` highlighted in orange in each entry. The log entries are as follows:

```
> 2024-03-15 08:52:25.982 level=error ts=2024-03-15T08:52:25.897257375Z caller=scheduler_processor.go:110 component=querier msg="error processing request" from scheduler"
err="rpc error: code = Canceled desc = context canceled" addr=10.136.134.10:9095
> 2024-03-15 08:52:25.982 level=error ts=2024-03-15T08:52:25.897245638Z caller=scheduler_processor.go:110 component=querier msg="error processing request" from scheduler"
err="rpc error: code = Canceled desc = context canceled" addr=10.137.29.112:9095
> 2024-03-15 08:52:25.982 level=error ts=2024-03-15T08:52:25.897234824Z caller=scheduler_processor.go:110 component=querier msg="error processing request" from scheduler"
err="rpc error: code = Canceled desc = context canceled" addr=10.137.29.112:9095
> 2024-03-15 08:52:25.982 level=error ts=2024-03-15T08:52:25.897223152Z caller=scheduler_processor.go:110 component=querier msg="error processing request" from scheduler"
err="rpc error: code = Canceled desc = context canceled" addr=10.137.29.112:9095
> 2024-03-15 08:52:25.982 level=error ts=2024-03-15T08:52:25.897211553Z caller=scheduler_processor.go:110 component=querier msg="error processing request" from scheduler"
err="rpc error: code = Canceled desc = context canceled" addr=10.137.29.112:9095
```

# Query Patterns - Logs

Rate of increase in keyword over time – helps spot trends

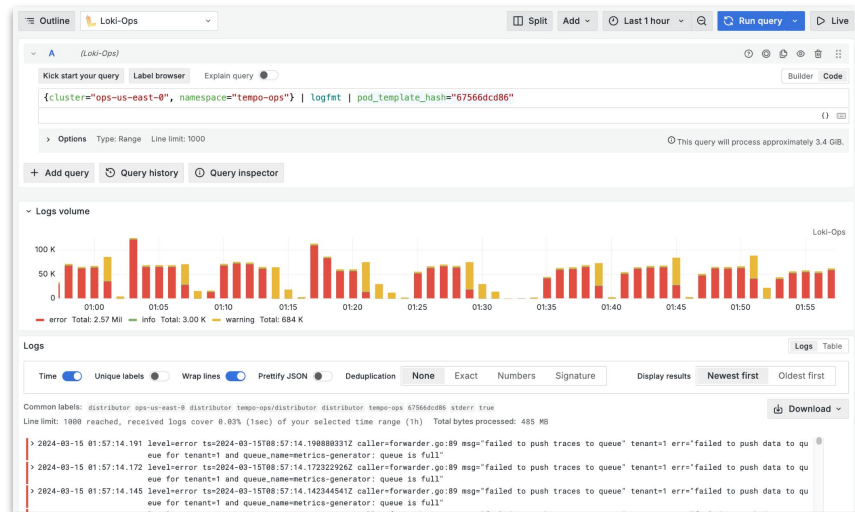
```
count_over_time(  
  {cluster="ops-us-east-0", namespace="loki-ops"}  
  |= "error processing request" [1m]  
)
```



# Query Patterns - Logs

Extensible schema – don't force early lock in

```
{cluster="ops-us-east-0", namespace="tempo-ops"}  
| logfmt  
| client_ip="192.168.0.10"
```





# Query Patterns - Traces

Search for specific labels and latency

```
{ cluster="ops-us-east-0" && namespace="tempo-ops" && duration > 2s }
```

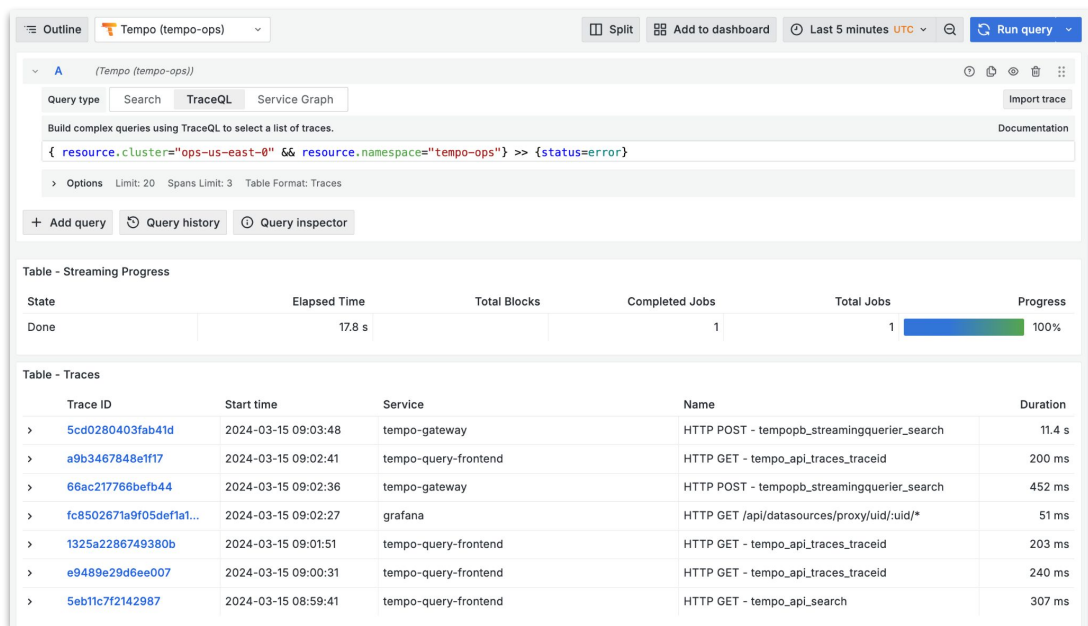
The screenshot shows the Tempo web interface. At the top, there's a navigation bar with 'Outline', 'Tempo (tempo-ops)', 'Split', 'Add to dashboard', 'Last 1 minute UTC', and 'Run query'. Below this is a search bar with 'Query type' set to 'TraceQL'. The query input field contains: `{ resource.cluster="ops-us-east-0" && resource.namespace="tempo-ops" && duration > 2s }`. Below the query, there are options for 'Limit: 20', 'Spans Limit: 3', and 'Table Format: Traces'. A progress bar shows 'Done' with '647 ms' elapsed time, '1' completed job, and '1' total job, reaching 100% progress. The main table, titled 'Table - Traces', lists several trace entries with columns for Trace ID, Start time, Service, Name, and Duration.

Trace ID	Start time	Service	Name	Duration
> 498b7d7aa66e4132	2024-03-15 09:02:27.629	tempo-querier	Poller.Do	2.56 s
> 6a2e180a85915	2024-03-15 09:02:26.656	tempo-querier	Poller.Do	2.73 s
> 6e788da12a41ebbb	2024-03-15 09:02:26.337	tempo-querier	Poller.Do	2.30 s
> 3bc26d8464d1f4b	2024-03-15 09:02:26.094	tempo-querier	Poller.Do	2.49 s
> 380cc3b92cc1ed56	2024-03-15 09:02:25.681	tempo-querier	Poller.pollTenantAndCreateIndex	2.83 s
> 4b4b422eb61f3b05	2024-03-15 09:02:25.036	tempo-compactior	Poller.Do	2.75 s
> 511b1690cd12ca30	2024-03-15 09:02:24.068	tempo-querier	Poller.pollTenantAndCreateIndex	2.49 s
> 6a2a0d67939b8cdb	2024-03-15 09:02:23.992	tempo-querier	Poller.Do	2.85 s

# Query Patterns - Traces

Search based on structure of trace

```
{ span.http.route = "/api/failing" } >> { status = error }
```



The screenshot shows the Tempo web interface. At the top, there's a navigation bar with 'Outline', 'Tempo (tempo-ops)', 'Split', 'Add to dashboard', 'Last 5 minutes UTC', and 'Run query'. Below this, the 'TraceQL' editor is active, showing the query: `{ resource.cluster="ops-us-east-0" && resource.namespace="tempo-ops" } >> { status=error }`. The interface also includes options for 'Limit: 20', 'Spans Limit: 3', and 'Table Format: Traces'. Below the query editor, there's a 'Table - Streaming Progress' section with a progress bar at 100%. The main section is 'Table - Traces', which lists several trace entries with columns for Trace ID, Start time, Service, Name, and Duration.

Trace ID	Start time	Service	Name	Duration
> 5cd0280403fab41d	2024-03-15 09:03:48	tempo-gateway	HTTP POST - tempopb_streamingquerier_search	11.4 s
> a9b3467848ef1f7	2024-03-15 09:02:41	tempo-query-frontend	HTTP GET - tempo_api_traces_traceid	200 ms
> 66ac217766befb44	2024-03-15 09:02:36	tempo-gateway	HTTP POST - tempopb_streamingquerier_search	452 ms
> fc8502671a9f05def1a1...	2024-03-15 09:02:27	grafana	HTTP GET /api/datasources/proxy/uid/uid/*	51 ms
> 1325a2286749380b	2024-03-15 09:01:51	tempo-query-frontend	HTTP GET - tempo_api_traces_traceid	203 ms
> e9489e29d6ee007	2024-03-15 09:00:31	tempo-query-frontend	HTTP GET - tempo_api_traces_traceid	240 ms
> 5eb1c7f2142987	2024-03-15 08:59:41	tempo-query-frontend	HTTP GET - tempo_api_search	307 ms

# Query Patterns - Traces

Arbitrary metrics from traces.

```
{ cluster = "foo" && namespace = "bar" && status = error }  
  | rate() by (span.customerID)
```

# Traces - Query Patterns

Arbitrary metrics from traces.

```
{ cluster = "foo" && namespace = "bar" && status = error }  
  | rate() by (span.customerID)
```

```
{ cluster = "foo" && namespace = "bar" && status = error &&  
span.customerID = "7283895" }  
  | rate() by (span.http.api)
```

```
{ cluster = "foo" && namespace = "bar" && status = error &&  
span.customerID = "7283895" && span.http.api = "/helloworld"}  
  | rate() by (span.http.method)
```

# Build on top of these query patterns!



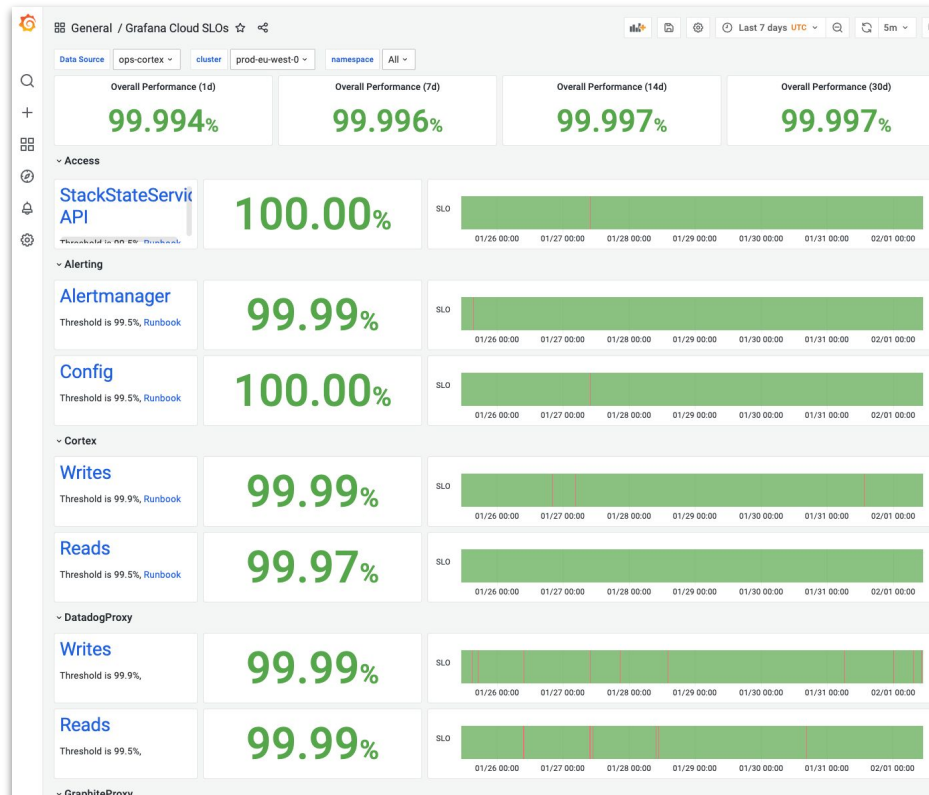
} Purpose built tools & automation.

# 1. Alert on the right data

## Alert for Symptoms, not causes

Ex: Alert for **user impact** and not on restarting pods. **Identify SLOs!**

Drive adoption!

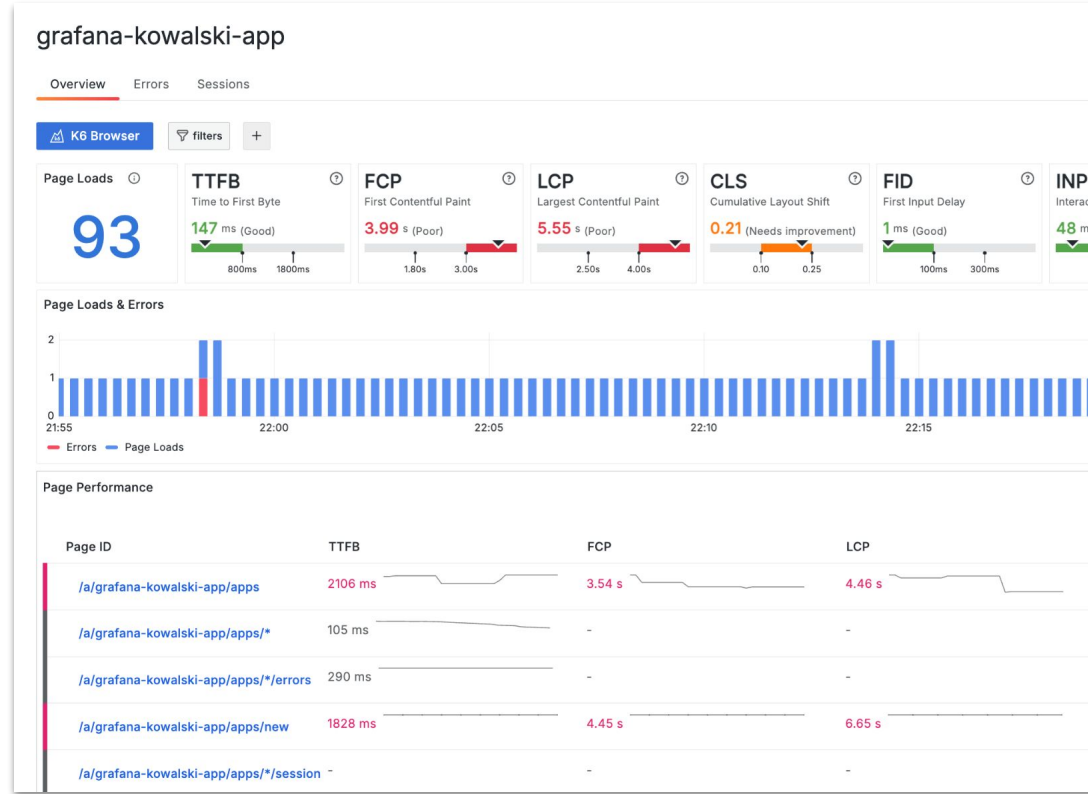


## 2. App specific debugging workflows

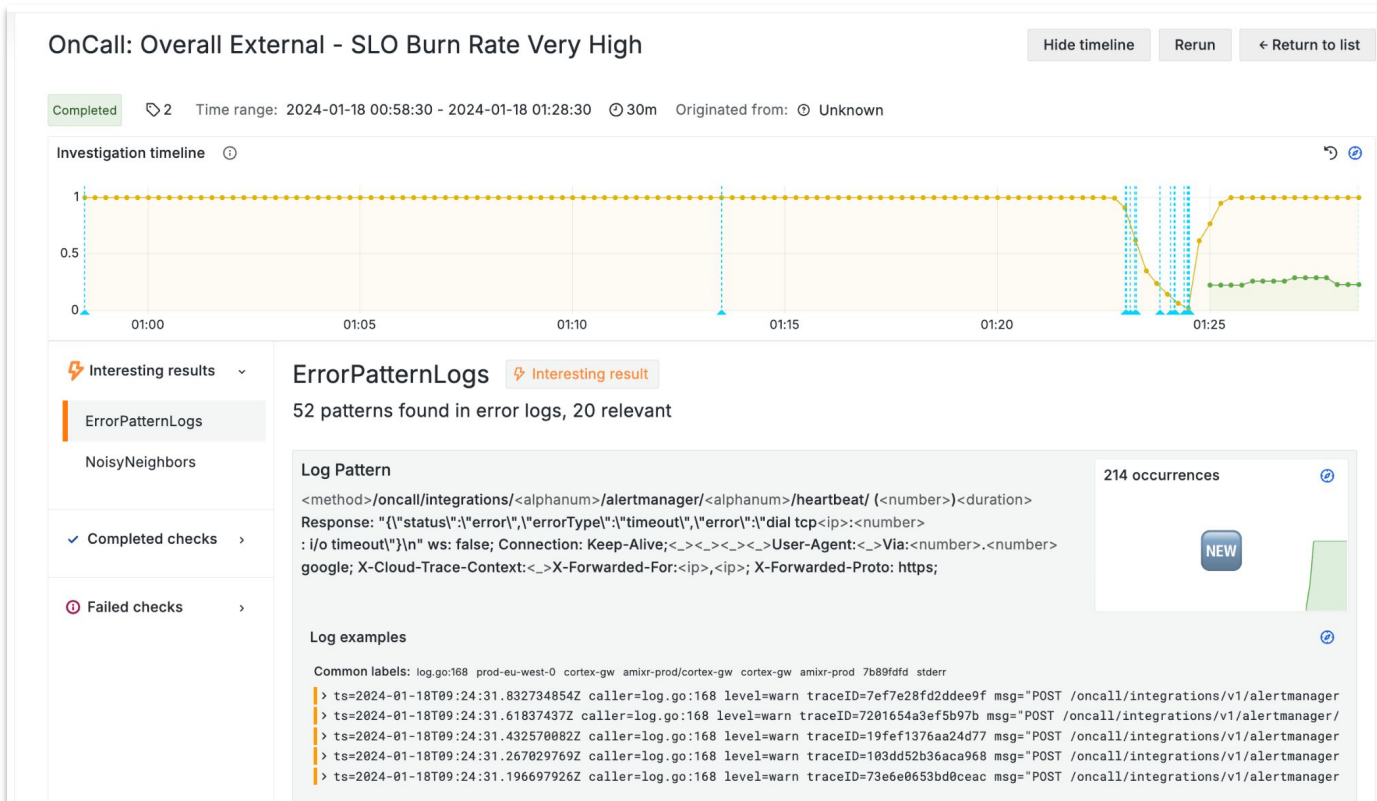
Build debugging workflows with knowledge of what the app does.

Is it a frontend app? Use **Real User Monitoring**.

Is it a backend app? Use **RED metrics**.

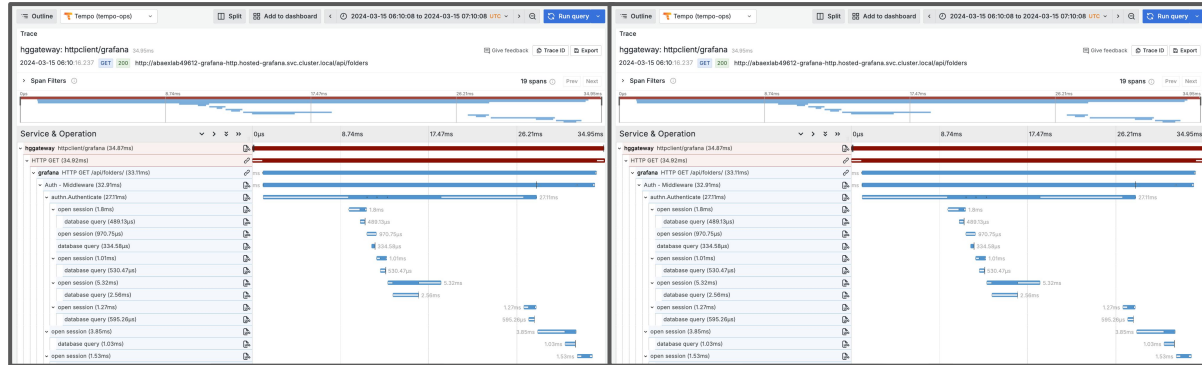


# 3. Automate change detection (logs)





# 3. Automate change detection (traces ex.)



Corporate needs you to find the differences between this picture and this picture.

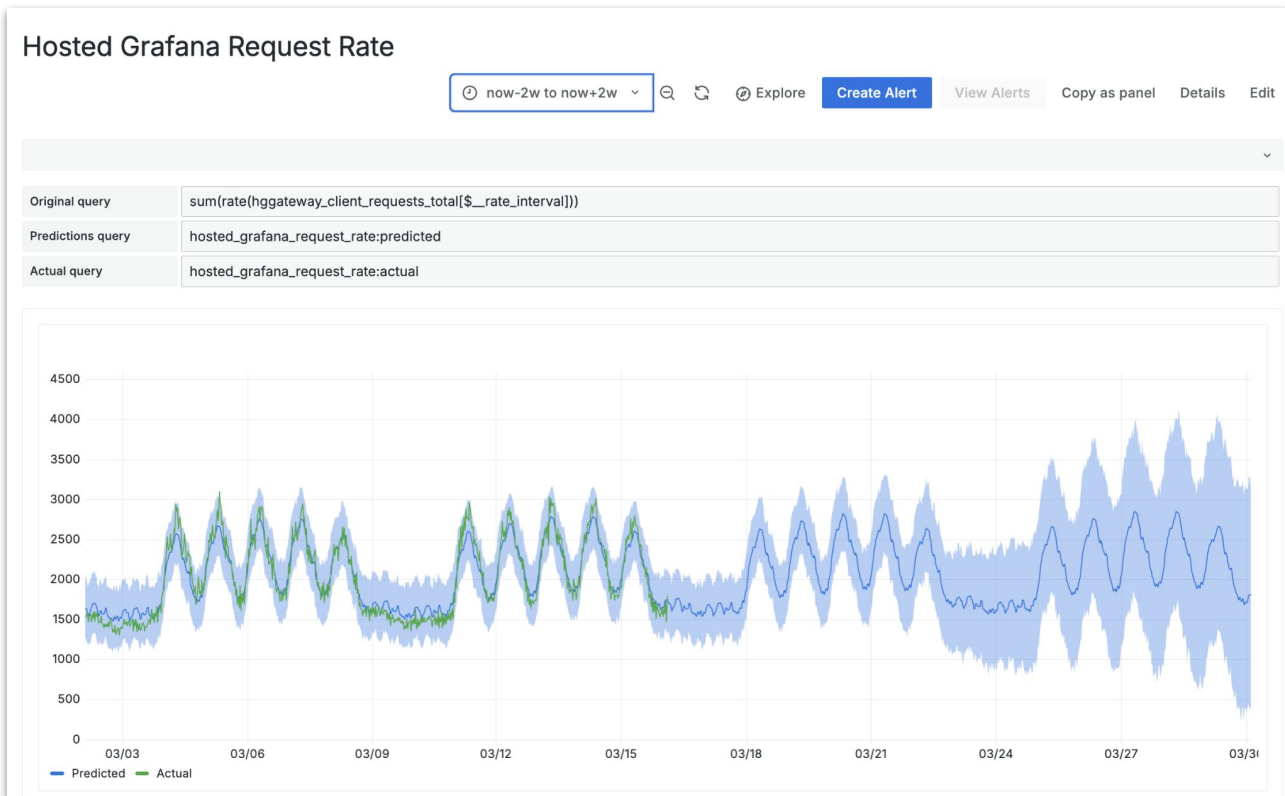


They're the same picture.

# 3. Automate change detection (metrics)



# 4. Forecasting



<https://github.com/facebook/prophet>

<https://grafana.com/docs/grafana-cloud/alerting-and-irm/machine-learning/>



**If you found any of these useful, we are continuously improving our open source software. Come talk to us at our booth or in our community calls!**

# Thank you! Questions?



@mrannanay

