Grafana Labs

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

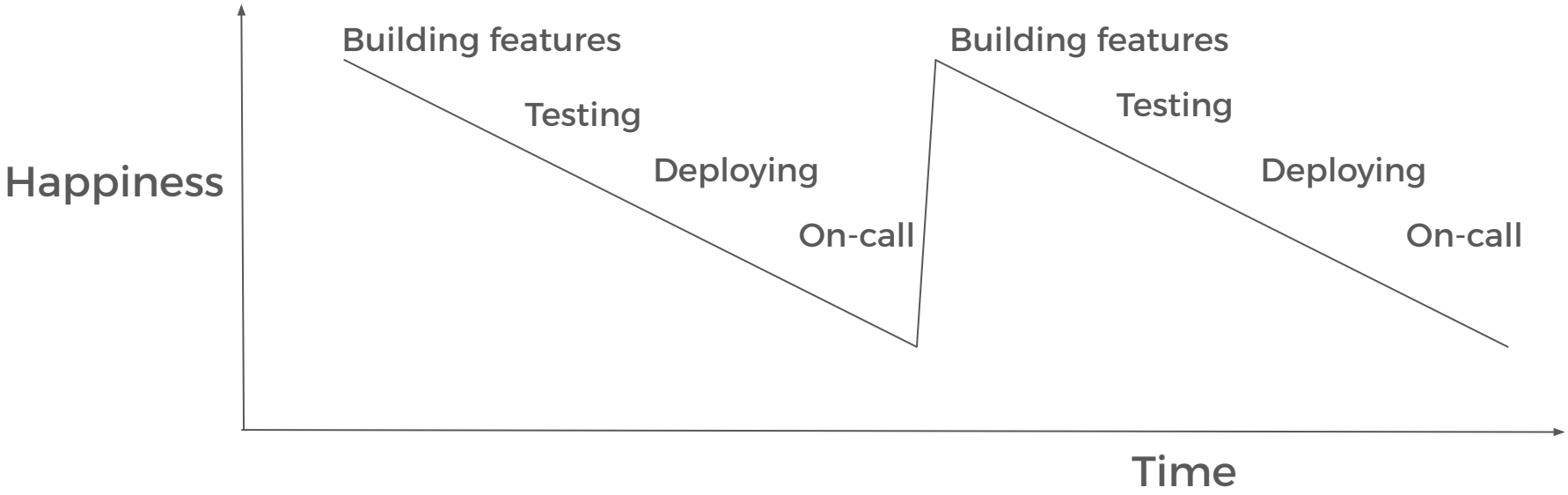

GRAFANA
FEST 2022


GRAFANA
WAY

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**.*

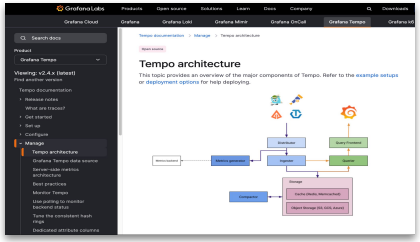
Stella report

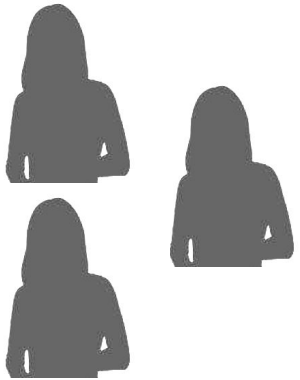
https://snafucatchers.github.io

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,[1] *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 environments 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,

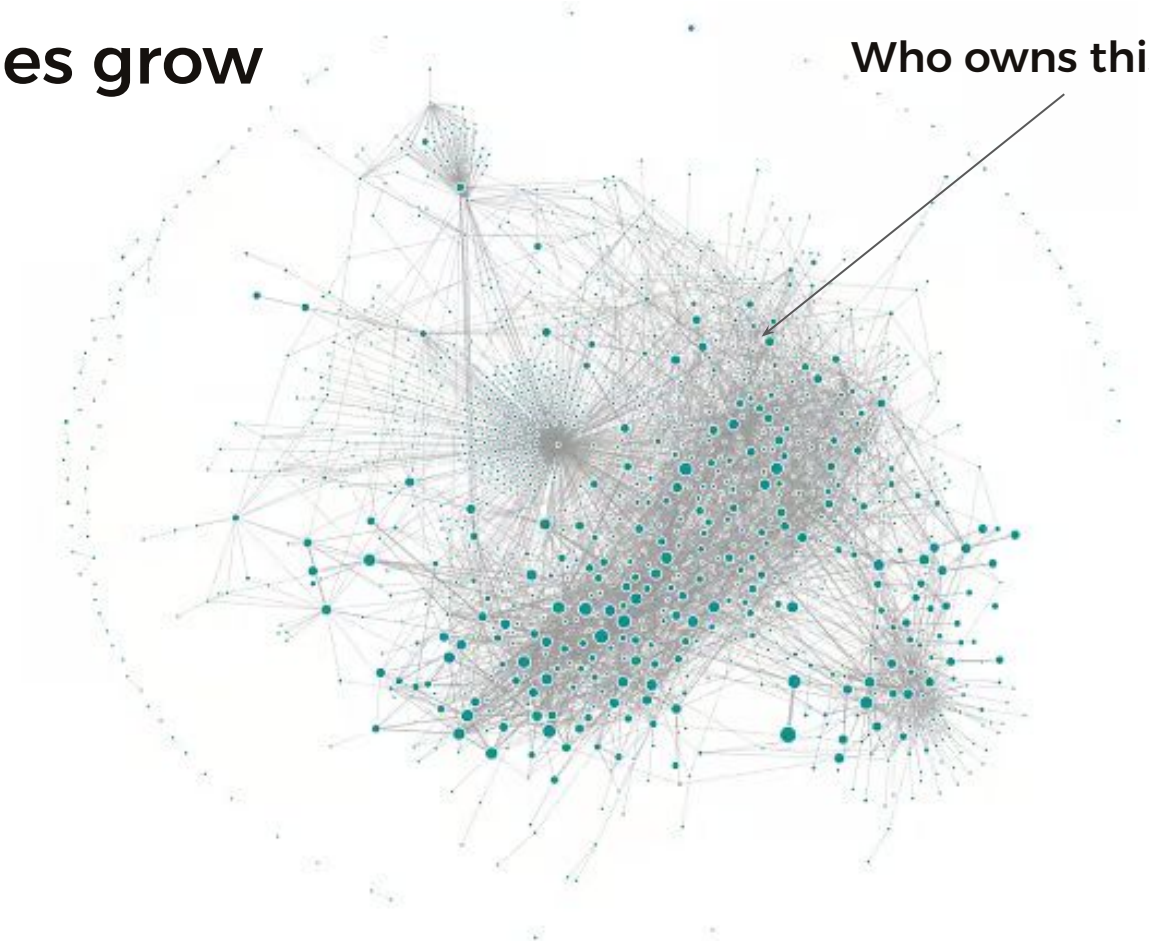
[1] Requests for reprints should be sent to Mica R. Endsley, Department of Industrial Engineering, Texas Tech University, Lubbock, TX 79409.

[Human Factors Study (1995)]

When you’re a smaller company its easier to coordinate

Everyone uses open floor plans these days anyway.

But companies grow



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 analysi

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

https://sre.google/sre-book/managing-incidents/

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

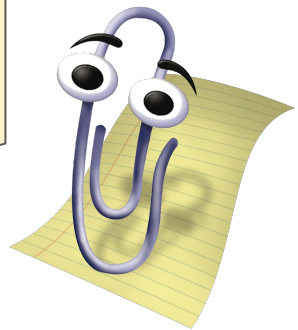
CPU, Memory, Networks – k8s nodes, database services, etc
Basic services – what does everything else depend on?
CI/CD – when was the last deployment? What versions are out there?

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

API

```
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()
}
```

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

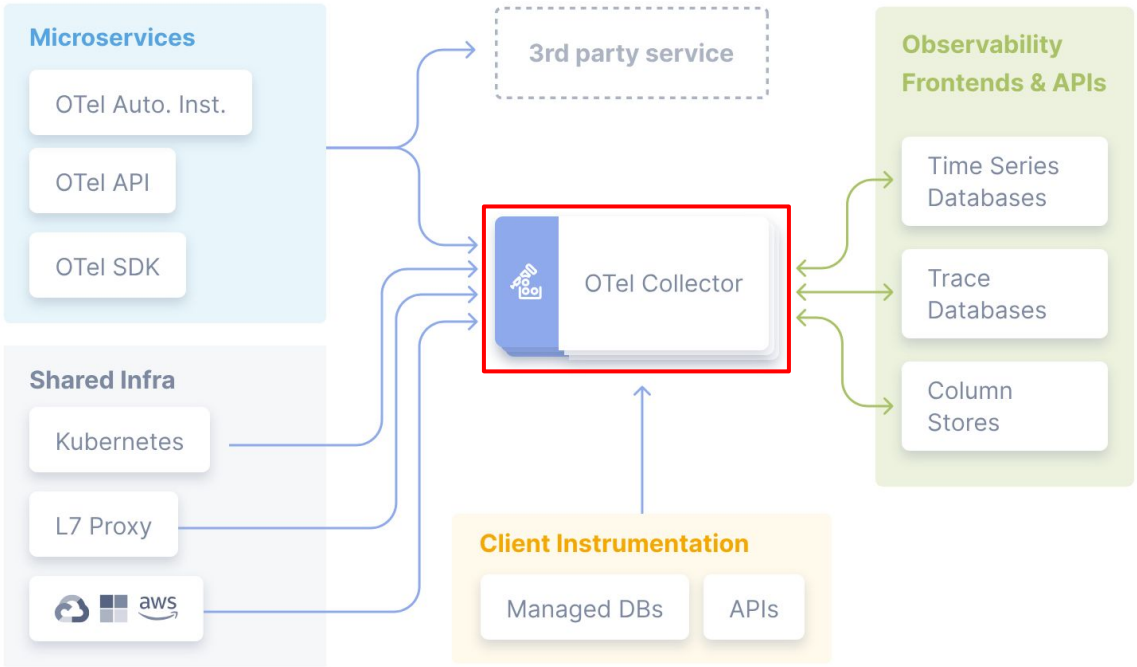
Instrumentation points

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

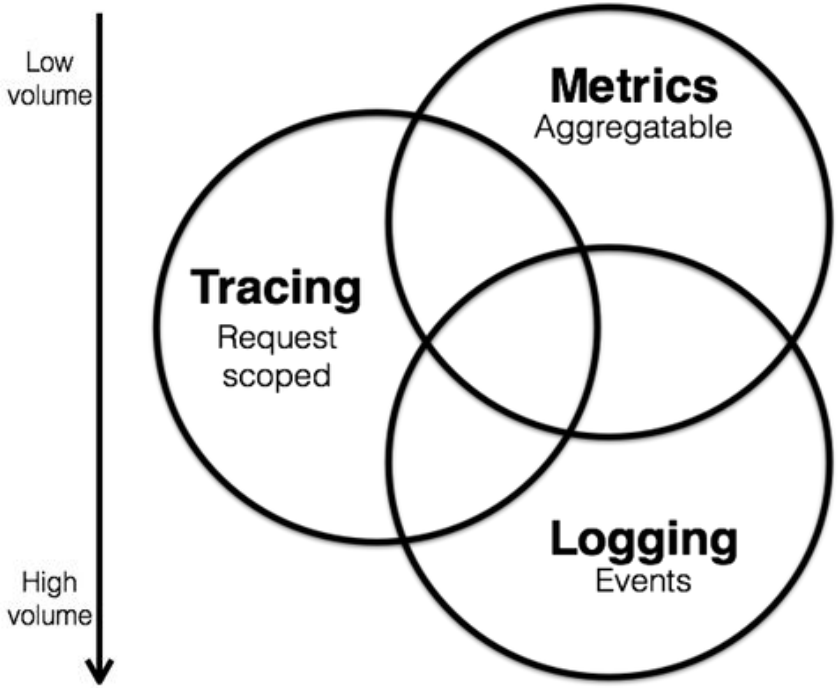
Ex: Collector (data pipeline)

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.

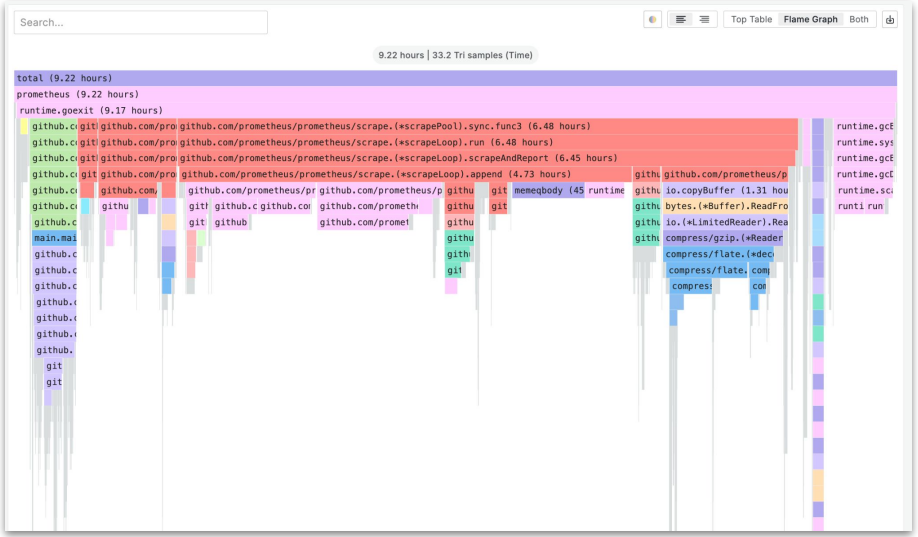
Signals



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

There can be more signals...

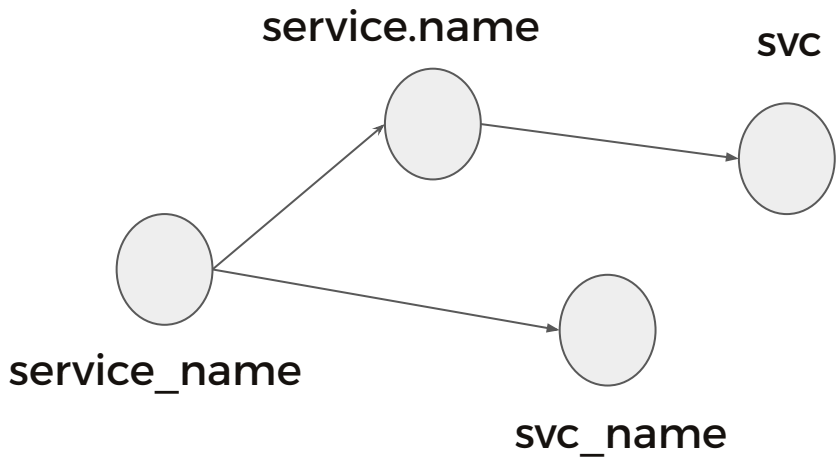
Continuous profiling helps finding and debugging painful performance issues down to the function and line of code.



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
….
```



https://opentelemetry.io/docs/specs/semconv/

Part 3: Query patterns and building the stack.

Choosing storage

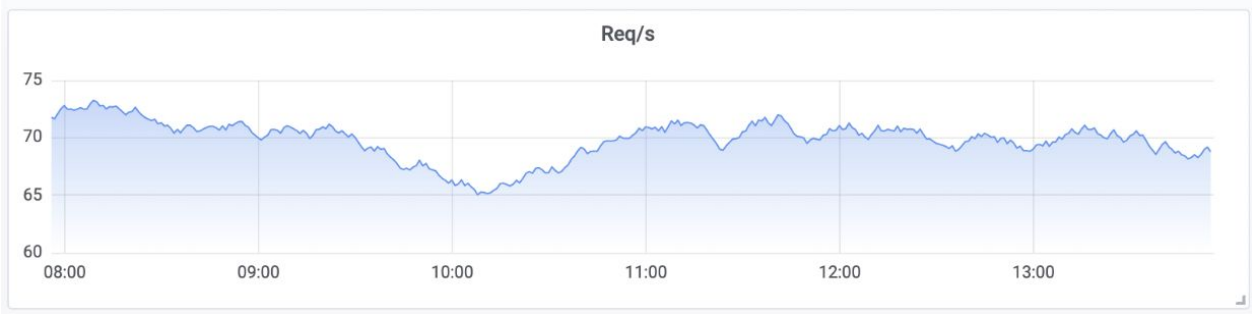
So far we've seen instrumentation and the data pipeline. That's where OTel specifications and implementations end.

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

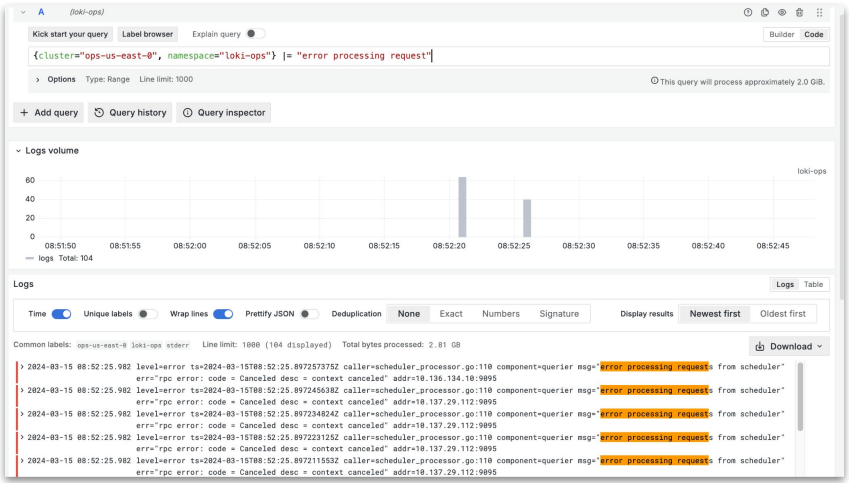
Percentiles

```
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

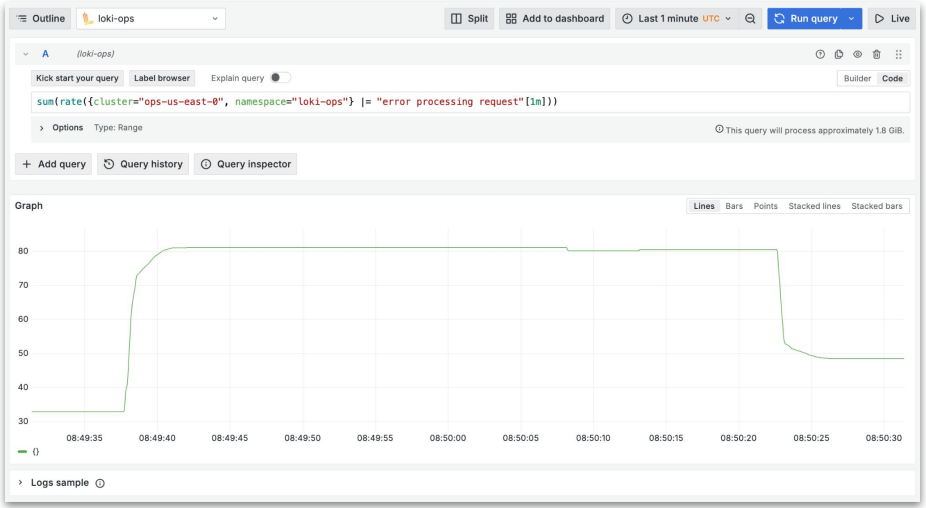
```
{cluster="ops-us-east-0", namespace="loki-ops"} |= "error"
```



Query Patterns - Logs

Rate of increase in keyword over time – helps spot trends

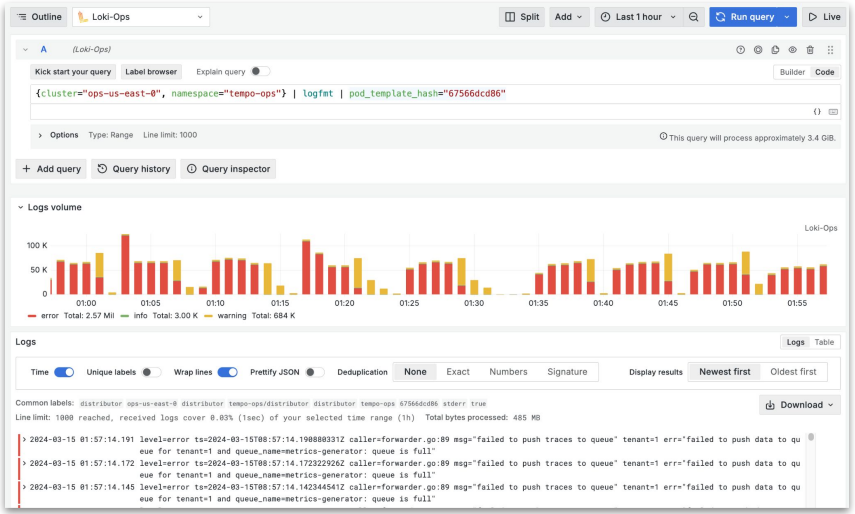
```
count_over_time(
  {cluster="ops-us-east-0", namespace="loki-ops"}
  |= "error processing request"[1m]
 )
```



Query Patterns - Logs

Extensible schema – don’t force early lock in

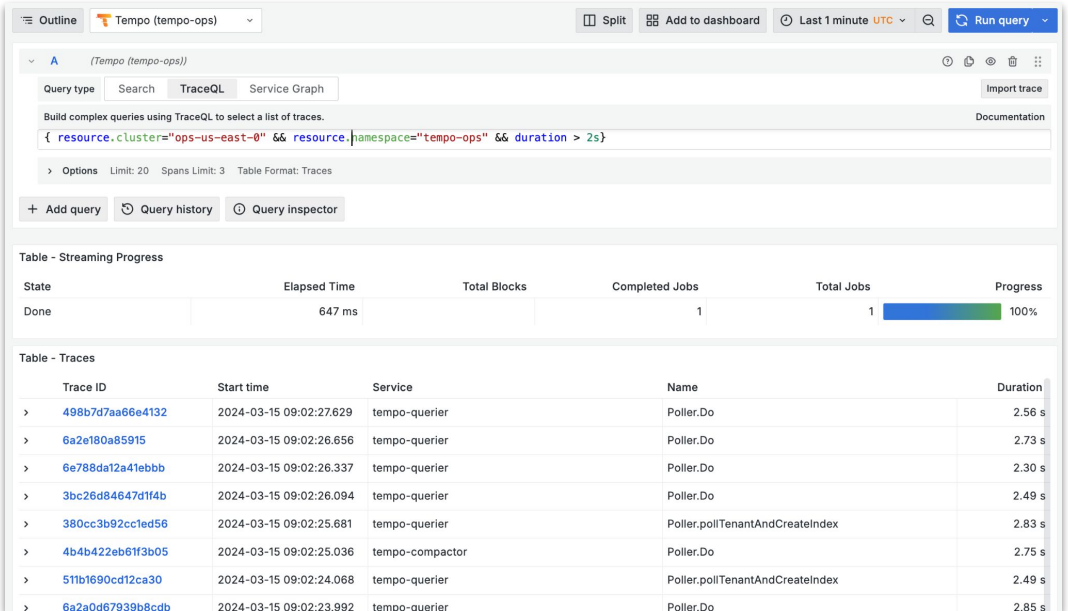
```
{cluster="ops-us-east-0", namespace="tempo-ops"}
  | logfmt
  | client_ip="192.168.0.10"
```



Query Patterns - Traces

Search for specific labels and latency

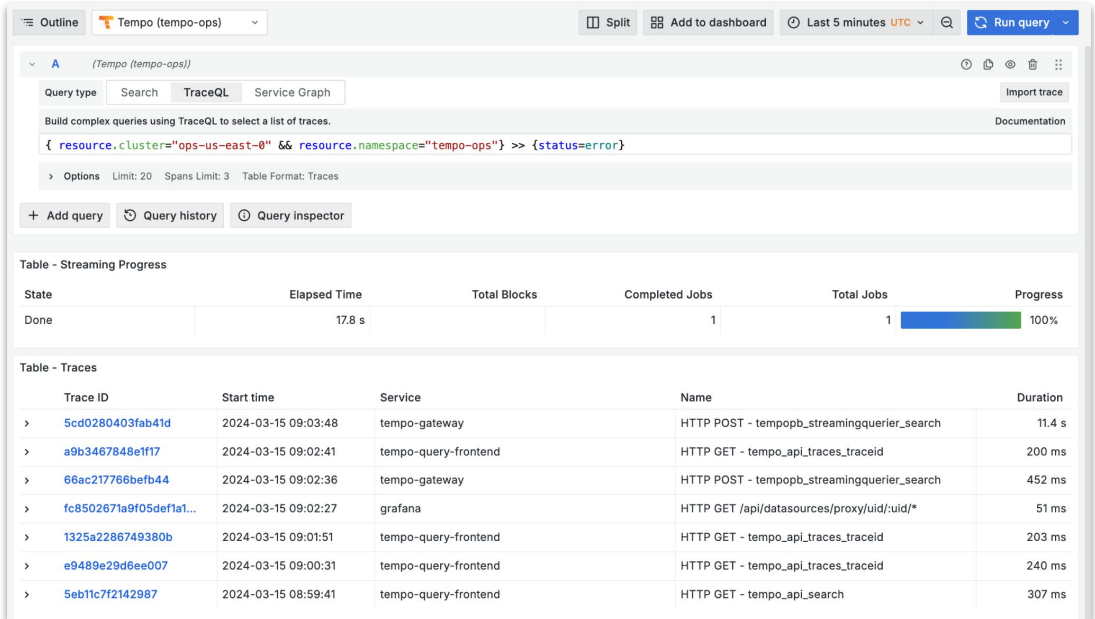
```
{ cluster="ops-us-east-0" && namespace="tempo-ops" && duration > 2s}
```



Query Patterns - Traces

Search based on structure of trace

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



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!

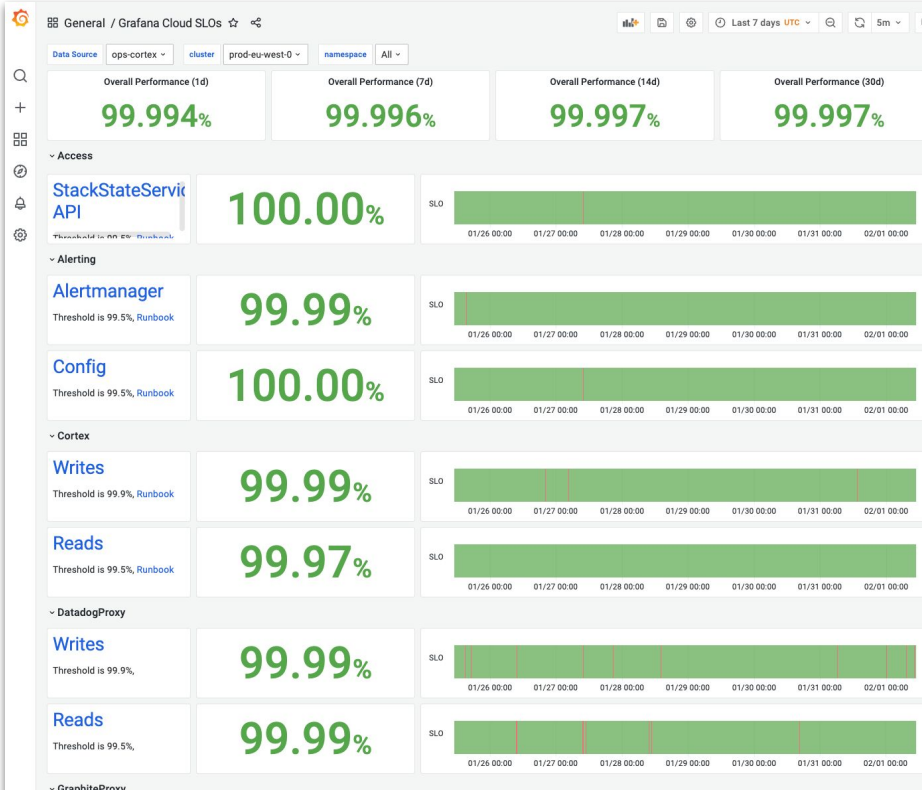
Purpose built tools & automation.

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!

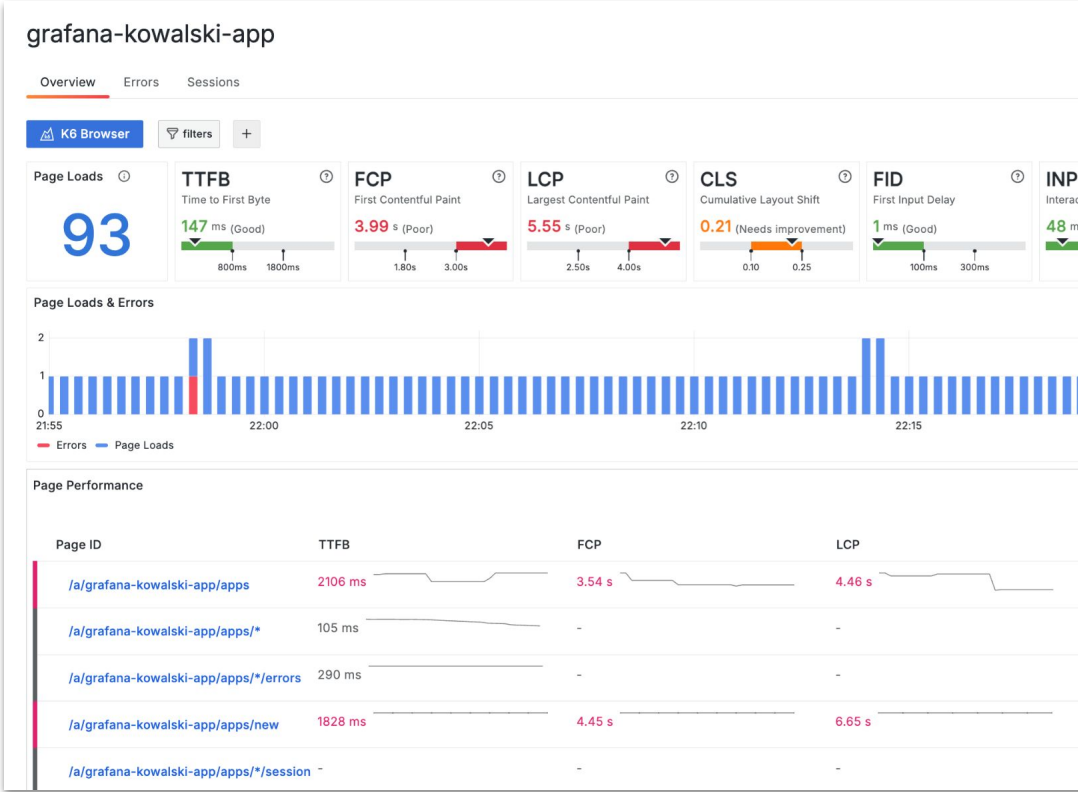


2. App specific debugging workflows

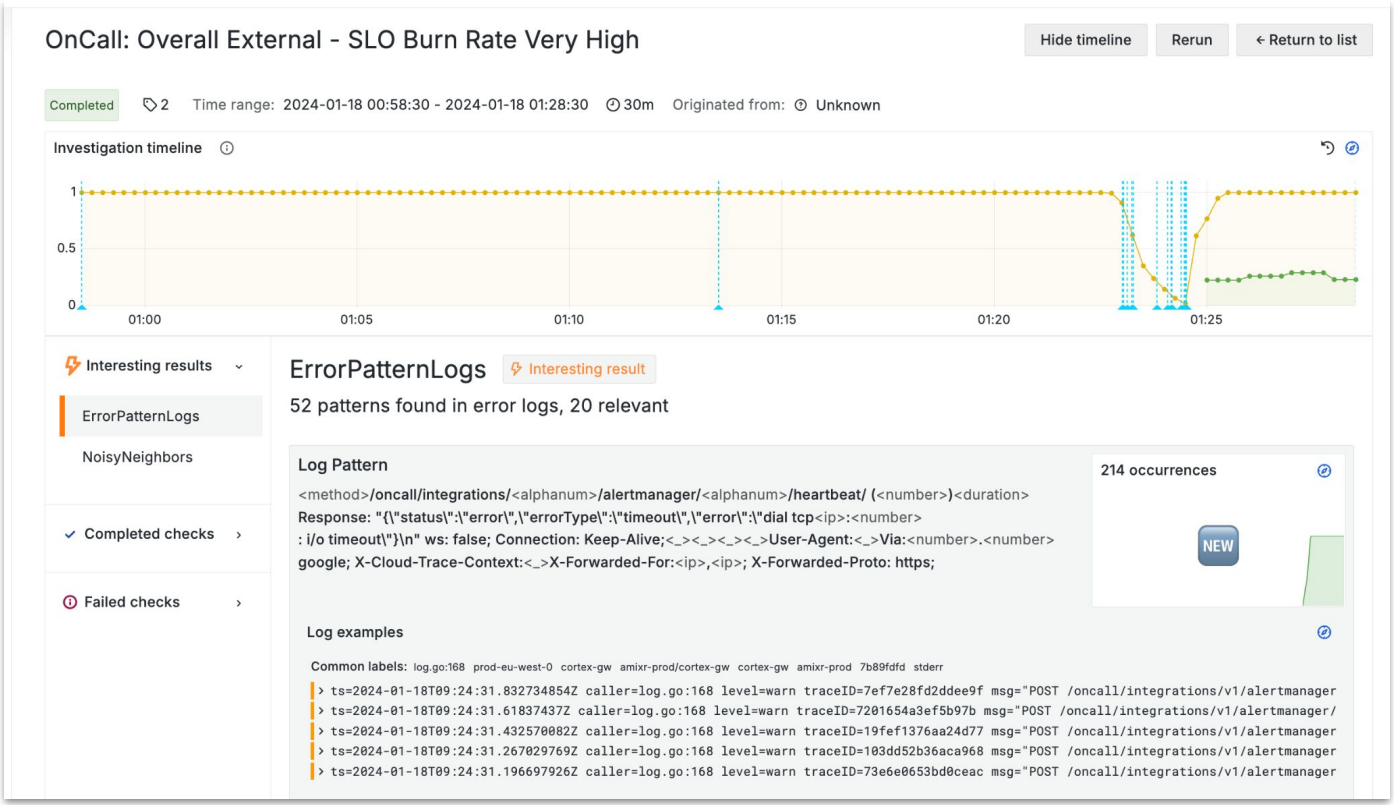
Build debugging workflows with knowledge of what the app does.

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

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

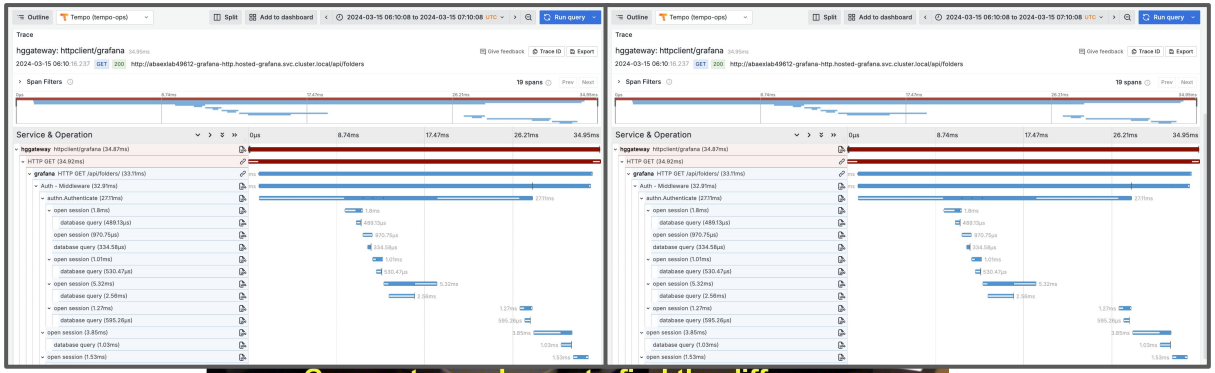


3. Automate change detection (logs)



https://jiemingzhu.github.io/pub/pjhe_icws2017.pdf

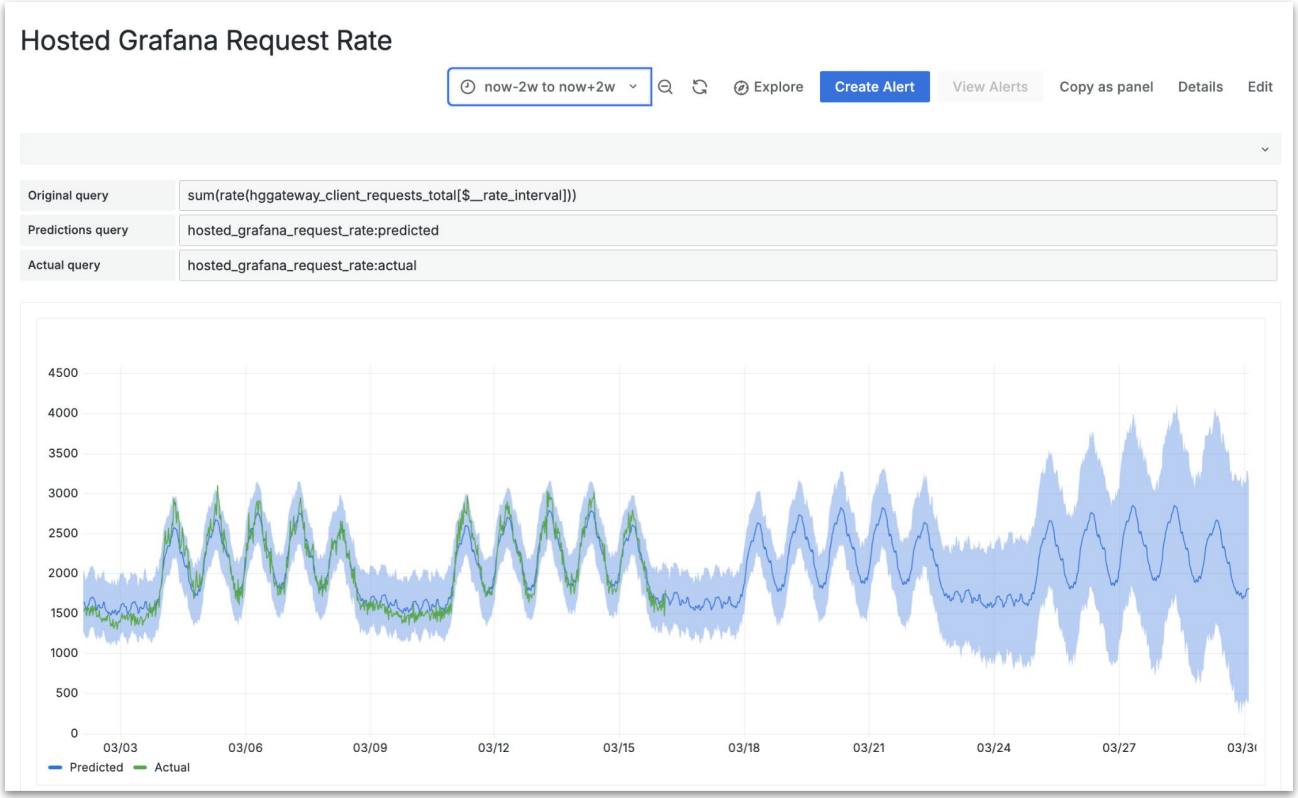
3. Automate change detection (traces ex.)



3. Automate change detection (metrics)

https://github.com/datastax-labs/hunter

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!

Grafana Labs

Thank you! Questions?

@mrannanay


Grafana Labs