
AWS
re:Invent
NOV. 28 – DEC. 2, 2022 | LAS VEGAS, NV

OPN302

Signal correlation: The why and the how

Michael Hausenblas

Product Manager for OpenTelemetry
AWS Open Source Observability Service Team

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Agenda

What problem are we trying to solve?

Observability signals

Correlation use cases

Correlation approaches

An example

Challenges and the road ahead

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

What problem are we trying to solve?

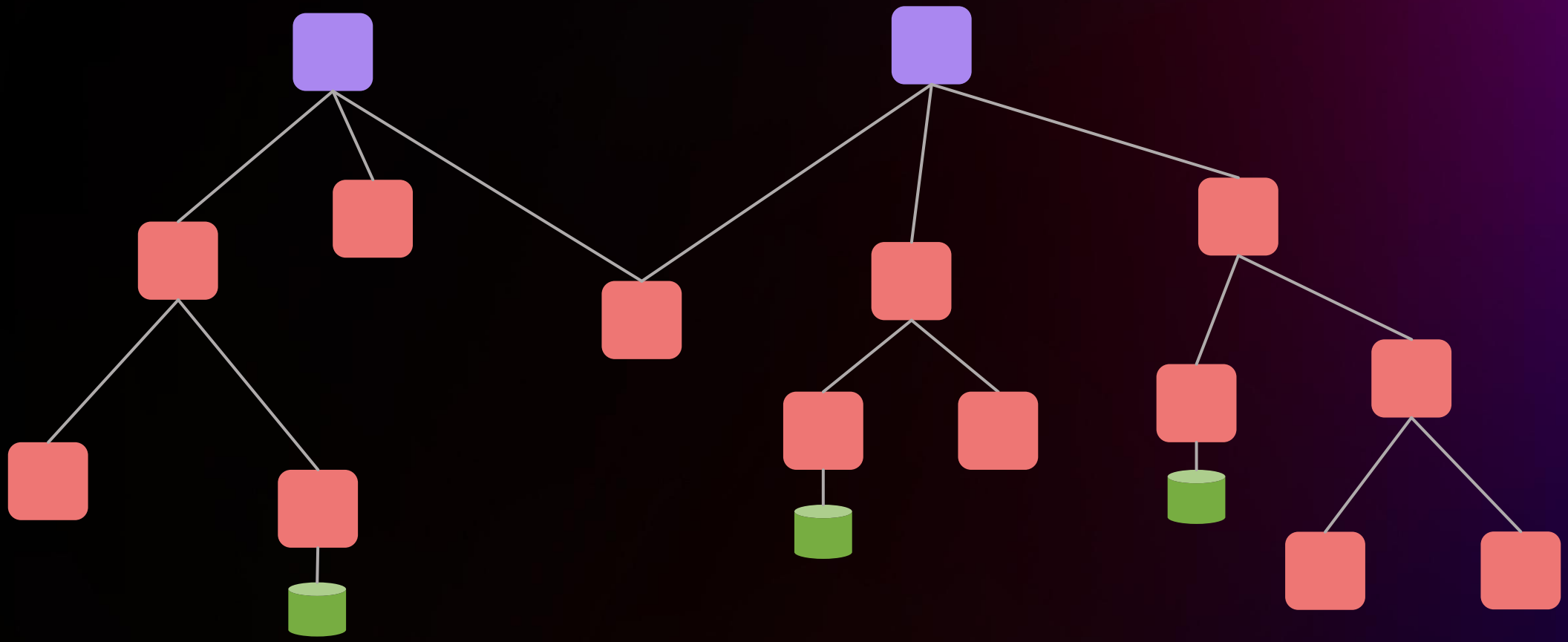
Observability is the capability to continuously generate and discover actionable insights based on signals from the system under observation. In other words, observability allows users to understand a system's state from its external output and take (corrective) action.

https://glossary.cncf.io/observability



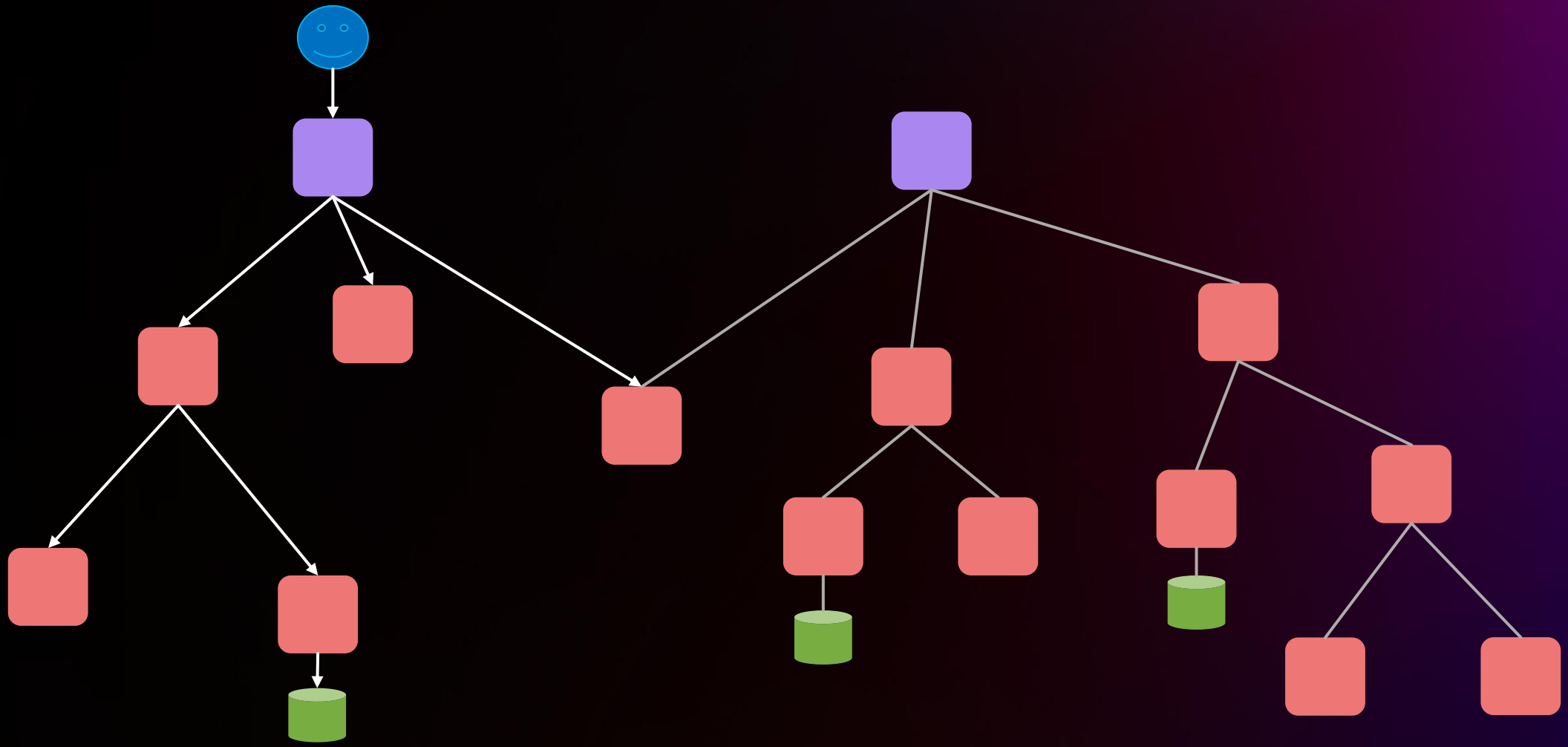
aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Context – Distributed system



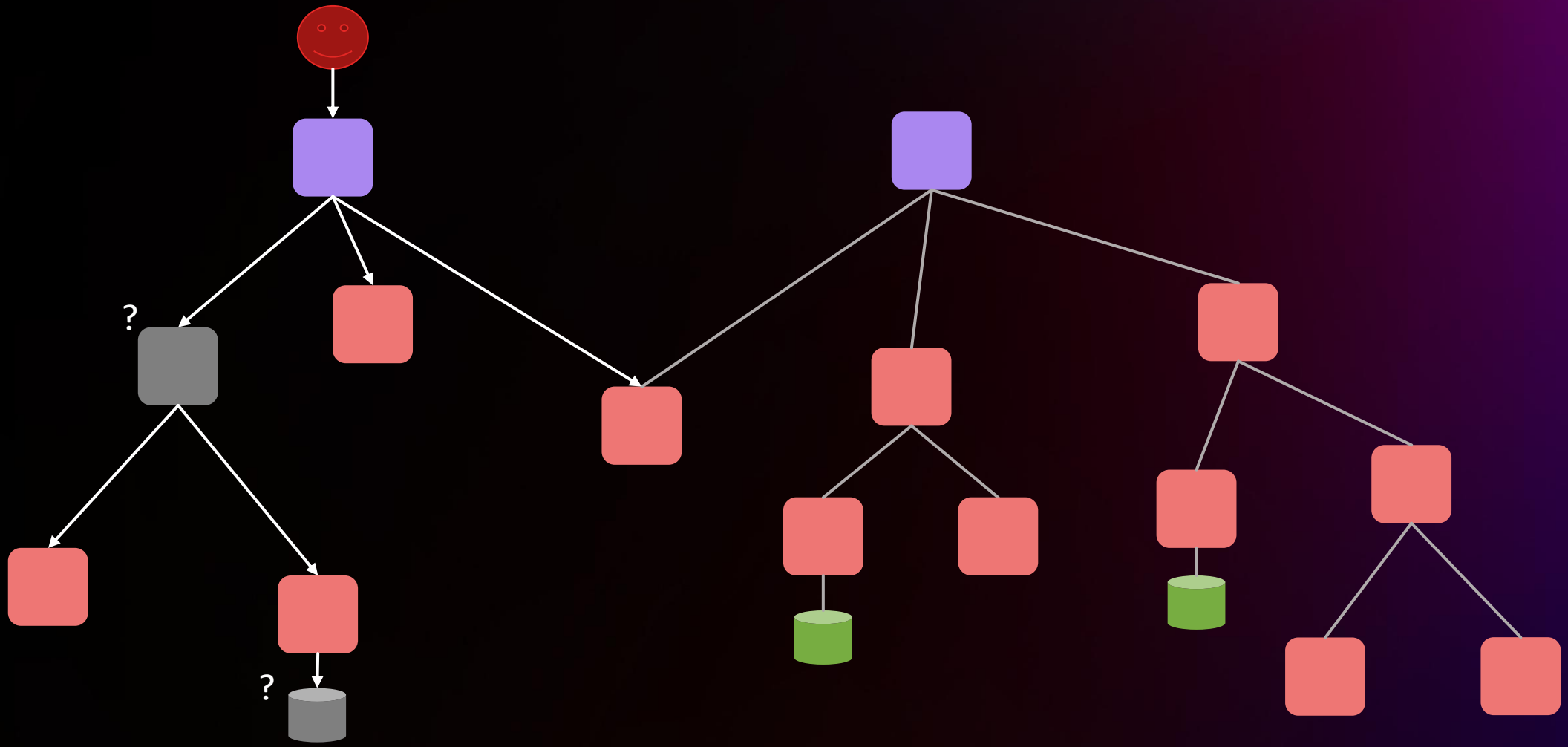
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Context – Distributed system



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Context – Distributed system



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Observability signals

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Fundamental signals

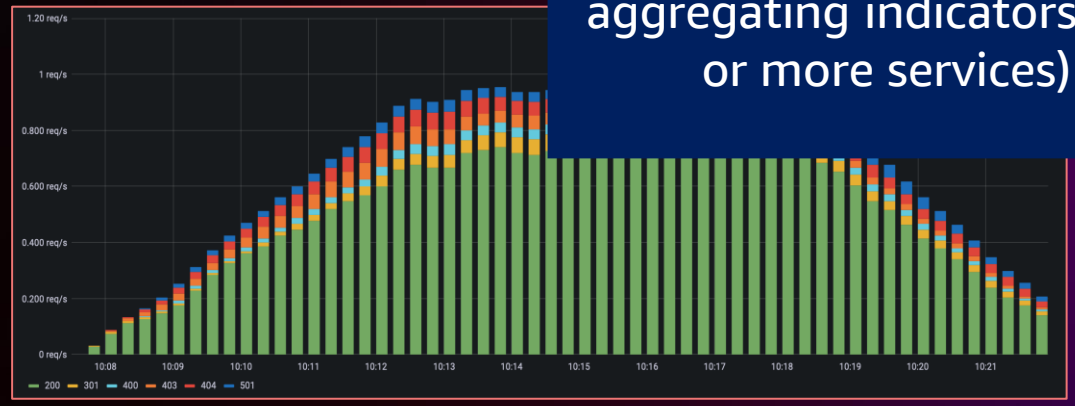
Logs

Timestamped, textual signals capturing in-service events



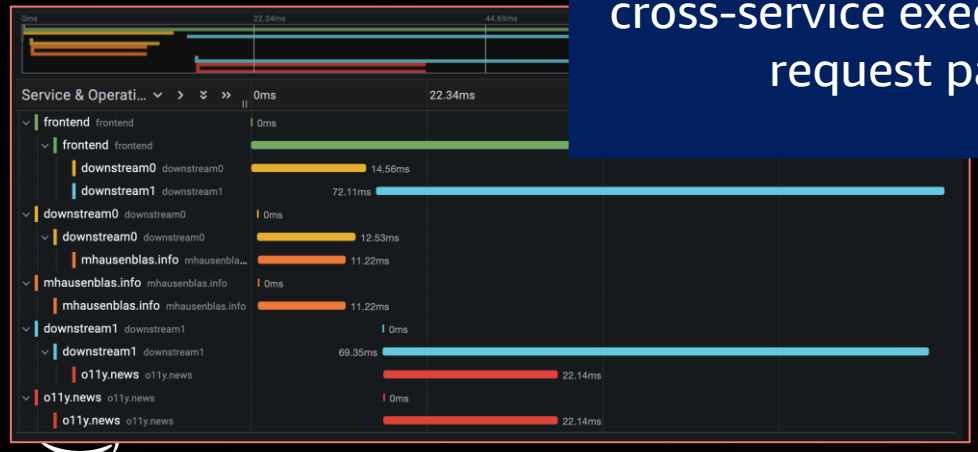
Metrics

Numerical signals, aggregating indicators (one or more services)



Traces

Distributed signals capturing cross-service execution of a request path

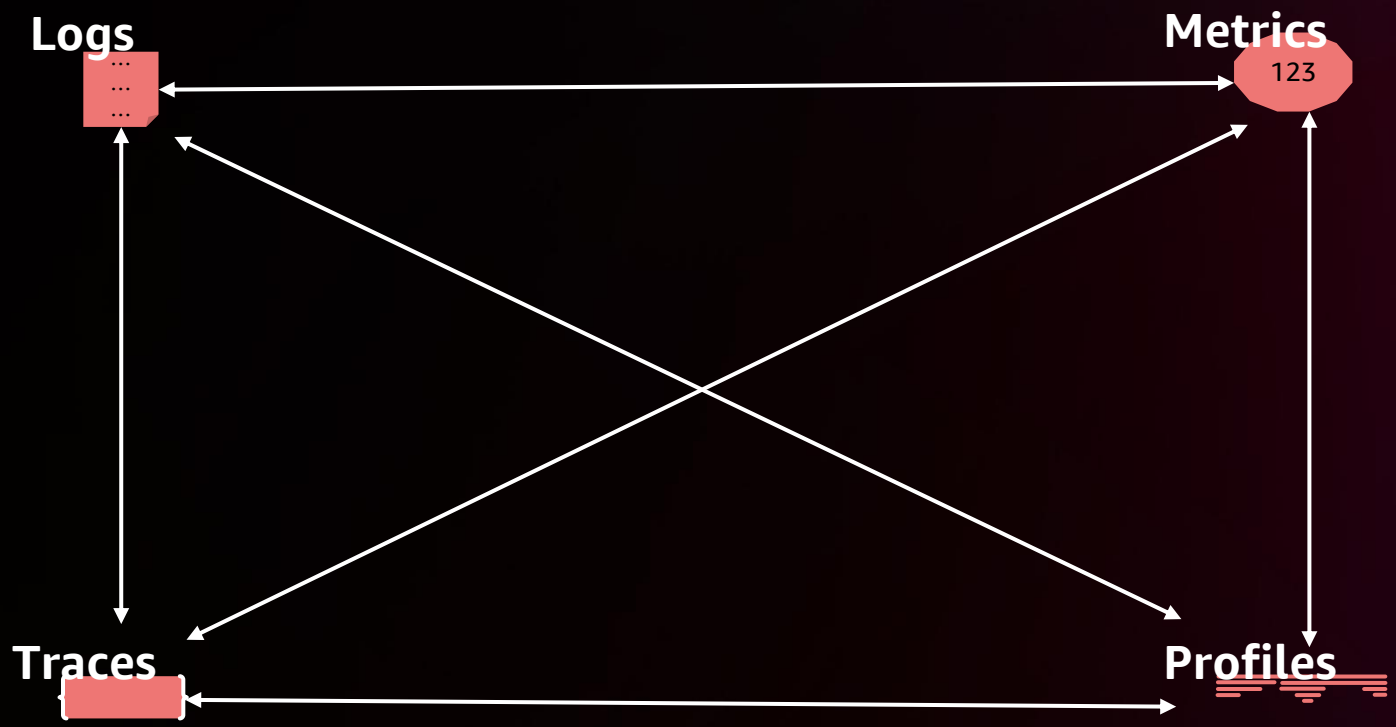


Profiles

Capturing in-service resource usage (CPU, memory, etc.)



Transitions between signals



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Use cases

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Correlation approaches

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Metrics-traces correlation example

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Instrumentation

```
invokeCounterEx = promauto.NewCounter(
        prometheus.CounterOpts{
            Name: "ho11y_ex_total",
            Help: "Total ho11y invokes with exemplar.",
        },
 )
...

invokeCounterEx.(prometheus.ExemplarAdder).AddWithExemplar(1, prometheus.Labels{"traceID": traceid})

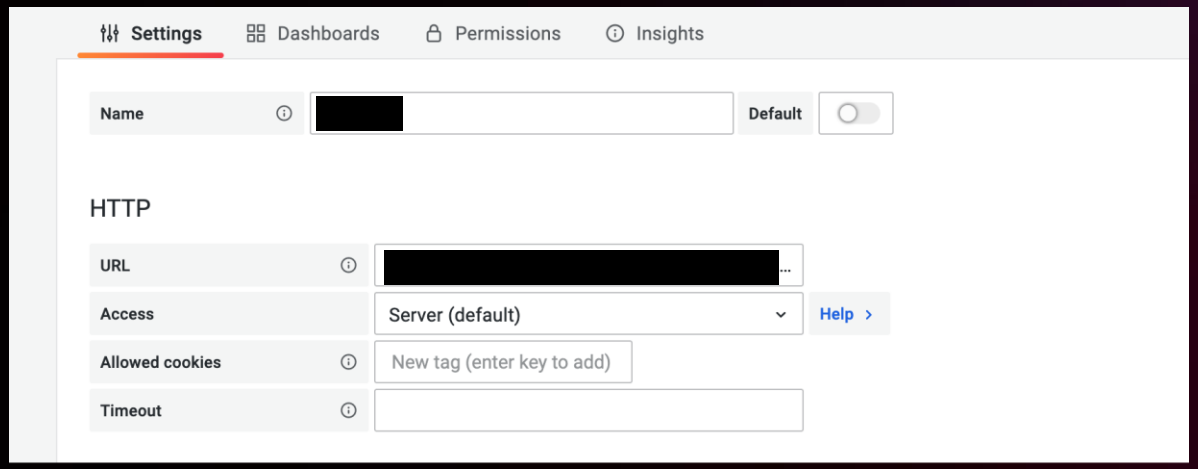
...

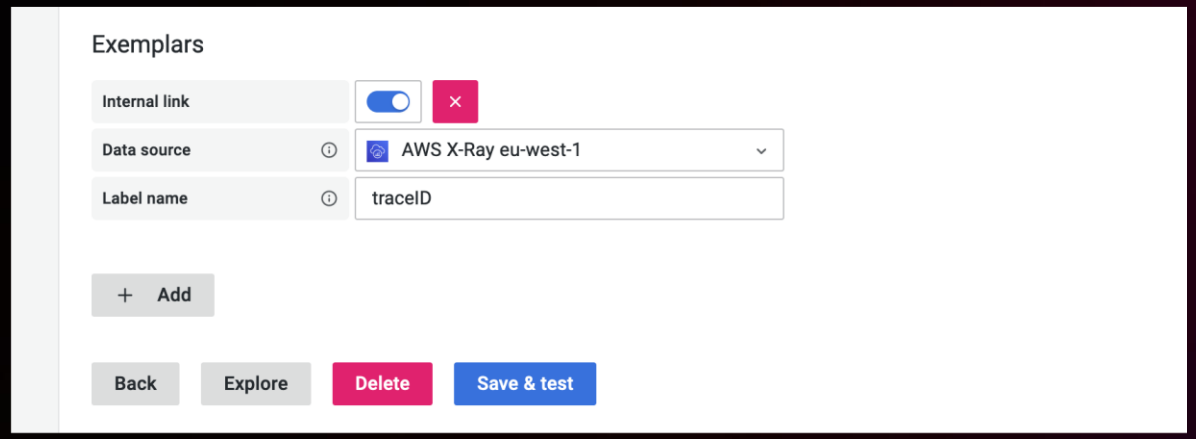
r.Handle("/metrics", promhttp.HandlerFor(
        prometheus.DefaultGatherer,
        promhttp.HandlerOpts{
            EnableOpenMetrics: true,
        },
))
```

```
# HELP ho11y_ex Total ho11y invokes with exemplar.
# TYPE ho11y_ex counter
ho11y_ex_total 382.0 # {traceID="1-6244311b-204f0cb68869a4dc12e0f62f"} 1.0 1.6486361878349538e+09
# HELP ho11y_randhist A histogram of normally distributed random numbers
```

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Grafana





© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Grafana



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Thank you!

Michael Hausenblas

hausenbl@amazon.com
@mhausenblas on Twitter



Please complete the session survey in the **mobile app**

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.