CRIBLCON 24

POWERED BY Cribl

# Correlating Traces with Logs for Enhanced Observability

**BHOOPESH**
Senior Software Engineer, Observability, Autodesk

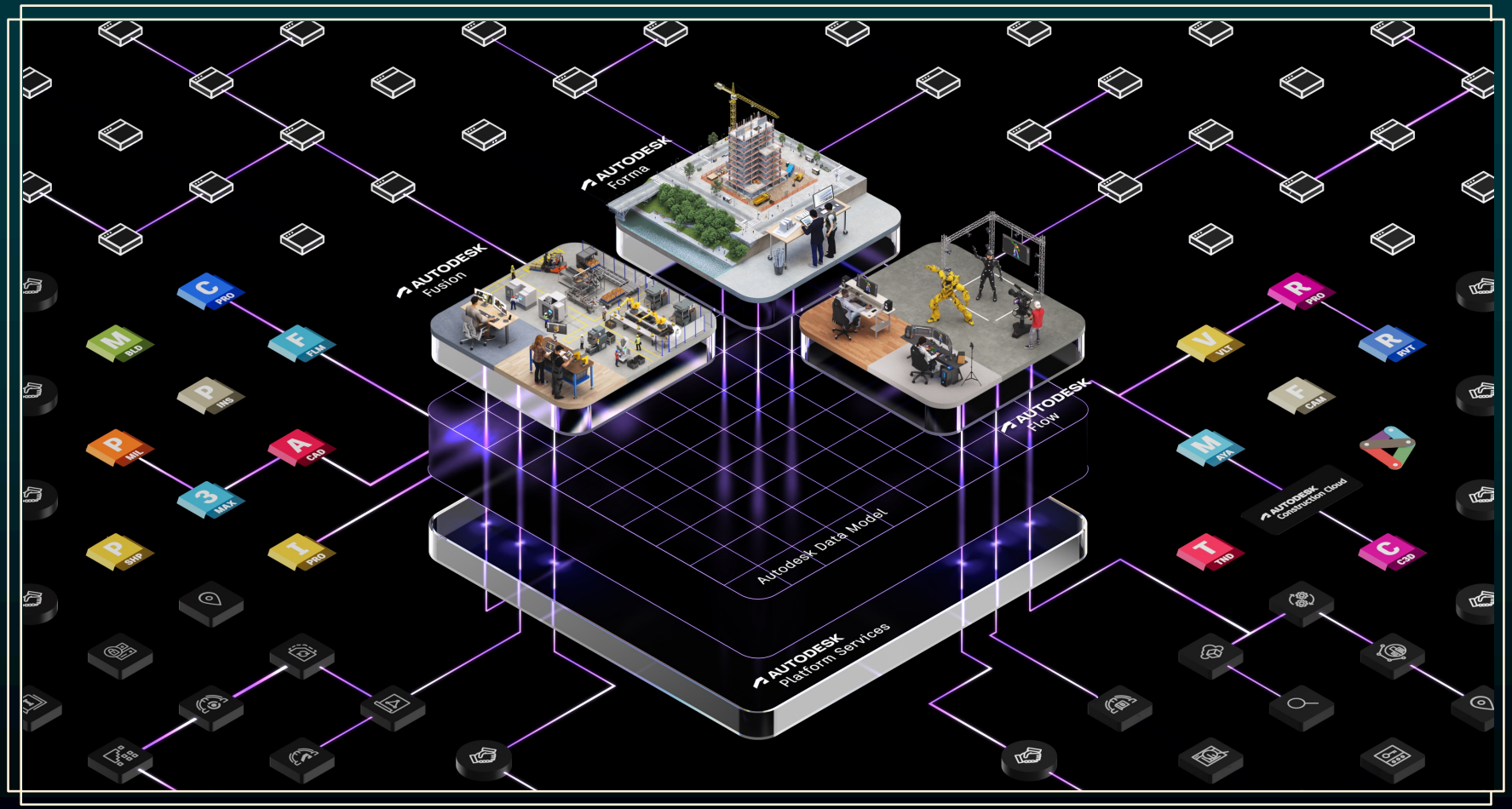
# BHOOPESH

Senior Software Engineer,
Observability, Autodesk

# What does Autodesk Do?

Autodesk is changing how the world is designed and made. Our technology spans architecture, engineering, construction, product design, manufacturing, media, and entertainment, empowering innovators everywhere to solve challenges big and small. From greener buildings to smarter products to more mesmerizing blockbusters, Autodesk software helps our customers to design and make a better world for all.

AUTODESK

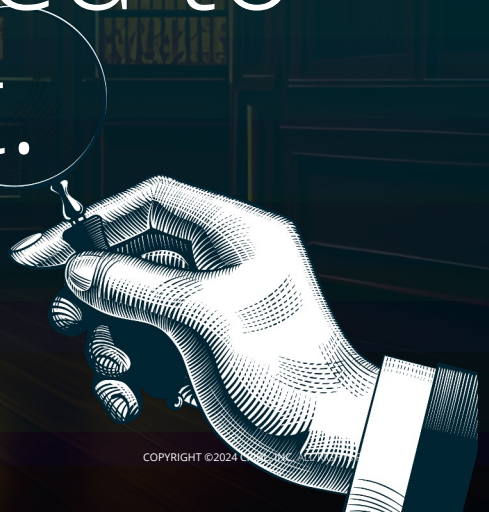


# Agenda

## Here's what to expect:

1. **INTRODUCTION**
   Setting the stage for enhanced observability in distributed apps
2. **THE CHALLENGE**
   Identifying issues with isolated observability data and siloed tools
3. **INTEGRATION**
   Learn the benefits of combining OTel traces with Splunk logs
4. **OUTCOME & HOW**
   Learn more about the impact and how to implement solution
5. **USE CASE**
   See how to enhance observability in an e-commerce application
6. **BEST PRACTICES**
   Sharing lessons learned and best practices for effective observability
7. **DASHBOARDS**
   Building a one-stop shop  for enhanced system insights

# Collecting data alone isn’t enough. You need to make sense of it.



# The Challenge

# The Challenge

1. Unifying diverse telemetry data requires tracking unique request identifiers across a distributed system
2. Switch between different tools and lose the valuable context pertinent to fixing performance issues
3. Manual mapping of service instrumentation to log identifiers is complex and time-consuming
4. Maintaining and expanding this system becomes increasingly difficult as new services are built and new languages, frameworks, and libraries are adopted, often resulting in visibility of only a subset of logs for any particular request

# The Challenge

Siloed observability data slows down effective troubleshooting and performance optimization

**Data silos**

**Incomplete insights**

**Extended downtime**

**Scalability issues**

# Impact of Challenges

## Reinforcing the need for integrated observability

**Increased mean time to respond (MTTR)**

Higher resolution times due to fragmented data

**Higher operational costs**

More resources needed for manual correlation

**Reduced system reliability**

Impact on uptime and user experience

“We can’t have all these things working on a separate basis. This problem has to be addressable.”

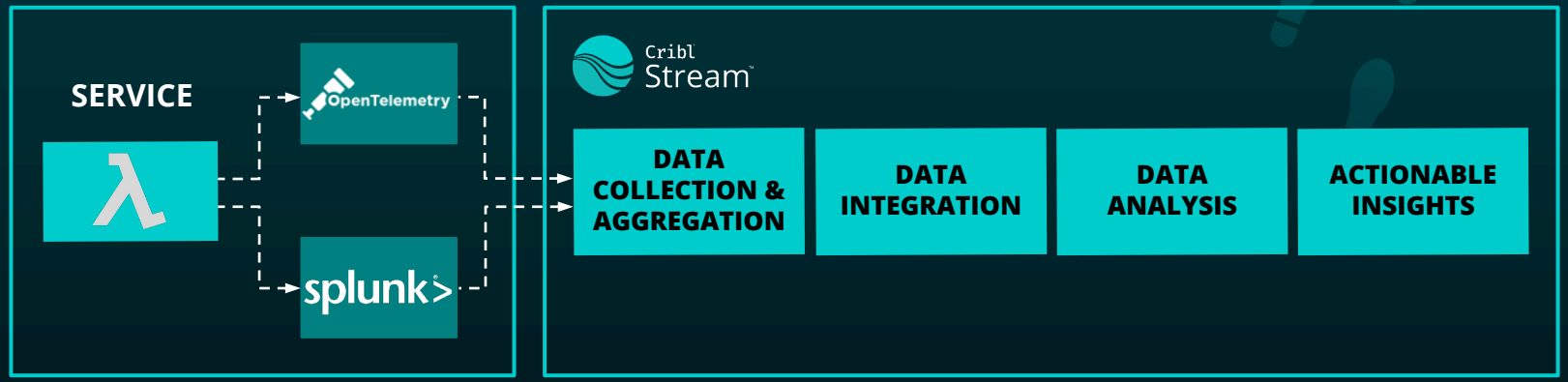
# The Outcome and how

# A unified view of system behavior

Bridge silos for a seamless flow of info

**APPLICATION TEAM**

**OBSERVABILTY TEAM**




# Integrating Traces and Logs

### Quick root cause identification

Faster root cause analysis by correlating events across services

### Service dependency mapping

Visualize how different services interact

### Metrics tracking

Monitor and analyze latency and Endpoint performance across various parts of the system

# Solution

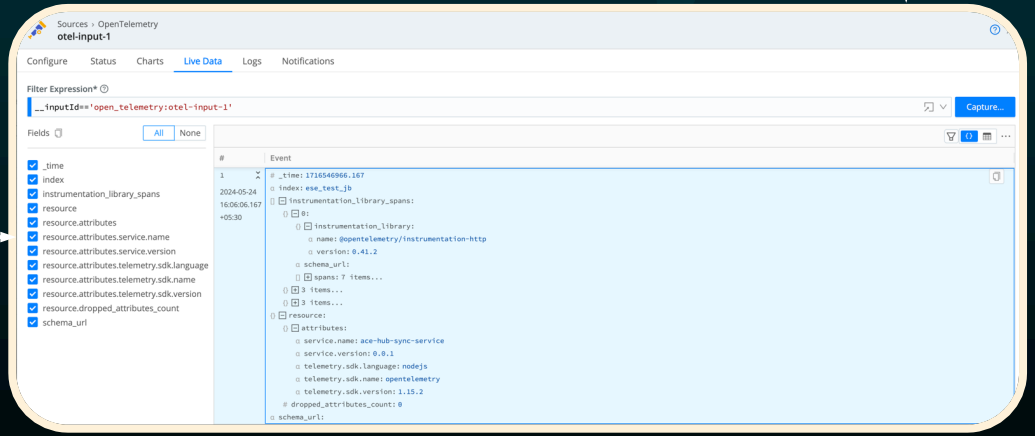
## Collect

```yaml
exporters:
  debug:
    verbosity: detailed
  otlp/cribl:
    #endpoint: ${CRIBL_GRPC_ENDPOINT}
    endpoint:
    tls:
      insecure_skip_verify: true
    keepalive:
      time: 10s
    retry_on_failure:
      max_elapsed_time: 10s
    sending_queue:
      queue_size: 10000
    compression: gzip
```

**Collect Trace data from OTel Collectors**

**Enable OTel Source**

OpenTelemetry




# Solution

## Reduce

```
#   Event
1   # _time: 1715009299.026
    α index: ese_test_jb
    [] instrumentation_library_spans:
      {} 0:
        {} instrumentation_library:
          α name: Mojo
          α version: unknown
        α schema_url:
        [] spans: 100 items...
    {} resource:
      {} attributes:
        α deployment.environment: prd
        α process.command: /home/lem/bundle/jruby/3.1.0/bin/puma
        # process.pid: 1
        α process.runtime.description: jruby 9.4.6.0 (3.1.4) 2024-02-20 576fab2c51 OpenJDK 64-Bit Server VM 25.402-b08 on 1.8.0_402-b08 +jit
                                       [x86_64-linux]
        α process.runtime.name: jruby
        α process.runtime.version: 3.1.4
        α service.name: lem
        α service.version: lem:1316
        α telemetry.sdk.language: ruby
        α telemetry.sdk.name: opentelemetry
        α telemetry.sdk.version: 1.2.1
      # dropped_attributes_count: 0
    α schema_url:
```

2024-05-06
20:58:19.026
+05:30

```
[] spans:
  {} 0:
    {} attributes:
      α component: http
      α deployment.environment: prd
      α http.client_ip: 10.40.226.123
      α http.method: GET
      α http.route: /service/entitlements/v2/users/7PUCM4N95JH3
      # http.status_code: 200
      α http.url: /service/entitlements/v2/users/7PUCM4N95JH3?includeExpired=false&includeParentAsset=true&source=legacy
      α router.params.includeExpired: false
      α router.params.includeParentAsset: true
      α router.params.reference_id: 7PUCM4N95JH3
      α router.params.source: legacy
    # dropped_attributes_count: 0
    # dropped_events_count: 0
    # dropped_links_count: 0
    # end_time_unix_nano: 1715009296381282300
    [] events:
    # kind: 2
    [] links: 0 items...
    α name: /service/entitlements/v2/users/:reference_id
    α parent_span_id:
    α span_id: a0690d522be14af4
    # start_time_unix_nano: 1715009296378018000
    {} status: 2 items...
    α trace_id: afc2a962c77043ec432b1a6a450ad1ca
    α trace_state:
  {} 15 items...
  {} 15 items...
  {} 15 items...
  {} 15 items...
  {} 15 items...
  {} 15 items...
  {} 15 items...
  {} 15 items...
```

# Solution

## Shape

1

2024-05-24
19:53:00.590
+05:30


```
_raw:
  process.pid: 1
  service_name: lem
  spans:
    attributes:
      component: http
      http.client_ip: 10.40.226.123
      http.method: GET
      http.route: /service/entitlements/v2/users/7PUCM4N95JH3
      http.status_code: 200
      http.url: /service/entitlements/v2/users/7PUCM4N95JH3?includeExpired=false&includeParentAsset=true&source=legacy
      router.params.includeExpired: false
      router.params.includeParentAsset: true
      router.params.reference_id: 7PUCM4N95JH3
      router.params.source: legacy
    kind: 2
    name: /service/entitlements/v2/users/:reference_id
    parent_span_id:
    span_id: a0690d522be14af4
    status:
      code: 0
      message:
    trace_id: afc2a962c77043ec432b1a6a450ad1ca
    trace_state:
    ttms: 3
_time: 1716560580.59
cribl_pipe: pipeline-splunk-ttl_master-v1
host: ip-10-110-220-24.us-west-2.compute.internal
index: ese_test_jb
instrumentation_library_spans: 1 item...
process.pid: 1
resource: 2 items...
schema_url:
source: open_telemetry:otel-input-1
sourcetype: ttl
```


# How

## Shed the Dead Weight

Unroll true

Filter
true

Description
Enter a description

Final No

Source Field Expression*
instrumentation_library_spans

Destination Field*
instrumentation_library_spans

Unroll true

Filter
true

Description
Enter a description

Final No

Source Field Expression*
instrumentation_library_spans.spans

Destination Field*
instrumentation_library_spans.spans

| | |
|---|---|
| Parser | true |
| Parser | true |
| Eval | true |
| Auto Timestamp | true |
| Parser | true |
| Eval | true |
| *add http fields below* | |
| Parser | true |
| *spans drop* | |
| Eval | true |
| Rename | true |
| Serialize | true |
| *final parsed fields removal* | |
| Eval | true |
| *FIN* | |

# Injection of trace metadata

## Trace Context Propagation

```python
from opentelemetry import trace
        You, 1 second ago • Uncommitted changes
def _log(self, level, msg, *a, **kw):
    kw["extra"] = self._data(kw.get("extra"))
    ctx = trace.get_current_span().get_span_context()
    kw["extra"]["otelTraceId"]=ctx.trace_id
    kw["extra"]["otelSpanId"]=ctx.span_id
    self._logger.log(level, msg, *a, **kw)
```

```python
from opentelemetry import trace

def process(self, msg, kwargs):
    kwargs["extra"] = {"data": {k: v for k, v in self._gdata(kwargs)}}
    if self.extra:
        kwargs["extra"]["data"].update(self.extra)
    ctx = trace.get_current_span().get_span_context()
    kwargs["extra"]["otelTraceId"] = ctx.trace_id
    kwargs["extra"]["otelSpanId"] = ctx.span_id        You, 3 weeks ago •

    return msg, kwargs
```

```python
class ContextFilter(logging.Filter):
    def filter(self, record):
        record.id = str(uuid4())
        record.utctime = _timestamp(DEFAULT_DATE_FORMAT)
        record.source = "urn:adsk.sfdc:moniker:SFDC-CPQ"
        record.message = record.getMessage()
        ctx = trace.get_current_span().get_span_context()
        record.otelTraceId = ctx.trace_id
        record.otelSpanId = ctx.span_id
        if hasattr(record, "data"):
            record.datacontenttype = "application/json"
        return 1
```

# Correlation begins here

## Route

- Route the traces to corresponding Index/source/sourcetype using eval
- Enrich the Log data with trace metadata i.e. traceIds and spanIds and environment metadata i.e. service.name and service.environment
- This will allow you to connect the dots between a specific operation (traces) and what was happening in your system at that time (logs)
  - To reconstruct the code path taken by reading a trace
  - To derive request or error ratios from any single point in the code path

# Demo

# Real-world use case

## E-commerce application

- Initial challenges with multiple services and data sources
- Integration of OTel traces with Splunk logs using Cribl Stream

**Results:**

- Improved issue resolution
- Enhanced performance monitoring
- Increased system reliability

# Lessons learned / Best practices

# Best practices

## How to achieve integrated observability with Cribl

Leverage Cribl Stream for routing

Use reduction techniques to minimize noise

Don’t forget to parse and enrich

# Visualization Best Practices...

...for better data readability and operational decision-making

## Custom dashboard creation

- Create custom views tailored to specific operational needs or teams
- Useful for environments where conditions change rapidly and up-to-date information is crucial

## Real-time monitoring and alerts

- Set up dashboards that reflect live data flows
- Trigger alerts based on predefined thresholds

# Visualization Best Practices…

…for better data readability and operational decision-making

## Correlating data across sources

- Pull data from multiple sources into a single dashboard
- Combine logs, metrics, and traces in one view to quickly pinpoint issues across the infrastructure

## Sharing and collaboration

- Share dashboards across teams to so all stakeholders can access relevant insights
- Trigger alerts based on predefined thresholds

# Thank you!