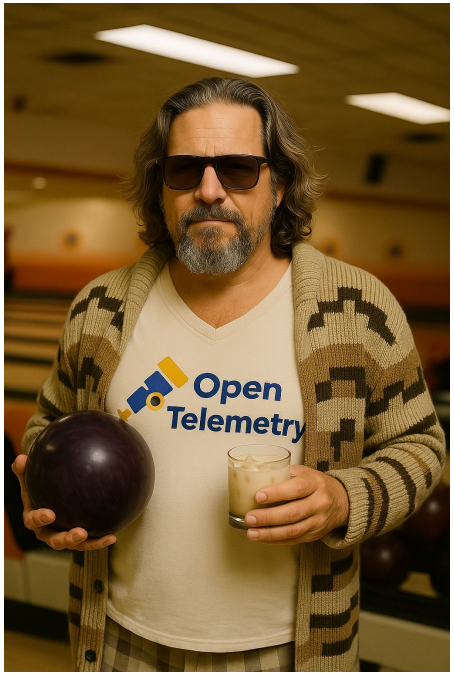
OpenTelemetry

Taming Metric Cardinality

Eric Anderson, CTO, Co-founder (The Dude)
Jon Reeve, CPO, Co-founder (Walter)

Taming Metric Cardinality: The Dude Abides by OTTL

Eric



https://bsky.app/profile/destari.bsky.social

Jon

https://bsky.app/profile/jreeve.bsky.social

We're not heroes. Just here to stop cardinality explosions. That’s our opinion, man.

Cardinality – What Is It, and Why Does It Hate You?

- Low cardinality: 10 time series
- High cardinality: **100,000+** time series

1000 customers, each customer has 100 users, and they have 10 projects:

`/api/{cust_id}/{project_id}/{user_id}/…`

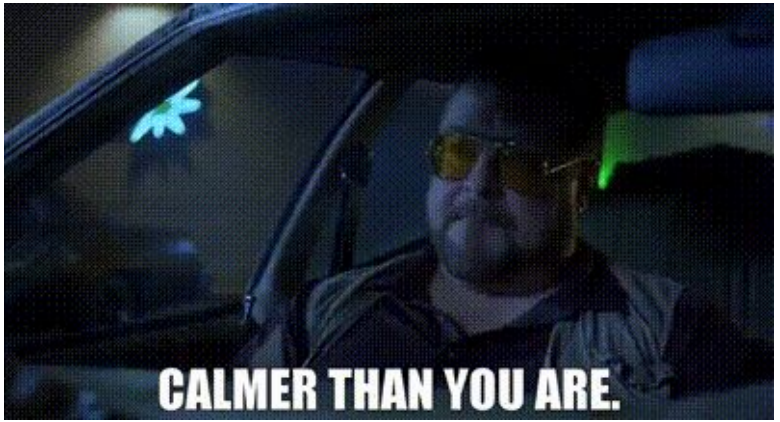
1,000 * 10 * 100 = ***1,000,000*** time series!

More time series = more cost (& noise)

Every new label value is a new time series. That innocent *user_id (session_id, request_id)*? It's a cost bomb…

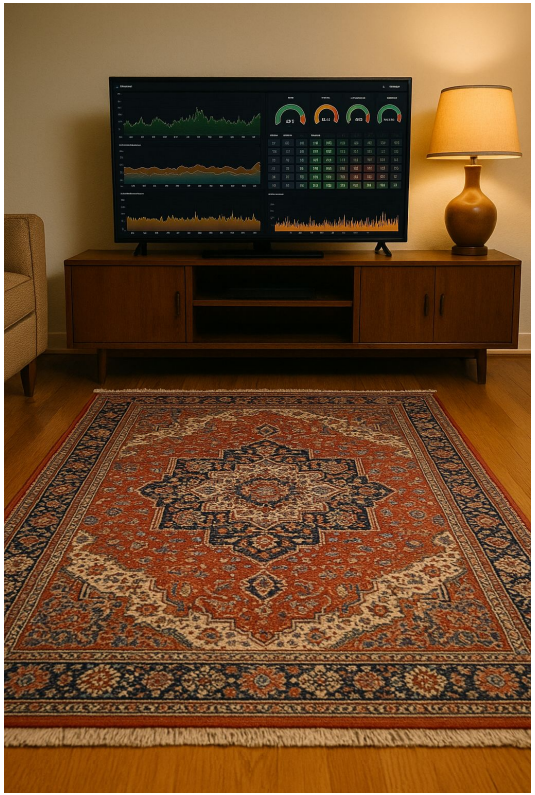
Cardinality – Examples

```
# User id is a unique identifier:
/api/users/123456/profile

# Order number is unique:
/api/orders/abcde-12345/details

# Parameters can be a killer too:
/api/search?q=foo&page=2&sort=asc
```




```
http_request_duration_seconds{user_id="123456"}

container_cpu_usage_seconds_total{container_id=
"docker://2ae8f091b5d..."}

http_requests_total{path="/user/123/profile"}
```

The Problem – A League of Extraordinary Tags



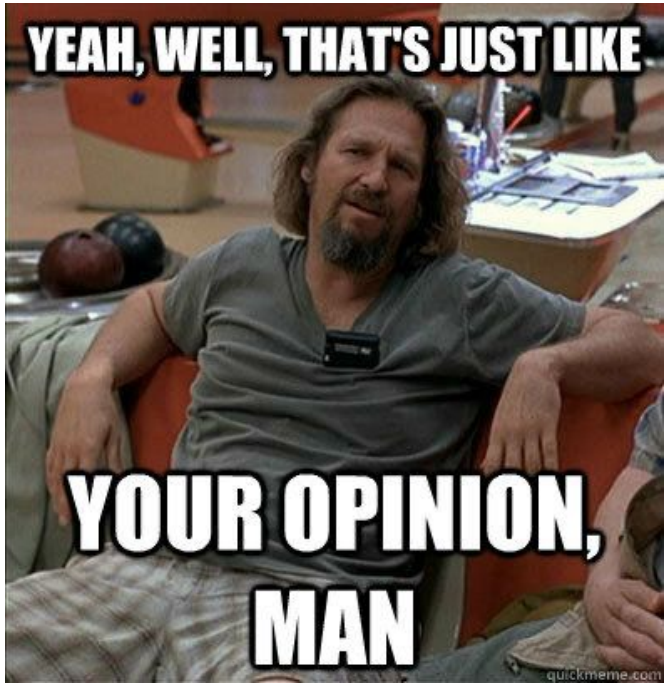
This rug tied the dashboard together…
until 50k time series walked in.

Solution - OpenTelemetry Collector – Our Rug of Sanity

Doesn’t care about your backend. Just wants peace… and fewer series.

What is “OTTL”?

- OTTL or “OpenTelemetry Transformation Language”
- Domain Specific Language in OTel collector to transform telemetry in flight
  - Filter
  - Enrich
  - Mask/Redact
  - Modify (conditionally)

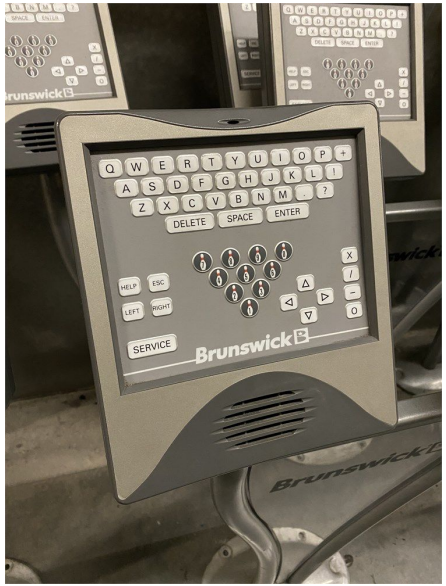


The Dude’s Five Steps

1. Standardize & measure
2. Transform
3. Filter
4. Aggregate
5. Test & Deploy

Standardize & Measure

- OTel Semantic Conventions
  - Common naming conventions
  - e.g. system.cpu.utilization vs (*host.cpu.utilization* AND *node.cpu.utilization* AND *mysnowflake.cpu.utilization*)
- Reference Tagging/label Namespace
  - e.g. your consistent labels/tags
- Measure Adherence



I'll tell you what I'm blathering about! I've got information, man!

Transform Processor

- Ideally update instrumentation at source
- When you can't → *transform* processor FTW!

```
replace_all_matches(attributes, "/user/*/list/*", "/user/{userId}/list/{listId}")

delete_key(attributes, attributes["http.request.header.authorization"])

set(attributes["environment"], "production") where attributes["service.name"] ==
"my_service"
```

This isn't Nam. There are rules. OTTL rules.

Filter Processor – The Nihilist Slayer

- Drop user-level tags
- Strip vanity metrics
- Block noisy pods

We don’t negotiate. We filter.

Filter Processor – The Nihilist Slayer

```
# Drop metric by name
metrics:
  metric:
    - 'name == "my.gauge"'

# Drop by tag key
metrics:
  metric:
    -'HasAttKeyOnDatapoint("user_id")'

# Drop by key and value
metrics:
  metric:
    - 'HasAttrOnDatapoint("user_id", "12345")'

# Drop by datapoint only
metrics:
  datapoint:
    - attributes["user_id"] == "12345"'
```

Aggregation

- We don’t always need every metric
- Max, Min, Avg or Percentile etc..
- e.g. P95 Service Latency
- Interesting connectors
  - Logs to Metrics
  - Spans to Metrics

This aggregation will not stand, man.

Aggregation

```
metric_statements:
  - context: metric
    statements:
      - copy_metric(name="my.second.histogram") where name == "my.histogram"
      - aggregate_on_attributes("sum", []) where name == "my.second.histogram"
```

Transform Processor Example

Aggregation - Count processor

Use the *count processor* to turn a stream of high cardinality telemetry into a counter.

```
processors:
  # Count processor to aggregate metrics
  count/aggregate:
    metrics:
      api.request.count:
        # Choose attributes to group by — fewer attributes = lower cardinality
        attributes:
          - key: project_id
        aggregation_type: sum
        aggregation_temporality: cumulative
```

Count Processor Example

Test & Deploy

- Pre-Collector
  - ottl.run (transform & filter processor)
  - Avoids “OTTL rage” (be careful with AI)
- Your Collector
  - Synthetic Telemetry
    - Otelgen; Telemetrygen
  - Your Telemetry
    - debug exporter
    - file exporter, otlpjsonfilereceiver
    - AWS S3 Exporter/Receiver
    - Tap Processor
  - Validation
    - OTelBin
- Deployment
  - Helm; OTel Operator; Control Planes (OpAMP)

I can get you a toe by 3 o’clock this afternoon… with nail polish.

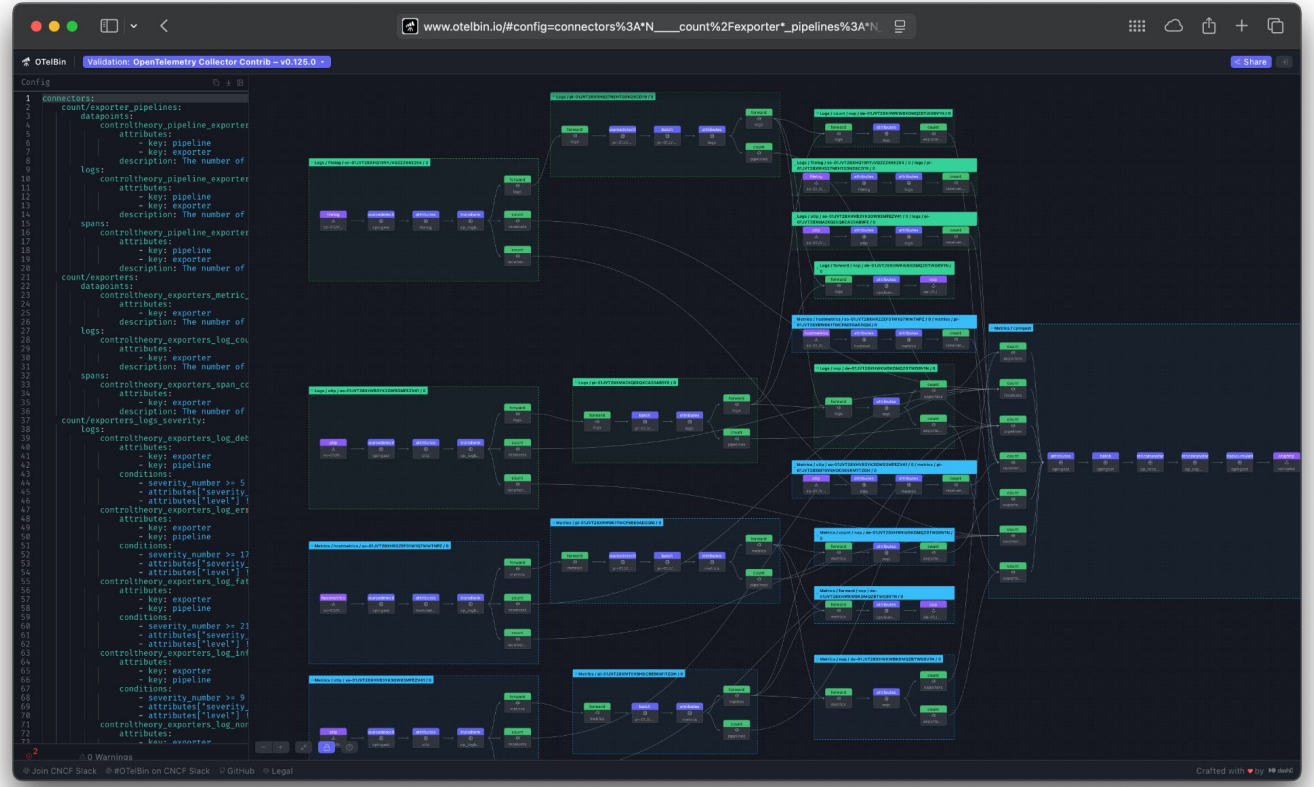
Synthetic Telemetry

```
# telemetrygen
docker run --rm
ghcr.io/open-telemetry/opentelemetry-collector-contrib/telemetrygen:latest \
  traces \
  --otlp-endpoint=host.docker.internal:4317 \
  --otlp-insecure

#otelgen
docker run --rm ghcr.io/krzko/otelgen \
  --otel-exporter-otlp-endpoint host.docker.internal:4317 \
  --rate 50 \
  --service-name test-app \
  --insecure \
  t m -t 150
```

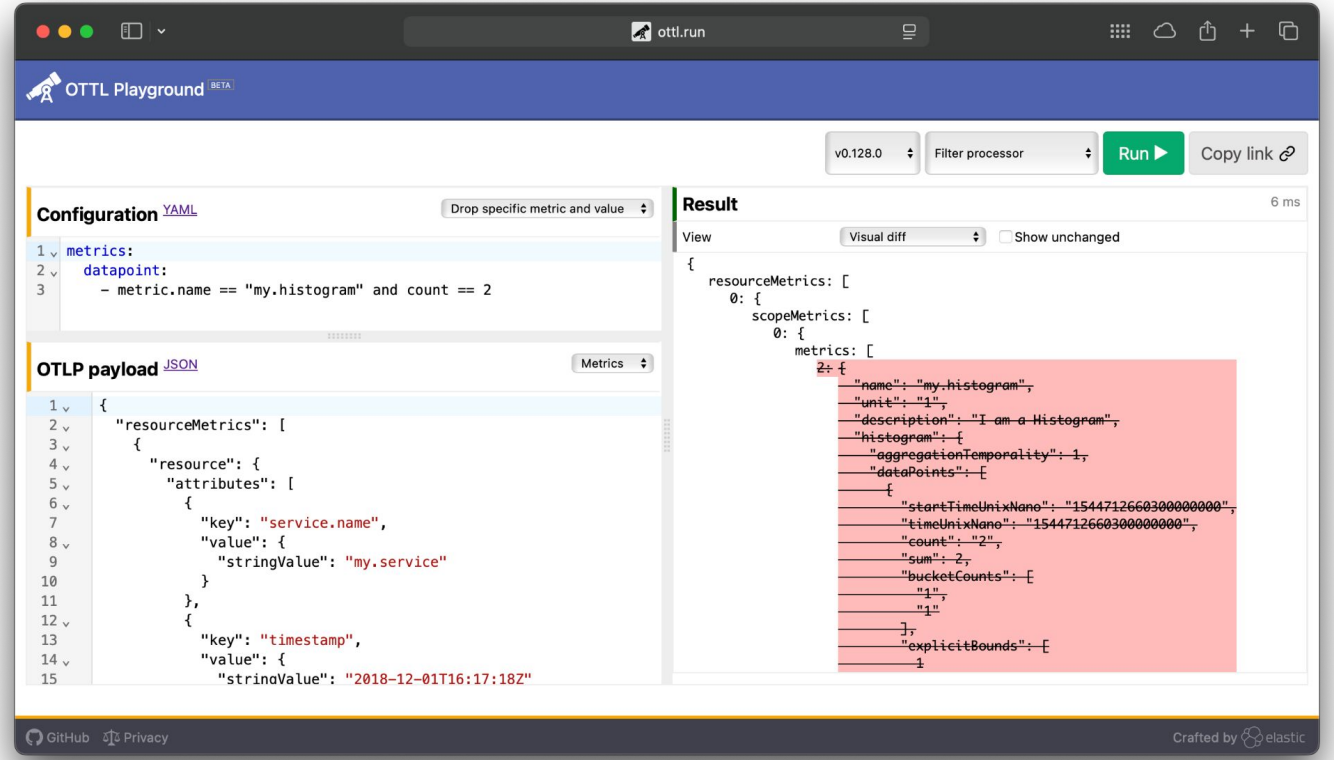
Synthetic Telemetry - Cardinality Spammer

```
# telemetrygen
# creates up to 5 million unique time series!

telemetrygen metrics \
  --otlp-endpoint=localhost:4317 \
  --otlp-insecure \
  --duration=60s \
  --rate=200 \
  --attributes user_id={1..500},project_id={1..100},order_id={1..500} \
  --metrics-count=1 \
  --metric-names=api.request.count \
  --value-distribution=normal
```

OTelBin - Config validator



Demo - OTTL Playground



Links That Tie the Room Together!

Resource	Link
OTel Collector	https://opentelemetry.io/docs/collector/
Transform Processor	https://github.com/open-telemetry/opentelemetry-collector-contrib/tree/main/processor/transformprocessor
Filter Processor	https://github.com/open-telemetry/opentelemetry-collector-contrib/tree/main/processor/filterprocessor
Telemetrygen	https://github.com/open-telemetry/opentelemetry-collector-contrib/tree/main/cmd/telemetrygen
Otelgen	https://github.com/krzko/otelgen
Semantic Conventions	https://opentelemetry.io/docs/concepts/semantic-conventions/
OTelBin	https://www.otelbin.io/
OTTL	https://github.com/open-telemetry/opentelemetry-collector-contrib/tree/main/pkg/ottl
OTTL Playground	https://ottl.run/
ControlTheory Resources	https://www.controltheory.com/resources/

TL;DR

- Cardinality blows up bills & noise
- Audit telemetry and set a common destination
- Fix instrumentation, & where you can’t:
  - Transform, Filter, Aggregate
- Test, deploy, repeat.

Talk & Resources

Come see us after at the lanes!

OpenTelemetry

CONTROL THEORY

Questions?

www.controltheory.com

Eric Anderson

CTO
ControlTheory

https://www.linkedin.com/in/eanderson/

@destari

@destari

Jon Reeve

CPO
ControlTheory

https://www.linkedin.com/in/jonathan-m-reeve-phd-8406a71/

@j0nr33v3

@j_m_reeve