KubeCon | CloudNativeCon

North America 2024

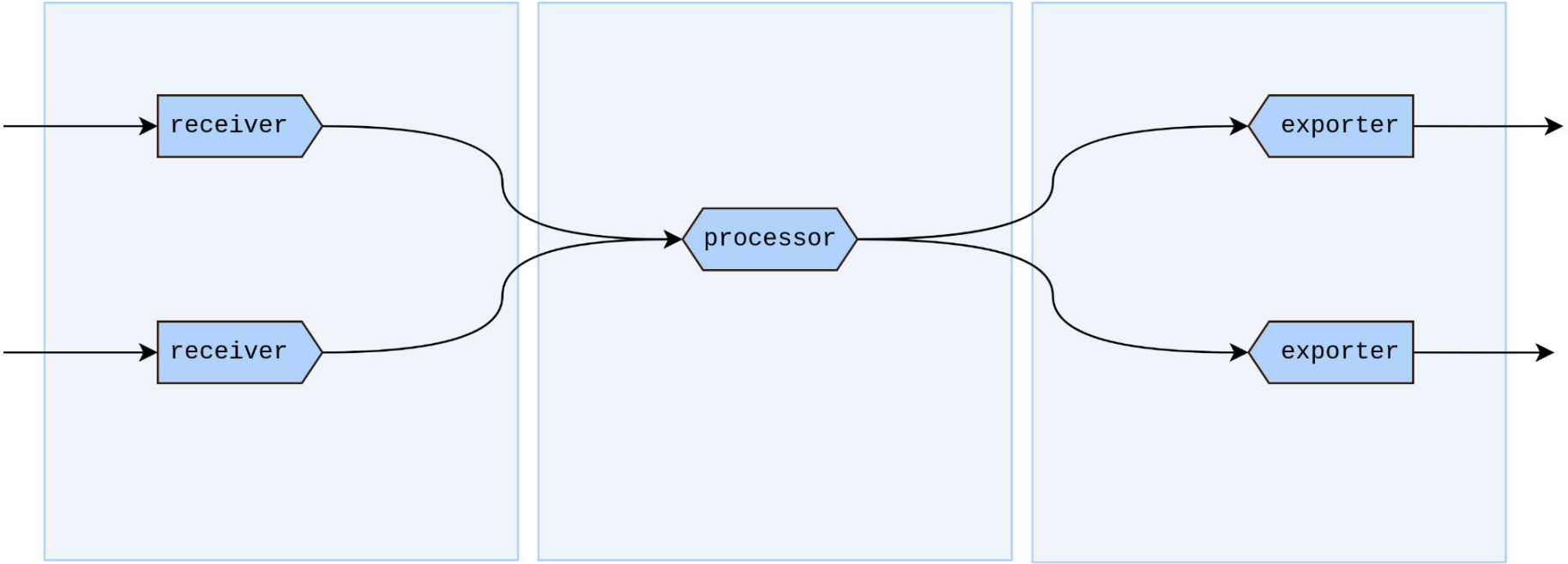
The OTTL Cookbook

Tyler Helmuth, Honeycomb
Evan Bradley, Dynatrace

Agenda

1. Quick intro to the OpenTelemetry Collector and OTTL
2. Solve a few scenarios with OTTL
3. Audience scenarios

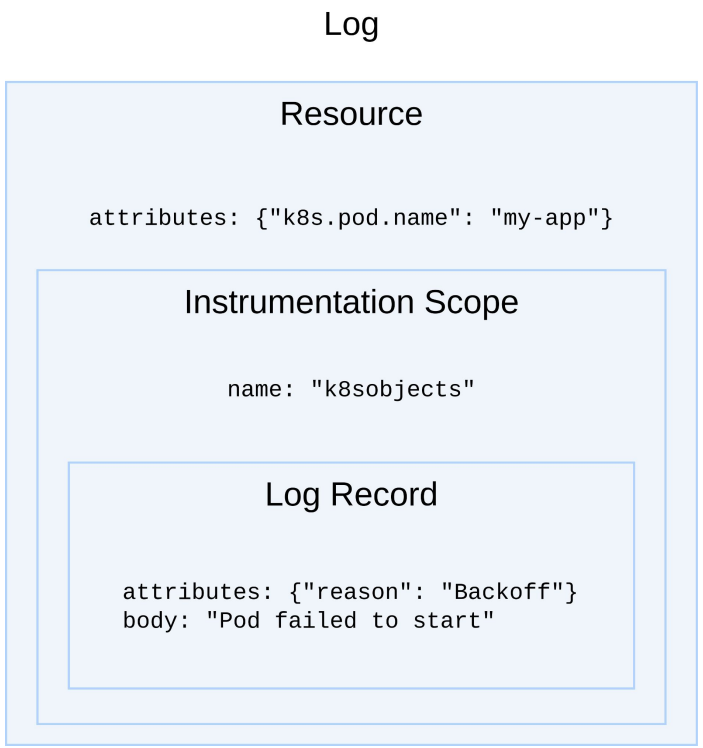
Quick Collector Intro



OpenTelemetry Transformation Language

OTTL is a Domain Specific Language (DSL) tailor-made for interacting with telemetry in the OTel Collector.

- Provides access to all OTLP fields.
- Enables expressing complex transformations using a simple syntax.



Use Case: Parse Unstructured Logs

We’re receiving unstructured MySQL logs

Recipes:

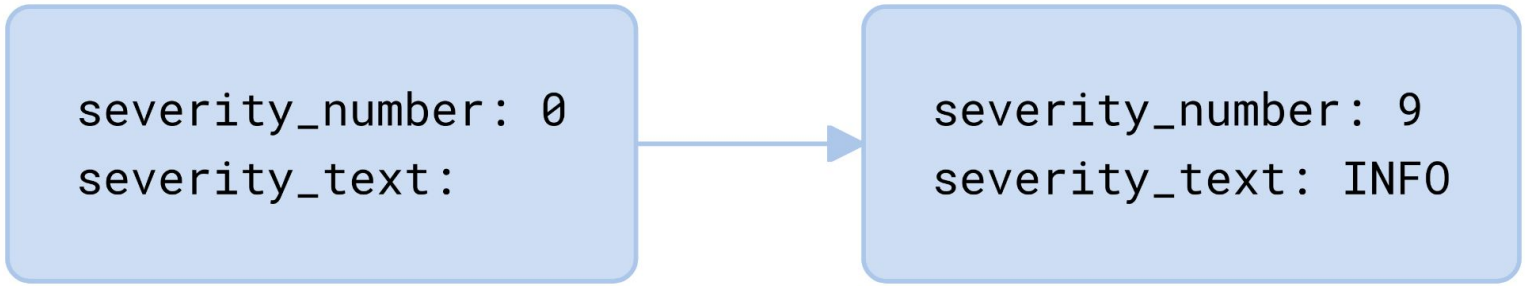
- Set OTLP log fields
- Parse into a structured format
- Conditionally set values
- Update an OTLP resource

Log example

```
# Time: 2024-10-10T19:34:04.232967Z
# Query_time: 0.776920
SELECT * FROM my_table
```

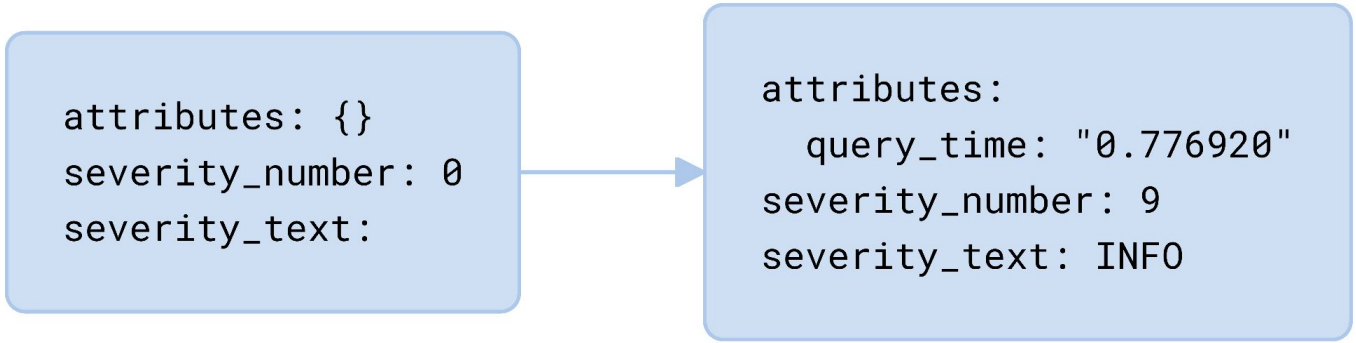
Recipe: Set a Field


```
set(severity_number, SEVERITY_NUMBER_INFO)
set(severity_text, "INFO")
```



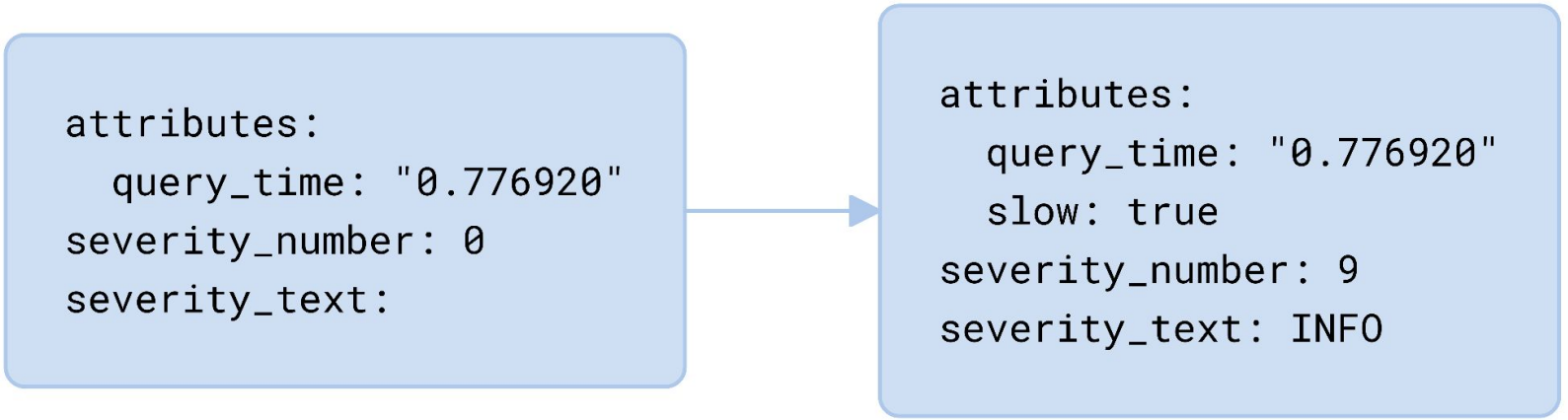
Recipe: Parse Unstructured Log


```
merge_maps(attributes, ExtractPatterns(body, "Query_time: (?P<query_time>[0-9\.]+)"), "upsert")
```



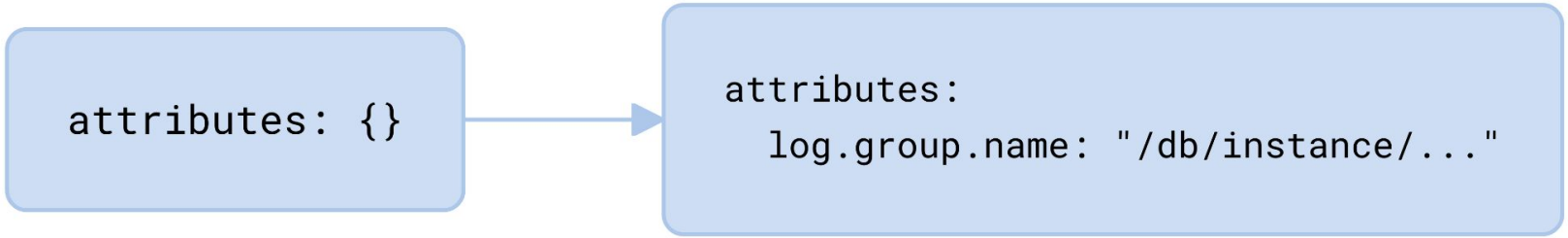
Recipe: Conditionally Set Values


```
set(attributes["slow"], true) where Int(attributes["query_time"]) > 0.7
```



Recipe: Update a Resource


```
Context: resource
set(attributes["log.group.name"], "/db/instance/production-mysql-mycompany")
```



All Together


```
log_statements:
  - context: resource
    statements:
      - set(attributes["log.group.name"], "/db/instance/production-mysql-mycompany")

  - context: log
    statements:
      - set(severity_number, SEVERITY_NUMBER_INFO)
      - set(severity_text, "INFO")
      - merge_maps(attributes, ExtractPatterns(body, "<big regex here"), "upsert")
      - set(attributes["slow"], true) where attributes["query_time"] > 0.7
```

Use Case: Manipulating JSON

We’re receiving JSON strings in our log bodies and need to process them.

Recipes:

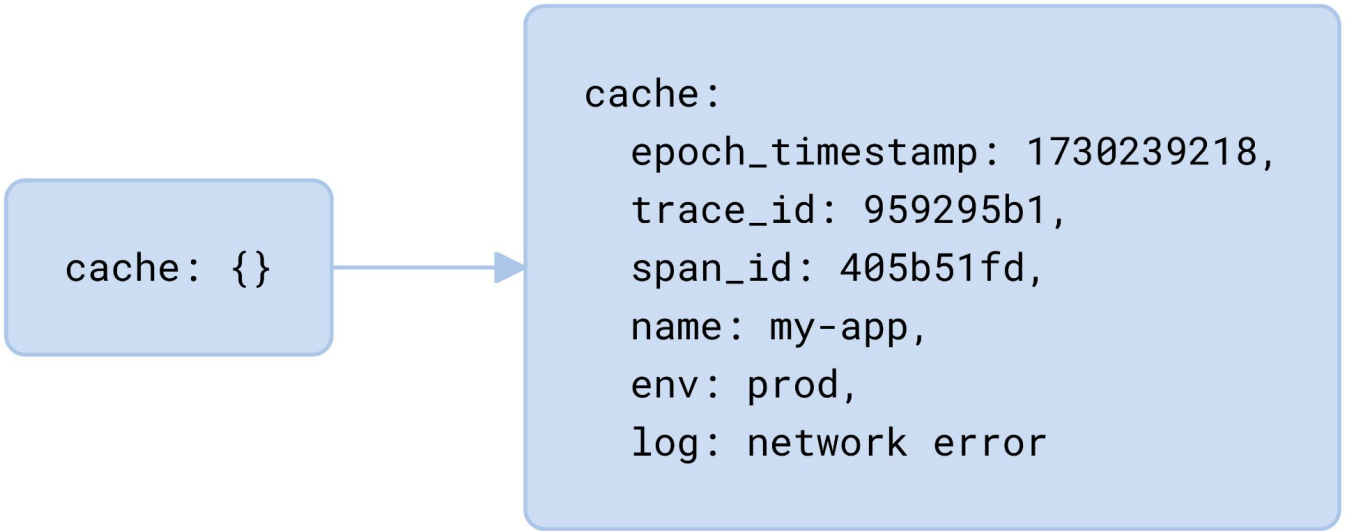
- Parse a JSON log
- Work with timestamps
- Handle trace/span IDs
- Manipulate strings

Formatted JSON body

```
{
  "object": {
    "timestamp": 2024-10-29T22:17:30Z,
    "trace_id": "959295b148d6a8c71728ef6a3eaaef80",
    "span_id": "405b51fdec8afd00",
    "brand": "otel",
    "item": "telescope",
    "log": "stuff is broke"
  },
  "version": "v1"
}
```

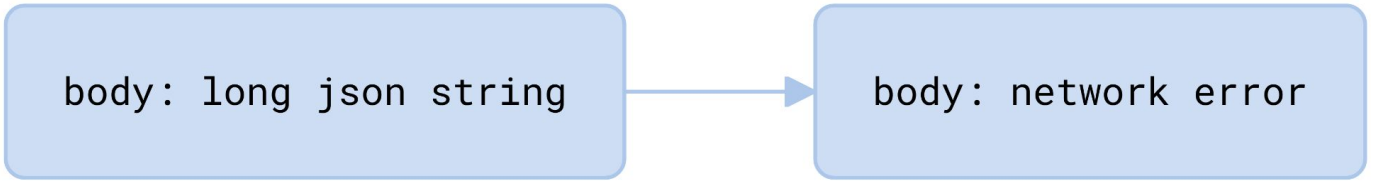
Recipe: Parse JSON


```
merge_maps(cache, ParseJSON(body)["object"], "upsert")
```



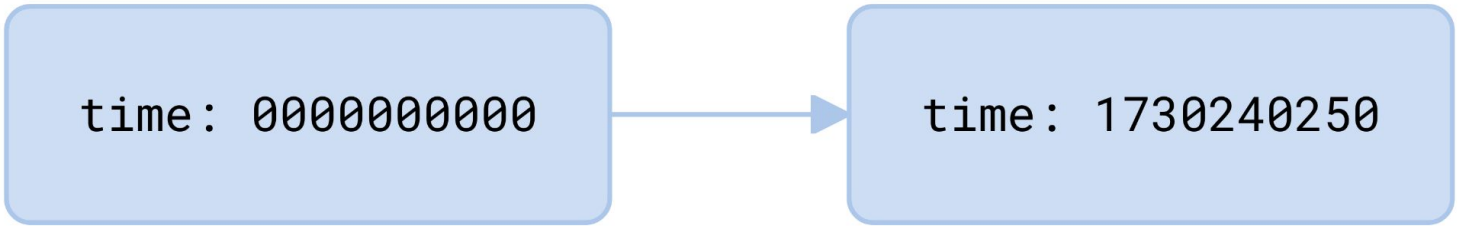
Recipe: Parse JSON


```
set(body, cache["log"])
```



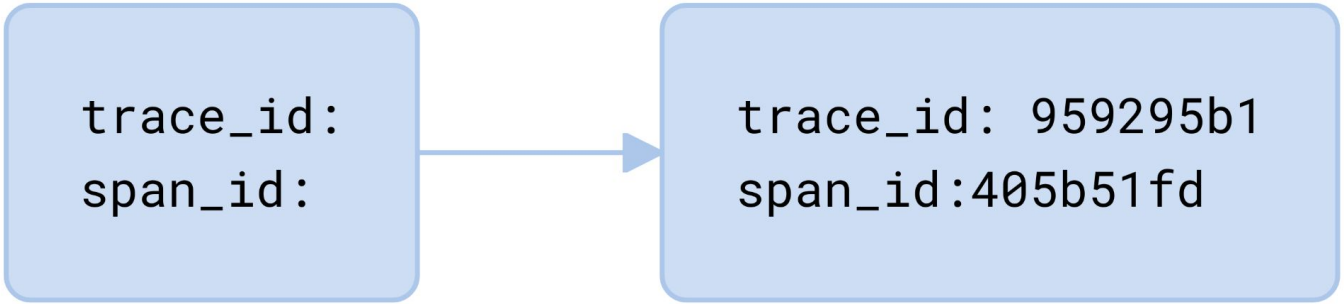
Recipe: Work with Timestamps


```
set(time, Time(cache["timestamp"], "%Y-%m-%dT%H:%M:%SZ"))
```



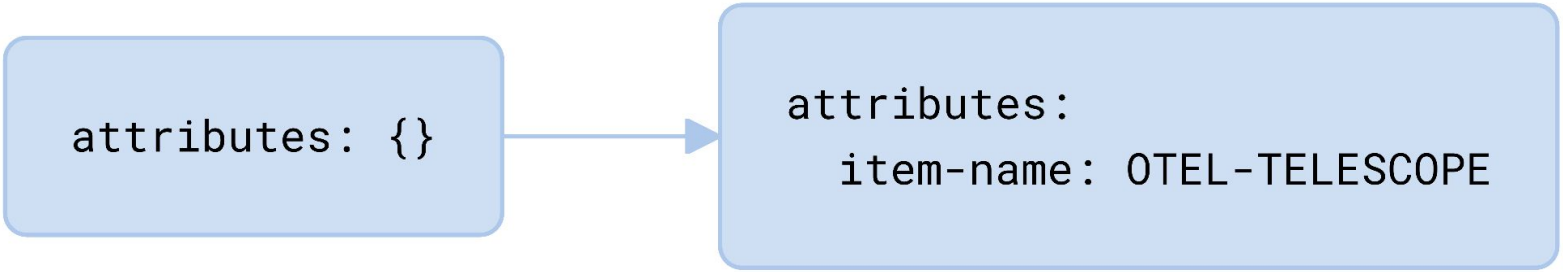
Recipe: Handle Trace/Span IDs


```
set(trace_id.string, cache["trace_id"])
set(span_id.string, cache["span_id"])
```



Recipe: Manipulate Strings


```
set(attributes["item-id"], ConvertCase(Concat([cache["brand"], cache["item"]], "-"), "upper"))
```



All Together

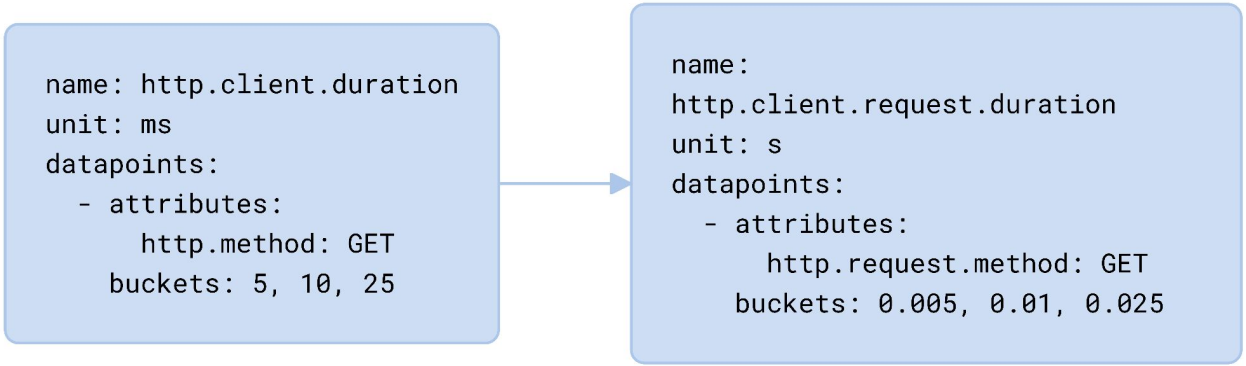

```
log_statements:
  - context: log
    statements:
      - merge_maps(cache, ParseJSON(body)["object"], "upsert")
      - set(body, cache["log"])
      - set(time, Time(cache["timestamp"], "%Y-%m-%dT%H:%M:%SZ"))
      - set(trace_id.string, cache["trace_id"])
      - set(span_id.string, cache["span_id"])
      - set(attributes["item-id"], ConvertCase(Concat([body["brand"], body["item"]], "-"), "upper"))
```

Use Case: Metrics to Stable SemConv

Need to coalesce different HTTP semantic conventions version

Recipe:

- Reuse conditions
- Rename an attribute
- Scaling a metric
- Rename a metric

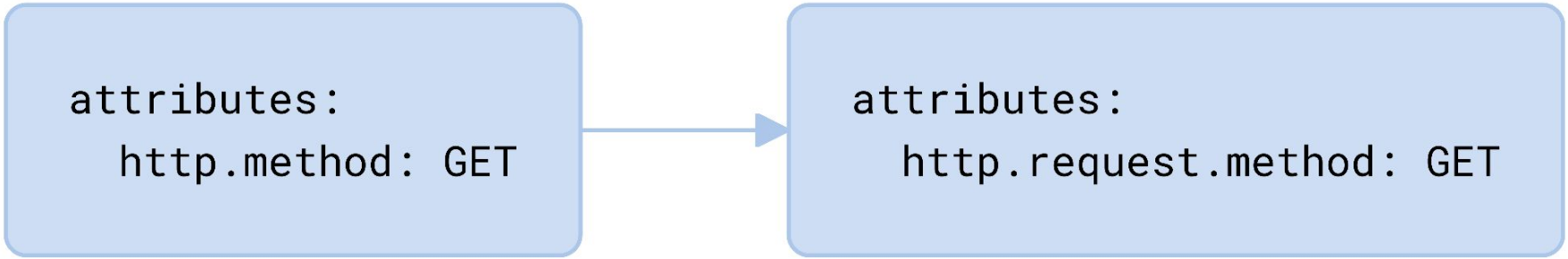


Recipe: Reuse Conditions


```
metric_statements:
    - context: datapoint
      conditions:
        - metric.name == "http.client.duration"
        - metric.name == "http.server.duration"
      statements:
        - [...]
```

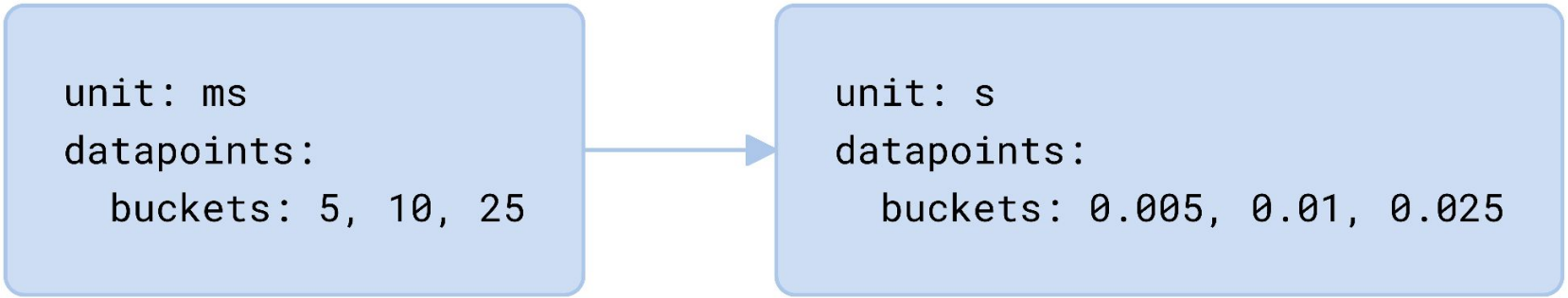
Recipe: Rename an Attribute


```
set(attributes["http.request.method"], attributes["http.method"])
delete_key(attributes, "http.method")
```



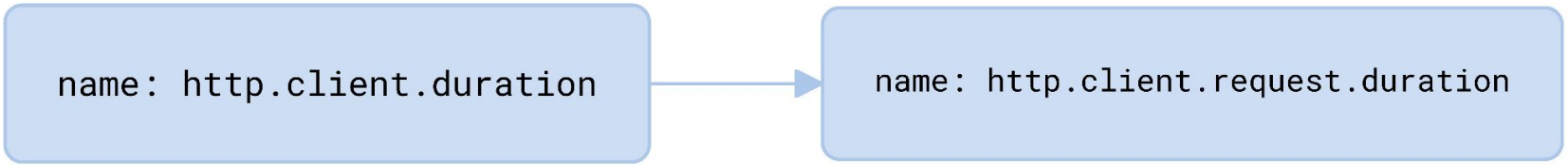
Recipe: Scale a Metric


```
scale_metric(0.001, "s") where unit == "ms"
```



Recipe: Rename a Metric


```
set(name, "http.client.request.duration") where name == "http.client.duration"
set(name, "http.server.request.duration") where name == "http.server.duration"
```



All Together


```
metric_statements:
  - context: datapoint
    conditions:
      - metric.name == "http.client.duration"
      - metric.name == "http.server.duration"
    statements:
      - set(attributes["http.request.method"], attributes["http.method"])
      - delete_key(attributes, "http.method")

  - context: metric
    conditions:
      - name == "http.client.duration"
      - name == "http.server.duration"
    statements:
      - scale_metric(0.001, "s") where unit == "ms"
      - set(name, "http.client.request.duration") where name == "http.client.duration"
      - set(name, "http.server.request.duration") where name == "http.server.duration"
```

Use Case: Converting Prometheus

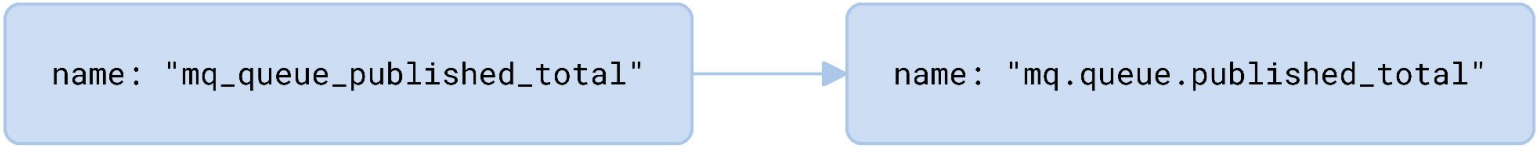
We are scraping the Prometheus endpoint of our message queue

Recipes:

- Dynamically rename a metric
- Aggregate a metric using the transform processor
- Restructure a metrics payload

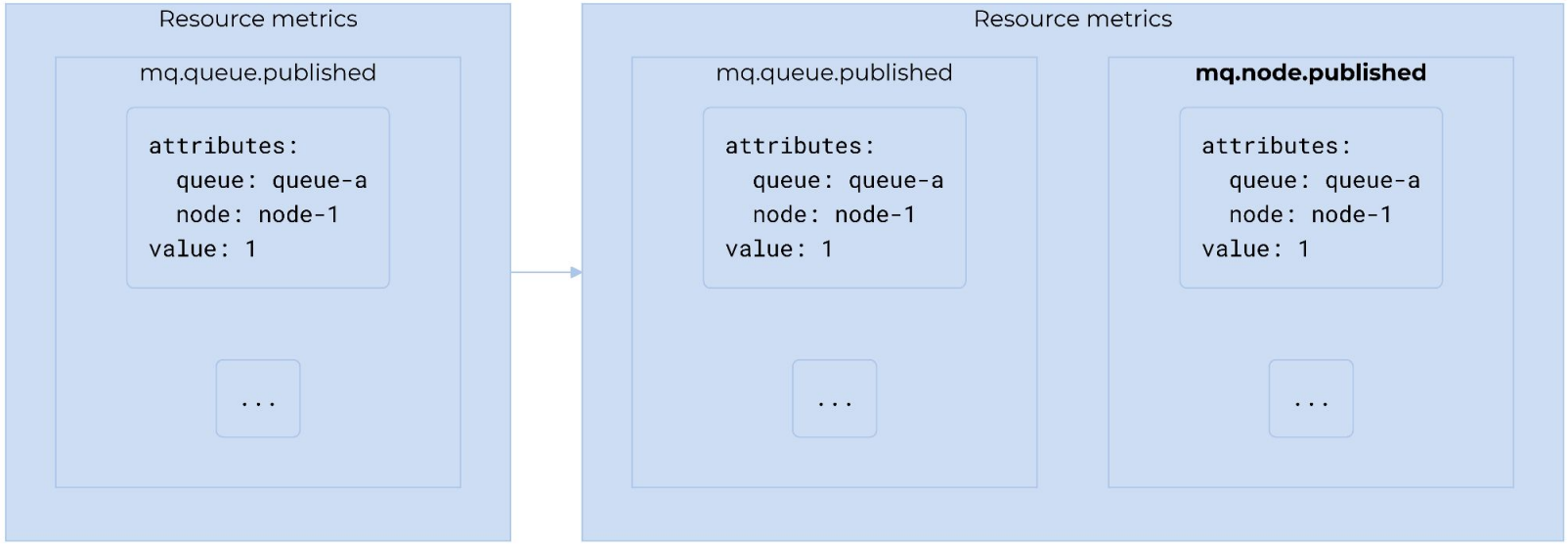
Recipe: Dynamically Rename a Metric


```
Context: metric
replace_pattern(name, "mq_queue_(.+)", "mq.queue.$$1")
```



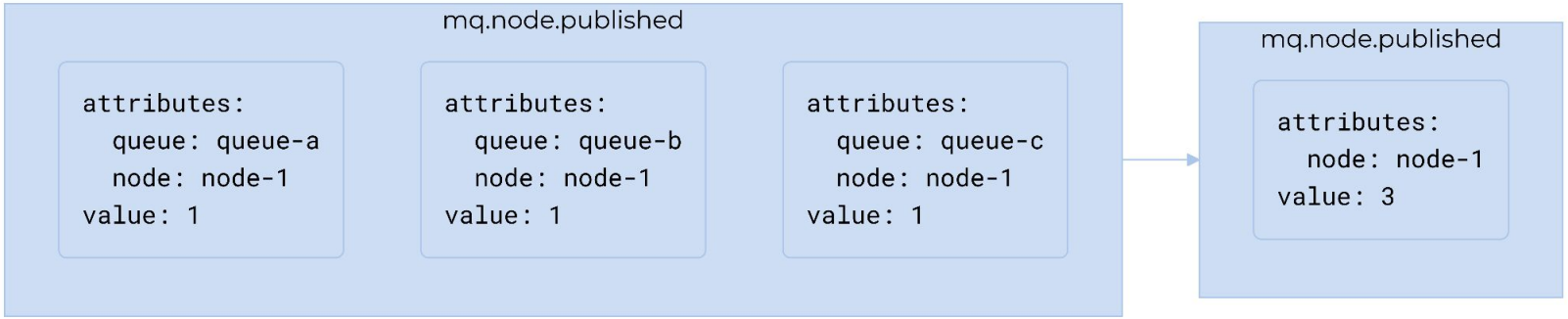
Recipe: Aggregate a Metric


```
Context: metric
copy_metric(name="mq.node.published") where name == "mq.queue.published"
```



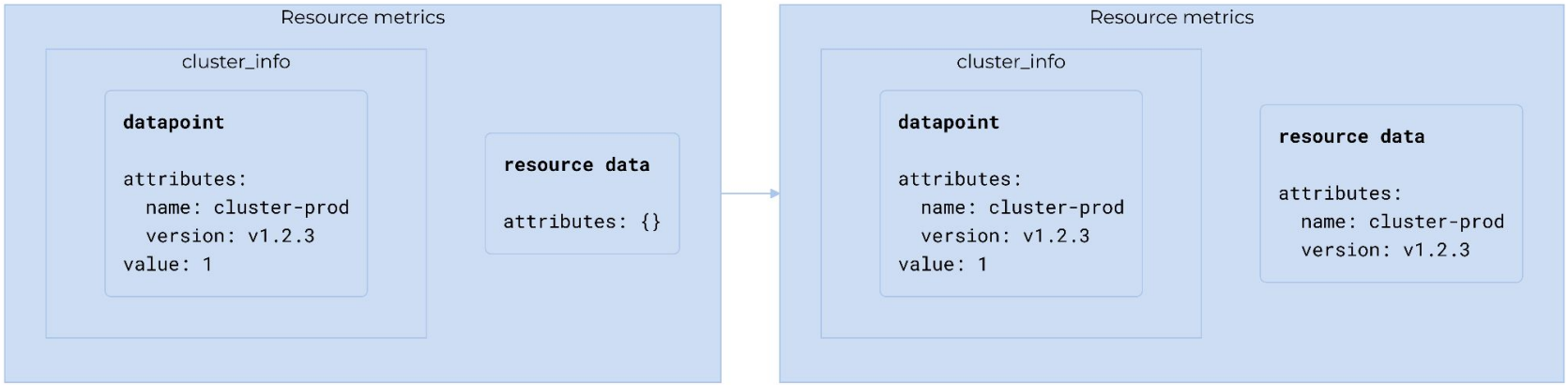
Recipe: Aggregate a Metric


```
Context: metric
aggregate_on_attributes("sum", ["node"]) where name == "mq.node.published"
```



Recipe: Restructure a Metrics Payload


```
Context: datapoint
merge_maps(resource.attributes, attributes, "upsert") where meric.name == "mq_cluster_info"
```



Recipe: Restructure a Metrics Payload


```
filter:
  metrics:
    metric:
        - name == "mq_cluster_info"
```

All Together

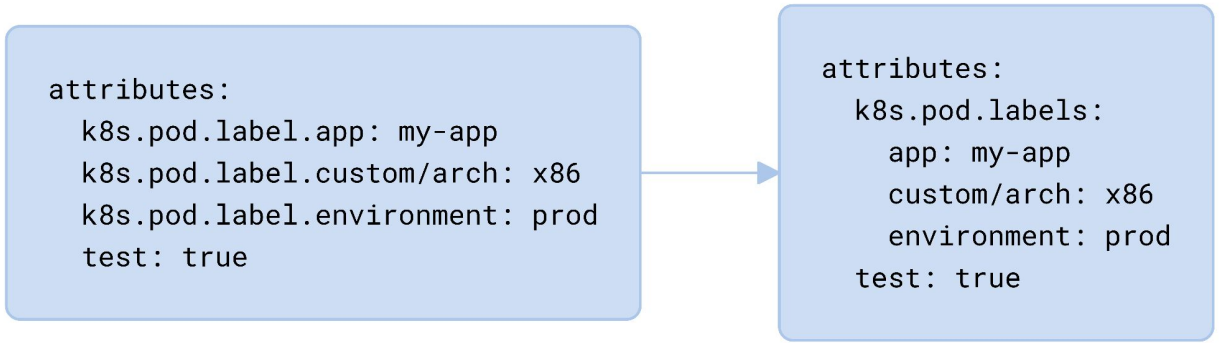

```
transform:
   metric_statements:
   - context: metric
     statements:
     - replace_pattern(name, "mq_queue_(.+)", "mq.queue.$$1")
     - copy_metric(name="mq.node.publish") where name == "mq.queue.published_total"
     - aggregate_on_attributes("sum", ["node"]) where name == "mq.node.publish"

   - context: datapoint
     statements:
     - merge_maps(resource.attributes, attributes, "upsert") where meric.name == "mq_cluster_info"

 filter:
   error_mode: ignore
   metrics:
     metric:
         - name == "mq_cluster_info"
```

Use Case: Clean Up Span Attributes

Our spans have some attributes that all share the same known prefix.

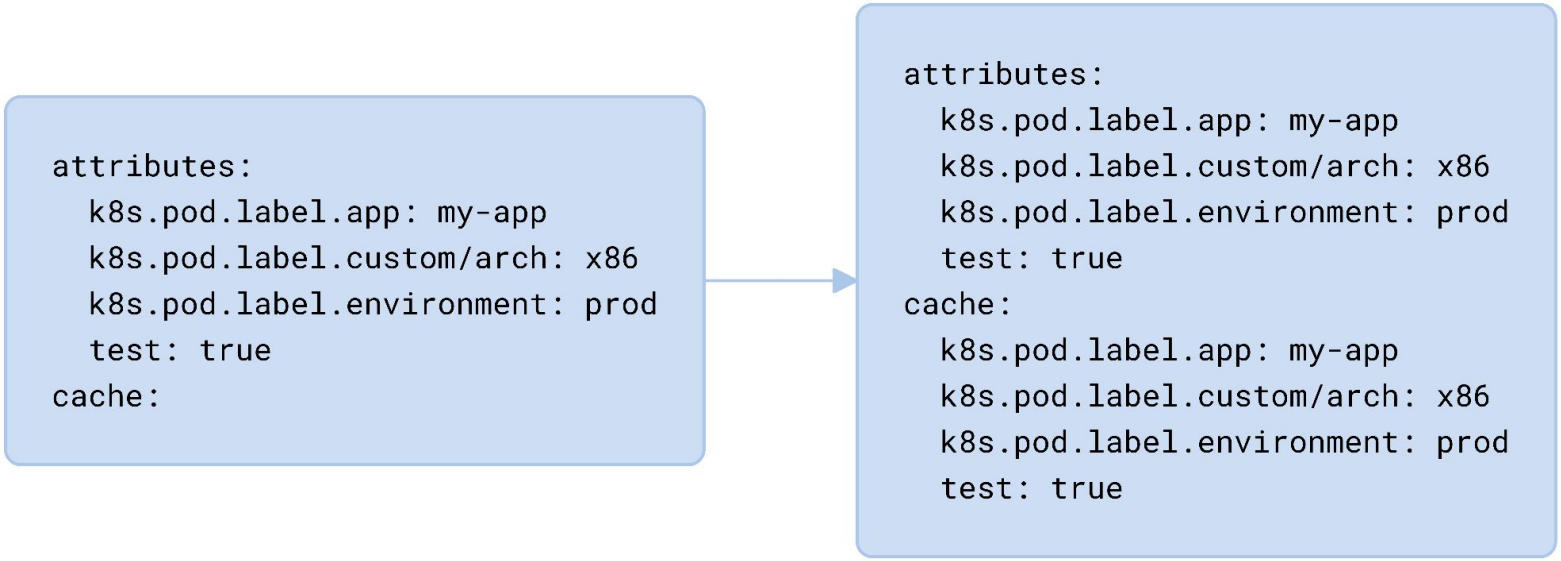
Recipe:

- Grouping related attributes



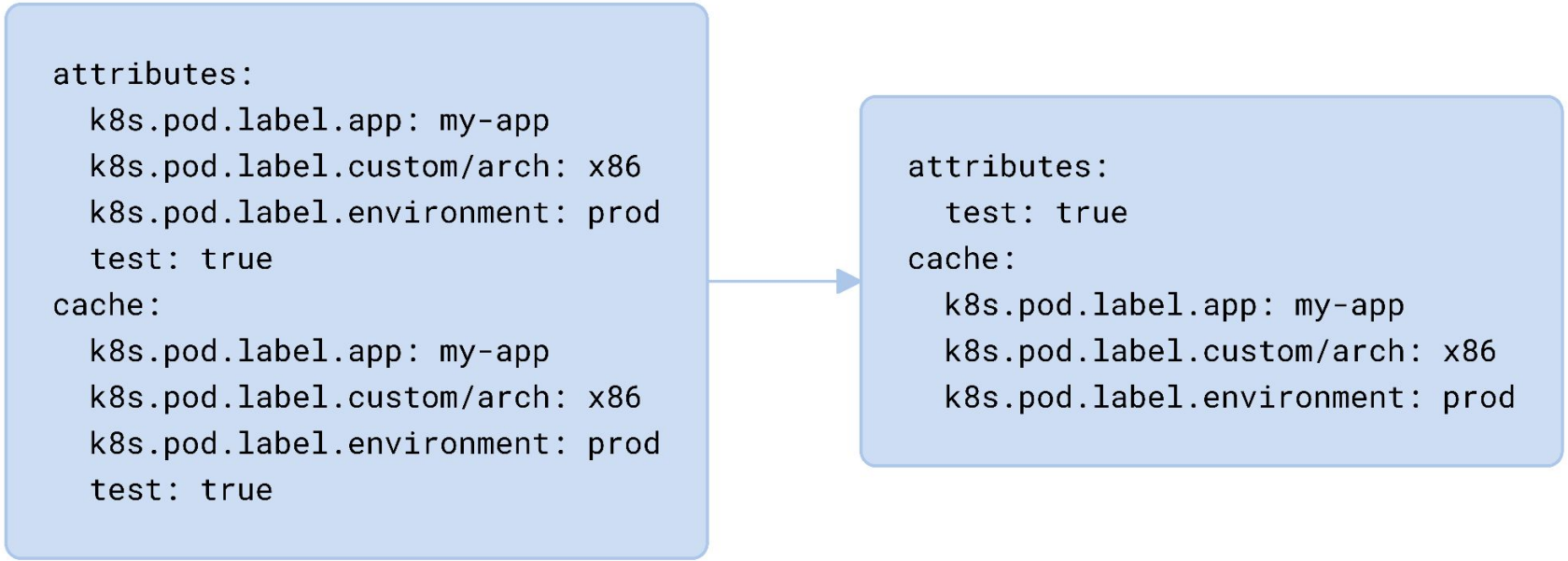
Recipe: Group Related Attributes


```
set(cache, attributes)
```



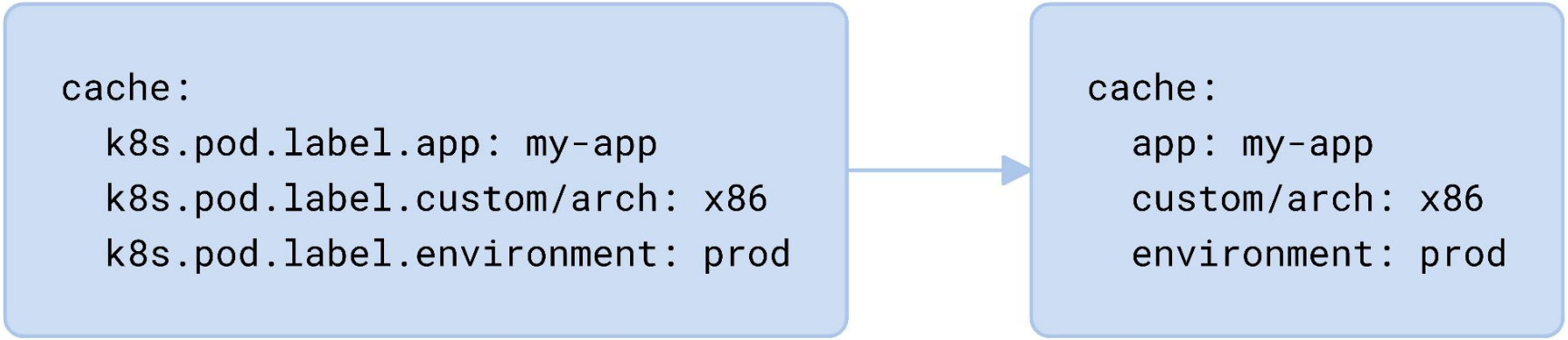
Recipe: Group Related Attributes


```
delete_matching_keys(attributes, "^k8s\.pod\.label\.")
keep_matching_keys(cache, "^k8s\.pod\.label\.")
```



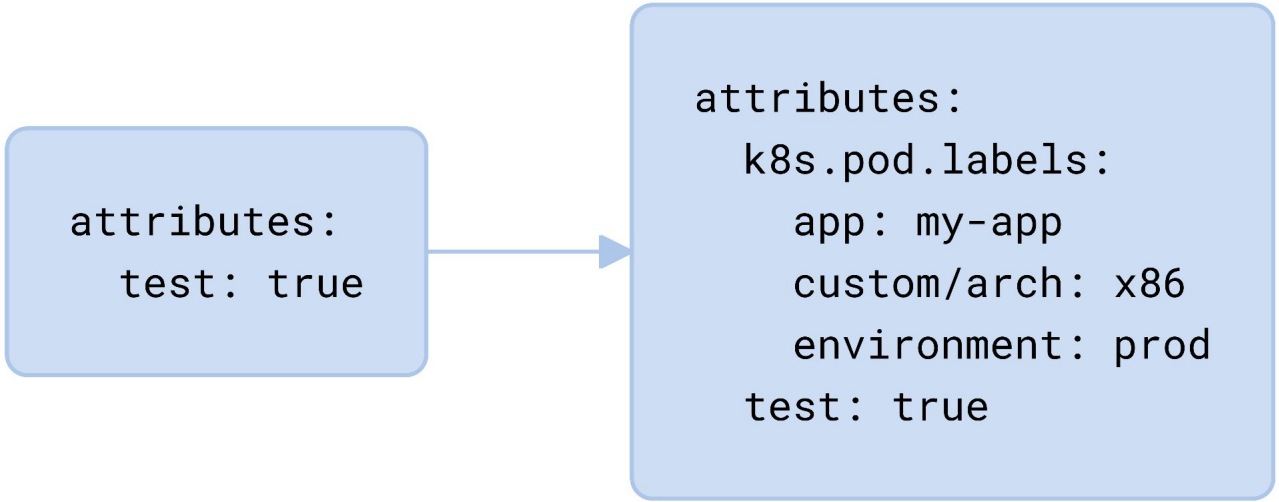
Recipe: Group Related Attributes


```
replace_all_patterns(cache, "key", "^k8s\.pod\.label\.", "")
```



Recipe: Group Related Attributes


```
set(attributes["k8s.pod.labels"], cache)
```



All Together


```
trace_statements:
  - context: resource
    statements:
      - set(cache, attributes)
      - delete_matching_keys(attributes, "^k8s\.pod\.label\.")
      - keep_matching_keys(cache, "^k8s\.pod\.label\.")
      - replace_all_patterns(cache, "key", "^k8s\.pod\.label\.", "")
      - set(attributes["k8s.pod.labels"], cache)
```

Your Turn!

If you have a use case in mind we'll try to solve it using OTTL

Restrictions:

- No stateful transformations
- No cross-signal transformations
- No Profile transformations - OTTL can't handle them yet.

We reserve the right to say "interesting problem, I need more time to solve that"

Your Turn!

We don’t need the “why” only the “what”.

Suggested templates for suggesting a scenario:

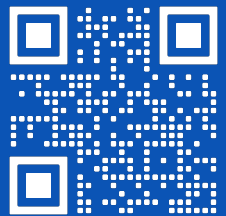
- **I need my data to** <blank> **instead of** <blank>
- **I need to drop data when** <blank>
- **I need to** <blank> **using** <blank>

Statements will be uploaded to the session page.

Want to Know More?

OTTL

https://bit.ly/ottl-docs



Transform processor

https://bit.ly/tp-docs