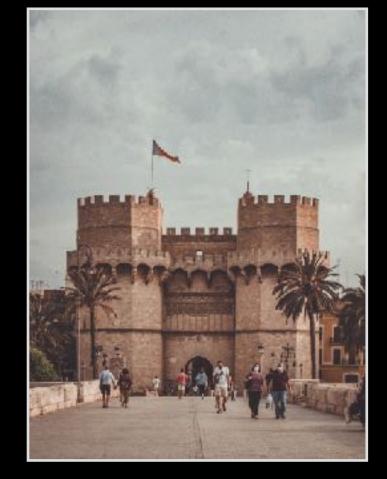


Diana Todea - Site Reliability Engineer - Elastic 7th of February 2024



Site Reliability Engineer at Elastic for 2 years
Live in Valencia, Spain
Mother of 2 young children
Love traveling, reading and sports



**UNSPLASH Juan Puyo** 

## CONTEXT

The main issue our organisation faced when migrating to serverless was scaling. Due to our very large dataset environment we needed our aggregations to be more efficient in dealing with cache results.

#### Two methods of manipulating data:

- -the rollup aggregations, which summarizes and stores historical data for later analysis (deprecated)
- -the transforms, which convert existing Elasticsearch indices into summarized indices (current)

# SUMMARY

**SLO CONCEPTS** 

**SLO INDICATOR TYPES** 

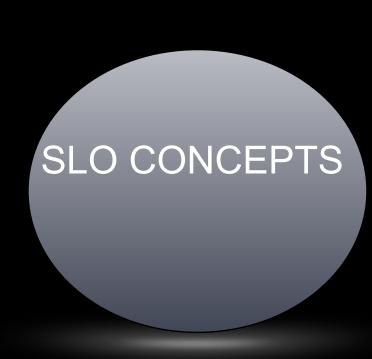
**SLO ARCHITECTURE** 

TRANSFORMS DEEP DIVE

**BURN RATE ALERTING** 

**DEMO** 

SLOS AND INCIDENT MANAGEMENT



#### Service Level Indicators

A measure of the service level provided. Usually defined as a ratio of good over total events. Range between 0% and 100%. Examples: Availability, throughput, request latency, error rate.

#### Service Level Objectives

Target value for a service level measured by an SLI. Above the threshold the service is compliant. Example: 95% of the successful requests are served under 100ms.

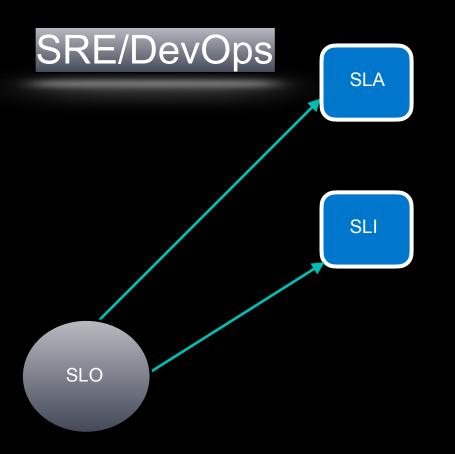
#### Error budget

Defined as 100% minus the SLO. Quantity of errors that is tolerated.

#### Burn rate

The rate at which we are burning the error budget over a defined period of time.

Very useful at alerting before exhausting the error budget.



SLA is the agreement that your company makes with your clients.

SLIs represent the real numbers on your system's performance.

SLOs are the goals your team must hit to meet the SLAs.

SLO is typically expressed as a percentage of time that a service is expected to meet a certain level of performance.

$$SLO = rac{ ext{Successful Operations}}{ ext{Total Operations}} imes 100$$

Burn rate is a measure of how quickly a company is using up its cash reserves.

 $Burn Rate = \frac{Initial Cash-Current Cash}{Time Period}$ 

Improve reliability and quality of services

#### GOOD SLO vs. BAD SLO

A bad SLO is vague and subjective, it lacks quantifiable metrics, it has an undefined threshold and has no observation window.

A good SLO is specific and measurable, it's user-centric, it's quantifiable and achievable and it's timeframe defined.

A well-defined SLO focuses on a crucial aspect of service quality, provides clarity, measurability and alignment with user expectations, which are essential elements for effective monitoring and evaluation of service reliability.

# SLO INDICATOR TYPES

#### APM AVAILABILITY

APM LATENCY

#### INDICATOR TYPES

CUSTOM KQL

CUSTOM METRIC

•HISTOGRAM METRIC

TIMESLICE METRICS

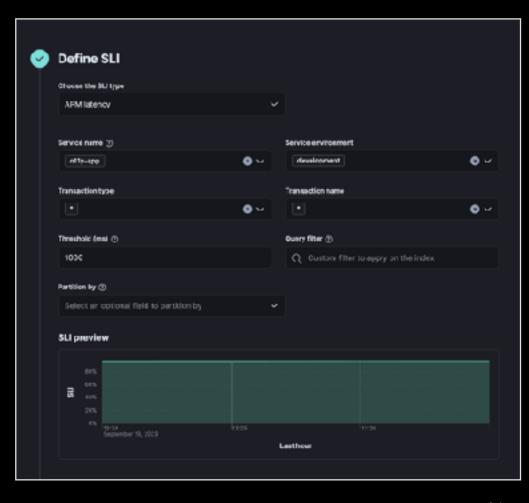
#### APM AVAILABILITY

- Designed to work exclusively with APM failed transaction rate
- Users can filter on service name,
   environment, transaction type and
   transaction name
- Support narrowing focus with a query filter
- Support partitioning by any field present in the APM data.
- Example SLOs: 99% of transactions are successful



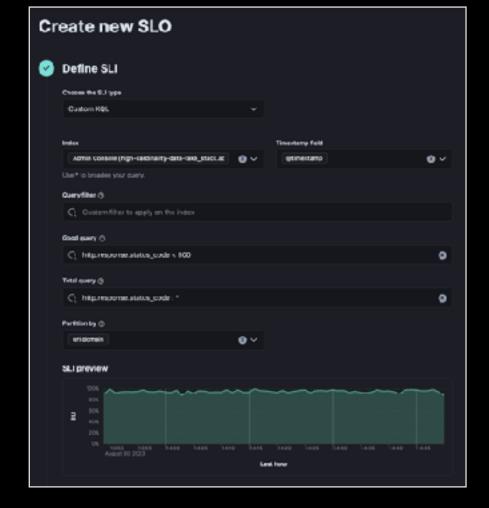
#### APM LATENCY

- Designed to work exclusively with APM transaction latency
- Users can filter on service name,
   environment, transaction type and
   transaction name
- Support narrowing focus with a query filter
- Support partitioning by any field present in the APM data.
- Example SLOs: 99% of transactions occur under 200ms



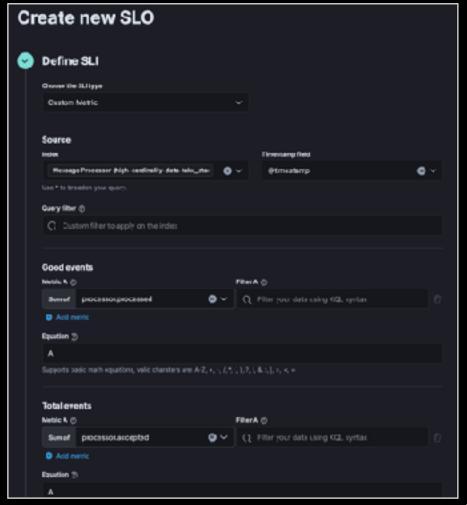
#### **CUSTOM KQL**

- Designed to work exclusively with custom logs
- The document count for the good filter is used for the numerator
- The document count for the total filter is used for the denominator
- Support partitioning by any keyword present in the index pattern.
- Example SLOs: 99% of requests occur under
   200ms, 99% of requests are successful, 99% of log messages are not errors



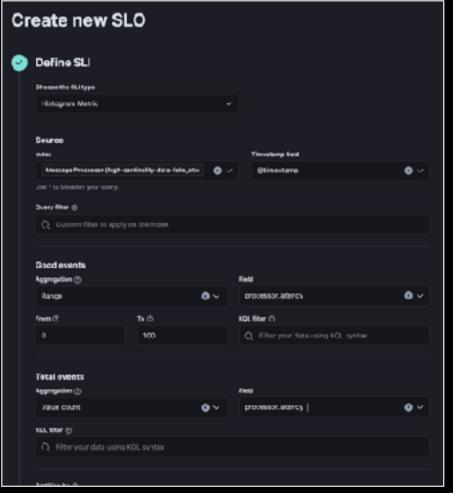
#### **CUSTOM METRIC**

- Designed to work with metric fields using a sum aggregation
- Supports basic math and boolean logic between multiple fields
- Support partitioning by any keyword present in the index pattern.
- Example SLOs: 99% of accepted events have been processed.



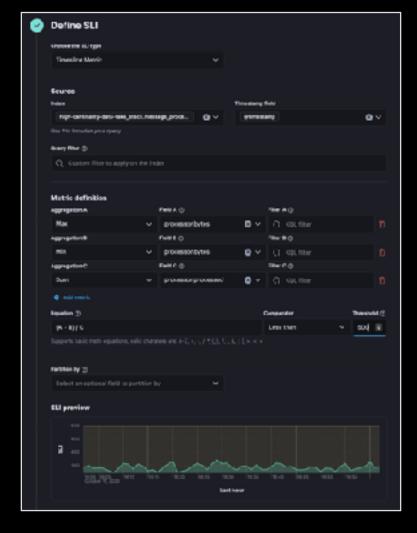
#### HISTOGRAM METRIC

- Designed to work with histogram fields.
- Histogram fields are useful for storing preaggregated high resolution data with minimal footprint.
- Users can define a range for the values, the counts are used for the good/total events.
- Support partitioning by any keyword present in the index pattern.
- Example SLOs: 99% of events have a latency between 0ms and 100ms



#### TIMESLICE METRIC

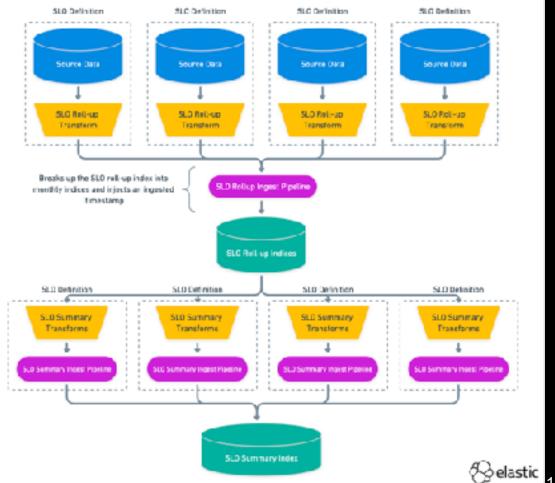
- Supports multiple aggregations: average,
   min, max, sum, cardinality, last value, std.
   deviation, percentiles and document count.
- Due to the nature of the aggregations and their relationship with the bucket size, this indicator forces the user to use a timeslice budgeting method.
- Allows the user to define a custom equation with multiple aggregations along with a threshold for defining a "good slice".



#### SLO ARCHITECTURE

SLOs rely on the **Transform** service to roll-up the source data into roll-up indices.

To support the group-by or the partition by feature we have added a second layer which summarizes the roll-up data into an entity-centric index for each SLO. This index also powers the search experience to allow users to search and sort by in any SLO dimension.



#### WHAT ARE TRANSFORMS?

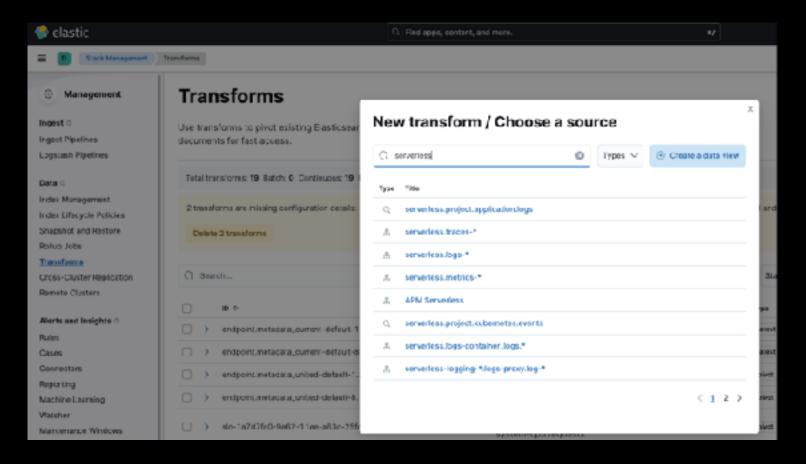
Transforms are persistent tasks that enable you to convert existing Elasticsearch indices into summarized indices, which provide opportunities for new insights and analytics. For example, you can use transforms to pivot your data into entity-centric indices that summarize the behavior of users or sessions or other entities in your data. Or you can use transforms to find the latest document among all the documents that have a certain unique key.

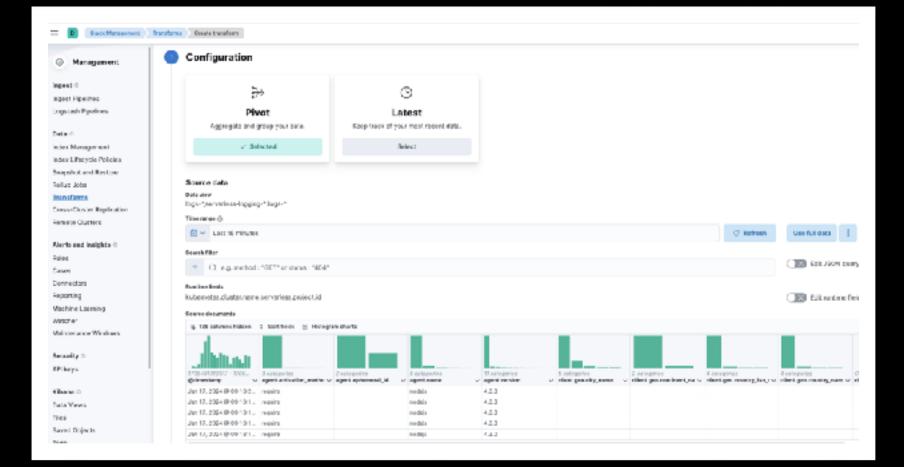
#### WHEN TO USE TRANSFORMS?

WHEN YOU NEED A
COMPLETE **FEATURE INDEX**RATHER THAN A TOP-N SET
OF ITEMS.

WHEN YOU NEED TO SORT AGRREGATION RESULTS BY A **PIPELINE AGGREGATION**. WHEN YOU WANT TO CREATE **SUMMARY TABLES**TO OPTIMIZE QUERIES.

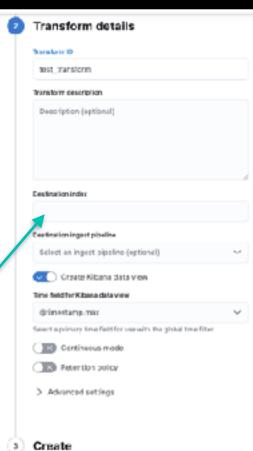
#### HOW TO CREATE TRANSFORMS?







Add a destination index: e.g. **serverless.logs** 



```
To Column 1 to 6 1440

To 200 Should 100 201 ...

200 Should 100 201 ...

200 Should 100 201 ...

201 Should 100 201 ...

202 Should 100 201 ...

203 Should 100 201 ...

204 Should 100 201 ...

205 Should 100 201 ...

205 Should 100 201 ...

206 Should 100 201 ...

207 Should 100 201 ...

208 Should 100 201 ...

208
```

#### Transform details

Transported

Best color roles

Service for the Service

Best color roles

Service for the Service

A Shore determined for constability to resolves

Element determined free final
global color roles

3. Advanced settings

#### Create

Change Charter transform and a fall map it now will be able to start the transform size by retaining to be assistance size.

\$100,00,000,000

Create and start

Copies to the disposal the Kissha bey Consile communition predict the transform.

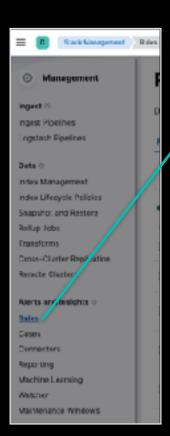
Overous and starts the transform. A transform will increase search confidencing load in

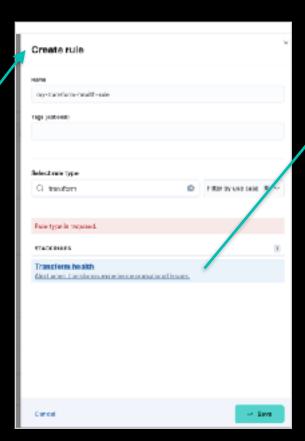
your duster. Please stop the transform if expensive lead in patienced. After the

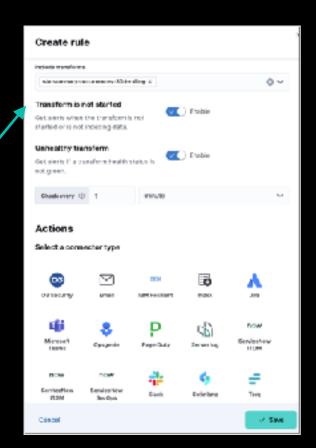
transform is started, you will be offered options to continue exploring the transform.

```
PUT transform/test transform
    "source": {
        "index": [
            "logs-*",
             "serverless-logging-*:logs-*"
        "runtime mappings": {
            "kubernetes.cluster.name": {
                 "script": {
                      "source": "String k8sClusterNameLabel = 'labels.k8s cluster name': \nString orchestratorLabel =
 'labels.orchestrator cluster name';\nString orchestratorField = 'orchestrator.cluster.name';\n\nif
(doc.containsKey(orchestratorField) && doc[orchestratorField].size() > 0) {\nemit(doc[orchestratorField].value);\n}
\nelse if (doc.containsKev(k8sClusterNameLabel) && doc[k8sClusterNameLabel].size() > 0)
{\nemit(doc[k8sClusterNameLabel].value);\n}\nelse if (doc.containsKey(orchestratorLabel) &&
doc[orchestratorLabel].size() > 0) {\nemit(doc[orchestratorLabel].value);\n}"
                  "type": "keyword"
            "serverless.project.id": {
                 "script": {
                     "source": "String projectIdKey = 'project.id'; \nString namespaceKey = 'kubernetes.namespace'; \nString
handlingProjectKey = 'handling project';\n\nif (doc.containsKey(projectIdKey) && doc[projectIdKey].size() > 0) {\n
emit(doc[projectIdKey].value); (n) else if (doc.containsKey(namespaceKey) & doc[namespaceKey].size() > 0 & and the containsKey(namespaceKey) & containsKey(namespaceKey
doc[namespaceKey].value.indexOf('project-') == 0) {\n emit(doc[namespaceKey].value.substring('project-'.length()));
\n\ else if (doc.containsKev(handlingProjectKev) && doc[handlingProjectKev].size() > 0) {\n
emit(doc[handlingProjectKey].value); \n}"
                 "type": "keyword"
    "pivot":
         "group by": {
             "http.response.status code": {
                 "terms": {
                      "field": "http.response.status code"
         "aggregations":
            "@timestamp.max": {
                 "max": {
                      "field": "@timestamp"
     "dest": {
        "index": "serverless.logs"
    "settings": {
         "num failure retries": 3
```

#### HOW TO GENERATE ALERTS FOR TRANSFORMS?







# The SLOs APIs can be used to manage your SLOs using your infrastructure as code system (e.g. Terraform)

PUT transform/<transform id> GET transform/<transform id>/ preview POST transform/<transform id>/ reset GET transform/<transform id> DELETE transform/<transform id> POST transform/<transform id>/ schedule now GET transform/ all POST transform/<transform id>/ start GET transform/\* POST transform/<transform id>/ stop GET transform/ stats POST transform/<transform id>/ update GET transform/ all/ stats POST \_transform/\_upgrade

GET transform/\*/ stats

### WHAT IS BURN RATE ALERTING?

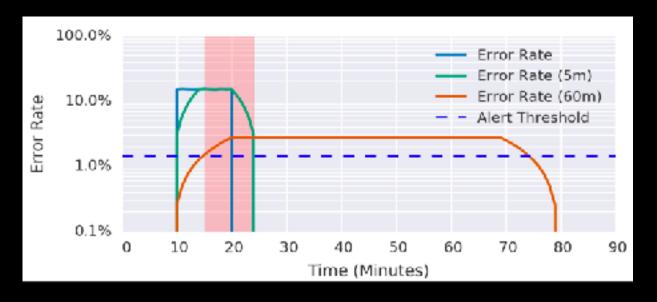
Burn rate alerting calculates the rate at which SLOs are failing over multiple windows of time.

Burn rate alerting is less sensitive to short term fluctuations by focusing on sustained deviations.

Burn rate alerting can give you an indication of how severly the service is degrading and it helps

prioritize multiple issues at the same time.

#### BURN RATES ALERTING WITH MULTIPLE WINDOWS



There are 2 windows for each severity: a short and a long. The short window is 1/12 of the long window.

When the burn rate for both windows exceeds the threshold, the alert is triggered.

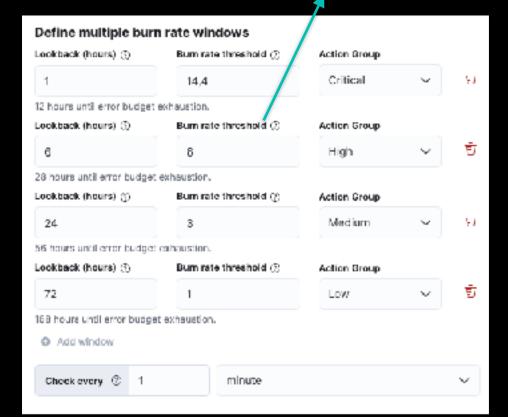
PROs: Reduced alert fatigue, improved user experience, a flexible alerting framework, good precision

CONs: Lots of options to configure

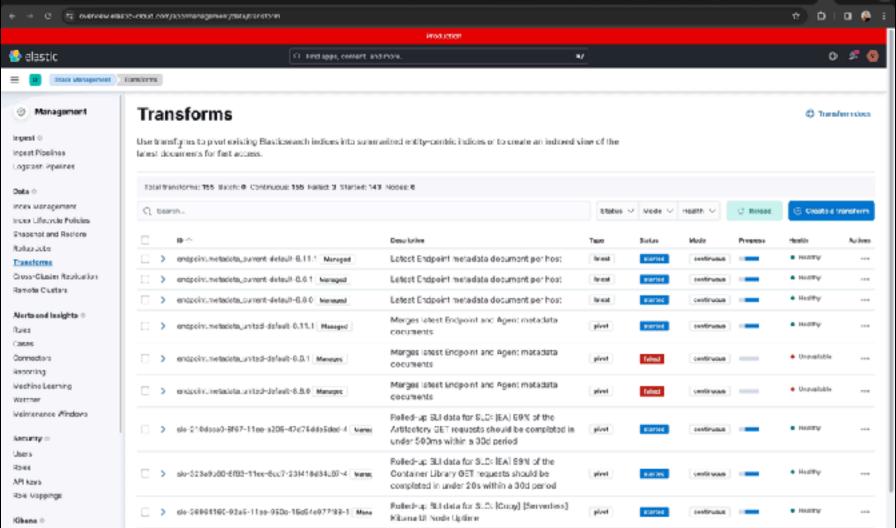
#### HOW TO CREATE A BURN RATE ALERT

How long you have until your error budget is exhausted.

Create rule	
Name	
my-burn-ratz-alert	
Tags (optional)	
SLO burn rate Alert when your SLC burn rate is too high over a defined period of time. Lea	× arn more
Choose a SLO to monitor	
[test][o11y]Latency of Kibana transactions	© ~



# DEMO TIME



CONTRACTOR OF A LANGUAGE PRODUCTION OF A PRODU

# SLOS AND INCIDENT MANAGEMENT

## **DEVELOPERS**

CUSTOMER SUPPORT/ EXPERIENCE TEAMS

SRE/DevOps



DATA SCIENTISTS

PRODUCT MANAGERS

### SRE/DEVOPS AND INCIDENT MANAGEMENT

- **MUNDERSTAND USER EXPECTATIONS**
- **M** DEFINE KEY SERVICE LEVEL INDICATORS
- **QUANTIFY SERVICE LEVEL INDICATORS**
- **CONSIDER USER IMPACT**
- **■** BALANCE RELIABILITY AND INNOVATION
- USE HISTORICAL DATA FOR INFORMED DECISIONS
- ✓ ITERATE AND REFINE
- MONITOR AND ALERT
- **M** DOCUMENT AND COMMUNICATE
- ☑ COLLABORATE ACROSS TEAMS



## HAPPY TRANSFORM, HAPPY SLO

# RESOURCES

- 1. <a href="https://www.elastic.co/guide/en/observability/current/slo.html">https://www.elastic.co/guide/en/observability/current/slo.html</a>
- 2. <a href="https://www.elastic.co/guide/en/observability/current/slo-create.html">https://www.elastic.co/guide/en/observability/current/slo-create.html</a>
- 3. https://www.elastic.co/guide/en/elasticsearch/reference/current/transforms.html
- 4. <a href="https://www.elastic.co/guide/en/elasticsearch/reference/current/transform-apis.html">https://www.elastic.co/guide/en/elasticsearch/reference/current/transform-apis.html</a>
- 5. <a href="https://docs.elastic.co/api-reference/observability/findslosop">https://docs.elastic.co/api-reference/observability/findslosop</a>
- 6. <a href="https://www.elastic.co/guide/en/elasticsearch/reference/current/transform-scale.html">https://www.elastic.co/guide/en/elasticsearch/reference/current/transform-scale.html</a>
- 7. <a href="https://www.elastic.co/guide/en/elasticsearch/reference/current/transform-examples.html">https://www.elastic.co/guide/en/elasticsearch/reference/current/transform-examples.html</a>
- 8. <a href="https://www.elastic.co/guide/en/observability/current/slo-burn-rate-alert.html">https://www.elastic.co/guide/en/observability/current/slo-burn-rate-alert.html</a>
- 9. <a href="https://www.elastic.co/guide/en/observability/current/observability-introduction.html">https://www.elastic.co/guide/en/observability/current/observability-introduction.html</a>
- 10. <a href="https://openapi.tools/">https://openapi.tools/</a>

# THANK YOU

- > https://www.linkedin.com/in/diana-todea-b2a79968/
- > https://github.com/didiViking/Conferences\_Talks/blob/main/WeAreDevelopersLive-%20Serverless%20Observability SLO slides.pdf