elastic

Elasticsearch Query Language

ES|QL

slides & demo

David Pilato - @dadoonet

Developer | Evangelist

Elastic and Kibana support a number of query languages

elastic

A brief history of Elasticsearch's analytical capabilities

Elasticsearch 0.9

Facets

Elasticsearch < 0.90

Facet terms-stats

Elasticsearch 1.0

Aggregations

Elasticsearch 2.0

Pipeline aggregations

Elasticsearch 8.11

ES|QL

elastic

ES|QL

- Language
- Engine
- Visualization

elastic

ES|QL

the language

elastic

ES|QL Features

- Unstructured and structured data
- Piped query language
- SQL-like filtering and data manipulation
- Lookups

elastic

ES|QL commands

Source (From, Row)

Filter (Where)

Processing (Eval)

Aggregation (Stats)

TopN (Sort + Limit)

Expansion (Enrich , MV_Exand)

Extraction (Dissect, Grok)

75+ functions:

- 10 aggregate
- 20+ math
- 10+ string
- 7 date-time
- 15 conversion
- 4 conditionals
- 12 multi-value / mv_

elastic

ES|QL

the engine

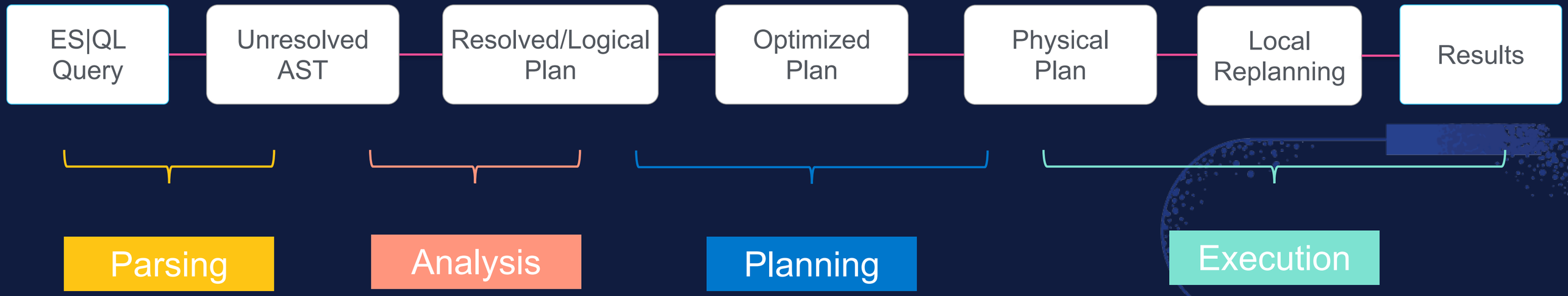
elastic

The new ES|QL execution engine was designed **with performance in mind** — it **operates on blocks** at a time instead of per row, **targets vectorization** and cache locality, and embraces specialization and **multi-threading**. It is a separate component from the existing Elasticsearch aggregation framework with different performance characteristics.

elastic

Query planner

- ✓ Flexible distributed execution
- ✓ Allow multiple roundtrips



elastic

Compute engine

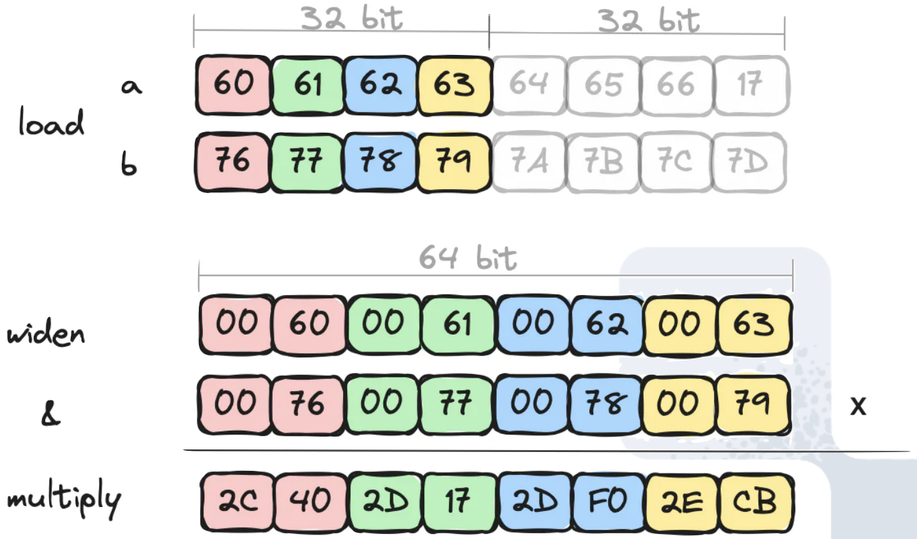
- ✓ Tabular data representation
- ✓ From 1 thread per shard to many
- ✓ Spilling to disk if needed
- ✓ Streaming of data across nodes

elastic

Vectorization

“convert from a scalar implementation, which processes a single pair of operands at a time, to a vector implementation, which processes one operation on multiple pairs of operands at once. “

```
for (i = 0; i < n; i++)
    c[i] = a[i] + b[i];
```

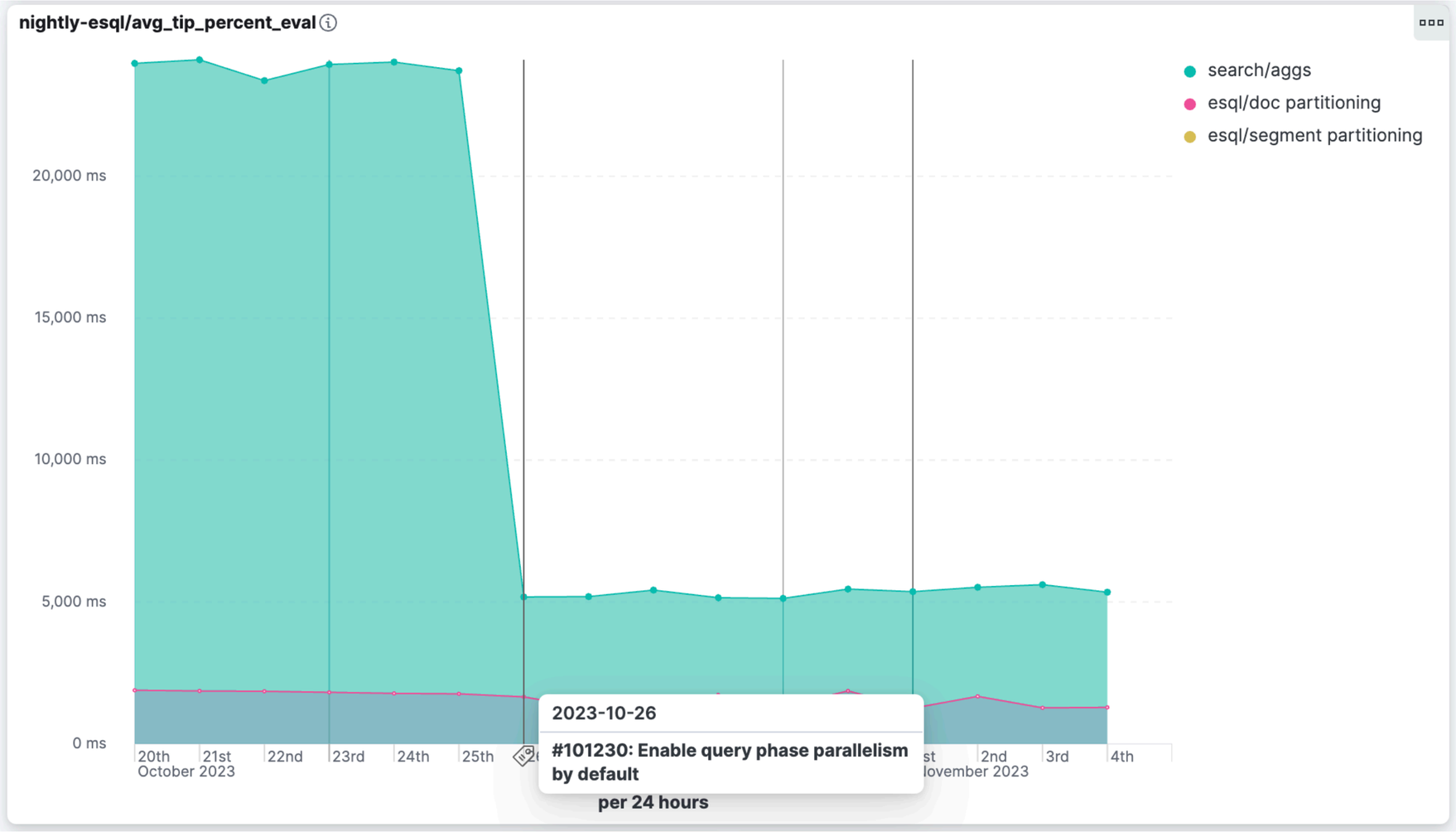


https://en.wikipedia.org/wiki/Automatic_vectorization

elastic

Benchmarks

https://elasticsearch-benchmarks.elastic.co/#tracks/esql/nightly/default/30d



ES|QL

in action

https://github.com/dadoonet/esql-demo

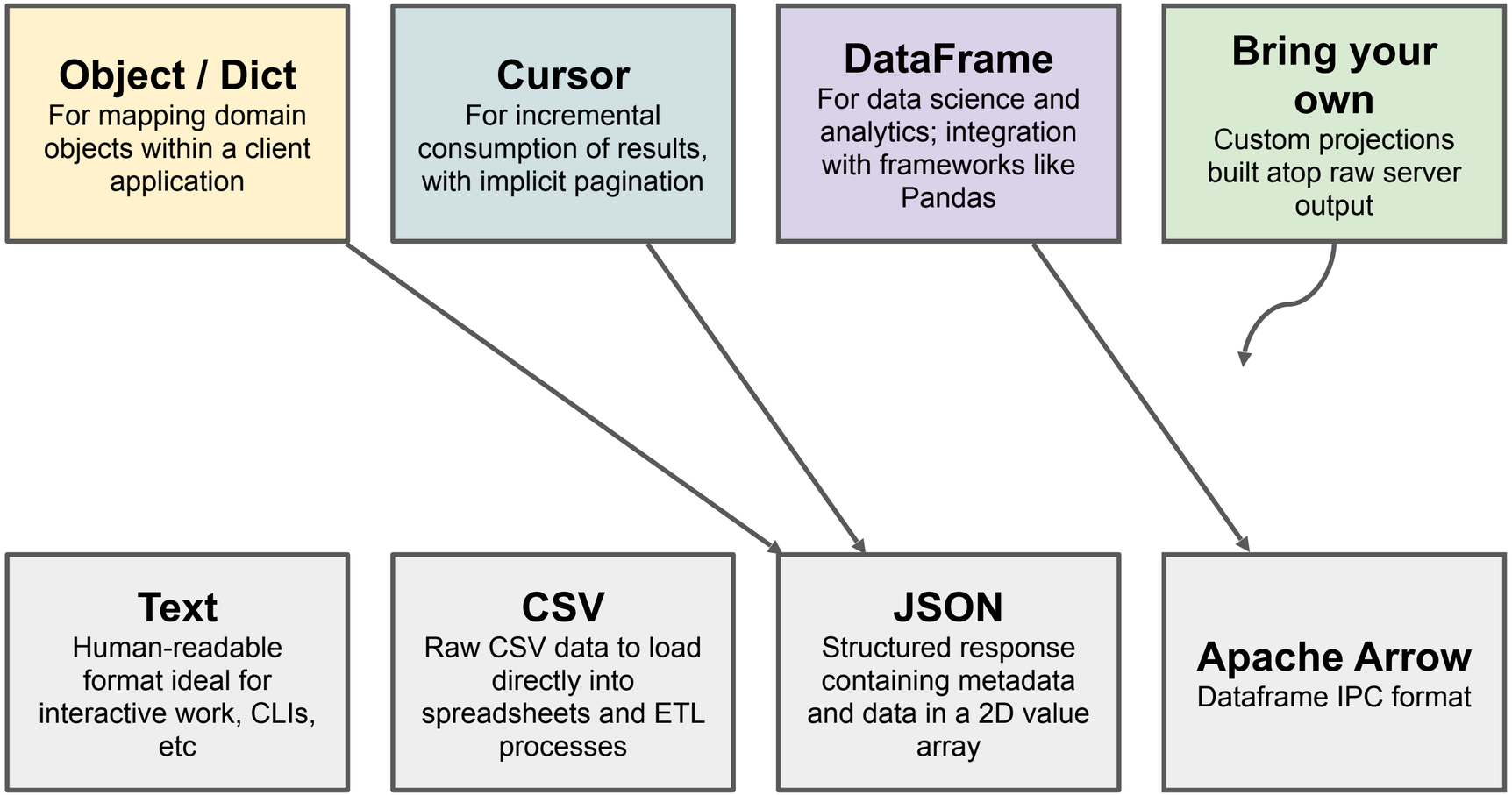
slides & demo

elastic

Ways to consume ES|QL results

PROJECTIONS

Each language client will offer a selection of projections relevant to that language ecosystem.



RESULT DATA

Users can consume raw data directly from the server output in one of several formats.

elastic

Object API

https://github.com/dadoonet/elasticsearch-java-client-demo

```
String query = """
    FROM persons
    | WHERE name == "David"
    | KEEP name
    | LIMIT 1
    """;

Iterable<Person> persons = client.esql()
    .query(ObjectsEsqlAdapter.of(Person.class), query);
for (Person person : persons) {
    assertNull(person.getId());
    assertNotNull(person.getName());
}
```

elastic

ResultSet JDBC API

https://github.com/dadoonet/elasticsearch-java-client-demo

```
String query = """
    FROM persons
    | WHERE name == "David"
    | KEEP name
    | LIMIT 1
    """;

try (ResultSet resultSet = client.esql()
    .query(ResultSetEsqlAdapter.INSTANCE, query)) {
        assertTrue(resultSet.next());
        assertEquals("David", resultSet.getString(1));
}
```

elastic

Named parameters

8.16

```
POST /_query
{
 "query": """
    from logs-*
    | stats x = ?function(?field) by ?breakdownField
    | where x >= ?value
 """,
 "params": [
    {"function" : {"identifier" : "avg"}},
    {"field" : {"identifier" : "network.bytes"}},
    {"breakdownField" : {"identifier" : "agent.name"}},
    {"value": 1000}
 ]
}
```

elastic

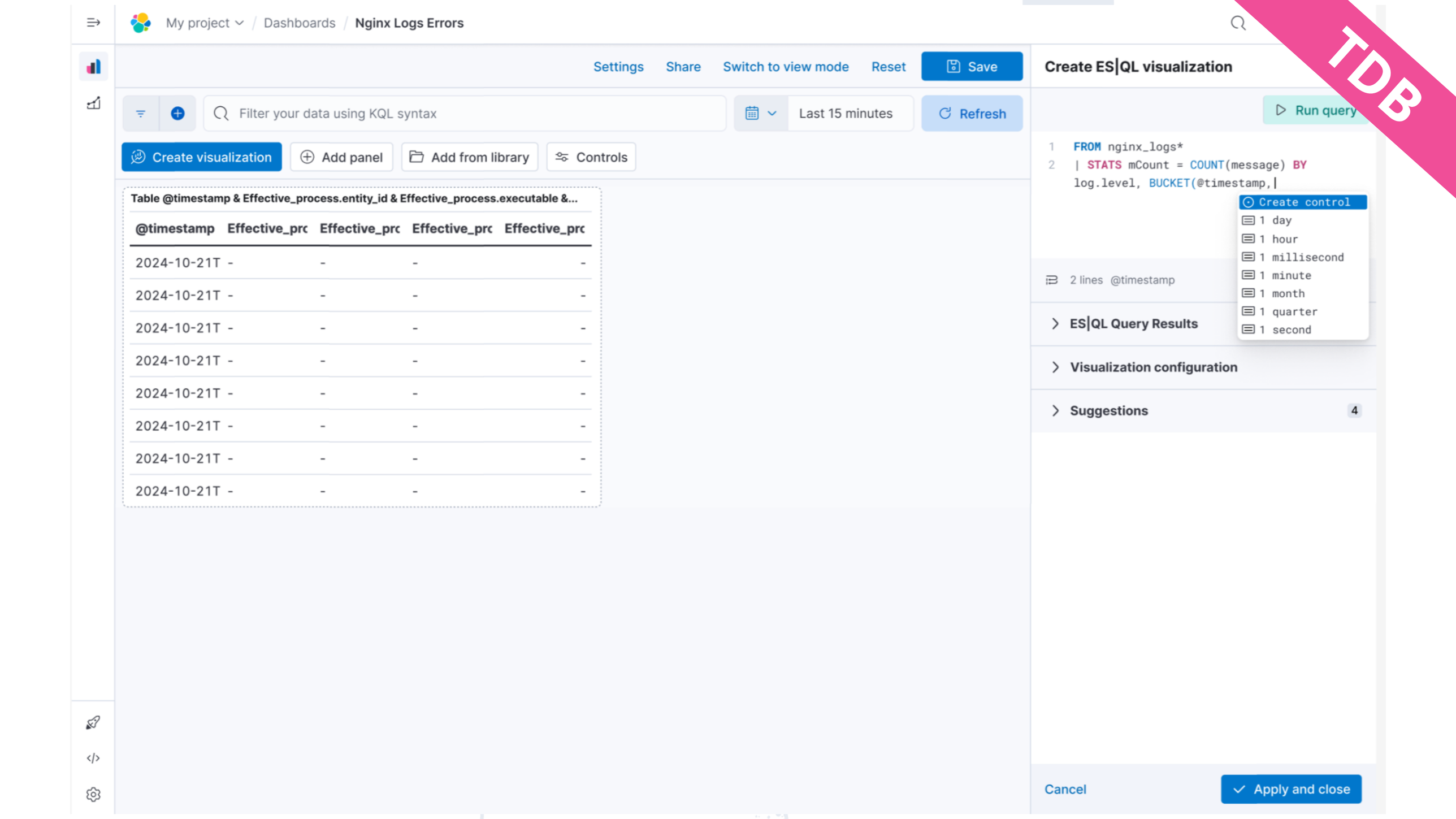
Java API with named parameters

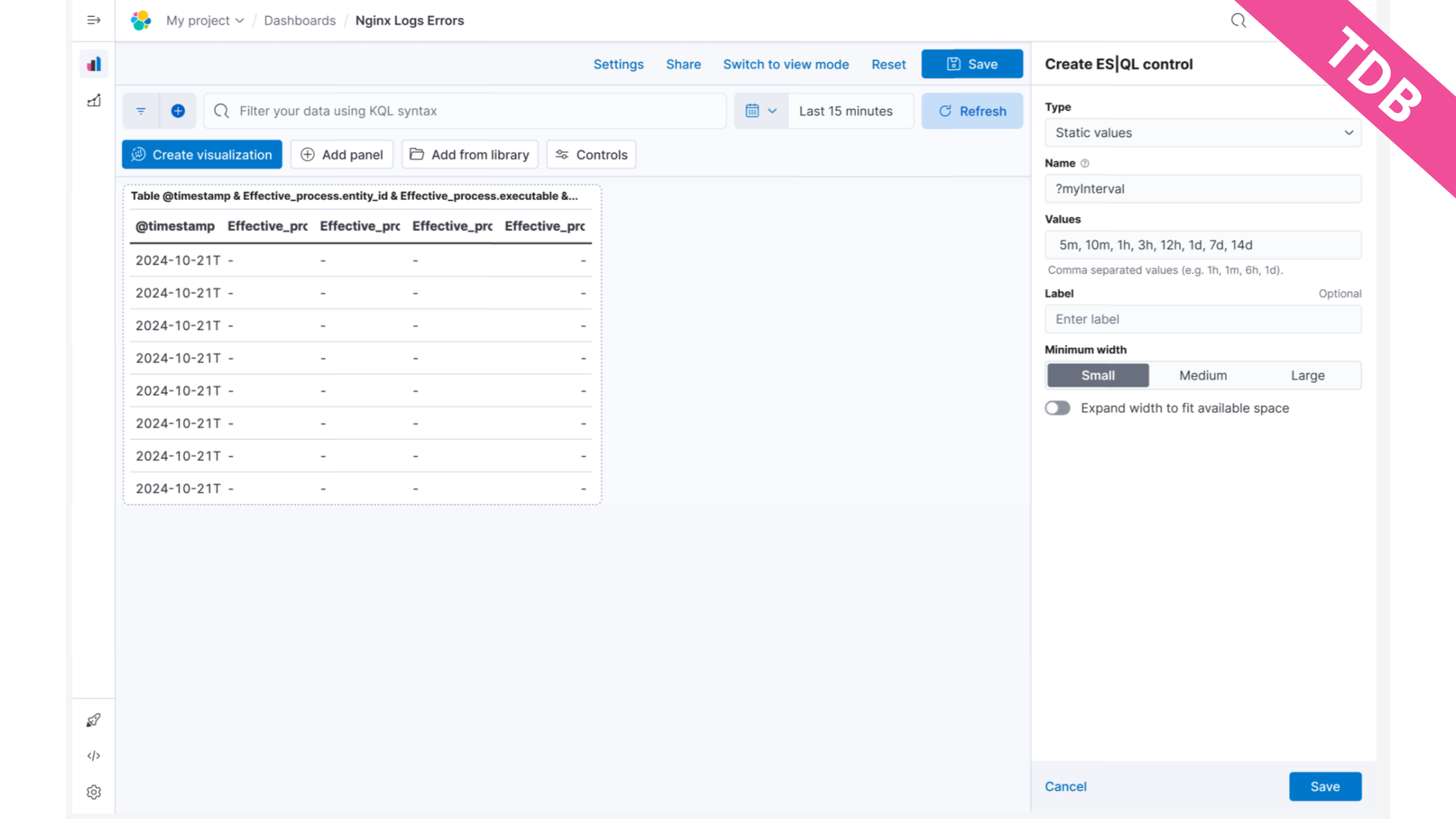
https://github.com/dadoonet/elasticsearch-java-client-demo

```
String query = """
    FROM persons
    | WHERE name == ?name
    | KEEP name
    | LIMIT 1
    """;

Iterable<Person> persons = client.esql()
    .query(ObjectsEsqlAdapter.of(Person.class), query,
       Map.of("name", "David"));
for (Person person : persons) {
    assertNull(person.getId());
    assertNotNull(person.getName());
}
```

elastic


My project / Dashboards / Nginx Logs Errors
Settings
Share
Switch to view mode
Reset
Save
Filter your data using KQL syntax
Last 15 minutes
Refresh
Create visualization
Add panel
Add from library
Controls
Table @timestamp & Effective_process.entity_id & Effective_process.executable &...
@timestamp
Effective_prc
Effective_prc
Effective_prc
Effective_prc
2024-10-21T -
Create ES|QL visualization
Run query
FROM nginx_logs*
| STATS mCount = COUNT(message) BY log.level, BUCKET(@timestamp,
Create control
1 day
1 hour
1 millisecond
1 minute
1 month
1 quarter
1 second
2 lines @timestamp
ES|QL Query Results
Visualization configuration
Suggestions
4
Cancel
Apply and close
TDB


My project
Dashboards
Nginx Logs Errors
Settings
Share
Switch to view mode
Reset
Save
Filter your data using KQL syntax
Last 15 minutes
Refresh
Create visualization
Add panel
Add from library
Controls
Table @timestamp & Effective_process.entity_id & Effective_process.executable &...
@timestamp
Effective_prc
Effective_prc
Effective_prc
Effective_prc
2024-10-21T
Create ES|QL control
Type
Static values
Name
?myInterval
Values
5m, 10m, 1h, 3h, 12h, 1d, 7d, 14d
Comma separated values (e.g. 1h, 1m, 6h, 1d).
Label
Optional
Enter label
Minimum width
Small
Medium
Large
Expand width to fit available space
Cancel
Save
TDB

TDB

Settings Share Switch to view mode Reset

Filter your data using KQL syntax

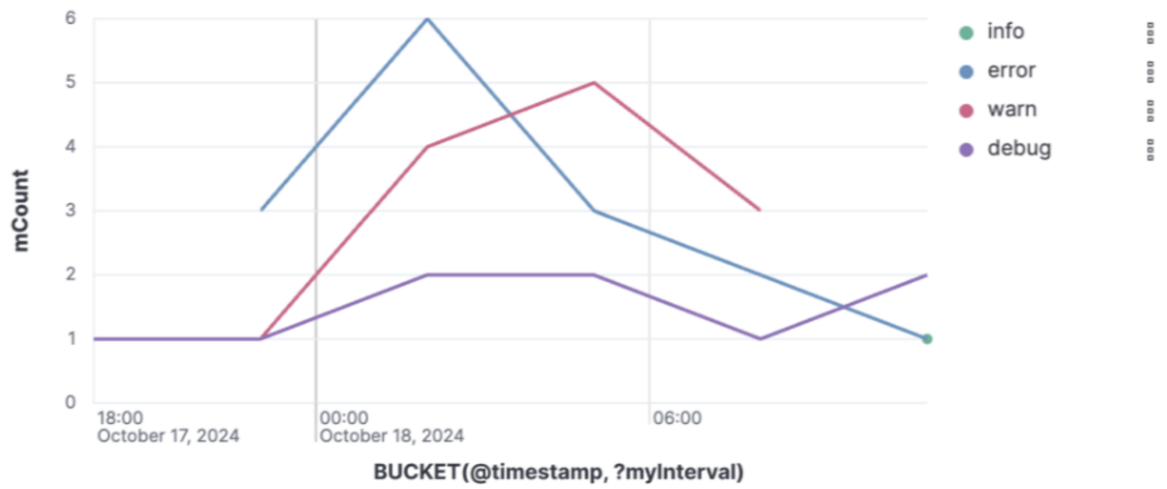
Last 15 minutes

Refresh

Create visualization Add panel Add from library Controls

Interval 1 5m

Messages by log level over time



8.18

Coming next

```
WHERE KQL("bytes>=1024")
```

elastic

JOINS!

TBD

```
INLINESTATS total_visits = COUNT()
```

```
joinType JOIN indexName (AS qualifier)? condition?

joinType: LOOKUP | LEFT | RIGHT | INNER

condition:
  ON identifier == identifier
| USING identifier
```

```
FROM employees
| SORT emp_no
| LOOKUP JOIN languages_lookup ON language_code
| KEEP emp_no, language_name
```

- No need to create an enrich policy
- A drag and drop experience in the UI

elastic

TBD

```
FROM search-movies METADATA _score, _id
| WHERE imdbrating > 7
| FORK [WHERE title:"Shakespeare" | SORT _score DESC | LIMIT 10] // fork1
       [WHERE semantic_title:"Shakespeare" | SORT _score DESC | LIMIT 10] // fork2
       [WHERE plot:"Shakespeare" | SORT _score DESC | LIMIT 10] // fork3
       [WHERE writers:"Shakespeare"| SORT _score DESC | LIMIT 10] // fork4
| KEEP title, semantic_title, _fork, _score, imdbrating, plot, writers
| DROP writers, semantic_title, plot
| RRF
```

_score	_fork	title	imdbrating
0.04918032786885246	[fork0, fork1, fork2]	Shakespeare in Love	7.199999889265137
0.031544957774465976	[fork1, fork3]	Othello	7.800000190734863
0.0315136476426799	[fork1, fork3]	Macbeth	7.5
0.030330882352941176	[fork1, fork3]	Much Ado About Nothing	7.400088095367432
0.016129032258064516	fork3	Julius Caesar	7.400088095367432
0.015873015873015872	fork1	Juliet of the Spirits	7.599999984632568
0.015873015873015872	fork3	Looking for Richard	7.400008895367432
0.015625	fork1	The King's Speech	8.100000381469727
0.015384615384615385	fork1	O Brother, Where Art Thou?	7.800088190734863
0.015151515151515152	fork3	Falstaff - Chimes at Midnight	7.900008895367432
0.014925373134328358	fork3	Maqbool	8.300000198734863
8.014925373134328358	fork1	Only Fools and Horses....	8.899999618530273
8.014785882352941176	fork3	Ran	8.300000198734863
0.014492753623188406	fork1	The Lovers on the Bridge	7.5
8.014285714285714285	fork1	Horatio Hornblower: The Duel	8.199999809265137

elastic

Elasticsearch Query Language

ES|QL

slides & demo

David Pilato - @dadoonet

Developer | Evangelist