How-to: standard RPA execution reporting with Kibana

Introduction: RPA execution logging summary

Execution Logs:

Each time an UIPath robot runs a process, there are at least 2 log messages that are being generated by the robot (unless logging is turned off): Execution Start, and Execution End. If any job reports an error, a specific Error Log Message is issued. These log messages contain details about the execution of the job (process name, job key, start time, status), the agent (robot name, machine ID, user name) as well as any custom fields that have been added by the RPA developer. If the robot is connected to it, the Robot sends the logs to Orchestrator, which adds a few additional fields and sends them to permanent storage (ElasticSearch and/or SQL Server).

Sample Job End log message:

⊙ @timestamp	⊕ ⊕ ▥ ✳	February 4th 2018, 16:02:32.949
? Component	⊕ ⊕ ▥ ✳ ⚠	-
t Source	⊕ ⊕ ▥ ✳	Robot
t _id	⊕ ⊕ ▥ ✳	AWFhIMtrcoOrq61-Wy2U
t _index	⊕ ⊕ ▥ ✳	fantastic-2018.02
# _score	⊕ ⊕ ▥ ✳	-
t _type	⊕ ⊕ ▥ ✳	logEvent
t fileName	⊕ ⊕ ▥ ✳	Main
t fingerprint	⊕ ⊕ ▥ ✳	646969de-f94e-499f-88a9-3317670b1493
t jobId	⊕ ⊕ ▥ ✳	747a3a1e-2c80-4ea6-b4ed-8b533c5dcd54
t level	⊕ ⊕ ▥ ✳	Info
# levelOrdinal	⊕ ⊕ ▥ ✳	2
t machineName	⊕ ⊕ ▥ ✳	DUMIHAI
t message	⊕ ⊕ ▥ ✳	ExcelDemo execution ended
t processName	⊕ ⊕ ▥ ✳	ExcelDemo
t processVersion	⊕ ⊕ ▥ ✳	1.0.6608.38633
t rawMessage	⊕ ⊕ ▥ ✳	(see below)

```
{
    "message": "ExcelDemo execution ended",
    "level": "Information",
    "timeStamp": "2018-02-04T14:02:32.9490039+00:00",
    "fingerprint": "646969de-f94e-499f-88a9-3317670b1493",
    "windowsIdentity": "DUMIHAI\\mihai.dunareanu",
    "machineName": "DUMIHAI",
    "processName": "ExcelDemo",
    "processVersion": "1.0.6608.38633",
    "fileName": "Main",
    "jobId": "747a3a1e-2c80-4ea6-b4ed-8b533c5dcd54",
    "robotName": "MDOrchPractice",
    "totalExecutionTimeInSeconds": 361,
    "totalExecutionTime": "00:06:01",
    "wbBusinessProcessName": "ExcelDemo"
}
```

t robotName	⊕ ⊕ ▥ ✳	MDOrchPractice
⊙ timeStamp	⊕ ⊕ ▥ ✳	February 4th 2018, 16:02:32.949
t totalExecutionTime	⊕ ⊕ ▥ ✳	00:06:01
# totalExecutionTimeInSeconds	⊕ ⊕ ▥ ✳	361
t wbBusinessProcessName	⊕ ⊕ ▥ ✳	ExcelDemo

Execution Log Types summary:

Default Logs:

- **Execution Start** is generated every time a process is started

- **Execution End** is generated every time a process is finalized
- **Transaction Start** is generated every time a transaction within a process is started
- **Transaction End** is generated every time a transaction within a process is finalized
- **Error Log** is generated every time the execution encounters an error and stops
- **...**

User-Defined Logs are generated when the execution triggers a "Log Message" activity or a "Write Line" activity, as designed in UIPath Studio.

The UIPath execution logs are formatted as JSON objects. This guide will use examples using only Default Logs, which should always be available in ElasticSearch/Kibana and do not require any specific custom logging to function.

ElasticSearch, Indexing and Kibana

Elasticsearch is a NOSQL, distributed full text database and search engine. This database is document-based rather than using classic database tables, and does not require a specific schema for the data being stored. This flexibility allows UIPath to send log messages (which, in this context, are individual documents in the database) without specifying in advance what fields these documents contain. As a result, the ElasticSearch collections of documents (named Index in ElasticSearch) that may have a very different structure but can still be efficiently retrieved and queried.

Indexes: UIPath execution logs are stored in ElasticSearch as monthly Indices (collections) which have the following naming convention: one index is generated every month for every Orchestrator tenant. For example, if you are using a Default tenant and a Finance tenant, at the beginning of February 2018 two new indices will be created: Default-2018.02, and Finance-2018.02.

Index Patterns are used for querying documents from specific indexes. They can use wildcard characters like *. A few examples:

- Default-* will retrieve documents from all the indices beginning with Default- (in other words, it will retrieve all execution logs that have been generated within the Default tenant, for all history)
- *2018* will retrieve documents from all tenants from 2018

Kibana is an open source analytics and visualization platform designed to work with ElasticSearch. You use Kibana to search, view, and interact with data stored in Elasticsearch indices.
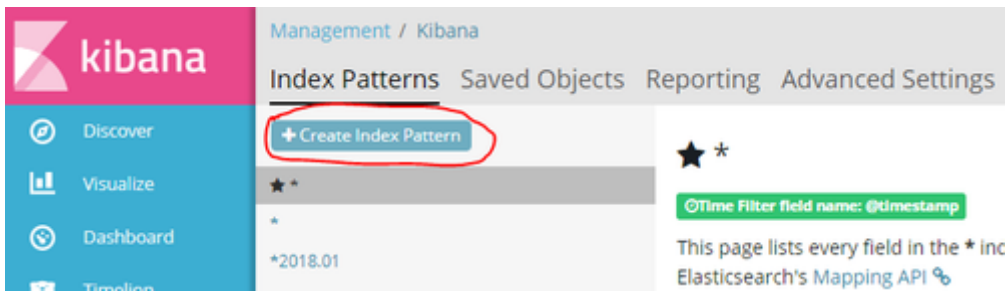
Creating a Kibana visualization

Step 0: create an Index Pattern if you have not done it already

Launch Kibana in your web browser. If you have not already created an index pattern for your tenant, create one by following the next steps:
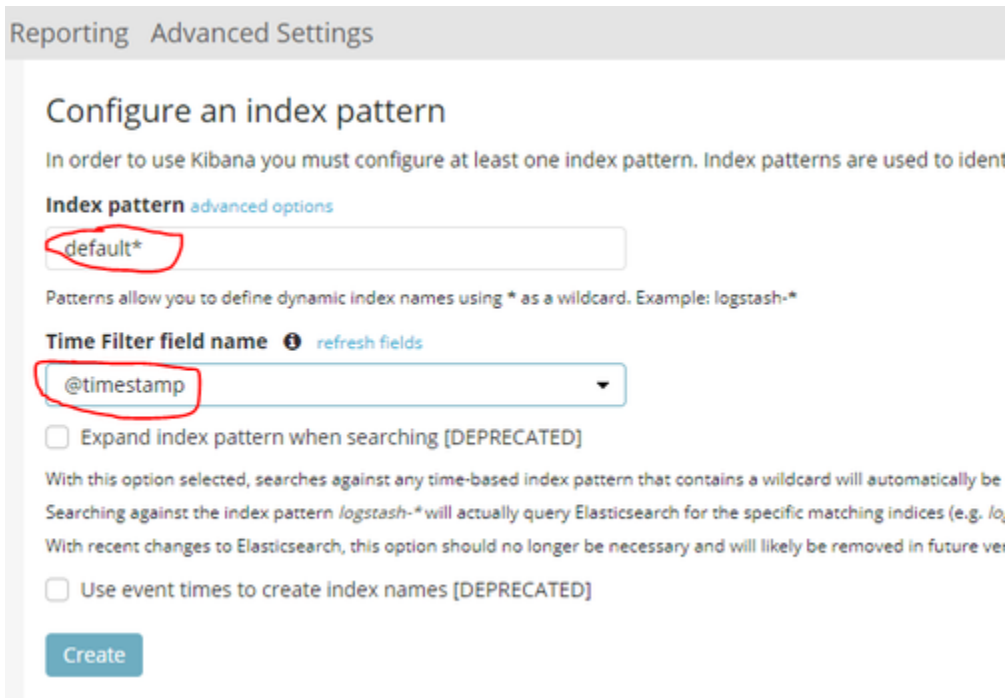
- Select the "Management" tab on the left and click on Index Patterns
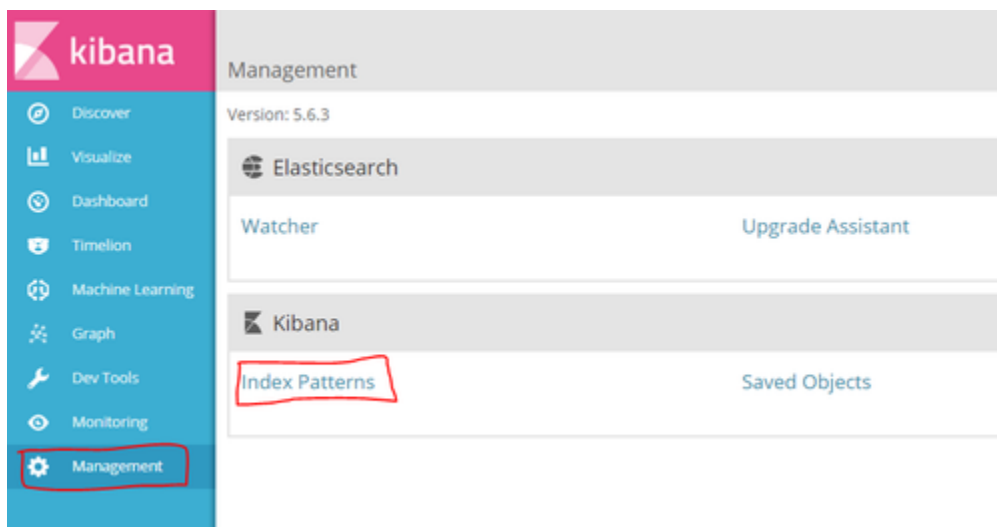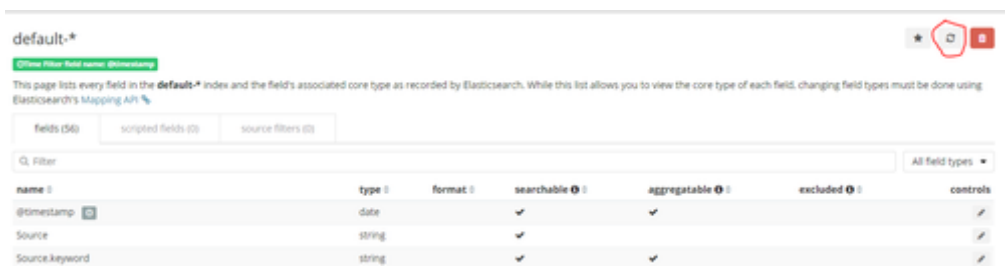


- Click on the Create Index Pattern button

- Input a suitable name for the index pattern (default-* if you only have the Default tenant), @timestamp as the time filter field name and click on Create

Configure an index pattern

In order to use Kibana you must configure at least one index pattern. Index patterns are used to ident

Index pattern advanced options

default*

Patterns allow you to define dynamic index names using * as a wildcard. Example: logstash-*

Time Filter field name ⓘ refresh fields

@timestamp ▼

☐ Expand index pattern when searching [DEPRECATED]

With this option selected, searches against any time-based index pattern that contains a wildcard will automatically be

Searching against the index pattern *logstash-** will actually query Elasticsearch for the specific matching indices (e.g. *lo*

With recent changes to Elasticsearch, this option should no longer be necessary and will likely be removed in future ver

☐ Use event times to create index names [DEPRECATED]

Create

- Refresh the index pattern fields by clicking the Refresh button on the top right. This action can also be useful when an index already exists but you cannot find some fields in the Discover or Visualization panes.


- Click on the Discover tab, select the index pattern and set the time frame to the required valuein the top right corner. Check that you do have logs in the central area of the page.
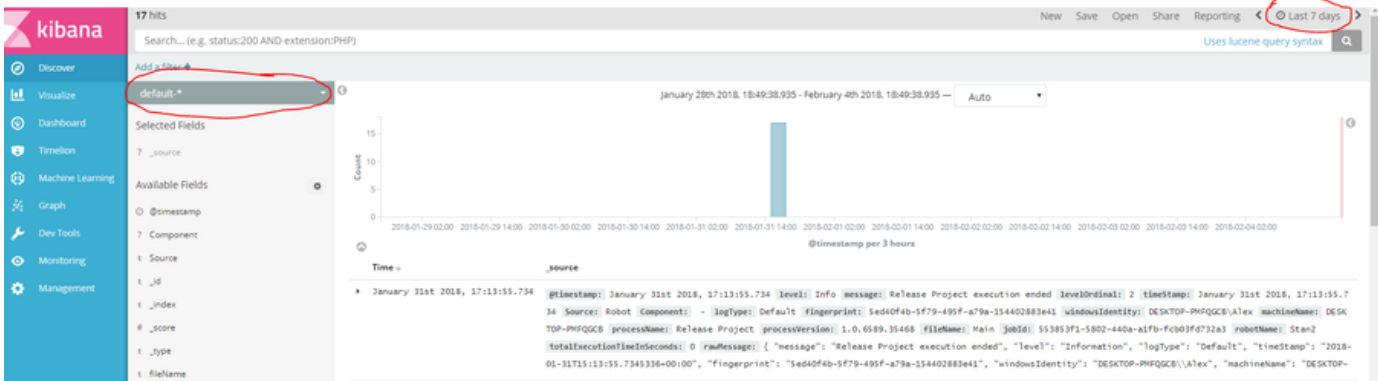
Step 1: Create a new visualization

- Click on Visualize in the left banner. If a pre-existing visualization appears, click again on Visualize to display the list of visualizations, search bar and the option to create a new one (+ button at the right of the search bar)
- Click on the **+** sign to the right of the search bar in the center to create a new visualization
- Select the visualization type by clicking on it. In this example, we will use the Vertical Bar type
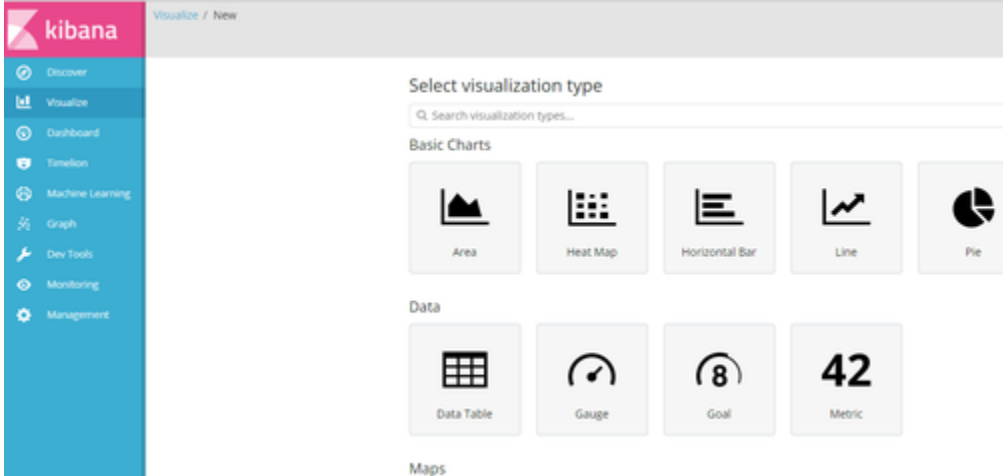
- Select the index pattern to use with this visualization. A new visualization will be created and displayed.
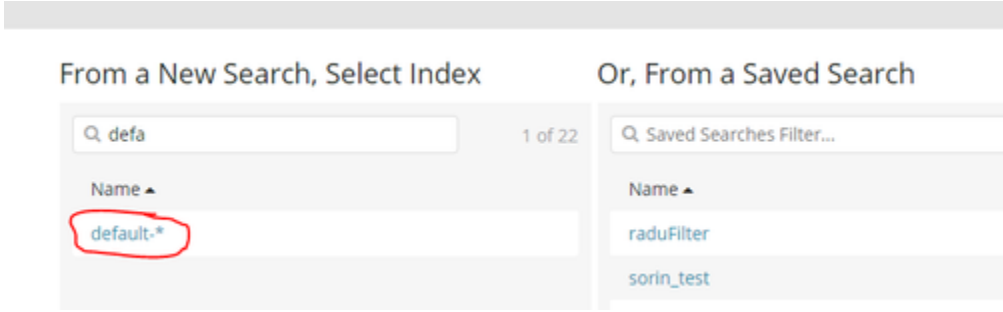
Step 3: Use a query to filter the logs that you want to report on

In this exercise, the objective is to report on the finalized process runs (also named jobs). To do that, you use a query that restricts the logs that ElasticSearch returns to the ones that mark the end of the execution (the Execution End logs described above). One easy way to filter the logs returned by the search is to specify a field that only exists in that log message. In our situation, the field **totalExecutionTime** only appears in the Execution End logs, so we can use it to filter what ElasticSearch returns.

Upon receiving a query, ElasticSearch uses the specific instructions to return a payload, which is a collection of documents (logs, in our case)

from the database, for which the query conditions are true. We are using the presence of the field **totalExecutionTime** for that purpose. All next steps will only apply to the payload returned by the query.

- Insert in the query window the following expression: *_exists_:"totalExecutionTime"* and press Enter.
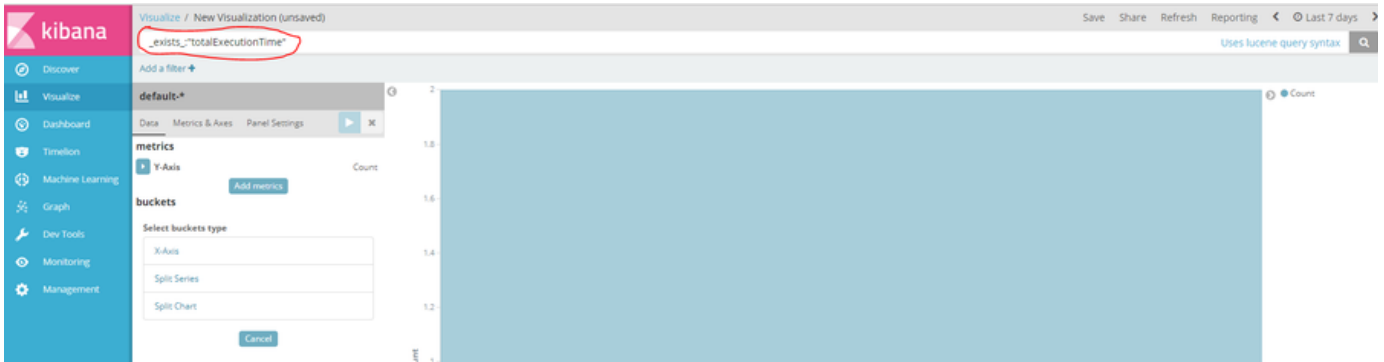
Step 4: Select and customize the metric(s) that you want to display

Kibana visualizations usually display a metric (a numeric value), broken down by various dimensions. The simplest metric that we are going to use here is the number of log messages that are being returned by the query from the previous step. As the payload contains exactly one Execution End log message for each finished process, the metric will simply be equal to the total number of process runs during the selected time frame.

- click on the Y-Axis arrow and make sure "Count" is selected under the Aggregation drop-down. Optionally, create a custom label for the metric name (here, "Process Runs")
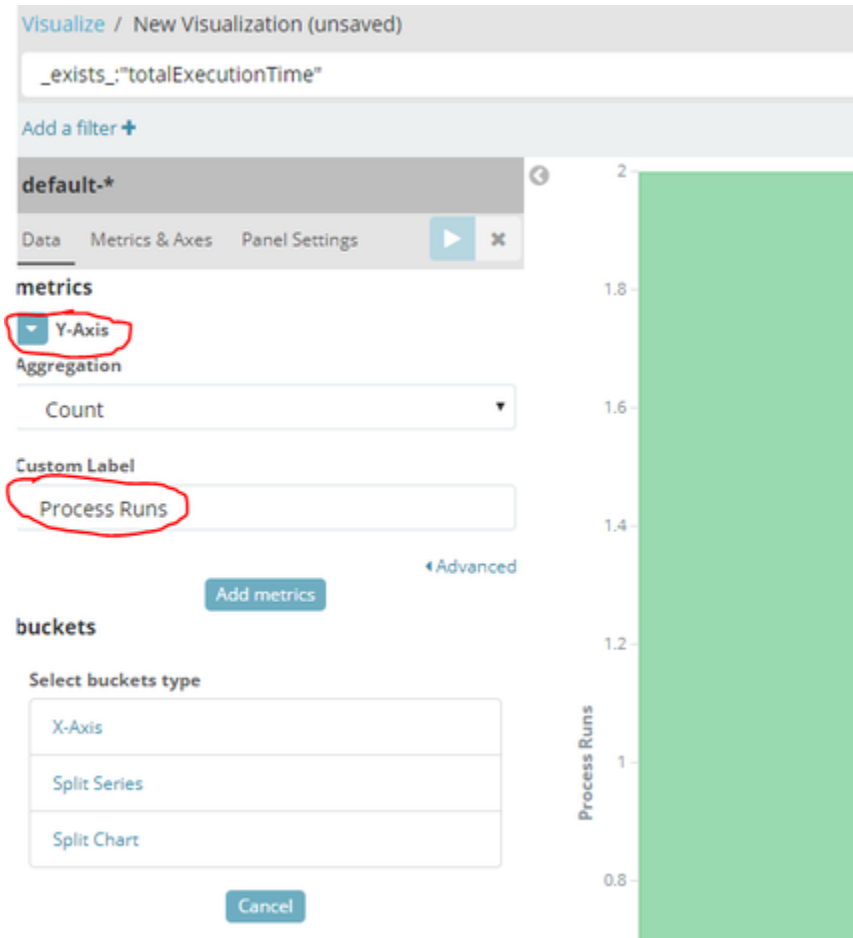
Step 5: Select and customize the dimensions(s) by which you want to break down the selected metric

In this situation, we want to display the number of runs for each separate process for the selected time frame. As a result, we want to use processName.keyword as the dimension name.

- Select the first dimension (bucket) by which to summarize the selected metric. To do that, click on the X-Axis label under the buckets section on the left
- Under X-Axis, select **Terms** in the Aggregation drop-down, **processName.keyword** under Field. Leave Order By as **metric: Process Runs** and increase the Size field to 10 or 15. Create a custom label for this dimension (for example Process or Process Name). Then click on the "Apply Changes" button (white triangle pointing to the right, on blue background, at the top of the section) to display the results.


Step 6: Finalize and save the visualization

At this point, the visualization should look like the one below. Click on the "Save" button in the top right ribbon, give it a name and click on "Save".

ATTENTION: the visualization name is also the title that is displayed at the top left in dashboards. Try to give it a clear and self-explanatory name like "Process Runs by Process Name" for this example

Note: you can change the color of the bars by clicking on the colored dots in the visualization legends
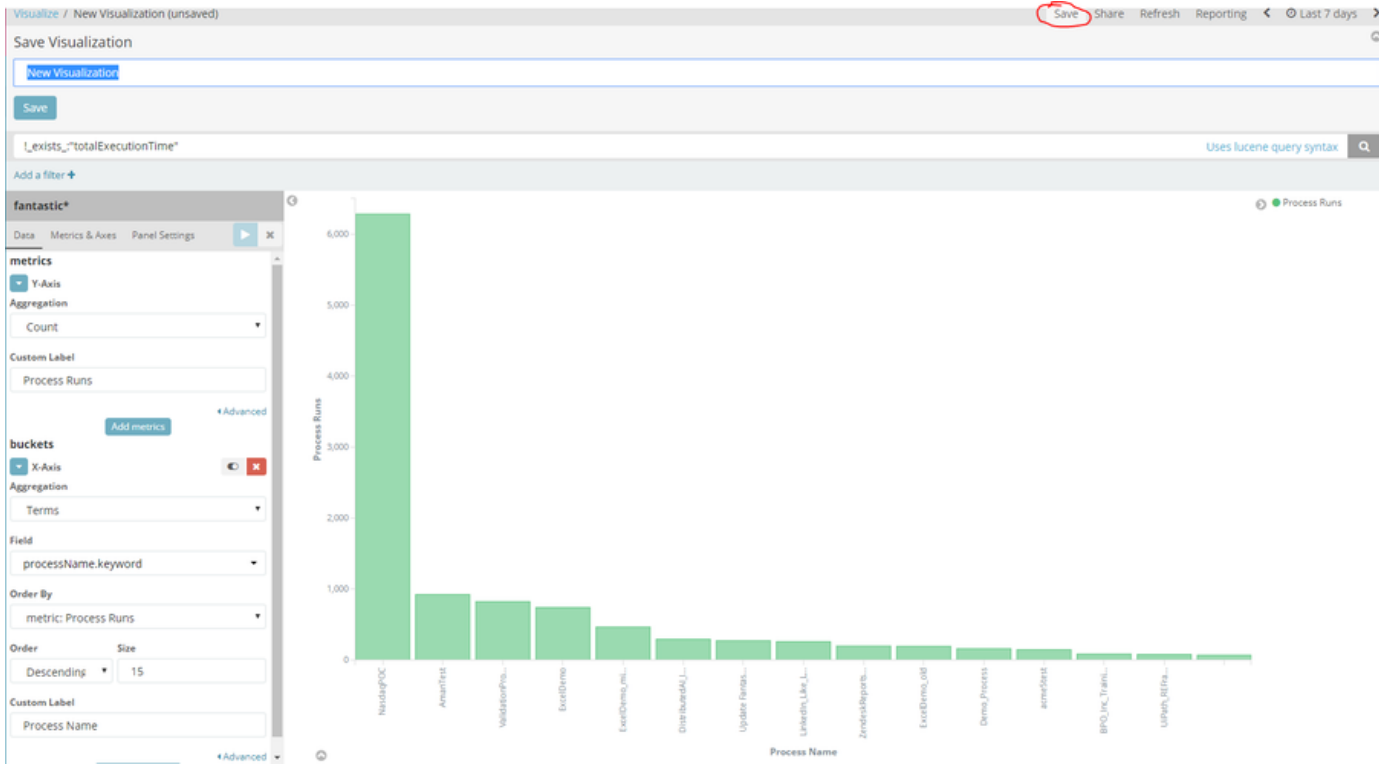
Optional Step 7: Duplicate the saved visualization to create a different one

Once you have created a visualization, you can easily duplicate it to display other things. In this example, we will use the process runs visualization to create a similar one that displays the total runtime for each robot name. You only need to make changes in the visualization settings part of the left, and save it with a new name.

- Under metrics, Y-Axis, change the Aggregation to Sum and the Field to totalExecutionTimeinSeconds
- Change the Custom Label for the metric to Total Exec Time
- in the buckets area, leave Aggregation as Terms and change the Field to robotName.keyword. Leave the Size field at 15 and change the dimension Custom Name to Robot Name
- Press the "Apply Changes" button at the top right of the visualization setup area
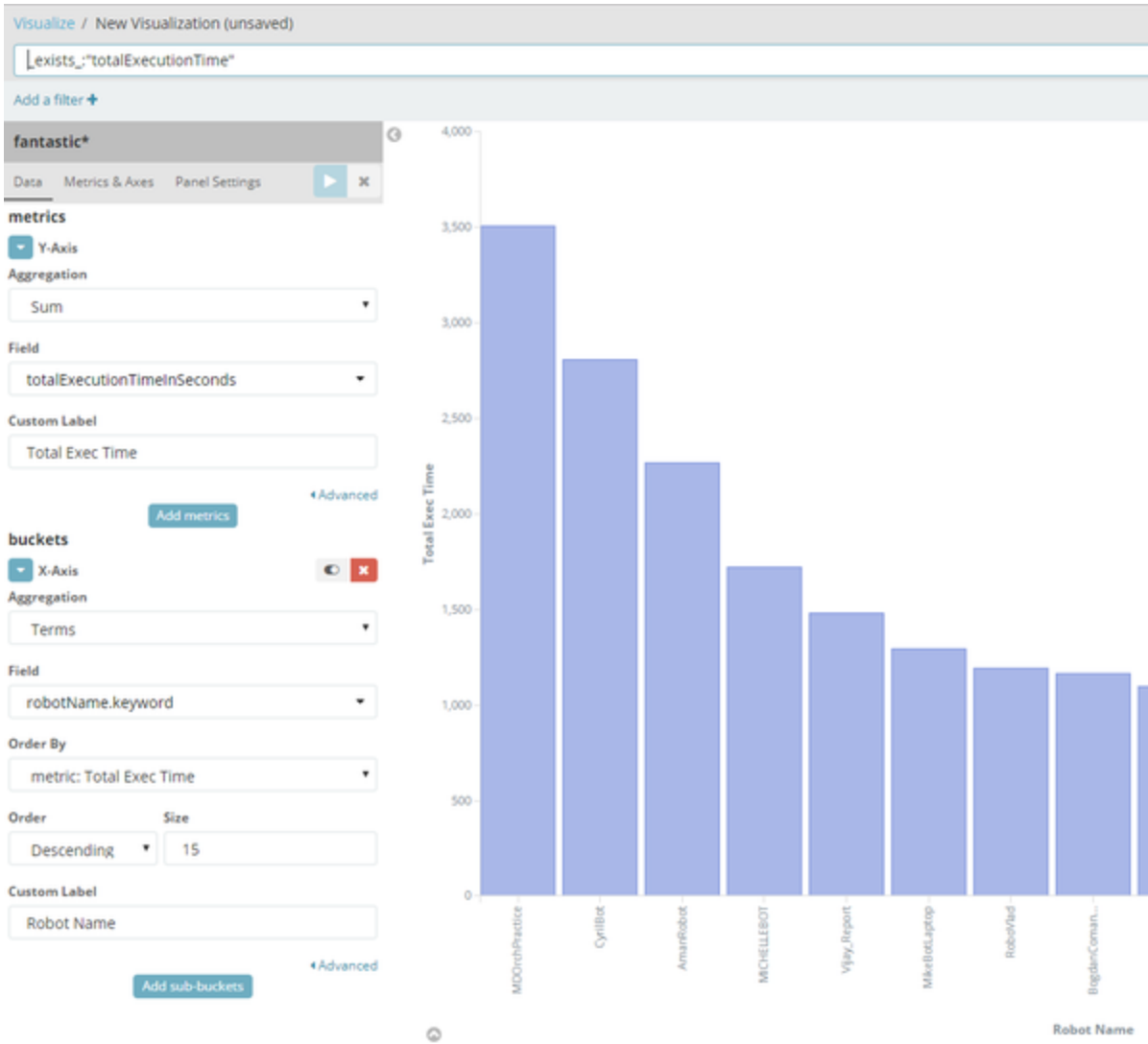
- To save the new visualization without overwriting the pre-existing one, click "Save" in the top right area of the browser, insert a new name and make sure you check the "Save as a new visualization" box before clicking the Save button.

Optional Step 8: Add a secondary dimension to the chart

Sometimes it can be useful to add another dimension to a bar chart. To do that, do the following in the visualization setup area (on the left of the chart):

- Click on the "Add sub-buckets" button at the bottom of the buckets area
- Select Split Series. A new dimension area appears under the existing one. In the new area, do the following:
- Select Terms in the Sub Aggregation field
- Select processName.keyword as the field and leave the Order By metric to be equal to metric: TotalExecTime
- Change the Size to 3 (this will only display the top 3 processes ran by every robot)
- Type "Process" in the Custom Label area of the secondary dimension.
- Click the "Apply Changes" button at the top and save the visualization.
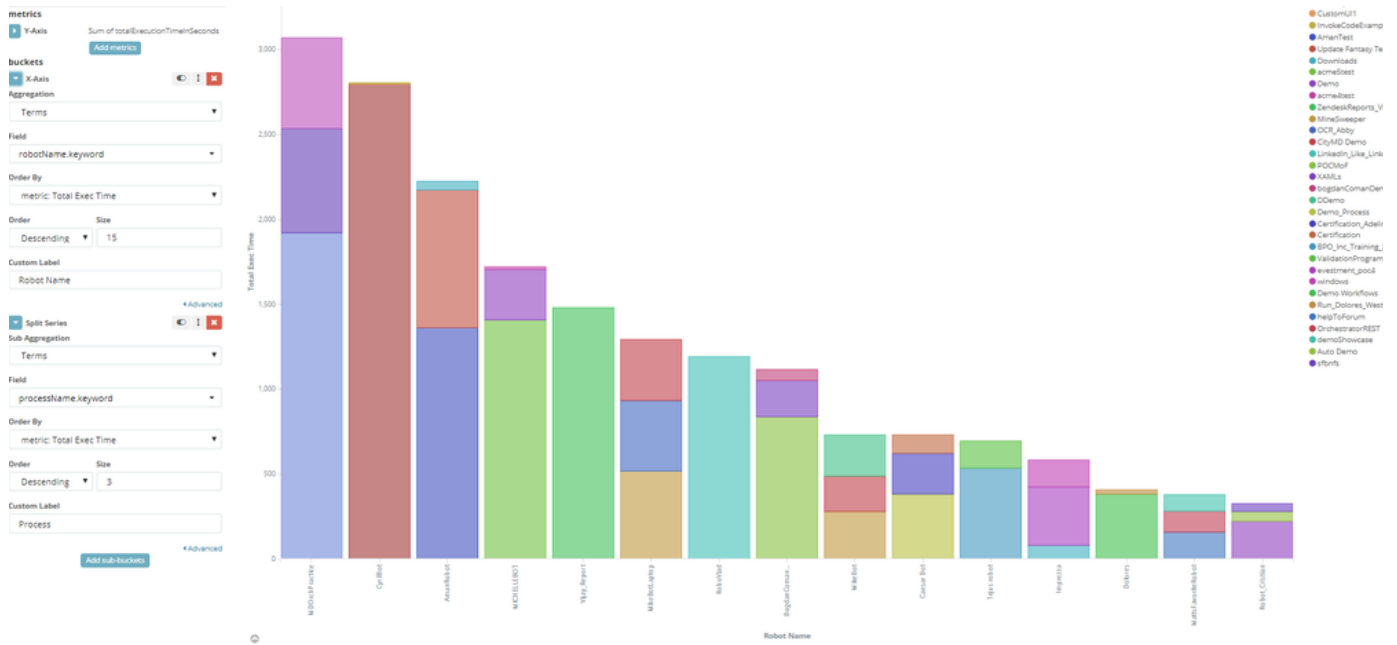
NOTE: More examples of useful RPA visualizations in the appendix of this document.

Creating a Kibana Dashboard

Once you have a set of related Kibana visualizations, it makes sense to put them together in dashboards that share the selected time frame and possibly other filters. The most useful functionality in a dashboard is clicking on a certain element of a visualization. That element is usually identifying a certain member of a dimension (for example, the Excel process), and following that click, all other visualizations in the dashboard will be only showing information pertaining to that element. As a result, you can use clicks on visualizations to filter the dashboard content on-the-fly. If you click on an element that has a combination of dimensions (for example, the Excel Process when it was run by Robot A, like in the example above), Kibana will ask you to validate which of the filters you actually want to apply (or whether you want both)

Step 1: Create a blank dashboard

- Click on Dashboard in the leftmost area of the browser. If an existing Dashboard is displayed, click again until you can see the existing dashboards list and the search bar
- Click on the **+** sign to the right of the search bar in the center to create a new Dashboard

Step 2: Add visualizations to the dashboard and arrange them
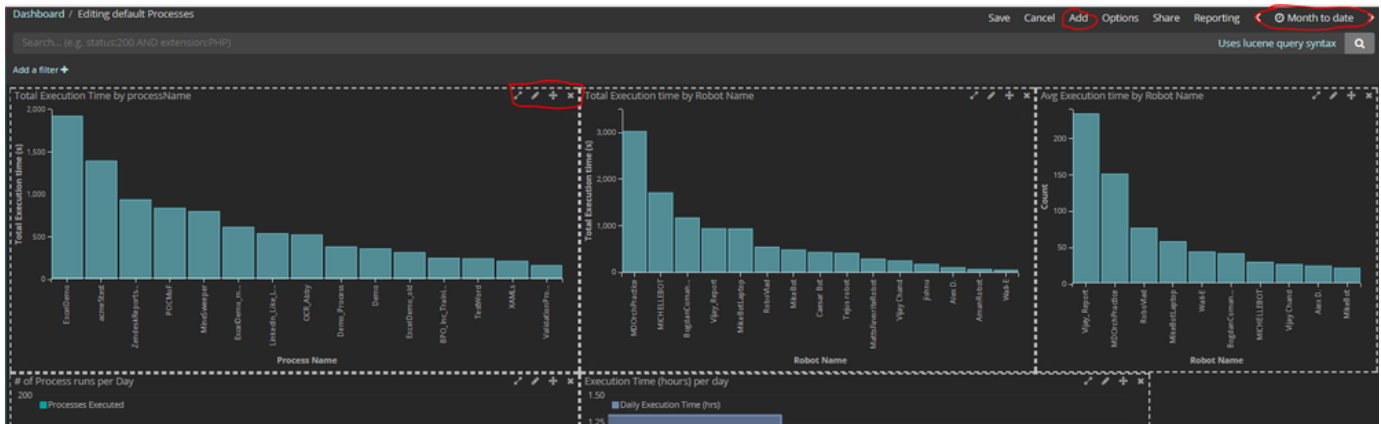
- Click on the "Add" button in the top right area of the Kibana interface, browse for the visualizations you want to display and click on each one of them ONCE to add them to the dashboard
- Use the controls in the top left corner of each visualization to move them around in the dashboard
- Resize visualizations as needed by dragging the margins and corners

Step 3: Finalize the dashboard and save it

- Click in the Options button in the top right banner to switch between light and dark theme
- Set the Dashboard time Frame that will apply to all of the dashboards (regardless which time frame they had individually when they were saved)
- You have the option to save the time frame along with the dashboard definition
- Give the dashboard a meaningful name and save it

Appendix A: Useful fields for extracting information from the default logs:

Fields to be used for queries:

- To report on finalized queue item transactions, select only the Transaction End logs by using the query ***_exists_:"transactionExecution Time"***
- To report on finished processes, select only the Execution End logs by using the query ***_exists_:"totalExecutionTime"***
- To report on error logs, select only Error-level logs by using the query ***level: "Error"***
- Use any custom fields for selecting process-specific logs

Default Fields to be used for visualization metrics (numeric):

- Just counting the logs retrieved provides the number of jobs run / transactions ran / errors raised when using the correct query as described above
- **totalExecutionTime** provides the duration of jobs, in seconds. Possible aggregations: Sum, Average...
- **transactionExecutionTime** provides the duration of queue item transactions. Possible aggregations: Sum, Average...

Default Fields to be used for visualization buckets (dimensions). Most used aggregation: Terms.

- processName.keyword
- robotName.keyword
- machineName.keyword
- windowsIdentity.keyword
- fileName.keyword
- Exception-related fields for queue item transactions:
    - transactionStatus
    - processingExceptionType
    - queueItemReviewStatus
    - queueItemPriority

- The timestamp field can be used for time-related charts.
- There are also a few free text fields like **message, processingExceptionReason** etc. that can be displayed in a Data Table visualization in Kibana as text. NOTE: free text fields longer than 256 bytes are not displayed in Kibana unless a specific mapping is applied to de index.

Appendix B: Other examples of Useful Visualizations

Total Execution time in seconds, by robot name

- Query: _exists_:"totalExecutionTime"
- Visualization Type: vertical bar chart
- Metric: totalExecutionTimeInSeconds
- Metric Aggregation: Sum
- Dimensions: robotName.keyword
- Metric Aggregation: Terms

metrics

▾ Y-Axis

Aggregation

Sum ▾

Field

totalExecutionTimeInSeconds ▾

Custom Label

Total Execution time (s)

◀ Advanced

[Add metrics]

buckets

▾ X-Axis ◐ ✕

Aggregation

Terms ▾

Field

robotName.keyword ▾

Order By

metric: Total Execution time (s) ▾

Order   **Size**

Descending ▾	15

Custom Label

Robot Name

Average Execution time in seconds, by robot name

- Query: _exists_:"totalExecutionTime"
- Visualization Type: vertical bar chart
- Metric: totalExecutionTimeInSeconds
- Metric Aggregation: Average
- Dimensions: robotName.keyword
- Metric Aggregation: Terms

metrics

▼ Y-Axis

Aggregation

Average ▼

Field

totalExecutionTimeInSeconds ▼

Custom Label

Average execution time (s)

◄ Advanced

Add metrics

buckets

▼ X-Axis   ◐ ✕

Aggregation

Terms ▼

Field

robotName.keyword ▼

Order By

metric: Average execution time (s) ▼

Order   **Size**

Descending ▼		10

Custom Label

Robot Name

**Number of errors raised, by process name**

- Query: *level: "Error"*
- Visualization Type: vertical bar chart
- Metric Aggregation: Count
- Dimensions: processName.keyword
- Metric Aggregation: Terms

Visualize / Errors Raised by processName

level: "Error"

Add a filter ✚

fantastic-*

Data   Metrics & Axes   Panel Settings   ▶   ✖

metrics

🔽 Y-Axis

Aggregation

Count ▾

Custom Label

◀Advanced

Add metrics

buckets

🔽 X-Axis   ◐ ✖

Aggregation

Terms ▾

Field

processName.keyword ▾

Order By

metric: Count ▾

Order   **Size**

Descending ▾   15

Custom Label

Process Name

◀Advanced

Add sub-buckets

Errors raised details table

- Query: *level: "Error"*
- Visualization Type: Data Table
- Metric Aggregation: Count
- Dimensions: message.keyword
- Metric Aggregation: Terms

Visualize / Errors detail

level: "Error"

Add a filter +

fantastic-*

| Data | Options |

metrics

Metric

Aggregation

Count

Custom Label

# of occurrences

‹ Advanced

Add metrics

buckets

Split Rows

Aggregation

Terms

Field

message.keyword

Order By

metric: # of occurrences

Order   Size

Descending   100

Custom Label

Message

‹ Advanced

Add sub-buckets

Message ⇕

Transaction Number 1. Work Block Task1, Fram

Transaction Number 2. Work Block Task1, Fram

Transaction Number 3. Work Block Task1, Fram

Transaction Number 4. Work Block Task1, Fram

Transaction Number 5. Work Block Task1, Fram

Transaction Number 6. Work Block Task1, Fram

Business rule exception.Exception of type 'UiPa

TransactionNumber 1, RetryNumber 0. Work B

TransactionNumber 2, RetryNumber 0. Work B

TransactionNumber 3, RetryNumber 0. Work B

TransactionNumber 4, RetryNumber 0. Work B

TransactionNumber 5, RetryNumber 0. Work B

TransactionNumber 6, RetryNumber 0. Work B

Invoke Extract Information workflow: VisualBas

Invoke SAP logon workflow: Click 'GuiButton bt

Private: Write line : Input string was not in a cor

Send SMTP Mail Message : The process cannot

System exception.The browser control is busy I

TransactionNumber 1, RetryNumber 0. Work B

TransactionNumber 2, RetryNumber 0. Work B

Export: Raw ⬇   Formatted ⬇

Appendix C: Timelion Visualizations

Timelion visualizations are specific visualizations that display the value of one or more metrics over time (the X-axis is always going to be a time). They have a specific look and feel and can be used to display for example the utilization rate of robots, in hours per day, like in the example below.

They have 2 parameters:

- the Interval of time over which the metric is aggregated - basically one value of the metric is calculated for every interval. In the example below, it is one day.
- The Timelion Expression which defines:
  - the query used to filter the required logs
  - the index pattern used to query
  - the metric (here, executionTimeInSeconds) and aggregation (here, sum) used to calculate the daily value, chained with any numeric adjustments (here, divided by 3600 to go from seconds to hours)
  - Formatting parameters for the visualization
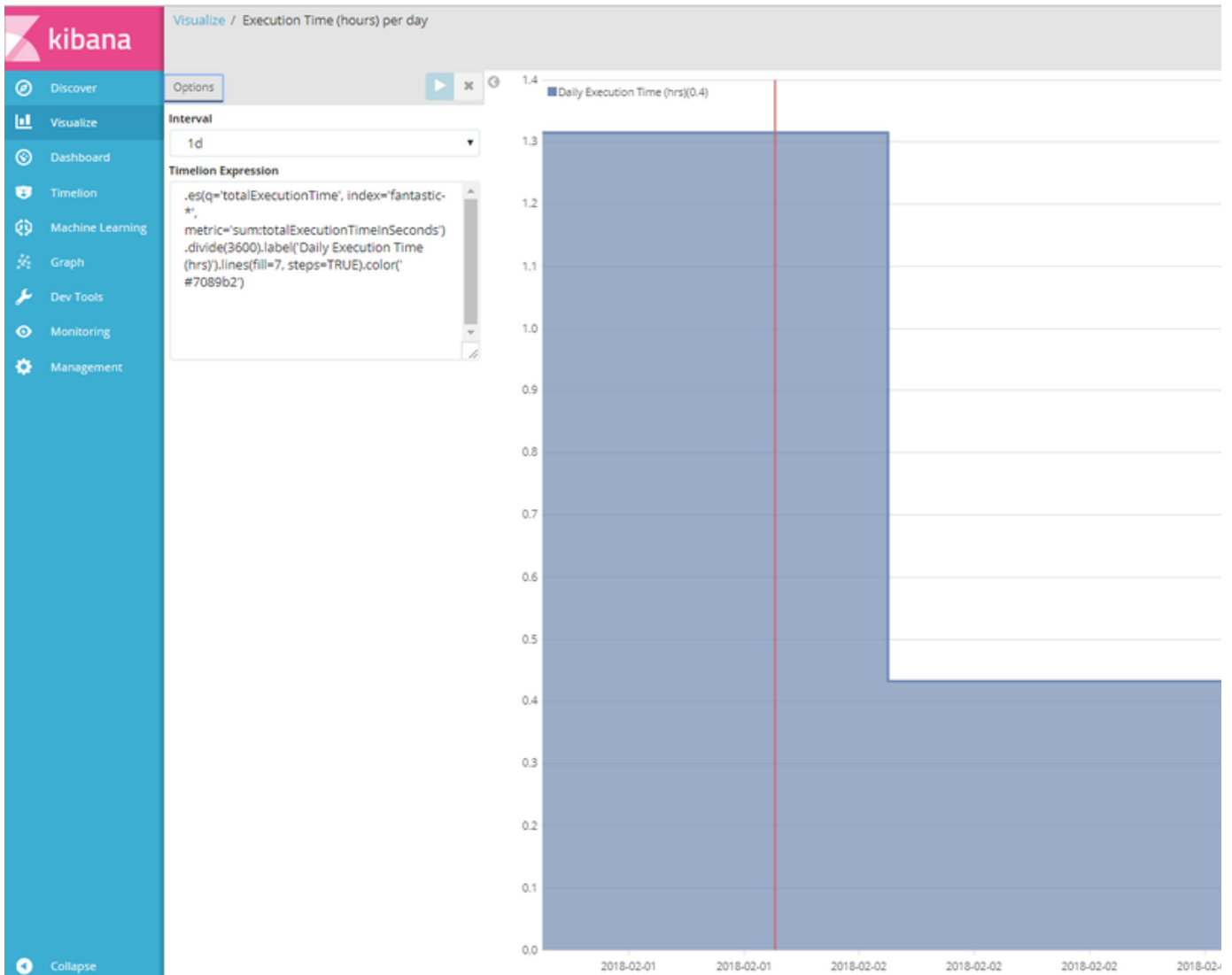
Please see the Timelion documentation for more information.