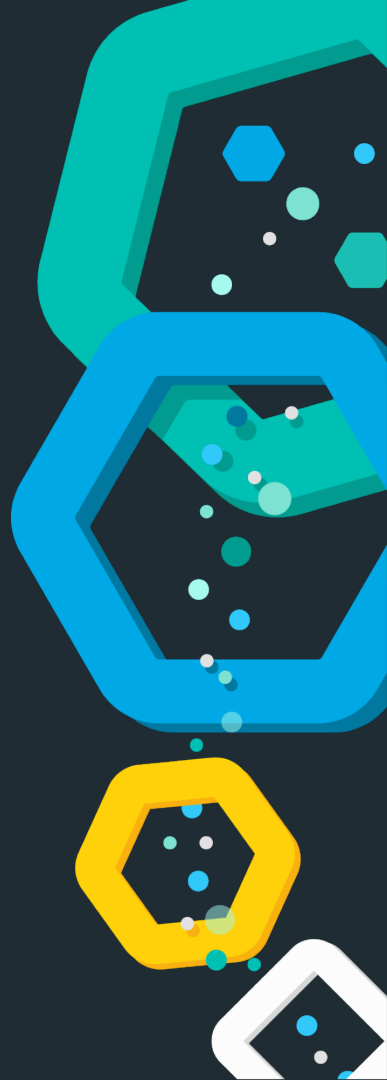
elastic

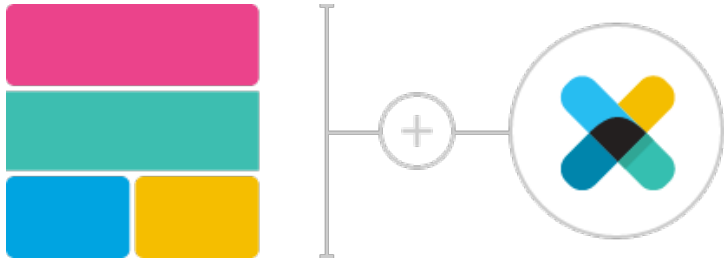
X-Pack Machine Learning Workshop - v5.5+

Rich Collier
Solutions Architect
@richcollier



X-Pack

Extensions for the Elastic Stack



Security	Alerting	Monitoring
Reporting	Graph Analytics	Machine Learning

elastic

Overview

- Anomaly Detection Concepts
- Lab 1: The Simplest Job
- From Concept to ML Configuration
- Operational Logistics
- Lab 2: Advanced Jobs
- Exploring Results with Explorer View
- Lab 3: Multi-Metric Jobs
- Comparison of Job Types
- Other kinds of analysis
- Lab 4: Multi-job Analysis
- Advanced Topics

Anomaly Detection Concepts

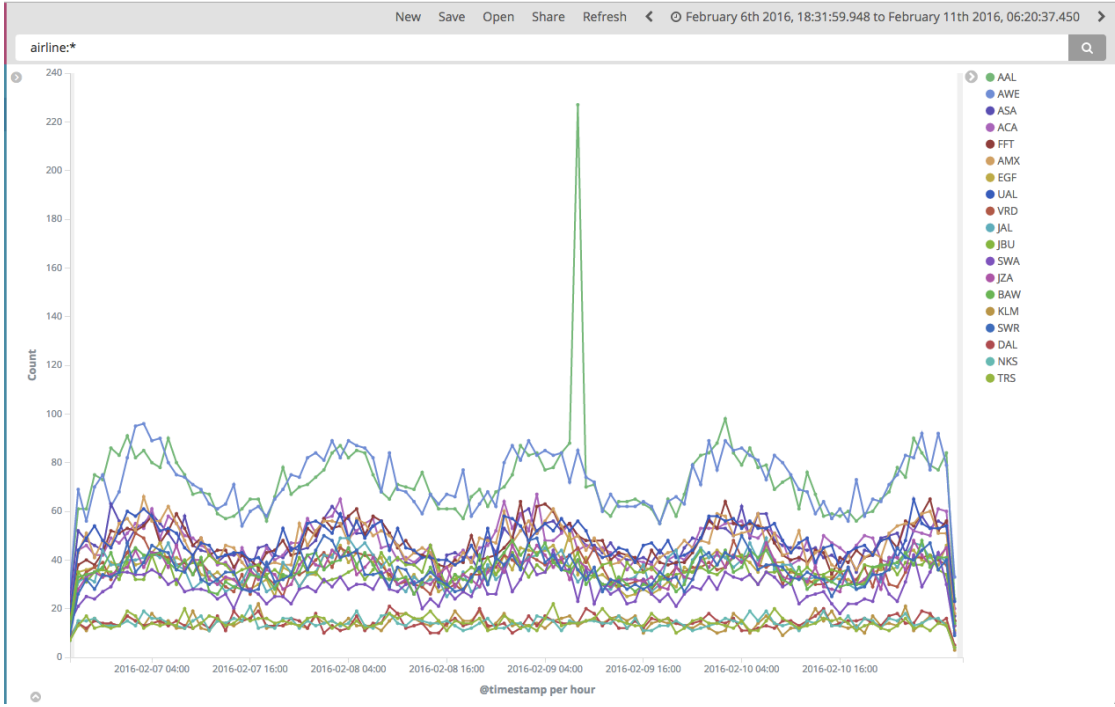
Terminology

- ***Machine Learning***
  - Broad term, but X-Pack Machine Learning is automated anomaly detection for time-series data (for now).
- ***Anomaly Detection***
  - Discovery of what's "weird" or "different", not what's "bad"
- ***Unsupervised Learning***
  - Learning without human-labeled examples (without being "taught"). Rely only on the data
- ***Bayesian***
  - An approach based on probability in which prior results are used to calculate probabilities of certain present or future events

What is “Abnormal”?

What’s abnormal here?

Why?

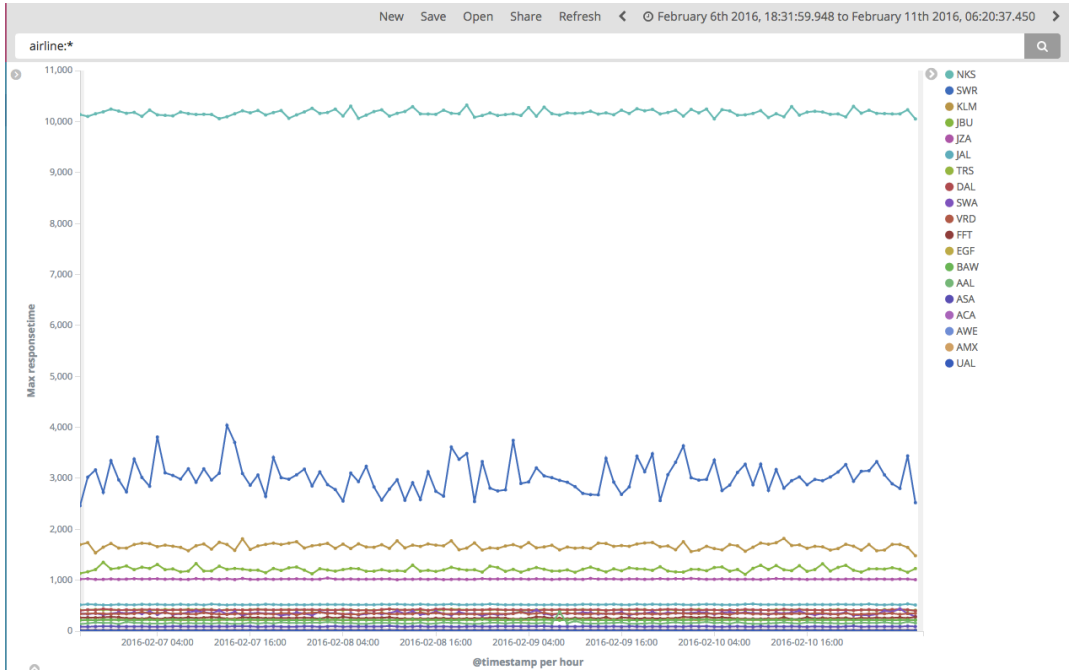


elastic

What is “Abnormal”?

What’s abnormal here?

Why?



elastic

What is “Abnormal”?

What’s abnormal here?

Why?

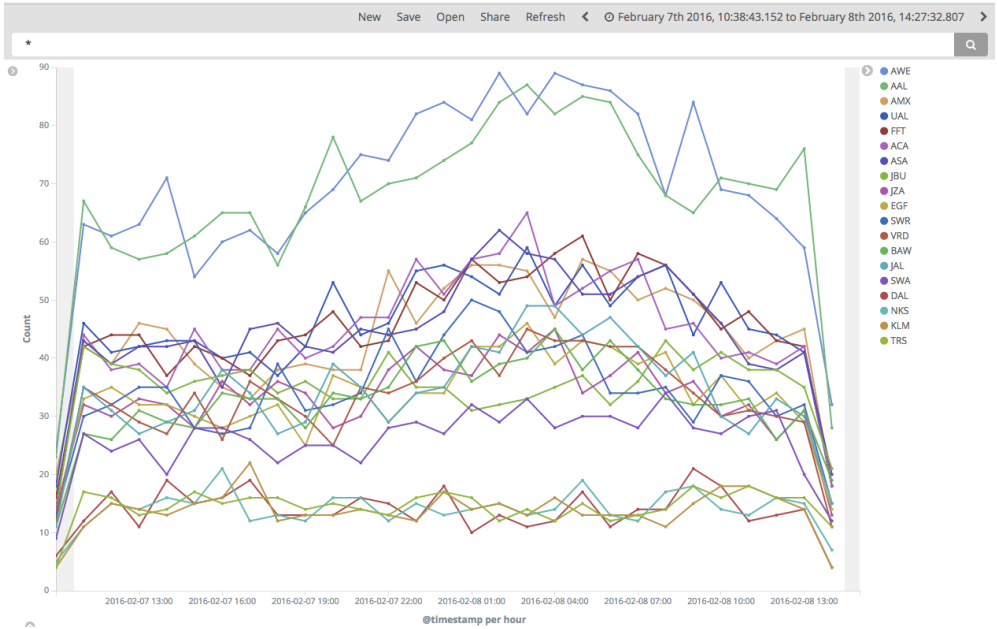
elastic

What is “Normal”?

In general, this question can be answered in two ways:

1) Something behaves in a consistent way with respect to itself, over time

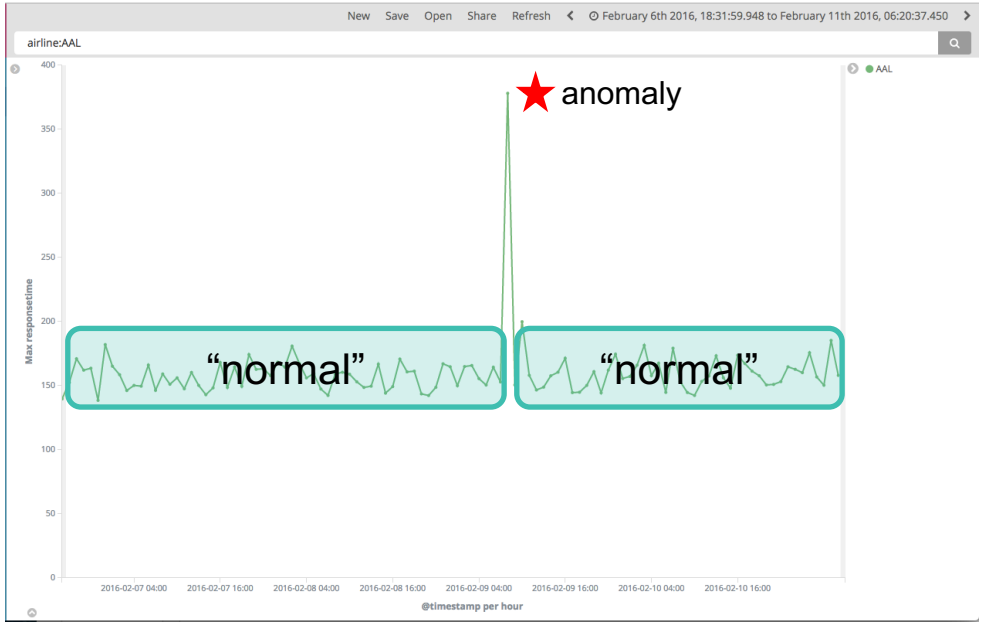
2) Something behaves in a consistent way compared against similar entities



elastic

What is “Abnormal”?

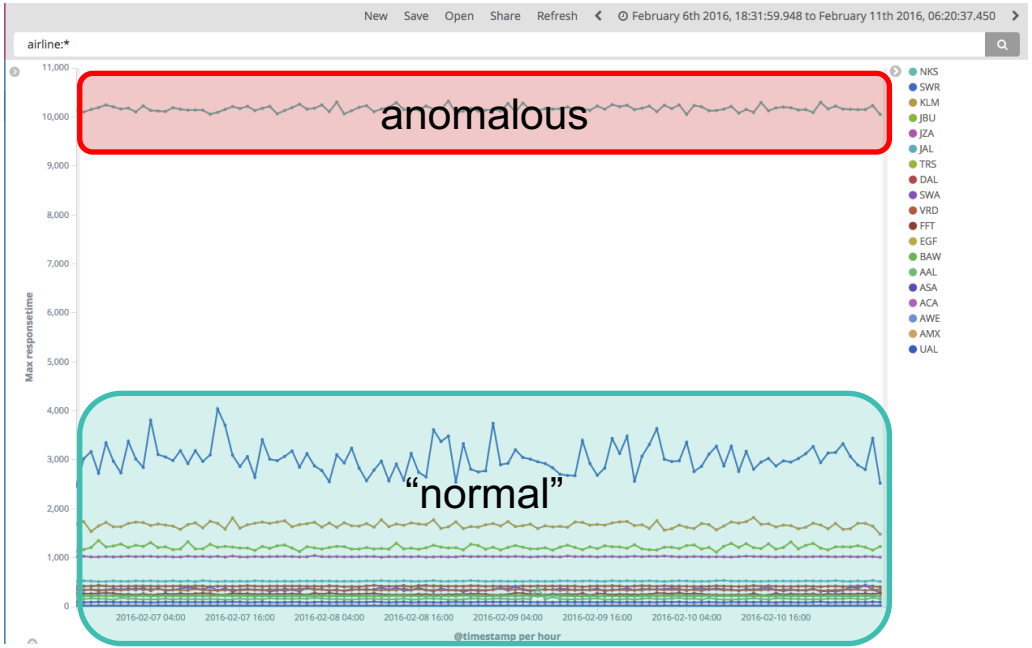
1) If something changes its behavior, compared to its own history – that change is ***anomalous***.



elastic

What is “Abnormal”?

2) If something is drastically different that others within a population, then that entity is ***anomalous***.

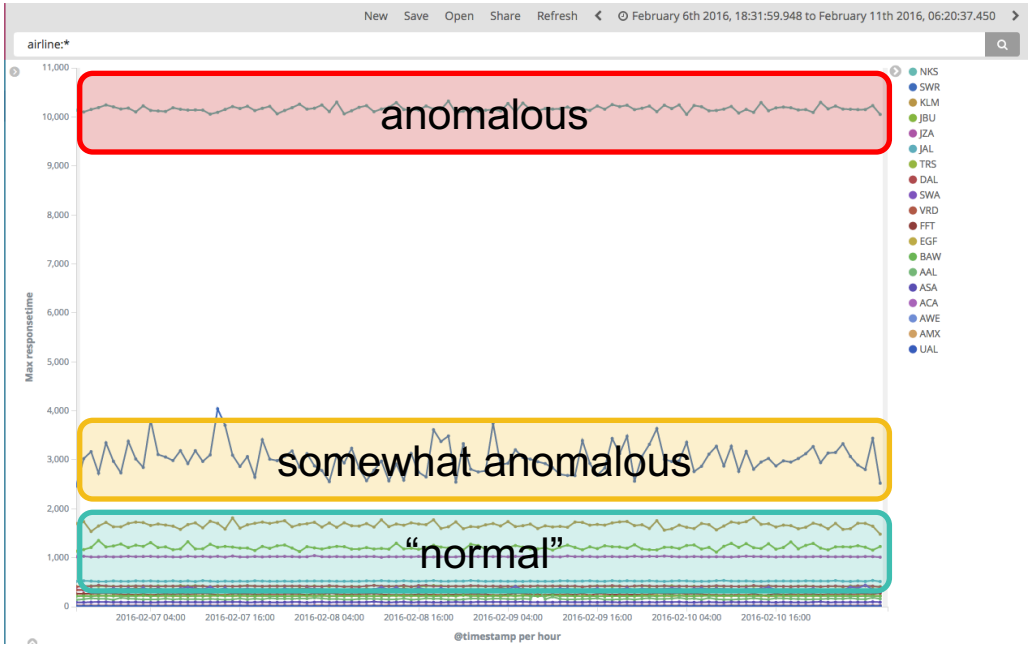


elastic

What is “Abnormal”?

2) If something is drastically different that others within a population, then that entity is ***anomalous***.

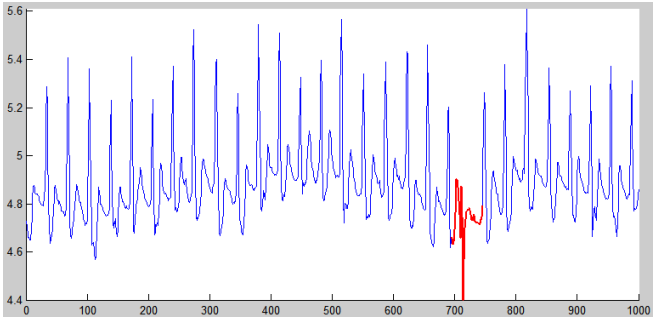
There’s also the concept of being “somewhat anomalous”



elastic

In Summary, Anomalousness is:

1) When an entities' ***behavior changes*** significantly and suddenly



2) When an entity is drastically ***different than others*** within a population

elastic

How to Learn “Normal”

An Analogy

- How could I learn how much Postal-mail you get daily and how could I use that information to predict how much you might get tomorrow?

elastic

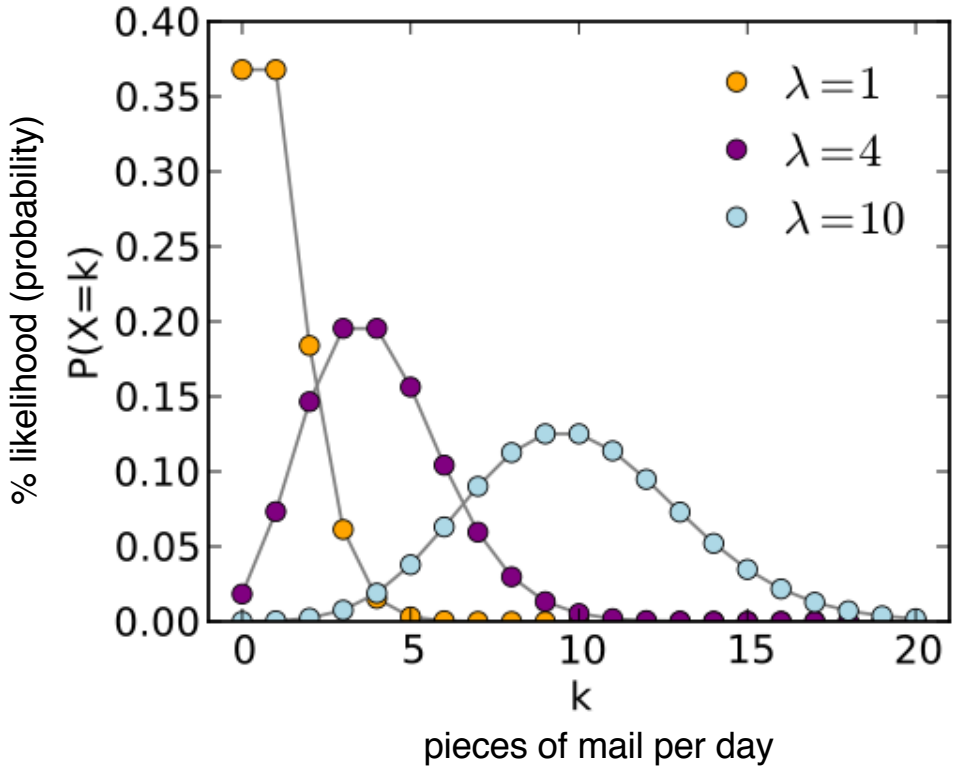
I could stand on your front porch and…

- Watch your mail delivery volume for a while, but how long?
  - 1 day?
  - 1 week?
  - 1 month?

- Notice, that you intuitively feel like you'll gain accuracy in your predictions with more data that you see.

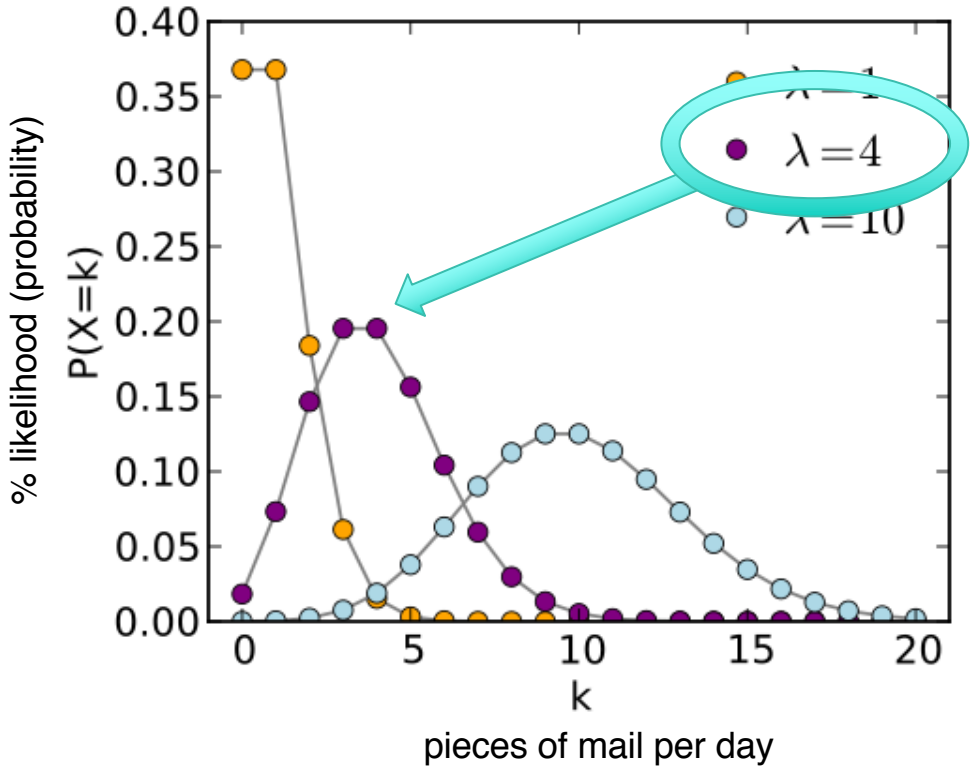
- Use those observations to create a model…

Probability Distribution Function



elastic

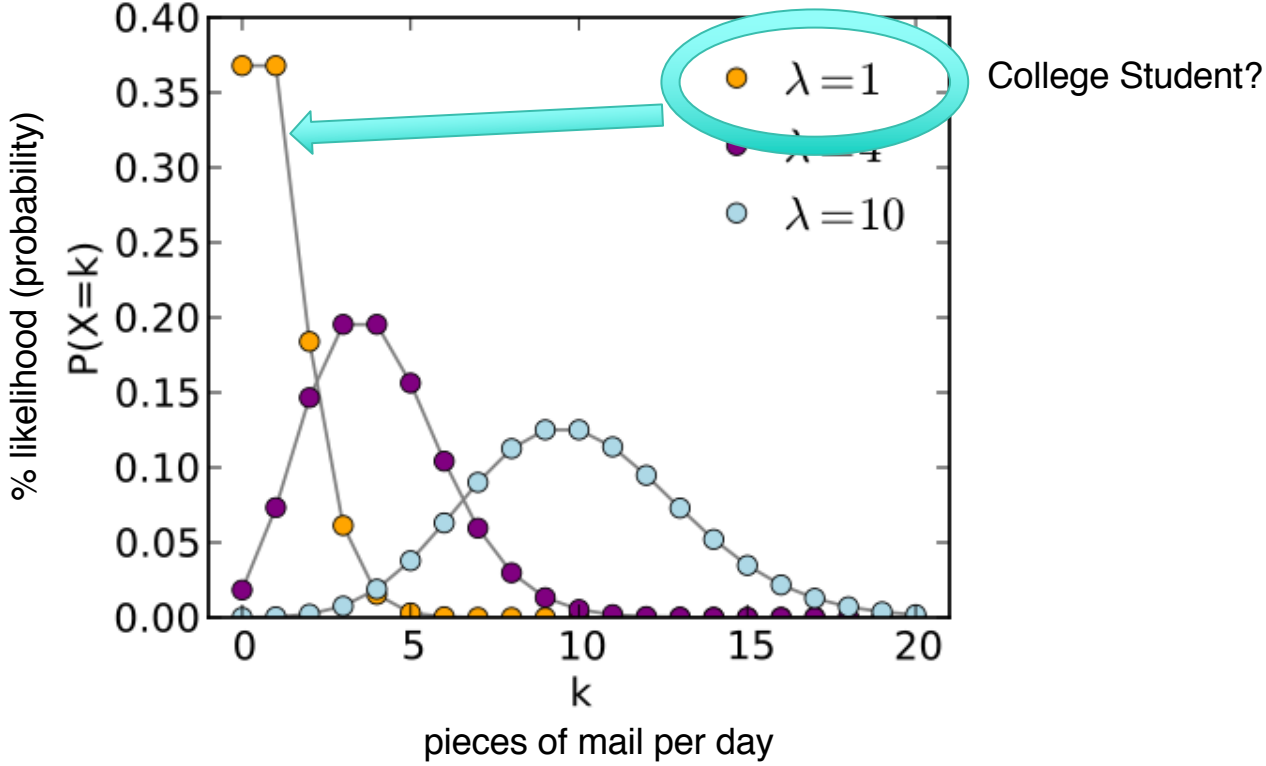
Probability Distribution Function



Best for my house

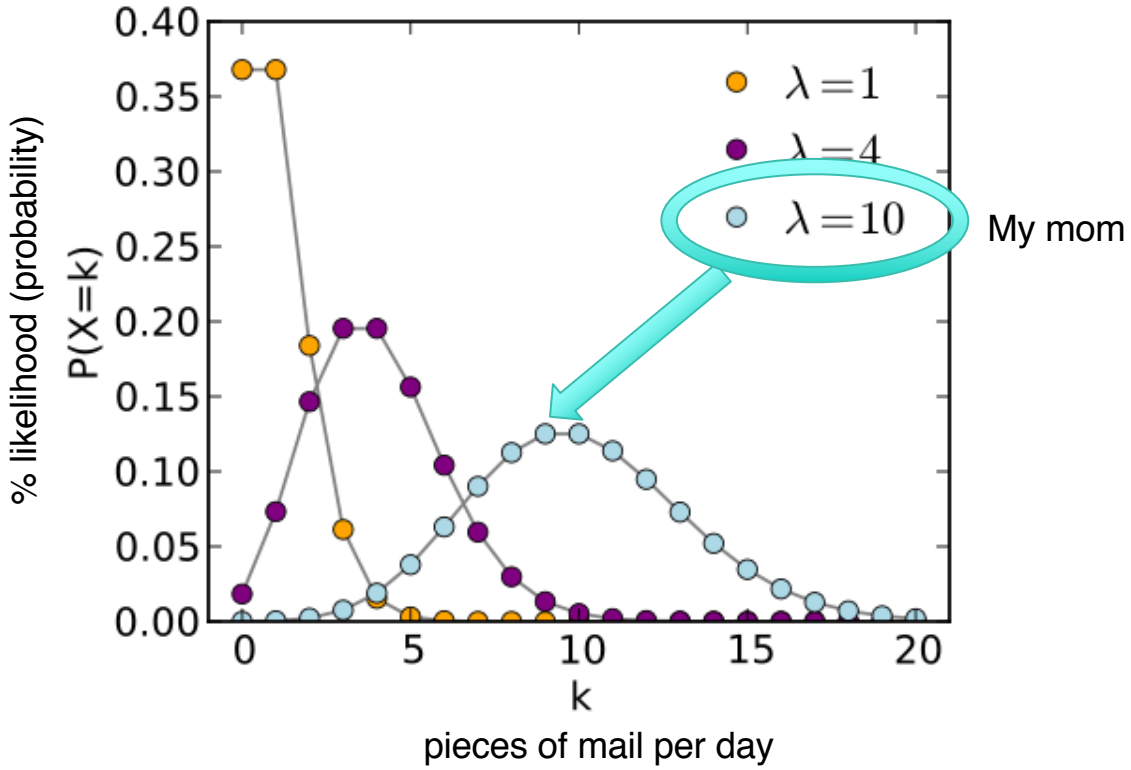
elastic

Probability Distribution Function



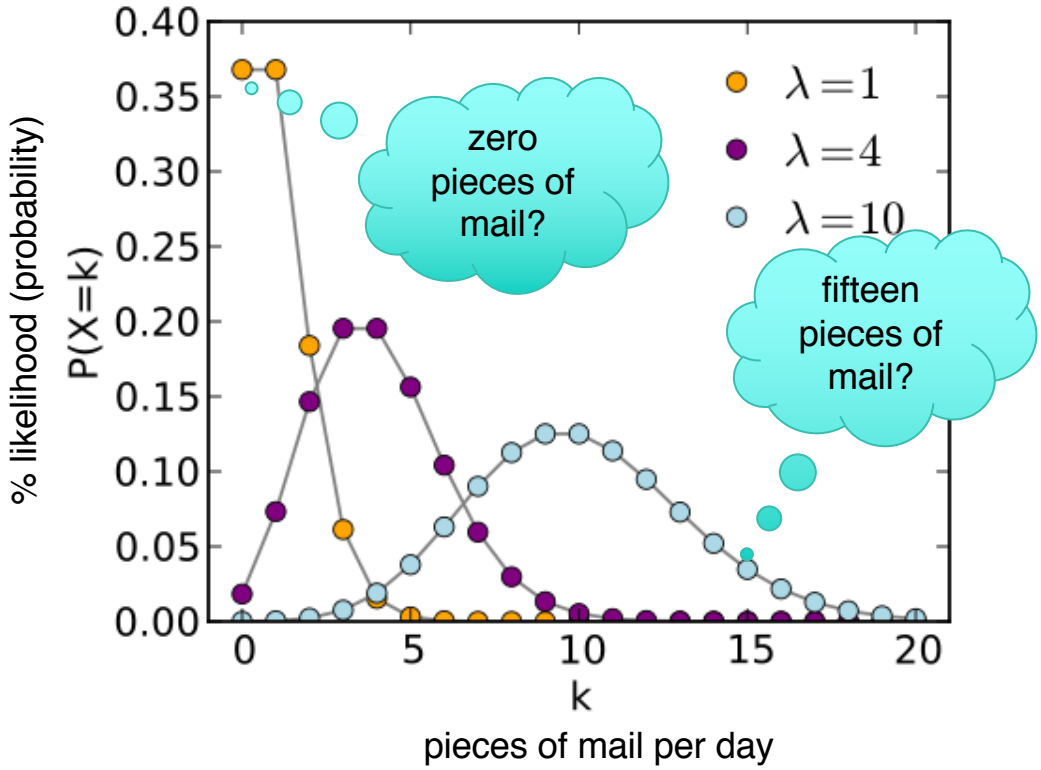
elastic

Probability Distribution Function



elastic

Using the Model to Find What is Unexpected



elastic

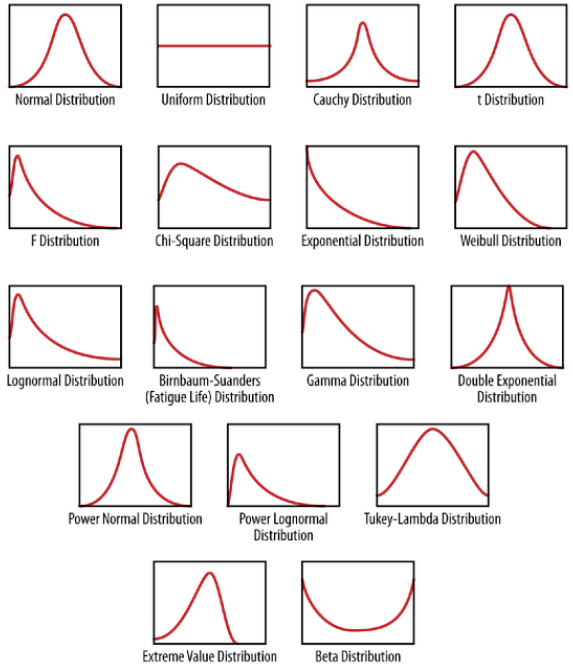
Relate back to IT/Security data

- # Pieces of mail = # events of a certain type
  - number of failed logins
  - number of errors in a log
  - number of events with certain status codes
- Or, metrics
  - response time
  - utilization
  - orders per minute

=> Every kind of data will need its own unique “model” (probability distribution function)

How does one pick a model?

Which one best fits your data?

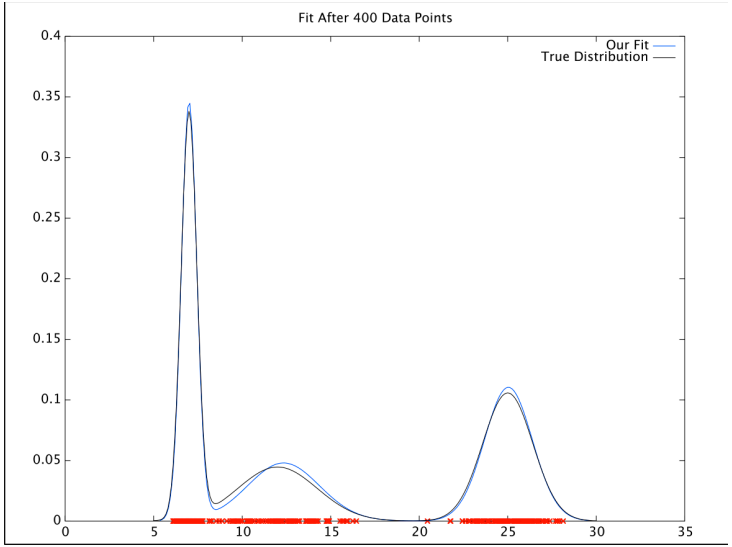


source: “Doing Data Science”
O’Neil & Schutt

elastic

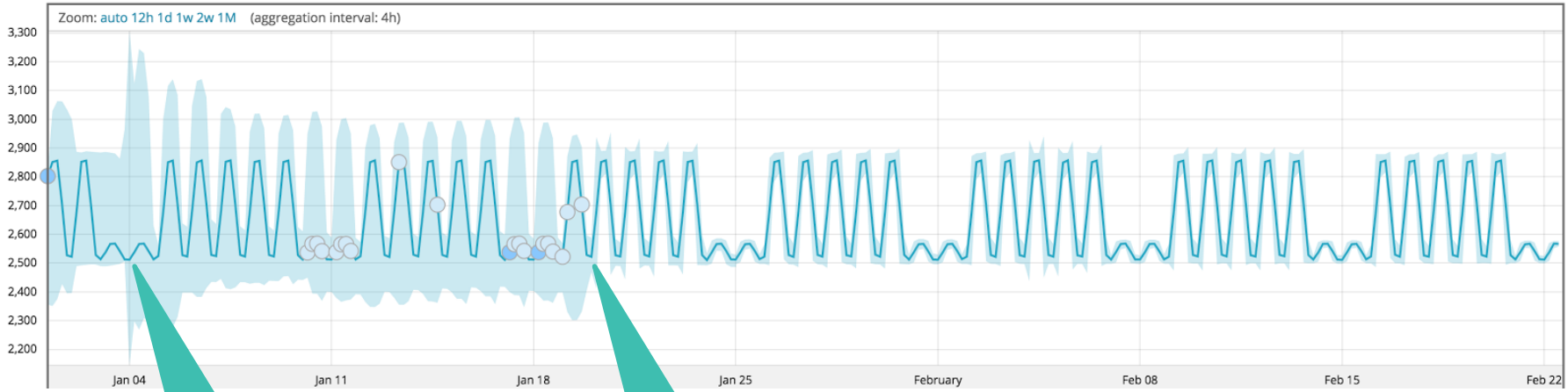
Machine Learning picks it for you

- ML uses sophisticated machine-learning techniques to best-fit the right statistical model for your data.
- Better models = better outlier detection = less false alarms
- Anomalies occur when observation is in low probability area



elastic

The Model's Evolution in Time



After 2 full days, daily periodicity has been learned.

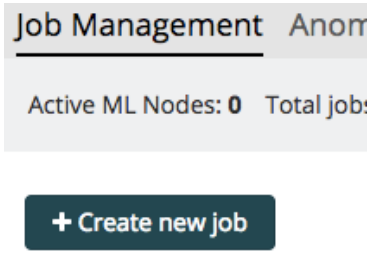
After 2 full weeks, weekly periodicity has been learned.

elastic

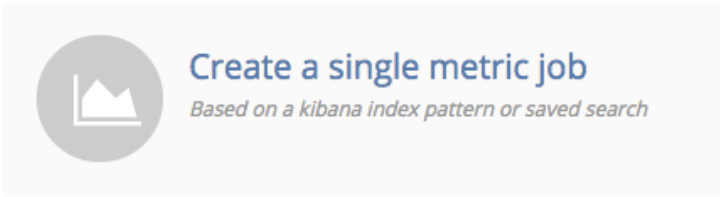
Lab 1: The Simplest Job

Steps to Complete

1) In Machine Learning, Create new job



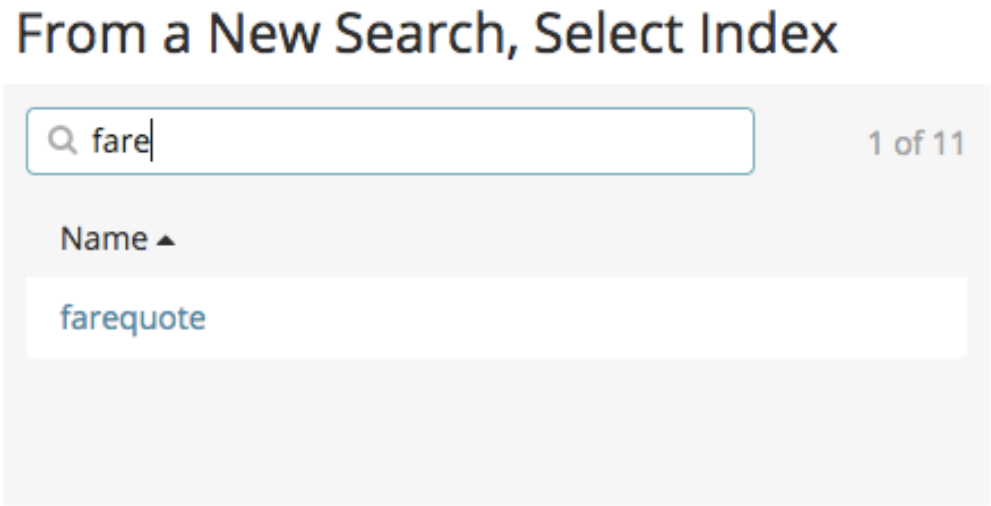
2) Choose single metric



elastic

Steps to Complete

3) pick farequote index



elastic

Steps to Complete

4) choose the “Count” aggregation

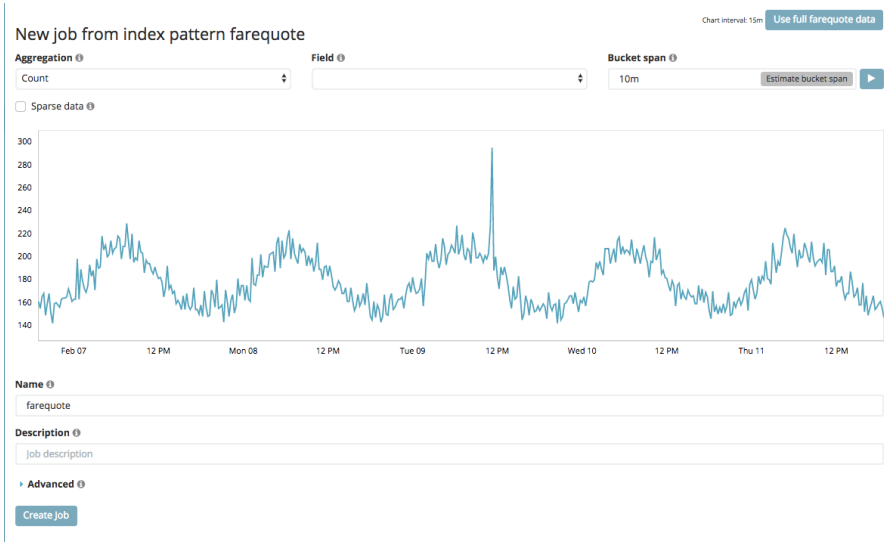
5) select 10m for bucket span

6) leave “field” blank (we don’t count fields, we’re counting documents)

7) click the “use full farequote data” button

8) name job “farequote”

9) click “Create Job”

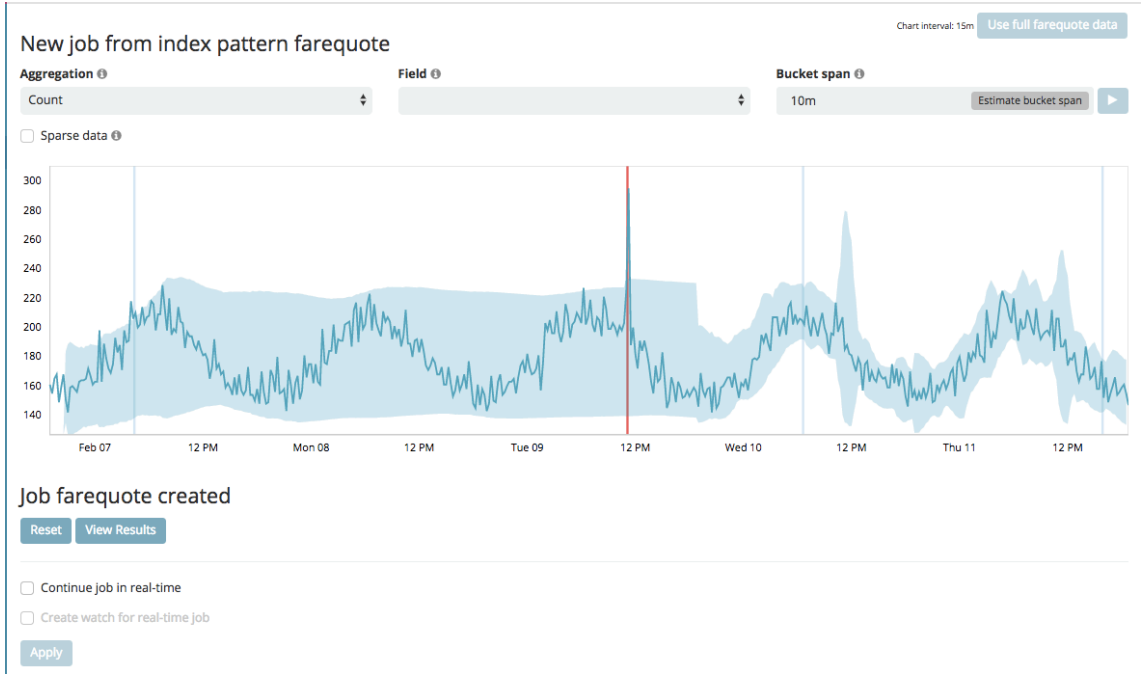


elastic

Steps to Complete

10) See animated learning

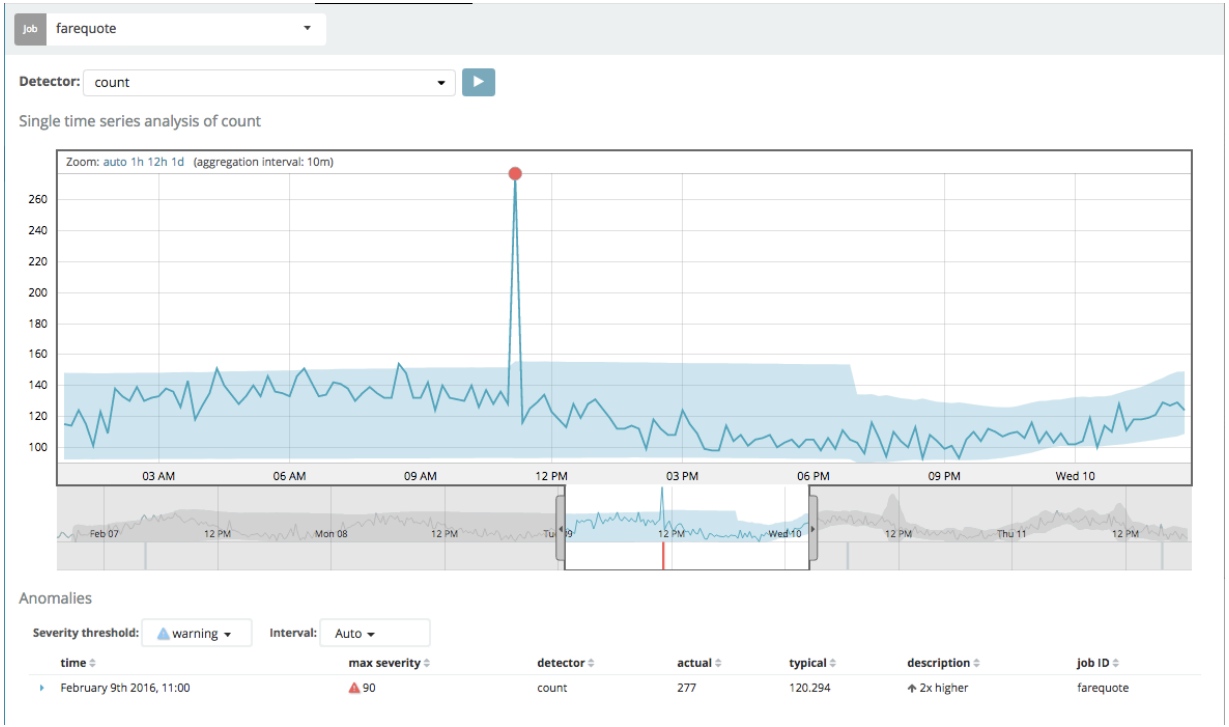
11) click “View Results”



elastic

Steps to Complete

12) Zoom in on anomaly



elastic

From Concept to ML Configuration

elastic

Recap:

- We have two relevant definitions of anomalousness
  - change with respect to self as a function of time
  - relative difference compared to peers within a population

- We know that we can “learn” normal merely by observing data over time and building a probabilistic model

- The accuracy of the model is important, but fortunately, ML can do the hard work of proper model selection for you
  - But, you need to first **select the features** of the data to model

Situation:

- Your data:

```
2016/02/08 06:20:43 INFO  [http-8680]: FareQuoteImpl - FareQuoteImpl.getFare(AAL): exiting: 92.5638

2016/02/08 06:20:44 INFO  [http-8680]: FareQuoteImpl - FareQuoteImpl.getFare(JZA): exiting: 990.4628

2016/02/08 06:20:46 INFO  [http-8680]: FareQuoteImpl - FareQuoteImpl.getFare(JBU): exiting: 877.5927

...
```

- Question: How can we use ML to find out what's unusual in this data?

elastic

Feature Selection

- Which attributes of a this data could be used to judge its unusualness?

raw logs

```
2016/02/08 06:20:43 INFO  [http-8680]: FareQuoteImpl - FareQuoteImpl.getFare(AAL): exiting: 92.5638
2016/02/08 06:20:44 INFO  [http-8680]: FareQuoteImpl - FareQuoteImpl.getFare(JZA): exiting: 990.4628
2016/02/08 06:20:46 INFO  [http-8680]: FareQuoteImpl - FareQuoteImpl.getFare(JBU): exiting: 877.5927
...
```

document


```
{
  "_index": "farequote",
  "_type": "response",
  "_id": "AVNQ1__XRcuaRiYtw-jH",
  "_score": 3.290889,
  "_source": {
    "sourcetype": "farequote",
    "airline": "AAL",
    "responsetime": "92.5638",
    "time": "2016-02-08T06:20:43+0000"
  }
}
```


feature 1 (categorical)

feature 2 (metric)

elastic

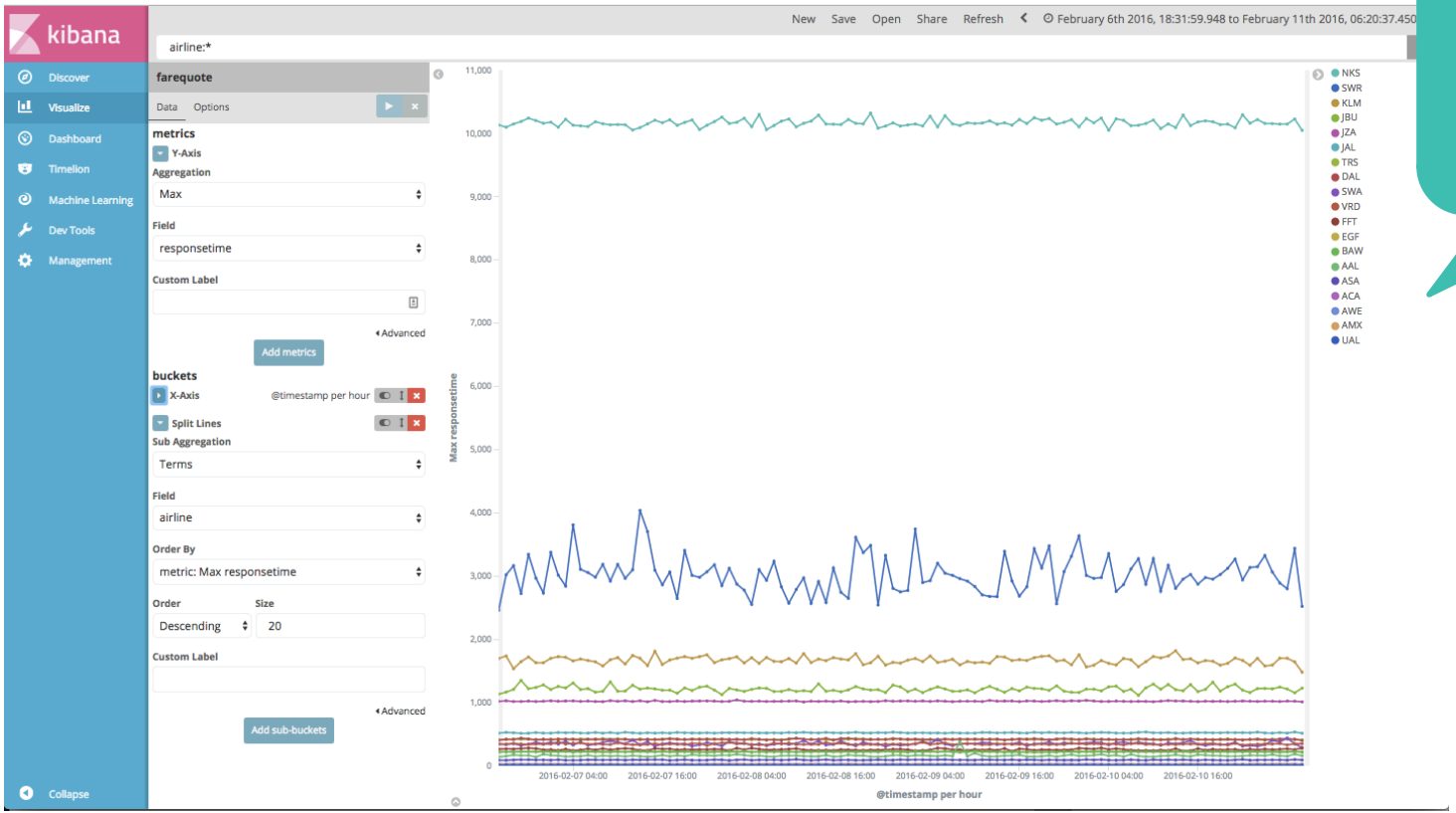
What Kinds of Questions Can be Answered?

QUESTION	ANSWERABLE?
Is there an unusual ***amount*** of requests per unit time (total)?	Yes
Is there any particular airlines with unusual ***amounts*** of requests per unit time?	Yes
Is the ***total*** response time of ***all*** API calls unusually long?	Yes
Is the response time of API calls ***per airline*** unusually long?	Yes
Are there any airlines with excessive take-off delays?	No

Let's focus on this one for now

elastic

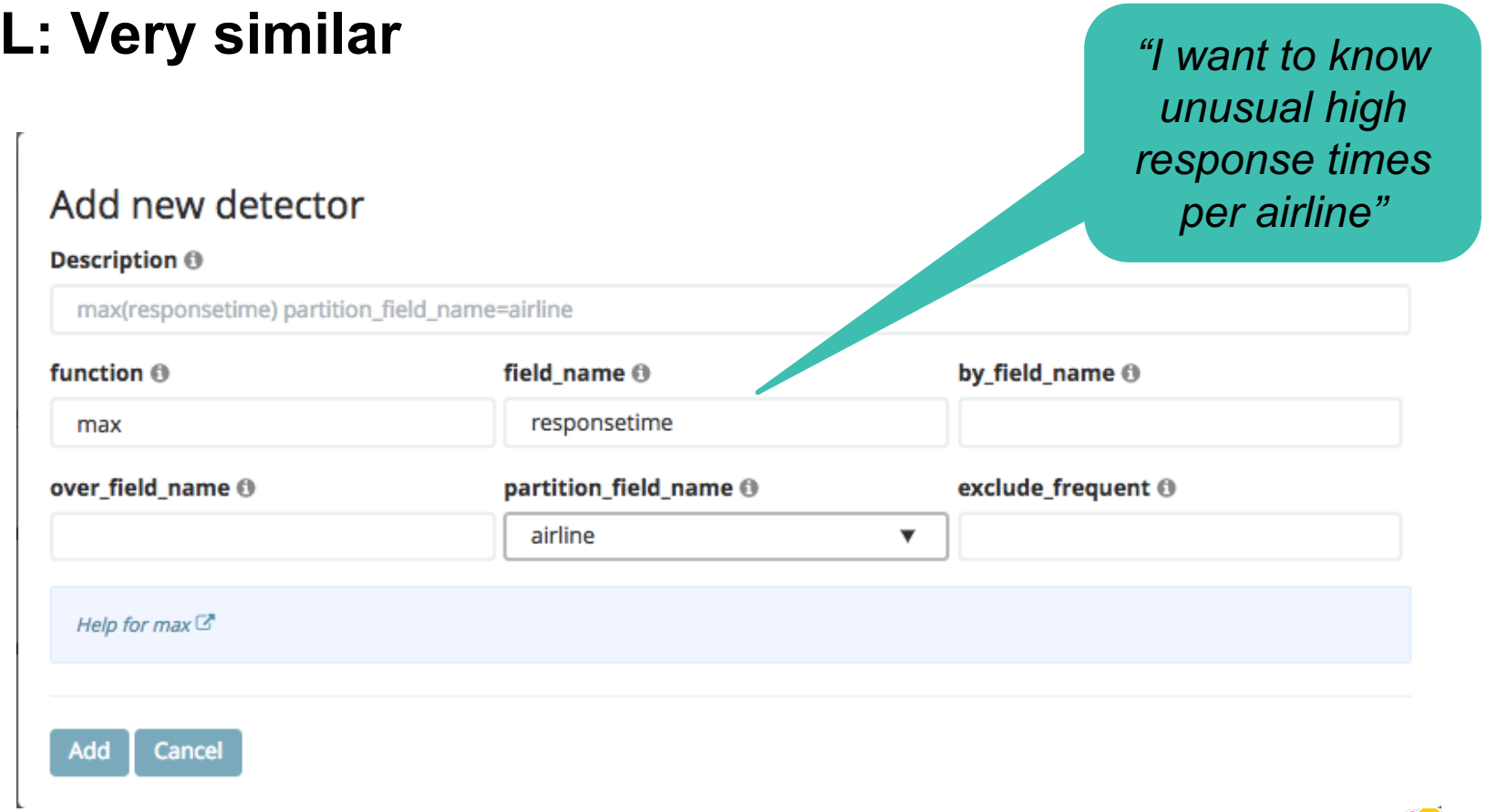
In Kibana: How to Answer that Question



"I want to know unusual high response times per airline"

elastic

In ML: Very similar



elastic

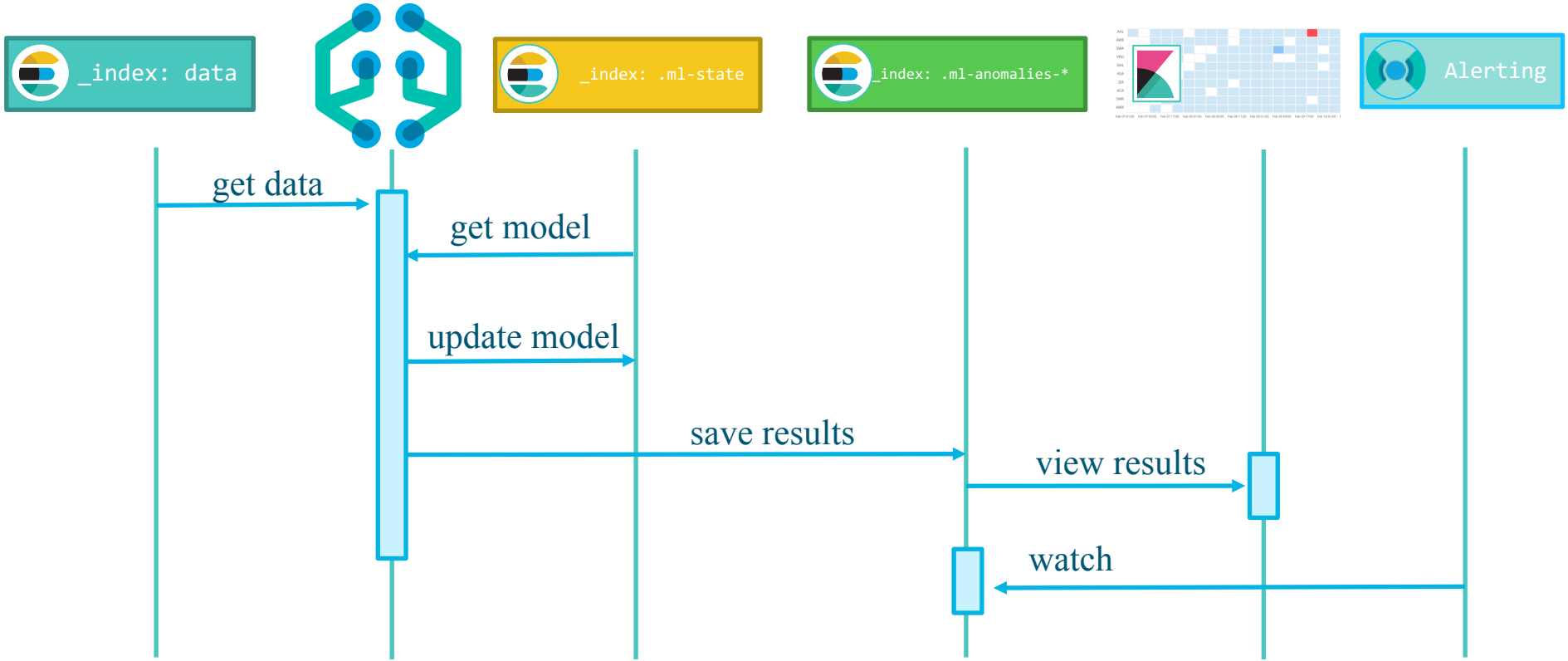
Choosing Meaningful Features is Important

- Which attributes of a dog could be used to judge its unusualness?

elastic

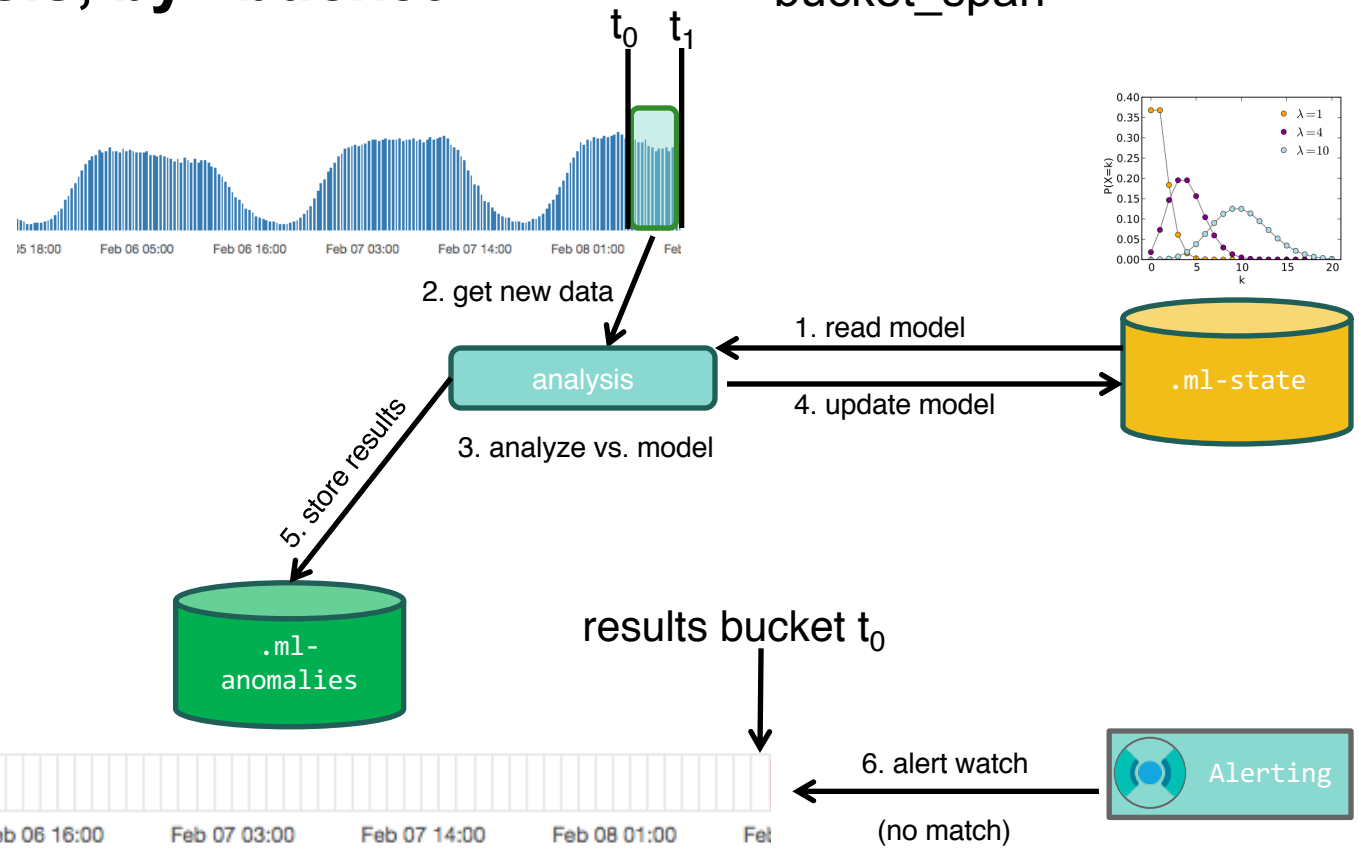
Operational Logistics

Sequence Diagram

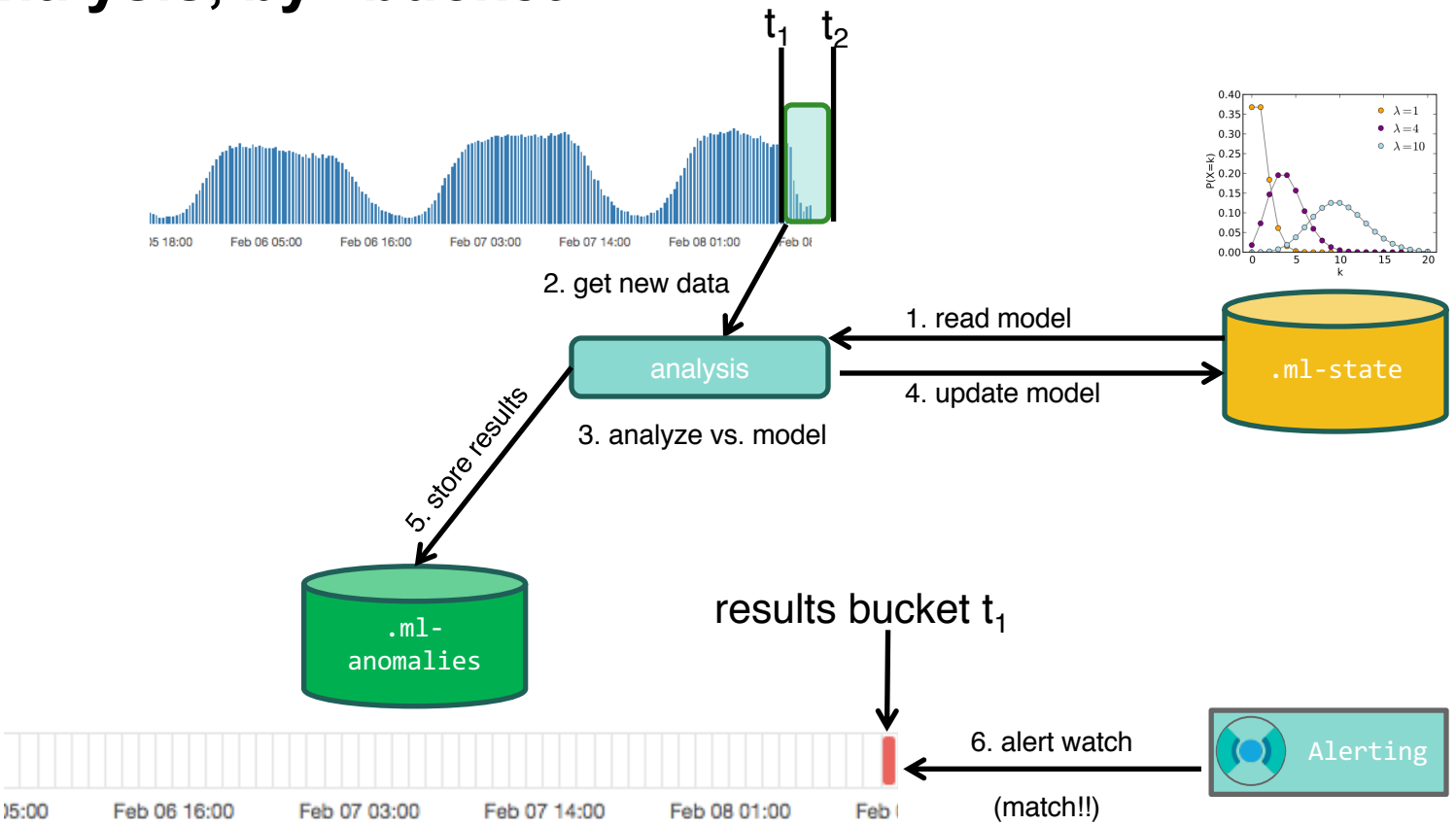


Analysis, by “bucket”

$t_n - t_{n-1}$ is called the “bucket_span”



Analysis, by “bucket”



elastic

Lab 2: Advanced Jobs

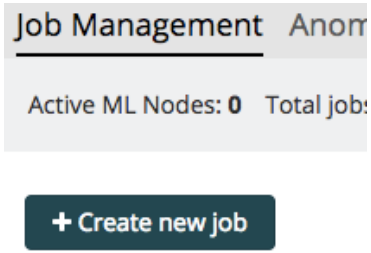
elastic

Steps to Complete

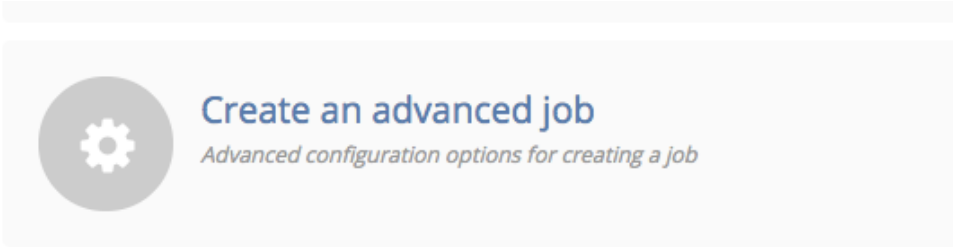
- Using the “farequote” data set:
- Get familiar with how the raw data looks in Kibana Discover
- Load the ML home screen and create a new “advanced” job to analyze unusually slow response times per airline
  - choose a bucket_span of 10m
  - pick an appropriate way to split data (hint: partition using airline)
  - select airline as influencer
- Run the job over the entire data set (data is not real-time)

Steps to Complete

1) Create new job



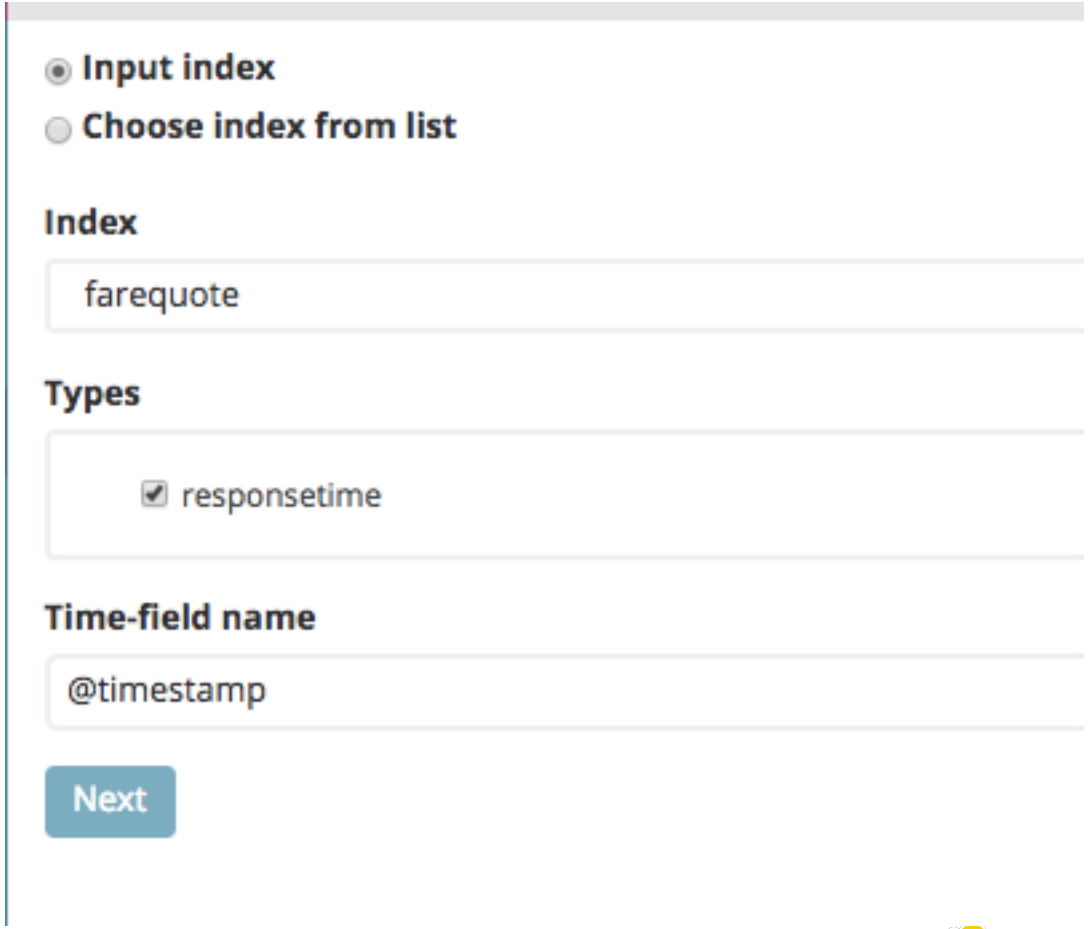
2) Choose advanced



elastic

Steps to Complete

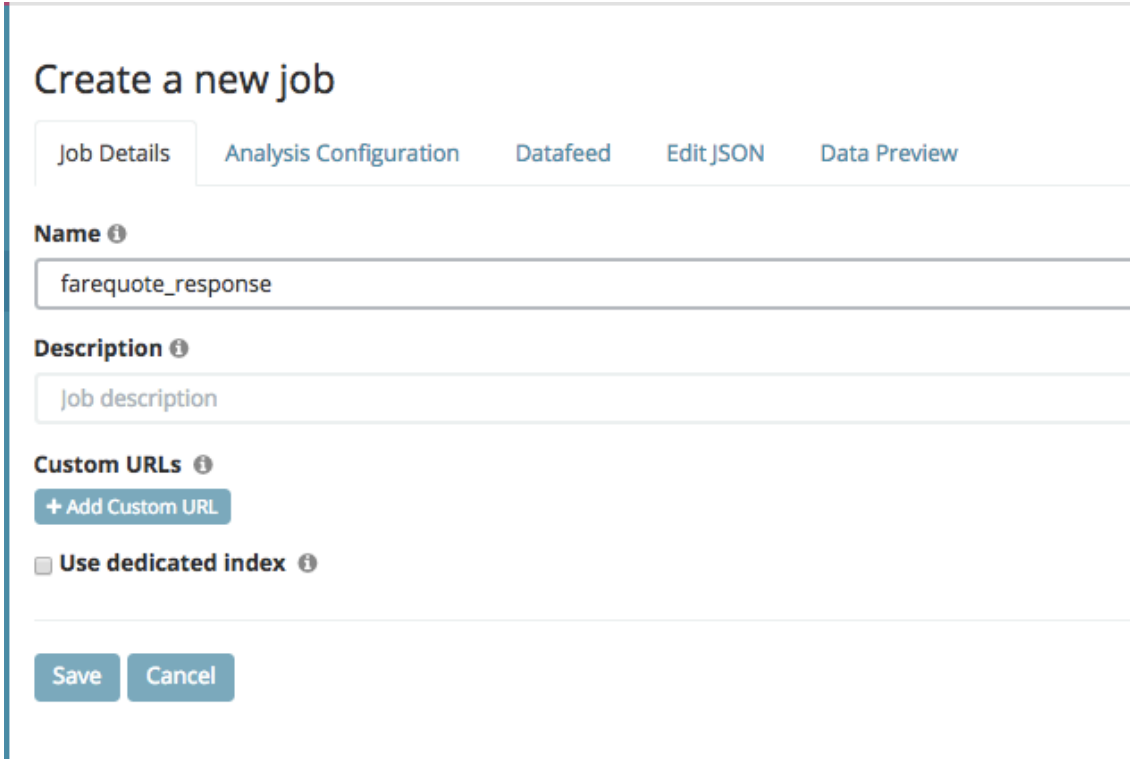
3) Pick farequote index



elastic

Steps to Complete

5) Name job
“farequote_response”



elastic

Steps to Complete

6) set bucket_span=10m

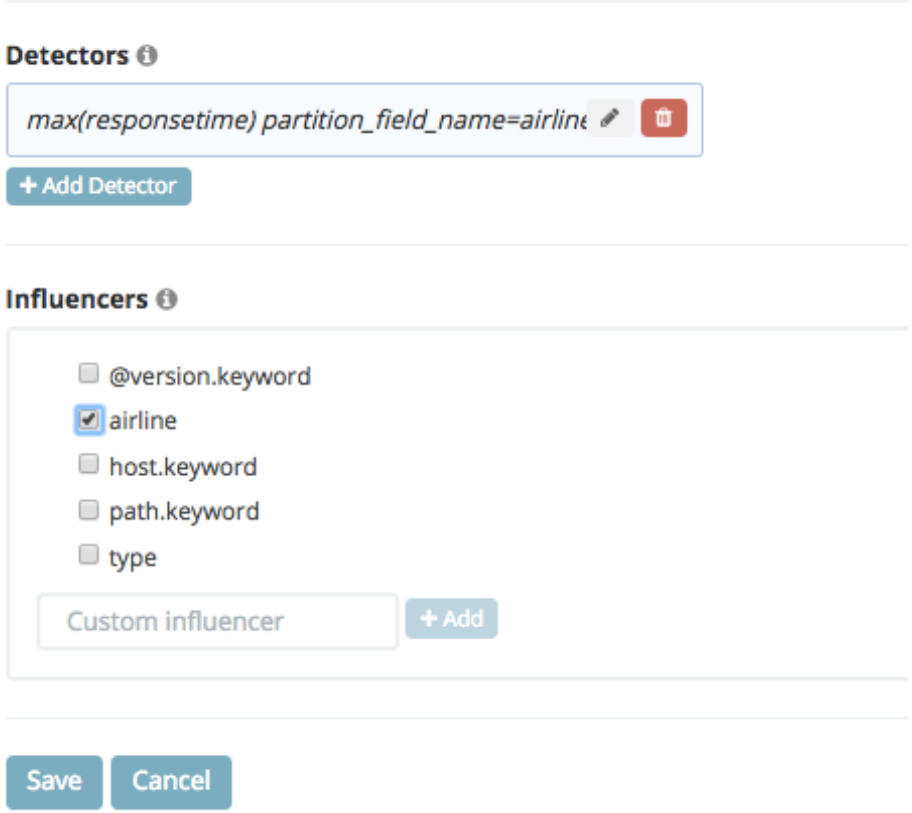
7) Add a Detector:

function: max
field_name: responsetime
partition_field_name: airline

elastic

Steps to Complete

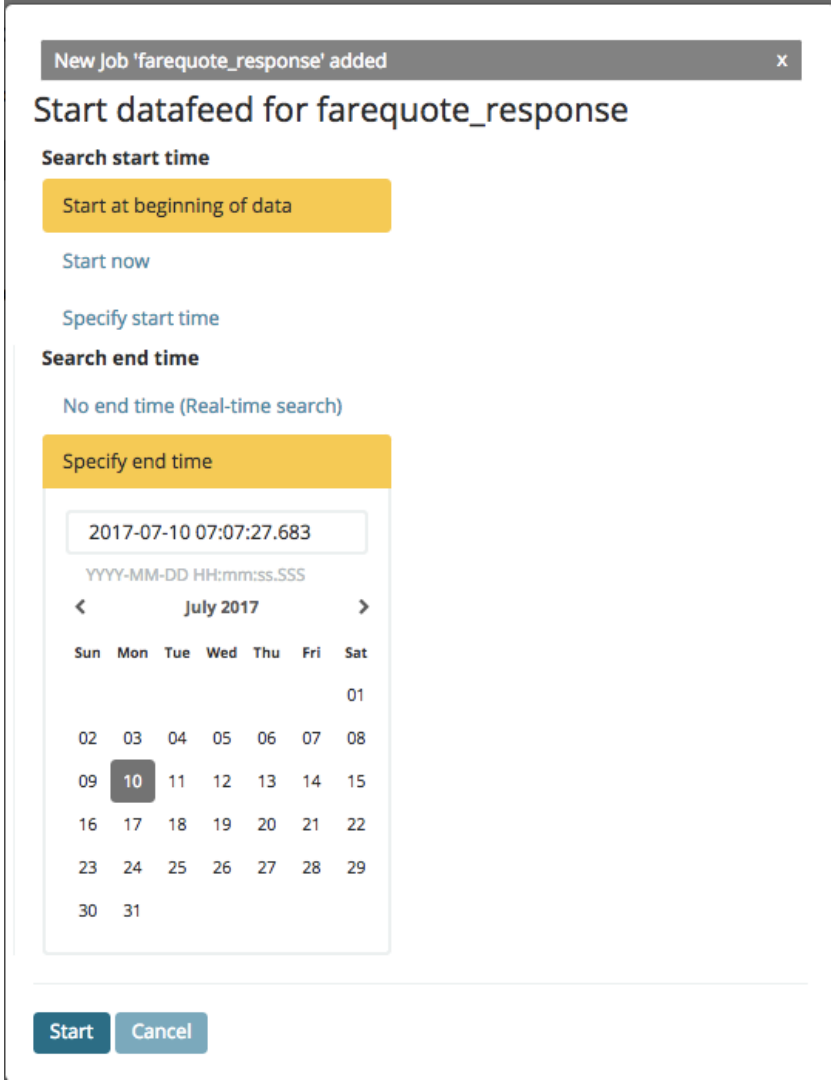
7) select Influencer,
save job
start datafeed



elastic

Steps to Complete

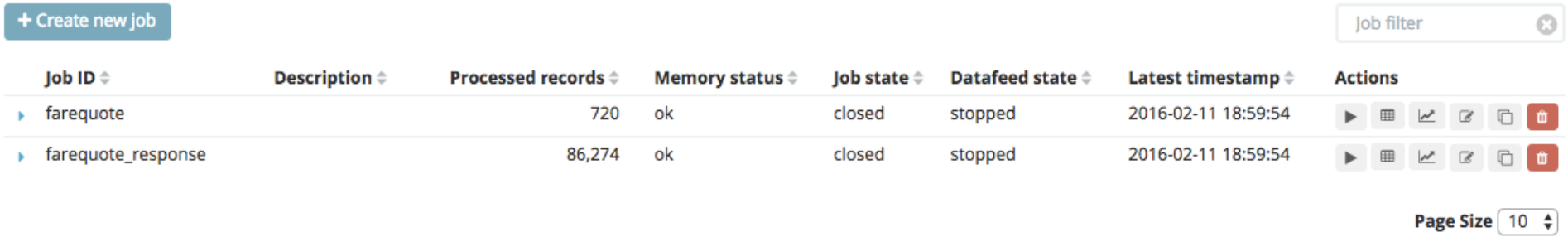
8) start at beginning of data
leave “now” as end time



elastic

Steps to Complete

9) wait until all 86275 events are processed

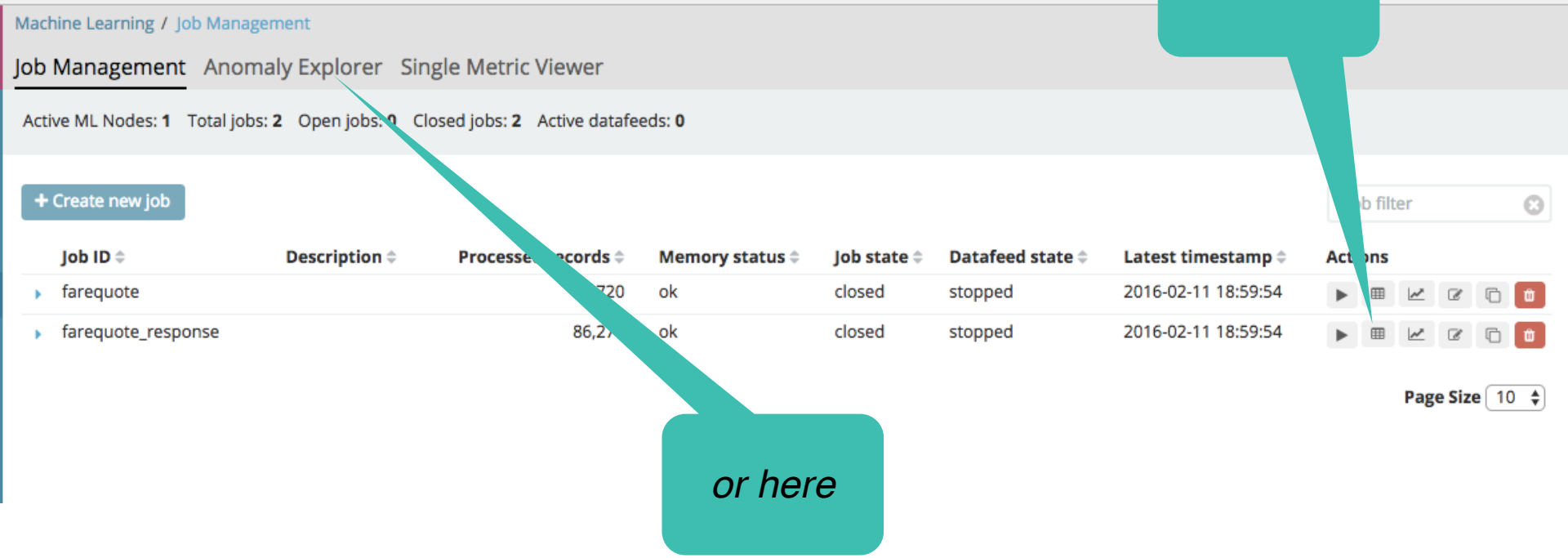


elastic

Exploring Results with Anomaly Explorer View

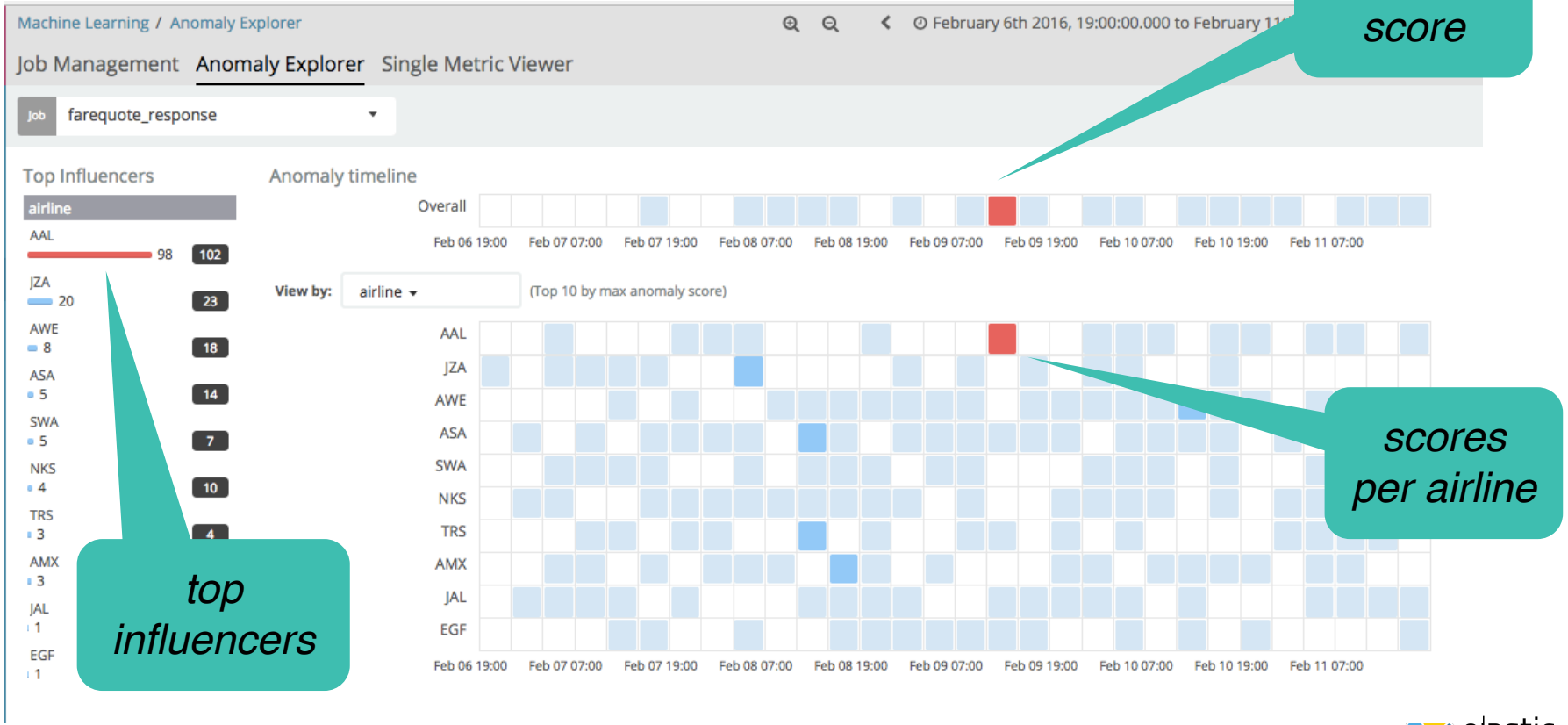
elastic

Explorer View of a Job



elastic

Anomaly Explorer



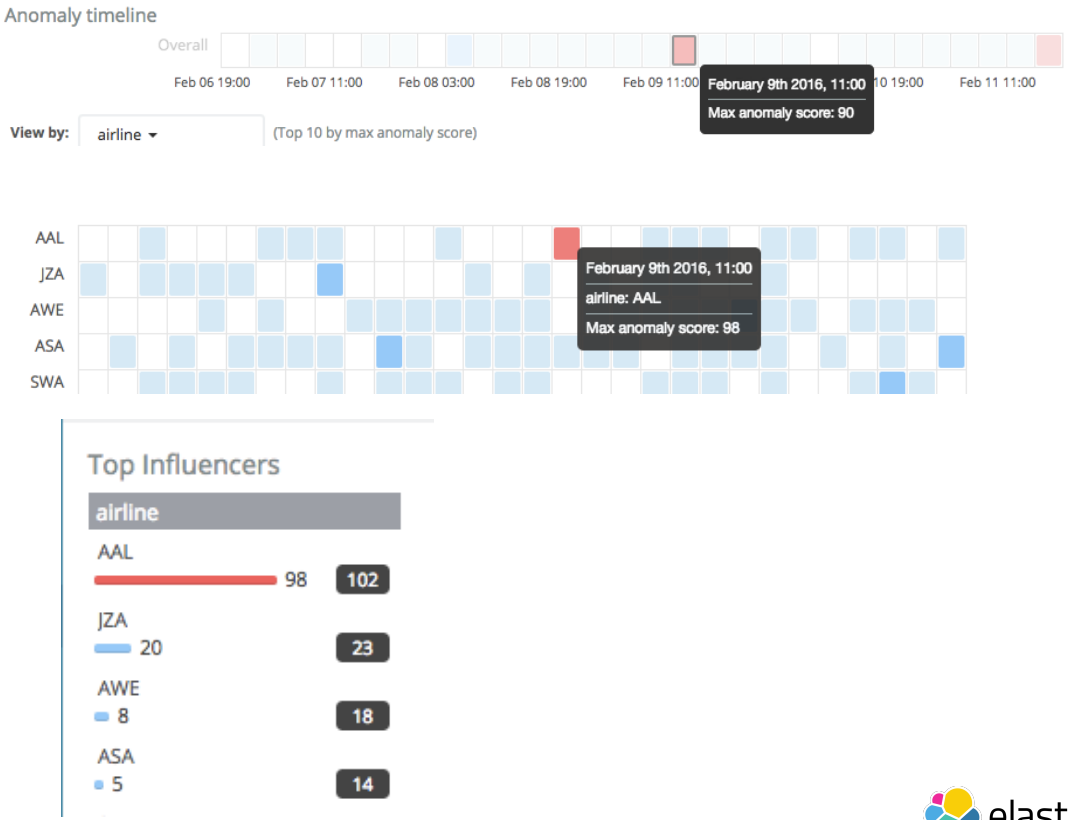
elastic

Concept: What is an Influencer?

- An Influencer is a field, selected at configuration time, that would be a logical entity ”to blame” if an anomaly were to exist
- Doesn’t have to be a field in the actual detector, but fields used to split the data are often good candidates
- Will get its own score based upon how influential that entity is on the anomaly

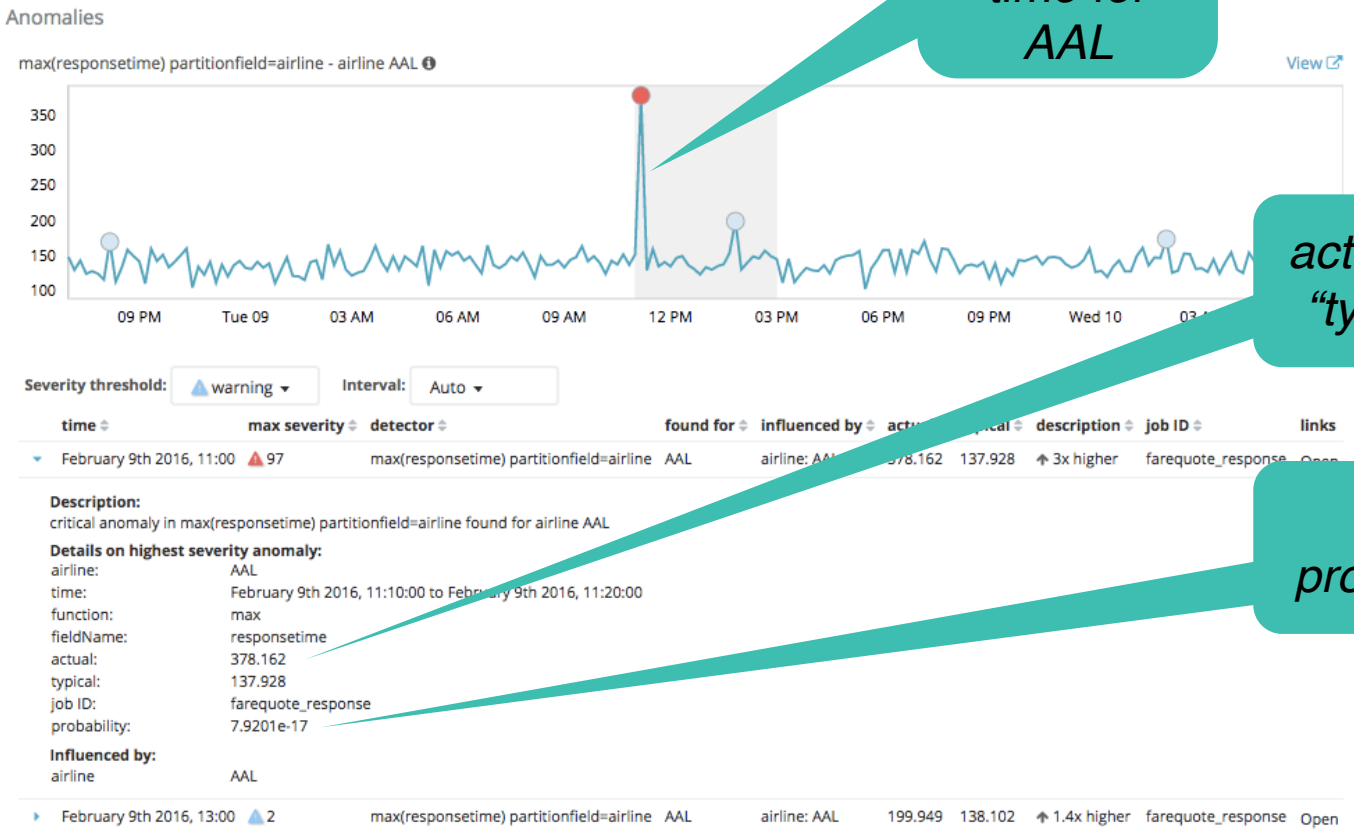
Scoring

- Overall Job score is 90
  - How unusual is that bucket, given all airlines?
- Detector score is 98
  - How unusual is the response time of airline=AAL?
- airline=AAL is the top influencer in this time range
  - 98 is the max anomaly score
  - 102 is the sum of anomaly scores in this time range



elastic

Anomaly Details



view of response time for AAL

actual vs. "typical"

raw probability

elastic

Lab 3: Multi-Metric Jobs

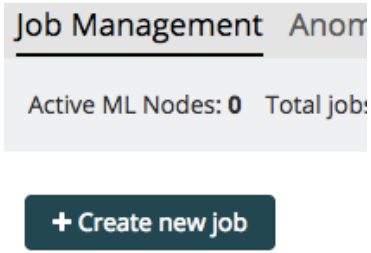
elastic

Steps to Complete

- Again, using the “farequote” data set:
- Re-create the “max(responsetime) per airline” job using a “multi-metric” job
- Also add “count per airline” in the same job

Steps to Complete

1) Create new job

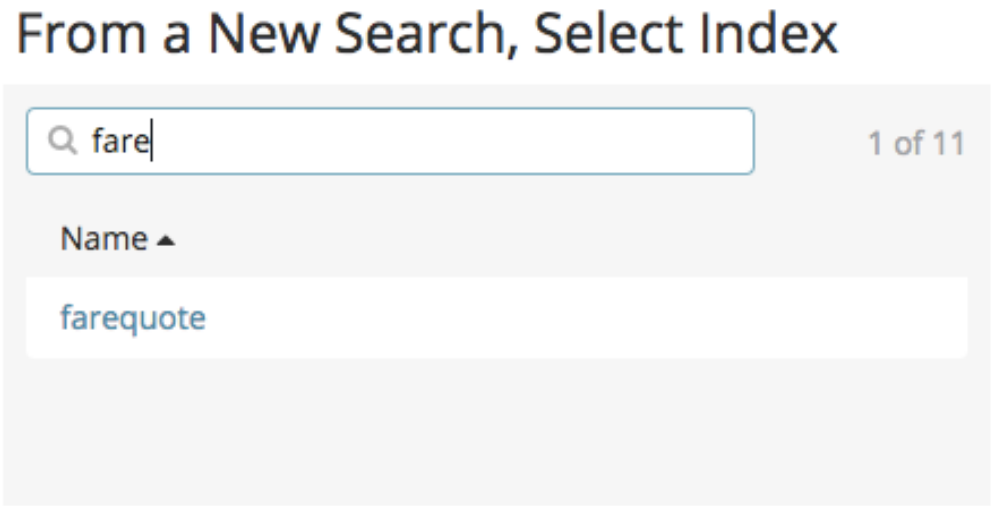


2) Choose multi metric

elastic

Steps to Complete

3) pick farequote index



elastic

Steps to Complete

4) choose
- event rate, count
- responsetime, max

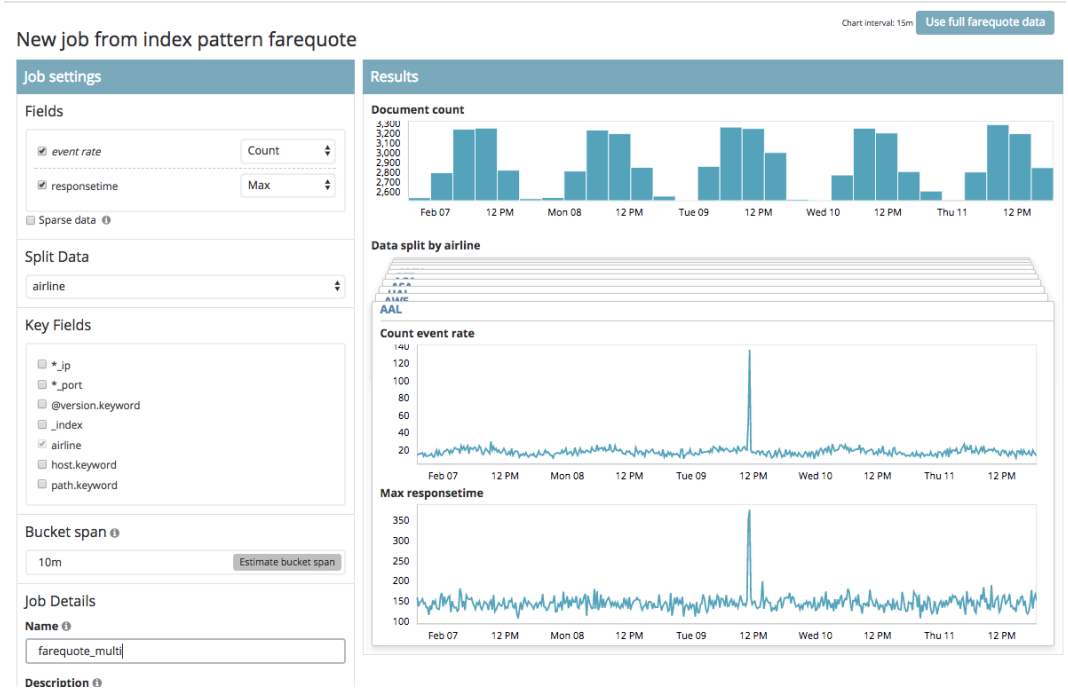
5) select 10m for bucket span

6) Split Data by airline (influencer for airline is chosen for you)

7) click “use full farequote data”

8) name job “farequote_multi”

9) click “Create Job”

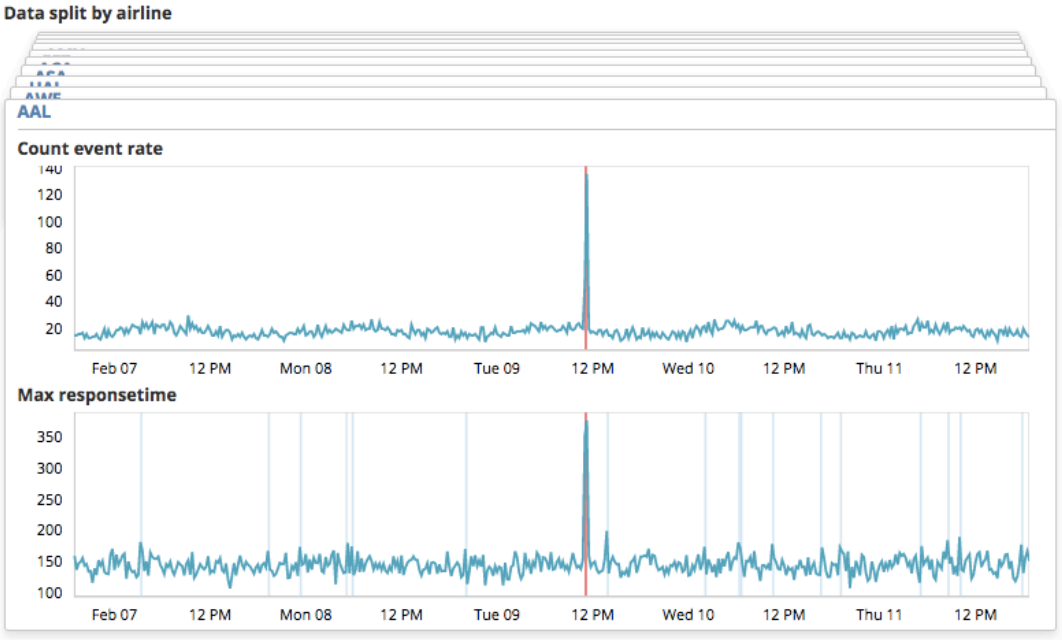


elastic

Steps to Complete

10) See animated learning

11) click “View Results”

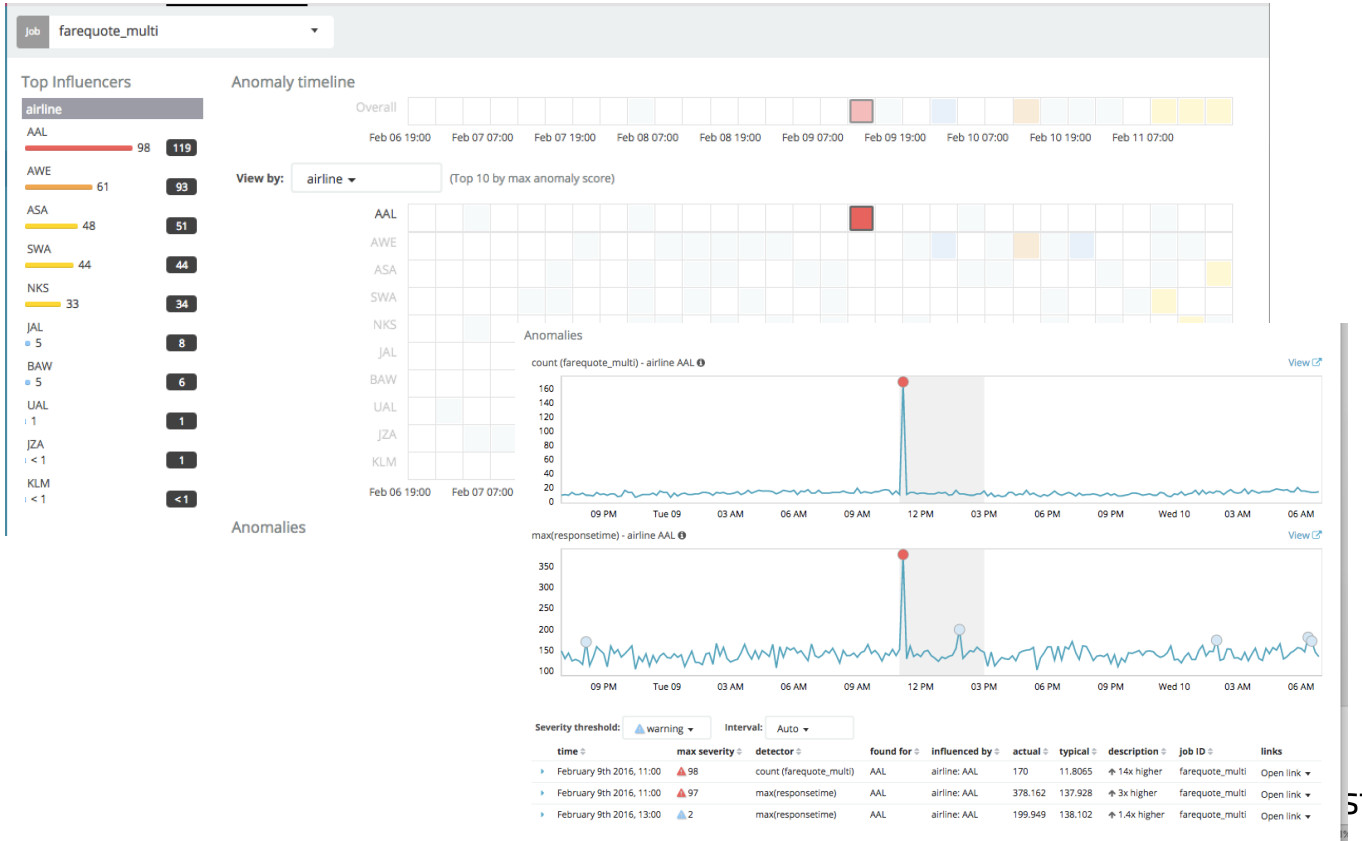


elastic

Steps to Complete

12) Result:

anomalies for AAL
in both count
and response time



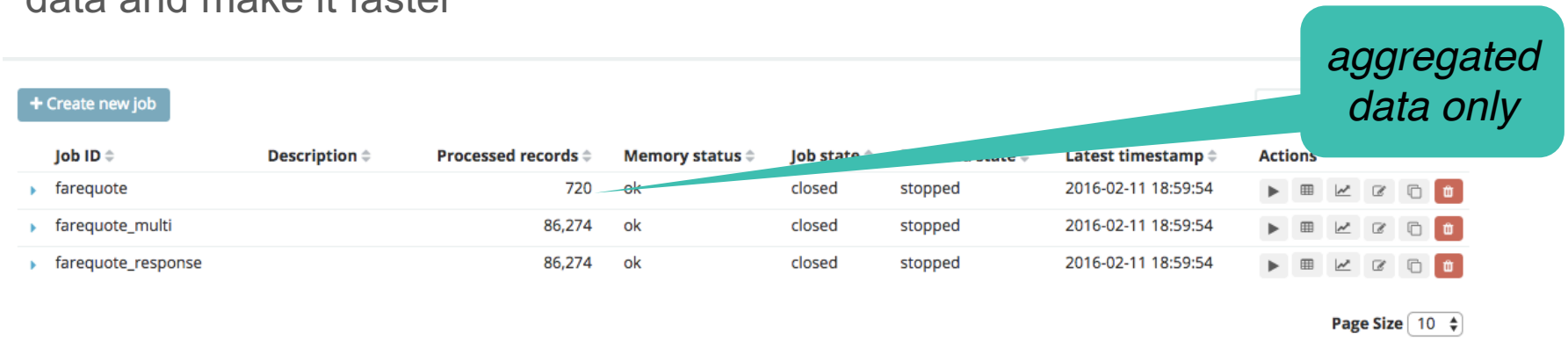
Comparison of Job Types

Advanced vs. Multi-metric Job

- End result was the same, so which one to use?
- Use simple job if don’t need lots of customization of job
  - No customization of search query for raw data or other low-level settings for the datafeed
  - No customization of features not in the UI, but in the raw job JSON (more on this later)

Single vs. Multi-metric Job

- Use single if you only have a single time series, with no reason to “split” the data
- Single metric jobs leverage Elasticsearch aggregations in the query to the data and make it faster



  - Advanced jobs can do this too, but requires manual edit to job JSON

elastic

Other Types of Analysis

Rare Analysis

- Finding items that rarely occur is also often useful
  - Rarely occurring log messages
  - Rare running process names
  - Rare connection destinations
- ML has a rare function, but it should be noted that:
  - It is relative, i.e. it takes into account the frequency of other field values, and is not an absolute measure of rarity based on, for example, the bucket length.
  - If rare was an absolute measure regardless of other field values, the result would be excessively noisy if there were many sparse field values per bucket length.
  - Therefore it works best when there are plenty of routine messages to contrast the rare ones

Example of Rare Analysis

- Use Case: Security team @ services company
- Wanted to profile typical processes on each host using netstat

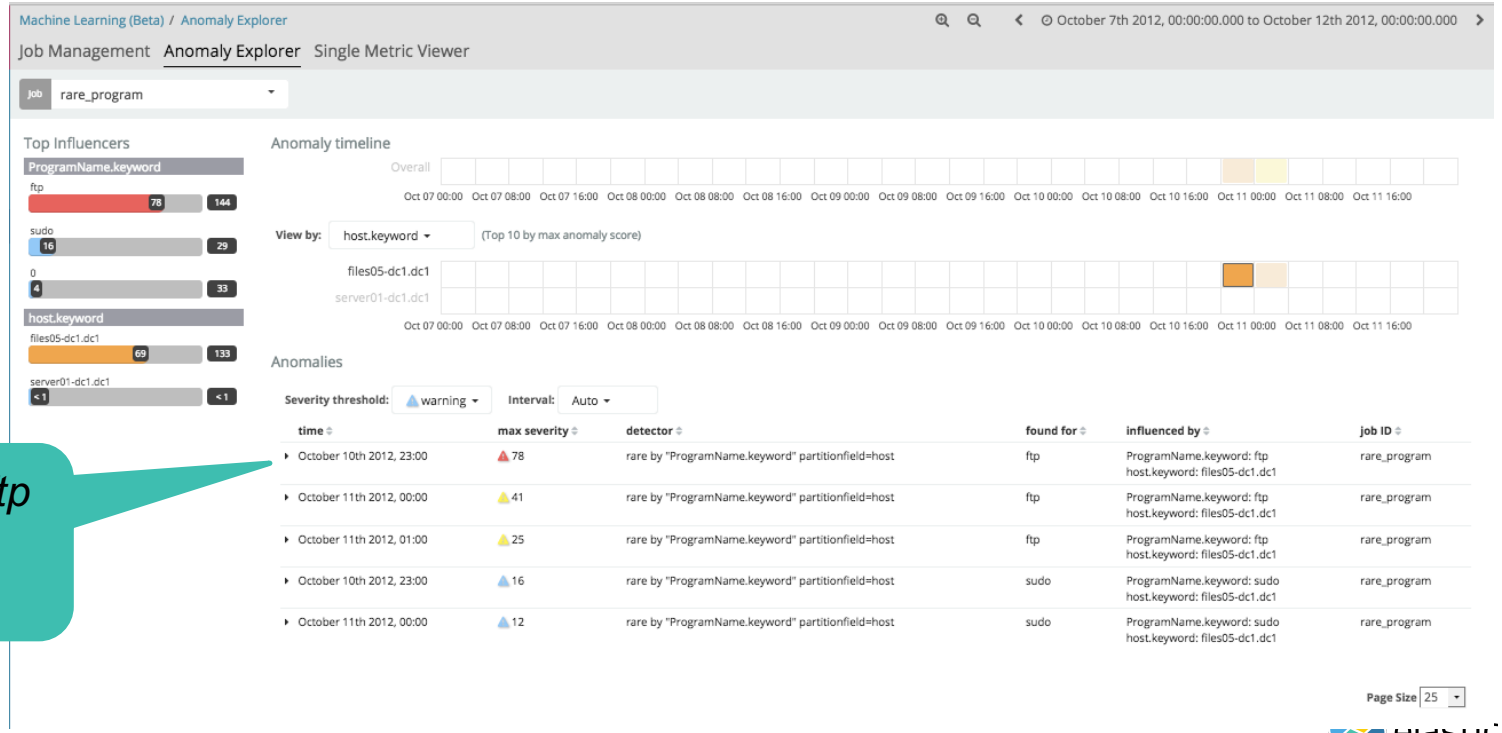
```
Active Internet connections (servers and established)
(index=netstat host="ids01-dc2" State=LISTEN (7/16/13 1:32:51AM))
Proto Recv-Q Send-Q Local Address               Foreign Address             State       PID/Program name
tcp        0      0  10.220.174.41:561         0.0.0.0:*                   LISTEN      8776/argus
tcp        0      0  0.0.0.0:22                0.0.0.0:*                   LISTEN      2033/sshd
tcp        0      0  0.0.0.0:1241              0.0.0.0:*                   LISTEN      4472/nessusd
tcp        0      0  0.0.0.0:8089              0.0.0.0:*                   LISTEN      4238/splunkd
tcp        0      0  127.0.0.1:25              0.0.0.0:*                   LISTEN      4194/master
tcp        0      0  0.0.0.0:8000              0.0.0.0:*                   LISTEN      4361/python
tcp        0      0  0.0.0.0:8834              0.0.0.0:*                   LISTEN      4472/nessusd
tcp        0      0  127.0.0.1:199             0.0.0.0:*                   LISTEN      2022/snmpd
```

- Goal was to identify rare processes that “start up and communicate” for each host, individually

elastic

Example of Rare Analysis

detector: *rare by ProgramName partition_field=host*



ProgramName: ftp
host: files05
rarely occurs

Categorization

- Application log events are often unstructured and contain variable data
- Example:
  - 07 Oct 2014 11:02:12 BST [qtp1362038001-155]
    INFO com.prelert.rs.resources.Data - Decompressing post data in job
    = 20141007104700-00016
- Categorization uses machine learning to observe the static parts of the message, cluster similar messages together, and classify (categorize) them in to message categories.
- Knowing the type of the message enables anomaly detection based on count or rarity of the message type.

Categorization Example

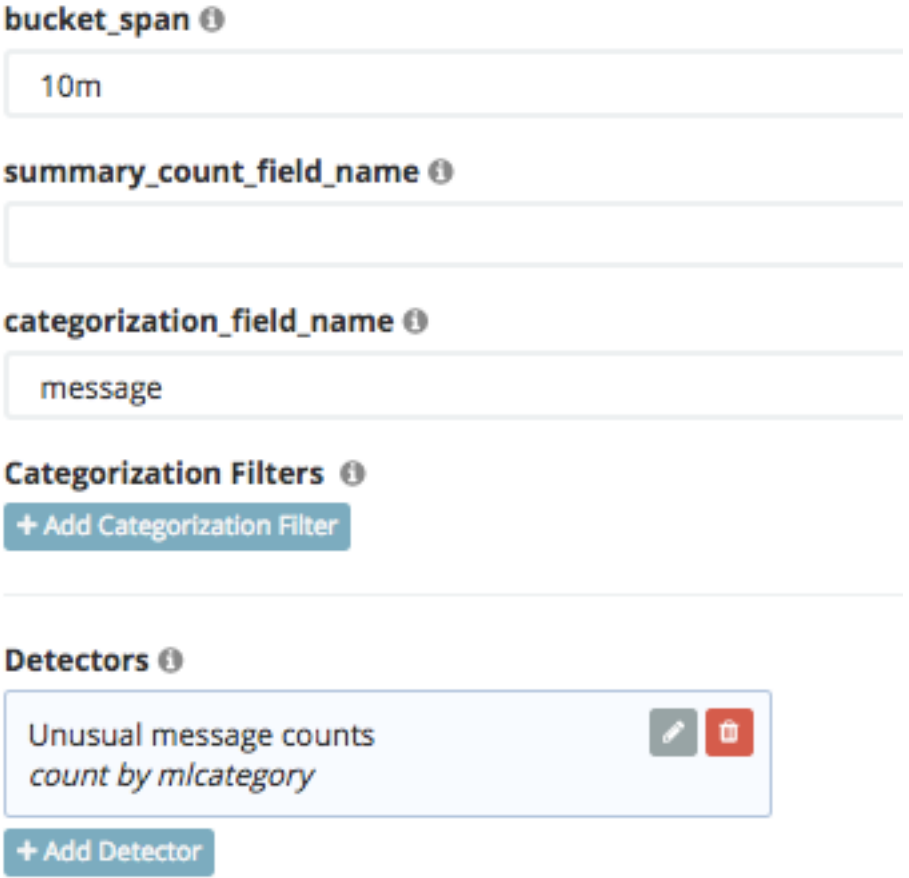
- Given the following log messages, indexed:


```
{
  "_index": "it_ops_logs",
  "_type": "logs",
  "_id": "AVkDPcnTt9AfBggy7XyS",
  "_score": 1,
  "_source": {
    "@timestamp": "2016-02-08T15:21:06.000Z",
    "message": "Opening Database = DRIVER={SQL Server};SERVER=127.0.0.1;network=dbmssocn;address=127.0.0.1 1433;DATABASE=svc_prod;;Trusted_Connection=Yes;AnsiNPW=No;dbhost=dbserver.a
  }
},
{
  "_index": "it_ops_logs",
  "_type": "logs",
  "_id": "AVkDPcnTt9AfBggy7XyU",
  "_score": 1,
  "_source": {
    "@timestamp": "2016-02-08T15:21:23.000Z",
    "message": "REC Not INSERTED [DB TRAN] Table;dbhost=dbserver.acme.com;physicalhost=esxserver1.acme.com;vmhost=app1.acme.com\r"
  }
},
{
  "_index": "it_ops_logs",
  "_type": "logs",
  "_id": "AVkDPcnmt9AfBggy7X0P",
  "_score": 1,
  "_source": {
    "@timestamp": "2016-02-02T07:36:00.000Z",
    "message": "Using: sssvcdbj1.acme.com!svc_prod#uid=dbadmin1;pwd=#####;dbhost=dbserver.acme.com;physicalhost=esxserver1.acme.com;vmhost=app1.acme.com\r"
  }
```


elastic

Categorization Example

- Configure an ML job to use:
  - “message” as the categorization_field_name
  - there will be a new, “magic” field called “mlcategory” that is dynamically created by ML to group similar messages together

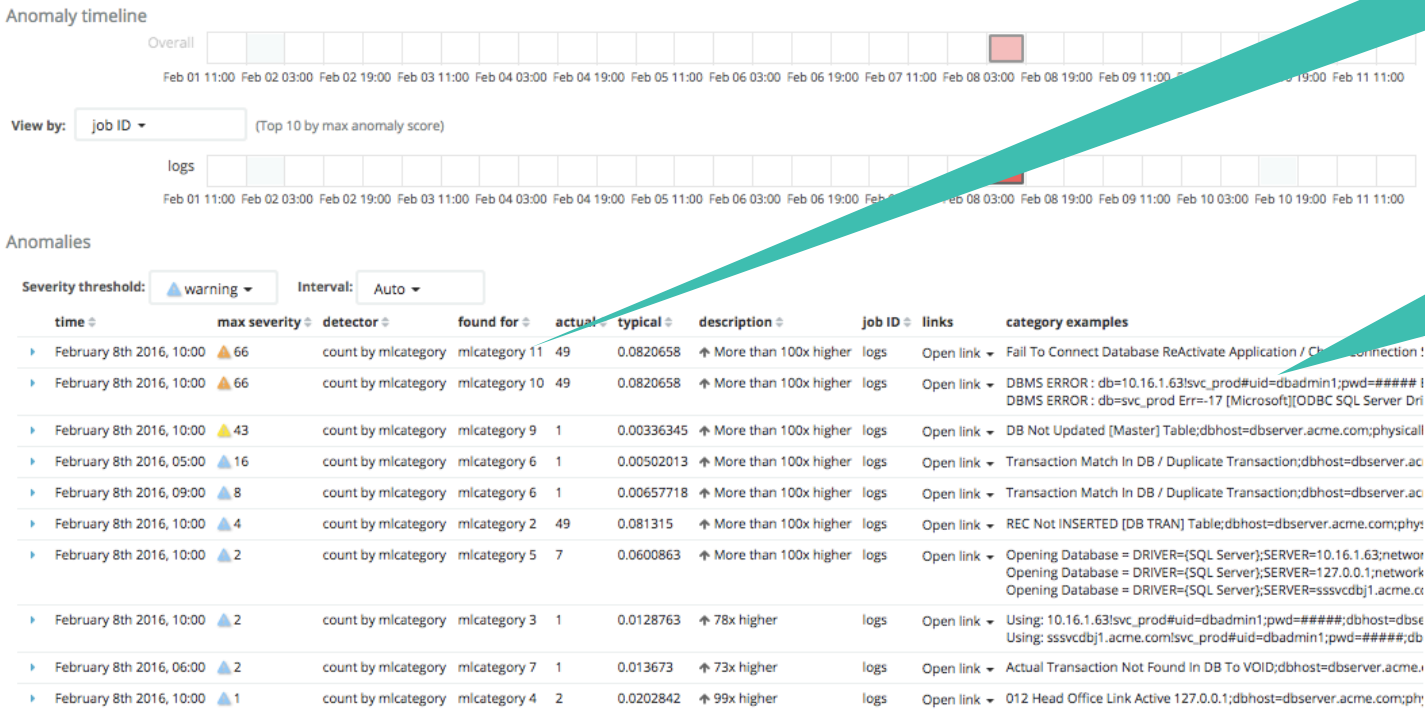


elastic

Categorization Example – count by mlcategory

category name

example matching log messags



elastic

Population Analysis

- We have already agreed that there are two relevant definitions of anomalousness
  - change with respect to itself as a function of time (temporal)
  - relative difference compared to peers within a population

- If you want Population Analysis, you must select an “over_field_name”
  - The field chosen defines the population
- If “over_field_name” is not chosen, then population analysis is NOT invoked and thus only temporal analysis is invoked

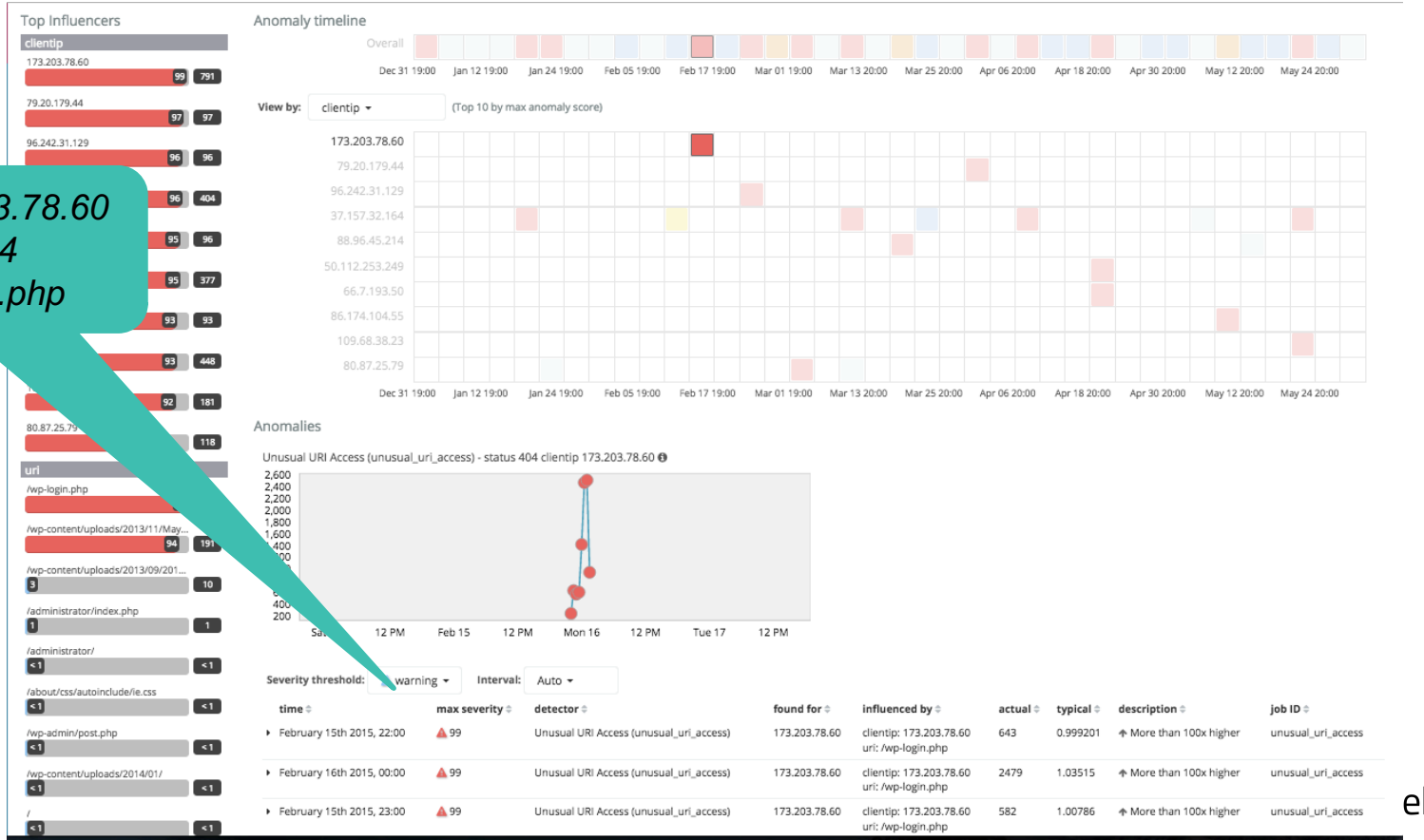
Population Analysis

- Useful when:
  - Entities have high-cardinality (i.e. external IP addresses)
  - Data for specific entities may be sparse in time (individual customers placing orders)
  - The behavior of the population as a whole is mostly homogeneous
- Not appropriate when:
  - Members of the population have vastly different behavior inherently.

Population Analysis

detector: *high_count over clientip partition_field_name=status*

clientip: 173.203.78.60
status: 404
uri: /wp-login.php



elastic

Lab 4: Multi-Job Analysis

Steps to Complete

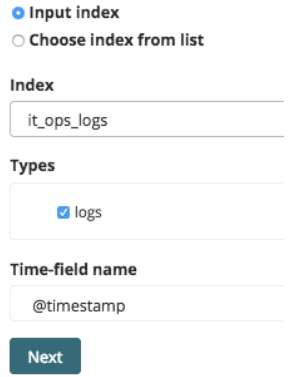
- Using the “it_ops_logs” data set:
  - Create a “count by mlcategory” job for the log events
    - use “message” as the categorization_field_name

- Using the “it_ops_metrics” data set:
  - Create a “mean(metricvalue) by metricname” job for the metrics

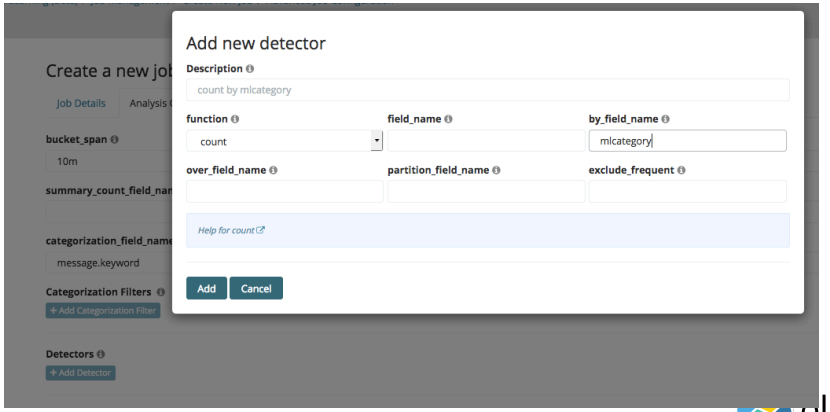
- View both jobs overlaid in the Explorer View

Steps to Complete

- Answer
  - For index:it_ops_logs
    - create an advanced job



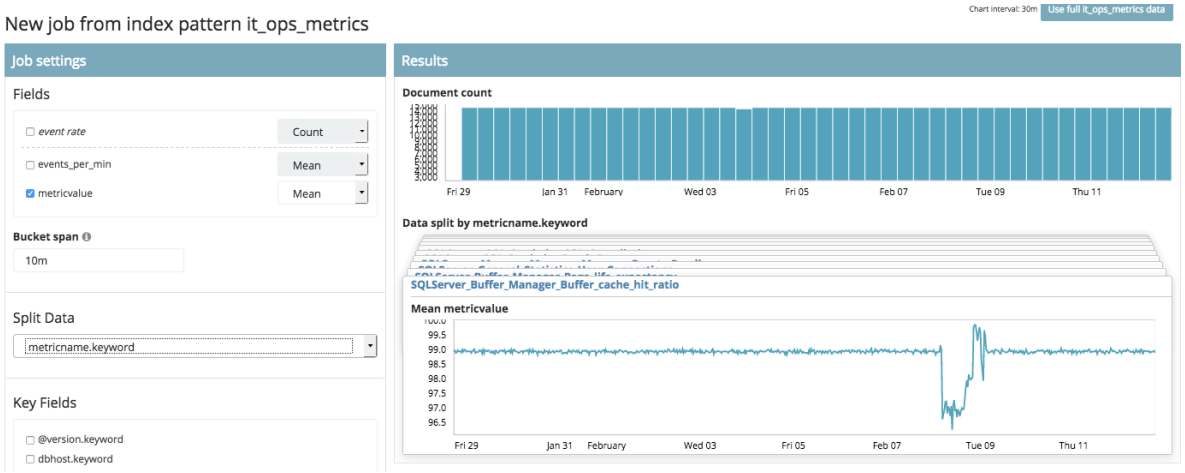
    - make sure you choose “message” for categorization_field_name
    - detector is:
count with by_field_name of “mlcategory”



elastic

Steps to Complete

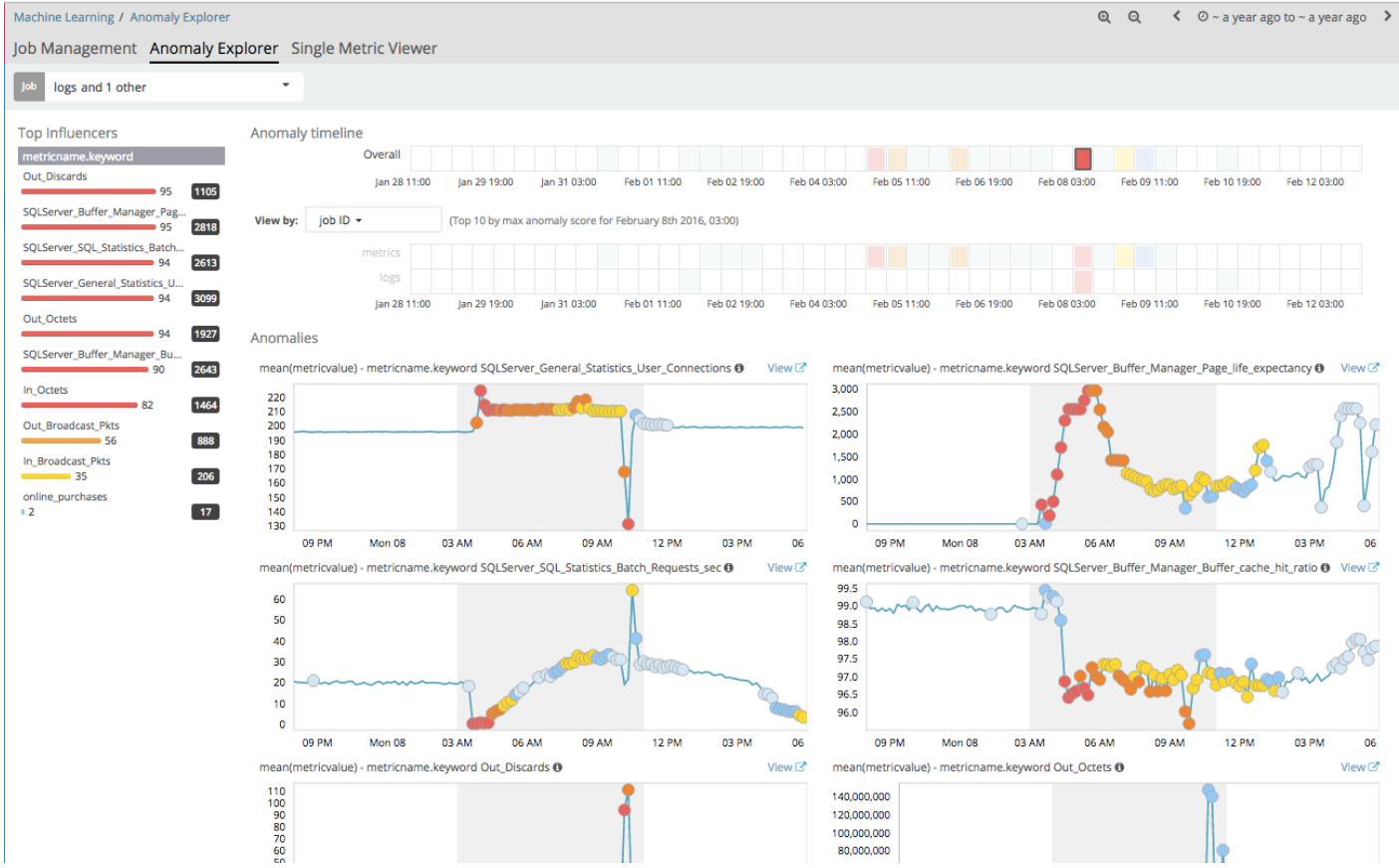
- Answer
  - For index:it_ops_metrics
    - create multi-metric job
    - mean of metricvalue split on metricname.keyword



elastic

Steps to Complete

- Your goal is to get this View:

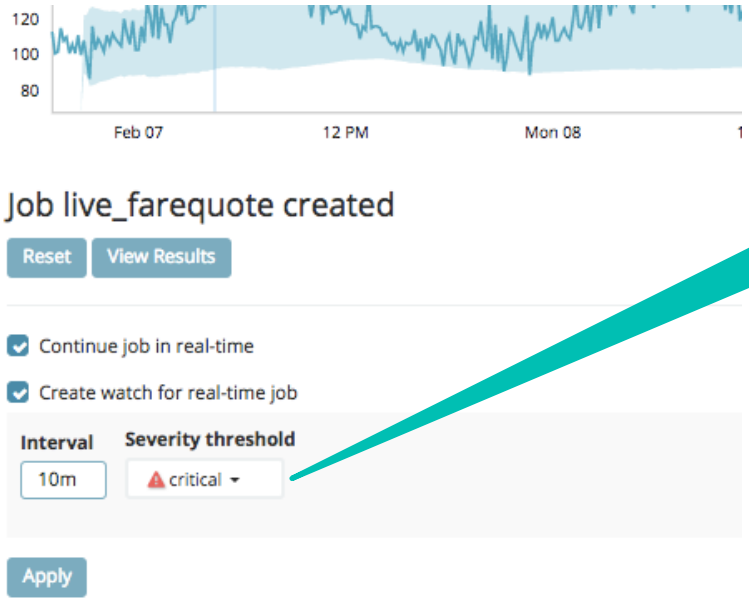


Advanced Topics

Alerting

Alerting on Single/Multi Metric Jobs

- After job configuration, see option for creating a “watch” on the live data:

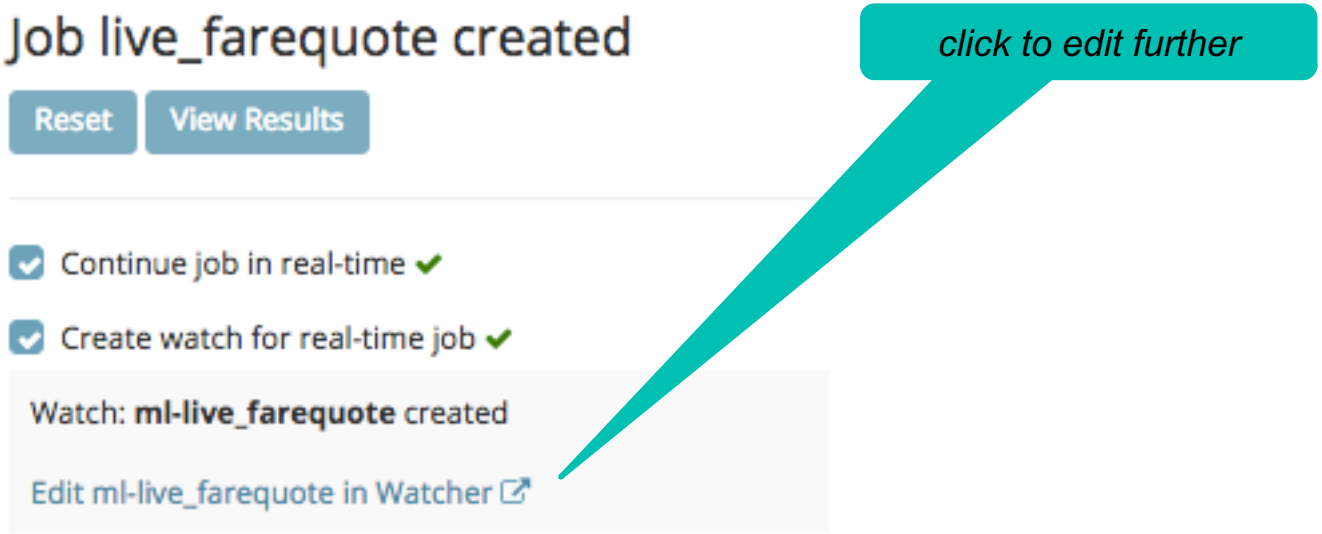


The minimum severity to alert upon

elastic

Alerting on Single/Multi Metric Jobs

- Clicking “Apply”:

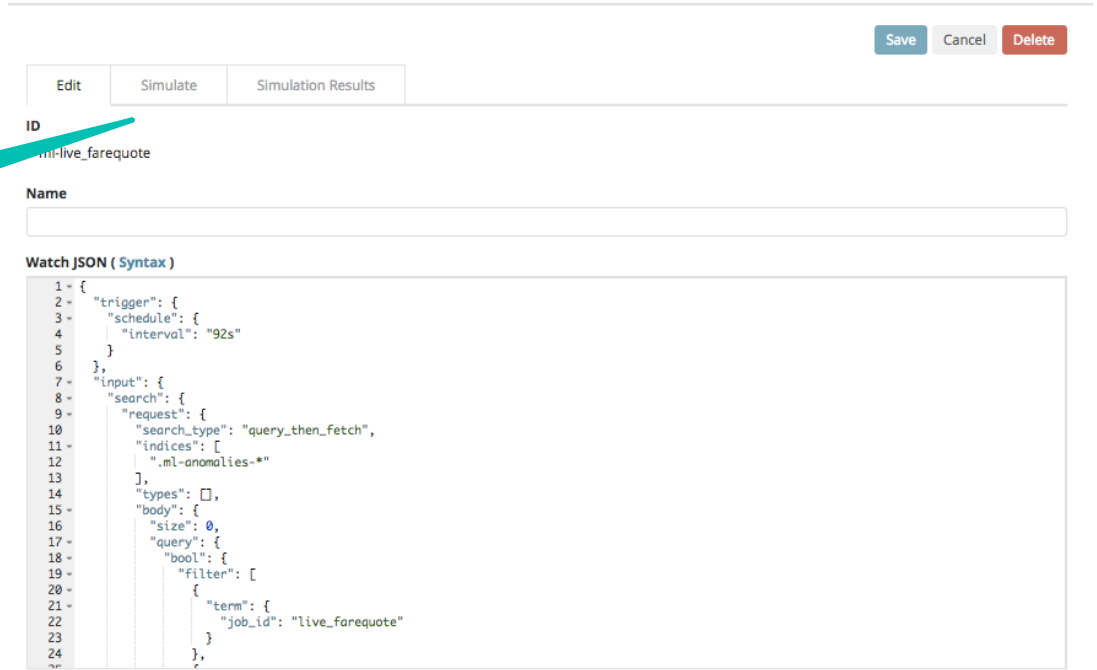


elastic

Alerting on Single/Multi Metric Jobs

- Now can edit Watch JSON to further customize things like action, etc.

or to test/simulate!



elastic

Under the Hood: Connecting ML to Alerting

- For all jobs - all anomalies are stored in `.ml-anomalies-*` indicies
- Every job also creates an index alias called `.ml-anomalies-<jobname>`
- You can configure Watches to query this index in 3 different ways:
  - "bucket" level
    - Answers: *How unusual was the job in a particular bucket of time?*
    - Essentially an aggregated anomaly score – useful for rate-limited alerts
  - "record" level
    - Answers: *What individual anomalies are present in a range of time?*
    - All the detailed anomaly information, but records can be numerous in big data
  - "influencers"
    - Answers: *What are the most unusual entities in a range of time?*
- *See https://github.com/elastic/examples/tree/master/Alerting/ml_examples*

Example – Querying for bucket results

- If we execute the query:

```
#farequote results bucket search
GET .ml-anomalies-*/_search
{
    "query": {
                "bool": {
                  "filter": [
                      { "range" : { "timestamp" : { "gte": "now-2y" } } },
                      { "term" :  { "result_type" : "bucket" } },
                      { "term" :  { "job_id" : "farequote_response" } },
                      { "range" : {"anomaly_score" : {"gte" : "75"}}}

                  ]
                }
    }
}
```

Example – An anomaly bucket

- We get the following:
  - `anomaly_score`: anomaly score of a bucket at time = `timestamp`
  - at this point, can also query “records” or “influencers” at this time for more detail
    - querying for records or influencers right away could lead to a lot of results for jobs with many detectors or entities in by/partition/over fields


```
{
  "took": 48,
  "timed_out": false,
  "_shards": {
    "total": 25,
    "successful": 25,
    "failed": 0
  },
  "hits": {
    "total": 1,
    "max_score": 0,
    "hits": [
      {
        "_index": ".ml-anomalies-shared",
        "_type": "result",
        "_id": "farequote_response_1455034500000_300",
        "_score": 0,
        "_source": {
          "job_id": "farequote_response",
          "timestamp": 1455034500000,
          "anomaly_score": 90.7,
          "bucket_span": 300,
          "initial_anomaly_score": 85.08,
          "record_count": 2,
          "event_count": 179,
          "is_interim": false,
          "bucket_influencers": [
            {
              "job_id": "farequote_response",
              "result_type": "bucket_influencer",
              "influencer_field_name": "airline",
              "initial_anomaly_score": 85.08,
              "anomaly_score": 90.7,
              "raw_anomaly_score": 43.1559,
              "probability": 2.661e-44,
              "timestamp": 1455034500000,
              "bucket_span": 300,
              "sequence_num": 5,
              "is_interim": false
            },
            {
              "job_id": "farequote_response",
              "result_type": "bucket_influencer",
              "influencer_field_name": "bucket_time",
```

A Simple Watch

- Trigger at a rate equal to or faster than bucket_span
- Define the input section as “search”
  - obviously make the “range” more “real-time” by using “now-1m”. It is longer in example because of date range of demo data.
- For the “condition” section, check to see if you get any payload hits
- For actions, can obviously do things more sophisticated than log (such as email, etc.)

```
# basic bucket watch
POST _xpack/watcher/watch/_execute
{
    "watch": {
    "trigger" : {
      "schedule" : { "interval" : "5m" }
    },
    "input" : {
      "search" : {
        "request" : {
          "indices" : [ ".ml-anomalies-*" ],
          "body" : {
            "query": {
              "bool": {
                "filter": [
                    { "range" : { "timestamp" : { "gte": "now-2y" } } },
                    { "term" :  { "result_type" : "bucket" } },
                    { "term" :  { "job_id" : "farequote_response" } },
                    { "range" : {"anomaly_score" : {"gte" : "75"}}}

                ]
              }
            }
          }
        }
      }
    },
    "condition" : {
      "compare" : { "ctx.payload.hits.total" : { "gt" : 0 }}
    },
    "actions" : {
      "log" : {
        "logging" : {
          "text" : "Anomalies:\n{{#ctx.payload.hits.hits}}score={{_source.anomaly_s
core}} at time={{_source.timestamp}}\n{{/ctx.payload.hits.hits}}"
        }
      }
  }
    }
  }
```

Watch Output

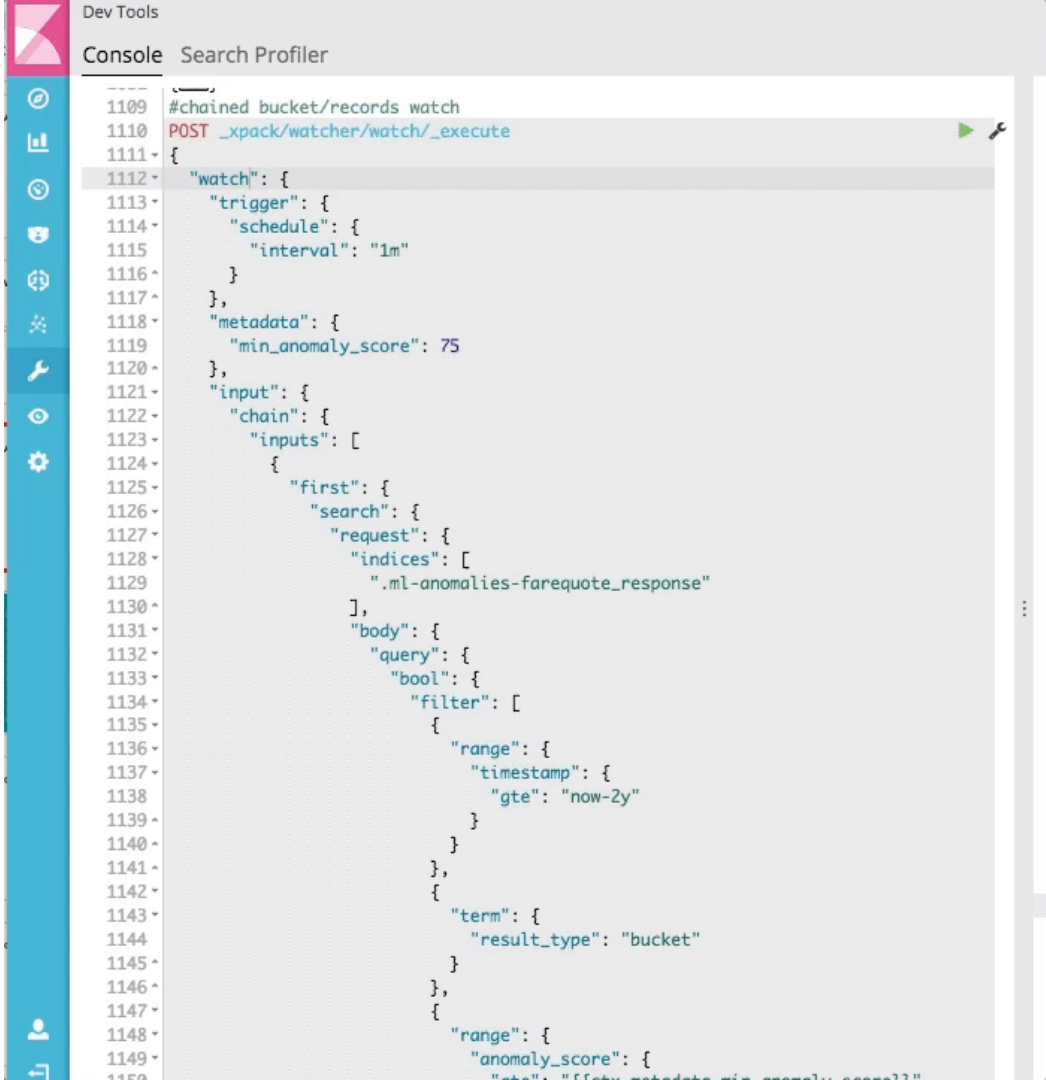
- The following is the resolved output of the watch, as seen in elasticsearch.log:

```
[2017-06-06T09:37:56,604][INFO ][o.e.x.w.a.l.ExecutableLoggingAction] [GJgDody] Anomalies:score=90.7 at time=1455034500000
```

A Chained Watch

- Same idea, but use “chained” inputs to make 2 queries:
  - First to “bucket” level to test aggregated anomaly score
  - Second to “records” for that bucket time to return the records, if the bucket score was worthy
  - Output:

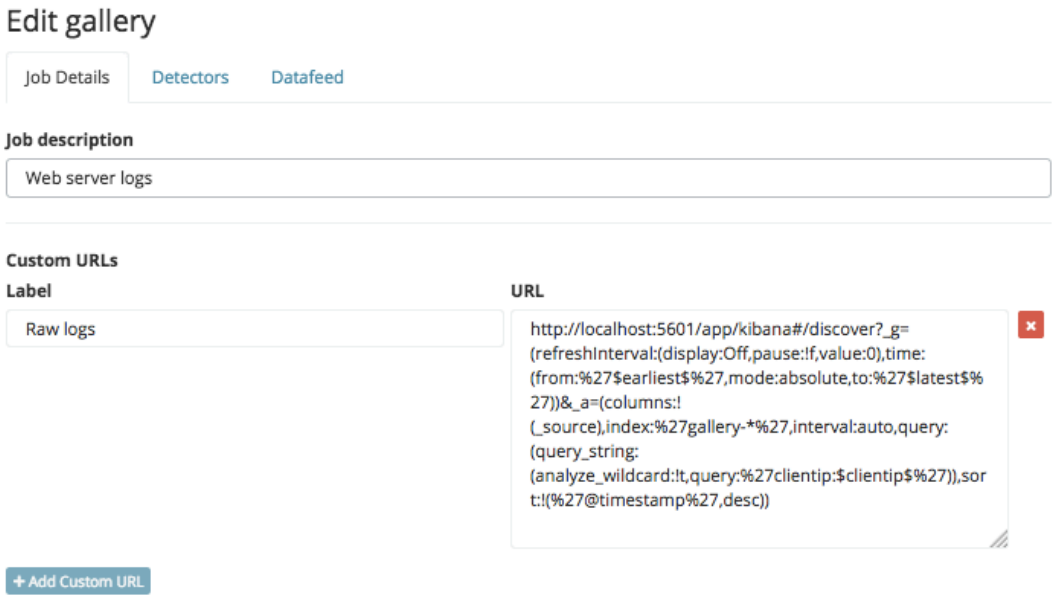
Anomaly of score=90.7 at 2016-02-09 11:15:00 Influenced by:
airline=AAL: score=95.4659, responsetime=296.19ms (typical=99.2962ms)
airline=AWE: score=0.00871093, responsetime=19.1644ms (typical=19.9918ms)



Custom Drilldowns

Custom Drilldown URLs

- Each job allows one or more custom URLs to facilitate contextual linking to another location, either internally to kibana or externally to any other site
- Example:



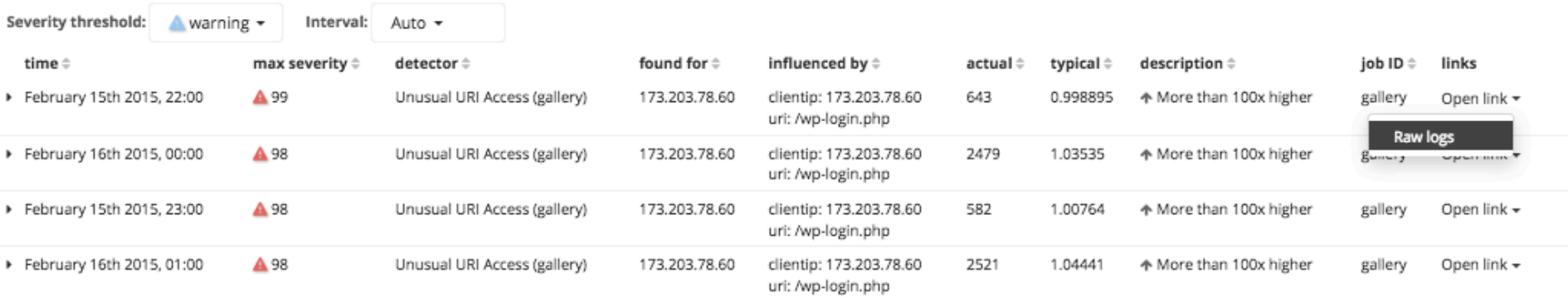
elastic

Custom Drilldown URLs

- Text of Example
  - `http://localhost:5601/app/kibana#/discover?_g=(refreshInterval:(display:Off,pause:!f,value:0),time:(from:%27$earliest$%27,mode:absolute,to:%27$latest$%27))&_a=(columns:!(_source),index:%27gallery-*%27,interval:auto,query:(query_string:(analyze_wildcard:!t,query:%27clientip:$clientip$%27)),sort:!(%27@timestamp%27,desc))`
- Notice the use of substitution variables to pass the context of time range and even values of fields to the URL

Custom Drilldown URLs

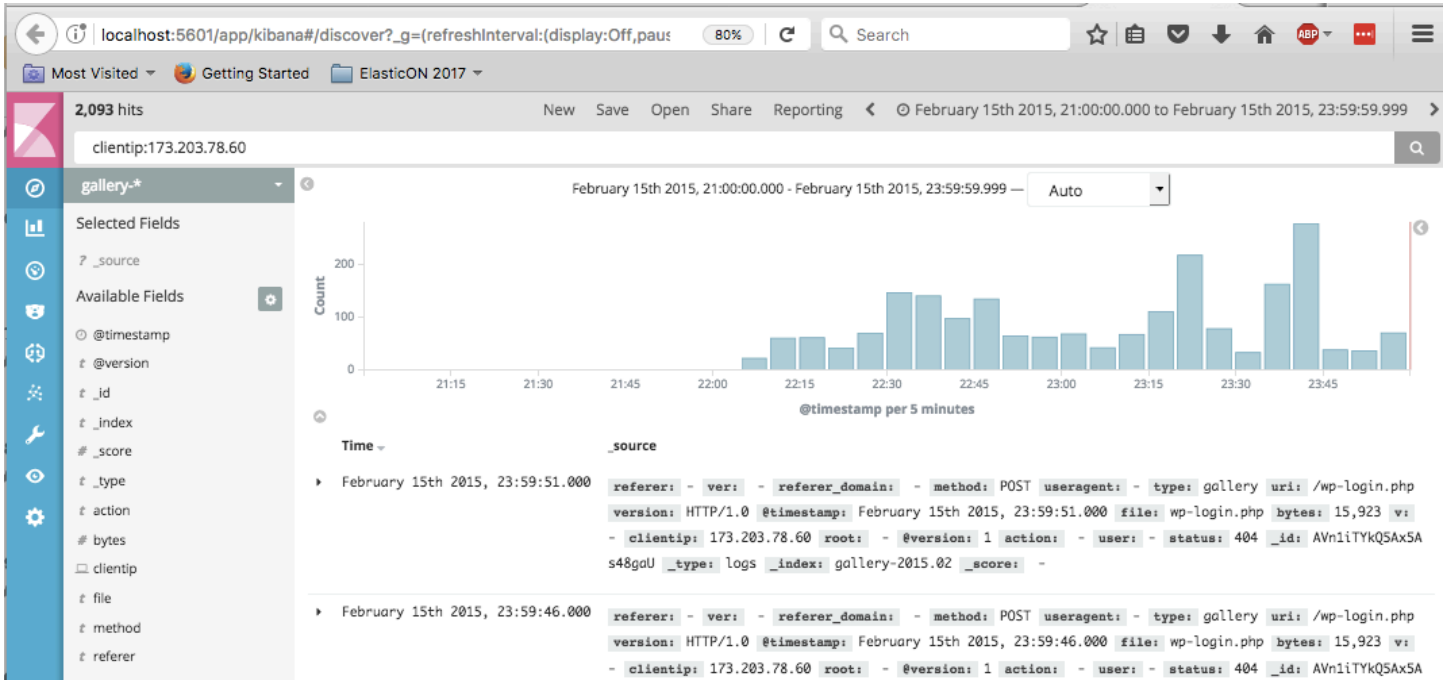
- In Explorer view, there is now a “links” column
- When selected, the date range and in this case, the value of “clientip:173.203.78.60” is passed to the URL



elastic

Custom Drilldown URLs

- Date filters and filter on “clientip” is applied

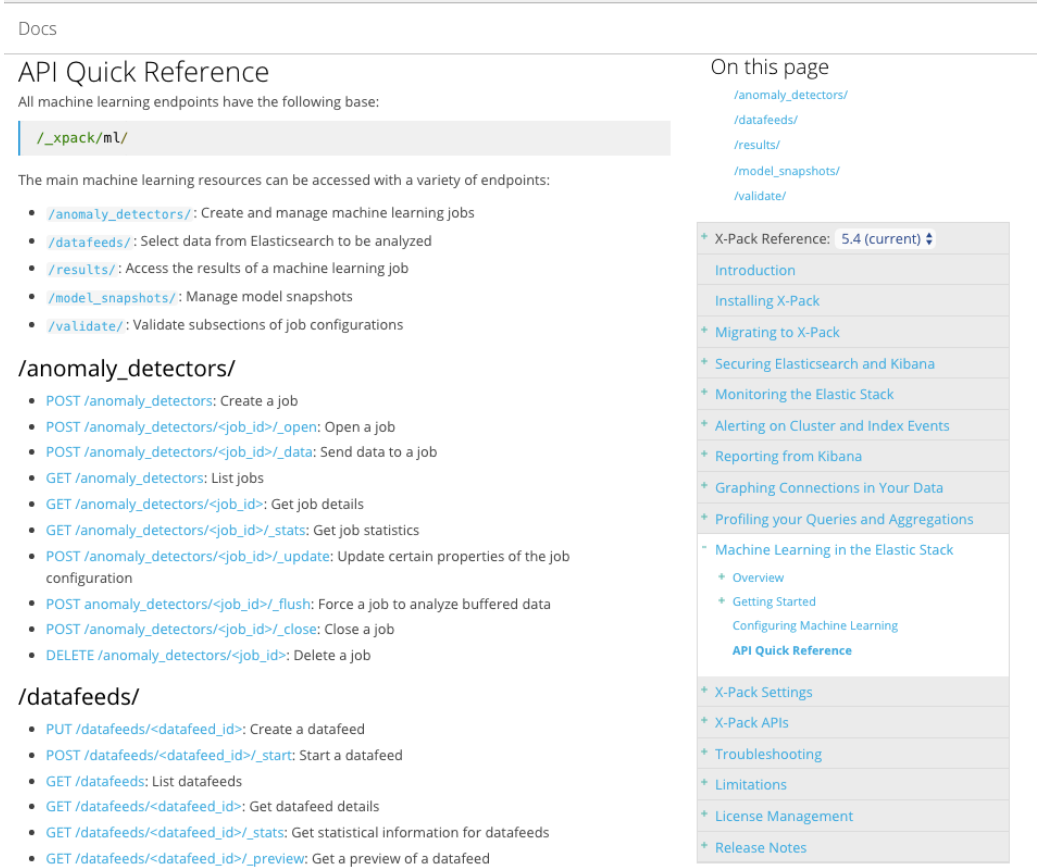


elastic

API Control

Controlling ML via API

- Full documentation of API available at https://www.elastic.co/guide/en/x-pack/current/ml-api-quickref.html



Docs

API Quick Reference

All machine learning endpoints have the following base:

```
/_xpack/ml/
```

The main machine learning resources can be accessed with a variety of endpoints:

- `/anomaly_detectors/`: Create and manage machine learning jobs
- `/datafeeds/`: Select data from Elasticsearch to be analyzed
- `/results/`: Access the results of a machine learning job
- `/model_snapshots/`: Manage model snapshots
- `/validate/`: Validate subsections of job configurations

/anomaly_detectors/

- POST /anomaly_detectors: Create a job
- POST /anomaly_detectors/<job_id>/_open: Open a job
- POST /anomaly_detectors/<job_id>/_data: Send data to a job
- GET /anomaly_detectors: List jobs
- GET /anomaly_detectors/<job_id>: Get job details
- GET /anomaly_detectors/<job_id>/_stats: Get job statistics
- POST /anomaly_detectors/<job_id>/_update: Update certain properties of the job configuration
- POST anomaly_detectors/<job_id>/_flush: Force a job to analyze buffered data
- POST /anomaly_detectors/<job_id>/_close: Close a job
- DELETE /anomaly_detectors/<job_id>: Delete a job

/datafeeds/

- PUT /datafeeds/<datafeed_id>: Create a datafeed
- POST /datafeeds/<datafeed_id>/_start: Start a datafeed
- GET /datafeeds: List datafeeds
- GET /datafeeds/<datafeed_id>: Get datafeed details
- GET /datafeeds/<datafeed_id>/_stats: Get statistical information for datafeeds
- GET /datafeeds/<datafeed_id>/_preview: Get a preview of a datafeed

On this page

/anomaly_detectors/
/datafeeds/
/results/
/model_snapshots/
/validate/

X-Pack Reference: 5.4 (current)
Introduction
Installing X-Pack
Migrating to X-Pack
Securing Elasticsearch and Kibana
Monitoring the Elastic Stack
Alerting on Cluster and Index Events
Reporting from Kibana
Graphing Connections in Your Data
Profiling your Queries and Aggregations
Machine Learning in the Elastic Stack
Overview
Getting Started
Configuring Machine Learning
API Quick Reference
X-Pack Settings
X-Pack APIs
Troubleshooting
Limitations
License Management
Release Notes

Example API Control

- All major operations are available via API
  - Create/Delete jobs and datafeeds
  - Job control (start/stop)
- Plus actions that are ONLY available via API
  - Model snapshot/restore

```
printf "\n\n== Creating job... \n"
curl -u elastic:changeme -s -X PUT -H 'Content-Type: application/json' ${JOBS}/${JOB_ID}?pretty -d '{
    "description" : "Unusual responsetimes by airlines",
    "analysis_config" : {
        "bucket_span": "5m",
        "detectors" :[{"function":"max", "field_name":"responsetime","by_field_name":"airline"}],
        "influencers" : [ "airline" ]
    },
    "data_description" : {
      "time_field":"@timestamp"
    }
}'

printf "\n\n== Creating datafeed... \n"
curl -u elastic:changeme -s -X PUT -H 'Content-Type: application/json' ${DATAFEEDS}/datafeed-${JOB_ID}?pretty -d '{
      "job_id" : "'"$JOB_ID"'",
      "indexes" : [
        "farequote"
      ],
      "types" : [
        "responsetime"
      ],
      "scroll_size" : 1000
}'


printf "\n\n== Opening job for ${JOB_ID}... "
curl -u elastic:changeme -X POST ${JOBS}/${JOB_ID}/_open

printf "\n\n== Starting datafeed-${JOB_ID}... "
curl -u elastic:changeme -X POST "${DATAFEEDS}/datafeed-${JOB_ID}/_start?start=1970-01-02T10:00:00Z&end=2017-01-01T00:00:00Z"

printf "\n\n== Finished ==\n\n"
```

elastic

The End

Extra Slides

bucket_span

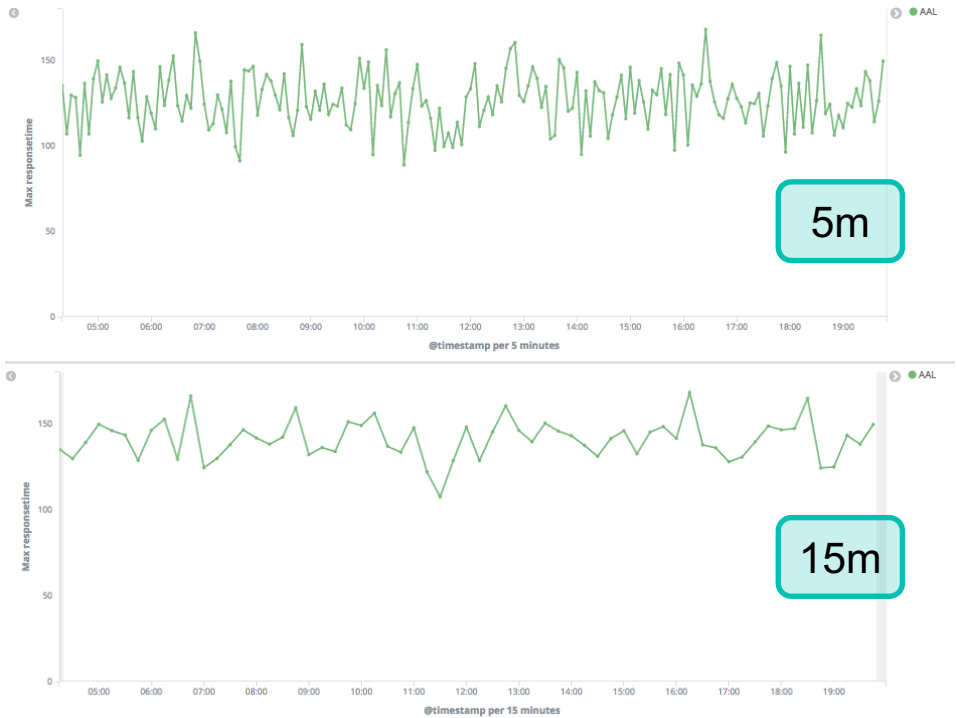
- Similar to a “Date Histogram” aggregation for an x-axis visualization in Kibana
- Defines the resolution (in time) that the data should be aggregated to

Date Histogram

elastic

What will bucket_span control?

- Granularity of data being analyzed
- Ultimately defines the volume of data analyzed per bucket
- How often the data is queried, when running in “real-time”



elastic

bucket_span: Rules of Thumb

- If you have “sampled” data (i.e. monitoring data every X minutes) then you could make bucket_span=X
- If you have arbitrary-rate data (i.e. log events), then pick a bucket_span that balances
  - timeliness (time to analyze/alert)
  - enough time to count, sum, etc. those events up to make the number of samples per bucket meaningful, given the velocity of data
- In other words, pick a bucket_span that’s approximately equivalent to the duration of an anomaly that you would care about.

Split Analysis?

- Splitting the data for parallel analysis by a categorical field is very useful
- If you want to “hard split” the analysis, select an “partition_field_name”
  - The field chosen should have < 10,000 distinct values, in general
  - Each instance of the field is like an independent variable
- If you want a “soft split”, select a “by_field_name”
  - The field chosen should have <100,000 distinct values, in general
  - More appropriate for attributes of an entity (dependent variables)

By vs. Partition

- Both methods split data to establish separate baselines.
- Can be applied together in one detector.

- **By**
  - Creates attributes in existing model
  - Easier to scale
  - Best used to find anomalies in system as a whole
  - Scoring considers history of other by-fields
- **Partition**
  - Establishes different models
  - Requires more memory to store models
  - Best used to find anomalies in individual behavior
  - Scoring is more independent

Complex Detector Example

- high_count by ErrorCode over Host partition_field=App
  - Partition: App
    - Split the data into separate data sets for each App. Create a model (and population) for each.
  - Over: Host
    - For each App, create a population model out of all the Hosts and detect when a Host stands out.
  - By: ErrorCode
    - When modeling and evaluating Hosts, baseline each ErrorCode separately.
  - Function: High_Count
    - Measure the number of times an ErrorCode is observed in the time bucket.