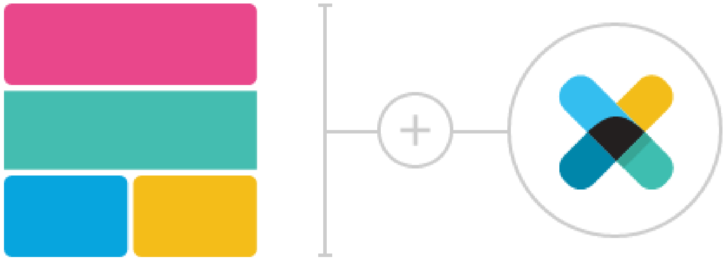
# X-Pack Machine Learning : Workshop

**Rich Collier**
Solutions Architect
@richcollier

# X-Pack

Extensions for the Elastic Stack



| | | |
|---|---|---|
| **Security** | **Alerting** | **Monitoring** |
| **Reporting** | **Graph Analytics** | **Machine Learning** |

# Anomaly Detection : Concept of Anomaly

elastic

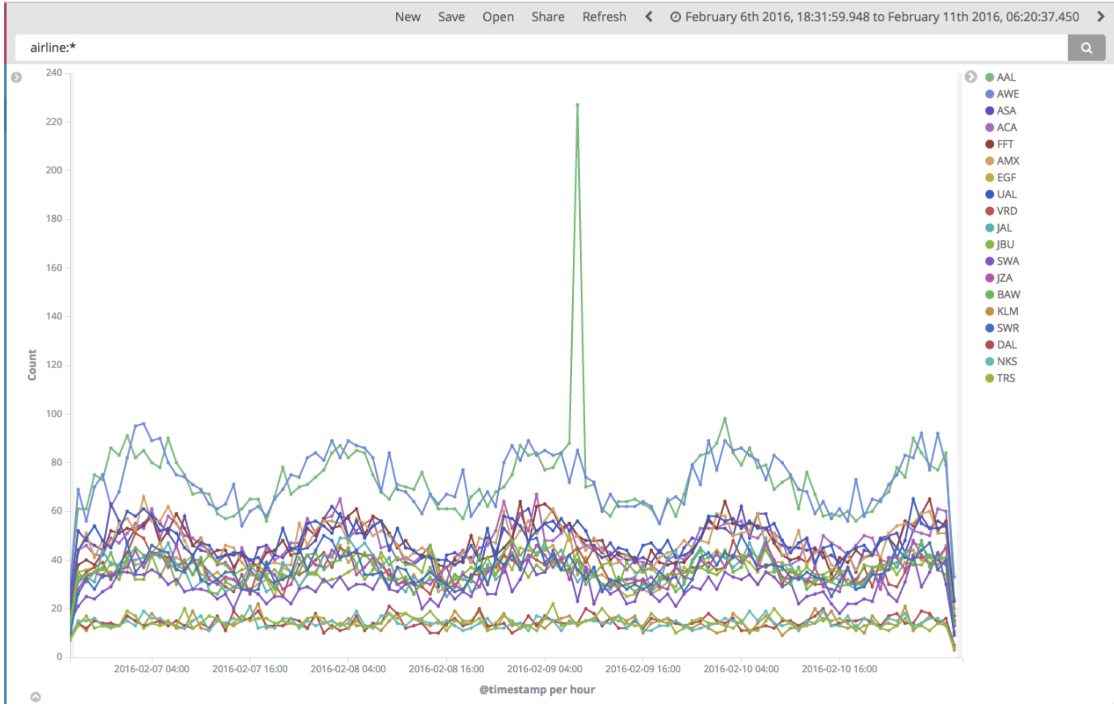
# Terminology

- ***Machine Learning***
  - Broad term, but X-Pack Machine Learning is automated anomaly detection for time-series data (for now).
- ***Anomaly Detection (not “bad activity”)***
  - 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
- ***Distributed 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?

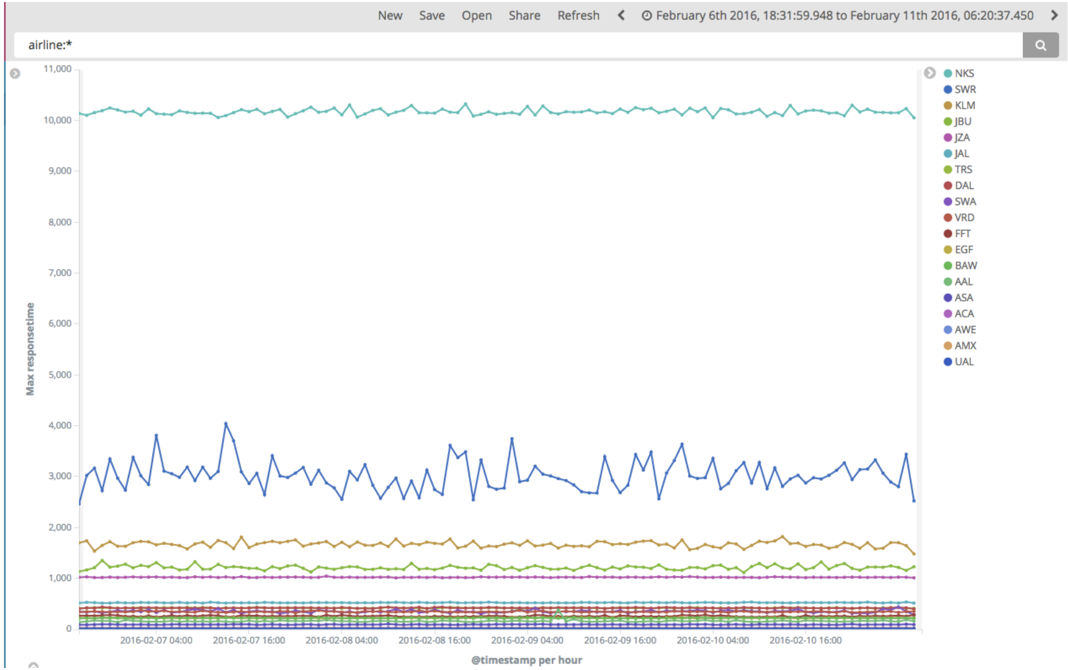
Why?



# What is “Abnormal”?

What’s abnormal here?

Why?




elastic

# What is “Abnormal”?

What’s abnormal here?

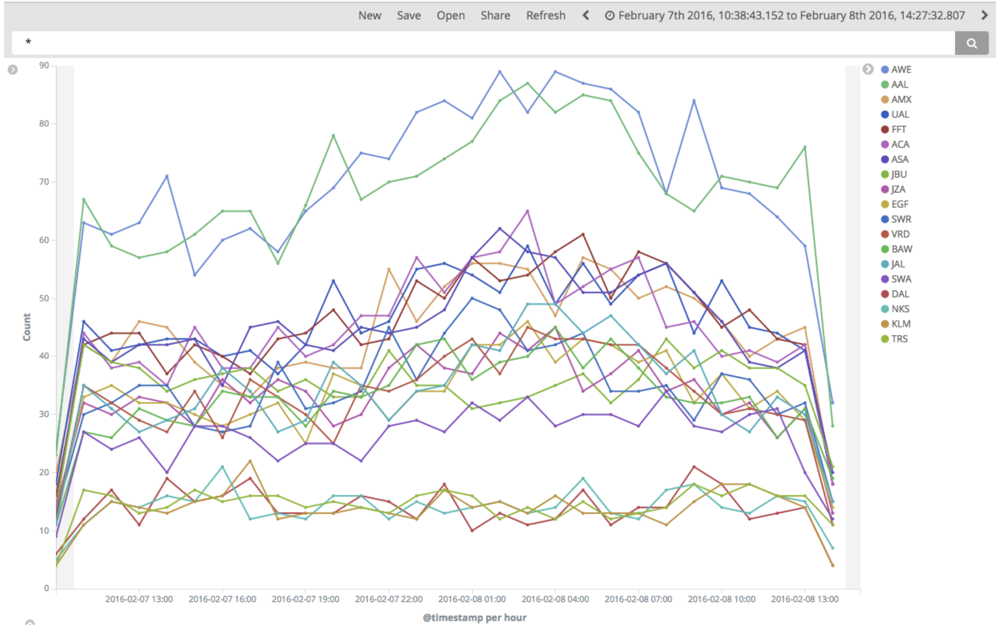
Why?

# 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

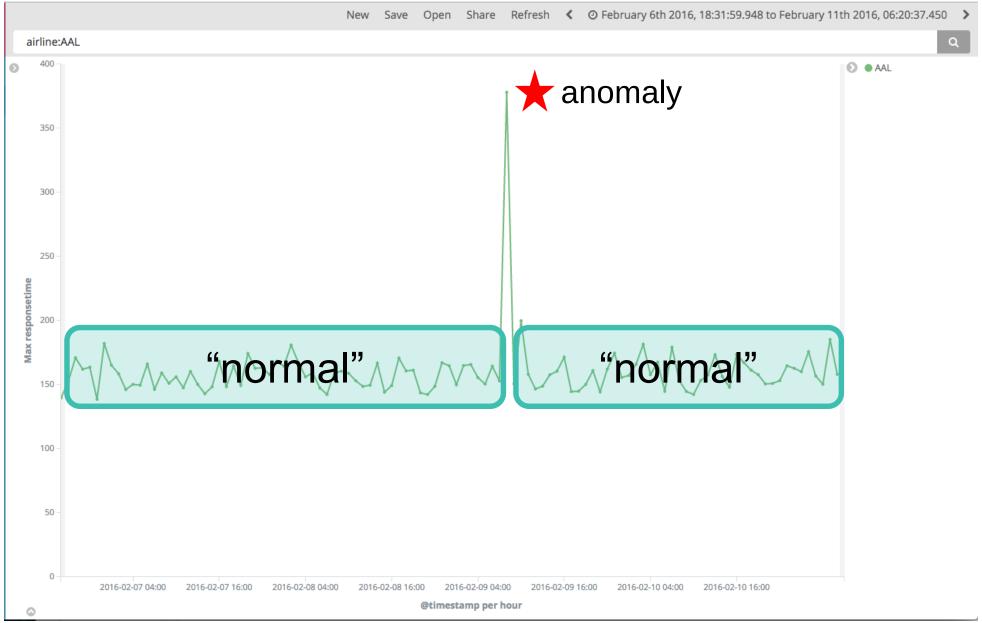
2) Something behaves in a consistent way compared against similar entities




elastic

# What is “Abnormal”?

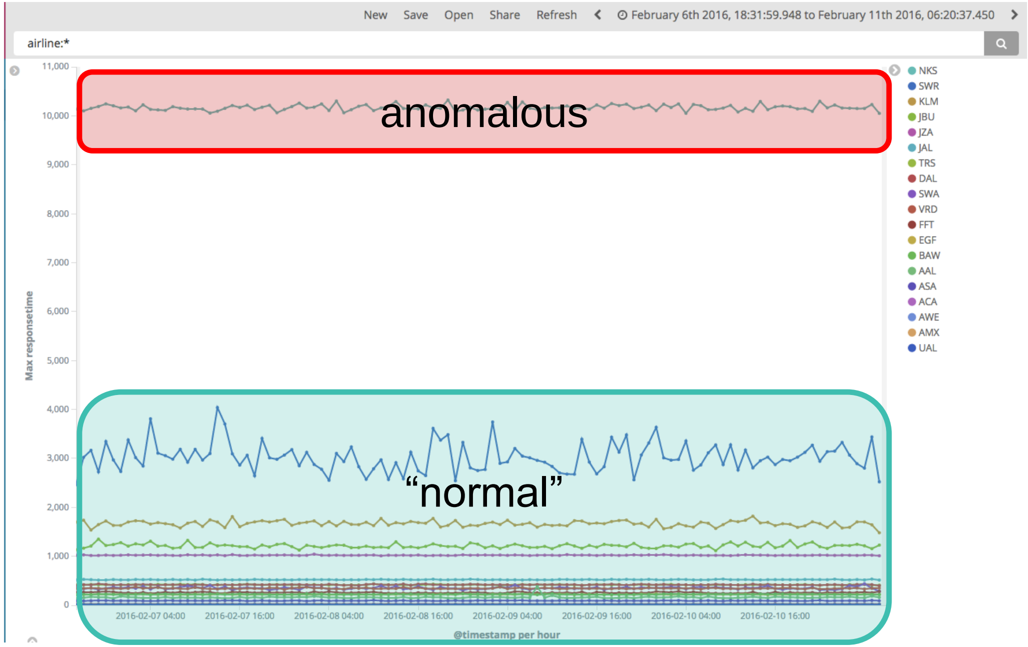
1) If something changes its behavior, compared to its own history – that change is ***anomalous***.




elastic

# What is “Abnormal”?

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

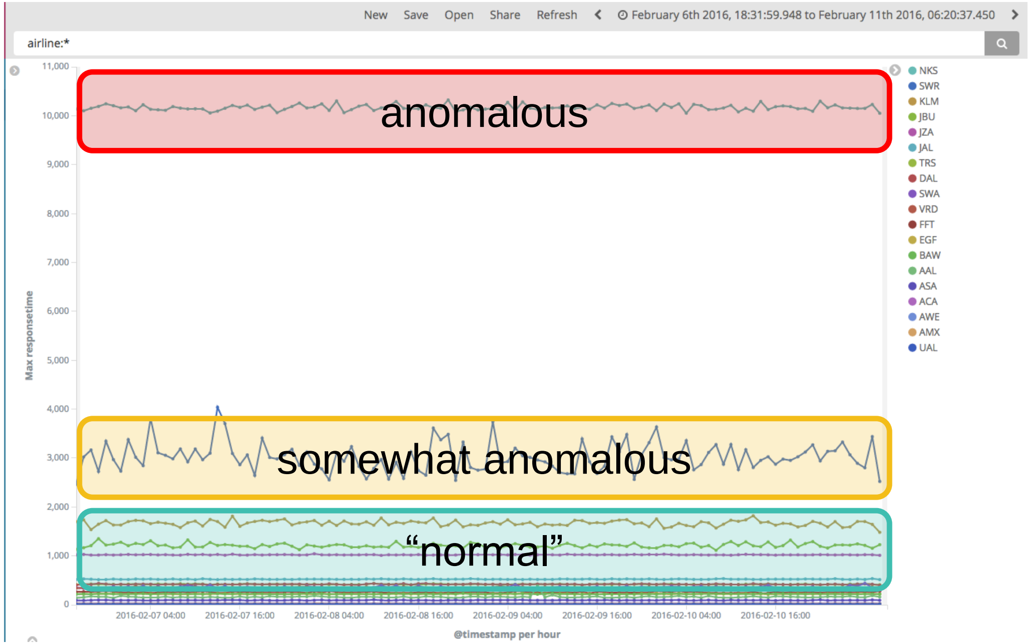



elastic

# What is “Abnormal”?

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

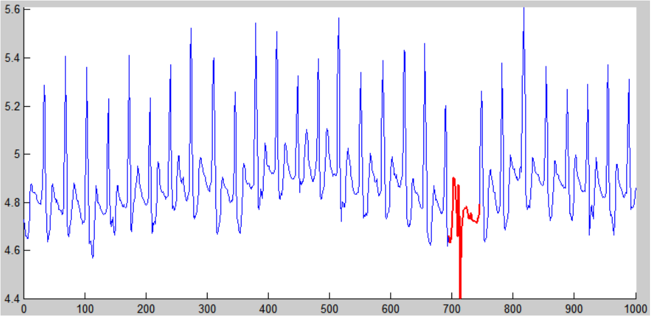
There’s also the concept of being “somewhat anomalous”




elastic

# In Summary, Anomalousness is:

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



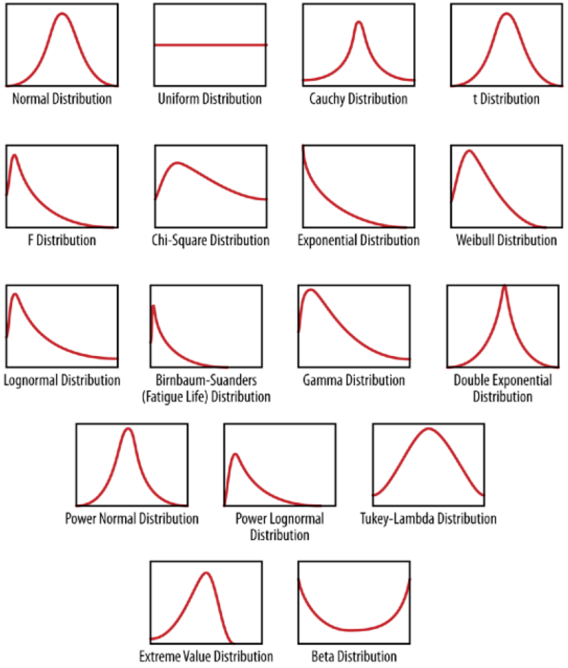

2) When an entity is drastically ***different than others*** within a population

elastic

# How to Learn “Normal”

# How does one pick a model?

Which one best fits your data?



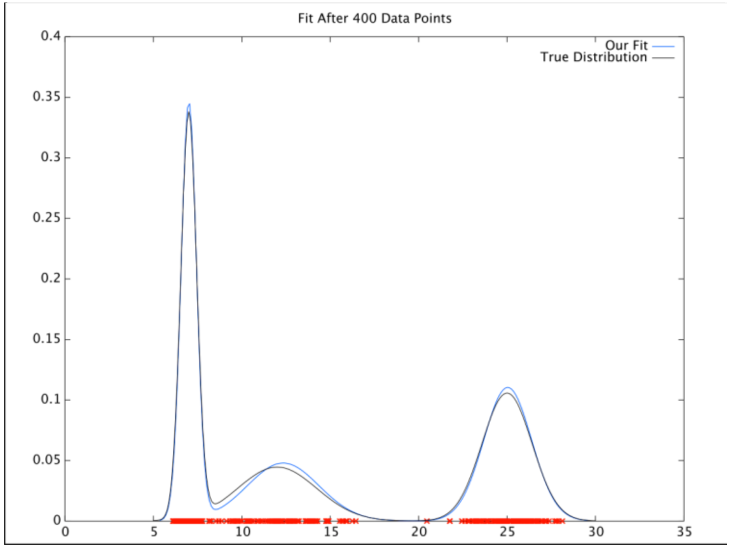

source: “Doing Data Science”
O’Neil & Schutt

# Machine Learning picks it for you

- ML uses sophisticated machine-learning techniques to best-fit the right statistical model for your data.

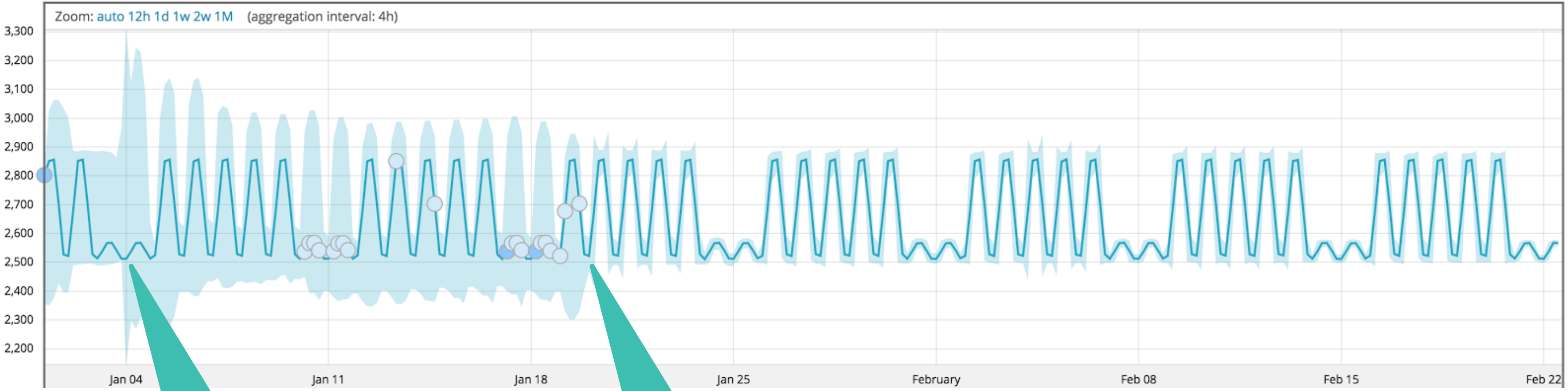
- Better models = better outlier detection = less false alarms

- Anomalies occur when observation is in low probability area




elastic

# The Model's Evolution in Time



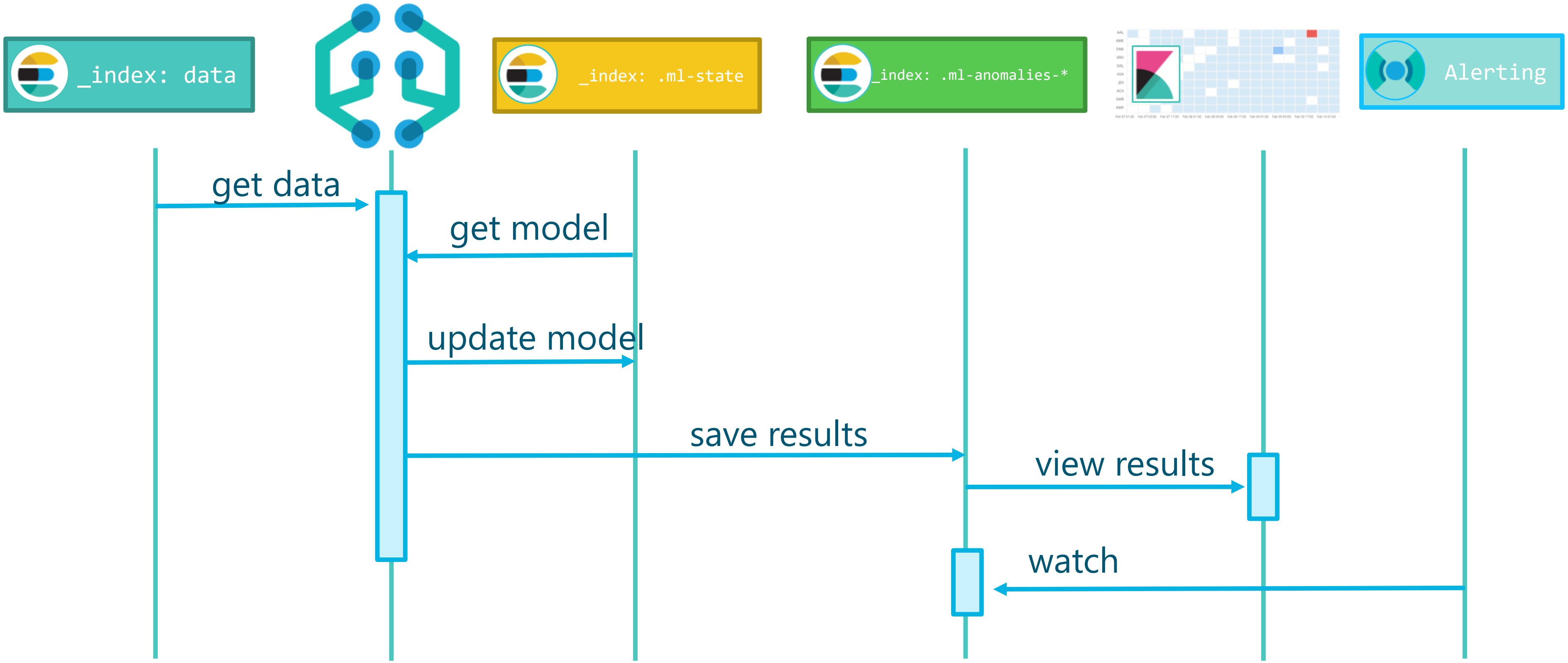

After 2 full days, daily periodicity has been learned.

After 2 full weeks, weekly periodicity has been learned.

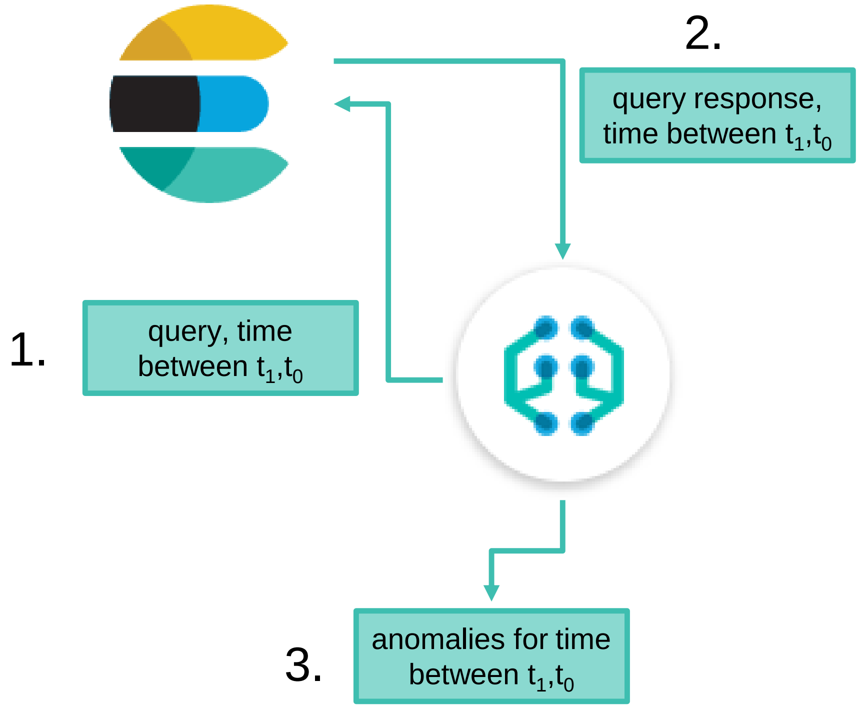
elastic

# Process Deep dive

# Sequence Diagram



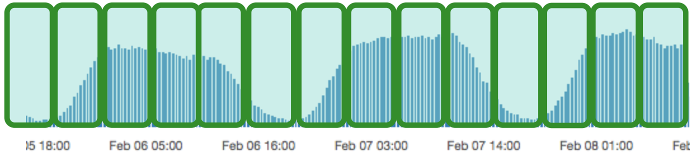
# Analysis, by “bucket”




- The time span between $t_n$ and $t_{n-1}$ is called the “bucket_span”
- ML queries Elasticsearch every bucket_span for the last bucket_span’s worth of data*
- Anomalies, if found, are produced in increments of bucket_span, but given a timestamp of $t_{n-1}$

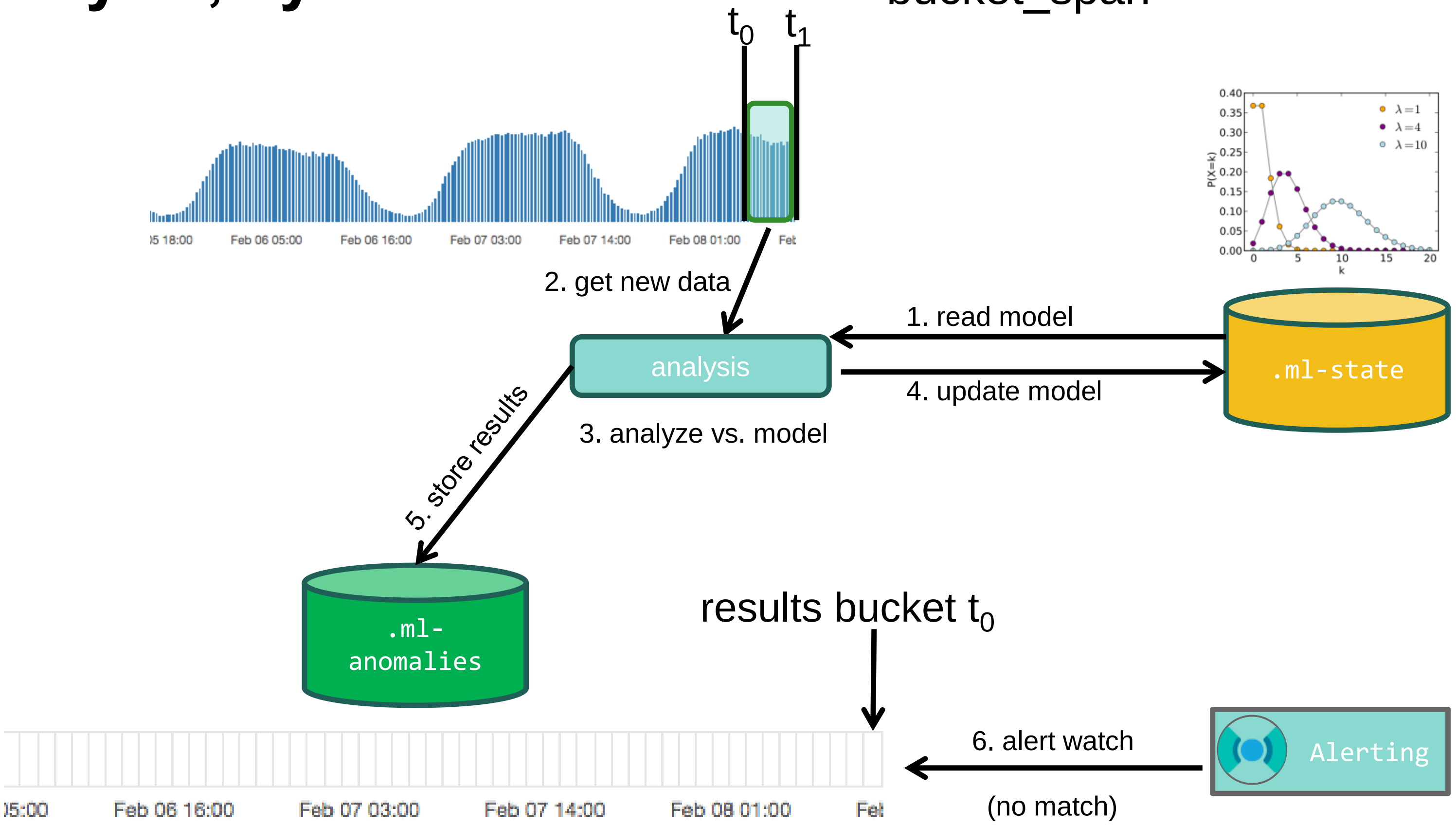
*unless a large chunk of historical data is being analyzed

# Analysis, by “bucket”



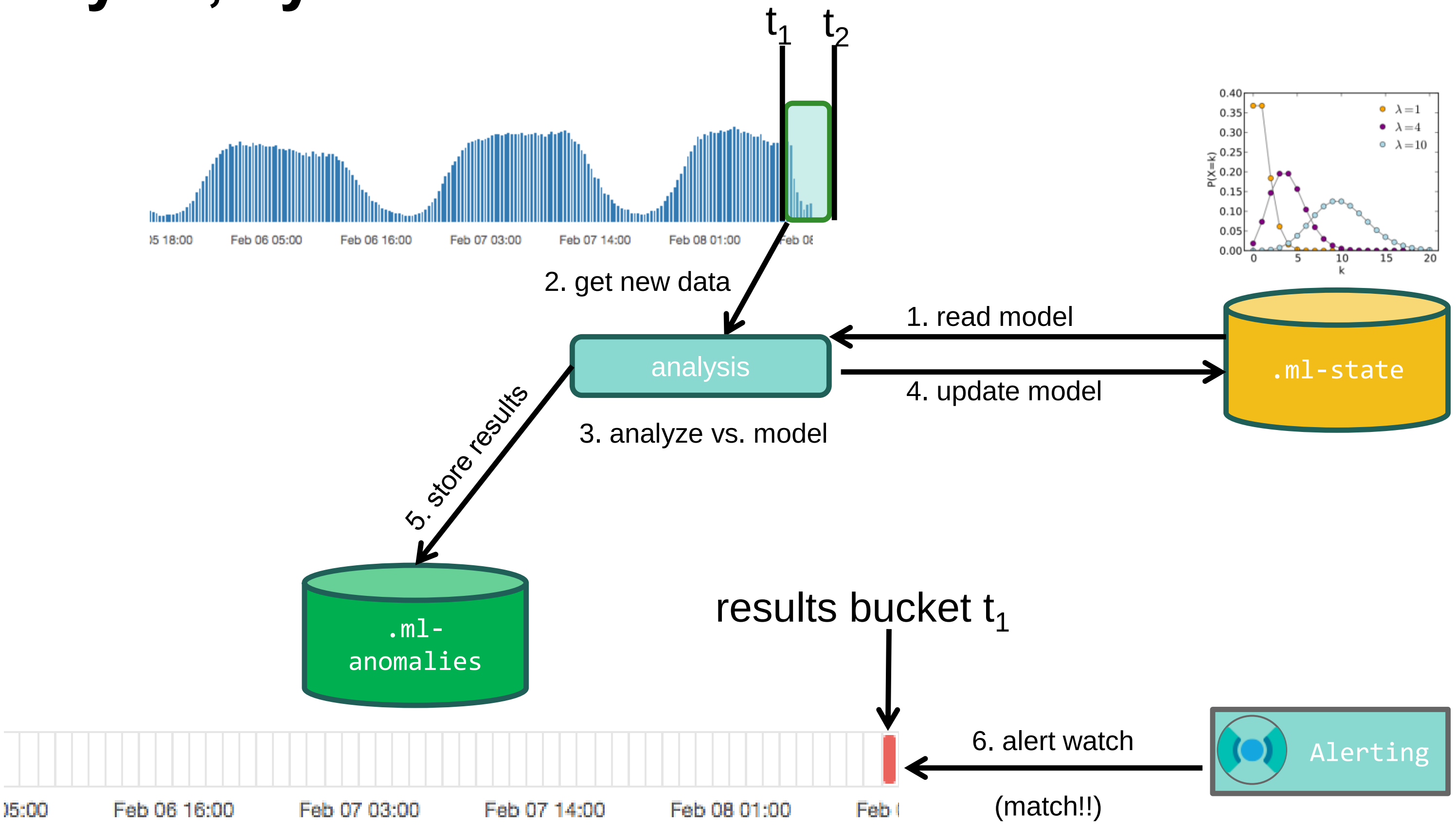
# Analysis, by “bucket”

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




elastic

# Analysis, by “bucket”



# 3가지 타입의 이상 징후를 탐지 (Important)

핵심 2

- Time series - 과거와 다른 행동 패턴 (by)
- Profiling - Outliers in population (using entity profiling) : 비슷한 다른 것들에 비교해서 다른 행동 패턴 (over)
- Rare / unusual rates in "categories" of events : 보기 드문 행동 패턴 (rare)

* 몇 십년 경험을 가진 시스템 아키텍트/관리자 및 보안 전문가의 노우하우(Know-How)를 시뮬레이션

# Time – Single Metric

# 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
...
```

ingest

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 |

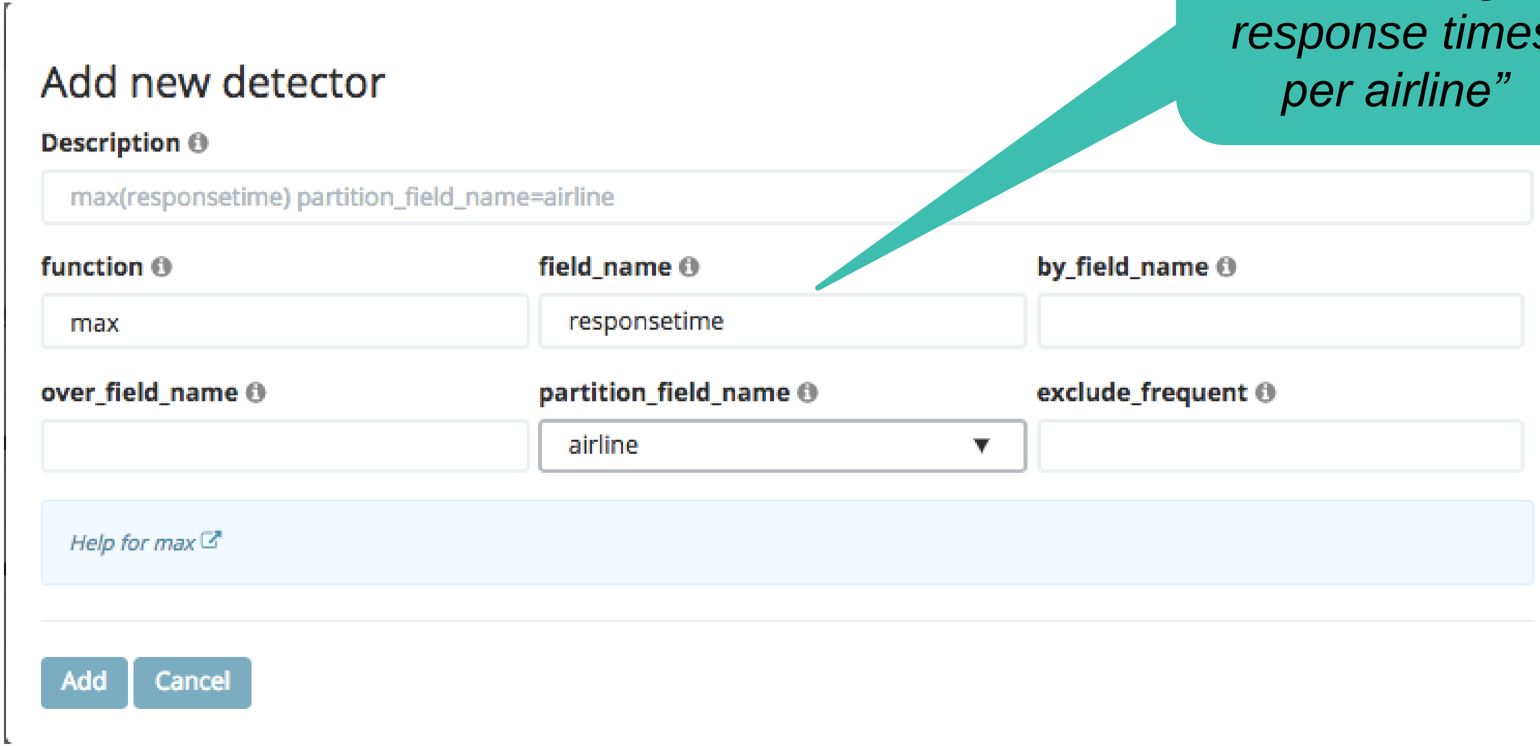
*Let's focus on this one for now*

elastic

# In Kibana: How to Answer that Question


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

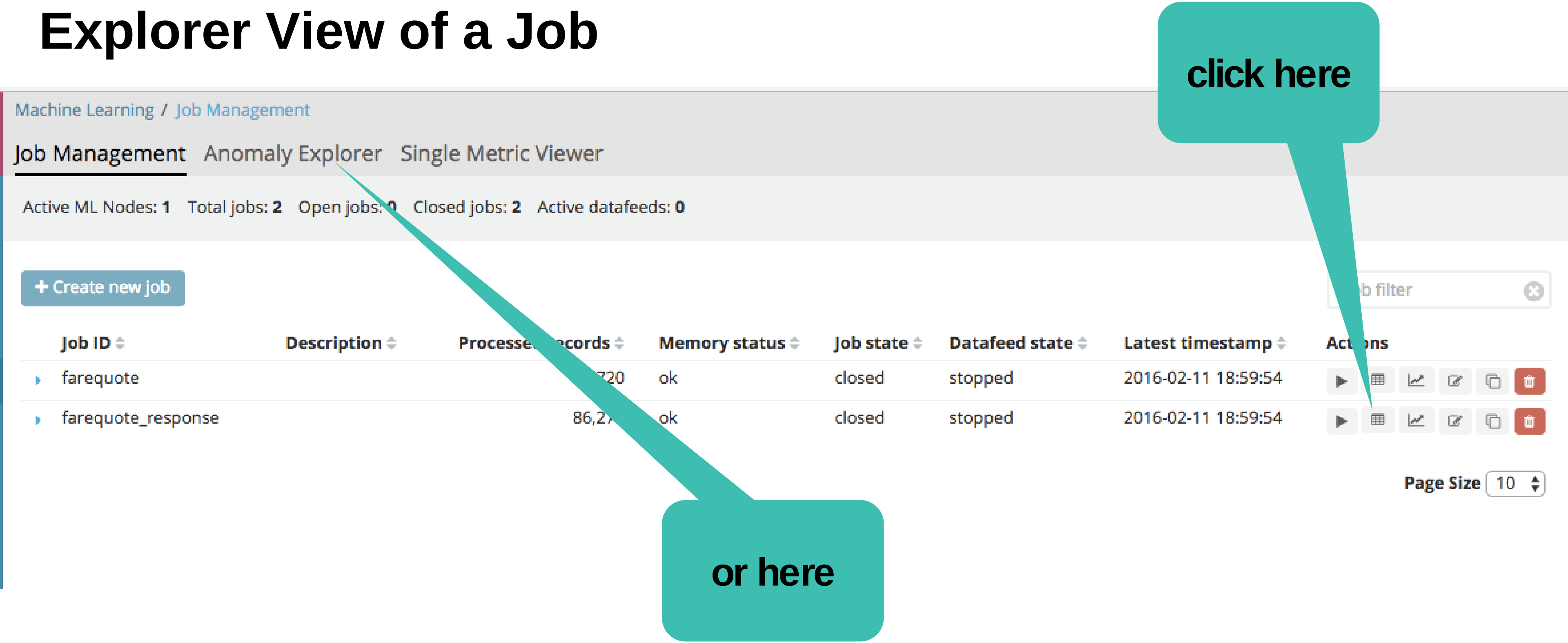
# In ML: Very similar




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

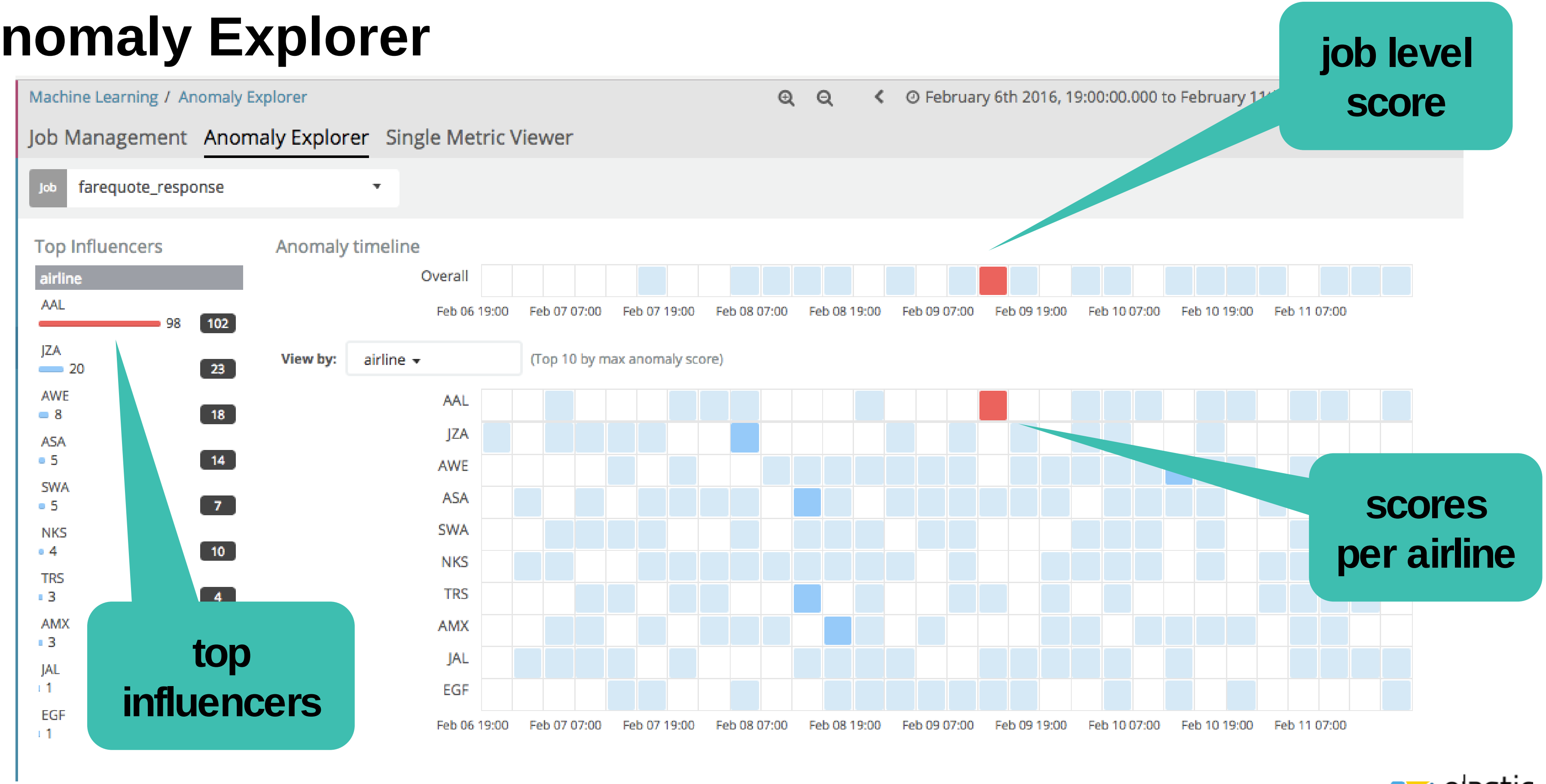
# Exploring Results with Anomaly Explorer View

# Explorer View of a Job




elastic

# Anomaly Explorer



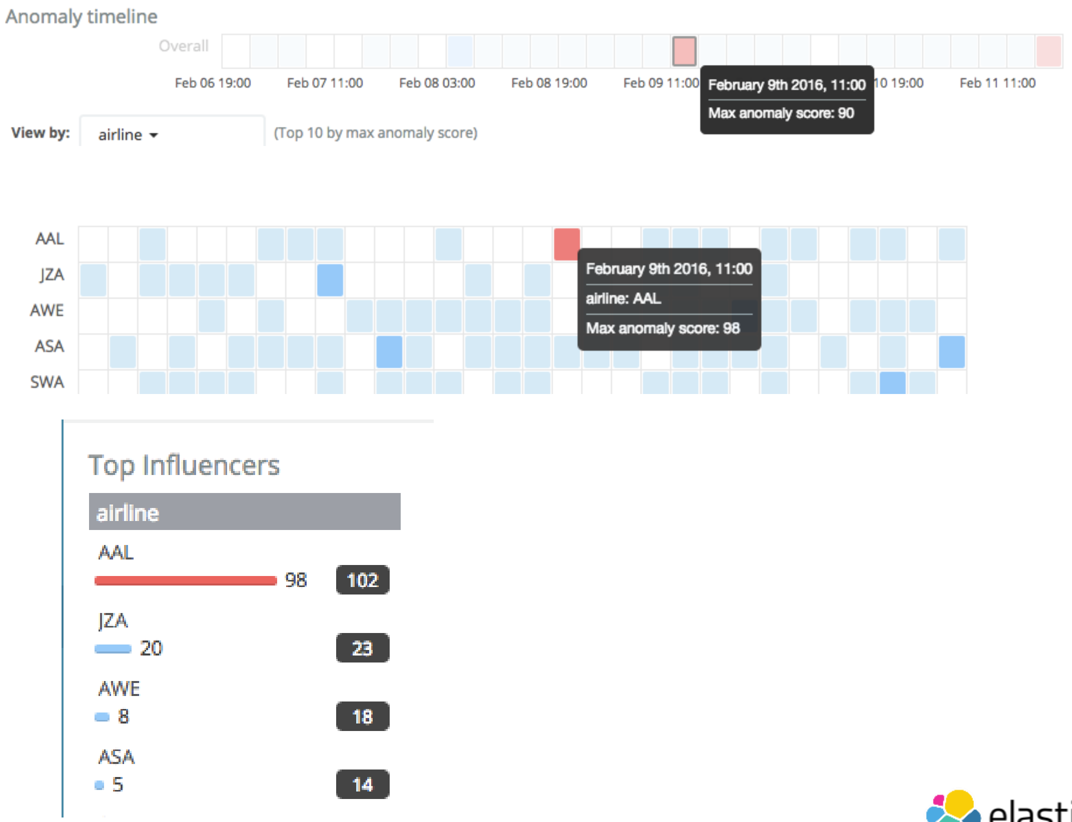

elastic

# 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

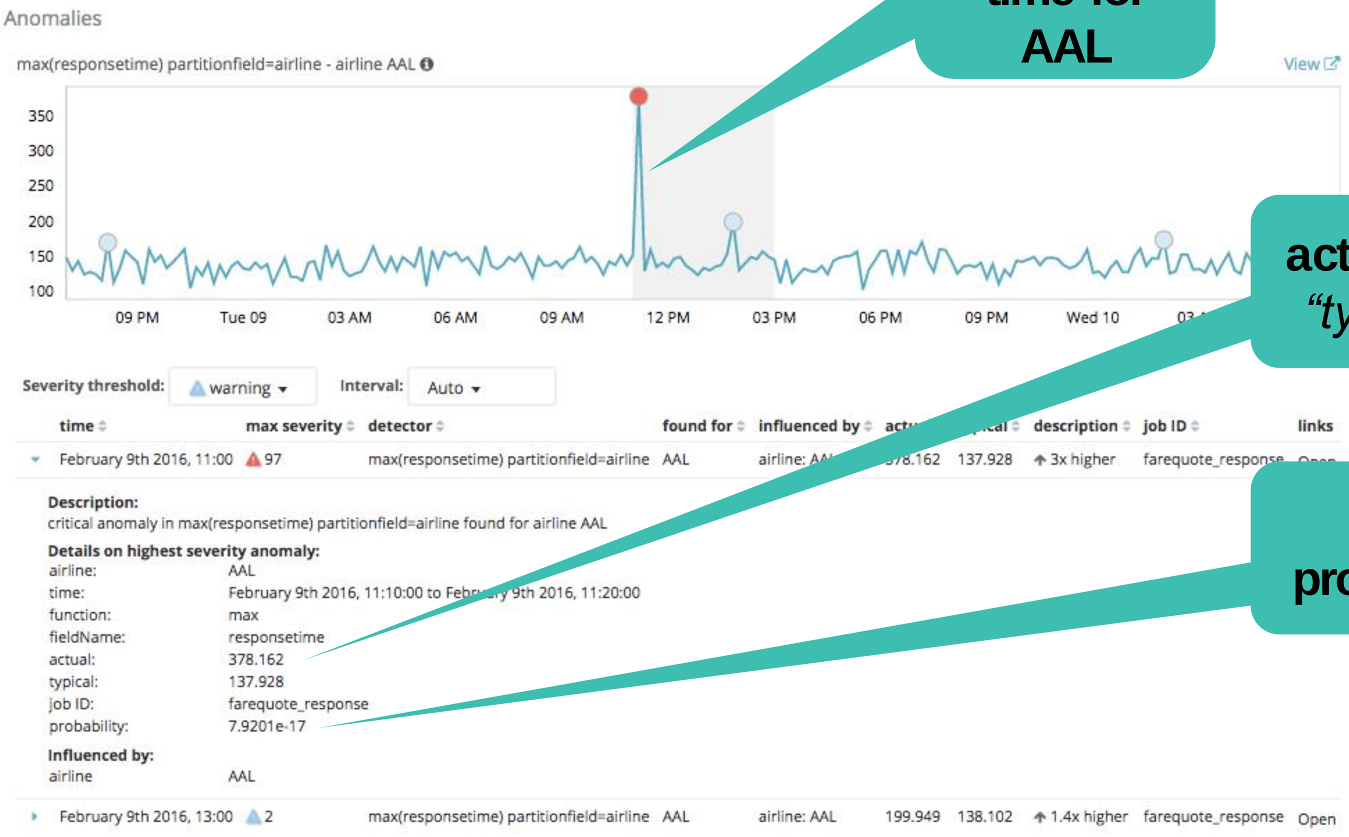
# Scoring

- 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"***

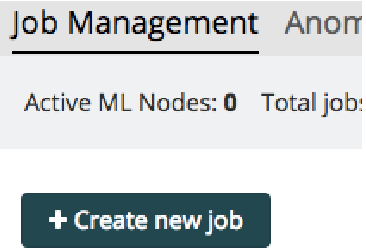
**raw probability**

elastic

# Time : Multi-Metric

# Steps to Complete

1) Create new job

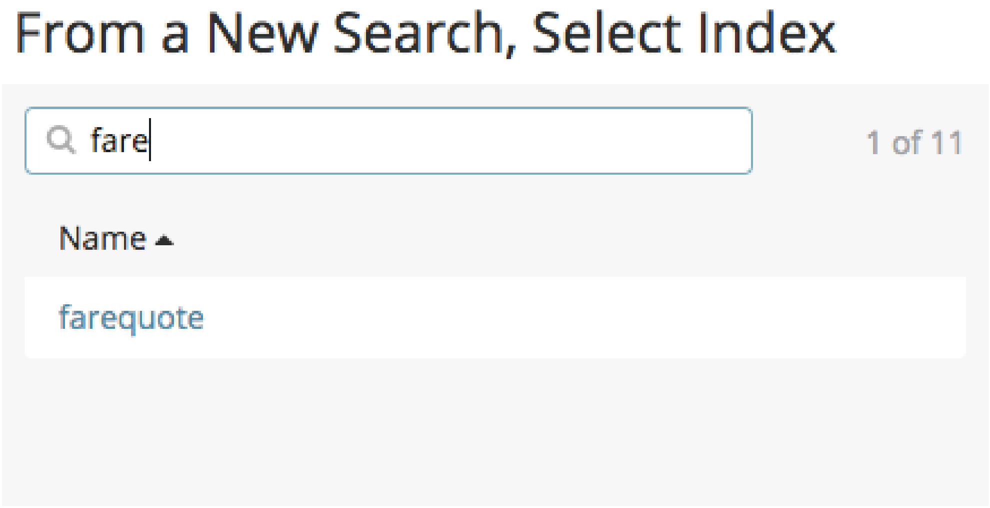



2) Choose multi metric


elastic

# Steps to Complete

3) pick farequote index



# Steps to Complete

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

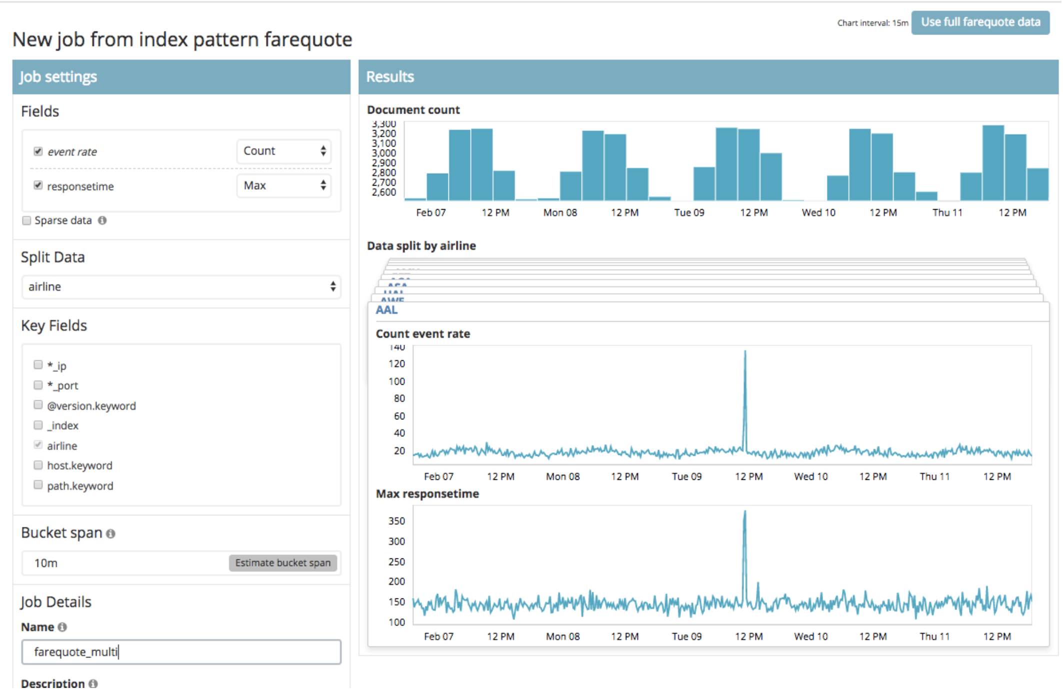
5) select 10m for bucket span

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

7) click “use full farequote data”

8) name job “farequote_multi”

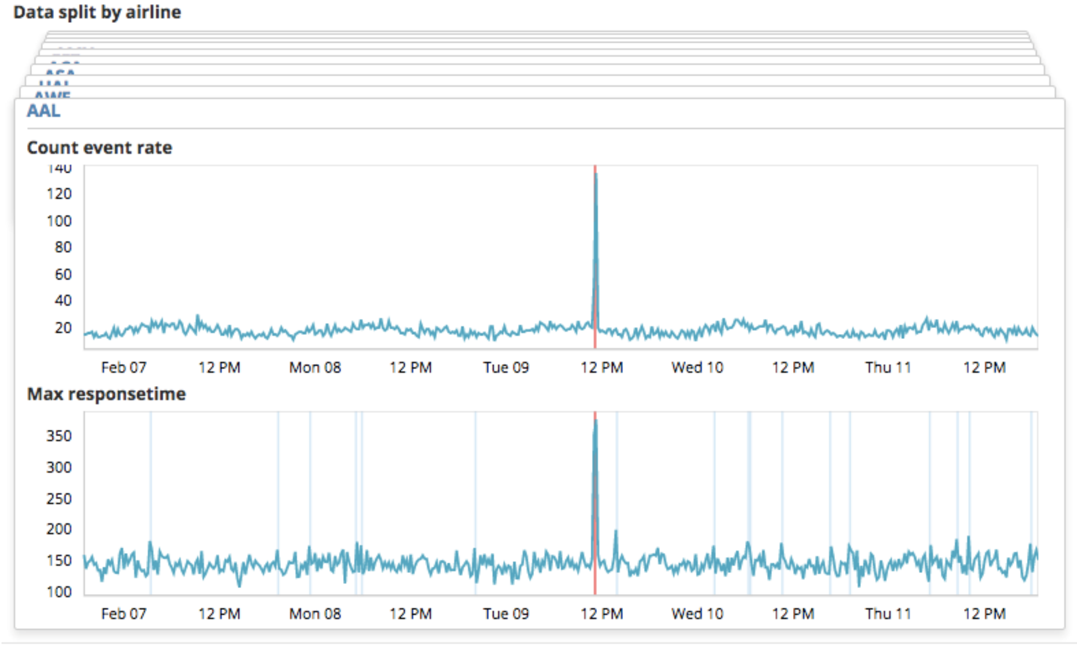
9) click “Create Job”



# Steps to Complete

10) See animated learning

11) click “View Results”

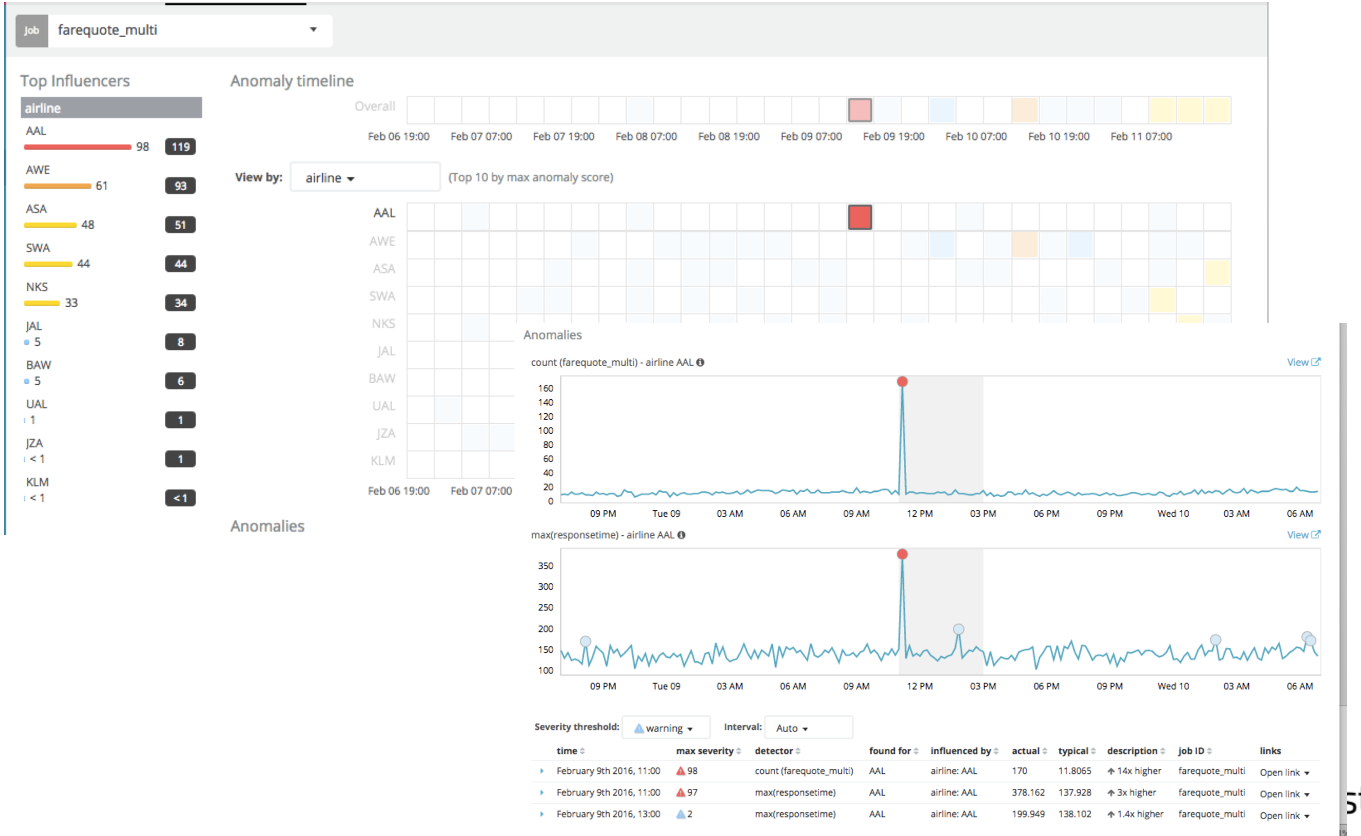



elastic

# Steps to Complete

12) Result:

anomalies for AAL
in both count
and response time



# Rare / Population Cases

# 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

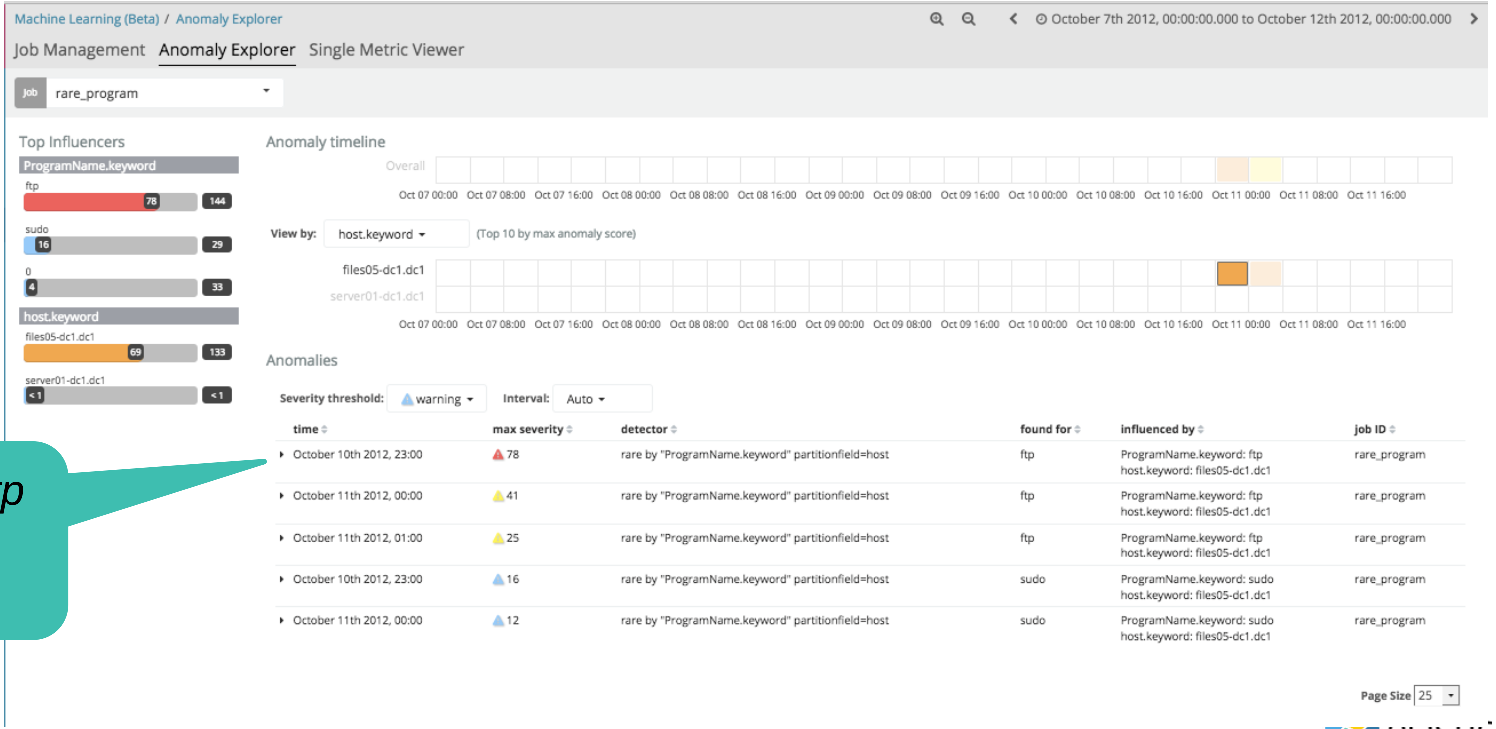
- Use Case: Security team @ services company
- Wanted to profile typical processes on each host using netstat

```
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

# 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.

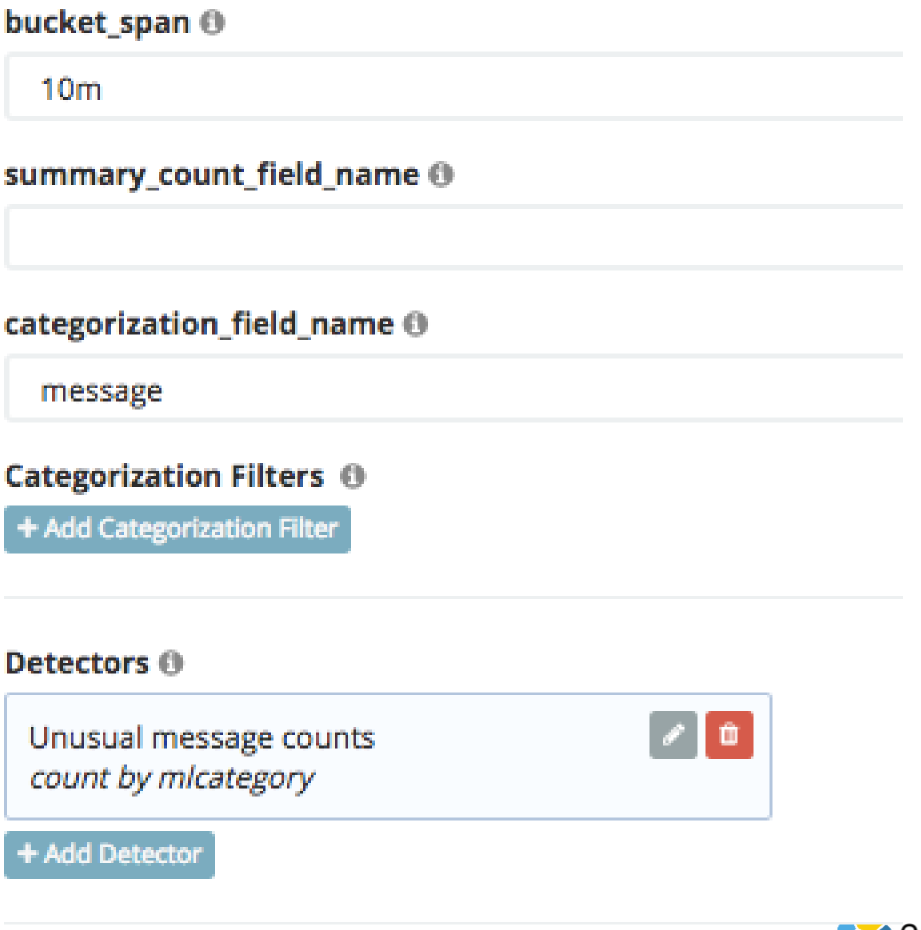
# Categorization Example

- 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"
  }
```

# 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


Anomalies

Severity threshold: warning    Interval: Auto

| time | max severity | detector | found for | actual | typical | description | job ID | links | category examples |
|---|---|---|---|---|---|---|---|---|---|
| February 8th 2016, 10:00 | 66 | count by mlcategory | mlcategory 11 | 49 | 0.0820658 | More than 100x higher | logs | Open link | Fail To Connect Database ReActivate Application / Ch onnection ! |
| February 8th 2016, 10:00 | 66 | count by mlcategory | mlcategory 10 | 49 | 0.0820658 | More than 100x higher | logs | Open link | DBMS ERROR : db=10.16.1.63!svc_prod#uid=dbadmin1;pwd=##### !<br>DBMS ERROR : db=svc_prod Err=-17 [Microsoft][ODBC SQL Server Dri |
| February 8th 2016, 10:00 | 43 | count by mlcategory | mlcategory 9 | 1 | 0.00336345 | More than 100x higher | logs | Open link | DB Not Updated [Master] Table;dbhost=dbserver.acme.com;physicall |
| February 8th 2016, 05:00 | 16 | count by mlcategory | mlcategory 6 | 1 | 0.00502013 | More than 100x higher | logs | Open link | Transaction Match In DB / Duplicate Transaction;dbhost=dbserver.ac |
| February 8th 2016, 09:00 | 8 | count by mlcategory | mlcategory 6 | 1 | 0.00657718 | More than 100x higher | logs | Open link | Transaction Match In DB / Duplicate Transaction;dbhost=dbserver.ac |
| February 8th 2016, 10:00 | 4 | count by mlcategory | mlcategory 2 | 49 | 0.081315 | More than 100x higher | logs | Open link | REC Not INSERTED [DB TRAN] Table;dbhost=dbserver.acme.com;phys |
| February 8th 2016, 10:00 | 2 | count by mlcategory | mlcategory 5 | 7 | 0.0600863 | More than 100x higher | logs | Open link | Opening Database = DRIVER={SQL Server};SERVER=10.16.1.63;networ<br>Opening Database = DRIVER={SQL Server};SERVER=127.0.0.1;network<br>Opening Database = DRIVER={SQL Server};SERVER=sssvcdbj1.acme.c |
| February 8th 2016, 10:00 | 2 | count by mlcategory | mlcategory 3 | 1 | 0.0128763 | 78x higher | logs | Open link | Using: 10.16.1.63!svc_prod#uid=dbadmin1;pwd=#####;dbhost=dbse<br>Using: sssvcdbj1.acme.com!svc_prod#uid=dbadmin1;pwd=#####;db |
| February 8th 2016, 06:00 | 2 | count by mlcategory | mlcategory 7 | 1 | 0.013673 | 73x higher | logs | Open link | Actual Transaction Not Found In DB To VOID;dbhost=dbserver.acme. |
| February 8th 2016, 10:00 | 1 | count by mlcategory | mlcategory 4 | 2 | 0.0202842 | 99x higher | logs | Open link | 012 Head Office Link Active 127.0.0.1;dbhost=dbserver.acme.com;ph |

# Who is the Outlier ?

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

# 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

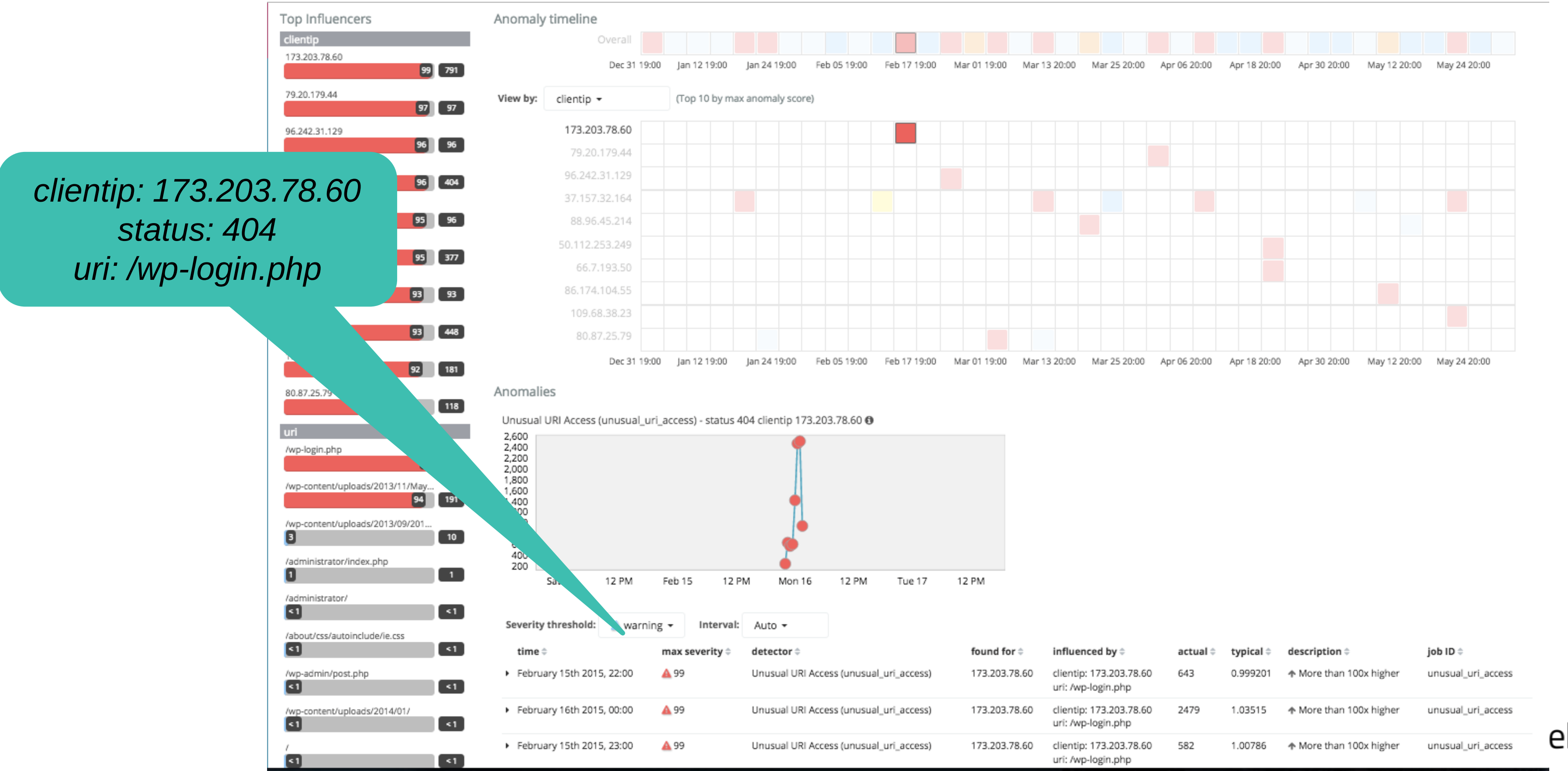
- If you want Population Analysis, you must select an “over_field_name”
  - The field chosen defines the population
- If “over_field_name” is not chosen, then population analysis is NOT invoked and thus only temporal analysis is invoked

# 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**




elastic

# Real Usecase : Multi-Job !!!

# Steps to Complete

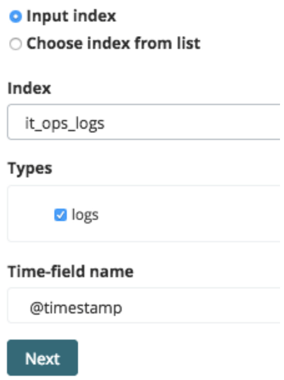
- Using the “it_ops_logs” data set:
  - Create a “count by mlcategory” job for the log events
    - use “message” as the categorization_field_name

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

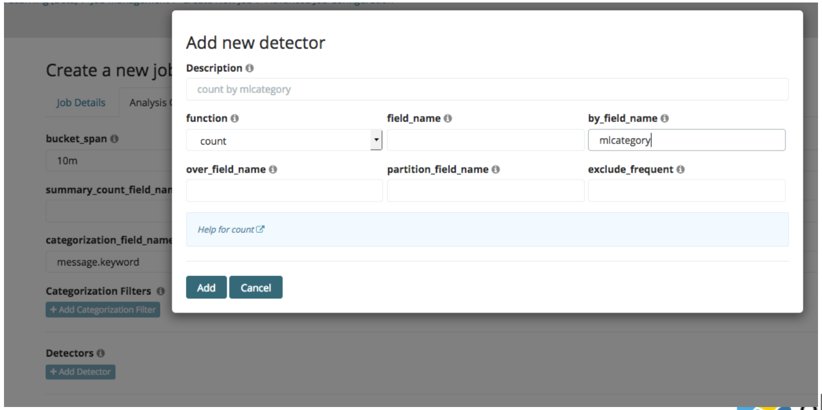
- View both jobs overlaid in the Explorer View

# 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”

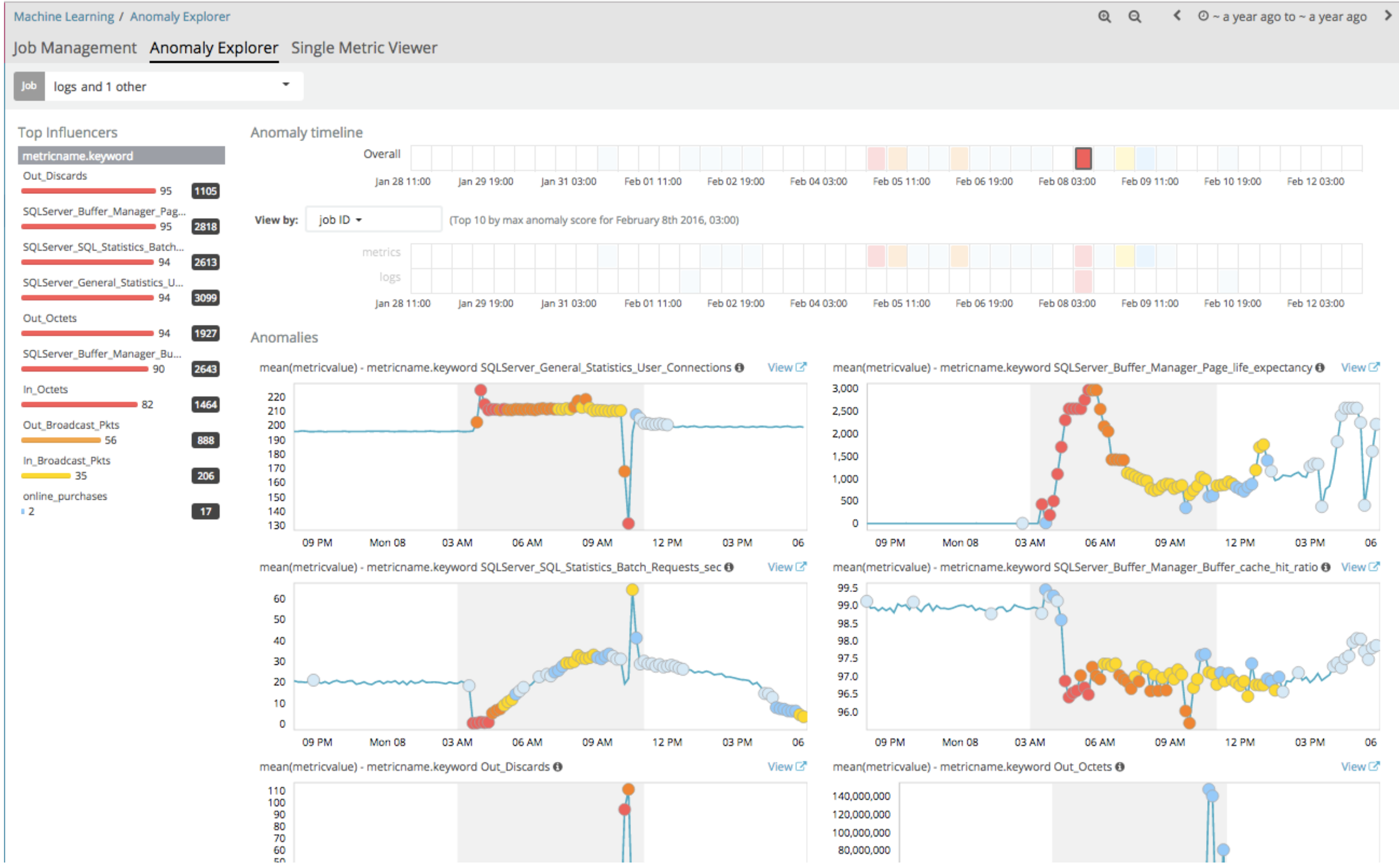


# Steps to Complete

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

# Steps to Complete

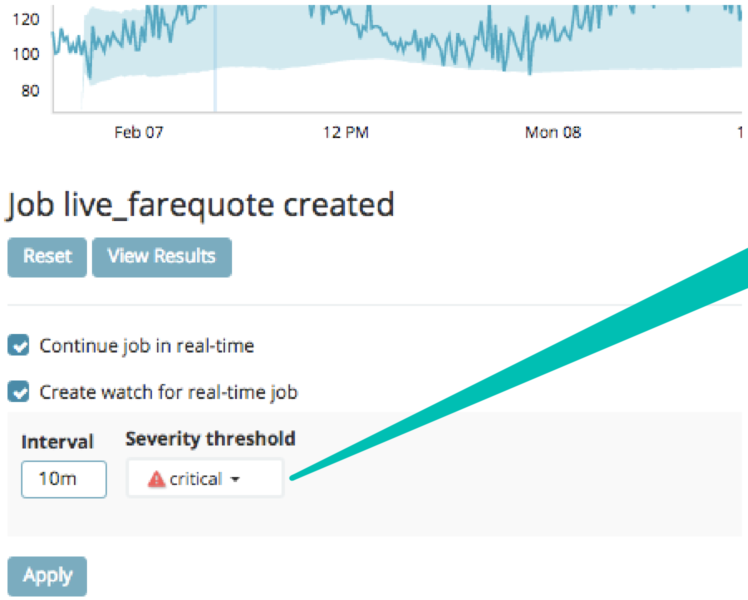
- Your goal is to get this View:



# Alerting

# Alerting on Single/Multi Metric Jobs

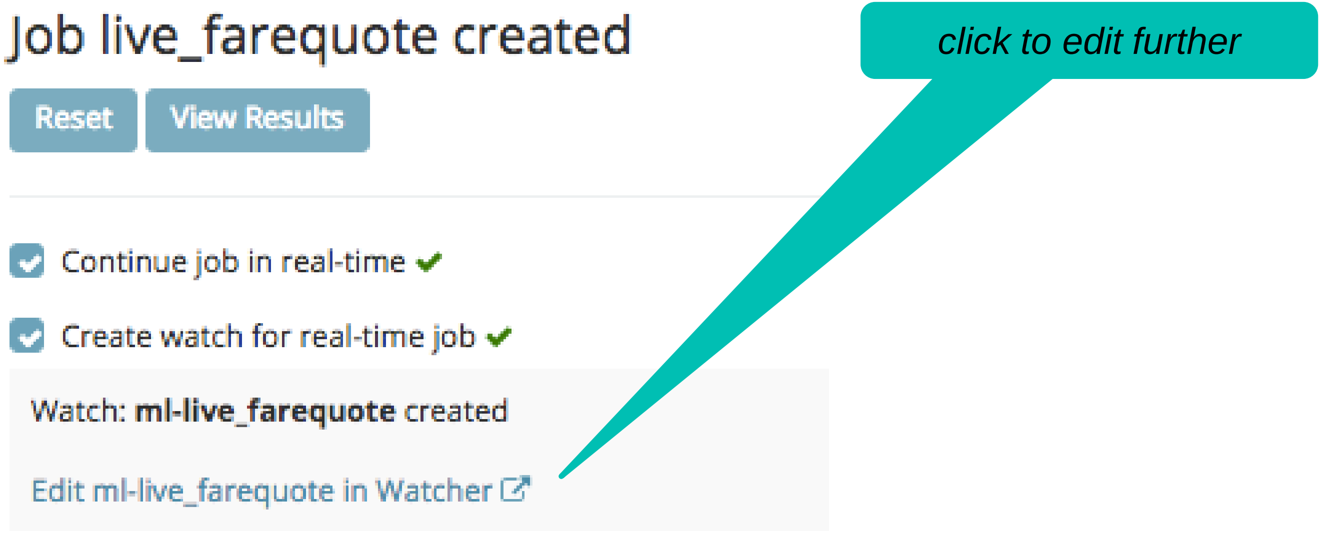
- After job configuration, see option for creating a “watch” on the live data:




*The minimum severity to alert upon*

# Alerting on Single/Multi Metric Jobs

- Clicking “Apply”:






elastic

# API Control

elastic

# 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

# 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"
```

# DEMO 2 :
# Operational Intelligence

elastic