# Introduction to Machine Learning Engineering

## Model governance for risk management

Musa Baloyi

August 14, 2018

# Table of contents

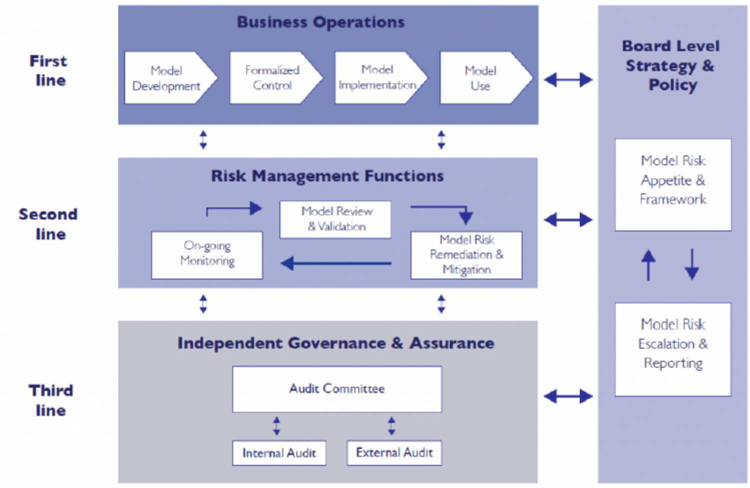
- Model governance for risk management
  - Model risk
  - Model management
  - Model governance
  - Application performance management
- Machine learning engineering
  - The Big Data Platform
  - The Elastic Stack
  - Data generation
  - Data movement
  - Architectural guidelines
- Getting started guide

# Model governance for risk management

- Logging
- Monitoring
- Metrics
- APM
- Model Governance
- Model Risk
- Risk Management
- Fit for Purpose

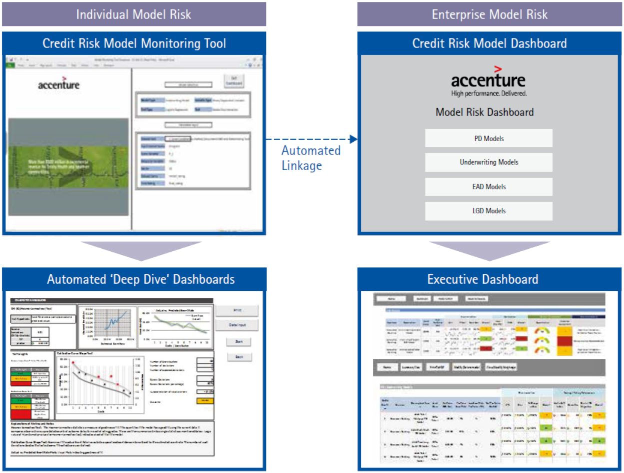
# Model governance

8 Key elements for a solid Model Governance framework


First line
Second line
Third line
Business Operations
Model Development
Formalized Control
Model Implementation
Model Use
Risk Management Functions
Model Review & Validation
On-going Monitoring
Model Risk Remediation & Mitigation
Independent Governance & Assurance
Audit Committee
Internal Audit
External Audit
Board Level Strategy & Policy
Model Risk Appetite & Framework
Model Risk Escalation & Reporting

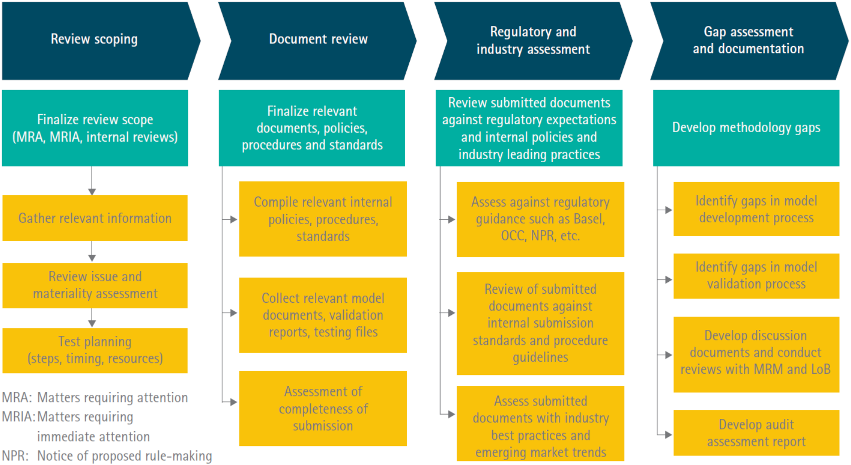
# Model governance

Figure 5: Accenture Credit Risk Model Monitoring Suite



# Model governance

Figure 4. Audit Review Framework



**Review scoping**

- Finalize review scope (MRA, MRIA, internal reviews)
- Gather relevant information
- Review issue and materiality assessment
- Test planning (steps, timing, resources)

**Document review**

- Finalize relevant documents, policies, procedures and standards
- Compile relevant internal policies, procedures, standards
- Collect relevant model documents, validation reports, testing files
- Assessment of completeness of submission

**Regulatory and industry assessment**

- Review submitted documents against regulatory expectations and internal policies and industry leading practices
- Assess against regulatory guidance such as Basel, OCC, NPR, etc.
- Review of submitted documents against internal submission standards and procedure guidelines
- Assess submitted documents with industry best practices and emerging market trends

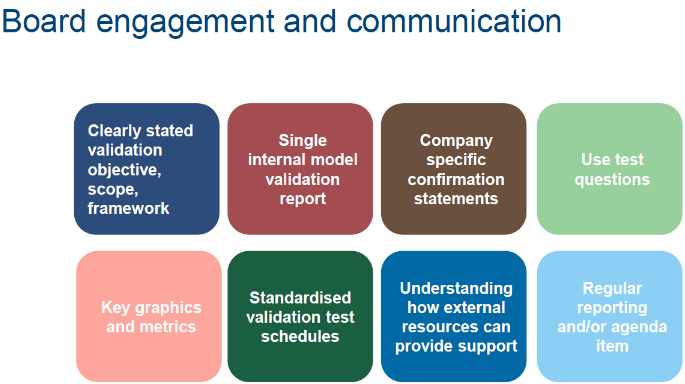
**Gap assessment and documentation**

- Develop methodology gaps
- Identify gaps in model development process
- Identify gaps in model validation process
- Develop discussion documents and conduct reviews with MRM and LoB
- Develop audit assessment report

| | |
|---|---|
| MRA: | Matters requiring attention |
| MRIA: | Matters requiring immediate attention |
| NPR: | Notice of proposed rule-making |

Source: Accenture, August 2015

# Model governance



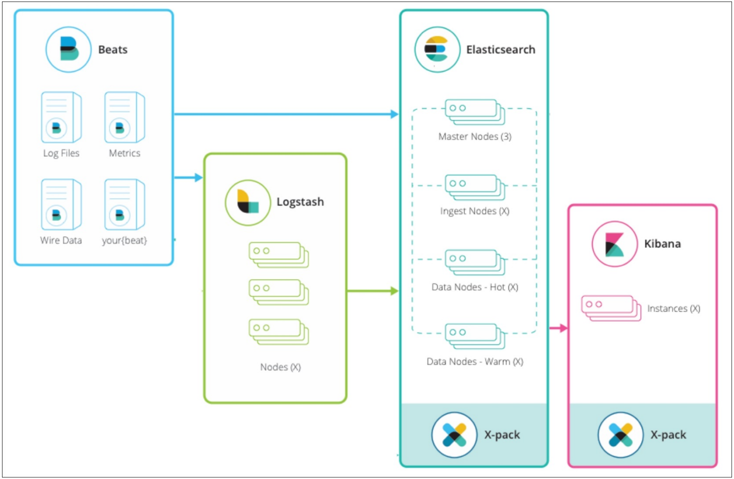
Board engagement and communication

- Clearly stated validation objective, scope, framework
- Single internal model validation report
- Company specific confirmation statements
- Use test questions
- Key graphics and metrics
- Standardised validation test schedules
- Understanding how external resources can provide support
- Regular reporting and/or agenda item

# The Elastic Stack

- The Elastic Stack
- Elasticsearch
- Kibana
- ES for Hadoop
- Architecture
- Kafka vs Beats
- Logstash
- Logging
- Monitoring
- Visualisation
- X-Pack and Machine Learning

# The Elastic Stack



# Installation guidelines

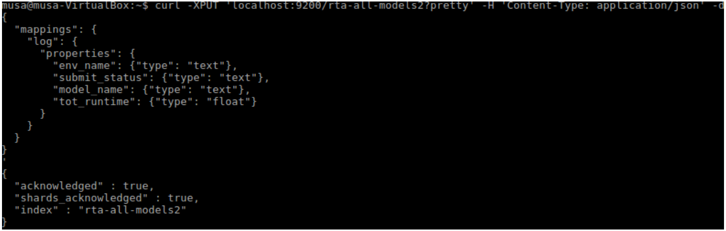
- Installing from source
- Using a package manager
- sebp/elk docker container
- elastic/stack-docker docker container
- Elastic Team SIT
- Elastic Team SLAM

# Elastic Stack demo

- sudo docker pull sebp/elk
- sudo sysctl -w vm.max_map_count=300000
- sudo docker run -p 5601:5601 -p 9200:9200 -p 5044:5044 -it –name elk sebp/elk
- Elasticsearch is running on http://localhost:9200
- Kibana is running on http://localhost:5601
- Logstash started at 5044

# Elasticsearch demo

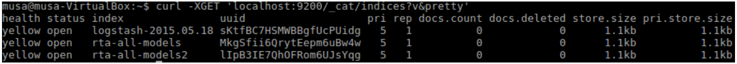
- curl -XGET 'localhost:9200/_cat/health?v&pretty'
- curl -XGET 'localhost:9200/_cat/nodes?v&pretty'
- curl -XGET 'localhost:9200/_cat/indices?v&pretty'
- Create index



```
musa@musa-VirtualBox:~$ curl -XPUT 'localhost:9200/rta-all-models2?pretty' -H 'Content-Type: application/json' -d
{
  "mappings": {
    "log": {
      "properties": {
        "env_name": {"type": "text"},
        "submit_status": {"type": "text"},
        "model_name": {"type": "text"},
        "tot_runtime": {"type": "float"}
      }
    }
  }
}
'
{
  "acknowledged" : true,
  "shards_acknowledged" : true,
  "index" : "rta-all-models2"
}
```

# Elasticsearch demo

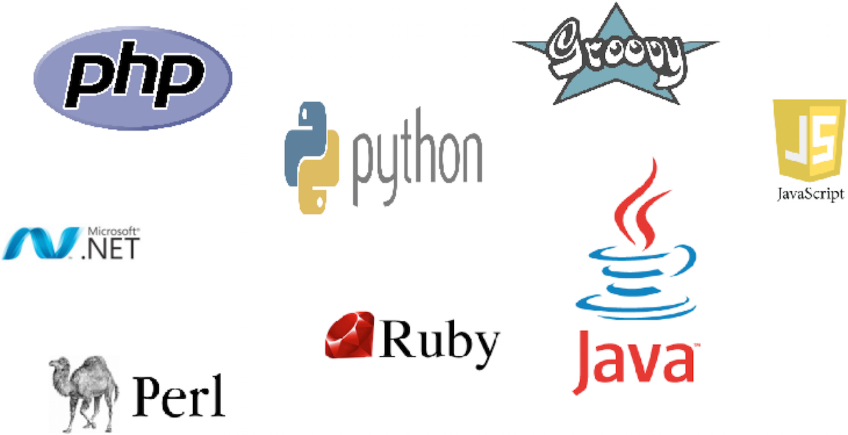
- curl -XGET 'localhost:9200/_cat/indices?v&pretty'



```
musa@musa-VirtualBox:~$ curl -XGET 'localhost:9200/_cat/indices?v&pretty'
health status index               uuid                   pri rep docs.count docs.deleted store.size pri.store.size
yellow open   logstash-2015.05.18 sKtfBC7HSMWBBgfUcPUidg   5   1          0            0      1.1kb          1.1kb
yellow open   rta-all-models      MkgSfii6QrytEepm6uBw4w   5   1          0            0      1.1kb          1.1kb
yellow open   rta-all-models2     lIpB3IE7QhOFRom6UJsYqg   5   1          0            0      1.1kb          1.1kb
```

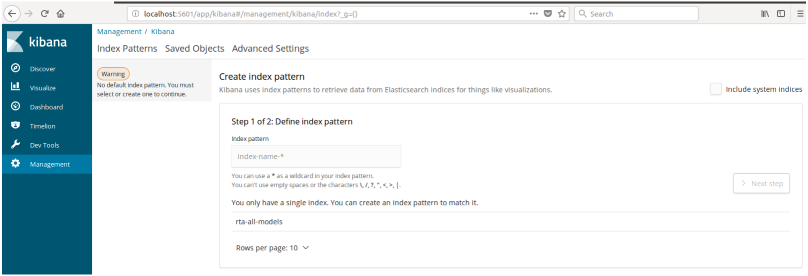
- curl -H 'Content-Type: application/x-ndjson' -XPOST 'localhost:9200/_bulk?pretty' –data-binary @rta_2018-03-13T08 37 47.143254.json

# Elasticsearch clients



# Kibana demo

- Access Kibana to see created indices and data



Warning

No default index pattern. You must select or create one to continue.

Create index pattern

Kibana uses index patterns to retrieve data from Elasticsearch indices for things like visualizations.

Step 1 of 2: Define index pattern

Index pattern: index-name-*

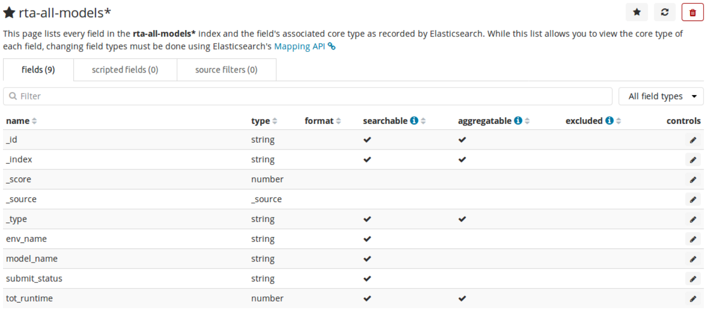
You can use a * as a wildcard in your index pattern.
You can't use empty spaces or the characters \, /, ?, ", <, >, |.

You only have a single index. You can create an index pattern to match it.

rta-all-models

# Kibana demo

- Create index pattern in Kibana



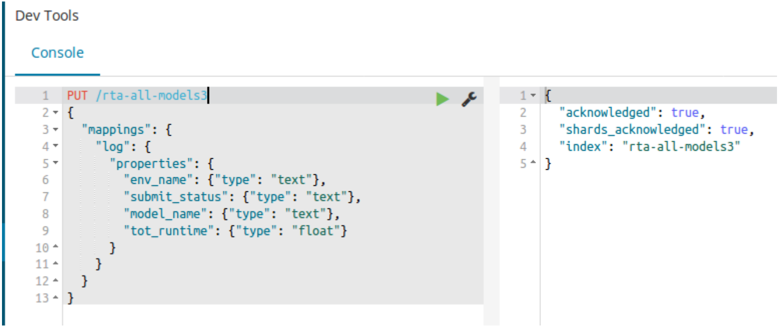
★ rta-all-models*

This page lists every field in the **rta-all-models*** index and the field's associated core type as recorded by Elasticsearch. While this list allows you to view the core type of each field, changing field types must be done using Elasticsearch's Mapping API

| name | type | format | searchable | aggregatable | excluded | controls |
|---|---|---|---|---|---|---|
| _id | string | | ✓ | ✓ | | ✎ |
| _index | string | | ✓ | ✓ | | ✎ |
| _score | number | | | | | ✎ |
| _source | _source | | | | | ✎ |
| _type | string | | ✓ | ✓ | | ✎ |
| env_name | string | | ✓ | | | ✎ |
| model_name | string | | ✓ | | | ✎ |
| submit_status | string | | ✓ | | | ✎ |
| tot_runtime | number | | ✓ | ✓ | | ✎ |

# Kibana Console

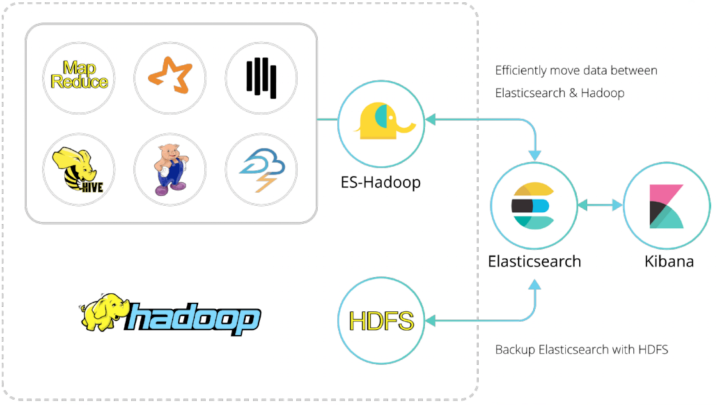
- Alternative to CURL



```
PUT /rta-all-models3
{
  "mappings": {
    "log": {
      "properties": {
        "env_name": {"type": "text"},
        "submit_status": {"type": "text"},
        "model_name": {"type": "text"},
        "tot_runtime": {"type": "float"}
      }
    }
  }
}
```

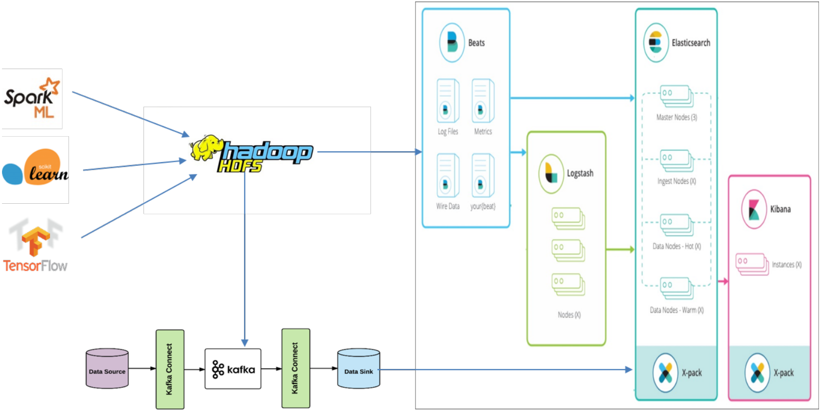
```
{
  "acknowledged": true,
  "shards_acknowledged": true,
  "index": "rta-all-models3"
}
```

# Elasticsearch for Hadoop



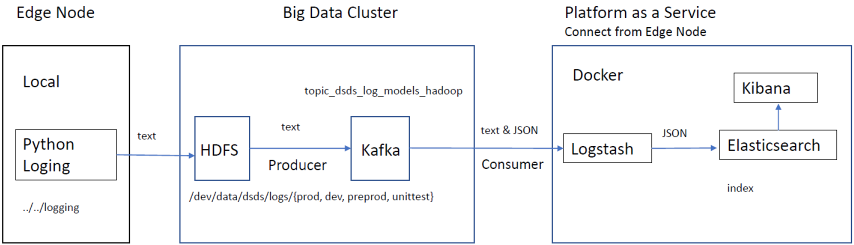
Map Reduce
HIVE
hadoop
ES-Hadoop
HDFS
Elasticsearch
Kibana
Efficiently move data between Elasticsearch & Hadoop
Backup Elasticsearch with HDFS

# Architectural overview



# Architectural overview

## Model Monitoring Architecture



Edge Node

Big Data Cluster

Platform as a Service

Connect from Edge Node

Local

Python Loging

../../logging

text

HDFS

text

Producer

Kafka

topic_dsds_log_models_hadoop

/dev/data/dsds/logs/{prod, dev, preprod, unittest}

text & JSON

Consumer

Docker

Logstash

JSON

Elasticsearch

index

Kibana

# Kafka

- Kafka is generally used for building real-time streaming
  - data pipelines that reliably get data between systems or applications
  - applications that transform or react to the streams of data
- Kafka is run as a cluster on one or more servers that can span multiple datacenters.
- The Kafka cluster stores streams of records in categories called topics.
- Each record consists of a key, a value, and a timestamp.

# Kafka installation

- Download the binary: kafka_2.12-1.0.1.tgz
- 7z x kafka_2.12-1.0.1.tgz && 7z x kafka_2.12-1.0.1.tar
- sudo mv kafka_2.12-1.0.1 /opt/Kafka

# Kafka demo

- cd /opt/Kafka/ kafka_2.12-1.0.1
- sudo bin/kafka-server-start.sh config/server.properties
- bin/kafka-console-consumer.sh –bootstrap-server localhost:9092 –topic testing –from-beginning
- bin/kafka-topics.sh –create –zookeeper localhost:2181 –replication-factor 1 –partitions 1 –topic testing
- bin/kafka-topics.sh –list –zookeeper localhost:2181
- Configure Kafka producer connect-file-source.properties
- Configure Kafka consumer connect-file-sink.properties
- bin/connect-standalone.sh
  config/connect-standalone.properties
  config/connect-file-source.properties
  config/connect-file-sink.properties

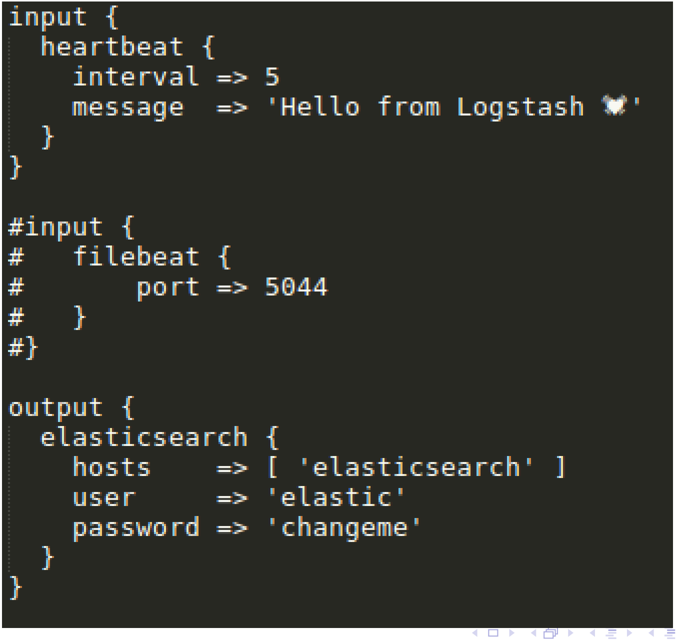
# Beats

- Seamlessly integrates with the Elastic Stack. Kafka requires a separate install.
- One way, Kafka is bidirectional
- Extensible
- Shippers: Filebeat, Metricbeat, Packetbeat, Winlogbeat, Auditbeat, Heartbeat.
- Filebeat configuration

# Beats demo: Heartbeat

- elastic/stack-docker

# Logstash.conf



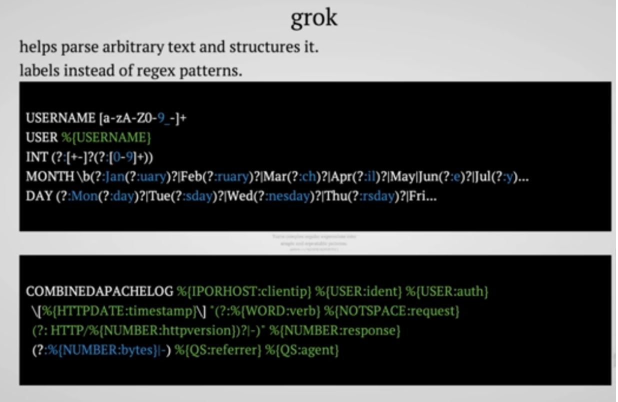
```
input {
  heartbeat {
    interval => 5
    message  => 'Hello from Logstash ♥'
  }
}

#input {
#   filebeat {
#       port => 5044
#   }
#}

output {
  elasticsearch {
    hosts    => [ 'elasticsearch' ]
    user     => 'elastic'
    password => 'changeme'
  }
}
```

# Logstash

- grok – better pattern matching
- https://www.elastic.co/guide/en/logstash/current/plugins-filters-grok.html



grok

helps parse arbitrary text and structures it.

labels instead of regex patterns.

```
USERNAME [a-zA-Z0-9_-]+
USER %{USERNAME}
INT (?:[+-]?(?:[0-9]+))
MONTH \b(?:Jan(?:uary)?|Feb(?:ruary)?|Mar(?:ch)?|Apr(?:il)?|May|Jun(?:e)?|Jul(?:y)...
DAY (?:Mon(?:day)?|Tue(?:sday)?|Wed(?:nesday)?|Thu(?:rsday)?|Fri...
```

Turns complex regular expressions into
simple and repeatable patterns
[illegible]

```
COMBINEDAPACHELOG %{IPORHOST:clientip} %{USER:ident} %{USER:auth}
 \[%{HTTPDATE:timestamp}\] "(?:%{WORD:verb} %{NOTSPACE:request}
 (?: HTTP/%{NUMBER:httpversion})?|-)" %{NUMBER:response}
 (?:%{NUMBER:bytes}|-) %{QS:referrer} %{QS:agent}
```

# Data generation

- Out of the box logging
- Custom logging
- Data dictionaries
- Data types
- File types

# Logging facility for Python

- This module defines functions and classes which implement a flexible event logging system for applications and libraries.
- The key benefit of having the logging API provided by a standard library module is that all Python modules can participate in logging, so your application log can include your own messages integrated with messages from third-party modules.

# Logging facility for Python

The basic classes defined by the module, together with their functions, are:

- Loggers expose the interface that application code directly uses.
- Handlers send the log records (created by loggers) to the appropriate destination.
- Filters provide a finer grained facility for determining which log records to output.
- Formatters specify the layout of log records in the final output.

# Logging facility for Python

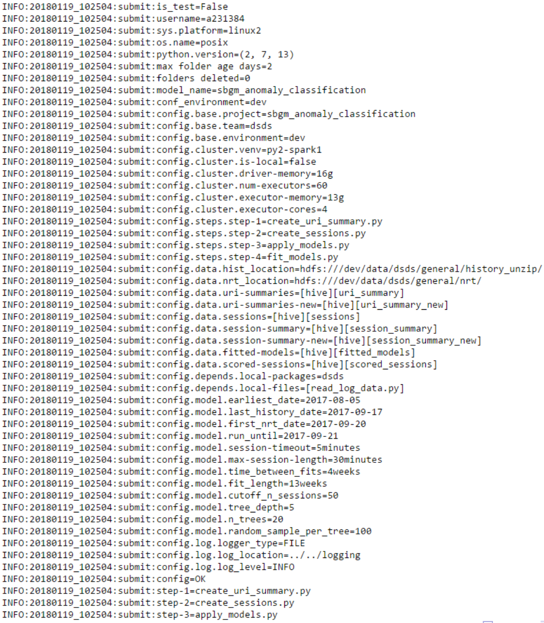
Logging serves two purposes:

- Diagnostic logging records events related to the application's operation. If a user calls in to report an error, for example, the logs can be searched for context.
- Audit logging records events for business analysis. A user's transactions can be extracted and combined with other user details for reports or to optimize a business goal.

# dsds_logging guide

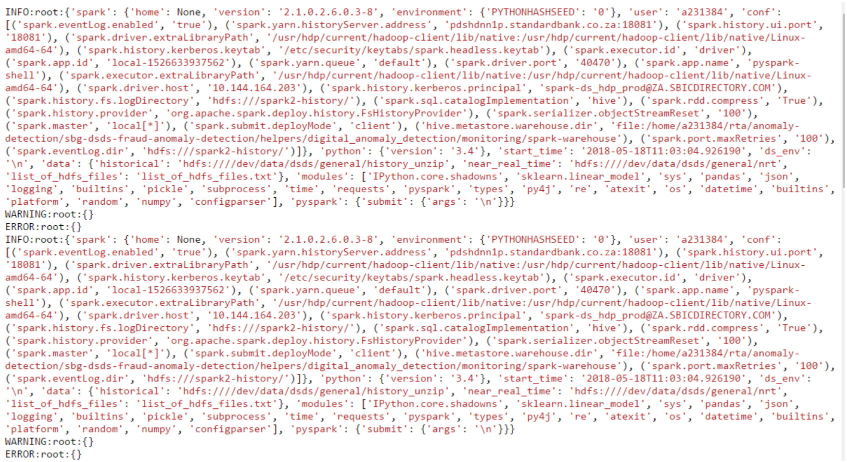
1. git clone https://<username>@tools.standardbank.co.za/bitbucket/scm/datas
   packages.git
2. sys.path.append("../python-packages/dsds")
3. import dsds.dsds_logging
4. config, logger = dsds_spark.get_config_and_logger(sys.argv)
5. sc, hiveContext = dsds_spark.get_contexts(config, sys.argv)
6. spark_main(config, logger, sc, hiveContext)
7. logger.info(’Feature_extraction.py’, ’feature_set_6’, feature_set_6.columns)

# Sample log (.txt)



```
INFO:20180119_102504:submit:is_test=False
INFO:20180119_102504:submit:username=a231384
INFO:20180119_102504:submit:sys.platform=linux2
INFO:20180119_102504:submit:os.name=posix
INFO:20180119_102504:submit:python.version=(2, 7, 13)
INFO:20180119_102504:submit:max folder age days=2
INFO:20180119_102504:submit:folders deleted=0
INFO:20180119_102504:submit:model_name=sbgm_anomaly_classification
INFO:20180119_102504:submit:conf_environment=dev
INFO:20180119_102504:submit:config.base.project=sbgm_anomaly_classification
INFO:20180119_102504:submit:config.base.team=dsds
INFO:20180119_102504:submit:config.base.environment=dev
INFO:20180119_102504:submit:config.cluster.venv=py2-spark1
INFO:20180119_102504:submit:config.cluster.is-local=false
INFO:20180119_102504:submit:config.cluster.driver-memory=16g
INFO:20180119_102504:submit:config.cluster.num-executors=60
INFO:20180119_102504:submit:config.cluster.executor-memory=13g
INFO:20180119_102504:submit:config.cluster.executor-cores=4
INFO:20180119_102504:submit:config.steps.step-1=create_uri_summary.py
INFO:20180119_102504:submit:config.steps.step-2=create_sessions.py
INFO:20180119_102504:submit:config.steps.step-3=apply_models.py
INFO:20180119_102504:submit:config.steps.step-4=fit_models.py
INFO:20180119_102504:submit:config.data.hist_location=hdfs:///dev/data/dsds/general/history_unzip/
INFO:20180119_102504:submit:config.data.nrt_location=hdfs:///dev/data/dsds/general/nrt/
INFO:20180119_102504:submit:config.data.uri-summaries=[hive][uri_summary]
INFO:20180119_102504:submit:config.data.uri-summaries-new=[hive][uri_summary_new]
INFO:20180119_102504:submit:config.data.sessions=[hive][sessions]
INFO:20180119_102504:submit:config.data.session-summary=[hive][session_summary]
INFO:20180119_102504:submit:config.data.session-summary-new=[hive][session_summary_new]
INFO:20180119_102504:submit:config.data.fitted-models=[hive][fitted_models]
INFO:20180119_102504:submit:config.data.scored-sessions=[hive][scored_sessions]
INFO:20180119_102504:submit:config.depends.local-packages=dsds
INFO:20180119_102504:submit:config.depends.local-files=[read_log_data.py]
INFO:20180119_102504:submit:config.model.earliest_date=2017-08-05
INFO:20180119_102504:submit:config.model.last_history_date=2017-09-17
INFO:20180119_102504:submit:config.model.first_nrt_date=2017-09-20
INFO:20180119_102504:submit:config.model.run_until=2017-09-21
INFO:20180119_102504:submit:config.model.session-timeout=5minutes
INFO:20180119_102504:submit:config.model.max-session-length=30minutes
INFO:20180119_102504:submit:config.model.time_between_fits=4weeks
INFO:20180119_102504:submit:config.model.fit_length=13weeks
INFO:20180119_102504:submit:config.model.cutoff_n_sessions=50
INFO:20180119_102504:submit:config.model.tree_depth=5
INFO:20180119_102504:submit:config.model.n_trees=20
INFO:20180119_102504:submit:config.model.random_sample_per_tree=100
INFO:20180119_102504:submit:config.log.logger_type=FILE
INFO:20180119_102504:submit:config.log.log_location=../../logging
INFO:20180119_102504:submit:config.log.log_level=INFO
INFO:20180119_102504:submit:config=OK
INFO:20180119_102504:submit:step-1=create_uri_summary.py
INFO:20180119_102504:submit:step-2=create_sessions.py
INFO:20180119_102504:submit:step-3=apply_models.py
```

# Sample log (.json)



```
INFO:root:{'spark': {'home': None, 'version': '2.1.0.2.6.0.3-8', 'environment': {'PYTHONHASHSEED': '0'}, 'user': 'a231384', 'conf':
[('spark.eventLog.enabled', 'true'), ('spark.yarn.historyServer.address', 'pdshdnn1p.standardbank.co.za:18081'), ('spark.history.ui.port',
'18081'), ('spark.driver.extraLibraryPath', '/usr/hdp/current/hadoop-client/lib/native:/usr/hdp/current/hadoop-client/lib/native/Linux-
amd64-64'), ('spark.history.kerberos.keytab', '/etc/security/keytabs/spark.headless.keytab'), ('spark.executor.id', 'driver'),
('spark.app.id', 'local-1526633937562'), ('spark.yarn.queue', 'default'), ('spark.driver.port', '40470'), ('spark.app.name', 'pyspark-
shell'), ('spark.executor.extraLibraryPath', '/usr/hdp/current/hadoop-client/lib/native:/usr/hdp/current/hadoop-client/lib/native/Linux-
amd64-64'), ('spark.driver.host', '10.144.164.203'), ('spark.history.kerberos.principal', 'spark-ds_hdp_prod@ZA.SBICDIRECTORY.COM'),
('spark.history.fs.logDirectory', 'hdfs:///spark2-history/'), ('spark.sql.catalogImplementation', 'hive'), ('spark.rdd.compress', 'True'),
('spark.history.provider', 'org.apache.spark.deploy.history.FsHistoryProvider'), ('spark.serializer.objectStreamReset', '100'),
('spark.master', 'local[*]'), ('spark.submit.deployMode', 'client'), ('hive.metastore.warehouse.dir', 'file:/home/a231384/rta/anomaly-
detection/sbg-dsds-fraud-anomaly-detection/helpers/digital_anomaly_detection/monitoring/spark-warehouse'), ('spark.port.maxRetries', '100'),
('spark.eventLog.dir', 'hdfs:///spark2-history/')]}, 'python': {'version': '3.4'}, 'start_time': '2018-05-18T11:03:04.926190', 'ds_env':
'\n', 'data': {'historical': 'hdfs:////dev/data/dsds/general/history_unzip', 'near_real_time': 'hdfs:////dev/data/dsds/general/nrt',
'list_of_hdfs_files': 'list_of_hdfs_files.txt'}, 'modules': ['IPython.core.shadowns', 'sklearn.linear_model', 'sys', 'pandas', 'json',
'logging', 'builtins', 'pickle', 'subprocess', 'time', 'requests', 'pyspark', 'types', 'py4j', 're', 'atexit', 'os', 'datetime', 'builtins',
'platform', 'random', 'numpy', 'configparser'], 'pyspark': {'submit': {'args': '\n'}}}
WARNING:root:{}
ERROR:root:{}
INFO:root:{'spark': {'home': None, 'version': '2.1.0.2.6.0.3-8', 'environment': {'PYTHONHASHSEED': '0'}, 'user': 'a231384', 'conf':
[('spark.eventLog.enabled', 'true'), ('spark.yarn.historyServer.address', 'pdshdnn1p.standardbank.co.za:18081'), ('spark.history.ui.port',
'18081'), ('spark.driver.extraLibraryPath', '/usr/hdp/current/hadoop-client/lib/native:/usr/hdp/current/hadoop-client/lib/native/Linux-
amd64-64'), ('spark.history.kerberos.keytab', '/etc/security/keytabs/spark.headless.keytab'), ('spark.executor.id', 'driver'),
('spark.app.id', 'local-1526633937562'), ('spark.yarn.queue', 'default'), ('spark.driver.port', '40470'), ('spark.app.name', 'pyspark-
shell'), ('spark.executor.extraLibraryPath', '/usr/hdp/current/hadoop-client/lib/native:/usr/hdp/current/hadoop-client/lib/native/Linux-
amd64-64'), ('spark.driver.host', '10.144.164.203'), ('spark.history.kerberos.principal', 'spark-ds_hdp_prod@ZA.SBICDIRECTORY.COM'),
('spark.history.fs.logDirectory', 'hdfs:///spark2-history/'), ('spark.sql.catalogImplementation', 'hive'), ('spark.rdd.compress', 'True'),
('spark.history.provider', 'org.apache.spark.deploy.history.FsHistoryProvider'), ('spark.serializer.objectStreamReset', '100'),
('spark.master', 'local[*]'), ('spark.submit.deployMode', 'client'), ('hive.metastore.warehouse.dir', 'file:/home/a231384/rta/anomaly-
detection/sbg-dsds-fraud-anomaly-detection/helpers/digital_anomaly_detection/monitoring/spark-warehouse'), ('spark.port.maxRetries', '100'),
('spark.eventLog.dir', 'hdfs:///spark2-history/')]}, 'python': {'version': '3.4'}, 'start_time': '2018-05-18T11:03:04.926190', 'ds_env':
'\n', 'data': {'historical': 'hdfs:////dev/data/dsds/general/history_unzip', 'near_real_time': 'hdfs:////dev/data/dsds/general/nrt',
'list_of_hdfs_files': 'list_of_hdfs_files.txt'}, 'modules': ['IPython.core.shadowns', 'sklearn.linear_model', 'sys', 'pandas', 'json',
'logging', 'builtins', 'pickle', 'subprocess', 'time', 'requests', 'pyspark', 'types', 'py4j', 're', 'atexit', 'os', 'datetime', 'builtins',
'platform', 'random', 'numpy', 'configparser'], 'pyspark': {'submit': {'args': '\n'}}}
WARNING:root:{}
ERROR:root:{}
```

# Monitoring

- Managing and monitoring statistical models is crucial if your organization periodically runs a large number (say, over 10) of statistical models.
- However, these issues are important even when there are just a few of them in production.

# Monitoring

Common challenges include the following:

- Keeping all the input correct and fresh.
- Making sure the outputs go to the right places, in the correct formats.
- Keeping the code organized for effective updating and maintenance.
- Creating and maintaining effective documentation.
- Assessing and tracking model performance.
- Effectively (preferably automatically) deciding when to update the model.

# Monitoring: all models

- Model: name.
- Environment: continuous development and integration; software and versions; hardware statistics; environment variables; run mode; current build version; source and run location; steps; extra packages and files; run command.
- Data: historical location; near real-time location; maximum folder age; logs start and end date; last history date; model last date; next run date; first NRT date.
- Results: submit status; total runtime; FLS alerts; loglines.

# Monitoring: supervised models

- Statastical process control: drift detection method (DDM); early drift detection method (EDDM).
- Sequential analysis: linear four rates (true –ve, false –ve, true +ve, false +ve) – specificity, recall, precision, accuracy; Monte Carlo sampling for significance level; Bonferoni correction for correlated tests.
- Error distribution monitoring: adaptive windowing (ADWIN)

# Monitoring: unsupervised models

- Clustering/novelty detection
- Feature distribution monitoring
- Model-dependent monitoring

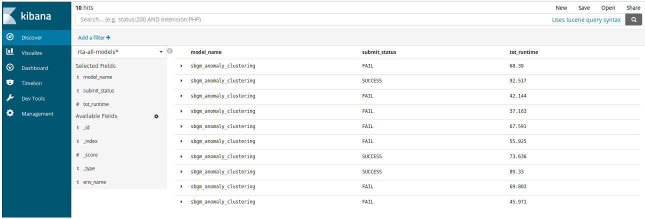
# Monitoring: random forests

- Number of URI's
- Time between fits
- Samples per tree
- Model start date
- Number of sessions to score
- Previous run date
- Maximum session length
- Last URI timestamp

# Monitoring: k-means

- Database name
- Results
- Alerts to FLS
- Model path
- List of features
- Clusters

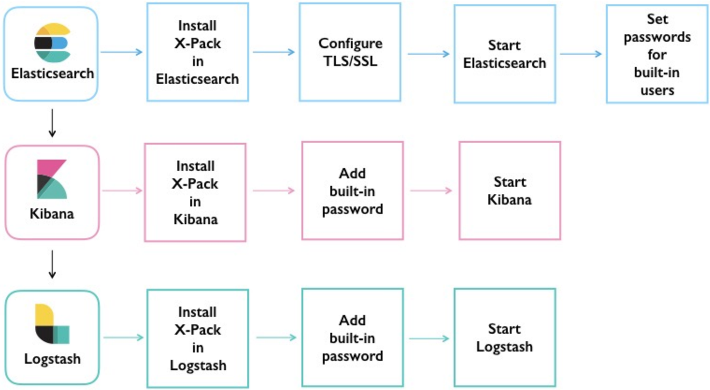
# Visualisation



10 hits

| model_name | submit_status | tot_runtime |
|---|---|---|
| sbgm_anomaly_clustering | FAIL | 68.39 |
| sbgm_anomaly_clustering | SUCCESS | 92.517 |
| sbgm_anomaly_clustering | FAIL | 42.144 |
| sbgm_anomaly_clustering | FAIL | 37.163 |
| sbgm_anomaly_clustering | FAIL | 67.591 |
| sbgm_anomaly_clustering | FAIL | 55.925 |
| sbgm_anomaly_clustering | SUCCESS | 73.636 |
| sbgm_anomaly_clustering | SUCCESS | 89.33 |
| sbgm_anomaly_clustering | FAIL | 69.803 |
| sbgm_anomaly_clustering | FAIL | 45.971 |

# X-Pack



Elasticsearch

Install X-Pack in Elasticsearch

Configure TLS/SSL

Start Elasticsearch

Set passwords for built-in users

Kibana

Install X-Pack in Kibana

Add built-in password

Start Kibana

Logstash

Install X-Pack in Logstash

Add built-in password

Start Logstash

# X-Pack capabilities

- Security
- Alerting
- Monitoring
- Reporting
- Graph
- Machine learning

# X-Pack: machine learning

- Time series analysis
- Anomaly detection
- https://www.youtube.com/watch?v=n6xW6YWYgs0

# References

1. Building Intelligent Systems: A Guide to Machine Learning Engineering. [Geoff Hulten] (Apress, 2018)
2. Trends in AI, Data Science, and Big Data. [Ben Lorica] (2017)
3. Building Evolutionary Architectures. [Rebecca Parsons; Patrick Kua; Neal Ford] (O'Reilly Media, 2017)
4. 5 things you should be monitoring. [Brian Brazil] (2018)
5. The Logstash Book. [James Turnbull] (Turnbull Press, 2013)
6. Beyond the Twelve-Factor App. [Kevin Hoffman] (O'Reilly Media, 2016)
7. Logs and real-time stream processing. [Jay Kreps] (2016)
8. I Heart Logs: Apache Kafka and Real-time Data Integration. [Jay Kreps] (2015)
9. The log: The lifeblood of your data pipeline. [Kiyoto Tamura] (2015)
10. Understanding the ELK stack. [Brian Anderson; Rafał Kuć] (2016)