ELASTICSEARCH

LOGSTASH

WHAT IS?

- Search engine document-oriented
- JSON based, Apache Lucene engine
- Schema free
  - Infer mapping from posted documents
  - Auto merge mappings when possible
- Distributed
  - Scale out (more nodes)
  - Scale up (more powerful nodes)
- API centric & RESTful

WHAT CAN IT DO?

- Unstructured search
  - Find all companies in the search domain
- Structured search
  - Find all companies founded between 2010 and 2012
- Analytics
  - Return the average, min, max etc. annual revenue for all companies
  - Histogram documents by date
- Dramaticaly increase experience via plugins

BASIC GLOSSARY

- cluster: a set of *nodes* running with the same *cluster_name*
- node: a running instance of elasticsearch
  - Master nodes
  - Data nodes
  - On startup a node will use multicast to discover existing cluster and will try to join it
- index: a named collection of documents
- shard: a single Lucene instance, shards are distributed across nodes and can move from node to node in case of failure
  - Primary
  - Replica
- document

CONFIGURATION: elasticsearch.yml

Very easy initial setup: base config is ready to be run and tested

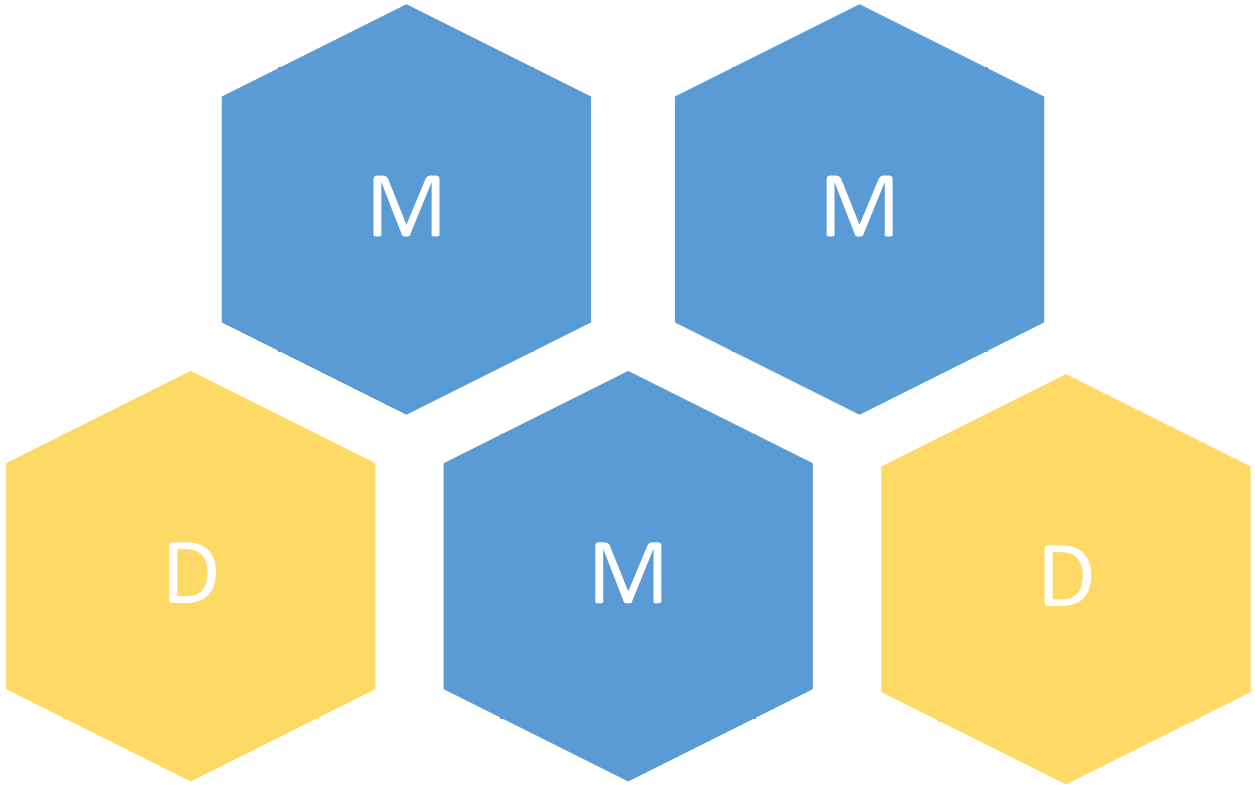
- cluster.name: "elasticsearch"
- **node.master: true**
- **node.data: true**
- index.number_of_shards = 5
- index.number_of_replicas = 1
- bootstrap.mlockall: true
- http.port: 9200
- transport.port: 9300

CONFIGURATION: elasticsearch.yml

configure node role with data & master flag

	node.master: true	**node.master:false**
node.data: true	all-round node, eligible as master and data storage	data storage
node.data: false	node coordinator	load balancer

HANDY STARTING CLUSTER CONFIGURATION



3 (eligible)master nodes, **2** data nodes

5 shards per index, **1** replica per shard

BASIC APIs

each HTTP verb is mapped to an ElasticSearch action

- `PUT`: insert documents into indexes
- `POST`: insert documents with autogenerated fields, or update docs
- `GET`: retrieve single a document
- `DELETE`: you guess
- `HEAD`: check if document is indexed(exists) w/o retrieving it

EXAMPLE API: index

```
PUT  'localhost:9200/some-index/doc-type/156' '{
     "name": "Vlad",
     "last_name": "Saftoiu",
     "age": 15
}'
```

index is also used for *re-indexing* documents (replacement)

response:

`201 created`	for first time
`200 ok`	for re-index

WHAT IS

the swiss-army knife for log management:

- collect logs from different sources (files, syslog, mqueues …)
- transform input data with different filters
- ship the result to any other destination (even reintroduce the log in the same stream)

LOGSTASH PIPELINE

INPUTS

Listen for changes in the source and pass events to the next step, virtually any source we encounter

- File
- Syslog
- Redis
- TCP, UDP
- Log4j
- …. many many more

FILTERS

Process events and transforms data, each filter can add / remove a field from the message, set tags

- grok
- json
- drop
- clone
- geoip
- combine filters (orderwise)

OUTPUTS

Ships transformed data to choosed destination(s)

- elasticsearch
- file
- graphite
- statsd

More destinations can be specified

EXAMPLE INPUT: file

```
input {
        file{
                path => [ "/var/log/messages", "/var/log/*.log" ]
                exclude => ["/var/log/boring.log"]
                add_field=> {"filesrc" => "no_boring" }
        }
}
```

- Like `tail -0F`
- each line represents one event
- Look for all files in a dir, or exclude some of them
- Add/remove fields

EXAMPLE FILTER: json

```
filter {
  json {
    source => "message"
  }
}
```

takes a json field **from** the event, parse it and add its properties to the event root

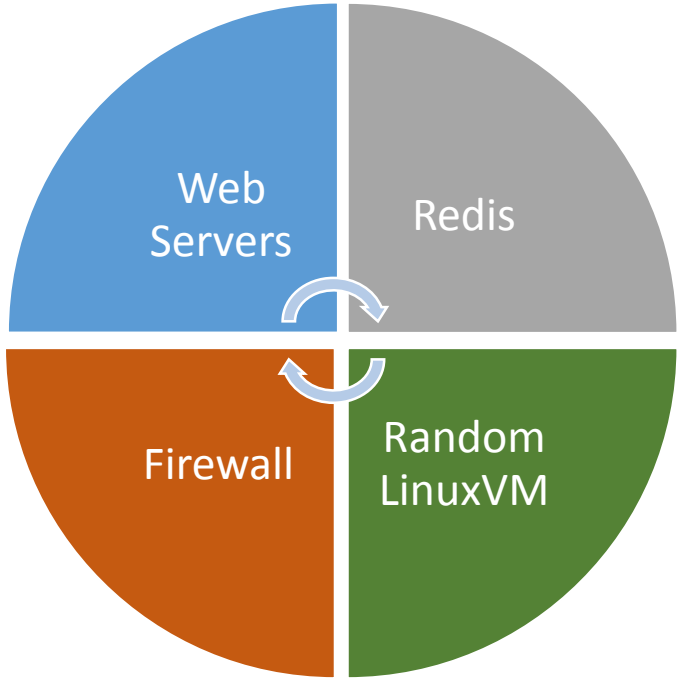
probably best option for shipping custom events to elasticsearch, since provides maximum flexbility for analytics

EXAMPLE FILTER: grok

- one of the most used filters for logstash
- It's a big catalog of regex expressions for quite any situation
- docs says: "Grok is currently the best way in logstash to parse crappy unstructured log data into something structured and queryable"
- grok pattern => `%{SYNTAX: semantic}`
- Examples:
- `%{NUMBER:bytes}`
- `%{LOGLEVEL:level}`
- `%{COMBINEDAPACHELOG} %{SYSLOG}`
- …and 120 more

PROBLEM

we have many different systems and programs that interact together but do not share their execution information logs



- one file per service/program
- authentication required on target machine
- different log format (datetime, bool …)
- Hard to get the big picture

POSSIBLE SOLUTION

- collect logs from each source
- transform fields reducing diversity
- centralize to elasticsearch