Elastic Brno meetup #2

Elasticsearch at NetSuite Infrastructure

Andrej Golis, System Engineering
Ondřej Kos, Cloud Test Engineering

January 22nd, 2019

What do we use Elasticsearch for?

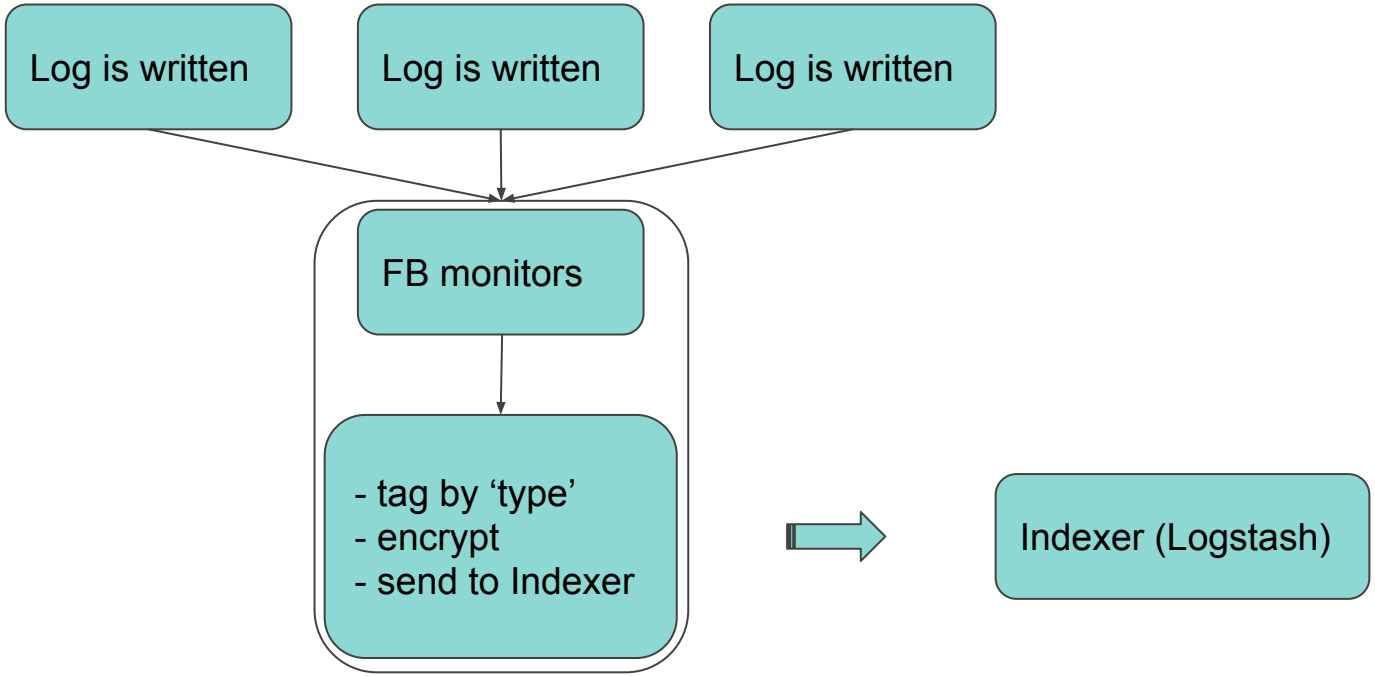
Mainly for log aggregation:

- Operation logs of systems - used by SRE and Infrastructure teams
- Application logs - used by Developers
- Monitoring logs - used by SRE team and ELK cluster maintainers (System Engineering)

But also for search:

- Indexing of items for search provided by the application
- Data back-propagation and QA tools

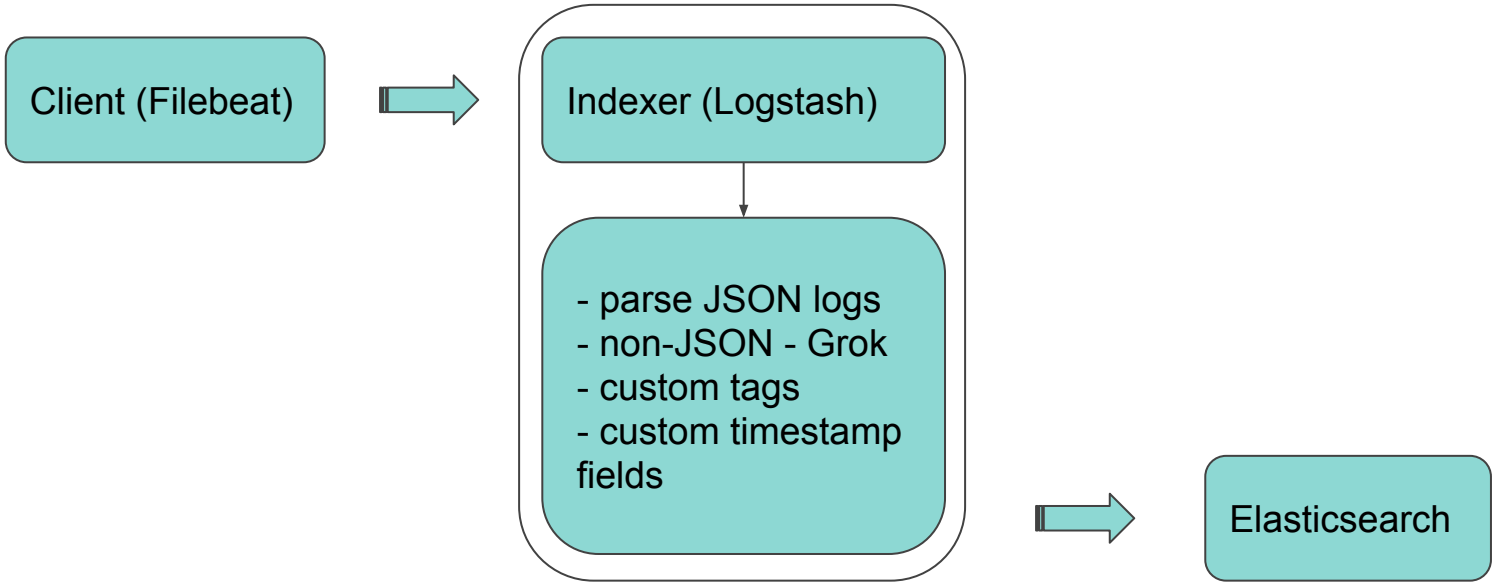
Log aggregation - Filebeat



Log aggregation - Filebeat

- Log aggregation begins with logs written to local files on the source system
- Filebeat daemon monitors specified files for new lines
- Multiple Filebeat instances running on the system depending on the type of system in order to ship logs to different Elastic clusters
- Filebeat tags logs by their ‘type’ and:
  - Ships them to indexers (logstash)
  - Encrypted
  - In batches (pipeline_batch_size: 500)

Log aggregation - Logstash



Log aggregation - Logstash

Once indexer / Logstash receives batches of logs from beats, it runs them through series of filters to prepare for indexing into Elasticsearch:

- Parse JSON logs into fields
- Parse non-JSON logs into fields using Grok patterns
- Set custom tags and custom timestamp fields:
  - Time when the log is received
  - Log's original timestamp

The latter enables us to compute **delay between log occurence and processing**.

Once processed, batches of logs are sent to ES - index name => append time of the log

Log aggregation - Elasticsearch

Each node in the cluster serves a unique role:

- Data
- Master
- Searcher
- Indexer (Logstash)

Each Data node runs 4 instances of ES:

- 3 warm instances are backed by 4 spindle drives - RAID5
- 1 hot instance is backed by FIO card - fast storage, main function is writing logs

Log aggregation - Logstash & ES

So where exactly does Logstash send the logs to?

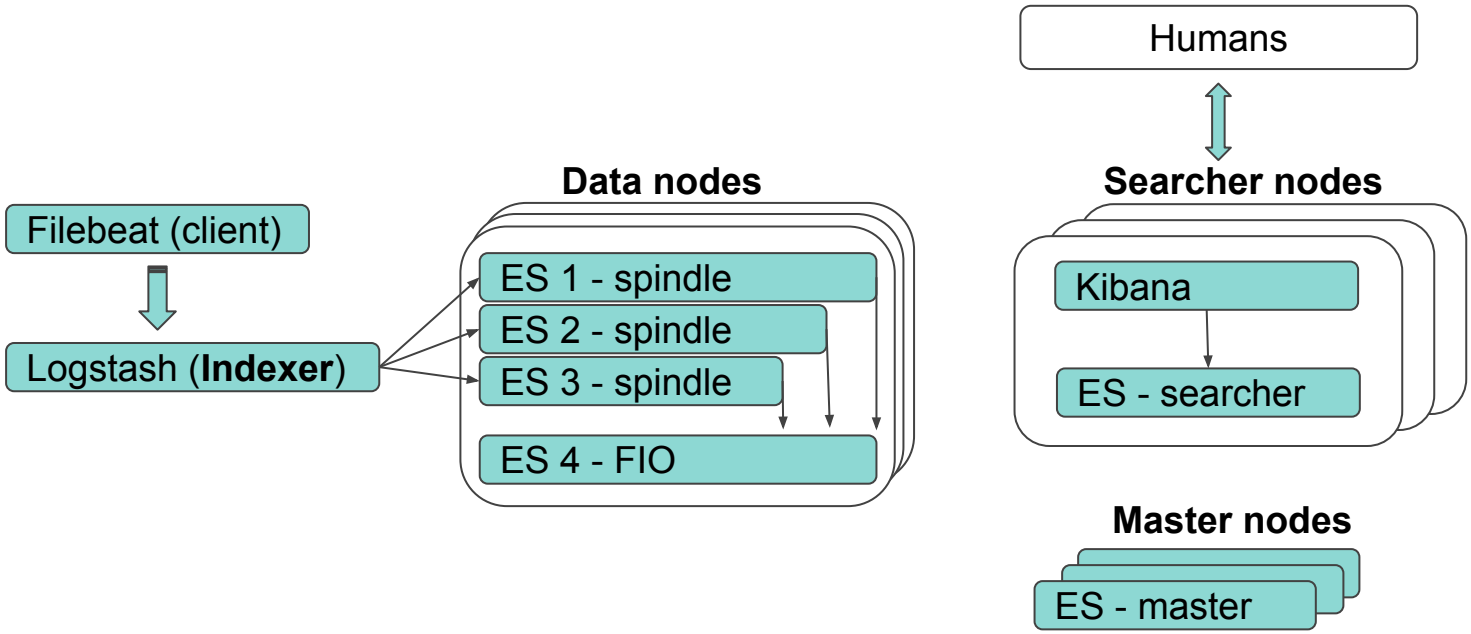
Logstash uses the 10G IPs and ports of the **3 instances** with spindle (slow) disks, NOT the FIO one.

Those 3 instances have very little load on their own and act as **coordinators** for bulk indexing **to fast instances** with FIO storage.

This frees up the fast instance to its main purpose - just writing logs.

Putting it all together

Application and Operation logs cluster set up

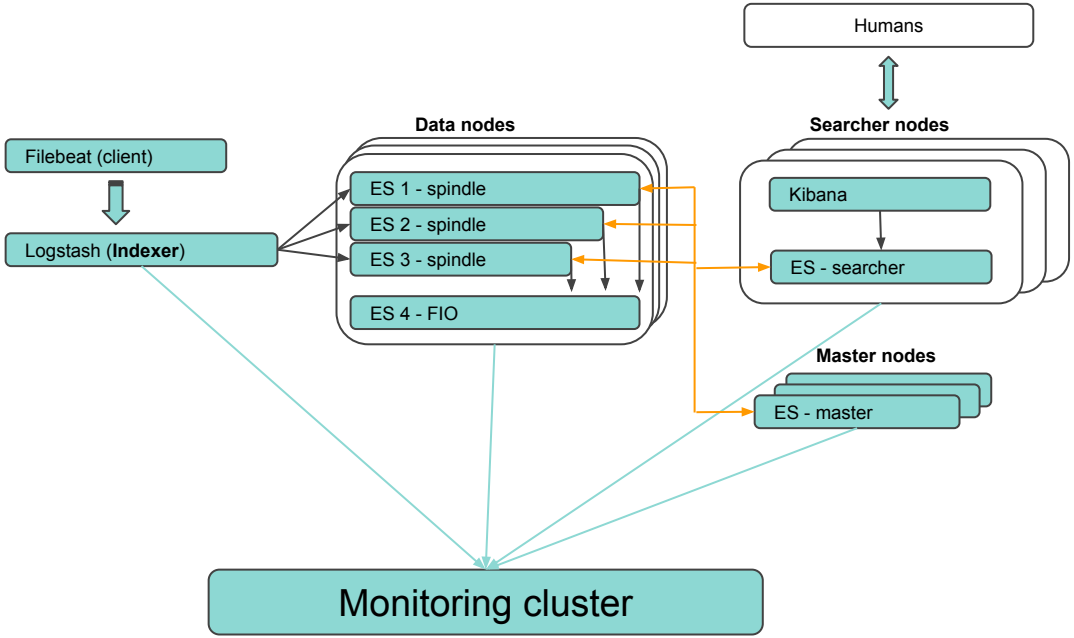


So where do the metrics go?

To a monitoring cluster!

All Elasticsearch, Kibana and Logstash instances send their metrics data to a separate Monitoring cluster, which is built just for this purpose.

Metrics sent to Monitoring cluster



Monitoring cluster

- It is a Logging cluster in every aspect
- Differentiates from the Application and Operation logging clusters by:
  - Receives metrics from all the other clusters including its own
  - Runs Curator on all logging clusters

Configuration of the monitoring cluster it the same as logging clusters.

Which metrics are gathered?

- Indexing rate and latency
- Search rate and latency
- Index size and document counts
- CPU and disk utilization
- Heap utilization
- JVM pool utilization
- GC frequency and duration

Curator

- Runs on the master nodes of the Monitoring cluster
- Maintains all clusters including itself

Basic function:

- Allocate indices from FIO drives to spindle drives (periods_till_warm)
- Close indices (periods_till_close)
- Delete indices (periods_till_delete)

Some stats now!

Application logging cluster

Elasticsearch

- 140 nodes
- 1100+ indices
- 630+ TB of data
- 700 000 000 000+ documents
- 4TB of JVM Heap

Indexing latency - ~1.5 ms

Search latency - ~30ms

Indexing rate - 130 000+ primary shards per second

Operation logging cluster

Elasticsearch

- 132 nodes
- 2000+ indices
- 250+ TB of data
- 300 000 000 000+ documents
- 3.9TB of JVM Heap

Indexing latency - ~0.5 ms

Search latency - ~30ms

Indexing rate - 150 000+ shards/ps

Monitoring cluster

Elasticsearch

- 30 nodes
- 2200+ indices
- 55+ TB of data
- 45 000 000 000+ documents
- 800GB of JVM Heap

Indexing latency - ~0.5 ms

Search latency - ~0.3ms

Indexing rate - 16 000+ shards/ps

Where do we test changes?

On our Development cluster, which is quite small, but still:

- 29 nodes
- 580+ indices
- 55+ TB of data
- 2 300 000 000+ documents
- 865 GB of JVM Heap

Indexing latency - ~0.2 ms

Search latency - ~0.2 ms

Indexing rate - 1 500+ shards/ps

Q & A

Thank you!