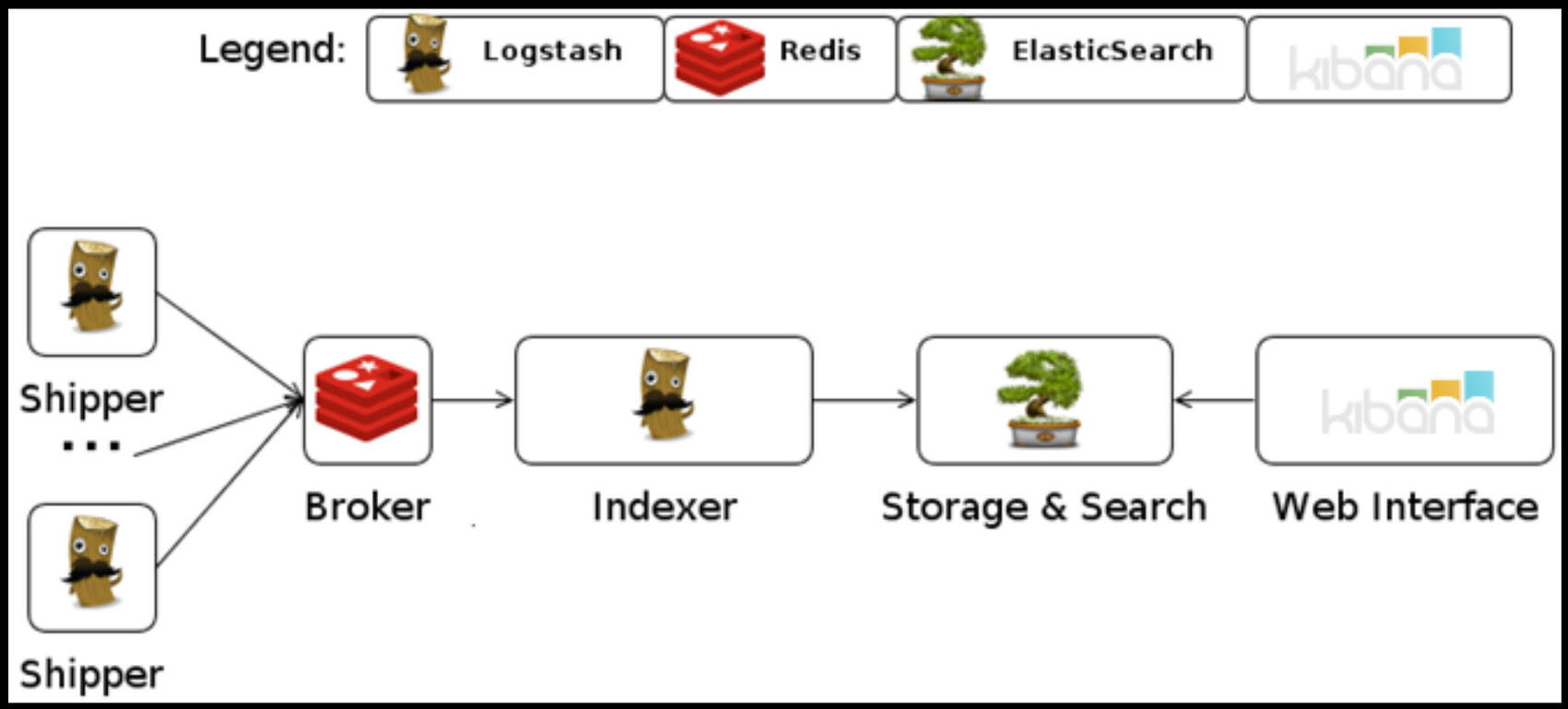
Herding ELK: Scaling Towards A Trillion Logs

Breandan Dezendorf
breandan@42lines.net
@bwdezend

What Is ELK?

elk, *n*;
1. Also called European elk. the moose, *Alces alces*.
2. Also called American elk, wapiti. a large North American deer, *Cervus canadensi*s, the male of which has large, spreading antlers.
3. A series of throughput bottlenecks cleverly disguised as a java-based log aggregation cluster

What Is ELK?



42 LINES

fitbit

The Road So Far

- Started at 35,000 logs/s, 6 days of retention (15-18 billion logs before replication)
- Currently averaging 150,000 log/s, 30 days of retention (350 billion logs before replication)
- Expected changes in architecture suggests we need to sustain > 300,000 logs/s by Q2 of 2017 with burst capacity at 400,000 logs/s.
- Current peak sustained indexing rate is 350,000 logs/s

fitbit

No Tracker Data!

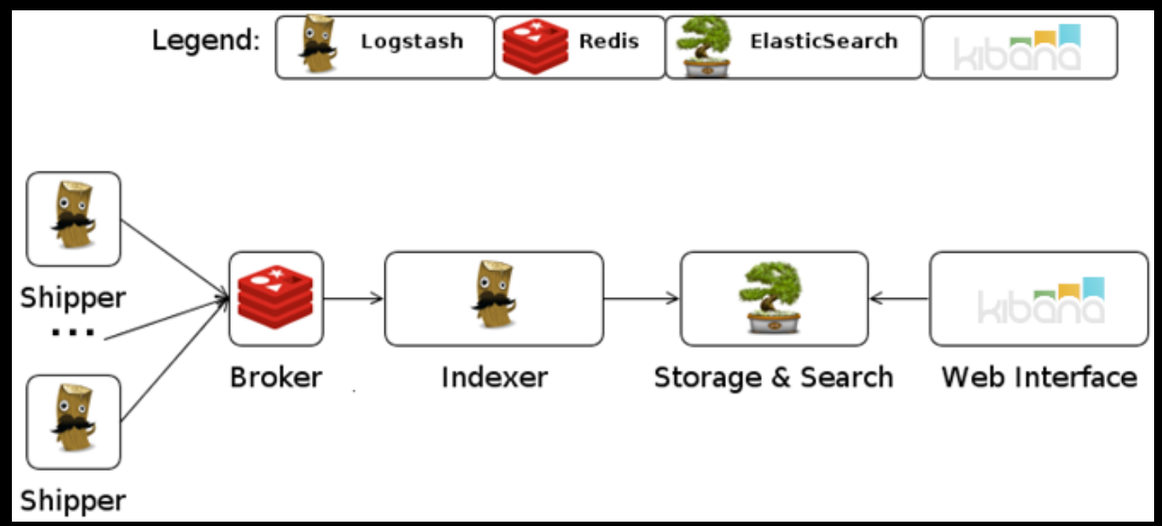
- Fitbit never puts tracker data in ELK
- No step counts, no heartbeats, no elevation changes
- Only application/firewall/database logs

fitbit

Three Kinds Of Search

- Release Monitoring / Incident Management
- Historical Surveys
- Historical Incidents

In The Beginning



- 35,000 logs/s average, with peaks of 45,000 during the day's high-water mark
- 55,000 logs/s during indexing stall recovery

42 LINES

fitbit

In The Beginning

- 3 masters (16GB RAM, spinning disks)
- 30 data nodes (64GB RAM, SSDs)
- 4 api nodes (64GB RAM, SSDs)
- 4 Redis hosts (16 GB RAM, spinning disks)

fitbit

Scaling and Performance

- **`indices.breaker.fielddata.limit`** can only guess how much memory will be used
- ZenDisco has a fixed 30 second timeout (with two retries)
- Indexing will fight with search for resources
- `facet` searches are expensive and consume lots of heap

Scaling and Performance

- haproxy logs were high enough volume that logstash/redis couldn't keep up. UDP to the rescue.
- Performance was still bad, so all haproxy logs with 200-series http status codes were dropped
- Grok parsing is expensive on logs
- mysql slow logs have binary data

“Change is the only constant.”

–Heraclitus of Ephesus

New Design Goals

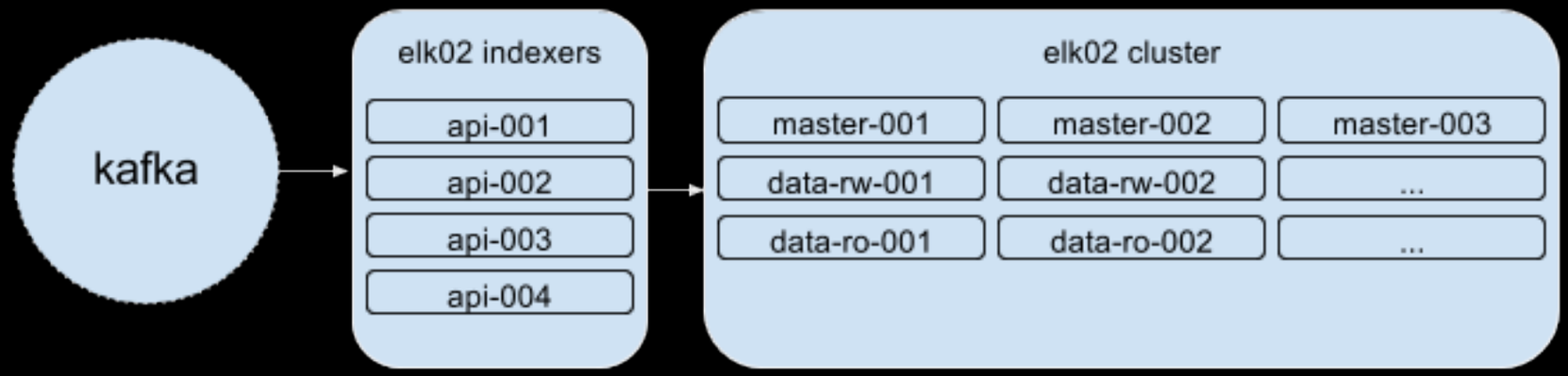
- Increase retention from 5 to 30 days
- Double the number of hosts sending logs
- Index a larger set of log types
- Stop dropping haproxy 200s
- Handle two years of growth, including spikes to 2x traffic

The New Cluster: ELK02

- Replace Redis with Apache Kafka
- Move to hot/warm architecture
- Archive logs to S3
- Analyze kibana query logs

fitbit

ELK02 Design



42 LINES

fitbit

Redis vs Kafka

- Redis has no cluster needs, and is simple to operate/understand
- With Logstash + Redis, there is limited backlog queuing (memory based redis queue)
- Data can only be consumed once
- Standing up more single points of failure (queues) wasn't going to scale

A Kafka Shaped Problem

- Visibility into queue depth and statistics
- Fault tolerant and performant on SATA drives
- Multiple consumers of the same data
- Remember: garbage collection tuning

A Kafka Shaped Problem

```
KAFKA_HEAP_OPTS="-Xmx2G -Xms2G"

KAFKA_JVM_PERFORMANCE_OPTS="-server -
XX:PermSize=48m -XX:MaxPermSize=48m -XX:
+UseG1GC -XX:MaxGCPauseMillis=20 -
XX:InitiatingHeapOccupancyPercent=35"

limit nofile 65536 65536
```

42 LINES

fitbit

Hot/Warm Tiers

- Separate search from indexing
- Live data is indexed into hosts with only 48 hours of indexes
- More memory, CPU and disk I/O available for indexing incoming data
- Older data has different access patterns, and spinning disks are good enough*

Hot/Warm Tiers

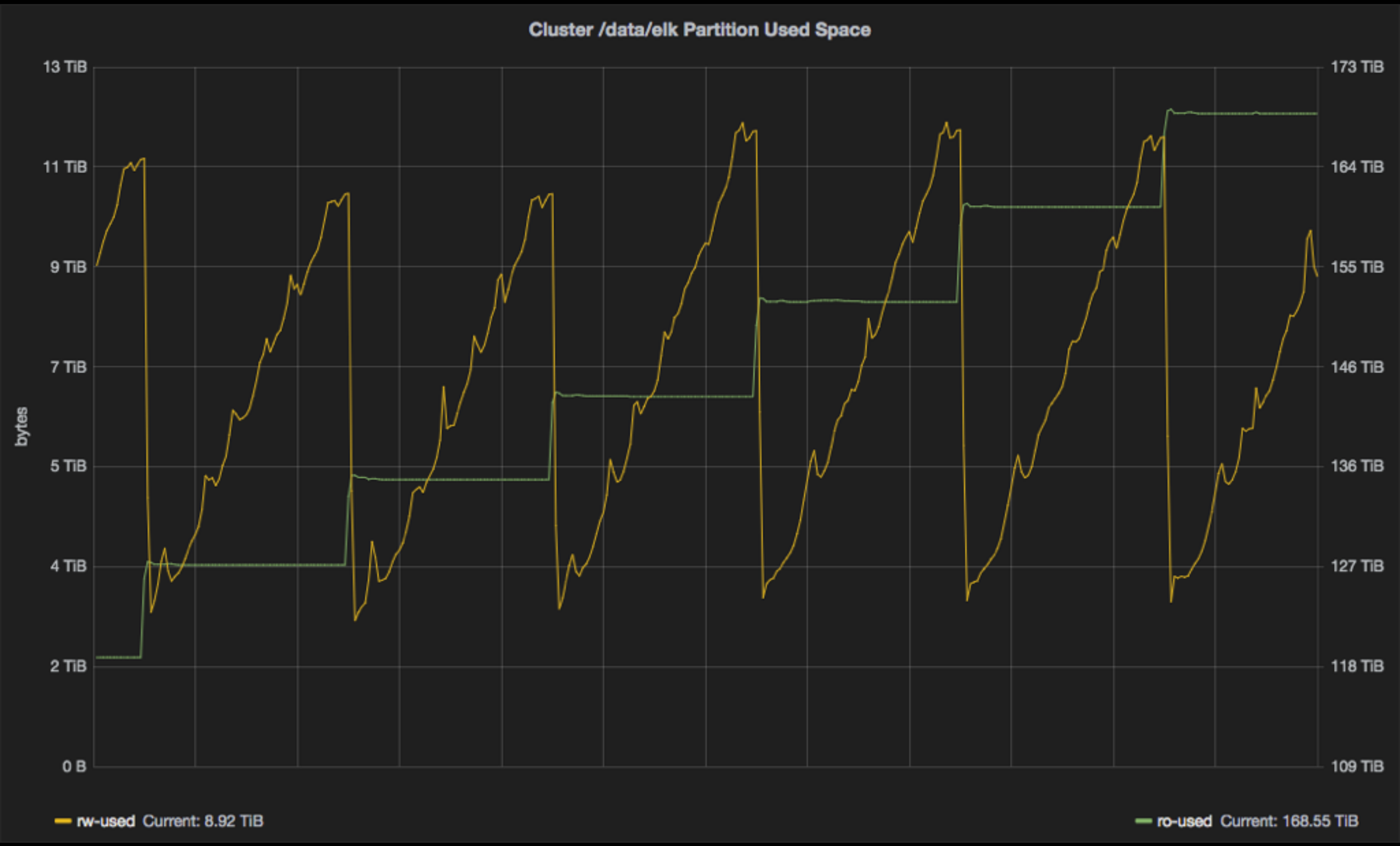
- Data migration between tiers is simple:

```
node.tag_tier: elk02-data-rw
```

```
node.tag_tier: elk02-data-ro
```

```
curl -XPUT localhost:9200/logstash-2015-01-01-main/_settings -d '{
    "index.routing.allocation.exclude.tag": "elk02-data-rw",
    "index.routing.allocation.include.tag": "elk02-data-ro"
}'
```

fitbit



42 LINES

fitbit

Kibana Query Logs

- Ingest elasticsearch index and slow query into elasticsearch
- Be careful of logging loops!
- You can now see the queries, how efficient they are, and what indexes are being accessed
- Very useful to understand what the users want to do, and helpful when there are problems

fitbit

Kibana Query Logs

```
if [type] == "elasticsearch" {
    grok {
        match => { "message" => "%
{DATESTAMP:timestamp}\]\[%{WORD:level}\s*\]\[%
{NOTSPACE:module}\s*\]( \[%{NOTSPACE:node}\s*
\])? %{GREEDYDATA:es_message}" }
    }
}
```

fitbit

```
  if [type] == "es_slow_query" {

    grok {
      match => { "message" => "%{DATESTAMP:timestamp}\]\[%{WORD:level}\s*\]\[%{NOTSPACE:module}\s*\]( \[%{NOTSPACE:node}\s*\])?
( \[%{NOTSPACE:index}\]\[%{NUMBER:index_shard}\])? took\[%{NOTSPACE:took}\], took_millis\[%{NUMBER:took_millis}\], types\[(%
{NOTSPACE:types})?\], stats\[(%{NOTSPACE:stats})?\], search_type\[(%{NOTSPACE:search_type})?\], total_shards\[%
{NUMBER:total_shards}\], source\[%{GREEDYDATA:source_query}\], extra_source\[(%{GREEDYDATA:extra_source})?\]," }
    }

    grok {
      match => { "source_query" => "{\"range\":{\"@timestamp\":{\"gte\":%{NUMBER:query_time_gte},\"lte\":%
{NUMBER:query_time_lte},\"format\":\"epoch_millis\"}}}" }
      match => { "source_query" => "{\"range\":{\"@timestamp\":{\"to\":\"now-%{DATA:query_time_to}\",\"from\":\"now-%
{DATA:query_time_from}\"}}}" }
      match => { "source_query" => "{\"range\":{\"@timestamp\":{\"to\":\"%{DATA:query_time_date_to}\",\"from\":\"%
{DATA:query_time_date_from}\"}}}" }
    }

    if [query_time_date_to] {
      ruby {
        code => "event['query_time_range_minutes'] = (( DateTime.parse(event['query_time_date_to']) -
DateTime.parse(event['query_time_date_from']) ) * 24 * 60 ).to_i"
      }
    }

    if [query_time_gte] {
      ruby {
        code => "event['query_time_range_minutes'] = (event['query_time_lte'].to_i - event['query_time_gte'].to_i)/1000/60"
      }

      ruby {
        code => "event['query_time_date_to'] = DateTime.strptime(event['query_time_lte'],'%Q').to_s"
      }

      ruby {
        code => "event['query_time_date_from'] = DateTime.strptime(event['query_time_gte'],'%Q').to_s"
      }
    }

    mutate {
      add_field => { "statsd_timer_name"  => "es.slow_query_duration" }
      add_field => { "statsd_timer_value" => "%{took_millis}" }
      add_tag   => [ "metric_timer" ]
    }

  }
```

Long Term Archiving

- Logstash reads from kafka and uses the file output plugin to store logs on disk
- Use pigz -9 to compress the files
- AES encrypt and sync to S3
- Set a lifecycle policy to roll logs into STANDARD_IA after 90 days

fitbit

Long Term Archiving

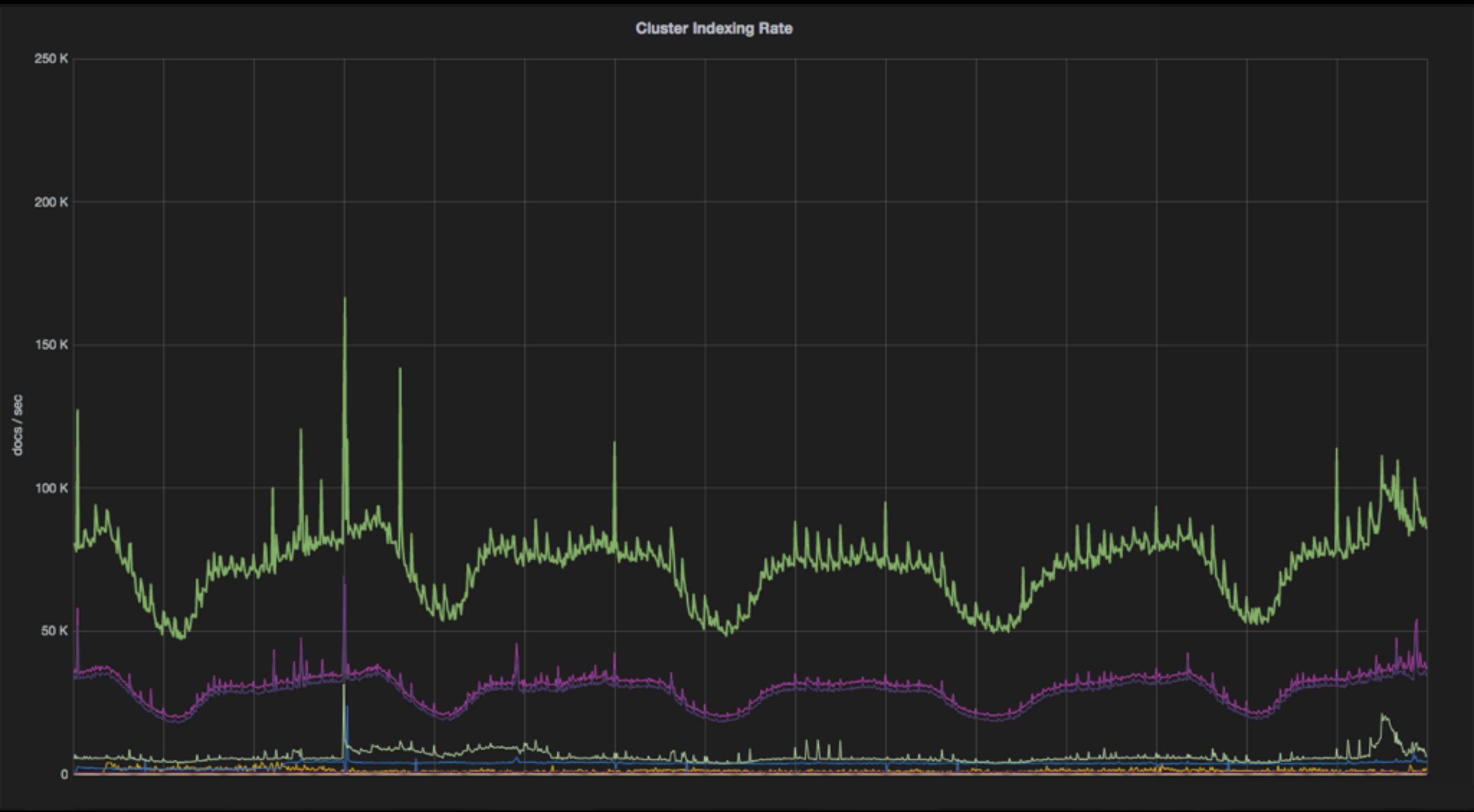
- Be careful to watch TCP window sizes
- Bandwidth delay products can be nasty here
- Logstash's default is a fixed window of 64KB, which on a high latency link will painfully limit replication speeds

fitbit

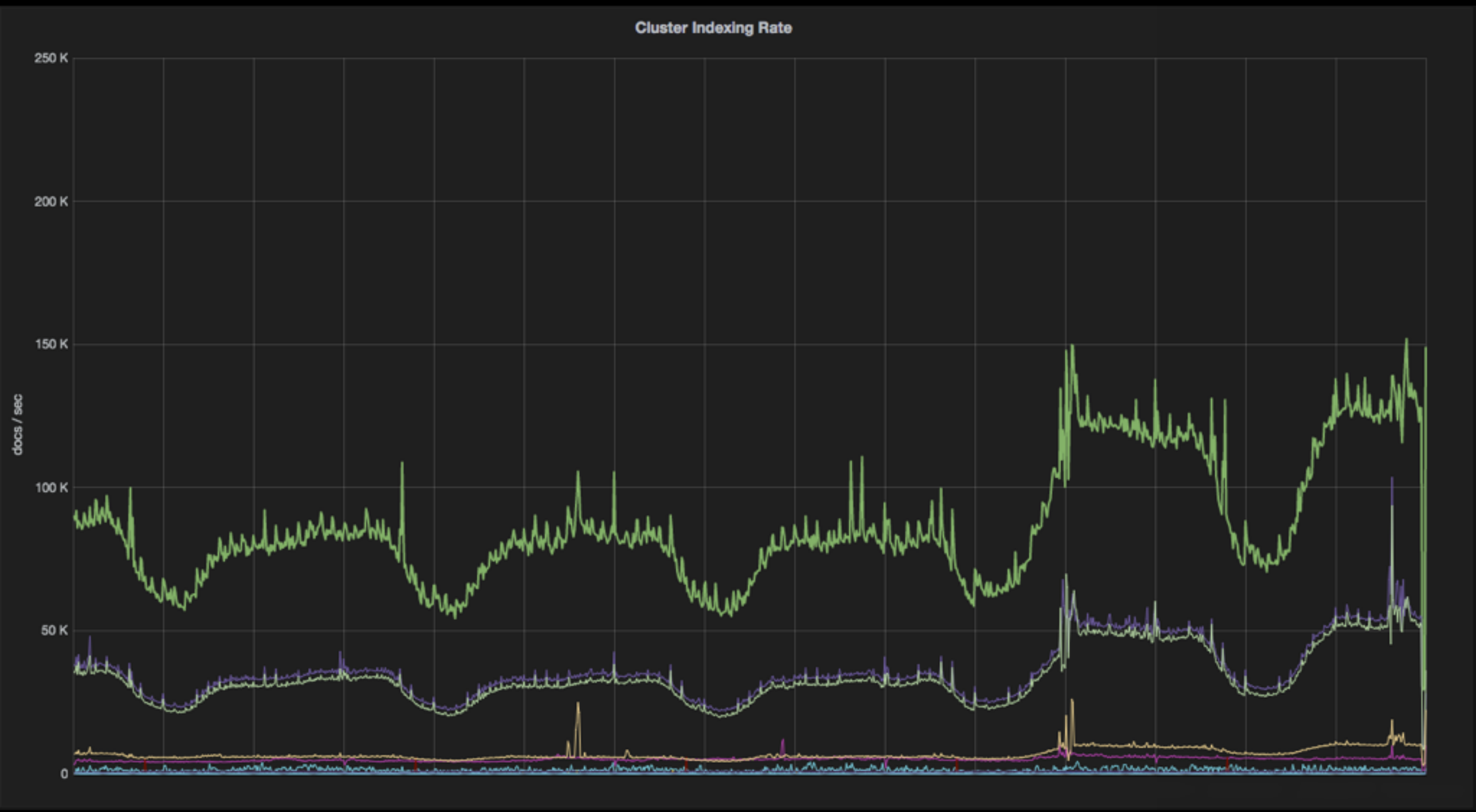
“Change is the only constant.”

–Heraclitus of Ephesus

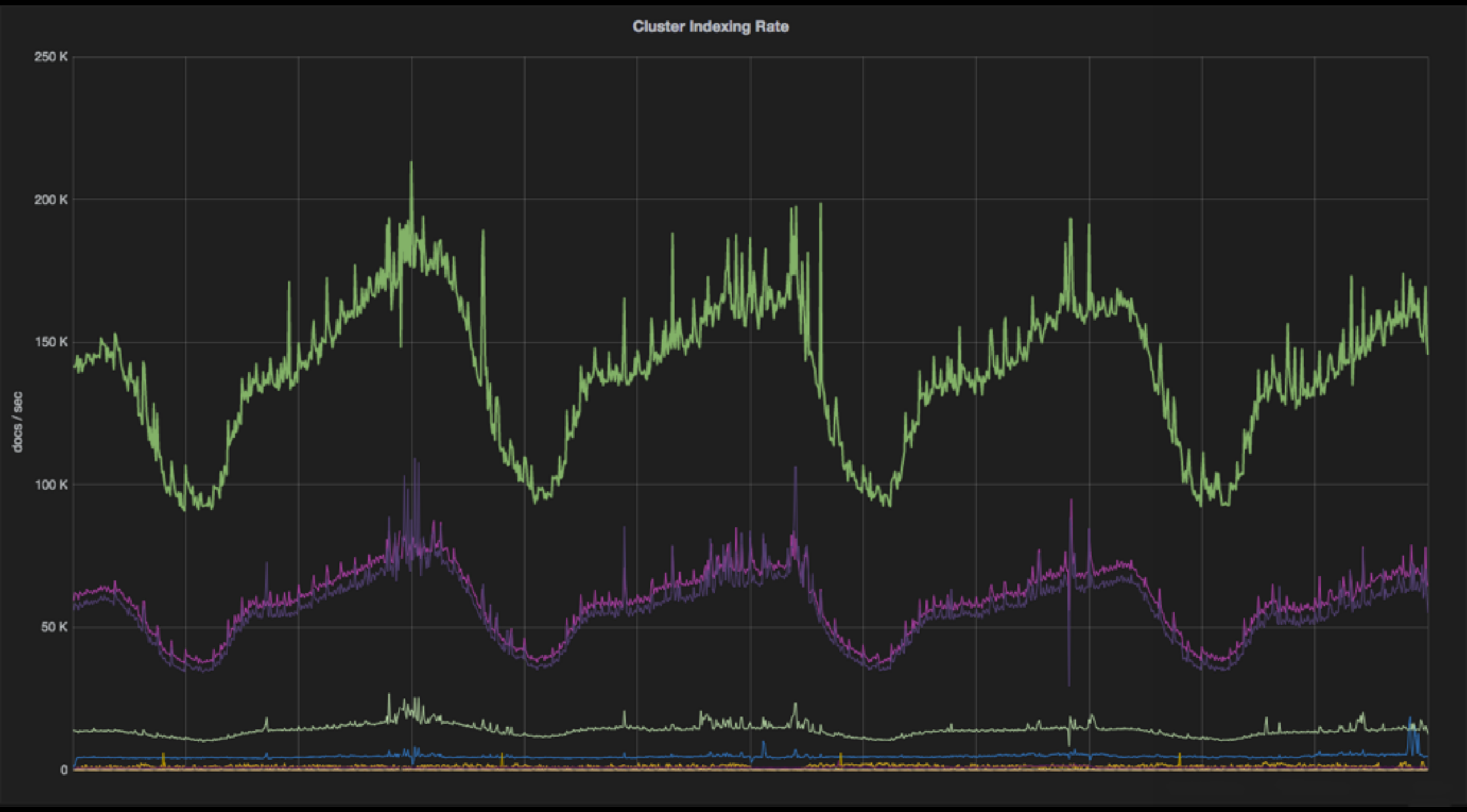
Typical 5 Day Indexing



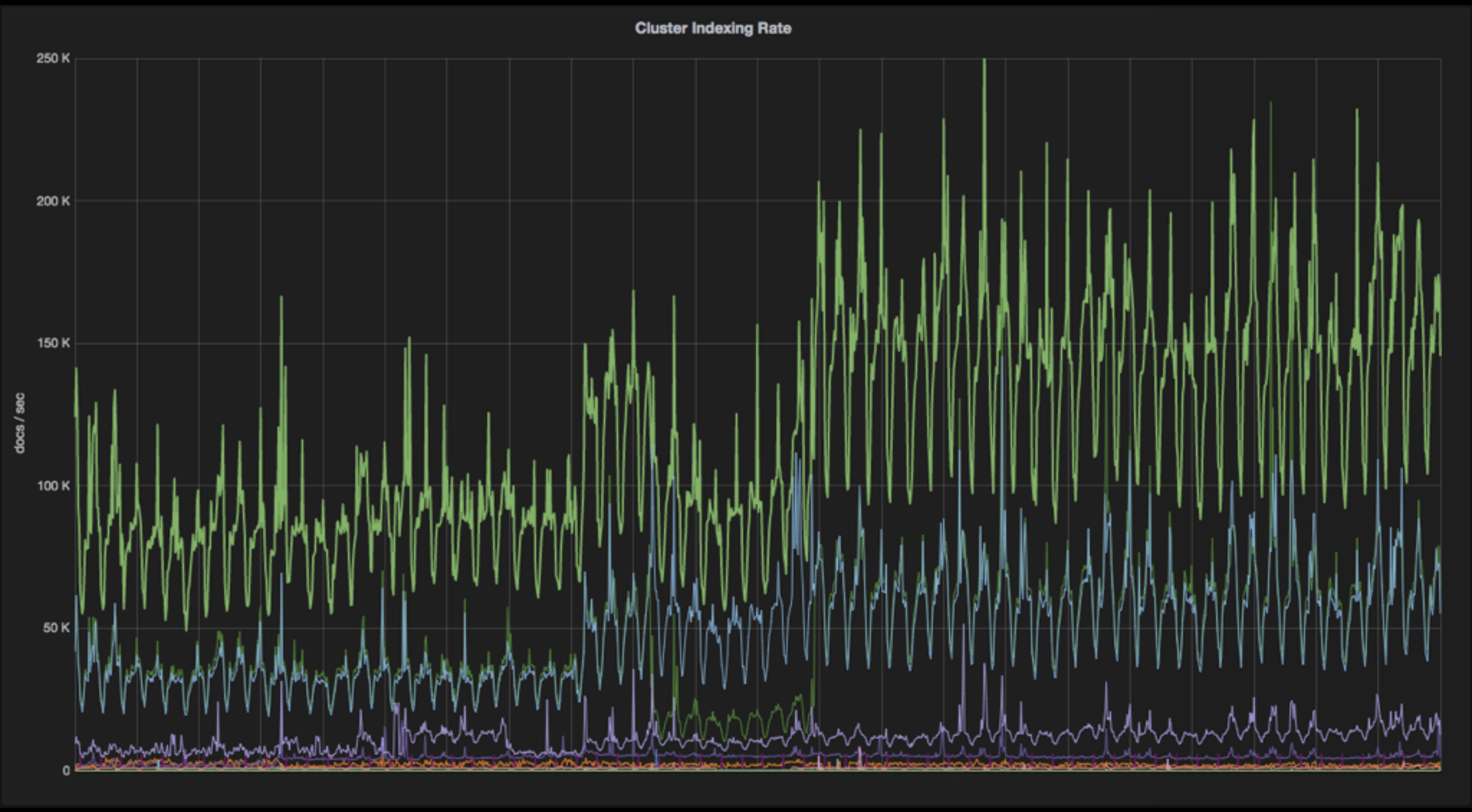
What's This?



This isn't a spike



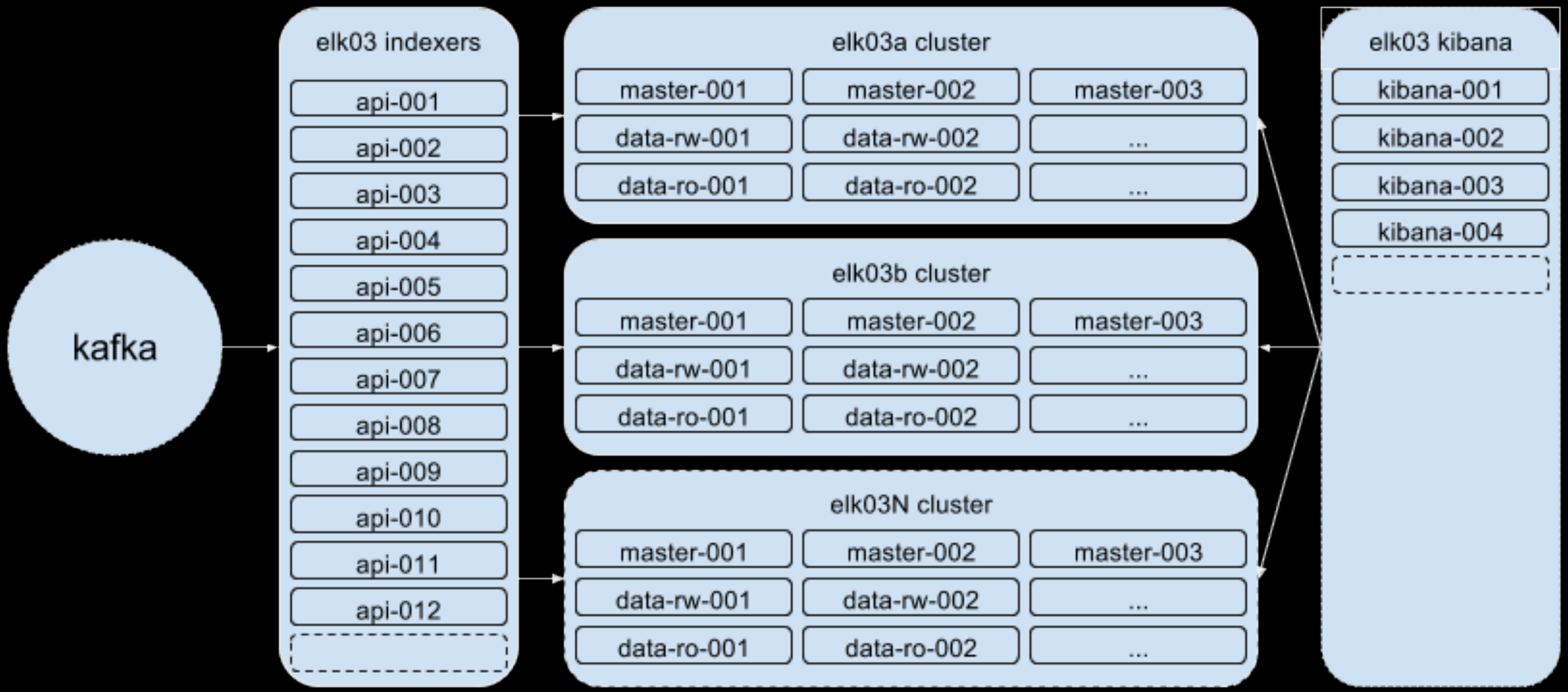
This is the new normal



The New New Cluster: ELK03

- Support arbitrary doublings of log traffic with little notice (3-6 weeks)
- Limit cluster node counts
- Limit the impact of failure
- elasticsearch 2.x and kibana4.x
- Kafka lets us run multiple clusters in parallel

ELK03{a,b...n}



42 LINES

fitbit

Tribe Nodes

- Tribe nodes act as a client node that talks to two clusters at the same time
- They are full members of each cluster, for all the good and bad that entails
- Hosting Kibana on a tribe node lets you query both at the same time and merge search results seamlessly

Limit Cluster Node Count

- Every node in an elasticsearch cluster makes (and holds open) 13 TCP connections to every other node
- Every node must ack changes to the cluster
- ZenDisco has timeouts that are set for a reason
- Using tribe nodes, we now scale ELK tested units rather than growing tiers

42 LINES

fitbit

Each subcluster has:

- 3 master hosts
- 30 data-rw hosts
- 80 data-ro hosts
- 8 indexer hosts

- 3 master hosts
- 30 data-rw hosts
- 50 data-ro hosts
- shared indexers

42 LINES

fitbit

es 2.x benefits

- Performance enhancements and bug fixes
- Kibana4 releases are tied to elasticsearch2.x releases
- Improvements for doc_values and other field related settings

es 2.x... opportunities

- Fieldnames may no longer contain dots
- No kibana3 support. Facets were deprecated in elasticsearch 1.x, and in 2.x they were removed
- Lots of testing needed to validate the impact of other changes

Consistent Hashing

- For cost and practicality reasons, the decision was made to only keep a portion internal load balancer logs

- Without hashing, we would already have more than 700 billion documents in the cluster, and need to scale to 1.25 trillion in the next 6 months

- How do you make sure to keep all logs from a single request, and consistently drop the rest?

Consistent Hashing

- Tag logs at the load balancer with a **`request_id`**
- Add to subsequent logs throughout the stack

```
if [request_id] {

  ruby {

    code => "require 'murmurhash3'; event['request_id_sampling'] =
(MurmurHash3::V32.str_hash(event['[request_id]']).to_f/4294967294)"

  }

}
```

42 LINES

fitbit

Consistent Hashing

- Also set a flag for "never drop my logs"

```
if [request_id_sampling] {

  # Drop a percentage of request_id_sampling
    #(Min: 0.0, Max: 0.999999999)

    if (( [request_id_sampling] < <%= @sampling_rate %> ) and
( [sample_ok] != "no" )) {

      drop {}

    }

}
```

fitbit

Consistent Hashing

- Apply sparingly!
- Be clear about what services and logging paths will be sampled
- Ideally, store a field in the record with the sampling rate so a consumer of the logs knows what % was being dropped

Disaster Recovery

- The disaster recovery logging cluster drops 99% of log data
- Many fewer hosts (1 master, 4 data, 3 api)
- Otherwise identical to production
- If there's an disaster event, scale up and change sampling rate

fitbit

No dots for you!

```
"foo": "val1"

"foo": { "bar": "val2" }
```

- **`foo`** is a field *and* a top level object
- The logstash **`de_dot`** plugin is not efficient (minimum of 2x CPU use to have it enabled)
- es 2.4.0 has a setting to enable dots again

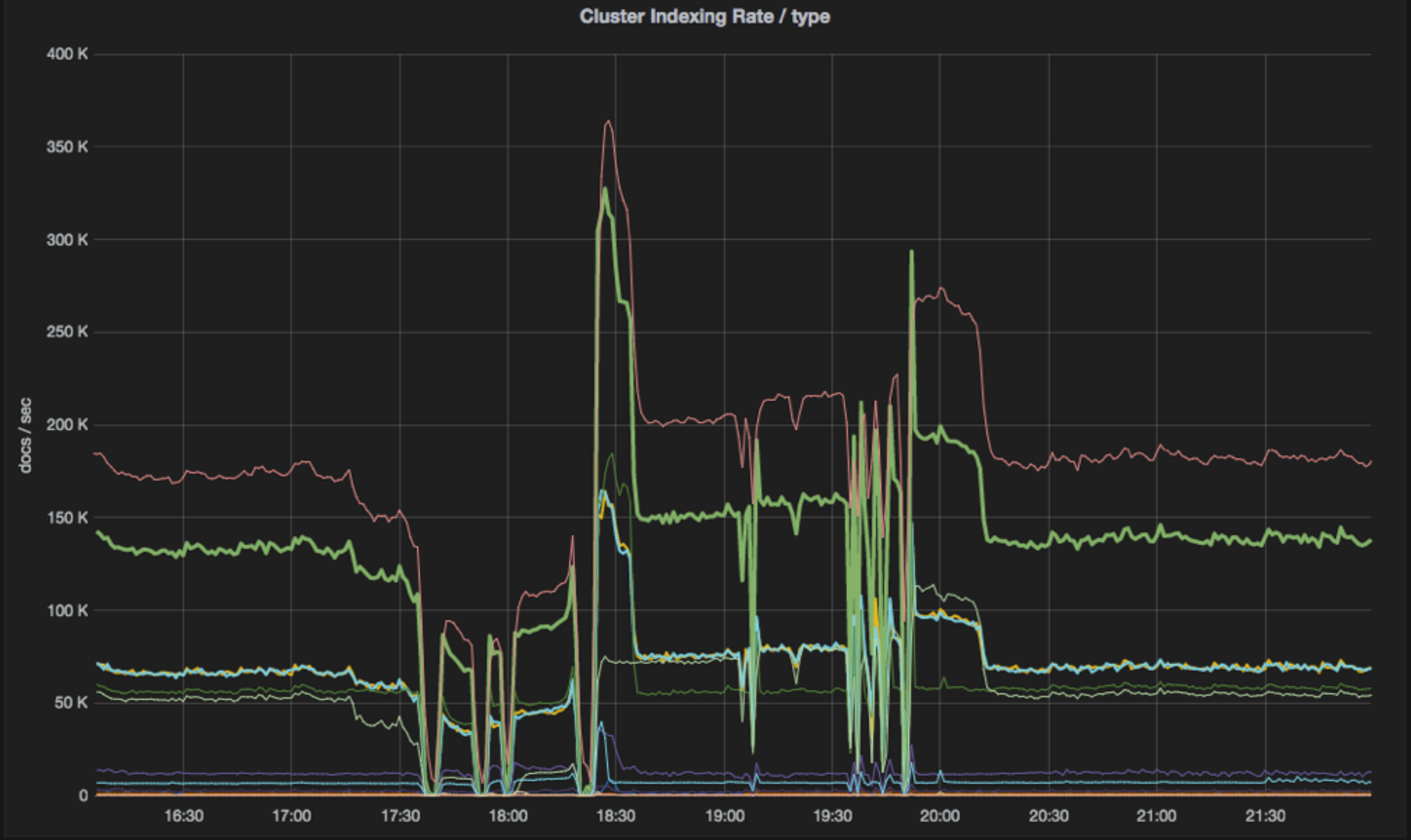
Hot/Warm Migrations

- Shard relocation in elasticsearch 2.x is considered an emergency action
- Moving 1,000 shards from data-rw to data-ro can block other cluster tasks, including mapping updates for new fields and index creation

MirrorMaker Issues

- Due to a number of Reasons, the new ELK cluster had to go in a different geographical region from the production application stack
- This meant two Kafka clusters, with MirrorMaker replicating topics between the two.
- Guess what we'd forgotten to set?

fitbit



42 LINES

fitbit

Bandwidth Delay Product

- Usual latency is 35ms
- When replication ground to a halt, latency was 120ms with spikes to 250ms

- Turning up the **`socket.buffer.bytes`** on the mirrormaker instances in the remote datacenter solved the issue

- Anything under 160ms is now transparent to replication

42 LINES

fitbit

*stormy

- elasticsearch 2.x assumes SSDs
- Queries with leading wildcards cause elasticsearch to read every field that may contain a match
- Yes. *Every* field. Of *every* possible match
- With SSDs and a small dataset, this causes a >10x performance penalty for a search

fitbit

*stormy

- With spinning disks and 300+ billion documents in the cluster it takes *hours* to read every document
- **`indices.query.query_string.allowLeadingWildcard: false`**
- And then restart every node in the cluster

Other Important Knobs

- `indices.query.query_string.allowLeadingWildcard: false`
- `cluster.routing.allocation.balance.threshold: 1.1f`
- `indices.memory.index_buffer_size: 50%`
- `bootstrap.mlockall: true`

fitbit

Questions?

- @bwdezend
- breandan@42lines.net
- http://github.com/bwdezend/elk-scaling-talk