ElasticSearch How-Tos.

Below are a number of useful links/tools/tips to instantly make yourself smarter, more beautiful and a hit at parties. We’ll first go slide-by-slide and then include all the stuff we couldn’t fit.

NOTE: These were true as of ES 5.6. I promise I’ll update for 6.0 once I update.

[Slide 2](#_o0zkmvkmoowz)

[Slide 11](#_cbgdspozb5j)

[Slide 12](#_8ka6x4k0lyke)

[Slide 15](#_e1r26opyfmp)

[Slide 16](#_wvrucrmvruwj)

[Slide 17](#_l38k33zgjchq)

[Slide 18](#_w6w9jlon447i)

[Slide 19](#_dvcor77tb7f6)

[Slide 20](#_j99497trhnuv)

[Slide 21](#_zi0hdptu5ffm)

[Slide 22](#_aaw10v2lax6t)

[Slide 23:](#_kzu68flzqd0r)

[Slide 24:](#_matolue6tx0a)

[Slide 30:](#_8pzx1ivc4dwi)

[BC Template](#_kr72ly6bt6oe)

[Common ElasticSearch Tasks](#_5s5hk4dxhuqy)

[Delete all Replica Copies](#_fwyjp6qgr19)

[Force-Sync all your indices](#_rcy9ksqcucx9)

[Inspect why shards are not recovering](#_gg0vb87y668v)

[Restarting and Recovering Quicker](#_8ixlt6bilxoz)

### Slide 2

Free Like a Puppy: Using Big Data Tools to Mine Security Data

<https://events.educause.edu/special-topic-events/security-professionals-conference/2016/proceedings/free-like-a-puppy-using-big-data-tools-to-mine-security-data>

### Slide 11:

TC Malloc Information From Seth Hall, Lead Developer and Founder of Corelight, a turnkey Bro box. Corelight should be on your short-list of providers if you want to simplify a Bro deployment.

<https://corelight.blog/2017/08/16/whats-the-riskiest-part-of-your-bro-deployment-it-may-be-you/>

Here’s a good whitepaper on really big Bro installs. Just another data point for folks to scale Bro better.

<https://www.cspi.com/wp-content/uploads/2016/09/Berkeley-100GIntrusionDetection.pdf>

Here’s some good Myricom adapters. Make sure to buy cards labeled Sniffer10G v3! Other driver sets will not work. They make bigger cards with other features than these if you need more, More, MORE!

<https://www.cspi.com/ethernet-products/adapters/c-class/>

If you use the Bro-Myricom plugin, instead of the libpcap functions within the Myricom, you are skipping the pcap abstraction layer. You can still filter using the -f switch in Bro, but the performance is slightly better.

### Slide 12:

To get the Tufts’ bro to talk to kafka easily, i added the logs-to-kafka plugin config and this to local.bro:

### JSON LOGGING

@load tuning/json-logs

# Set the log separator

redef Log::default\_scope\_sep = "\_";

# Set the time in iso format

redef LogAscii::json\_timestamps = JSON::TS\_ISO8601;

When you get to ES6.0+ the dots in bro logs will cause trouble. We changed them to \_.

### Slide 15:

Kafka:

<https://kafka.apache.org/>

How to Configure Bro to Send to Kafka:

<https://www.bro.org/sphinx/components/bro-plugins/kafka/README.html>

In the Logs-to-kafka.bro file, I made a change to add in the policy id as an attribute… here is my logs-to-kafka.bro file. There are commented out bits if you want to enumerate all the policies you want to send, but by default, I put them all in.:

---

export {

# redefine this in your script to identify the logs

# that should be sent up to bro.

# for example:

#

# redef KafkaLogger::logs\_to\_send = set(HTTP::LOG, Conn::LOG, DNS::LOG);

#

# that will send the HTTP, Conn, and DNS logs up to Kafka.

#

const logs\_to\_send = set() &redef;

# const logs\_to\_send: set[Log::ID] &redef;

}

event bro\_init() &priority=-5

{

for (stream\_id in Log::active\_streams)

{

# if (stream\_id in Kafka::logs\_to\_send)

# {

local filter: Log::Filter = [

$name = fmt("kafka-%s", stream\_id),

$writer = Log::WRITER\_KAFKAWRITER,

$config = table(["stream\_id"] = fmt("%s", stream\_id))

];

Log::add\_filter(stream\_id, filter);

# }

}

}

---

Syslog-ng to Kafka:

WARNING: This \_will not\_ work for medium-to-large Bro installs, but it works great for random Syslog data without a terribly heavy volume. Certainly worth playing around with, and very handy to keep in your mental ‘toolbox’.

<https://syslog-ng.com/documents/html/syslog-ng-ose-latest-guides/en/syslog-ng-ose-guide-admin/html/configuring-destinations-kafka.html>

### Slide 16:

This is the first of a number of Grafana graphs.

Grafana is a really great tool to *display* metrics.

To *collect and store* metrics, I use graphite, because its straightforward, flexible and easy.

To *fetch and gather* random metrics I use python or shell scripts… or occasionally a tool like diamond. I’ve included some of them below.

<https://grafana.com/>

<https://graphiteapp.org/>

<https://github.com/python-diamond>

This particular graph is made up of the following elements.

Kafka Lag:

This is the total number of records across all the partitions of a Kafka queue which have not been picked up by logstash. It assumes you are running this host on the same system as the kafka queue (127.0.0.1) and that you are using logstash (default group name is ‘logstash). This script throws a warning, but I ignore it.

/opt/kafka/bin/kafka-consumer-offset-checker.sh --zookeeper=127.0.0.1 --topic <your\_kafka\_topic> --group logstash

Basically, I total up the ‘Lag’ column with a short python script.

Bro->Kafka:

This is actually part of the ‘diamond’ process running on my Bro box. On it, I have the ‘Kafka’ collector enabled which collects how many entries were pushed into Kafka by queue.

The entry is stored in graphite as:

<prefix>.kafka.BrokerTopicMetrics.MessagesInPerSec.OneMinuteRate

Logstash Process Rate:

This is part of the logstash ingestion process. While processing logs, logstash can optionally gather metrics and then output them to graphite. Below are some snippets of the logstash filter.

<https://www.elastic.co/guide/en/logstash/current/plugins-filters-metrics.html>

Filter {

#I have one of these stanzas inside each ‘if’ statement in my logstash config so I can

#keep metrics with different names for different processed metrics

metrics {

meter => "events"

add\_tag => "brometric"

}

}

Output {

graphite {

host => "<logging host>"

port => 2003

metrics\_format => "logstash.bro.\*"

include\_metrics => [ "events.\*" ]

fields\_are\_metrics => true

}

}

### Slide 17:

Kafka Manager:

<https://github.com/yahoo/kafka-manager>

### Slide 18:

Cluster State Metric

This gathers the current state by character size. This is \_not\_ a literal expression of the true size, but a representative one. The more characters, the bigger it is. I pass this value into graphite using a cron script which gathers the number every 5 minutes. This only really expresses the total number of shards, see slide 19.

/usr/bin/curl -s http://<es-node-name>:9200/\_cluster/state | wc -c"

## Slide 19:

How many shards do I have?

curl <http://localhost:9200/_cluster/health?pretty>

In the output of this command you’ll find everything from your cluster status to how many shards are assigned or are moving around your cluster. These are important to know when understanding how your cluster is behaving. The number of “active\_shards” is what is present in the cluster, and stored in the heap.

NOTE: Adding ?pretty or ?pretty=true to the end of the URL will display the JSON in a nice tabular format. Its useful on almost every command to display data in a nice orderly way for a human to read. Its unnecessary for machine-processing, as it just adds extra spaces and newlines.

## Slide 20

On this slide, we touch upon Template configurations, but hold that thought… we’ll show the entire template further down

## Slide 21:

<https://www.elastic.co/blog/a-heap-of-trouble>

Seriously, 26G. You don’t \_need\_ to do this, but there is a performance difference. To see if you have Compressed Oops enabled \_at all\_ (Its Good).

curl -XGET 'http://localhost:9200/\_nodes/\_all/jvm?pretty'

And look for a line **for each node** like:

"using\_compressed\_ordinary\_object\_pointers" : "true"

## Slide 22:

There are tons of decent scripts on facebook for pulling Elasticsearch details and put them in Graphite. Node stats are pulled from the script in this github instance. Why this one? It fit the bill and its coded well. I could run it without the docker instance easily enough by just downloading the raw file.

<https://github.com/logzio/logzio-es2graphite>

https://raw.githubusercontent.com/logzio/logzio-es2graphite/master/logzio-es2graphite/scripts/go.py

## Slide 23:

Mappings! See Template definition below.

## Slide 24:

Clearing the search field cache

curl -XPOST localhost.edu:9200/\_cache/clear

## Slide 30:

Logstash config! This is based on my Bro event stream modifications above. Why is it like this? Because it works for me, its compatible with Bro’s kafka plugin, and I never went back to change it. You might have better luck with bro json output, or something else. Here’s how it works...The ‘metrics’ stanza helps make stats for graphite. The grok stanza takes the outer-most brackets, which have the topicID and pulls it off, then makes a new ‘message’ with two fields ‘topicID’ and ‘newmessage’. The ‘mutate’ stanza changes the ‘id’ field and the ‘version’ field in the ‘newmessage’ value to preserve them so they don’t get overwritten when Elasticsearch ingests the entry. The ‘json’ stanza processes the rest of the ‘newmessage’ string as JSON. The ‘data’ stanza processes the epoch value stored in ‘ts’ and stores it as a true date field called ‘broTs’. If you don’t do this, it will be stored as a string and not useable by ElasticSearch as a date, and then overwritten when you ingest the log. The last ‘mutate’ stanza removes all the extra fields before its pushed into Elasticsearch so we don’t index extra pieces.

if [type] == "bro-perimeter" {

metrics {

meter => "events"

add\_tag => "metric"

}

grok {

match => { "message" => "\{\"%{WORD:topicID}\": %{GREEDYDATA:newmessage}\}" }

}

mutate {

gsub => [ "newmessage", "\"id\"", '"broId"']

gsub => [ "newmessage", "\"version\"", '"sslVersion"' ]

}

json {

source => "newmessage"

}

date {

match => [ "ts", "UNIX" ]

target => "broTs"

}

mutate {

remove\_field => ["message","newmessage","ts"]

}

}

}

## BC Template

Note, we use logstash to put data into ES, which uses its own template by default. To see what your logstash template is:

curl -XGET <http://localhost:9200/_template/logstash?pretty>

If you call your template ‘logstash’ you can have a namespace collision, so just be careful.

You can add a template below by using the command The name of the template will be the outer-most stanza.

curl -XPUT <http://localhost:9200/_template/> -d ‘

<insert spec here>

’

Note: You may need to set your ‘type’ values in the mapping to match the types you set in Logstash. I have two types. ‘Bro-perimeter’ is for my Bro instance on the perimeter of my network.

{

"template\_1" : {

"order" : 0,

"template" : "bro\*",

"settings" : {

"index" : {

"refresh\_interval" : "15s",

"number\_of\_shards" : "20",

"translog" : {

"flush\_threshold\_size" : "512mb",

"durability" : "async"

},

"number\_of\_replicas" : "1",

"merge" : {

"scheduler" : {

"max\_thread\_count" : "2"

}

}

}

},

"mappings" : {

"bro-perimeter" : {

"properties" : {

"broId" : {

"type" : "text",

"doc\_values" : false

},

"uid" : {

"type" : "text",

"doc\_values" : false

},

"client\_cert\_chain\_fuids" : {

"type" : "text",

"doc\_values" : false

},

"cert\_chain\_fuids" : {

"type" : "text",

"doc\_values" : false

},

"fuid" : {

"type" : "text",

"doc\_values" : false

},

"conn\_uids" : {

"type" : "text",

"doc\_values" : false

},

"certificate.serial" : {

"type" : "text",

"doc\_values" : false

},

"resp\_fuids" : {

"type" : "text",

"doc\_values" : false

},

"orig\_fuids" : {

"type" : "text",

"doc\_values" : false

},

"parent\_fuid" : {

"type" : "text",

"doc\_values" : false

},

"fuids" : {

"type" : "text",

"doc\_values" : false

}

}

}

},

"aliases" : { }

}

What is all this?

For all indices named ‘bro\*’....

Lower the refresh\_interval to 15 seconds….

Set the number of shards to 20 (remember, less indices, reasonable shard counts)…

Adjust the size of the translog and don’t write to disk with the fsyncs of the OS…

Set the replicas to 1 (to reduce the number of indices)

Set the max\_thread\_count (of writes) to 2. (You can use more if you have SSDs, but I use spinning disks, so there’s no point to setting this number high)

Then comes the big part, the mappings. There might be MORE mappings which have a high cardinality for which partial sorts are useless, but these were the ones we started with. NOTE: This will only work with **new** indices you would create… old indices will need to be ‘reindexed’ but that’s beyond the scope of this.

## Common ElasticSearch Tasks

**Disable Cluster Routing (Very useful when you need to restart nodes, see procedure below)**

curl -XPUT http://localhost:9200/\_cluster/settings -d

'{

"transient": {

"cluster.routing.allocation.enable": "none"

}

}'

**Enable Cluster Routing (**and sets the number of recovered nodes to 10, so it processes faster**)**

curl -XPUT locahost:9200/\_cluster/settings -d '

{ "transient" : {

"cluster" : {

"routing" : {

"allocation" : {

"node\_concurrent\_recoveries" : "10",

"enable" : "all"

}

}

}

}

}'

## **Delete all *Replica* Copies**

(useful if you have to recover lots of shards. It halves the number of copies and eliminates the possibility of checksums which aren’t equivalent. You will be without backup copies until you change this setting again. Make sure when you change this setting to turn it back on you match how many replicas set by your template)

curl -XPUT http://localhost:9200/bro\*/\_settings -d '{ "index" : { "number\_of\_replicas" : 0 } }'

## Force-Sync all your indices

(useful if you need to restart nodes or perform rolling upgrades. Doing this takes a few minutes, but can save hours of time. DO THIS AFTER YOU DISABLE CLUSTER ROUTING. See below.)

Curl -XPOST <http://localhost:9200/_flush/synced>

## Inspect why shards are not recovering

Curl <http://localhost:9200/_cluster/allocation/explain>

## Restarting and Recovering Quicker

When you need to restart a node, your cluster may show as RED for a...long, long, time. To get the cluster to recover quicker, we’ve been running this combination:

1. First, disable shard reallocation. Otherwise when ES sees a node go away it tries to rebalance itself and then when the node comes back it rebalances itself again.

curl -XPUT -H 'Content-Type: application/json' [http://elastic1:9200/\_cluster/settings](http://elastic1.uit.tufts.edu:9200/_cluster/settings) -d '{ "transient": { "cluster.routing.allocation.enable": "none" } }'

1. Now, flush all the accumulated stuff in the caches.

curl -XPOST<http://elastic1:9200/_flush/synced>

1. Then restart the node(s) that needs it.

service elasticsearch restart

1. And re-enable shard allocation.

curl -XPUT -H 'Content-Type: application/json' [http://elastic1:9200/\_cluster/settings](http://elastic1.uit.tufts.edu:9200/_cluster/settings) -d '{ "transient": { "cluster.routing.allocation.enable": "all" } }'

This seems to also help in rolling upgrades.