Tuning the Search tier

> *If you are running a version of ElasticSearch that is different from what came with Apache Metron 0.4.2 and older, the below information is outdated. As of METRON-939, ElasticSearch and Kibana have been upgraded and the below guide may no longer be accurate.*

Sizing your hardware

Elasticsearch prefers to run on a large number of small servers, as opposed to multiple instances on big servers. See this great post for more details.

If you are going to run multiple instance on a single server, be sure to provided dedicated disks to every instance.

Elasticsearch

On installation

1. Make sure to update your ES templates to properly index the fields of what you're sending in, especially if your logs have any custom fields. Note that `bro_index.template` only currently handles DNS and HTTP Bro logs. In the future I may be able to contribute back a template that handles all of the standard Bro logs, but I just haven't gotten a chance to push that out yet.
 - File(s): `incubator-metron/metron-deployment/roles/metron_elasticsearch_templates/files/es_templates/*.template`
2. (optional) Alter or disable the cron jobs to purge ES indices (and HDFS sensor data) every 30 days as necessary.
 - File(s): `incubator-metron/metron-deployment/roles/metron_streaming/tasks/{es_purge.yml,hdfs_purge.yml}`
3. Assuming that ES will get behind from time to time, you may want to increase the indexing kafka topic size. You should also consider increasing the partitioning kafka topics in general to distribute the load better and increase parallelism.
 - File(s): `incubator-metron/metron-deployment/roles/metron_kafka_topics/defaults/main.yml`
4. Adjust the batch size of your indexes to line up with Elastic's recommendations of 5MB-15MB per batch. Note that this is done via the indexing topology's indexingBolt, but it uses the enrichment topology's zookeeper config (batchSize). See the below JIRA to split this out.
 - File(s): `incubator-metron/metron-platform/metron-enrichment/src/main/config/zookeeper/enrichments/*.json`
 - Relevant JIRA: METRON-470

Post-installation but before data ingest

1. (optional) Consider multiple ES nodes per physical server. Lots of guides online about this, but tread carefully. My default would be to not pursue this unless you have large(ish) machines running as dedicated search nodes. As far as I'm aware, Metron has no built-in way to provision this, and there may be some downstream .
2. Tune `/etc/elasticsearch/elasticsearch.yml` - all configs below are optional and should be individually evaluated for your scenario.
 - Set `indices.store.throttle.type: none` to prioritize indexing above searching.
 - Set `discovery.zen.fd.ping_timeout: 300s` to timeout after 300s instead of the default of 30s. Busy clusters sometimes take time > 30s to respond.
 - Set `bootstrap.mlockall: true` to prevent Elasticsearch memory from being swapped.
 - Set `indices.fielddata.cache.size: 5%` to evict caches for infrequently used indexes more frequently. You may want to play around with setting this number between 5% and 20%.
 - Set `indices.breaker.fielddata.limit: 50%` which prevents the size of fielddata from surpassing 50% of the heap - important to prevent large queries from knocking over a cluster. Note that this must be higher than your `indices.fielddata.cache.size` setting.
 - Set `indices.store.throttle.type: none` to prevent merges, which is helpful on systems bound by disk IO, as merge operations can be a killer. You could also look at setting `indices.store.throttle.type: merge` and then configuring `indices.store.throttle.max_bytes_per_sec` to throttle the merge activity.
3. Tune the elasticsearch service parameters.
 - Set your ES_HEAP_SIZE properly - long story short you want to make sure that this value is set to the highest value that uses compressed Oops, **with a maximum of 1/2 your total system RAM**. This can be done by editing `/etc/init.d/elasticsearch`. In the below example, you would want to set it to `32766` because it is <= .5 * 129013, and still uses compressed Oops.

```
# free -m | grep Mem: | awk '{print $2}'
129013
# java -Xmx32766m -XX:+PrintFlagsFinal 2> /dev/null | grep UseCompressedOops

bool UseCompressedOops := true {lp64_product}
# java -Xmx32767m -XX:+PrintFlagsFinal 2> /dev/null | grep UseCompressedOops
bool UseCompressedOops = false {lp64_product}

```

 - Other params to look into include `MAX_OPEN_FILES`, `MAX_LOCKED_MEMORY`, and `MAX_MAP_COUNT`, among others.
4. Tune your system (information from here).

`/etc/sysctl.conf`
 - Set `vm.swappiness=1` to turn off swapping.
 - Set `net.core.somaxconn=65535` to increase the number of connections per port.
 - Set `vm.max_map_count=262144` to increase the number of memory map areas a process may have.
 - Set `fs.file-max=518144` to increase the number of file-handles that the Linux kernel will allocate.

`/etc/security/limits.conf`
 - Set `elasticsearch - nofile 65535` to increase the max number of open file descriptors, assuming the user for elasticsearch is "elasticsearch".
 - Set `elasticsearch - memlock unlimited` to increase the max locked-in-memory address space, assuming the user for elasticsearch is "elasticsearch".

`/etc/pam.d/system-auth`

- Ensure `session required pam_limits.so` exists to set limits on the system resources that can be obtained in a user-session.

Final notes
The tweaks above are meant to be persistent but not necessarily modify the current state of a system. That said, I'd recommend a reboot then verify to make sure it is all configured correctly, as opposed to injecting the changes.
If your ES ingest is slow, look upstream at tuning storm and kafka. Probably not relevant to this thread, so leaving my notes for that out. Happy to discuss further if it would be helpful to anyone else.

SOLR

TODO - However, most of the Elasticsearch tuning applies because they both use Lucene underneath.