Elasticsearch tuning

This guide summarizes the relevant configurations that allow us to optimize Elasticsearch.

Memory locking

Elasticsearch performs poorly when the system is swapping the memory. It is vitally important to the health of your node that none of the JVM is ever swapped out to disk.

We will set the *bootstrap.memory_lock* setting to true, so Elasticsearch will lock the process address space into RAM, preventing any Elasticsearch memory from being swapped out.

Step 1: Set bootstrap.memory_lock

Uncomment or add this line to the `/etc/elasticsearch/elasticsearch.yml` file:

```
bootstrap.memory_lock: true
```

Step 2: Edit limit of system resources

Where to configure systems settings depends on which package and operating system you choose to use for Elasticsearch installation.

- In a case where **systemd** is used, system limits need to be specified via systemd. First, create the folder executing the command: `mkdir -p /etc/systemd/system/elasticsearch.service.d/`, add a file called `elasticsearch.conf` and specify any changes in that file:

```
[Service]
LimitMEMLOCK=infinity
```

- In other case, edit the proper file `/etc/sysconfig/elasticsearch` for RPM or `/etc/default/elasticsearch` for Debian:

```
MAX_LOCKED_MEMORY=unlimited
```

Step 3: Limit memory

The previous configuration might cause node instability or even node death (with an *OutOfMemory* exception) if Elasticsearch tries to allocate more memory than is available. JVM heap limits will help us to define the memory usage and prevent this situation.

There are two rules to apply when setting the Elasticsearch heap size:

- No more than 50% of available RAM.
- No more than 32 GB.

In addition, you must take into account the memory usage by the operating system, services and software running on the host.

By default, Elasticsearch is configured with a 1 GB heap. You can change the heap size via JVM flags using the `/etc/elasticsearch/jvm.options` file:

```
# Xms represents the initial size of total heap space
# Xmx represents the maximum size of total heap space

-Xms4g
-Xmx4g
```

> **Warning**
>
> Ensure that the min (Xms) and max (Xmx) sizes are the same, this prevents JVM heap resizing at runtime, a very costly process.

Step 4: Restart Elasticsearch

Finally, restart Elasticsearch service:

a. For Systemd:

```
systemctl daemon-reload
systemctl restart elasticsearch
```

b. For SysV Init:

```
service elasticsearch restart
```

After starting Elasticsearch, you can see whether this setting was successfully applied by checking the value of `mlockall` in the output of the next request:

```
curl -XGET 'localhost:9200/_nodes?filter_path=**.mlockall&pretty'
```

```
{
  "nodes" : {
    "sRuGbIQRRfC54wzwIHjJWQ" : {
      "process" : {
        "mlockall" : true
      }
    }
  }
}
```

The request has failed when you see the above output have `"mlockall" : false` field. You will also see a line with more information in the logs (*/var/log/elasticsearch/elasticsearch.log*) with the words *Unable to lock JVM Memory*.

Reference:

- Memory lock check.
- bootstrap.memory_lock.
- Enable bootstrap.memory_lock.
- Heap: Sizing and Swapping.
- Limiting memory usage.

Shards and replicas

Elasticsearch provides the ability to split an index in multiple pieces called shards. Each shard is in itself a fully-functional and independent “index” that can be hosted on any node in the cluster. Sharding is important for two primary reasons:

- It allows you to horizontally split/scale your content volume.
- It allows you to distribute and parallelize operations across shards thus increasing performance/throughput.

Also, Elasticsearch allows you to make one or more copies of your index’s shards into what are called replica shards, or replicas for short. Replication is important for two primary reasons:

- It provides high availability in case a shard/node fails.
- It allows you to scale out your search volume/throughput since searches can be executed on all replicas in parallel.

> **⚠ Warning**
>
> The number of shards and replicas can be defined per index at the time the index is created. After the index is created, you may change the number of replicas dynamically anytime but you cannot change the number of shards after-the-fact.

How many shards should my index have?

Due to is not possible to *reshard* (changing the number of shards) without reindexing, you must answer to this question before create your first index. However, we can't decide how many shards use without talk about nodes. In general, you will get the optimal performance by using the same number of shards as nodes. So, a cluster with 3 nodes will have 3 shards while a cluster with one node will only have a shard.

How many replicas should my index have?

Let's consider what will happen in a cluster with 3 nodes and 3 shards:

- No replica: Each node has 1 shard. If a node falls down, we will have a incomplete index of 2 shards.
- 1 replica: Each node has 1 shard and 1 replica. If a node falls down, we will have a complete index.
- 2 replicas: Each node has 1 shard an 2 replicas (the full index). Now, the cluster can work with just one node. This looks like the best solution but also it increase the storage requirements.

Setting number of shards and replicas

The default installation of Elastic Stack with RPM or Debian packages will configure each index with 5 primary shards and 1 replica.

In case you want to change these settings you need to edit the Elasticsearch template. In the following example, we configure the proper values for shards and replicas in a cluster with only 1 node.

> **⚠ Warning**
>
> We assume that your index has not yet been created, otherwise you will have to reindex after editing the template.

1. Download the Wazuh Elasticsearch template:

```
curl https://raw.githubusercontent.com/wazuh/wazuh/2.1/extensions/elasticsearch/wazuh-elastic5-
template.json -o w-elastic-template.json
```

2. Edit the template in order to set 1 shard a 0 replicas:

```
nano w-elastic-template.json

{
  "order": 0,
  "template": "wazuh*",
  "settings": {
    "index.refresh_interval": "5s",
    "number_of_shards" :   1,
    "number_of_replicas" : 0
  },
  "mappings": {
  "...": "..."
  }
}
```

3. Load the template:

```
curl -XPUT 'http://localhost:9200/_template/wazuh' -H 'Content-Type: application/json' -d @w-elastic-
template.json
```

Changing number of replicas

The number of replicas can be changed dynamically using the Elasticsearch API.

In a cluster with 1 node, the number of replicas should be 0:

```
curl -XPUT 'localhost:9200/wazuh-*/_settings?pretty' -H 'Content-Type: application/json' -d'
{
    "settings": {
        "number_of_replicas" : 0
    }
}
'
```

Reference:

- Shards & Replicas.