HCP Upgrade Guide 1

Upgrading Elasticsearch

Date of Publish: 2018-12-21

https://docs.hortonworks.com/

Contents

Upgrading Elasticsearch for HCP

Hortonworks Cybersecurity Platform has upgraded its support for Elasticsearch. When you upgrade to HCP 1.7.0 or later, you will need to make certain changes to your ES templates and mappings.

Upgrading Elasticsearch Alert Field

Beginning with HCP 1.7.0, the Elasticsearch metaalert alert nested field has been changed to metron_alert. Due to this change, HCP 1.7.0 and later is unable to use indices containing the alert field.

You must adjust your templates and mappings to reflect the new field name metron_alert, then create new indices with the new template and mapping, and migrate existing data to the new indices.

Changes to Supported Elasticsearch Properties Resulting from Migrating From Transport to REST Client

Hortonworks Cybersecurity Platform (HCP) has migrated the Elasticsearch Java client from the Transport client to the new Java REST client. The motivation for this change is that TransportClient will be deprecated in Elasticsearch 7.0 and removed entirely in 8.0. This migration changes some of the supported properties. You will need to modify some of your Elasticsearch properties to support the new REST client.

The following table provides a mapping of the old Elasticsearch properties to the new supported properties.

Old Property	New Property
client.transport.ping_timeout	N/A
N/A	connection.timeout.millis
N/A	socket.timeout.millis
N/A	max-retry.timeout.millis
N/A	num.cient.connect.threads
es.client.class	N/A
es.xpack.username	xpack.username
es.xpack.password.file	xpack.password.file
xpack.security.transport.ssl.enabled	ssl.enabled
xpack.ssl.key	N/A
xpack.ssl.certificate	N/A
xpack.ssl.certificate_authorities	N/A
N/A	keystore.type
keystore.path	keystore.path
N/A	keystore.password.file

Property Name	Type	Required?	Default Value	Description
connection.timeout.millis	Integer	No	1000	Sets connection timeout.
socket.timeout.millis	Integer	No	30000	Sets socket timeout.

Property Name	Type	Required?	Default Value	Description
max.retry.timeout.millis	Integer	No	30000	Sets the maximum timeout (in milliseconds) to honour in case of multiple retries of the same request.
num.client.connection.threads	Integer	No	1	Number of worker threads used by the connection manager. Defaults to Runtime.getRuntime().availableProcessors().
xpack.username	String	No	null	X-Pack username.
xpack.password.file	String	No	null	1-line HDFS file where the X-Pack password is set.
ssl.enabled	Boolean	No	false	Turn on SSL connections.
keystore.type	String	No	"jks"	Allows you to specify a keytstore type. See https://docs.oracle.com/javase/8/docs/technotes/guides/security/StandardNames.html#KeyStore for more details.
keystore.path	String	No	null	Path to the Trust Store that holds your Elasticsearch certificate authorities and certificate.
keystore.password.file	String	No	null	1-line HDFS file where the keystore password is set.

Upgrading to Elasticsearch 5.6.2

There are a number of template changes in Elasticsearch 5.6.2, most notably around string type handling, that may cause issues when upgrading.

For Elasticsearch 5.x, the existing indexes and templates need to upgraded. For more information, see:

- Updating Elasticsearch Templates to Work with Elasticsearch 5.x
- Updating Existing Indexes to Work with Elasticsearch 5.x

If you are upgrading from Elasticsearch 2.x to Elasticsearch 5.6.2, you will need to re-index.

Type Mapping Changes

Type mappings in Elasticsearch 5.6.2 have changed from ES 2.x. This section provides an overview of the most significant changes.

The following is a list of the major changes in Elasticsearch 5.6.2:

- String fields replaced by text/keyword type
- Strings have new default mappings as follows:

```
{
  "type": "text",
  "fields": {
    "keyword": {
      "type": "keyword",
      "ignore_above": 256
    }
```

```
    }
}
```

- There is no longer a _timestamp field that you can set "enabled" on.

  This field now causes an exception on templates. The Metron model has a timestamp field that is sufficient.

The semantics for string types have changed. In 2.x, index settings are either "analyzed" or "not_analyzed" which means "full text" and "keyword", respectively. Analyzed text means the indexer will split the text using a text analyzer, thus allowing you to search on substrings within the original text. "New York" is split and indexed as two buckets, "New" and "York", so you can search or query for aggregate counts for those terms independently and match against the individual terms "New" or "York." "Keyword" means that the original text will not be split/analyzed during indexing and instead treated as a whole unit. For example, "New" or "York" will not match in searches against the document containing "New York", but searching on "New York" as the full city name will match. In Elasticsearch 5.6 language, instead of using the "index" setting, you now set the "type" to either "text" for full text, or "keyword" for keywords.

Below is a table listing the changes to how String types are now handled.

sort, aggregate, or access values	Elasticsearch 2.x	Elasticsearch 5.x	Example
no	`"my_property" : { "type": "string", "index": "analyzed" }`	`"my_property" : { "type": "text" }` Additional defaults: "index": "true", "fielddata": "false"	"New York" handled via in-mem search as "New" and "York" buckets. No aggregation or sort.
yes	`"my_property": { "type": "string", "index": "analyzed" }`	`"my_property": { "type": "text", "fielddata": "true" }`	"New York" handled via in-mem search as "New" and "York" buckets. Can aggregate and sort.
yes	`"my_property": { "type": "string", "index": "not_analyzed" }`	`"my_property" : { "type": "keyword" }`	"New York" searchable as single value. Can aggregate and sort. A search for "New" or "York" will not match against the whole value.
yes	`"my_property": { "type": "string", "index": "analyzed" }`	`"my_property": { "type": "text", "fields": { "keyword": { "type": "keyword", "ignore_above": 256 } } }`	"New York" searchable as single value or as text document, can aggregate and sort on the sub term "keyword."

If you want to set default string behavior for all strings for a given index and type, you can do so with a mapping similar to the following (replace ${your_type_here} accordingly):

```
# curl -XPUT 'http://${ES_HOST}:${ES_PORT}/_template/
default_string_template' -d '
{
    "template": "*",
    "mappings" : {
        "${your_type_here}": {
            "dynamic_templates": [
                {
                    "strings": {
                        "match_mapping_type": "string",
                        "mapping": {
                            "type": "text"
                            "fielddata": "true"
                        }
                    }
                }
            ]
        }
    }
}
```

By specifying the template property with value *, the template will apply to all indexes that have documents indexed of the specified type (${your_type_here}).

The following are other settings for types in ES:

- doc_values
  - On-disk data structure
  - Provides access for sorting, aggregation, and field values
  - Stores same values as _source, but in column-oriented fashion better for sorting and aggregating
  - Not supported on text fields
  - Enabled by default
- fielddata
  - In-memory data structure
  - Provides access for sorting, aggregation, and field values
  - Primarily for text fields
  - Disabled by default because the heap space required can be large