Atlantis Search

Elastic Search

Elastic Search is the data store we use to efficiently index and search node related data.

Changing the Replica Count

> ElasticSearch Docs for this API can be found here

PUT {index-name-here}/_settings

```
{
  "index" : {
    "number_of_replicas" : 1
  }
}
```

Re-Indexing API

> ElasticSearch API Documentation can be found here

You can copy documents from one index to another by using the Reindex API.

This was useful during beta testing to create a new index with a different shard count and copy all the documents to the new index, without going through the Atlantis-Search indexing process all over again.

We could likely do this in the future when we want to scale to a larger tier and want to increase the shard count. We would need to temporarily scale to a tier that has enough storage to duplicate the index, but once we've reindexed, we could delete the original and scale back down to the desired tier.

Example

POST _reindex?requests_per_second=-1&wait_for_completion=false

```
{
 "source": {
  "index": "node-index"
 },
 "dest": {
  "index": "node-index-1",
  "routing": "keep"
 }
}
```

- `requests_per_second`: setting this to `-1` makes it so no throttling occurs
- `wait_for_completion`: setting this to `false` makes it so the operation runs asynchronously. (You want to do this)
- `routing`: setting this to `keep` makes it so the `_routing` values stays the same as in the source. We do use routing so you will want to make sure this is set.

Aggregations

> Official ElasticSearch Documentation can be found here

Example: Average Length of File Content

We had a case where we wanted to see what the average length of the ***fileContent*** property was. We were able to use aggregations to get statistics about the property including the ***count***, ***minimum length***, ***maximum length***, ***average length***, and ***entropy*** using the string_stats aggregation.

We used the API Console in Elastic Cloud to run the aggregation query.

> We used the _async_search API because the query took so long to run. During this time searches slowed to a crawl. We were running this at night so it wasn't a huge deal. If you have an async search that is running too long and / or causing performance issues you can stop the request by sending a DELETE call to _async_search/{async search id here}, and this is detailed further below.

Request

- Method: POST
- Path: node-index/_async_search?size=0
- Body (see below)

```
{
 "query": {
  "exists": {
   "field": "fileContent.wildcard"
  }
 },
 "aggs": {
  "message_stats": { "string_stats": { "field": "fileContent.wildcard" } }
```

```
  }
}
```

Response

Initial Response Body


```
{
  "is_partial": true,
  "is_running": true,
  "id": "FjZFS3lJTnd0UTNla29TcnFFRnR5NncfT0NycU9EcnFRYjZqRkdTRFQwdi1PZzo0NDY0MjA5NA==",
  "expiration_time_in_millis": 1716350043005,
  "response": {
    "hits": {
      "hits": [],
      "total": {
        "relation": "gte",
        "value": 0
      },
      "max_score": null
    },
    "_shards": {
      "successful": 0,
      "failed": 0,
      "skipped": 0,
      "total": 400
    },
    "took": 1031,
    "timed_out": false,
    "terminated_early": false,
    "num_reduce_phases": 0
  },
  "start_time_in_millis": 1715918043005
}
```


The request is running asynchronously, and you can monitor progress with the following request

- Method: GET
- Path: _async_search/FjZFS3lJTnd0UTNla29TcnFFRnR5NncfT0NycU9EcnFRYjZqRkdTRFQwdi1PZzo0NDY0MjA5NA==

```
{
  "is_partial": true,
  "is_running": true,
  "id": "FjZFS3lJTnd0UTNla29TcnFFRnR5NncfT0NycU9EcnFRYjZqRkdTRFQwdi1PZzo0NDY0MjA5NA==",
  "expiration_time_in_millis": 1716350043005,
  "response": {
    "hits": {
      "hits": [],
      "total": {
        "relation": "gte",
        "value": 10000
      },
      "max_score": null
    },
    "_shards": {
      "successful": 17,
      "failed": 0,
      "skipped": 0,
      "total": 400
    },
    "took": 215630,
    "timed_out": false,
    "terminated_early": false,
    "num_reduce_phases": 4,
    "aggregations": {
      "message_stats": {
        "count": 8880102,
        "min_length": 1,
        "max_length": 12175466,
        "entropy": 4.977113043941186,
        "avg_length": 12041.666635923777
      }
    }
  },
  "start_time_in_millis": 1715918043005
}
```


You can delete the search query (if it's running too long or causing other performance issues with the following request

- Method: DELETE
- Path: _async_search/FjZFS3lJTnd0UTNla29TcnFFRnR5NncfT0NycU9EcnFRYjZqRkdTRFQwdi1PZzo0NDY0MjA5NA==

Elastic Search

List Shard Sizes (and other details)

```
GET /_cat/shards?v&h=index,shard,prirep,state,store,node&s=store:desc
```

Force Merge

> Official ElasticSearch documentation can be found here

Additional Notes

We use the force merge API in the past, when we've had a very large account purged and needed to free up disk space more quickly than letting the cluster clean it up on its own in the background and have seen disk space free up by 10% or more.

We have only done this on one index at a time in the past.

There is an optional parameter called `only_expunge_deletes` that basically, when set to true, indicates that you only want to free up disk space of deleted documents. If this is set to true, it seems that you don't need to worry about turning off writes to the index while it is running.

If you do not set `only_expunge_deletes` to true, it is recommended that you turn off writes to the index well the force merge task runs.

This could change over time, but at the time of writing, you can do this by turning off / disabling the following function apps / specific functions

- Turn off ***AtlantisSearch-Indexing*** function app
- Disable ***TikaExtractComplete*** function in ***revver-textExtractionProcessing*** function app
- Disable ***OCRExtractComplete*** function in ***revver-ocrProcessing*** function app
- Disable ***queue triggered functions in the AtlantisFileProcessing-ExtractedTextManagment*** function app (this is probably overkill, but I have not researched exactly with functions write to the index)

Search Shard Routing

> Official ElasticSearch documention can be found here

Additional Notes

If the way we are storing documents per account starts leading the shards that are way too big, we could manage what shards different accounts are stored on in our own mapping table and specify in our elastic search requests to store and read data from a specific shard by setting preference to `preference=_shards:{shard number here}`.

Fix Watermark Errors (Flood-Stage)

https://www.elastic.co/guide/en/elasticsearch/reference/current/fix-watermark-errors.html

This occurs when a node disk size has exceeded 95% (or whatever you've set the threshold to)

LEGACY: ElasticSearch Backup/Restore from S3

Steps to restore an ElasticSearch Snapshot from an S3 backup:

*make sure same version for less issues

1. Install S3 plugin
2. allow insecure settings (on local ElasticSearch instance)
   a. open ProgramData -> Elastic -> ElasticSearch -> config -> jvm.options
   b. edit the file and add the line "-Des.allow_insecure_settings=true" to the bottom.
   c. restart the ElasticSearch service.
3. Register the S3 bucket as a snapshot repository

   a. Do a PUT to "http://localhost:9200/_snapshot/s3_repository " with a body: {
   "type": "s3",
   "settings": {
   "bucket": "elasticsearch-monitor",
   "region": "us-east-1",
   "access_key": "{access_key}",
   "secret_key": "{key_secret}"
   }
   }
4. Create the snapshot
   a. Do a POST "
   http://localhost:9200/_snapshot/s3_repository/{snapshot_name}?wait_for_completion=true " with a body:
   {
   "type": "s3",
   "settings": {
   "bucket": "efc-elasticsearch-export",
   "region": "us-east-1",
   "access_key": "{access_key}",
   "secret_key": "{key_secret}"
   }

}

5. Register S3 repository with hosted ElasticSearch: Following this article or article2
   a. Once you have the role arn and S3 access key and secret, do a PUT request to the AWS ElasticSearch instance using Postman.
   b. Postman has the ability to do a "Signed Request" which is important for this PUT request to complete successfully. You do this by going to the Authorization tab, selecting Type: AWS Signature. Enter the access key, and secret, region (probably us-east-1) and Service Name (es).
   c. The URL and body are as follows:
   URL: https://{your url}.us-east-1.es.amazonaws.com/_snapshot/s3_repository?pretty&verify=false
   Body (JSON): {
   "type": "s3",
   "settings": {
   "bucket": "efc-elasticsearch-export",
   "role_arn": "arn:aws:iam::{your arn}:role/es-to-s3-repository-staging-role",
   "region": "us-east-1"
   }
   }
   d. Validate snapshot is available: GET "https://{yoururl}.us-east-1.es.amazonaws.com/_snapshot/s3_repository/_all"
   e. Restore the snapshot with a POST "https://{your_url}.us-east-1.es.amazonaws.com/_snapshot/s3_repository/{snapshot_name}/_restore"

LEGACY: Installing Elastic Search

> **We don't currently support running ElasticSearch locally w/ the Atlantis Search Solution. This article pertains to running ElasticSearch locally with our legacy search solution. We have kept the article around because we may support running elastic search locally in the future and this article would be a good starting point for doing that.**

Install Using Docker

Taken from Elastic Search with Plugins in Docker

1. Download Docker Desktop (Windows): Docker Desktop
2. Create a Dockerfile. This file can be saved anywhere you choose. There is many ways of doing this, but what needs to be done is to create a file that does not have an extension on it and call it Dockerfile. One way is to open an editor(VS Code, Notepad++, Sublime Text, etc) then save the file with that name and no extension. Sometimes you have to save the name with quotes around it (exp. "Dockerfile").
3. Once file is created enter the following code into the Dockerfile:

```
FROM elasticsearch:6.8.23


RUN /usr/share/elasticsearch/bin/elasticsearch-plugin install --batch ingest-attachment
```

4. Open power shell and navigate to the file location of the Dockerfile. Then run the following to build the image in docker:

```
docker build -t elasticsearch-ingest-attachment .
```

5. Next in power shell run the following to create the container and run for the first time:

```
docker run --name elastic-search-6.8.23 -p 9200:9200 -p 9300:9300 -e "discovery.type=single-node"
elasticsearch-ingest-attachment
```

6. Elastic search is now up and running. The only problem is it is running in power shell and will stay up on the screen. To resolve this please close the power shell which will stop

elastic search. Then open Docker, in the container menu find the elastic search container and hit the start button. Elastic search will be then good to go.

Note: This install of Elastic Search includes the Ingest add-on that is required for Rubex.

For Testing if it is up and running. Run the following code in power shell: `curl -GET "localhost:9200"`

The following will then display which shows it is up and running:

```
{
  "name" : "64adb598c086",
  "cluster_name" : "docker-cluster",
  "cluster_uuid" : "_jyHQb0LTeSlJxenN4MO1g",
  "version" : {
    "number" : "6.8.23",
    "build_flavor" : "default",
    "build_type" : "docker",
    "build_hash" : "ef48eb35cf30adf4db14086e8aabd07ef6fb113f",
    "build_date" : "2020-03-26T06:34:37.794943Z",
    "build_snapshot" : false,
    "lucene_version" : "8.4.0",
    "minimum_wire_compatibility_version" : "6.8.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}
```

DEBUGGING:

Sometimes elastic search can get into readonly mode. You can tell if this has happened when you can connect to elasticsearch successfully, but it fails to index or purge any new data. You can verify this is the issue by checking the error messages in utopia, or in the docker elasticsearch logs you can find this string.

```
max virtual memory areas vm.max_map_count [65530] is too low, increase to at least [262144]
```

To increase the max memory allocated to this container, run the following commands in PowerShell:

```
wsl -d docker-desktop
```

```
sysctl -w vm.max_map_count=262144
```

Stop the elasticsearch container and restart it.

NOTE:
If the database is still in read-only mode, run the following curl command from the docker terminal.

```
curl -XPUT -H "Content-Type: application/json" http://localhost:9200/nodebaseindexitemindex/_settings -d '{"index.blocks.read_only_allow_delete": null}'
```

If this STILL doesn't work (or the command fails), download and unzip Kaizen.zip
Then, once unzipped run kaizen.bat, and connect to localhost elasticsearch, then delete the index and re-run utopia to re-create the index.

It should work...

Turn off old elastic search indexing

- File Processing
  - set the feature flag ***AtlantisFileProcessingOldElasticSearchIndexing*** to false
- Utopia
  - set the feature flag ***UtopiaOldFileIndexing*** to false

How To Run (locally pointed at staging)

this is how to run utopia and atlantis-search pointed at demo but running locally for debugging.

1. connect to VPN (or be in office)
2. change the Utopia appsettings.json DbConnectionConfigurationData to the staging db
3. temporarily add an AtlantisSearchBaseUrl property to the same appsettings.json file
    1. value for this is "http://localhost:7008"
4. temporarily add an AtlantisSearchAppKey property to the same appsettings.json file
    1. value for this is ____ possibly a secret so starts with ZA... (Check Azure for it)
5. run atlantis-search and utopia projects
6. your data should be connected to staging but the code that is running is all local.
7. obviously log in and run a search.

(this isn't rocket science but it took me a while to figure out so i figured I'd share)

In Progress

Deployment?

spin up azure resources for search

setup identities and permissions to key vault

set up app settings on function apps

configure all the key vault and config settings (You are currently configuring key vault and azure config values)

- es api key can be created in kibana=> stack management => security => api keys
- file processing function app needs networking configured as well as app settings (Environ, Region, SearchServiceAddress, and AppConfig)

deploy with pipeline

use management tools to setup up elastic search

setting replicas to 0

PUT index-name-here/_settings

```
{
 "index" : {
  "number_of_replicas":0
 }
}
```

....

add file processing function app stuffs

All

- DONE

UK

- turn on new search

CA

- turn on new search

GOV

- migrate extracted text
- index all account data
- turn on new search