Elasticsearch Bulk Api for data migration

Craig Godden-Payne
Oct 26, 2018 · 5 min read ★

At work, I recently worked on a project to move from a self hosted elasticsearch cluster in Azure, to a managed elasticsearch cluster in AWS. The cluster contained analytics tracking information from the website, and we held approximately 3 million documents per day. Some extra challenges that I faced were that I was going from 5.2 of elasticsearch to 6.2, and the mapping that I used had some deprecated types, so the data migration would be somewhat challenging.

The bigger picture

Without giving too much away, we scraped the logs from the webserver behind the load balancer for the whole site, alongside sending custom events from the client. For this we used filebeat, and logstash to interpret the log entry, and build up the final document to elasticsearch. The documents are quite large (usually containing at least 50 individual properties), and we had about a years worth of data. We were not able to reindex the data from source, as we only maintained the logs for 30 days, so the idea was to migrate the data from the old instance to the new instance using the bulk api.

Migration only involved the instance of elasticsearch, filebeat and logstash were to be upgraded, but were going to be maintained, in place.

Migration to AWS using terraform

It was relatively straightforward to setup the elasticsearch service cluster in AWS, using terraform. Below is pretty much what was setup (minus all the security group and dns settings etc.)

```
resource "aws_elasticsearch_domain" "craigs-elasticsearch" {
  domain_name           = "craigs-analytics"
  elasticsearch_version = "6.2"

  cluster_config {
    instance_type            = "${terraform.workspace == "prod" ?
"m4.large.elasticsearch" : "t2.medium.elasticsearch"}"
    instance_count           = "1"
    dedicated_master_enabled = false
    zone_awareness_enabled   = false
  }

  vpc_options {
    security_group_ids = ["${aws_security_group.craigs-
elasticsearch.id}"]
    subnet_ids         = ["${data.aws_subnet_ids.private.ids[0]}"]
  }

  advanced_options {
    "rest.action.multi.allow_explicit_index" = "true"
    "indices.query.bool.max_clause_count"    = 1024
  }

  ebs_options {
    ebs_enabled = true
    volume_type = "gp2"
    volume_size = "${terraform.workspace == "prod" ? "500" : "35" }"
  }

  snapshot_options {
    automated_snapshot_start_hour = 0
  }

  tags {
    Domain = "craigs-elasticsearch-${terraform.workspace}"
  }
}

resource "aws_elasticsearch_domain_policy" "craigs-elasticsearch" {
  domain_name = "${aws_elasticsearch_domain.craigs-
elasticsearch.domain_name}"

  access_policies = <<POLICIES
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": "es:*",
            "Principal": "*",
            "Effect": "Allow",
            "Resource": "${aws_elasticsearch_domain.craigs-
elasticsearch.arn}/*"
        }
    ]
}
POLICIES
}
```

Once the cluster was up and running, I tried one of the more well known tools for data migration, ElasticDump. I was faced with the problem, where I had a breaking change within the mapping document. I frustratingly looked through the documentation to see if there was a way to transform the data, after retrieving the records, before sending to the new cluster, but didn't have much luck.

I started looking at NEST (because I'd used it before and seemed quite good), but it was incredibly slow.

I basically wrote my own tool, which (using the bulk api) read from the old cluster, transformed the data, and wrote to the new cluster. It took a while to migrate the data, but it was much faster than the alternatives. Not the best code, I admit, but did the job in a very limited timeframe!

```
using System;
using System.Diagnostics;
using System.Net.Http;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using Newtonsoft.Json;
using Newtonsoft.Json.Linq;

namespace migration
{
    class Program
    {
        static void Main(string[] args)
        {
            try
            {
                if (args.Length != 4)
                    throw new Exception("You must specify 4 args,
[second to start from] [day] [month] [year]");

                var seconds = int.Parse(args[0]);
                var day = int.Parse(args[1]);
                var month = int.Parse(args[2]);
                var year = int.Parse(args[3]);

                Task.WaitAny(Timer(), Migrate(seconds, day, month,
year));
            }
            catch (Exception e)
            {
                Console.WriteLine(e);
                Console.ReadKey();
            }
        }

        public static Task Timer()
        {
            var task = new Task(() =>
            {
                while (true)
                {
                    Thread.Sleep(5000);
                    var proc = Process.GetCurrentProcess();
                    var mem = proc.WorkingSet64;
                    var cpu = proc.TotalProcessorTime;
                    Console.WriteLine("Still working. CPU used:" +
cpu.TotalMilliseconds + " MEM used:" + mem);
                }
            });
            task.Start();
            return task;
        }
        public static Task Migrate(int startIndex, int day, int
month, int year)
        {
            //The bulk size must not exceed 10000 records within the
timeframe, or it will be missed from the bulk get.
            var bulkSizeInOfTime = TimeSpan.FromMinutes(10);

            const string oldServerUrl = "http://10.128.1.1:9200";
            const string newServerUrl = "http://craig-
elasticsearch.prod.goddenpayne.com:80";
            var oldServerClient = new HttpClient { BaseAddress = new
Uri(oldServerUrl) };
```

```
            var newServerClient = new HttpClient { BaseAddress = new 
Uri(newServerUrl) };
            var index = "logstash-" + year.ToString("0000") + "." + 
month.ToString("00") + "." + day.ToString("00");

            var task = new Task(() =>
            {
                Console.WriteLine("Using Index [" + index + "]");
                for (int second = startIndex; second < 86400;) 
//86400 seconds in a day
                {
                    var startDate = new DateTime(year, month, day, 
0, 0, 0, 0);
                    startDate = startDate.AddSeconds(second);
                    var startDateString = startDate.ToString("yyyy-
MM-ddTHH:mm:ssZ");
                    var endDateString = 
startDate.AddSeconds(bulkSizeInOfTime.TotalSeconds).ToString("yyyy-
MM-ddTHH:mm:ssZ");
                    Console.WriteLine("Currently working on " + 
startDateString + " to " + endDateString);

                    //fetch data
                    var records = oldServerClient.PostAsync(
                        "logstash-" + year + ".*/_search?
typed_keys=true&search_type=query_then_fetch", new StringContent(
                        @"{
                        ""from"" : 0,
                        ""size"" : 10000,
                        ""query"": {
                            ""range"": {
                                ""requesttime"": {
                                    ""gte"": """ + startDateString + 
@""",
                                    ""lte"": """ + endDateString + 
@""" 
                                }
                            }
                        }
                        }", Encoding.UTF8, 
"application/json")).Result;

                    var data = 
JObject.Parse(records.Content.ReadAsStringAsync().Result);
                    var payload = "";

                    var count = 0;
                    foreach (var hit in data["hits"]["hits"])
                    {
                        var id = hit["_id"].Value<string>();
                        var document = 
hit["_source"].ToString(Formatting.None);
                        payload += "{ \"create\" : { \"_index\" : 
\"" + index + "\", \"_type\" : \"doc\", \"_id\" : \"" + id + "-1" + 
"\" } }\n";
                        payload += document + "\n";
                        count++;
                    }
                    if (payload != "")
                    {
                        var result =
newServerClient.PostAsync("_bulk", new StringContent(payload,
Encoding.UTF8, "application/json")).Result;
                        Console.WriteLine("Bulk insert " + count + "
records: " + result.StatusCode);
                    }
                    second = second +
((int)bulkSizeInOfTime.TotalSeconds);
                }
            });
            task.Start();
            return task;
        }
    }
}
```

. . .

Originally published at craig.goddenpayne.co.uk.

Elasticsearch API Csharp Dotnetcore