Prerequisites

Install JRE
https://www.java.com/ru/download/manual.jsp
and JDK
https://www.oracle.com/java/technologies/downloads/
JDBC input plugin. I think I run tool like it was advised here
Install JDBC driver. Make sure versions are compatible
https://docs.microsoft.com/en-us/sql/connect/jdbc/system-requirements-for-the-jdbc-driver?view=sql-server-ver16

Data table - SQL

```
USE [ProductStock]
GO

SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[ElasticEntities](
	[Id] [int] IDENTITY(1,1) NOT NULL,
	[Name] [nvarchar](50) NULL,
	[Description] [nvarchar](200) NULL,
	[IsActive] [bit] NOT NULL,
	[CreatedAt] [datetimeoffset](7) NOT NULL,
	[ModifiedAt] [datetimeoffset](7) NULL,
	[CreatedBy] [int] NULL,
	[ModifiedBy] [int] NULL,
 CONSTRAINT [PK_ElasticEntities] PRIMARY KEY CLUSTERED
(
	[Id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]
) ON [PRIMARY]
GO
```

Direct data synchronizing

I used mostly **this article**
For MS SQL Server I came up with such configuration:

```
input {
  jdbc {
    jdbc_driver_library => "C:/Program Files/Microsoft JDBC DRIVER 10.2 for SQL
Server/enu/mssql-jdbc-10.2.1.jre17.jar"
    jdbc_driver_class => "com.microsoft.sqlserver.jdbc.SQLServerDriver"
    jdbc_connection_string =>
"jdbc:sqlserver://localhost;databaseName=ProductStock;encrypt=true;trustServerCerti
ficate=true"
    jdbc_user => "sa2"
    jdbc_password => "sa2"
    jdbc_paging_enabled => true
    tracking_column => "unix_ts_in_secs"
    use_column_value => true
    tracking_column_type => "numeric"
    schedule => "*/5 * * * * *"
      statement => "SELECT *, DATEDIFF(SECOND,'1970-01-01', ISNULL(ModifiedAt,
CreatedAt)) AS unix_ts_in_secs FROM ElasticEntities WHERE (DATEDIFF(SECOND,'1970-
01-01', ISNULL(ModifiedAt, CreatedAt)) > :sql_last_value AND (ModifiedAt <
GETUTCDATE() OR ModifiedAt IS NULL)) ORDER BY ModifiedAt ASC OFFSET 0 ROWS"
  }
}
filter {
  mutate {
    copy => { "id" => "[@metadata][_id]"}
    remove_field => ["id", "@version", "unix_ts_in_secs"]
  }
}
output {
  stdout { codec =>  "rubydebug"}
  elasticsearch {
      hosts => ["https://localhost:9200"]
         ssl => true
         cacert => './config/certs/ca.pem'
         #user => "elasticsearch"
         #password => "9w=8x+XZY5xS1_q0VirV"
         user => "logstash_internal"
      password => "x-pack-test-password"
      index => "rdbms_sync_idx"
      document_id => "%{[@metadata][_id]}"
  }
}
```

This config works with assumptions:

1. logstash_internal user was created in Kibana’s Dev. console:

```
POST _security/user/logstash_internal
{
  "password" : "x-pack-test-password",
  "roles" : [ "logstash_writer"],
  "full_name" : "Internal Logstash User"
}
```

2. JDBC driver was extracted into `C:/Program Files/Microsoft JDBC DRIVER 10.2 for SQL Server`
3. Database `ProductStock` exists with table `ElasticEntities`. CreatedAt if filled automatically, ModifiedAt is nullable (differs from the article)

To run Logstash I passed config as a parameter in Windows Powershell:

```
>cd D:\logstash-8.3.2
>.\bin\logstash.bat -f ..\DEV\demostock\logstash\jdbc.config
```

I also had to copy certificate from Elasticsearch installation directory (from Docker container in my case) into D:\logstash-8.3.2\config\certs and change file extension from .crt to .pem

.NET client

Documentation for Elastic.Clients.Elasticsearch library
https://www.elastic.co/guide/en/elasticsearch/client/net-api/current/index.html
For Autofac DI, I created a separate module. elastic client is instantiated as object in RegisterClient method:

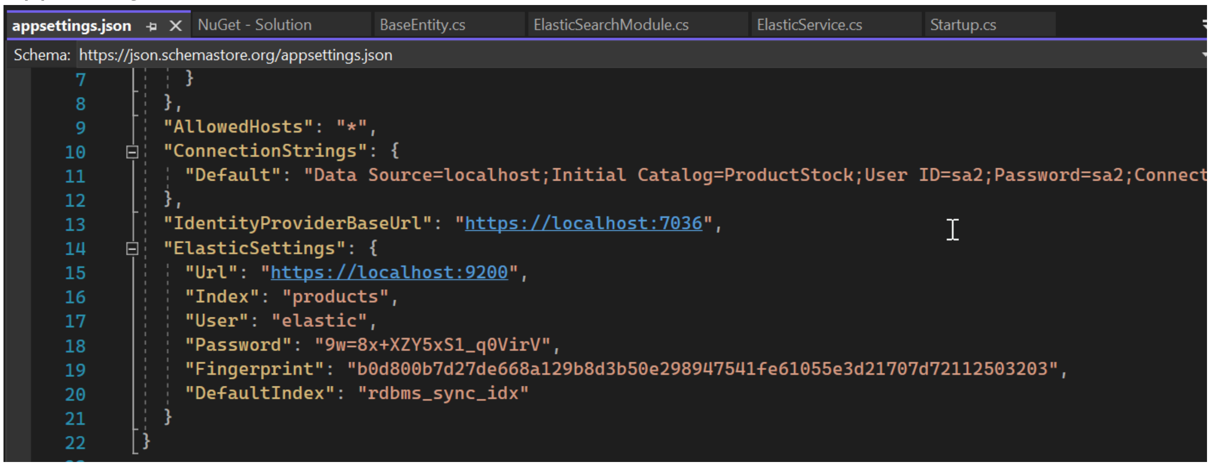
```
using System;
using Autofac;
using Elastic.Clients.Elasticsearch;
using Elastic.Transport;
using ProductStock.Business;
using ProductStock.Dto;
using ProductStock.Infrastructure.Abstractions;

namespace ProductStock.Bootstrapper
{
    public class ElasticSearchModule : Autofac.Module
    {
        private ElasticSettings elkSettings;

        public ElasticSearchModule(ElasticSettings elkSettings)
        {
            this.elkSettings = elkSettings;
        }

        protected override void Load(ContainerBuilder builder)
        {
            base.Load(builder);
            RegisterClient(builder);
            RegisterDomainService(builder);
        }

        private void RegisterDomainService(ContainerBuilder builder)
        {
            builder.RegisterType<ElasticService>()
                .As<IFilterableDomainService<int, ElasticDto>>()
                .InstancePerLifetimeScope();
        }

        private void RegisterClient(ContainerBuilder builder)
        {
            var settings = new ElasticsearchClientSettings(new
Uri(elkSettings.Url))
                .CertificateFingerprint(elkSettings.Fingerprint)
                .DefaultIndex(elkSettings.DefaultIndex)
                .Authentication(new BasicAuthentication(elkSettings.User,
elkSettings.Password));
            var client = new ElasticsearchClient(settings);

            builder.RegisterInstance(client).As<ElasticsearchClient>();
        }
    }
}
```

In Startup class, we read settings from appsettings.json and call module:

```
var elkSettings =
Configuration.GetSection(typeof(ElasticSettings).Name).Get<ElasticSettings>();
builder.RegisterModule(new ElasticSearchModule(elkSettings));
```

where settings is a class:
```
namespace ProductStock.Bootstrapper
{
    public class ElasticSettings
    {
        public string Url { get; set; }
        public string DefaultIndex { get; set; }
        public string User { get; set; }
        public string Password { get; set; }
        public string Fingerprint { get; set; }
    }
}
```

Appsettings have data:

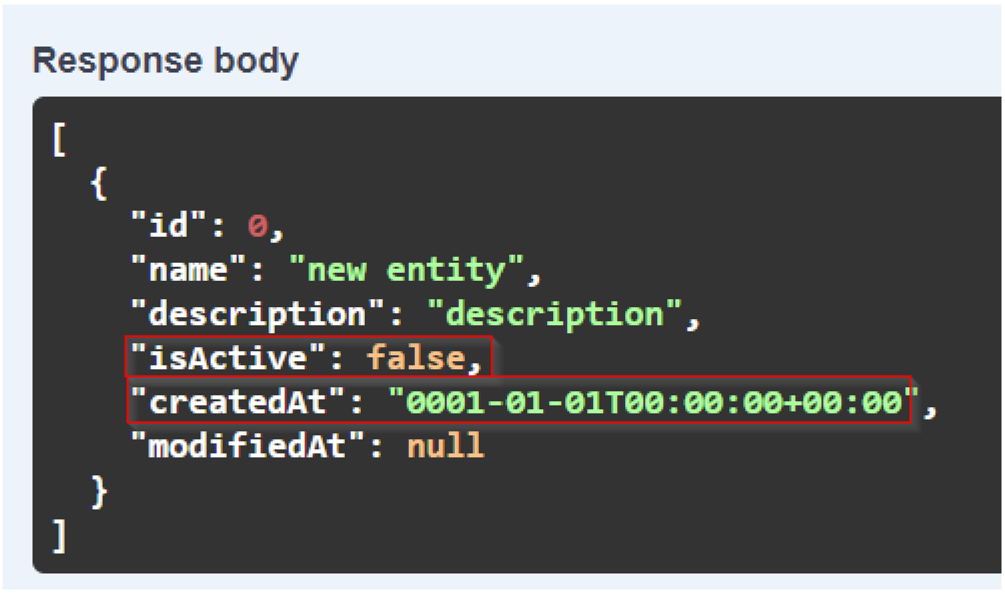


```
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "Default": "Data Source=localhost;Initial Catalog=ProductStock;User ID=sa2;Password=sa2;Connect
  },
  "IdentityProviderBaseUrl": "https://localhost:7036",
  "ElasticSettings": {
    "Url": "https://localhost:9200",
    "Index": "products",
    "User": "elastic",
    "Password": "9w=8x+XZY5xS1_q0VirV",
    "Fingerprint": "b0d800b7d27de668a129b8d3b50e298947541fe61055e3d21707d72112503203",
    "DefaultIndex": "rdbms_sync_idx"
  }
}
```

To retrieve data, we need to inject client and execute query
```
using System.Linq;
using System.Collections.Generic;
using System.Threading.Tasks;
using Elastic.Clients.Elasticsearch;
using Elastic.Clients.Elasticsearch.QueryDsl;
using ProductStock.Dto;
using ProductStock.Data.Models;
using ProductStock.Infrastructure.Abstractions;
using ProductStock.Infrastructure.Exceptions;

namespace ProductStock.Business
{
    public class ElasticService : DomainService<int, ElasticEntity, ElasticDto>,
IFilterableDomainService<int, ElasticDto>
    {
        ElasticsearchClient _client;
        public ElasticService(ElasticsearchClient client,
IRepository<ElasticEntity> repository, IEntityConverter entityConverter)
            : base(repository, entityConverter)
        {
            this._client = client;
        }
        /// <summary>
        /// Executes query:
        /// POST rdbms_sync_idx/_search
        ///    "query": {
        ///        "bool": {
```

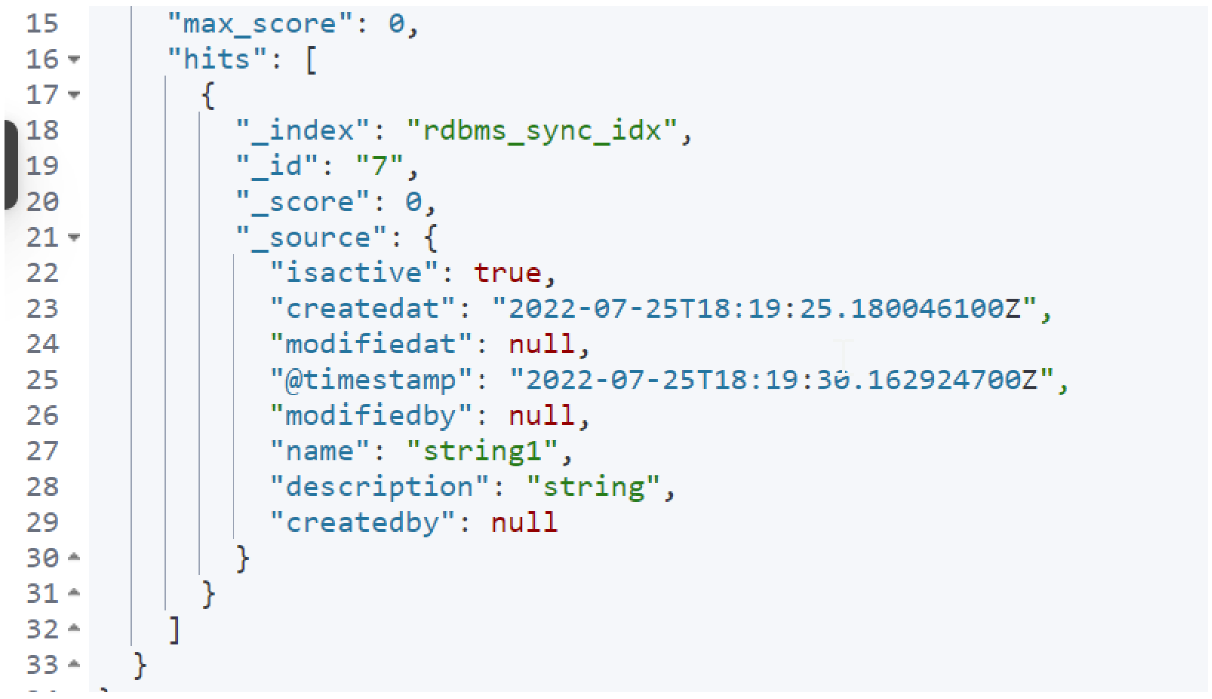
```
        ///             "filter": [
        ///                 { "term": { "isactive": true }},
        ///                 { "query_string": { "query": "*string1*" }}
        ///             ]
        ///         }
        ///     }
        /// </summary>
        /// <param name="keyword"></param>
        /// <returns></returns>
        public async Task<IReadOnlyCollection<ElasticDto>> Filter(string keyword)
        {
            var searchResponse = await _client.SearchAsync<ElasticEntity>(
                    s => s.Query(
                        b => b.Bool(m => m.Filter(
                            t => t.Term(new TermQuery { Field = new
Field("isactive"), Value = true }),
                            q => q.QueryString(
                                d => d.Query('*' + keyword + '*')
                    )))).Size(5000));
            var hits =
_entityConverter.ConvertTo<IReadOnlyCollection<ElasticEntity>,
IReadOnlyCollection<ElasticDto>>(searchResponse.Documents);
            return hits;
        }
……
    }
}
```

There is a little problem with mapping: because of lower-case registry, properties are not correct in a final result:


Response body

```
[
  {
    "id": 0,
    "name": "new entity",
    "description": "description",
    "isActive": false,
    "createdAt": "0001-01-01T00:00:00+00:00",
    "modifiedAt": null
  }
]
```

in spite of data extraction is correct in kibana’s dev. console:



```
    "max_score": 0,
    "hits": [
      {
        "_index": "rdbms_sync_idx",
        "_id": "7",
        "_score": 0,
        "_source": {
          "isactive": true,
          "createdat": "2022-07-25T18:19:25.180046100Z",
          "modifiedat": null,
          "@timestamp": "2022-07-25T18:19:30.162924700Z",
          "modifiedby": null,
          "name": "string1",
          "description": "string",
          "createdby": null
        }
      }
    ]
  }
```