Alibaba Cloud Elasticsearch

開発者ガイド

Document Version20200407

目次

1 RESTful API

Elasticsearch は RESTful API をサポートしているので、HTTP API を使用してエイリアスの追加、削除、変更、検索、設定などの操作が可能です。 このトピックでは、ビジネス運用に RESTful API を使用する方法について説明します。

注：

詳細については、『Elasticsearch RESTful API』をご参照ください。

Elasticsearch reference [5.5]

Single document APIs

- Index API
- Get API
- Delete API
- Update API

Multi-document APIs

- Multi get API
- Bulk API
- Delete by query API
- Update by query API
- Reindex API

Java REST クライアントを使用したクラスターアクセス

HTTP と TCP を介したアクセスのみがサポートされています。 Java REST クライアントを使用することを推奨します。

Java API を使用したクラスターアクセス

Elasticsearch は、Java ユーザー向けのクラスターアクセスを提供しています。 Java API の詳細については、『Java API』をご参照ください。

トランスポートクライアント

トランスポートクライアントは、クラスター内のノードにリクエストを送信します。 ただし、トランスポートクライアントはクラスターの一部ではありません。

トランスポートクライアントは、Elasticsearch Transport Protocol を使用して、ポート 9300 でクラスターと通信します。

また、クラスター内のノードも、ポート 9300 を使用して相互に通信します。 ノードをクラスターにグループ化する前に、ノードのポート 9300 を開く必要があります。

 注：

Java クライアントとノードは、相互に認識できるよう、同じ Elasticsearch バージョンを使用する必要があります。

RESTful API (HTTP)

他の言語は RESTful API を使用して、ポート 9200 経由で Elasticsearch と通信できます。 任意の Web クライアントや curl コマンドを使用して、Elasticsearch と通信できます。

 注：

Elasticsearch は、Groovy、Javascript、.NET、PHP、Perl、Python、Ruby などの言語向けの公式クライアントを提供しています。

コミュニティで提供されるクライアントとプラグインの詳細については、『コミュニティで提供されるクライアント』をご参照ください。

HTTP を介した **curl** リクエストの構造

curl -X<VERB> '<PROTOCOL>://<HOST>:<PORT>/<PATH>? <QUERY_STRING>' -d '<BODY>'

- VERB：HTTP メソッド (GET、POST、PUT、HEAD、DELETE)。
- PROTOCOL：http または https。 Elasticsearch で HTTPS が有効になっている場合のみ、https を使用します。
- HOST：Elasticsearch クラスター内のノードのホスト名。 ローカルノードを使用する場合、このフィールドを localhost に設定します。
- PORT：Elasticsearch HTTP サービスを実行するポート。 デフォルトのポート番号は、9200 です。
- PATH：API エンドポイント。 たとえば、 _count はクラスター内のドキュメント数を返します。 PATH には、_cluster/stats や _nodes/stats/jvm など、複数の要素が含まれます。
- QUERY_STRING：クエリパラメーター (オプション)。 たとえば、?pretty パラメーターを指定すると、リクエストによって返される JSON 形式のデータが読みやすくなります。
- BODY：JSON 形式のリクエスト本文。 このフィールドは、本文を必要とするリクエストの場合にのみ必要です。

例

Elasticsearch クラスター内のドキュメント数のカウント

```
curl -XGET 'http://localhost:9200/_count?pretty' -d '
{
  "query": {
    "match_all": {}
  }
}'
```

レスポンス：

```
{
    "count" : 0,
    "_shards" : {
      "total" : 5,
      "successful" : 5,
      "failed" : 0
    }
}
```

curl -i コマンドを使用して HTTP ヘッダーを取得することができます (curl -i -XGET 'localhost:9200/' など)。

完全なサンプルリクエスト

```
curl -XGET 'localhost:9200/_count? pretty' -d '
{
    "query": {
      "match_all": {}
    }
}'
```

短縮版のサンプルリクエスト

```
GET /_count
{
    "query": {
      "match_all": {}
    }
}
```

2 Java API

2.1 概要

このドキュメントでは、Elasticsearch の Java High Level Rest Client の原則、バージョンの互換性、使用例について説明します。 Java REST クライアントを使用して、Alibaba Cloud Elasticsearch サーバーと対話できます。 Java REST クライアントは、取得と分析の実行に役立ちます。

既存のコードを Transport クライアントから REST クライアントに移行する

Transport クライアントは、Elasticsearch の最初のバージョンで公開された特別なクライアントです。 Transport クライアントは、TCP を使用して Elasticsearch インスタンスと通信します。 したがって、Transport クライアントが異なるバージョンの Elasticsearch インスタンスと通信する場合、互換性の問題が発生する可能性があります。 詳細については、『Motivations around a new Java client』をご参照ください。

Low Level REST クライアントは、2016 年に Elasticsearch の公式 Web サイトでリリースされました。 Low Level REST クライアントは Apache HTTP クライアントに基づいており、HTTP を使用したすべてのバージョンの Elasticsearch クラスターとの通信を可能にします。 Elasticsearch は、Low Level REST クライアントに基づいた High Level Rest Client をリリースしました。

Transport クライアントは Elasticsearch 7.0 では使用されなくなり、Elasticsearch 8.0 では使用できなくなりました。 したがって、Java REST クライアントを使用することを推奨します。 REST クライアントは HTTP リクエストを使用して、リクエストとレスポンスのシリアル化の問題を処理し、ビジネス開発を容易にします。

Java REST クライアント

Elasticsearch は、次の 2 種類の Java REST クライアントをサポートしています。

- Java Low Level REST Client：HTTP リクエストを使用して Elasticsearch クラスターとの通信を可能にします。 API は、データのエンコードとデコードを行いません。 すべての Elasticsearch バージョンと互換性があります。
- Java High Level Rest Client ：このタイプのクライアントを例として使用します。 Java High Level Rest Client は、Java Low Level REST Client に基づいています。 API 固有のメソッドを公開するように設計されています。 Java High Level Rest Client は、Elasticsearch のコアプロジェクトに依存しています。 クライアントは request オブジェクトを引数として受け取

り、response オブジェクトを返します。 すべての API は、同期的または非同期的に呼び出すことができます。

- 同期メソッドは response オブジェクトを返します。
- 名前に async サフィックスを持つ非同期メソッドには、リスナー引数が必要です。 リスナー引数は、応答またはエラーを受信したときに続行するように、対応するメソッドに通知します。

Java High Level Rest Client 6.3.x および 6.7.x の使用方法の詳細については、「Java REST Client 6.3.x」および「Java REST Client 6.7.x」 をご参照ください。

Elasticsearch の使用方法の詳細については、こちら をクリックしてください。

2.2 Java REST Client 6.3.x

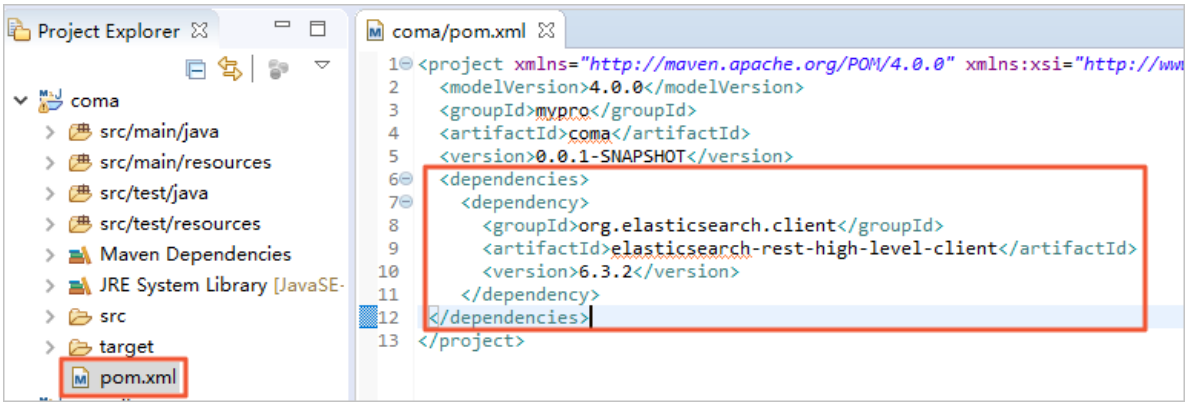
このドキュメントでは、Elasticsearch Java API の使用方法について説明します。 Java High Level REST Client 6.3.x を例として使用します。

前提条件

- JDK がインストールされていること。 JDK バージョンは 1.8 以降である必要があります。
- Alibaba Cloud Elasticsearch インスタンスが作成されていること。 インスタンスのバージョンは、 High Level Client のバージョン以上である必要があります。 このドキュメントでは、バージョン 6.3.2 のインスタンスが作成されます。

> **:**
> High Level Client は上位互換性があります。 たとえば、バージョン 6.3.2 の High Level Client は、バージョン 6.3.2 以降の Elasticsearch クラスターと通信できます。 最新のクライアントの機能を使用できるようにするには、クラスターバージョンと同じ高レベルのクライアントバージョンを使用することを推奨します。

- Java Maven プロジェクトを作成し、 Pom 依存関係 を Java プロジェクトの pom.xml ファイルに追加していること。



注：

このドキュメントでは、Elasticsearch 6.3.2 を例として使用しています。

Pom 依存関係

```
<dependency>
    <groupId>org.elasticsearch.client</groupId>
    <artifactId>elasticsearch-rest-high-level-client</artifactId>
    <version>6.3.2</version>
</dependency>
```

例

次のサンプルコードは、 **index** API を使用してドキュメントのインデックスを作成し、 **delete** API を使用してドキュメントを削除します。

```
// The file path is src/main/java/RestClientTest.java.
import org.apache.http.HttpHost;
import org.apache.http.auth.AuthScope;
import org.apache.http.auth.UsernamePasswordCredentials;
import org.apache.http.client.CredentialsProvider;
import org.apache.http.impl.client.BasicCredentialsProvider;
import org.apache.http.impl.nio.client.HttpAsyncClientBuilder;

import org.elasticsearch.action.delete.DeleteRequest;
import org.elasticsearch.action.delete.DeleteResponse;
import org.elasticsearch.action.index.IndexRequest;
import org.elasticsearch.action.index.IndexResponse;
import org.elasticsearch.client.RestClient;
import org.elasticsearch.client.RestClientBuilder;
import org.elasticsearch.client.RestHighLevelClient;

import java.io.IOException;
import java.util.HashMap;
import java.util.Map;

public class RestClientTest {
    public static void main(String[] args) {

        // The Alibaba Cloud Elasticsearch cluster requires basic authentication.
```

```
        final CredentialsProvider credentialsProvider = new BasicCredentialsProvider();
       // Use the username and password that you have specified when creating the
Alibaba Cloud Elasticsearch instance. The username and password are also used to log
on to the Kibana console.
        credentialsProvider.setCredentials（AuthScope.ANY、new UsernamePasswordCred
entials（"{username}"、 "{password}"））；

       // Create a REST client by using the builder and configure HttpClientConfigCallback
for the HTTP client.
       // Click the ID of the Elasticsearch instance. On the Basic Information page, obtain the
 public IP address of the Elasticsearch cluster.
        RestClientBuilder builder = RestClient.builder(new HttpHost("{Elasticsearch cluster
address}", 9200))
                .setHttpClientConfigCallback(new RestClientBuilder.HttpClientConfigCallback() {
                    @Override
                    public HttpAsyncClientBuilder customizeHttpClient(HttpAsyncClientBuilder
httpClientBuilder) {
                        return httpClientBuilder.setDefaultCredentialsProvider(credentialsProvider);
                    }
                });

        // Create a RestHighLevelClient instance by using the REST low-level client builder.
        RestHighLevelClient highClient = new RestHighLevelClient(builder);

        try {
            // Create a request.
            Map<String, Object> jsonMap = new HashMap<>();
           // field_01 and field_02 are the field names, and value_01 and value_02 are the
corresponding values.
            jsonMap.put("{field_01}", "{value_01}");
            jsonMap.put("{field_02}", "{value_02}");
            // index_name is the index name, type_name is the type name, and doc_id is the
document ID.
            IndexRequest indexRequest = new IndexRequest("{index_name}", "{type_name}", "{
doc_id}").source(jsonMap);

            // Run the following command in parallel.
            IndexResponse indexResponse = highClient.index(indexRequest);

            long version = indexResponse.getVersion();

            System.out.println("Index document successfully! " + version);
            // index_name is the index name, type_name is the type name, and doc_id is the
 document ID. The index name, type name, and document ID are the same as that you
have specified when creating the index.
            DeleteRequest request = new DeleteRequest("{index_name}", "{type_name}", "{
doc_id}");
            DeleteResponse deleteResponse = highClient.delete(request);

            System.out.println("Delete document successfully!") ;

            highClient.close();

        } catch (IOException ioException) {
            // Handle exceptions.
        }
    }
}
```

中かっこ {} 内に含まれる前述のサンプルコードのパラメーターを、サービス固有のパラメーターに置き換えることができます。 詳細については、コードのコメントをご参照ください。

2.3 Java REST Client 6.7.x

このドキュメントでは、Elasticsearc Java API の使用方法について説明します。 Java High Level REST Client 6.7.x を例として使用しています。

前提条件

- JDK がインストールされていること。 JDK のバージョンは 1.8 以降である必要があります。
- Alibaba Cloud Elasticsearch インスタンスが作成されていること。 インスタンスのバージョンは、高レベルクライアントのバージョン以上である必要があります。 このドキュメントでは、バージョン 6.7.0 のインスタンスが作成されます。

  :
  高レベルのクライアントは上位互換性があります。 たとえば、バージョン 6.7.0 の高レベルクライアントは、バージョン 6.7.0 以降のバージョンの Elasticsearch クラスターと通信できます。 最新のクライアントの機能を使用できるようにするには、クラスターバージョンと同じ高レベルのクライアントバージョンを使用することを推奨します。
- Java Maven プロジェクトを作成し、 Pom 依存関係を Java プロジェクトの pom.xml ファイルに追加していること。

注：
このドキュメントでは、Elasticsearch 6.7.0 を例として使用しています。

Pom 依存関係

```
<dependency>
    <groupId>org.elasticsearch.client</groupId>
    <artifactId>elasticsearch-rest-high-level-client</artifactId>
    <version>6.7.0</version>
```

```
</dependency>
```

RequestOptions

REST クライアント 6.3.2 と比べて、バージョン 6.7.0 には RequestOptions クラスが追加されています。 通常の Elasticsearch リクエストに影響を与えることなくオプションを指定できます。

次のサンプルコードでは、ResponseConsumer 設定を調整して、Java 仮想マシン (JVM) メモリが制限されているクライアント環境で非同期応答が使用するキャッシュのサイズを制限しています。 すべてのサンプルコードの詳細については、「例」をご参照ください。

```
private static final RequestOptions COMMON_OPTIONS;

static {
   RequestOptions.Builder builder = RequestOptions.DEFAULT.toBuilder();

   // The default cache size is 100 MB. Change it to 30 MB.
   builder.setHttpAsyncResponseConsumerFactory(
         new HttpAsyncResponseConsumerFactory
               .HeapBufferedResponseConsumerFactory(30 * 1024 * 1024));
   COMMON_OPTIONS = builder.build();
}
```

```
// Run the following command in parallel and use the custom RequestOptions (
COMMON_OPTIONS).
IndexResponse indexResponse = highClient.index(indexRequest, COMMON_OPTIONS);
```

RequestOptions クラスの使用方法の詳細については、 『 RequestOptions』をご参照ください。

例

次のコードは、 RequestOptions のすべてのサンプルコードです。 サンプルコードでは **index** API を使用してドキュメントのインデックスを作成し、 **delete** API を使用してドキュメントを削除します。 ResponseConsumer 設定を調整して、Java 仮想マシン (JVM) メモリが制限されているクライアント環境で非同期応答が使用するキャッシュのサイズを制限しています。

```
// The file path is src/main/java/RestClientTest67.java.
import org.apache.http.HttpHost;
import org.apache.http.auth.AuthScope;
import org.apache.http.auth.UsernamePasswordCredentials;
import org.apache.http.client.CredentialsProvider;
import org.apache.http.impl.client.BasicCredentialsProvider;
import org.apache.http.impl.nio.client.HttpAsyncClientBuilder;

import org.elasticsearch.action.delete.DeleteRequest;
import org.elasticsearch.action.delete.DeleteResponse;
import org.elasticsearch.action.index.IndexRequest;
import org.elasticsearch.action.index.IndexResponse;
import org.elasticsearch.client.*;

import java.io.IOException;
import java.util.HashMap;
import java.util.Map;
```

```
public class RestClientTest67 {

    private static final RequestOptions COMMON_OPTIONS;

    static {
        RequestOptions.Builder builder = RequestOptions.DEFAULT.toBuilder();

        // The default cache size is 100 MB. Change it to 30 MB.
        builder.setHttpAsyncResponseConsumerFactory(
                new HttpAsyncResponseConsumerFactory
                        .HeapBufferedResponseConsumerFactory(30 * 1024 * 1024));
        COMMON_OPTIONS = builder.build();
    }

    public static void main(String[] args) {
        // The Alibaba Cloud Elasticsearch cluster requires basic authentication.
        final CredentialsProvider credentialsProvider = new BasicCredentialsProvider();
       // Use the username and password that you have specified when creating the Alibaba Cloud Elasticsearch instance. The username and password are also used to log on to the Kibana console.
        credentialsProvider.setCredentials(AuthScope.ANY, new UsernamePasswordCredentials("{username}", "{password}");

        // Create a REST client by using the builder and configure HttpClientConfigCallback for the HTTP client.
       // Click the ID of the Elasticsearch instance. On the Basic Information page, obtain the public IP address of the Elasticsearch cluster.
        RestClientBuilder builder = RestClient.builder(new HttpHost("{Elasticsearch cluster address}", 9200))
                .setHttpClientConfigCallback(new RestClientBuilder.HttpClientConfigCallback() {
                    @Override
                    public HttpAsyncClientBuilder customizeHttpClient(HttpAsyncClientBuilder httpClientBuilder) {
                        return httpClientBuilder.setDefaultCredentialsProvider(credentialsProvider);
                    }
                });

        // Create a RestHighLevelClient instance by using the REST low-level client builder.
        RestHighLevelClient highClient = new RestHighLevelClient(builder);

        try {
            // Create a request.
            Map<String, Object> jsonMap = new HashMap<>();
           // field_01 and field_02 are the field names, and value_01 and value_02 are the corresponding values.
            jsonMap.put("{field_01}", "{value_01}");
            jsonMap.put("{field_02}", "{value_02}");
            // index_nameはインデックス名、type_nameはタイプ名、doc_idはドキュメントIDです。
            IndexRequest indexRequest = new IndexRequest("{index_name}", "{type_name}", "{doc_id}").source(jsonMap);

            // Run the following command in parallel and use the custom RequestOptions (COMMON_OPTIONS).
            IndexResponse indexResponse = highClient.index(indexRequest, COMMON_OPTIONS);

            long version = indexResponse.getVersion();

            System.out.println("Index document successfully! " + version);
            // index_name is the index name, type_name is the type name, and doc_id is the document ID. The index name, type name, and document ID are the same as that you have specified when creating the index.
```

```
        DeleteRequest request = new DeleteRequest("{index_name}", "{type_name}", "{
doc_id}");
        DeleteResponse deleteResponse = highClient.delete(request, COMMON_OPTIONS);

        System.out.println("Delete document successfully! \n" + deleteResponse.toString()
 + "\n" + deleteResponse.status());

        highClient.close（）;

      } catch (IOException ioException) {
        // Handle exceptions.
      }
    }
}
```

中括弧 {} 内に含まれる前述のサンプルコードのパラメーターを、サービス固有のパラメーターに置き換えることができます。 詳細については、コードのコメントをご参照ください。