Elasticsearch

By: Kumar, Rajnish

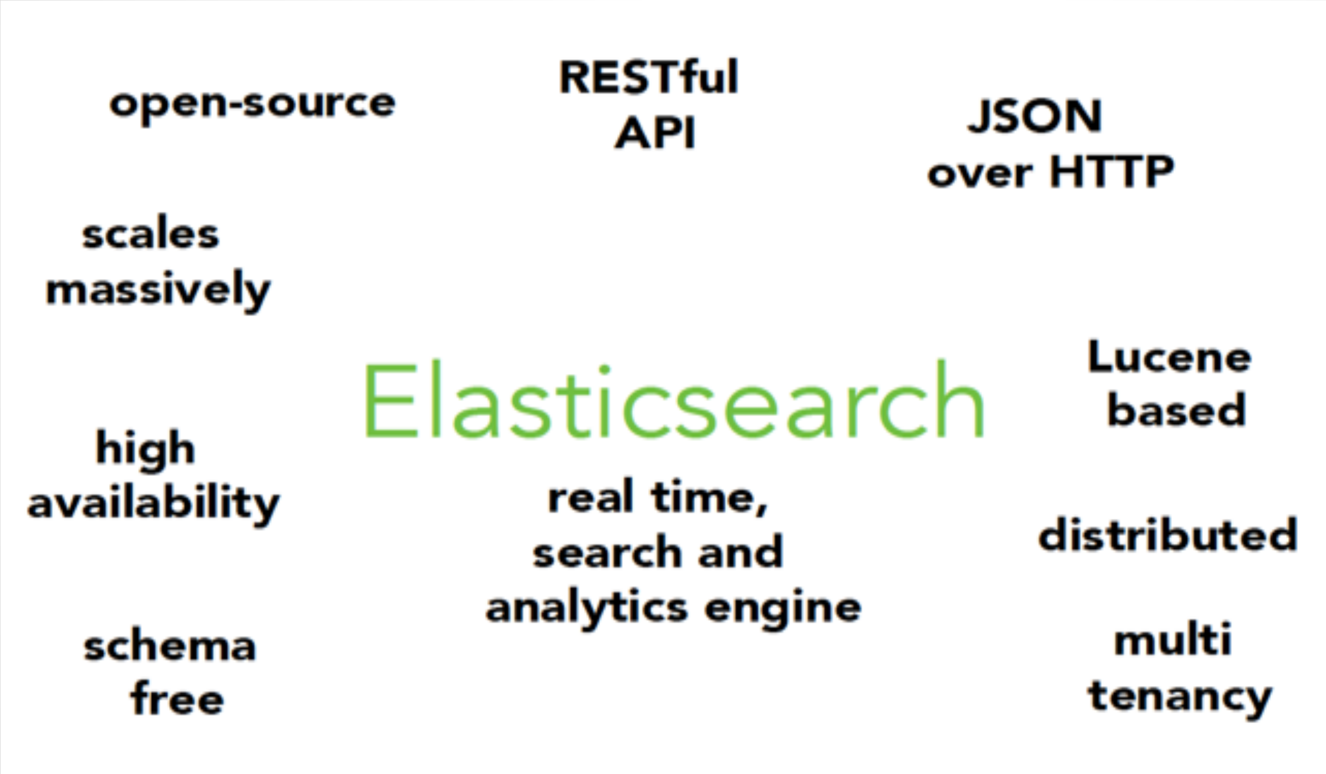
What is Elasticsearch?

❑ **Elasticsearch** is a great tool for document indexing and powerful full text search used for **Big Data Analytics**.

❑ It’s JSON based Domain Specific query Language is simple and powerful, making it the defacto standard for search integration in any web app.

❑ The combination of storage and querying / aggregation services makes elasticsearch really special and distant from the “document storage only” tools.

Elasticsearch (Cont..)



open-source

RESTful API

JSON over HTTP

scales massively

Elasticsearch

Lucene based

high availability

real time, search and analytics engine

distributed

schema free

multi tenancy

When to use Elasticsearch?

❑ Searching text and structured data (product search by name + properties)

❑ Data Integration

❑ Geo Search

❑ JSON document storage

Basic Concepts used in Elasticsearch

❑ Cluster: A set of Nodes (servers) that holds all the data

❑ Node: A single server that holds some data and participate on the cluster's indexing and querying

❑ Index: Forget SQL Indexes. Each ES Index is a set of Documents.

❑ Shards: A subset of Documents of an Index.

❑ Document: A JSON Object with some data.

Elasticsearch for Big Data Analytics

Three reasons:

❑ It is very easy to get a toy instance of Elasticsearch running with a small sample dataset.

❑ Elasticsearch JSON based query language is much easier to master than more complex systems like **Hadoop's MapReduce**.

❑ Application developers are more comfortable maintaining a second Elasticsearch integration over a completely new technology stack like **Hadoop**.

Elasticsearch Setup

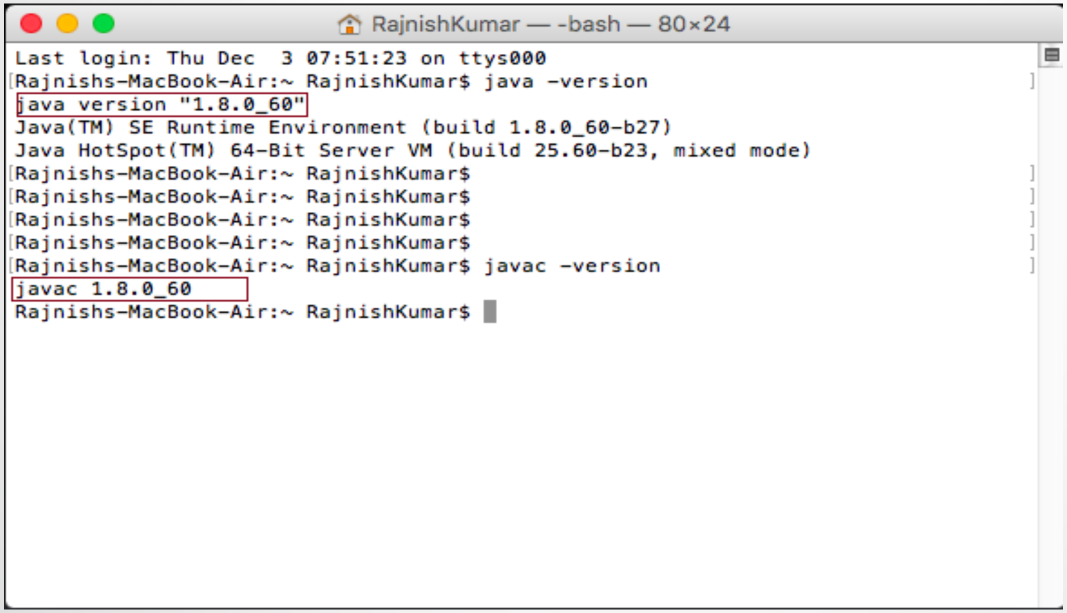
❑ Step 1: Check Java Version

❑ Step 2: Download from www.elasticsearch.co

❑ Step 3: Installation: $ bin/./elasticsearch

Elasticsearch Setup (Cont.)

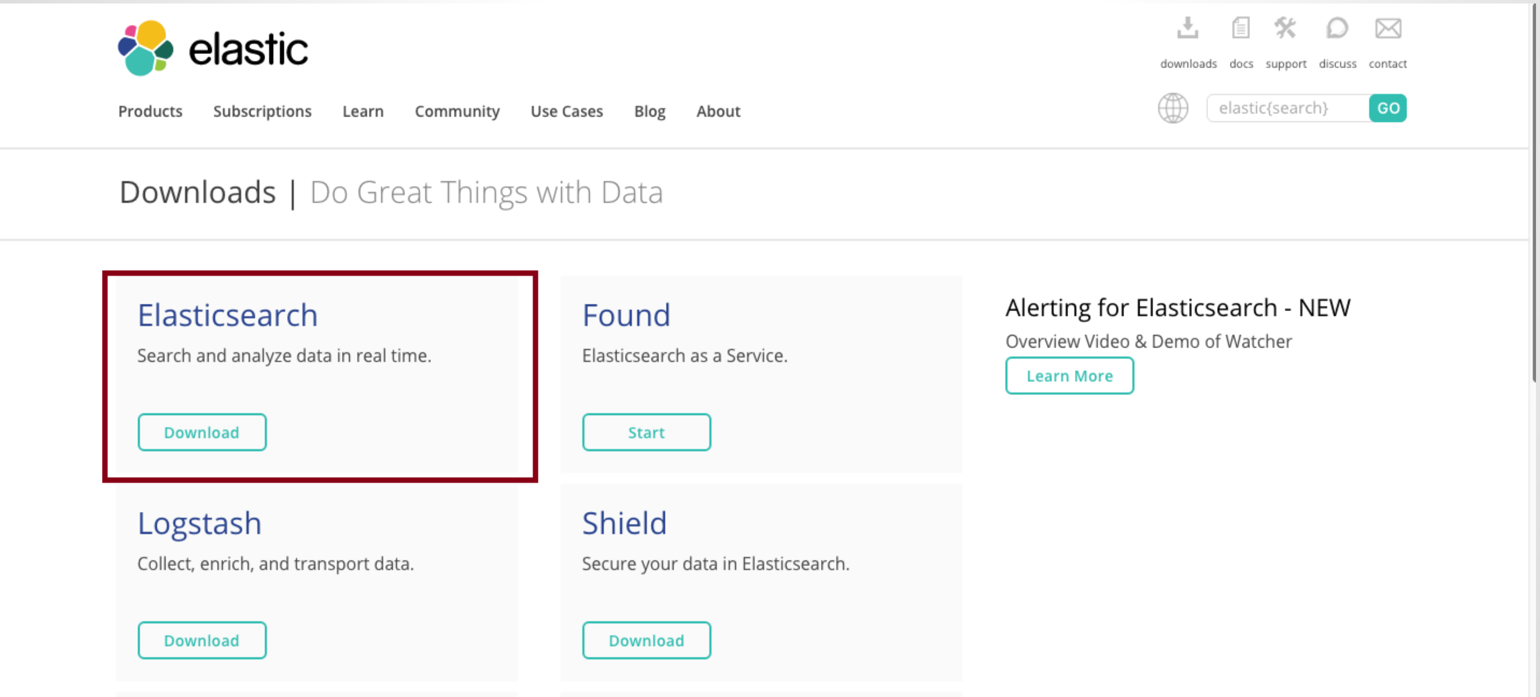
❑ Step 1: Check Java Version (Make sure java version should > 1.6)



```
Last login: Thu Dec  3 07:51:23 on ttys000
Rajnishs-MacBook-Air:~ RajnishKumar$ java -version
java version "1.8.0_60"
Java(TM) SE Runtime Environment (build 1.8.0_60-b27)
Java HotSpot(TM) 64-Bit Server VM (build 25.60-b23, mixed mode)
Rajnishs-MacBook-Air:~ RajnishKumar$
Rajnishs-MacBook-Air:~ RajnishKumar$
Rajnishs-MacBook-Air:~ RajnishKumar$
Rajnishs-MacBook-Air:~ RajnishKumar$
Rajnishs-MacBook-Air:~ RajnishKumar$ javac -version
javac 1.8.0_60
Rajnishs-MacBook-Air:~ RajnishKumar$
```

Elasticsearch Setup (Cont.)

- Step 2: Download from www.elasticsearch.co



Downloads | Do Great Things with Data

Elasticsearch

Search and analyze data in real time.

Download

Found

Elasticsearch as a Service.

Start

Alerting for Elasticsearch - NEW

Overview Video & Demo of Watcher

Learn More

Logstash

Collect, enrich, and transport data.

Download

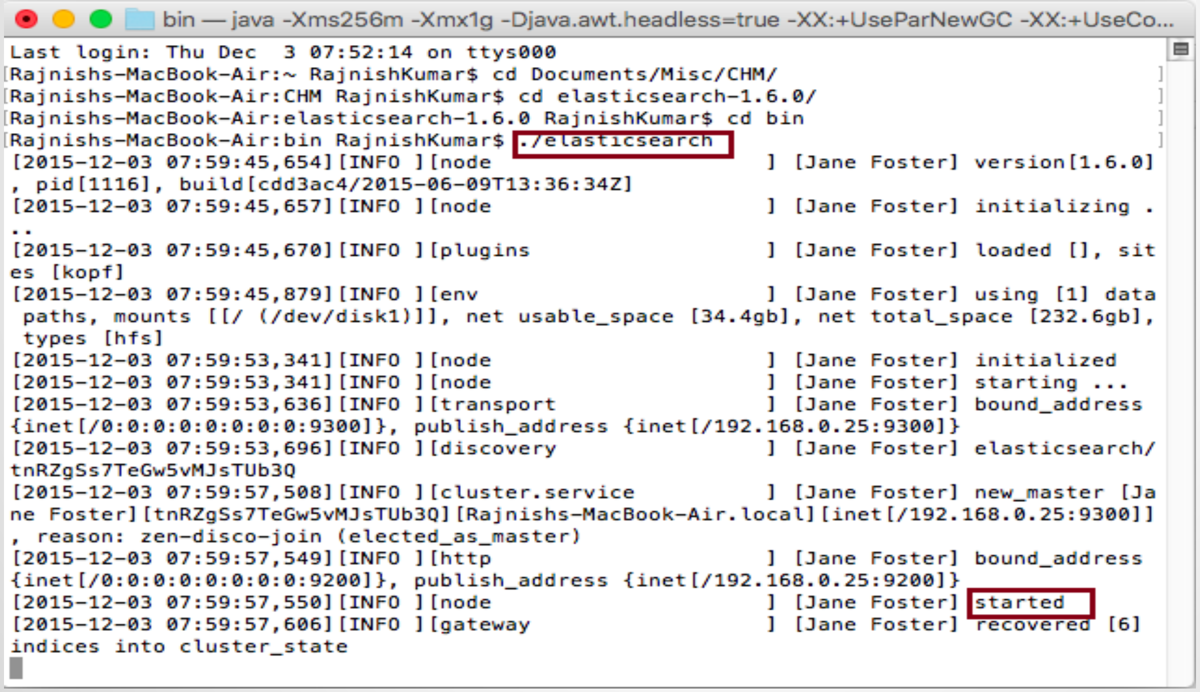
Shield

Secure your data in Elasticsearch.

Download

Elasticsearch Setup (Cont.)

❑ Step 3: Installation: $ bin/./elasticsearch


bin — java -Xms256m -Xmx1g -Djava.awt.headless=true -XX:+UseParNewGC -XX:+UseCo...

```
Last login: Thu Dec  3 07:52:14 on ttys000
[Rajnishs-MacBook-Air:~ RajnishKumar$ cd Documents/Misc/CHM/
[Rajnishs-MacBook-Air:CHM RajnishKumar$ cd elasticsearch-1.6.0/
[Rajnishs-MacBook-Air:elasticsearch-1.6.0 RajnishKumar$ cd bin
[Rajnishs-MacBook-Air:bin RajnishKumar$ ./elasticsearch
[2015-12-03 07:59:45,654][INFO ][node                     ] [Jane Foster] version[1.6.0]
, pid[1116], build[cdd3ac4/2015-06-09T13:36:34Z]
[2015-12-03 07:59:45,657][INFO ][node                     ] [Jane Foster] initializing .
..
[2015-12-03 07:59:45,670][INFO ][plugins                  ] [Jane Foster] loaded [], sit
es [kopf]
[2015-12-03 07:59:45,879][INFO ][env                      ] [Jane Foster] using [1] data
 paths, mounts [[/ (/dev/disk1)]], net usable_space [34.4gb], net total_space [232.6gb],
 types [hfs]
[2015-12-03 07:59:53,341][INFO ][node                     ] [Jane Foster] initialized
[2015-12-03 07:59:53,341][INFO ][node                     ] [Jane Foster] starting ...
[2015-12-03 07:59:53,636][INFO ][transport                ] [Jane Foster] bound_address
{inet[/0:0:0:0:0:0:0:0:9300]}, publish_address {inet[/192.168.0.25:9300]}
[2015-12-03 07:59:53,696][INFO ][discovery                ] [Jane Foster] elasticsearch/
tnRZgSs7TeGw5vMJsTUb3Q
[2015-12-03 07:59:57,508][INFO ][cluster.service          ] [Jane Foster] new_master [Ja
ne Foster][tnRZgSs7TeGw5vMJsTUb3Q][Rajnishs-MacBook-Air.local][inet[/192.168.0.25:9300]]
, reason: zen-disco-join (elected_as_master)
[2015-12-03 07:59:57,549][INFO ][http                     ] [Jane Foster] bound_address
{inet[/0:0:0:0:0:0:0:0:9200]}, publish_address {inet[/192.168.0.25:9200]}
[2015-12-03 07:59:57,550][INFO ][node                     ] [Jane Foster] started
[2015-12-03 07:59:57,606][INFO ][gateway                  ] [Jane Foster] recovered [6]
indices into cluster_state
```

ES Configuration

- Each Physical server has minimum 1 master node and 3 data nodes.
- Each node is configured to make use of pointer compression
- Lock all memory to prevent ES memory from being swapped cut
- Max file descriptors = 32k for user that run ES
- Node discovery using unicast

ES Index Configuration

❑ Mapping

❑ Templates

❑ Temporal indexes

❑ Aliases

Indexing and Querying

❑ Rest API: Elasticsearch uses a Rest API for searching and storing documents.

```
$ curl -XPUT 'http://localhost:9200’ -d '{
"author": “rajnish",
"tags": ["java", "web"],
"title": “cin",
"context": ”Pvt. Ltd."
}'
```

❑ Indexing: creates some internal data structures that makes the query perform better.

❑ Querying: Based on coordinates, numeric range queries that is used to aggregate data

Kopf

❑Kopf is a plugin which is used for elasticsearch.

❑It is generally used for large clusters.

❑It offers filters in Nodes by Name and type, indexes by Name and type.

❑Kopf provides access a lots of elasticsearch API features.

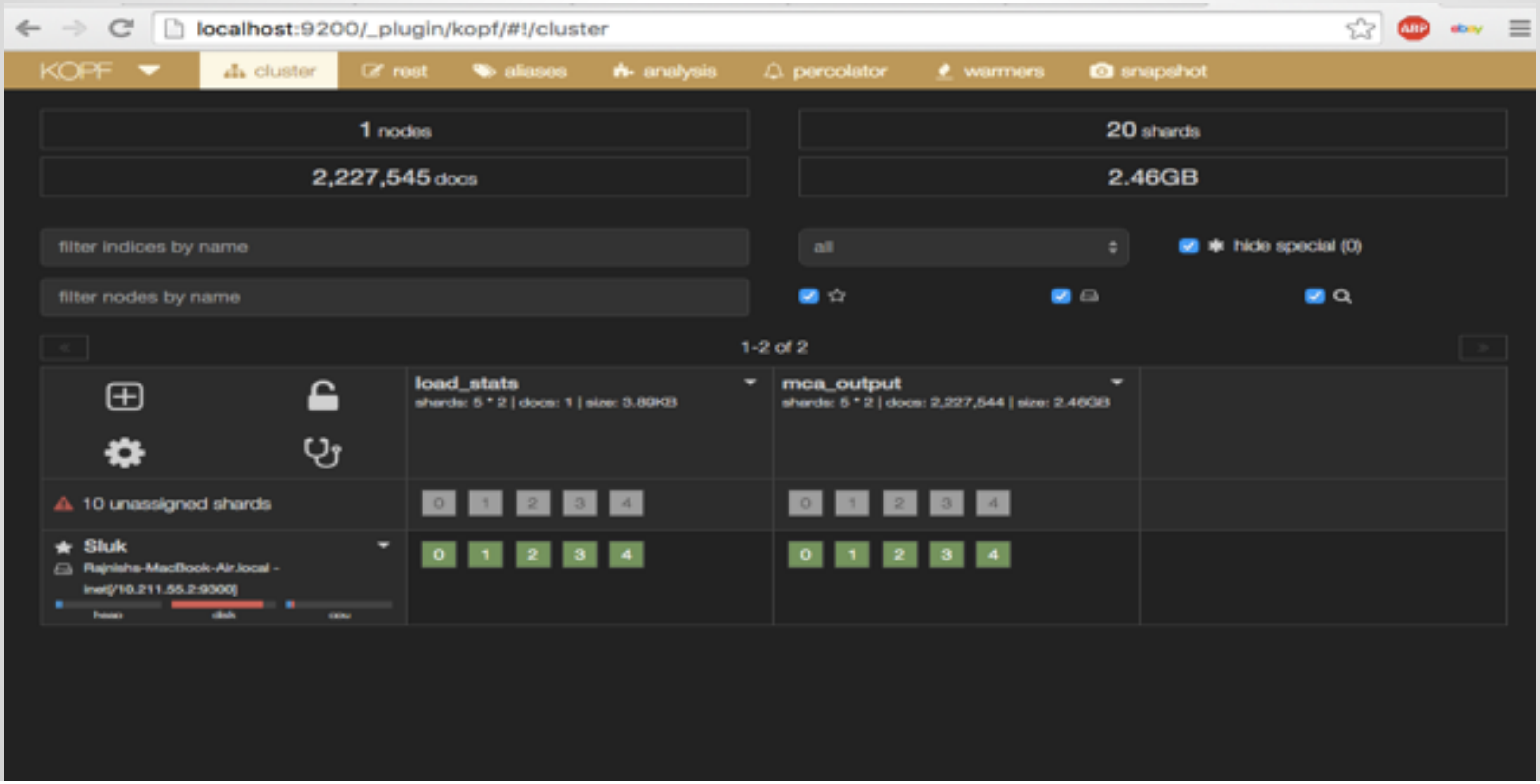
❑It also provides standardization and optimization.

❑It hides special indexes created internally by elasticsearch.

❑It offers a JSON Editor for RESTful services.

❑We can retrieve information from the json object.

Kopf (Cont..)



1 nodes

20 shards

2,227,545 docs

2.46GB

1-2 of 2

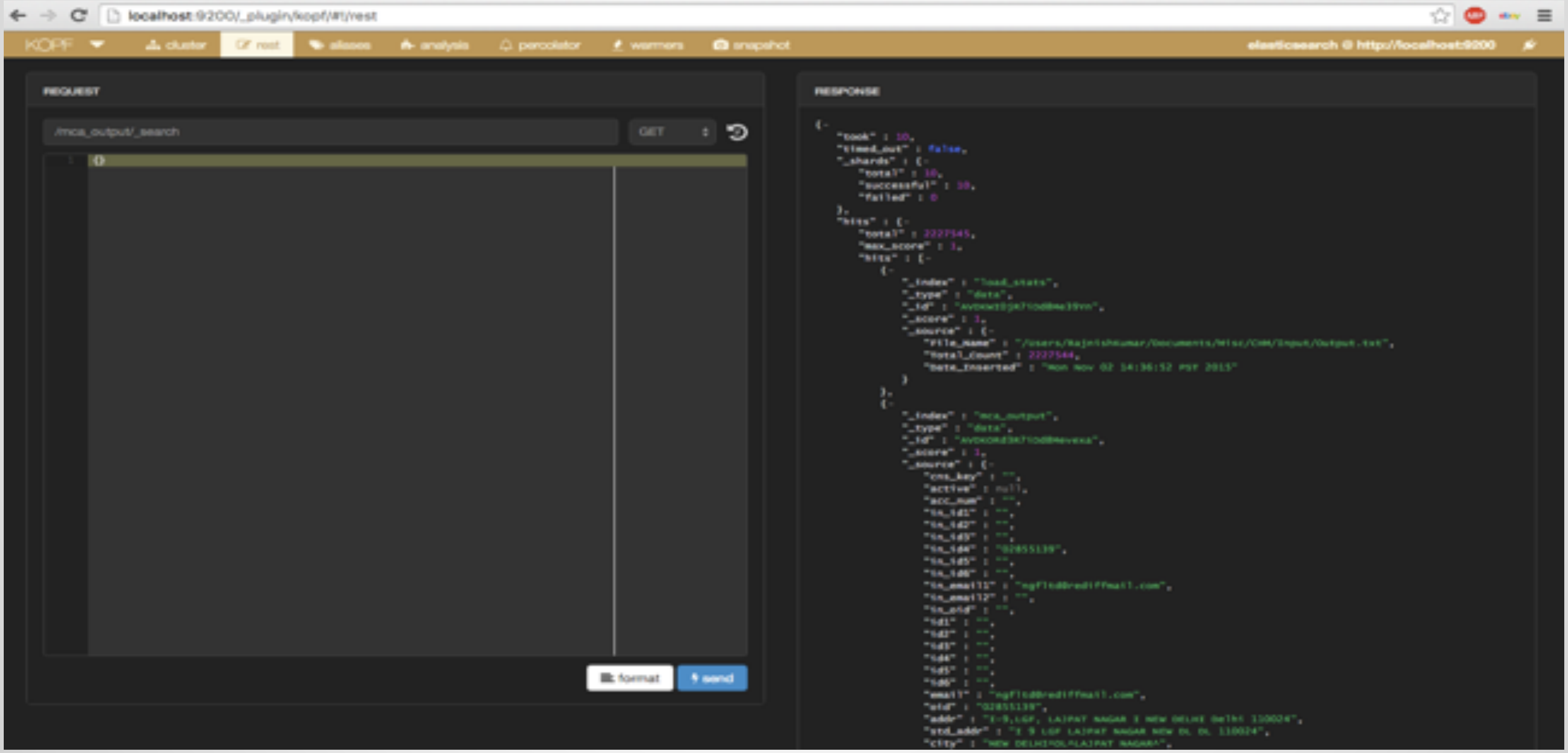
load_stats
shards: 5 * 2 | docs: 1 | size: 3.89KB

mca_output
shards: 5 * 2 | docs: 2,227,544 | size: 2.46GB

10 unassigned shards: 0 1 2 3 4 | 0 1 2 3 4

Sluk
Rajnishs-MacBook-Air.local - inet[/10.211.55.2:9300]: 0 1 2 3 4 | 0 1 2 3 4

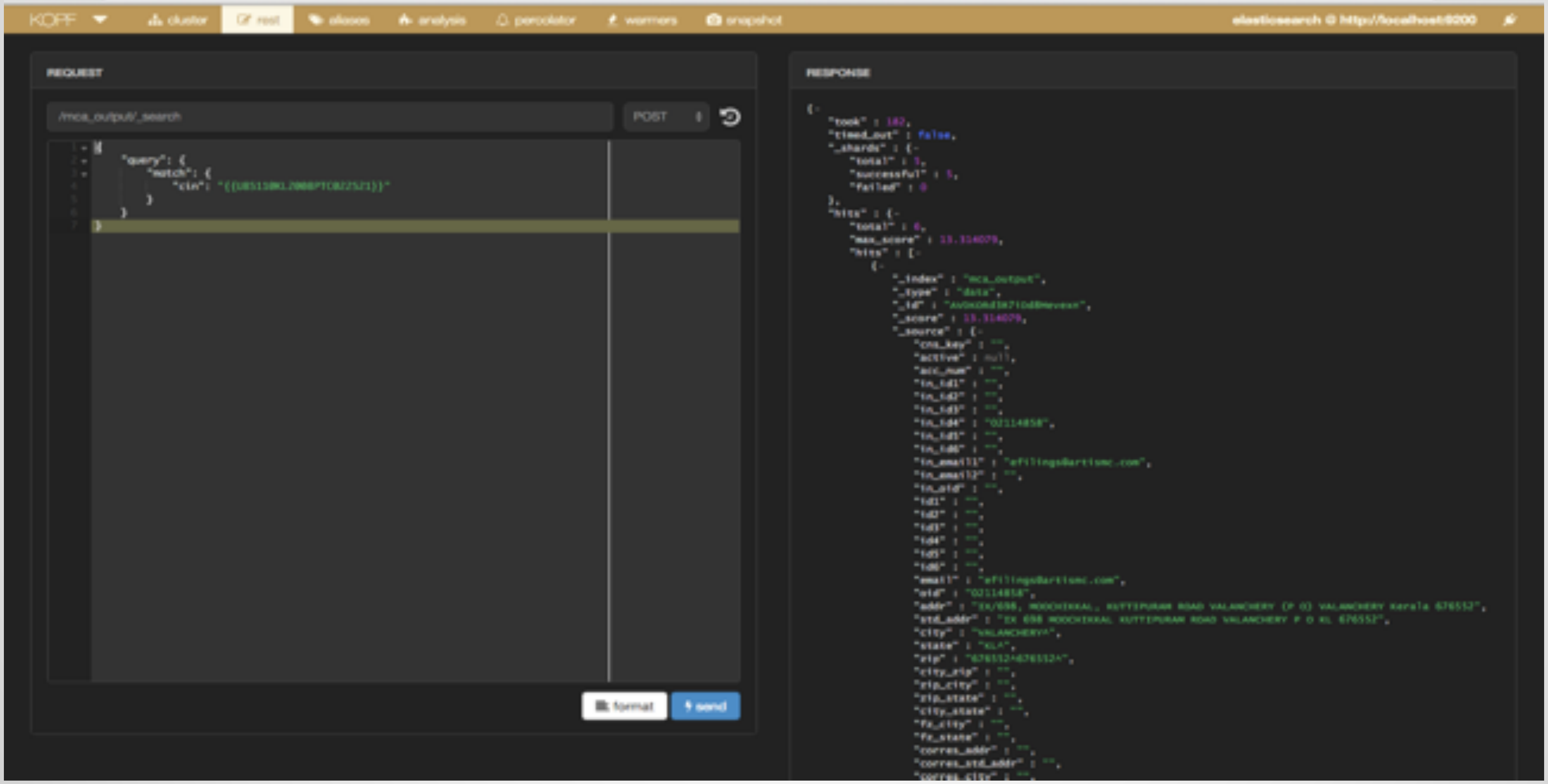
Kopf (Cont..)



```
{}
```

```
{
    "took" : 10,
    "timed_out" : false,
    "_shards" : {
        "total" : 10,
        "successful" : 10,
        "failed" : 0
    },
    "hits" : {
        "total" : 2227545,
        "max_score" : 1,
        "hits" : [
            {
                "_index" : "load_stats",
                "_type" : "data",
                "_id" : "AVDKWtDjR7iOdBMe39Vn",
                "_score" : 1,
                "_source" : {
                    "File_Name" : "/Users/RajnishKumar/Documents/Misc/CHM/Input/Output.txt",
                    "Total_Count" : 2227544,
                    "Date_Inserted" : "Mon Nov 02 14:36:52 PST 2015"
                }
            },
            {
                "_index" : "mca_output",
                "_type" : "data",
                "_id" : "AVDKORd3R7iOdBMevexa",
                "_score" : 1,
                "_source" : {
                    "cns_key" : "",
                    "active" : null,
                    "acc_num" : "",
                    "in_id1" : "",
                    "in_id2" : "",
                    "in_id3" : "",
                    "in_id4" : "02855139",
                    "in_id5" : "",
                    "in_id6" : "",
                    "in_email1" : "ngfltd@rediffmail.com",
                    "in_email2" : "",
                    "in_oid" : "",
                    "id1" : "",
                    "id2" : "",
                    "id3" : "",
                    "id4" : "",
                    "id5" : "",
                    "id6" : "",
                    "email1" : "ngfltd@rediffmail.com",
                    "oid" : "02855139",
                    "addr" : "I-9,LGF, LAJPAT NAGAR I NEW DELHI Delhi 110024",
                    "std_addr" : "I 9 LGF LAJPAT NAGAR NEW DL DL 110024",
                    "city" : "NEW DELHI^DL^LAJPAT NAGAR^",
```

Kopf (Cont..)



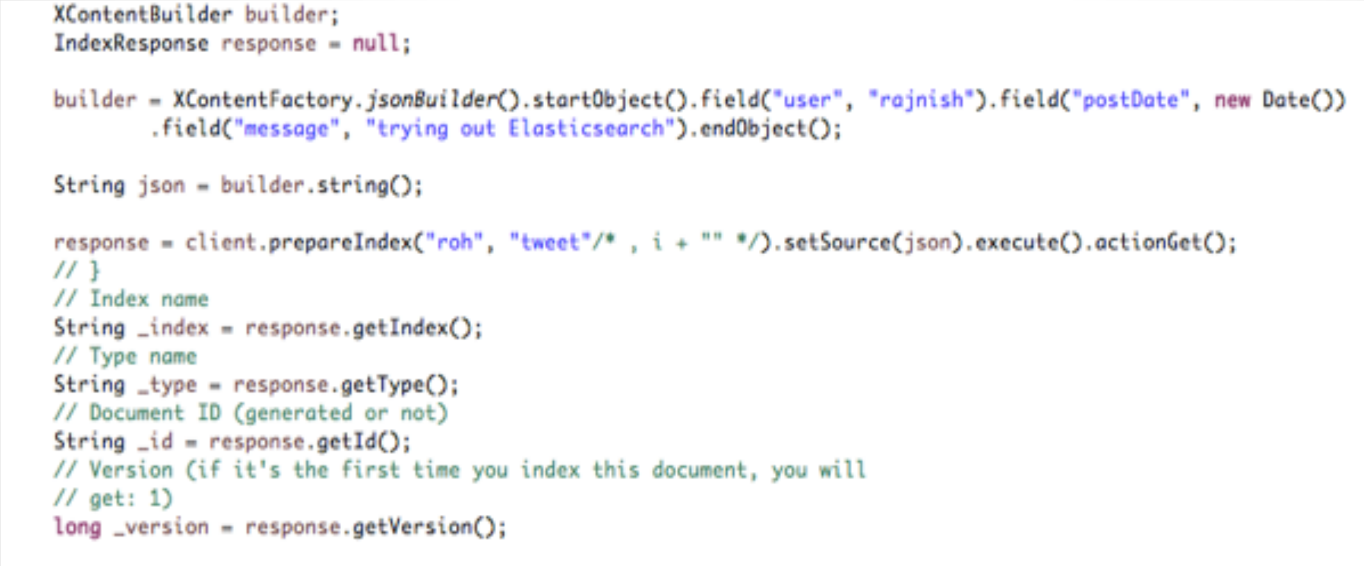
/mca_output/_search

```
{
    "query": {
        "match": {
            "cin": "{{U85110KL2008PTC022521}}"
        }
    }
}
```

```
{-
    "took" : 182,
    "timed_out" : false,
    "_shards" : {-
        "total" : 5,
        "successful" : 5,
        "failed" : 0
    },
    "hits" : {-
        "total" : 6,
        "max_score" : 13.314079,
        "hits" : [-
            {-
                "_index" : "mca_output",
                "_type" : "data",
                "_id" : "AVOkORdI87IOd8Mevexn",
                "_score" : 13.314079,
                "_source" : {-
                    "cns_key" : "",
                    "active" : null,
                    "acc_num" : "",
                    "in_id1" : "",
                    "in_id2" : "",
                    "in_id3" : "",
                    "in_id4" : "02114858",
                    "in_id5" : "",
                    "in_id6" : "",
                    "in_email1" : "efilings@artismc.com",
                    "in_email2" : "",
                    "in_std" : "",
                    "id1" : "",
                    "id2" : "",
                    "id3" : "",
                    "id4" : "",
                    "id5" : "",
                    "id6" : "",
                    "email1" : "efilings@artismc.com",
                    "std" : "02114858",
                    "addr" : "IX/698, MOOCHIKKAL, KUTTIPURAM ROAD VALANCHERY (P O) VALANCHERY Kerala 676552",
                    "std_addr" : "IX 698 MOOCHIKKAL KUTTIPURAM ROAD VALANCHERY P O KL 676552",
                    "city" : "VALANCHERY^",
                    "state" : "KLA",
                    "zip" : "676552^676552^",
                    "city_zip" : "",
                    "zip_city" : "",
                    "zip_state" : "",
                    "city_state" : "",
                    "fz_city" : "",
                    "fz_state" : "",
                    "corres_addr" : "",
                    "corres_std_addr" : "",
                    "corres_city" : "",
```

creating index and establishing connection

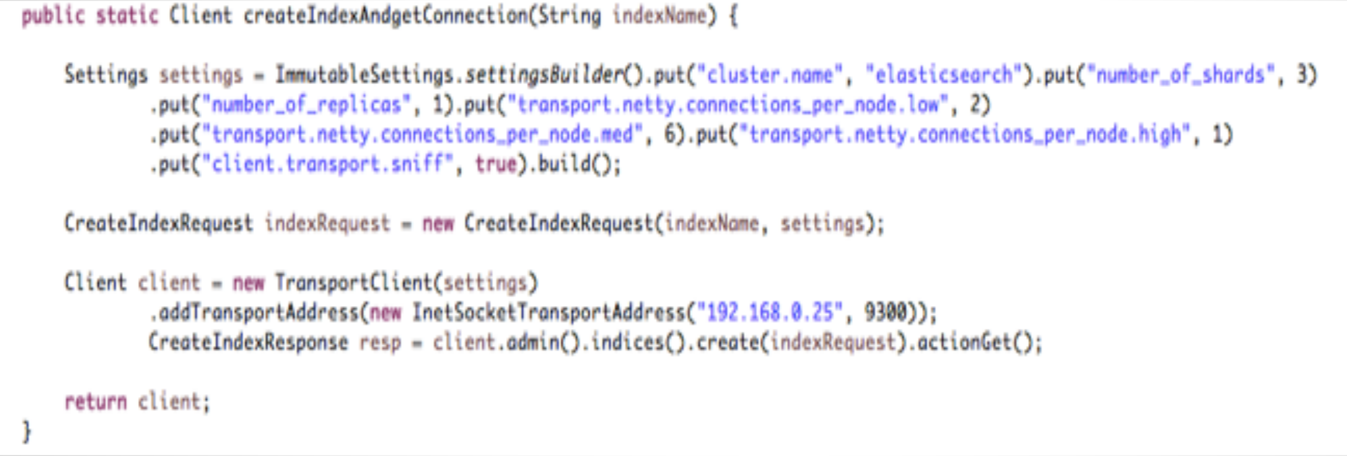
Building index



```
XContentBuilder builder;
IndexResponse response = null;

builder = XContentFactory.jsonBuilder().startObject().field("user", "rajnish").field("postDate", new Date())
        .field("message", "trying out Elasticsearch").endObject();

String json = builder.string();

response = client.prepareIndex("roh", "tweet"/* , i + "" */).setSource(json).execute().actionGet();
// }
// Index name
String _index = response.getIndex();
// Type name
String _type = response.getType();
// Document ID (generated or not)
String _id = response.getId();
// Version (if it's the first time you index this document, you will
// get: 1)
long _version = response.getVersion();
```

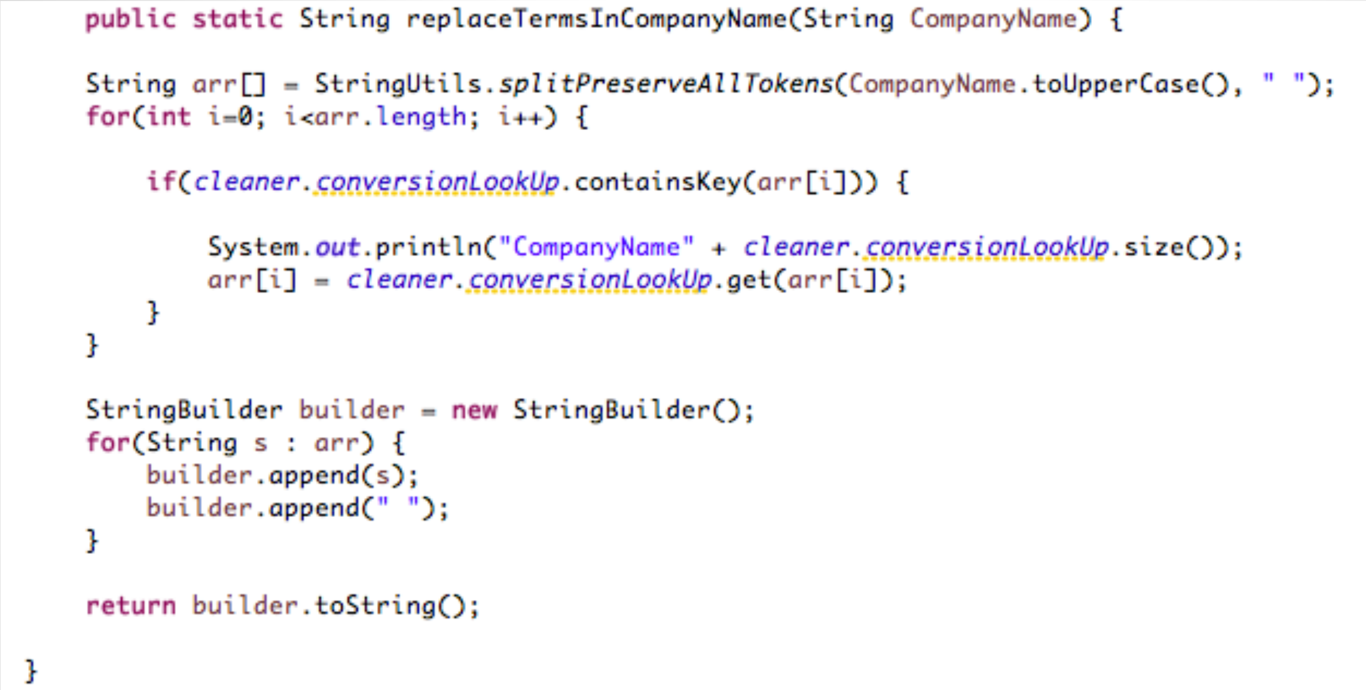
creating index and establishing connection (Cont..)

Creating Index and setting with connection:



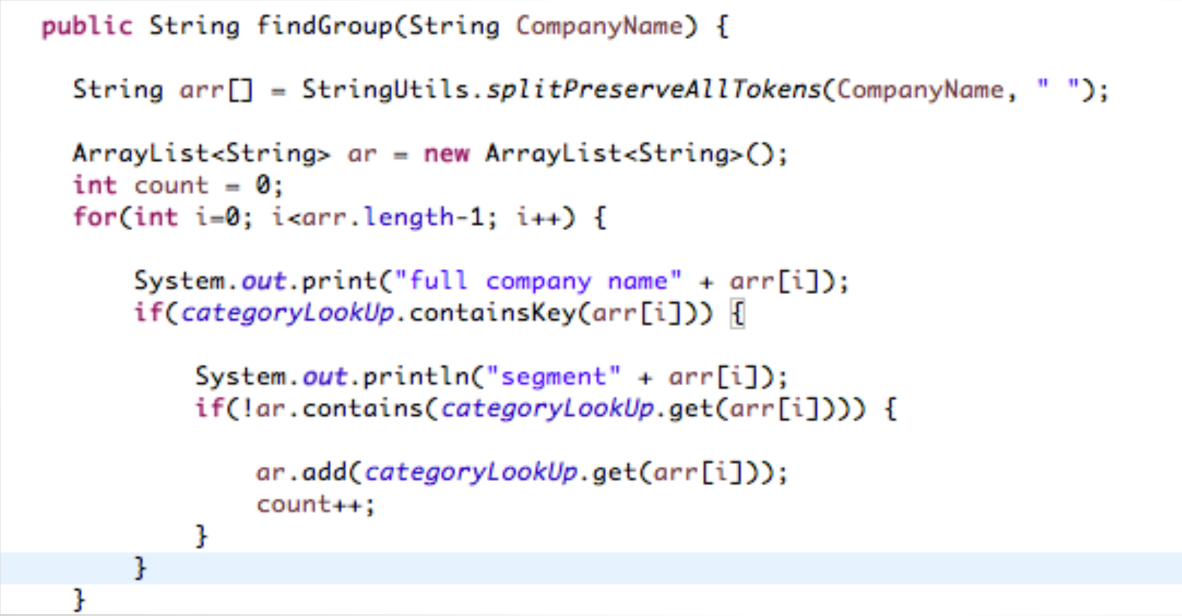
```
public static Client createIndexAndgetConnection(String indexName) {

    Settings settings = ImmutableSettings.settingsBuilder().put("cluster.name", "elasticsearch").put("number_of_shards", 3)
            .put("number_of_replicas", 1).put("transport.netty.connections_per_node.low", 2)
            .put("transport.netty.connections_per_node.med", 6).put("transport.netty.connections_per_node.high", 1)
            .put("client.transport.sniff", true).build();

    CreateIndexRequest indexRequest = new CreateIndexRequest(indexName, settings);

    Client client = new TransportClient(settings)
            .addTransportAddress(new InetSocketTransportAddress("192.168.0.25", 9300));
            CreateIndexResponse resp = client.admin().indices().create(indexRequest).actionGet();

    return client;
}
```

Replacement Standardization



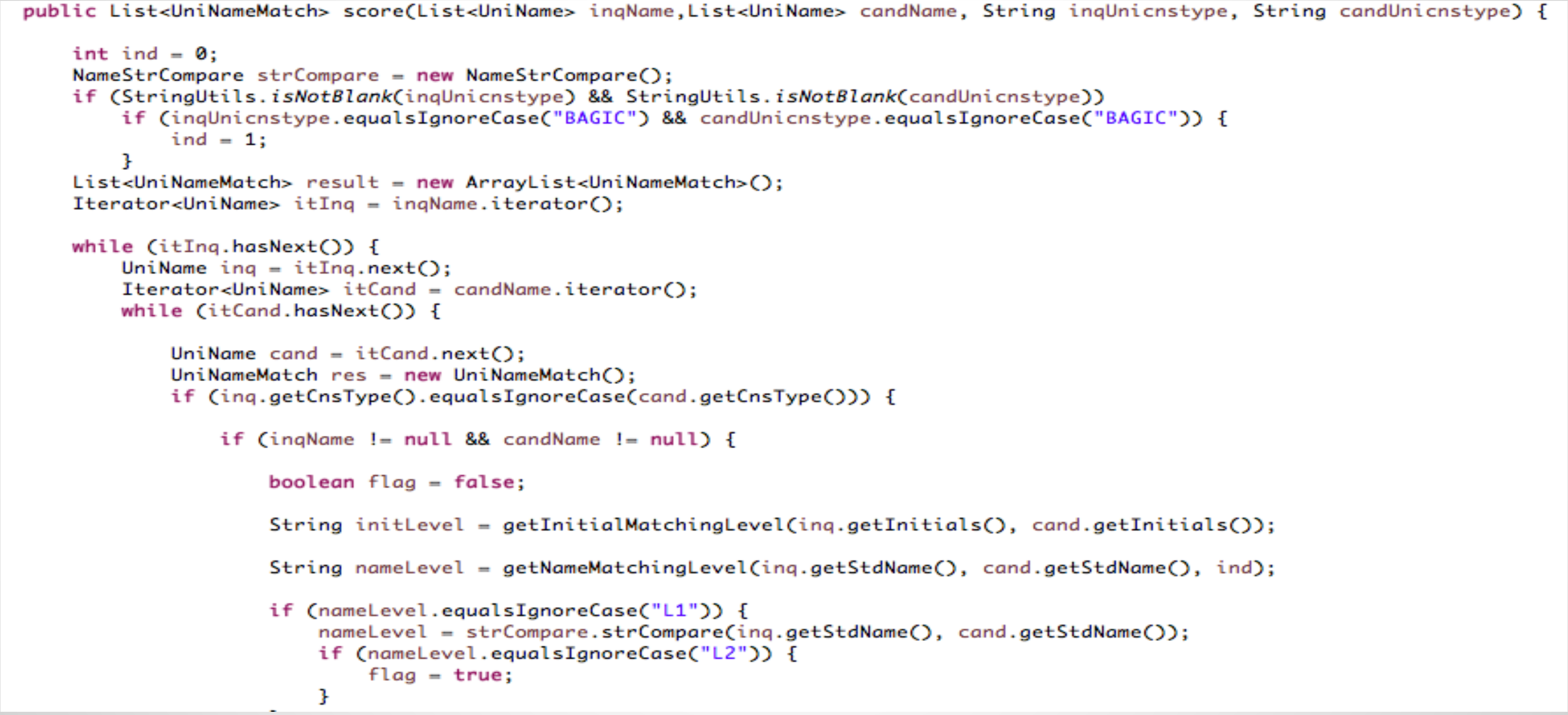
```
    public static String replaceTermsInCompanyName(String CompanyName) {

    String arr[] = StringUtils.splitPreserveAllTokens(CompanyName.toUpperCase(), " ");
    for(int i=0; i<arr.length; i++) {

        if(cleaner.conversionLookUp.containsKey(arr[i])) {

            System.out.println("CompanyName" + cleaner.conversionLookUp.size());
            arr[i] = cleaner.conversionLookUp.get(arr[i]);
        }
    }

    StringBuilder builder = new StringBuilder();
    for(String s : arr) {
        builder.append(s);
        builder.append(" ");
    }

    return builder.toString();

}
```

Group Standardization



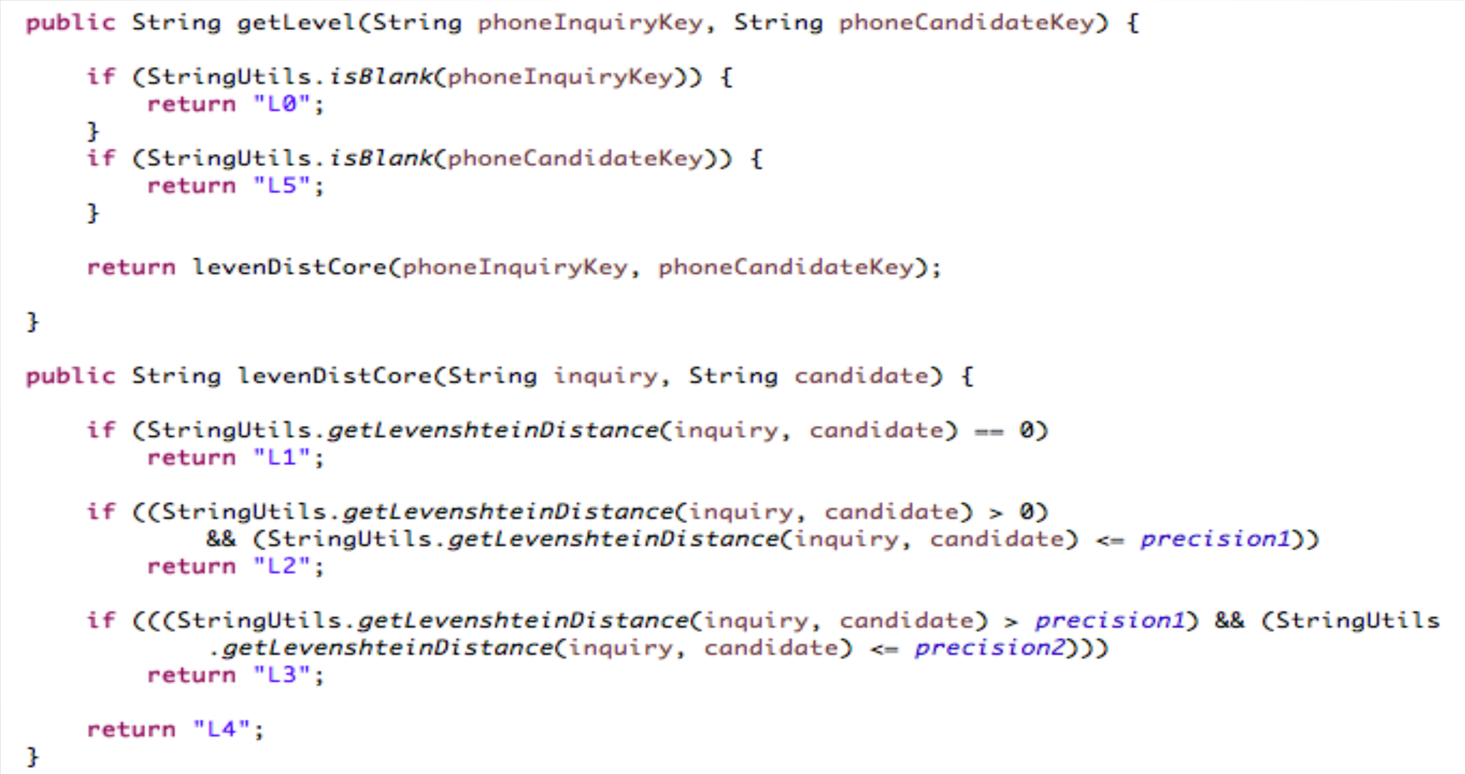
```
public String findGroup(String CompanyName) {

  String arr[] = StringUtils.splitPreserveAllTokens(CompanyName, " ");

  ArrayList<String> ar = new ArrayList<String>();
  int count = 0;
  for(int i=0; i<arr.length-1; i++) {

      System.out.print("full company name" + arr[i]);
      if(categoryLookUp.containsKey(arr[i])) {

          System.out.println("segment" + arr[i]);
          if(!ar.contains(categoryLookUp.get(arr[i]))) {

              ar.add(categoryLookUp.get(arr[i]));
              count++;
          }
      }
  }
```

Name Standardization



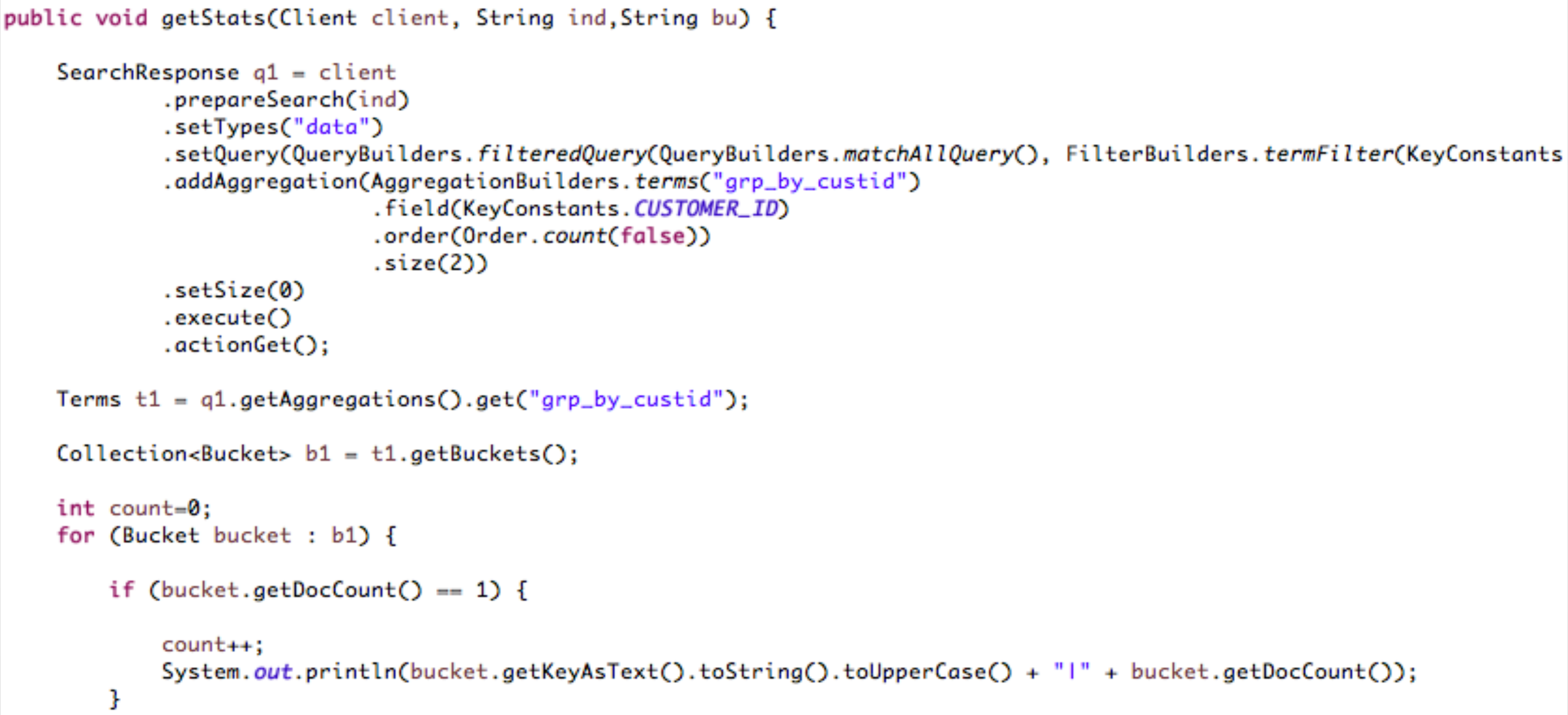
```
public List<UniNameMatch> score(List<UniName> inqName,List<UniName> candName, String inqUnicnstype, String candUnicnstype) {

    int ind = 0;
    NameStrCompare strCompare = new NameStrCompare();
    if (StringUtils.isNotBlank(inqUnicnstype) && StringUtils.isNotBlank(candUnicnstype))
        if (inqUnicnstype.equalsIgnoreCase("BAGIC") && candUnicnstype.equalsIgnoreCase("BAGIC")) {
            ind = 1;
        }
    List<UniNameMatch> result = new ArrayList<UniNameMatch>();
    Iterator<UniName> itInq = inqName.iterator();

    while (itInq.hasNext()) {
        UniName inq = itInq.next();
        Iterator<UniName> itCand = candName.iterator();
        while (itCand.hasNext()) {

            UniName cand = itCand.next();
            UniNameMatch res = new UniNameMatch();
            if (inq.getCnsType().equalsIgnoreCase(cand.getCnsType())) {

                if (inqName != null && candName != null) {

                    boolean flag = false;

                    String initLevel = getInitialMatchingLevel(inq.getInitials(), cand.getInitials());

                    String nameLevel = getNameMatchingLevel(inq.getStdName(), cand.getStdName(), ind);

                    if (nameLevel.equalsIgnoreCase("L1")) {
                        nameLevel = strCompare.strCompare(inq.getStdName(), cand.getStdName());
                        if (nameLevel.equalsIgnoreCase("L2")) {
                            flag = true;
                        }
```

Phone Standardization



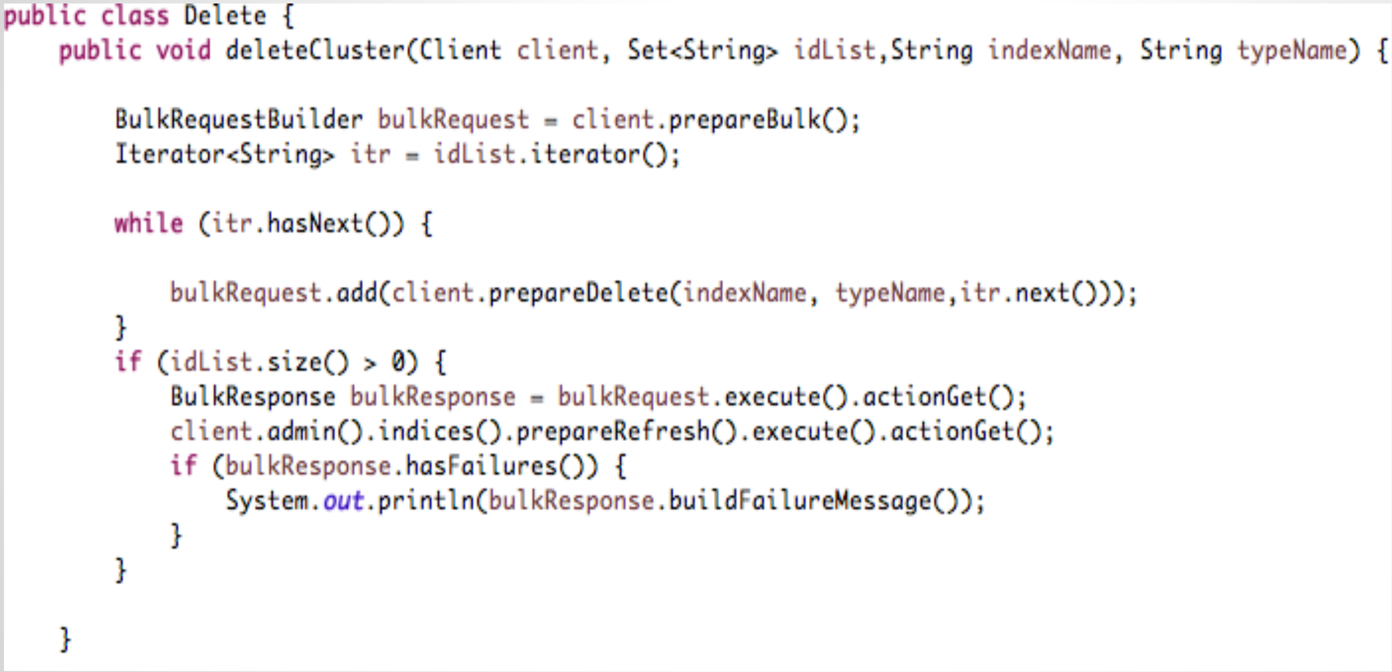
```
public String getLevel(String phoneInquiryKey, String phoneCandidateKey) {

    if (StringUtils.isBlank(phoneInquiryKey)) {
        return "L0";
    }
    if (StringUtils.isBlank(phoneCandidateKey)) {
        return "L5";
    }

    return levenDistCore(phoneInquiryKey, phoneCandidateKey);

}

public String levenDistCore(String inquiry, String candidate) {

    if (StringUtils.getLevenshteinDistance(inquiry, candidate) == 0)
        return "L1";

    if ((StringUtils.getLevenshteinDistance(inquiry, candidate) > 0)
            && (StringUtils.getLevenshteinDistance(inquiry, candidate) <= precision1))
        return "L2";

    if (((StringUtils.getLevenshteinDistance(inquiry, candidate) > precision1) && (StringUtils
            .getLevenshteinDistance(inquiry, candidate) <= precision2)))
        return "L3";

    return "L4";
}
```

Duplicate Standardization



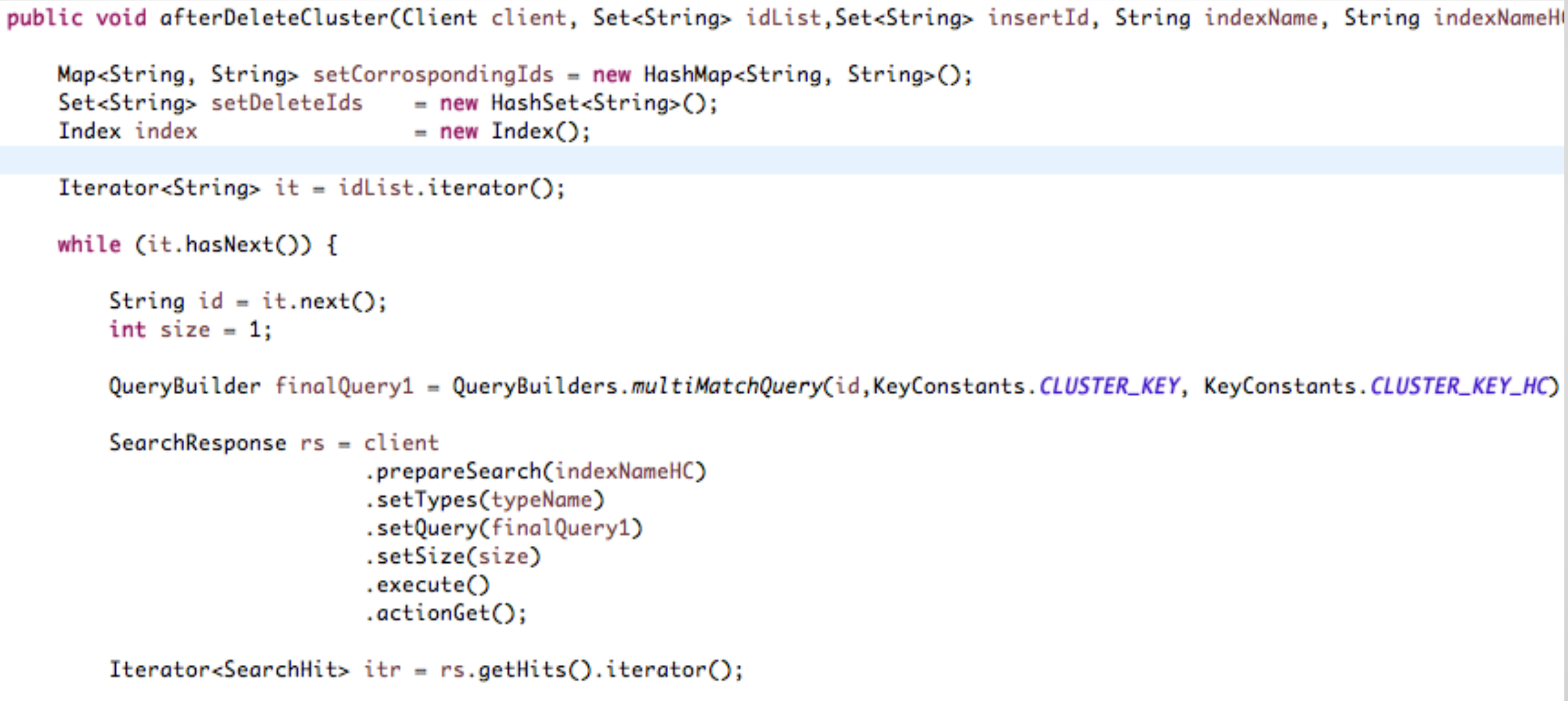
```
public void getStats(Client client, String ind,String bu) {

    SearchResponse q1 = client
            .prepareSearch(ind)
            .setTypes("data")
            .setQuery(QueryBuilders.filteredQuery(QueryBuilders.matchAllQuery(), FilterBuilders.termFilter(KeyConstants
            .addAggregation(AggregationBuilders.terms("grp_by_custid")
                            .field(KeyConstants.CUSTOMER_ID)
                            .order(Order.count(false))
                            .size(2))
            .setSize(0)
            .execute()
            .actionGet();

    Terms t1 = q1.getAggregations().get("grp_by_custid");

    Collection<Bucket> b1 = t1.getBuckets();

    int count=0;
    for (Bucket bucket : b1) {

        if (bucket.getDocCount() == 1) {

            count++;
            System.out.println(bucket.getKeyAsText().toString().toUpperCase() + "|" + bucket.getDocCount());
        }
```

Delete Standardization



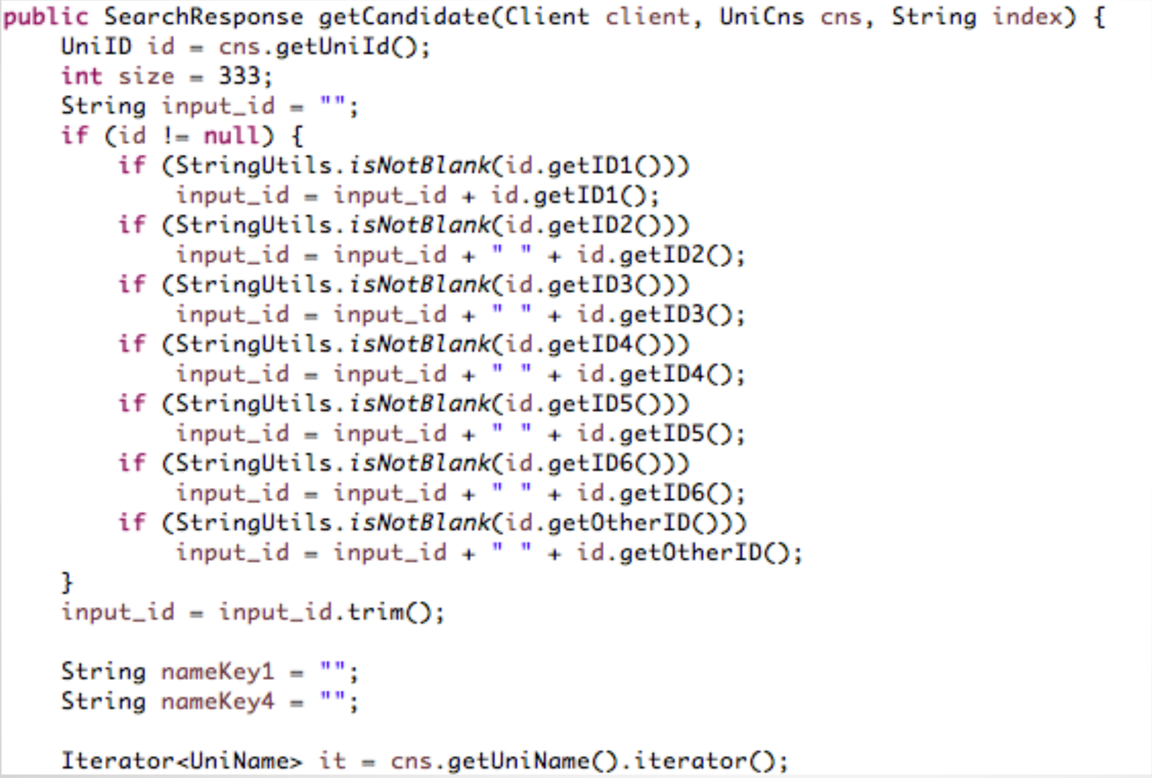
```java
public class Delete {
    public void deleteCluster(Client client, Set<String> idList,String indexName, String typeName) {

        BulkRequestBuilder bulkRequest = client.prepareBulk();
        Iterator<String> itr = idList.iterator();

        while (itr.hasNext()) {

            bulkRequest.add(client.prepareDelete(indexName, typeName,itr.next()));
        }
        if (idList.size() > 0) {
            BulkResponse bulkResponse = bulkRequest.execute().actionGet();
            client.admin().indices().prepareRefresh().execute().actionGet();
            if (bulkResponse.hasFailures()) {
                System.out.println(bulkResponse.buildFailureMessage());
            }
        }

    }
```

Cluster Formation after Delete



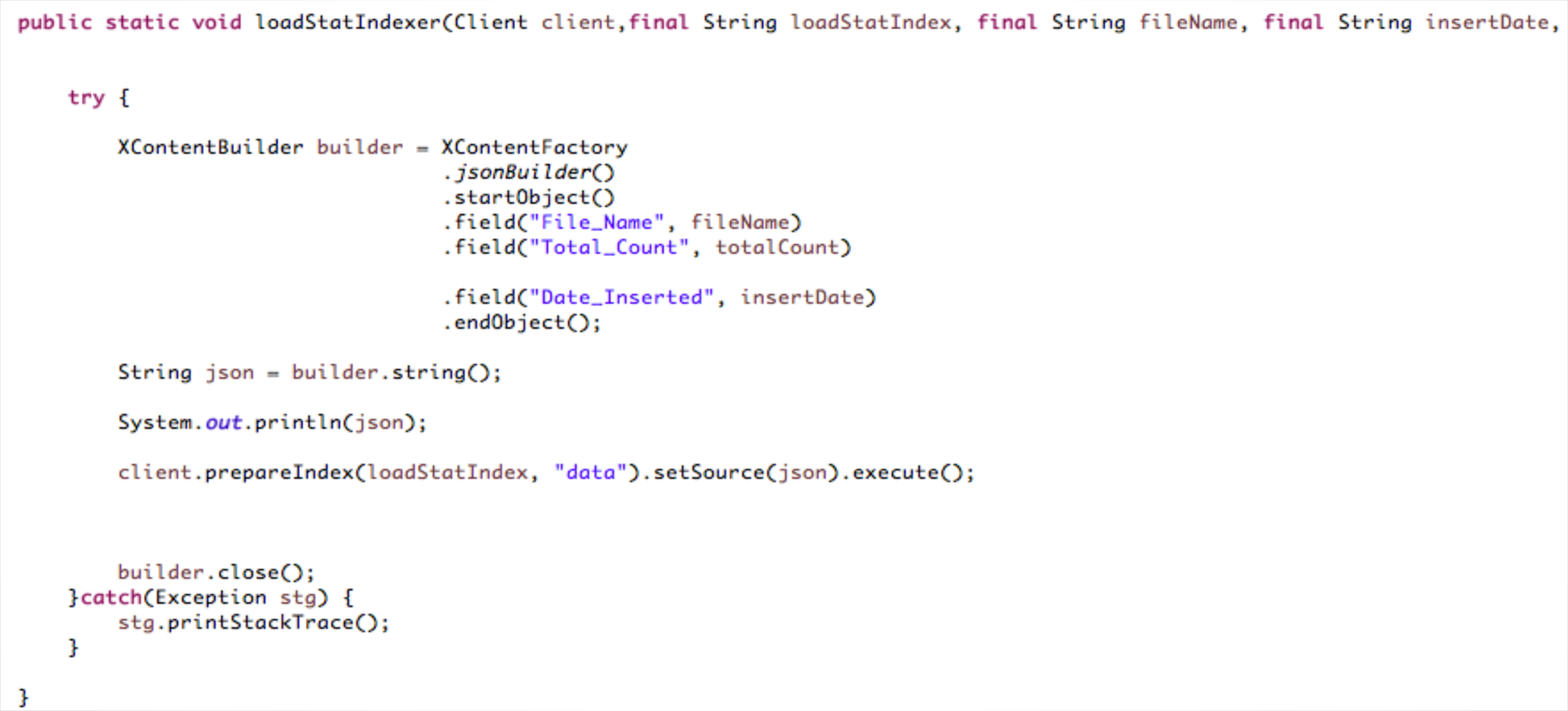
```
public void afterDeleteCluster(Client client, Set<String> idList,Set<String> insertId, String indexName, String indexNameH

    Map<String, String> setCorrospondingIds = new HashMap<String, String>();
    Set<String> setDeleteIds     = new HashSet<String>();
    Index index                  = new Index();

    Iterator<String> it = idList.iterator();

    while (it.hasNext()) {

        String id = it.next();
        int size = 1;

        QueryBuilder finalQuery1 = QueryBuilders.multiMatchQuery(id,KeyConstants.CLUSTER_KEY, KeyConstants.CLUSTER_KEY_HC)

        SearchResponse rs = client
                            .prepareSearch(indexNameHC)
                            .setTypes(typeName)
                            .setQuery(finalQuery1)
                            .setSize(size)
                            .execute()
                            .actionGet();

        Iterator<SearchHit> itr = rs.getHits().iterator();
```

Fetching candidate Value



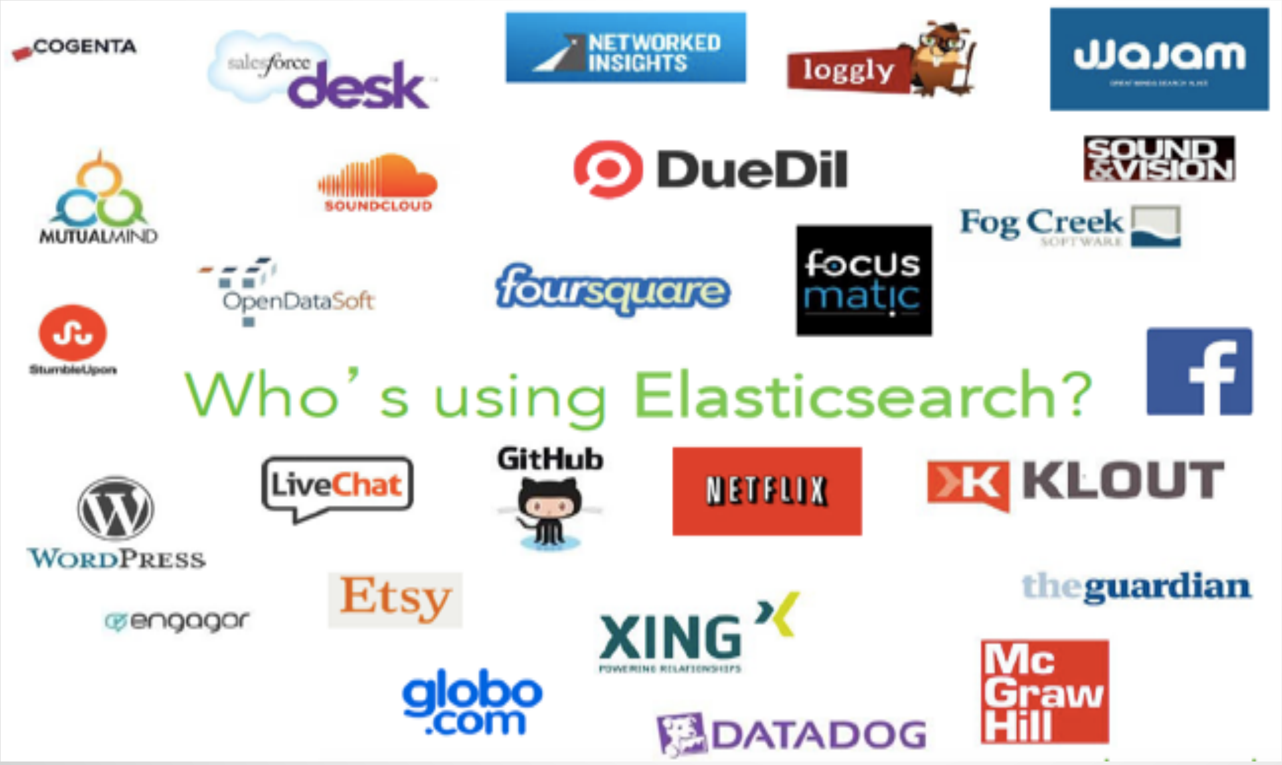
```
public SearchResponse getCandidate(Client client, UniCns cns, String index) {
    UniID id = cns.getUniId();
    int size = 333;
    String input_id = "";
    if (id != null) {
        if (StringUtils.isNotBlank(id.getID1()))
            input_id = input_id + id.getID1();
        if (StringUtils.isNotBlank(id.getID2()))
            input_id = input_id + " " + id.getID2();
        if (StringUtils.isNotBlank(id.getID3()))
            input_id = input_id + " " + id.getID3();
        if (StringUtils.isNotBlank(id.getID4()))
            input_id = input_id + " " + id.getID4();
        if (StringUtils.isNotBlank(id.getID5()))
            input_id = input_id + " " + id.getID5();
        if (StringUtils.isNotBlank(id.getID6()))
            input_id = input_id + " " + id.getID6();
        if (StringUtils.isNotBlank(id.getOtherID()))
            input_id = input_id + " " + id.getOtherID();
    }
    input_id = input_id.trim();

    String nameKey1 = "";
    String nameKey4 = "";

    Iterator<UniName> it = cns.getUniName().iterator();
```

Loading Index in Cluster



```
public static void loadStatIndexer(Client client,final String loadStatIndex, final String fileName, final String insertDate,


    try {

        XContentBuilder builder = XContentFactory
                                  .jsonBuilder()
                                  .startObject()
                                  .field("File_Name", fileName)
                                  .field("Total_Count", totalCount)

                                  .field("Date_Inserted", insertDate)
                                  .endObject();

        String json = builder.string();

        System.out.println(json);

        client.prepareIndex(loadStatIndex, "data").setSource(json).execute();



        builder.close();
    }catch(Exception stg) {
        stg.printStackTrace();
    }

}
```

Who is using Elasticsearch?



Advantages of Elasticsearch

❏ Fast, Incisive Search against Large Volumes of Data

❏ Indexing Documents to the repository

❏ Denormalized Document Storage

❏ Broadly Distributed and Highly Scalable

Thank You