import java.io.BufferedReader;  
import java.io.DataInputStream;  
import java.io.FileInputStream;  
import java.io.IOException;  
import java.io.InputStreamReader;  
import java.net.InetAddress;  
import org.elasticsearch.action.bulk.BulkRequestBuilder;  
import org.elasticsearch.action.bulk.BulkResponse;  
import org.elasticsearch.client.Client;  
import org.elasticsearch.client.transport.TransportClient;  
import org.elasticsearch.common.settings.Settings;  
import org.elasticsearch.common.transport.InetSocketTransportAddress;  
import org.json.simple.JSONObject;  
import org.json.simple.parser.JSONParser;  
import org.json.simple.parser.ParseException;  
public class IndexData {

public static void main(String[] args) throws ParseException, IOException {

// configuration setting  
Settings settings = Settings.settingsBuilder()  
.put(“cluster.name”, “test-cluster”).build();  
TransportClient client = TransportClient.builder().settings(settings).build();

String hostname = “<Your-Hostname>”;  
int port = 9300;  
client.addTransportAddress(new InetSocketTransportAddress(InetAddress.getByName(hostname),port));

// bulk API  
BulkRequestBuilder bulkBuilder = client.prepareBulk();

long bulkBuilderLength = 0;  
String readLine = “”;  
String index = “testindex”;  
String type = “testtype”;  
String \_id = null;

BufferedReader br = new BufferedReader(new InputStreamReader(new DataInputStream(new FileInputStream(“accounts.json”))));  
JSONParser parser = new JSONParser();

while((readLine = br.readLine()) != null){  
// it will skip the metadata line which is supported by bulk insert format  
if (readLine.startsWith(“{\”index”)){  
continue;  
} else {

Object json = parser.parse(readLine);  
if(((JSONObject)json).get(“account\_number”)!=null){  
\_id = String.valueOf(((JSONObject)json).get(“account\_number”));  
System.out.println(\_id);  
}

//\_id is unique field in elasticsearch  
JSONObject jsonObject = (JSONObject) json;  
bulkBuilder.add(client.prepareIndex(index, type, String.valueOf(\_id)).setSource(jsonObject));  
bulkBuilderLength++;

try {  
if(bulkBuilderLength % 100== 0){  
System.out.println(“##### ” + bulkBuilderLength + ” data indexed.”);  
BulkResponse bulkRes = bulkBuilder.execute().actionGet();  
if(bulkRes.hasFailures()){  
System.out.println(“##### Bulk Request failure with error: ” +                                              bulkRes.buildFailureMessage());  
}  
bulkBuilder = client.prepareBulk();  
}  
} catch (Exception e) {  
e.printStackTrace();  
}  
}  
}  
br.close();

if(bulkBuilder.numberOfActions() > 0){  
System.out.println(“##### ” + bulkBuilderLength + ” data indexed.”);  
BulkResponse bulkRes = bulkBuilder.execute().actionGet();  
if(bulkRes.hasFailures()){  
System.out.println(“##### Bulk Request failure with error: ” + bulkRes.buildFailureMessage());  
}  
bulkBuilder = client.prepareBulk();  
}  
}

}