bulk api奇特的 json 格式

```
奇特的 json 格式
{"action":{"meta"}}\n
{"data"}\n
{"action":{"meta"}}\n
{"data"}\n

常见的 json 格式
[{
    "action":{

    },
    "data":{

    }
}]

```

1、 bulk中的每个操作都可能要转发到不同的 node 的 shard 去执行

2、如果采用比较良好的 json 数组格式

允许任意的换行，整个可读性非常棒，读起来很爽， es 拿到那种标准格式的 json 串以后，要按照下述流程去进行处理

1. 将 json 数组解析为 JSONArray 对象，这个时候，整个数据，就会在内存中出现一份一模一样的拷贝，一份数据 json 文本，一份数据是 JSONArray 对象
2. 解析 json 数组里的每个 json， 对每个请求中的 document 进行路由
3. 为路由到同一个 shard 上的多个请求，创建一个请求数组
4. 将这个请求数组序列化
5. 将序列化后的请求数组发送到对应的节点上去

3、耗费更多内存，更多的 jvm gc 开销

bulk size 最佳大小的那个问题， 一般建议说是在几千条那样， 然后大小在 10MB左右，所以说，可怕的事情来了。假设说现在100个bulk 请求发送到了一个节点上去，然后每个请求是 10MB， 100个请求，就是 1000MB = 1GB，然后每个请求的 json 都 copy 一份为 jsonarray 对象， 此时内存中的占用就会翻倍， 就会占用2GB，甚至还不止。因为弄成 jsonarray 之后，还可能会多搞出一些其他数据结构，2GB+的内存占用。

占用更多的内存可能就是挤压其他请求，分析请求，等等，此时就可能会导致其他请求的性能急剧下降
另外的话，占用内存更多，就会导致 java 虚拟机的垃圾回收次数增多，更加频繁，每次要回收的垃圾对象更多，好肥的时间更多，导致 es 的java 虚拟机停止工作线程的时间更多。

4、现在的奇特格式

1. 直接按照换行符切割 json
2. 对每两个一组的 json，读取meta，进行documnet 路由
3. 直接将对应的 json 发送到 node 上去

5、最大的优势在于，不需要将 json 数组 解析为一个 JSONArray 对象，形成一份大数据的拷贝，浪费内存空间