**ESRally进阶文档**

V1.0.0

吴志彬 @ Enlink 数据中心

**目录**

[1. 概述 3](#_Toc18644)

[1.1. 文档内容 3](#_Toc17851)

[1.2. 自定义数据集与自定义测试方案的目的 3](#_Toc3601)

[1.3. ESrally目录结构 3](#_Toc24224)

[2. 测试构成 4](#_Toc31788)

[2.1. 自定义数据格式 4](#_Toc4620)

[2.2. 自定义track 4](#_Toc28753)

[3. track.json分析 5](#_Toc29574)

[4. operation.json分析 7](#_Toc22217)

[4.1. Operation-type的可选项 8](#_Toc20878)

[4.1.1. bulk 8](#_Toc29192)

[4.1.2. index-stats 8](#_Toc8940)

[4.1.3. node-stats 9](#_Toc19254)

[4.1.4. force-merge 9](#_Toc2986)

[4.1.5. search 10](#_Toc8135)

[4.1.6. cluster-health 10](#_Toc7856)

[4.1.7. put-pipeline 11](#_Toc32041)

[4.1.8. refresh 12](#_Toc26961)

[4.1.9. create-index 12](#_Toc657)

[5. challenge.json 14](#_Toc87)

[5.1. 同时执行任务 16](#_Toc16997)

[6. 运行与打标签 17](#_Toc16969)

[6.1. 运行 17](#_Toc18402)

[6.2. 打标签 17](#_Toc24831)

[6.3. 使用不同的car运行 17](#_Toc10976)

# 概述

## 文档内容

本文档中介绍了如何自定义数据集和自定义测试方案

## 自定义数据集与自定义测试方案的目的

在ESRally中，定义的测试方案

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 描述 | 文档数 | 文件大小 |
| geonames | POIs from Geonames | 1139560 | 3.3G |
| Geopoint | Point coordinates from PlanetOSM | 60844404 | 2.3G |
| http\_log | HTTP server log data | 247249096 | 31.1G |
| Nyc\_taxis | Taxi rides in New York in 2015 | 165346692 | 74.3G |

自定义数据集和测试方案是为了更符合实际情况

## ESrally目录结构

~/benchmark

distribuctions

races

terms

tracks

data

data ：存放数据源

distribuctions ：存放ElasticSearch

races ：存放每次的压测记录，以时间命名文件夹，分别存储

terms ：存放car和插件

tracks ：存放压测方案

# 测试构成

## 自定义数据格式

自定义一个数据集需要一个json文件和一个bz2的压缩文件存放于data目录下

在json文件中，每行都是一个document

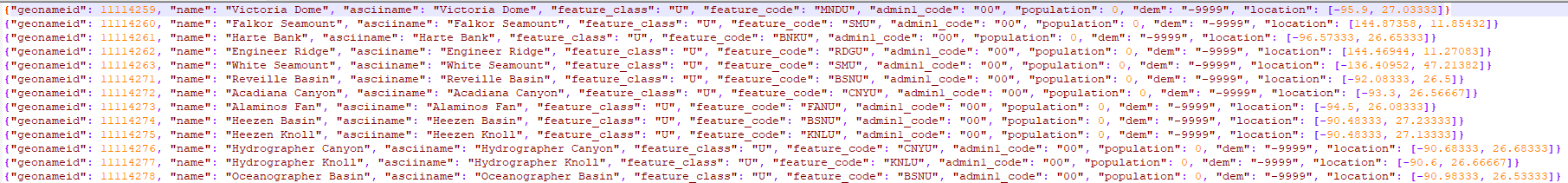


图2-1 数据格式样例

而压缩文件则是由json文件压缩而来

使用命令

|  |
| --- |
| tar cvjf documents.json.bz2 documents.json |

## 自定义track

Track目录介绍

~/tracks

operations

files.txt

index.json

track.json

challenges

challenges : 存放是一个json文件，其中定义了整个压测方案，由operations 中定义

operations ：存放的是一个json文件，其中定义了一个个的压测

files.txt : 存放的是数据源的名称

incex.json : 定义的是索引模板

track.json : 定义的是索引和数据源的相关信息

# track.json分析

表 3-1 track.json 分析

|  |
| --- |
| {% import "rally.helpers" as rally with context %}  {  "version": 2, #描述的是轨道规范版本的整数，0.9.0之后都是2  "description": "Enlink user", #描述这个轨道  "indices": [  {  "name": "enuser", #索引名称  "body": "index.json", #调用create index API 用作主体的文件名称  "types": [ "doc" ] #索引类型  }  ],  "templates": [ #定义索引模板  {  "name": "my-default-index-template", #索引模板名称  "index-pattern": "my-index-\*", #与索引模板匹配的索引  "delete-matching-indices": true, #在开始压测之前删除与提供的索引模式匹配的索引  "template": "default-template.json" #索引文件名  }  ],  "corpora": [ #定义数据文件  {  "name": "enadmin", #该数据源的名称  "documents": [  {  "source-file": "documents-2.json.bz2", #数据文件的压缩文件，放在data中，文件名相对应(如果对于本地文件，使用.json，如果是使用base-url，推荐使用压缩文件。必须是一个具有相同名称的JSON文件)  "document-count": 200, #数据条数，Rally使用此数字来确定哪些客户端索引文档语料库的哪个部分（N个客户端中的每一个都获得文档语料库的第N个）  "compressed-bytes": 973, #压缩后的数据大小  "uncompressed-bytes": 30378 #未压缩的数据大小  #其他属性  "base-url": "" #一个http或https地址，指向Rally可以获取相应源文件的根路径  "source-formate": "" #指定Rally以何种方式解析指定的数据文件source-file,默认bulk.也仅支持bulk  "target-index": "" #如果在indices中定义了index，则Rally会自动派生此值  "target-type": "" #如果在indices中定义了type，则Rally会自动派生此值  },  {  ... #定义多个document  }  ]  }  ],  "operations": [  {{ rally.collect(parts="operations/\*.json") }} #详细的操作，如增删改查  ],  "challenges": [  {{ rally.collect(parts="challenges/\*.json") }} #调用operations中的\*.json，再增加了client等参数，使用名称相对应  ]  } |

# operation.json分析

表4-1 operation.json分析

|  |
| --- |
| {  "name": "index", #此操作的名称，在定义计划时引用该操作  "operation-type": "bulk", #操作类型，下面有具体的介绍  "bulk-size": {{bulk\_size | default(5000)}},  "ingest-percentage": {{ingest\_percentage | default(100)}} #用于定义将对文档语料库进行批量索引的程度，默认100  "include-in-reporting": "true" #这个操作是否应该包含在命令行报告中。例如，您可能希望Rally为您创建一个索引，但您对有关它的详细指标不感兴趣。注意，Rally仍然会记录metrics存储中的所有指标。  },  {  "name": "percolator\_with\_content\_president\_bush", #此操作的名称，在定义计划时引用该操作  "operation-type": "search", #操作类型  "body": {  "query": {  "percolate": {  "field": "query",  "document": {  "body": "president bush"  }  }  }  }  },  {  ...  } |

## Operation-type的可选项

### bulk

表 4-2 bulk详情

|  |
| --- |
| {  "name": "",  "operation-type": "bulk", #（必填）类型  "bulk-size": 123, #（必填）定义文档数量的批量大小  "ingest-percentage": 100, #（可选）用于定义对文档语料库进行批量索引的程度，数字介于0-100，默认100  "corpora": "", #（可选）仅当corpora部分包含多个文档语料库且当你不想使用此操作索引所有文档时才需要  "indices": "", #（可选）索引名称列表，用于定义此批量索引操作应使用的索引，然后，Rally将仅选择具有target-index指定匹配的文档文件  "batch-size": "", #（可选）定义Rally一次读取的文档数，为了避免非常小的体积大小的意外瓶颈  "pipeline": "", #（可选）定义使用的管道  "conflicts": "" #（可选）模拟索引类型冲突，如未指定，则不模拟冲突，有效值"sequential"，文档ID替换为顺序递增ID。有效值""random"，文档ID替换为随机ID    } |

### index-stats

表 4-3 index-stats 详情

|  |
| --- |
| {  "name": "",  "operation-type": "index-stats" #不支持任何参数，吞吐量将报告为每秒完成的'index-stats'操作数  } |

### node-stats

表 4-4 node-stats详情

|  |
| --- |
| {  "name": "",  "operation-type": "node-stats" #不支持任何参数，吞吐量将报告为每秒完成的'node-stats'操作数  } |

### force-merge

表 4-5 force-merge详情

|  |
| --- |
| {  "name": "",  "operation-type": "bulk", #（必填）类型  "bulk-size": 123, #（必填）定义文档数量的批量大小  "ingest-percentage": 100, #（可选）用于定义对文档语料库进行批量索引的程度，数字介于0-100，默认100  "corpora": "", #（可选）仅当corpora部分包含多个文档语料库且当你不想使用此操作索引所有文档时才需要  "indices": "", #（可选）索引名称列表，用于定义此批量索引操作应使用的索引，然后，Rally将仅选择具有target-index指定匹配的文档文件  "batch-size": "", #（可选）定义Rally一次读取的文档数，为了避免非常小的体积大小的意外瓶颈  "pipeline": "", #（可选）定义使用的管道  "conflicts": "" #（可选）模拟索引类型冲突，如未指定，则不模拟冲突，有效值"sequential"，文档ID替换为顺序递增ID。有效值""random"，文档ID替换为随机ID    } |

### search

表 4-6 search详情

|  |
| --- |
| {  "name": "",  "operation-type": "search",  "index": "", #(可选)一种索引模式，用于定义查询应定位的索引，仅当该index包含多个索引时才需要，否则，Rally将自动派生索引使用，如果已经定义多个索引，且想查询到所有索引，则指定"index": "\_all"  "type": "" #(可选)指定索引中的类型，默认情况下，是检索所有的types。且考虑到在以后的版本中会去除type属性，这项不需要指定  "cache": "" #(可选)是否使用缓存，此项依赖ES的版本  "body": { #(强制)搜索正文  "query": {  "match\_all": {}  }  },  "pages": "" #(可选)要检索的页数，如果存在此项，则执行滚动查询，如果要检索所有结果页面，请使用'all'  } |

### cluster-health

表 4-7 cluster-health 详情

|  |
| --- |
| 这是一个管理操作，默认情况下不报告，如果需要报告，则需要设置强制"include-in-reporting"到true，然后报告每秒强制合并操作的数量  {  "name": "check-cluster-green",  "operation-type": "cluster-health",  "index": "logs-\*", #(可选)应用于检查索引的名称  "request-params": { #(可选)包含集群运行状况API允许的任何请求参数的结构  "wait\_for\_status": "green",  "wait\_for\_no\_relocating\_shards": "true"  }  } |

### put-pipeline

表 4-8 put-pipeline 详情

|  |
| --- |
| 需要安装插件  这是一个管理操作，默认情况下不报告，如果需要报告，则需要设置强制"include-in-reporting"到true，然后报告每秒强制合并操作的数量  {  "name": "define-ip-geocoder", #  "operation-type": "put-pipeline", #  "id": "ip-geocoder", #(必填)管道ID  "body": { #(必填)管道定义  "description": "描述信息",  "processors": [  {  "geoip": {  "field": "clientip",  "properties": [  "city\_name",  "country\_iso\_code",  "country\_name",  "location"  ]  }  }  ]  }  } |

### refresh

表 4-9 refresh 详情

|  |
| --- |
| {  这是一个管理操作，默认情况下不报告，如果需要报告，则需要设置强制"include-in-reporting"到true，然后报告每秒强制合并操作的数量  "name": "",  "operation-type": "refresh",  "index": "" #(可选)应刷新的索引的名称，默认 \_all  } |

### create-index

分为两种模式：

1. 创建在轨道indices部分中指定的所有索引
2. 创建一个特定索引

表4-10 创建在indices中指定的索引

|  |
| --- |
| 这是一个管理操作。默认情况下不报告度量标准。报告可以通过设置强制include-in-reporting到true。  {  这段代码将创建已在indices部分中定义的所有索引，将重用所有已定义的索引，但是会覆盖分片数量  "name": "create-all-indices",  "operation-type": "create-index",  "settings": { #(可选)允许指定其他索引设置，这些设置将与该部分中索引正文指定的索引设置合并成indices  "index.number\_of\_shards": 1  },  "request-params": { #(可选)包含create Index API允许的任何请求参数的结构  "wait\_for\_active\_shards": true  } |

表 4-11 创建一个特定索引

|  |
| --- |
| 这段代码将创建一个未在indices部分定义的新索引，将直接在正文中指定索引s  {  "name": "create-an-index",  "operation-type": "create-index",  "index": "people", #(必填)应创建的索引  "body": { #(可选)由于创建索引API调用的主体  "settings": {  "index.number\_of\_shards": 0  },  "mappings": {  "docs": {  "properties": {  "name": {  "type": "text"  }  }  }  }  }  } |

# challenge.json

*表 5-1 challenge.json 详情*

|  |
| --- |
| {  "name": "append-no-conflicts",  "description": "描述",  "default": true, #是否默认，当没有选择挑战时，则使用true的选项  "schedule": [ #包含Rally执行的任务列表  {  "operation": "delete-index" #(必选)此属性引用在operations中定义的操作的名称，或直接定义内联操作  },  {  "operation": {  "operation-type": "create-index",  "settings": {{index\_settings | default({}) | tojson}}  }  },  {  "name": "check-cluster-health",  "operation": {  "operation-type": "cluster-health",  "index": "geonames",  "request-params": {  "wait\_for\_status": "{{cluster\_health | default('green')}}",  "wait\_for\_no\_relocating\_shards": "true"  }  }  },  {  "operation": "index-append",  "warmup-time-period": 120, #(可选)Rally考虑对基准候选者进行预热的时间段（以秒为单位），预热期间捕获的所有数据都不会显示在测量结果中  "clients": {{bulk\_indexing\_clients | default(8)}} #(可选)同时执行任务的客户端数，默认1  "time-period": 3600 #()Rally考虑测量的时间段(以秒为单位)，请注意，对于批量索引，通常不设定此时间段。Rally将对所有文档进行批量索引，并将预热时间段后的每个样本视为测量样本    },  {  "operation": "index-stats",  "clients": 1, #(可选)同时执行任务的客户端数  "warmup-iterations": 500, #(可选)每个客户端执行的预热基准候选者的迭代次数，预热迭代不会显示在测量结果中，默认为0  "iterations": 1000,  "target-throughput": 100 #(可选)定义基准模式。如果未定义，则Rally假定这是吞吐量基准测试，并将尽可能快地执行任务。这主要是批量式操作所必需的，其中更重要的是实现最佳吞吐量而不是可接受的延迟。如果已定义，则它指定所有客户端上每秒的请求数。例如，如果您指定8个客户端，则意味着每个客户端将每秒发出125（=1000/8）个请求。总的来说，所有客户端每秒会发出1000个请求。如果Rally报告的吞吐量低于指定的吞吐量，那么Elasticsearch根本无法访问它。target-throughput: 1000  },  {  "parallel": { #同时执行任务  "warmup-iterations": 50,  "iterations": 100,  #"clients": 2 #这边明确定义了所有任务只有2个客户端  "tasks": [  {  "operation": "default",  "clients": 4,  "target-throughput": 50  },  {  "operation": "term",  "target-throughput": 200  },  {  "operation": "phrase",  "target-throughput": 200  }  ]  }  } } |

## 同时执行任务

将任务包装在parallel元素中来同时执行

*表 5-2 同时执行任务信息*

|  |
| --- |
| client: (可选)提供执行任务的客户端数  warmup-time-period: (可选)允许为parallel元素的所有任务定义默认值  time-period: (可选)允许为parallel元素的所有任务定义默认值  task: (必选)定义同时执行任务的列表，列表中的任务可以定义上面已经定义的属性  {  "parallel": {  "warmup-iterations": 50,  "iterations": 100,  "tasks": [  {  "operation": "default",  "clients": 4,  "target-throughput": 50  },  {  "operation": "term",  "target-throughput": 200  },  {  "operation": "phrase",  "target-throughput": 200  }  ]  }  } |

# 运行与打标签

## 运行

|  |
| --- |
| esrally race --pipeline=benchmark-only --target-hosts=localhost:9200 --target-path=/usr/local/esrally/benchmarks/track/enadmin --challenge=search |

其中 --target-path中输入的是track.json所在的目录。

## 打标签

|  |
| --- |
| esrally race --pipeline=benchmark-only --target-hosts=localhost:9200 --track-path=/usr/local/esrally/benchmarks/track/enadmin --user-tag="K:V"  eg:  esrally race --pipeline=benchmark-only --target-hosts=localhost:9200 --target-path=/usr/local/esrally/benchmarks/track/enadmin --user-tag="data:10000,client:8,bulksize:5000" |

打标签使用的是 --user-tag = “K:V”

## 使用不同的car运行

|  |
| --- |
| esrally race --pipeline=benchmarks-only --target-host=localhost:9200 --car="4gheap"  esrally race --pipeline=benchmarks-only --target-host=localhost:9200 --car="8gheap"  ... |

可以使用

|  |
| --- |
| esrally list cars |

查看所有的car选项