**ESRally使用文档**

V1.0.0

吴志彬 @ Enlink 数据中心

**目录**

[1. 概述 3](#_Toc23498)

[1.1. 文档内容 3](#_Toc14115)

[1.2. ESRally的简介 3](#_Toc11305)

[2. 使用ESRally 4](#_Toc6616)

[2.1. 第一次运行需要配置数据存储位置 4](#_Toc1207)

[2.2. 开始第一次压测 5](#_Toc6593)

[2.3. 压测后的结果展示 6](#_Toc17758)

[3. 相关术语 7](#_Toc32585)

[3.1. Track 7](#_Toc25716)

[3.1.1. Operations/default.json 8](#_Toc24728)

[3.1.2. Challengs/default.json 9](#_Toc31725)

[3.2. Car 11](#_Toc667)

[3.3. Race 12](#_Toc3260)

[3.4. Tournament 12](#_Toc23368)

[3.5. Pipeline 14](#_Toc4746)

[4. 压测结果分析 15](#_Toc1798)

[5. 结合Kibana分析压测结果 16](#_Toc4599)

[5.1. Rally-metrices-\* 16](#_Toc21097)

[5.2. Rally-result-\* 17](#_Toc24054)

[5.3. rally-races-\* 19](#_Toc17047)

[6. 常用指令 19](#_Toc29963)

[6.1. 常用组合命令 19](#_Toc6678)

# 概述

## 文档内容

本文档包含ESRally的基本介绍、指令的使用与压测结果的分析

## ESRally的简介

Esrally是Elastic官方开源的一款基于Python3实现的针对ES的压测工具。主要功能：

* 自动创建、压测和销毁ES集群
* 完善的压测数据展示，支持压测数据对比分析，也可以将数据存储到指定的ES中进行二次分析
* 支持收集JVM详细信息，比如内存、GC等数据来定位性能问题

Elastic官方也是基于ESRally进行ES的性能测试，并将结果实时发布到[https://elasticsearch-benchmarks.elastic.co/](https://link.jianshu.com/?t=https://elasticsearch-benchmarks.elastic.co/" \t "https://www.jianshu.com/p/_blank) 我们也可以从该网站查看ES的性能。官方是使用两台服务器进行压测的，一台运行ESRally，一台运行ES，服务器配置如图1-1：

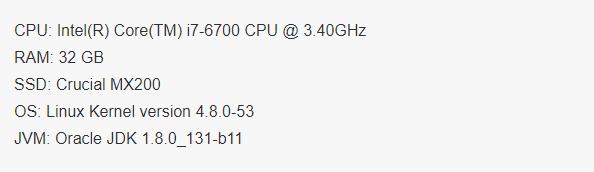


图 1-1 ES官方用的测试配置

# 使用ESRally

## 第一次运行需要配置数据存储位置

|  |
| --- |
| esrally configure --advanced-config |

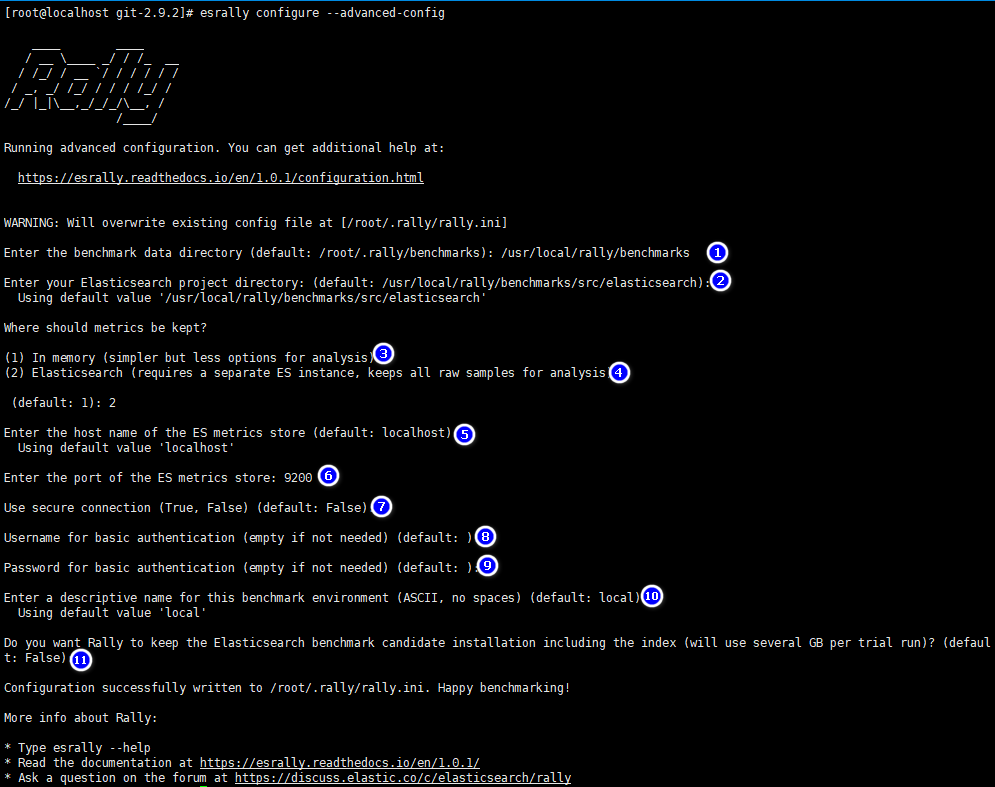


图 2-1 ESRally配置图

1. 配置压测数据的目录
2. 配置ES程序的目录，在配置完成第一条目录的时候，这个目录是自动配置的
3. 将测试结果保存到内存中
4. 将测试结果保存到ES中。如图2-6，本例选择的是选择ES（选择ES后，压测需要启动ES节点）
5. 输入主机名（默认localhost）
6. 输入ES端口
7. 使用安全连接（使用默认false即可）
8. 安全认证用户名（默认为空，即不存在）
9. 安全认证密码（默认为空，即不存在）
10. 为这个基准测试环境输入名称（使用默认值local即可）
11. 是否希望ESRally保留ES基准测试候选安装包（使用默认值false即可）

## 开始第一次压测

ESRally不允许root用户启动，所以要创建一个用户（ES启动的时候就是需要非root用户，所以可以使用ES的用户启动ESRally）

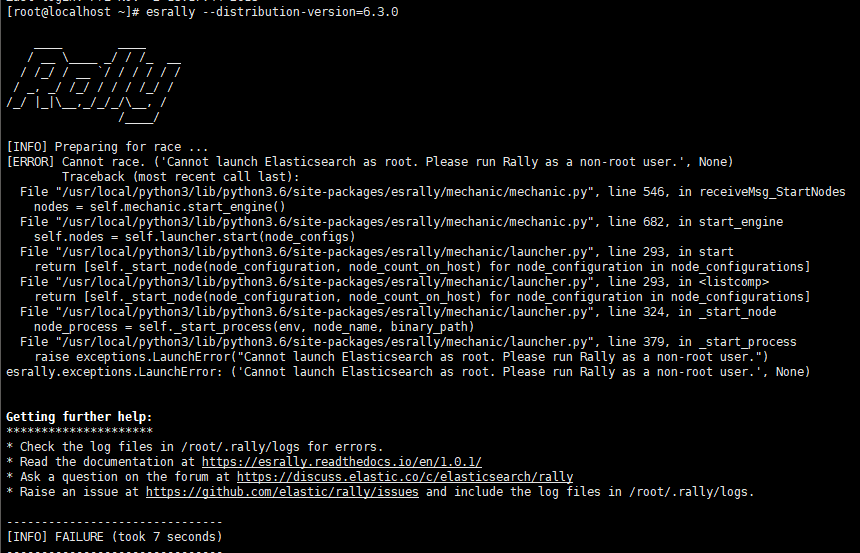


图 2-2 使用root用户启动ESRally报错

使用命令

|  |
| --- |
| esrally --distribution-version=6.3.0 |

这个指令会下载Elasticsearch 6.3.0并进行Rally的默认轨道-geonames轨道，在压测结束后，会将摘要报告写入命令行。

推荐使用本地ES节点进行压测，可以使用root用户

使用命令

|  |
| --- |
| esrally race --pipeline=benchmark-only --target-hosts=localhost:9200 |

## 压测后的结果展示

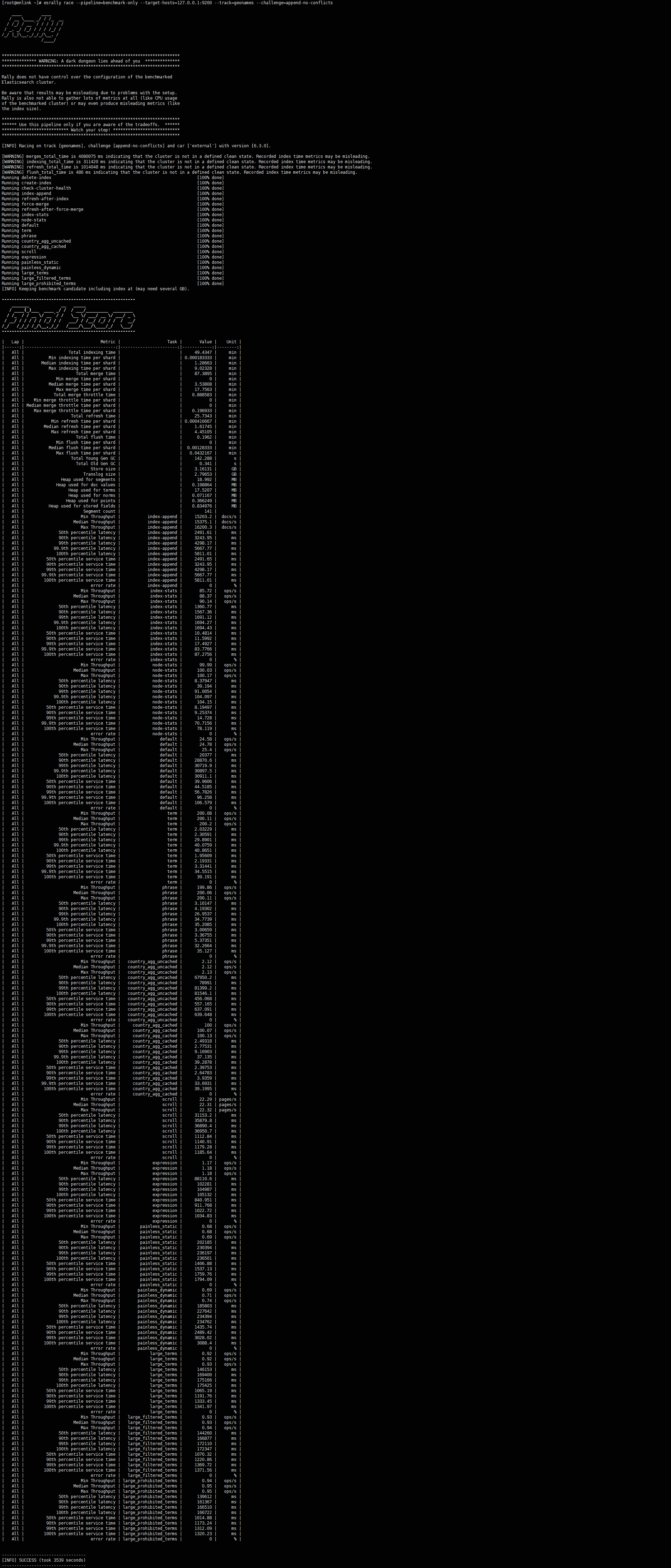


图 2-3 ESRally 压测结果

# 相关术语

rally是汽车拉力赛的意思，ESRally就是将压测比作了汽车拉力赛，因此其中很多术语都是从汽车拉力赛中借鉴来的

## Track

查看ESRally当前可以用的track

|  |
| --- |
| esrally list tracks |

Track是赛道意思，这里指的是压测用的数据和测试策略，ESRally自带的track都在github中，地址是：<https://github.com/elastic/rally-tracks> ,在这个仓库中，有许多测试数据，比如geonames,geopoint,eventdata,nyc\_taxis等，每个数据文件夹中的README.md中都有详细的介绍，其中的track.json便是压测策略的定义文件

下面便是geonames的track.json

表 3-1 geonames/track.json文件

|  |
| --- |
| {% import "rally.helpers" as rally with context %}  {  "version": 2,  "description": "POIs from Geonames",  "data-url": "http://benchmarks.elasticsearch.org.s3.amazonaws.com/corpora/geonames",  "indices": [  {  "name": "geonames",  "body": "index.json",  "types": [ "type" ]  }  ],  "corpora": [  {  "name": "geonames",  "base-url": "http://benchmarks.elasticsearch.org.s3.amazonaws.com/corpora/geonames",  "documents": [  {  "source-file": "documents-2.json.bz2",  "document-count": 11396505,  "compressed-bytes": 264698741,  "uncompressed-bytes": 3547614383  }  ]  }  ],  "operations": [  {{ rally.collect(parts="operations/\*.json") }}  ],  "challenges": [  {{ rally.collect(parts="challenges/\*.json") }}  ]  } |

track.json文件几个主要部分：

description:主要用于描述该json的

data-url:指明测试数据的下载根路径，与下方的corpora.documents.source-file可以得到数据下载地址

indices：包含轨道使用的索引列表：

<name>：索引的名称

<body>：将在create Index API中调用主题的相应索引定义的文件名

<types>：此索引中的类型名称列表

operations:指定具体的操作，比如index索引数据的操作、force-merge强制合并Segment的操作、search的操作等等。即表3-2的 ”operation-type”

challenges:通过组合operations定义一系列task，在组合成一个压测流程，在表3-3中有详细内容

### Operations/default.json

表 3-2 operations/default.json文件部分

|  |
| --- |
| {  "name": "index-append",  "operation-type": "bulk",  "bulk-size": {{bulk\_size | default(5000)}},  "ingest-percentage": {{ingest\_percentage | default(100)}}  },  {  "name": "index-update",  "operation-type": "bulk",  "bulk-size": {{bulk\_size | default(5000)}},  "ingest-percentage": {{ingest\_percentage | default(100)}},  "conflicts": "random",  "on-conflict": "{{on\_conflict | default('index')}}",  "conflict-probability": {{conflict\_probability | default(25)}},  "recency": {{recency | default(0)}}  },  {  "name": "default",  "operation-type": "search",  "body": {  "query": {  "match\_all": {}  }  }  }  } |

该operations中会定义所有计划操作：

* 插入数据时：

<name>:强制属性，指明操作的名称

<operation-type>:强制操作，指明这个操作的类型（index）

<bulk-size>:在插入数据时强制属性，定义文档数量的批量大小(默认5000)

<ingest-percentage>:用于定义将对文档内容进行批量索引的程度（默认100）

* 查询数据时：

<name>:强制属性，指明操作的名称

<operation-type>:强制操作，指明这个操作的类型（search）

<body>:在查询时强制属性，查询正文

### Challengs/default.json

表 3-3 challengs/default.json 文件部分

|  |
| --- |
| {  "name": "append-no-conflicts",  "description": "Indexes the whole document corpus using Elasticsearch default settings. We only adjust the number of replicas as we benchmark a single node cluster and Rally will only start the benchmark if the cluster turns green. Document ids are unique so all index operations are append only. After that a couple of queries are run.",  "default": true,  "schedule": [  {  "operation": "delete-index"  },  {  "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,  "clients": {{bulk\_indexing\_clients | default(8)}}  },  {  "name": "refresh-after-index",  "operation": "refresh",  "clients": 1  },  {  "operation": "force-merge",  "clients": 1  },  {  "operation": "painless\_static",  "clients": 1,  "warmup-iterations": 200,  "iterations": 100,  "target-throughput": 1.5  }  } |

这里定义了一个名为append-no-conflicts的challenge，由于每次压测只能运行一个challenge，这里的default参数指的是当压测未指定时默认运行，schedule指定了challenge中运行task的顺序，每个task都指定一个operation，此外，还可以指定 ”client” （并发客户端）、warmup-iterations(预热的循环次数)、iterations（operation执行的循环次数）

## Car

Car是赛车的意思，在ESRally中指的是ES的实例，通过命令可以查看当前可用的car

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

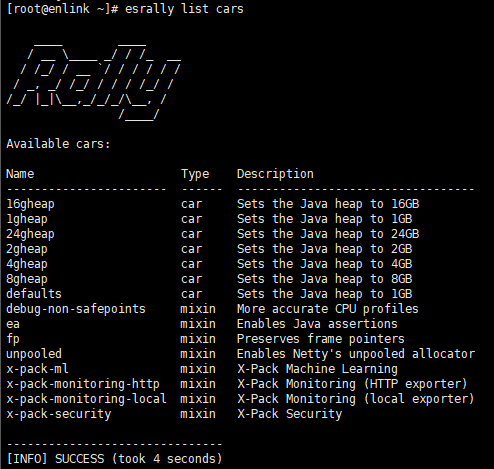


图 3-1 查看ESRally中可用的cars

Rally的默认汽车存储于指定的目录中(/usr/local/rally/benchmarks/teams/default/cars)，即在配置ESRally的时候指定（详情见[章节2.1](#_第一次运行需要配置数据存储位置)）

## Race

Race是一次比赛的意思，这里指的是一次压测，要比赛，就需要赛车和赛道，如果不指定赛车，则是默认的default配置，如果不指定赛道，则默认使用的是geonames track。

表 3-4 使用本地集群

|  |
| --- |
| esrally race --pipeline=benchmark-only --target-hosts=127.0.0.1:9200 --track=geonames --challenge=append-no-conflicts |

在表3-4 使用本地集群的指令中，使用的是geonames的track，运行的是该track中的append-no-conflicts（[章节3.1.2](#_Challengs/default.json)中的第一个 ”name” 属性）

## Tournament

Tournament是锦标赛的意思，由多个race组成，用来比较两次比赛

查看所有race

|  |
| --- |
| esrally list races |

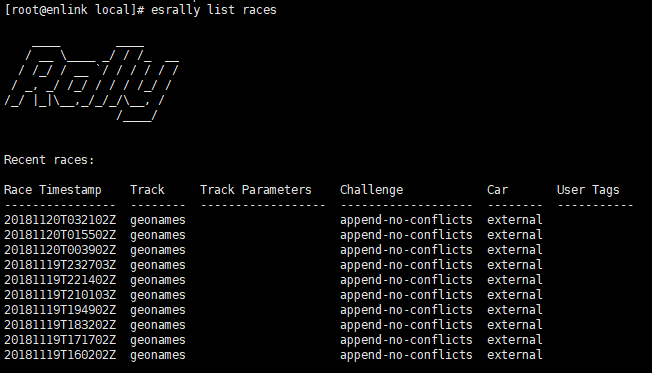


图 3-2 查看所有race

可以使用标签来标记每次race的意图

|  |
| --- |
| esrally race --pipeline=benchmark-only --target-hosts=127.0.0.1:9200 --track=geonames --challenge=append-no-conflicts --user-tag=”intention:ceshi” |

运行指令

|  |
| --- |
| esrally compare --baseline=20181120T032102Z --contender=20181119T160202Z |

baseline指的是用作基准的一次race，contender指的是用来与基准进行比较的一次race

可以很快的比较出两次比赛之见的差异，很好的用来做性能的调试

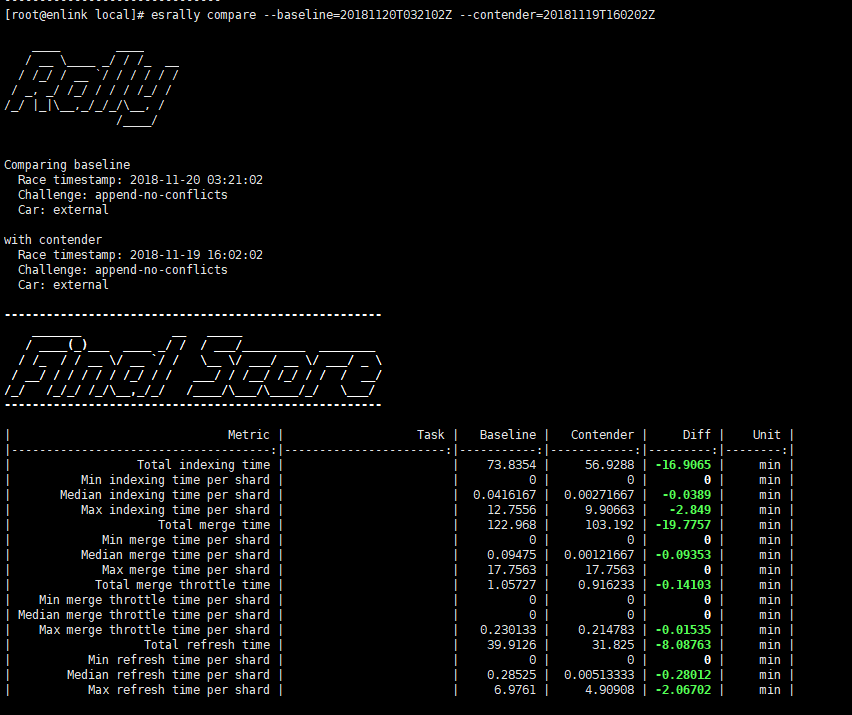


图 3-3 比较两个race

## Pipeline

Pipeline在这里指的是压测的一个流程

|  |
| --- |
| esrally list pipelines |

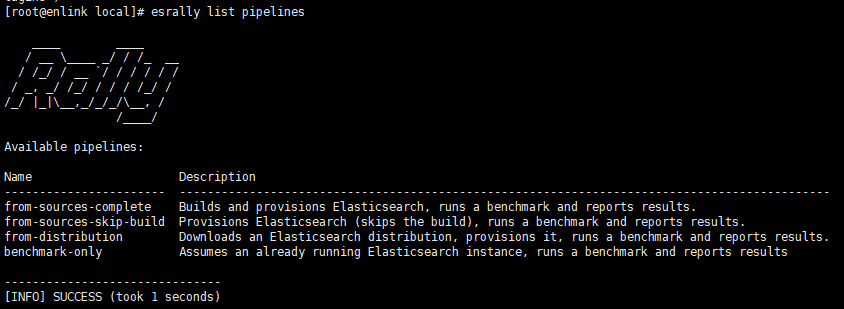


图 3-4 查看所有pipeline

from-source-complete 是从源码编译ES后再运行，可以通过 --revision参数指明要编译的commit hash，这样就可以针对某一次提交版本进行测试

from-sources-skip-build 如果已经编译好了，使用该pipeline，可以跳过编译流程，节省测试时间

from-distribution 通过 --distribution-version 指定ES版本，ESRally会从官网下载该版本的可执行文件，然后进行测试

benchmark-only 该pipeline将ES集群的管理交由用户来处理，ESRally只进行压测，如果想针对已有集群进行测试，则使用该pipeline

# 压测结果分析

压测结束后，ESRally会将结果输出到终端和存放与ESRally的目录benchmark/races中

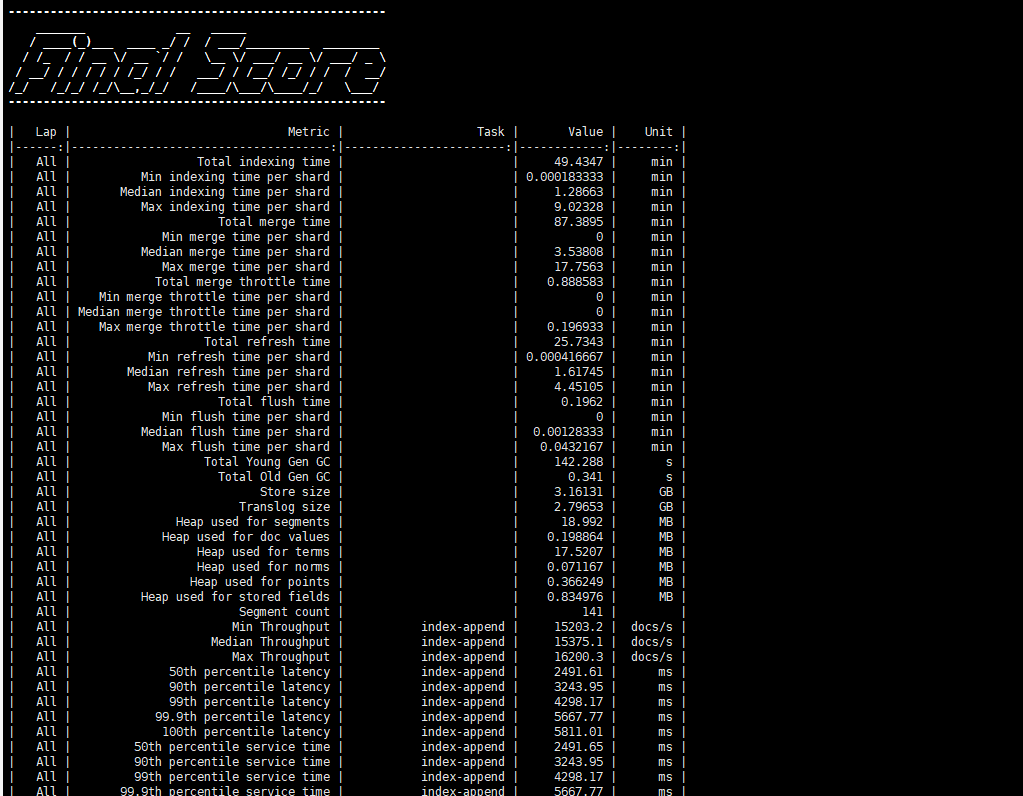


图 4-1 压测结果部分展示

在Metric一栏中，可以看到许多指标，一般关注的有：

throughput ： 每个操作的吞吐量，比如Index，search

latency : 每个操作的响应时长

heap used for x ：记录堆栈的使用情况

每次压测都会以压测的时间命名存于 /races中 ，比如20181120T032102Z，则在安装目录中/race中对应的是文件夹2018-11-20-03-21-02中的race.json

# 结合Kibana分析压测结果

在[章节2.1](#_第一次运行需要配置数据存储位置)中，我们已经将测试结果储存到了ES中

ESRally会将数据存储到3个index中（使用\*代替月份）

rally-metrices-\*

rally-races-\*

rally-results-\*

## Rally-metrices-\*

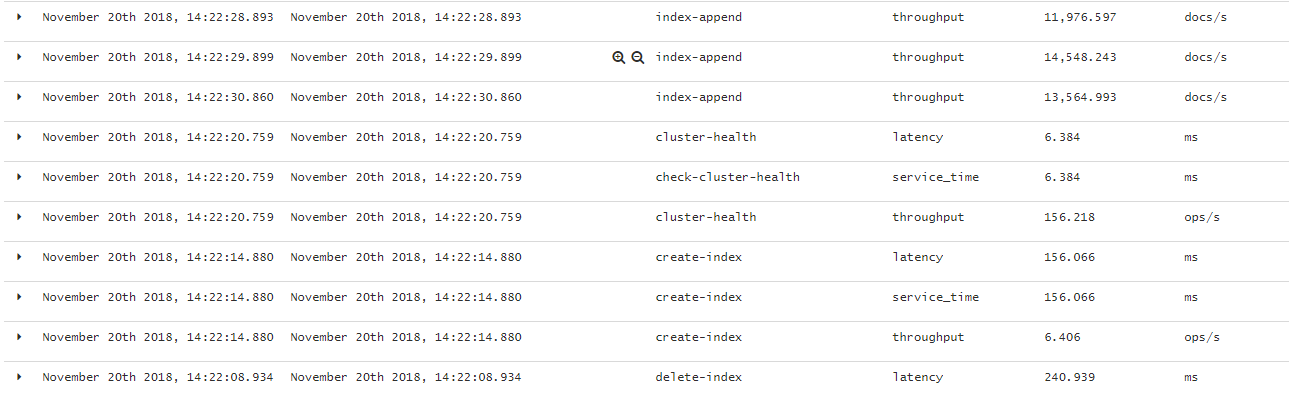


图 5-1 rally-metrices-\*部分展示

第一列时间是指某一次压测的时间，第二列时间是指的指标采集时间，第三列operation指的是具体执行的操作，operation为空的话则指的是总计类，比如indexing total time 记录的就是总索引数据的时间、Segment\_count是总段数等。

其他的operation都记录了每个曹组哦的数据，需要注意的是，这里记录的operation的所有采样数据，不是一个最终汇总数据，从图5-1 也可以看出一个 Index-append有多想throughout的指标数据，但是他们的采集时间是不同的，所以可以做出某一次race中某个指标的可视化视图。如想观察race中index-append这个task的throughout指标数据，便可以通过如下方式查看：

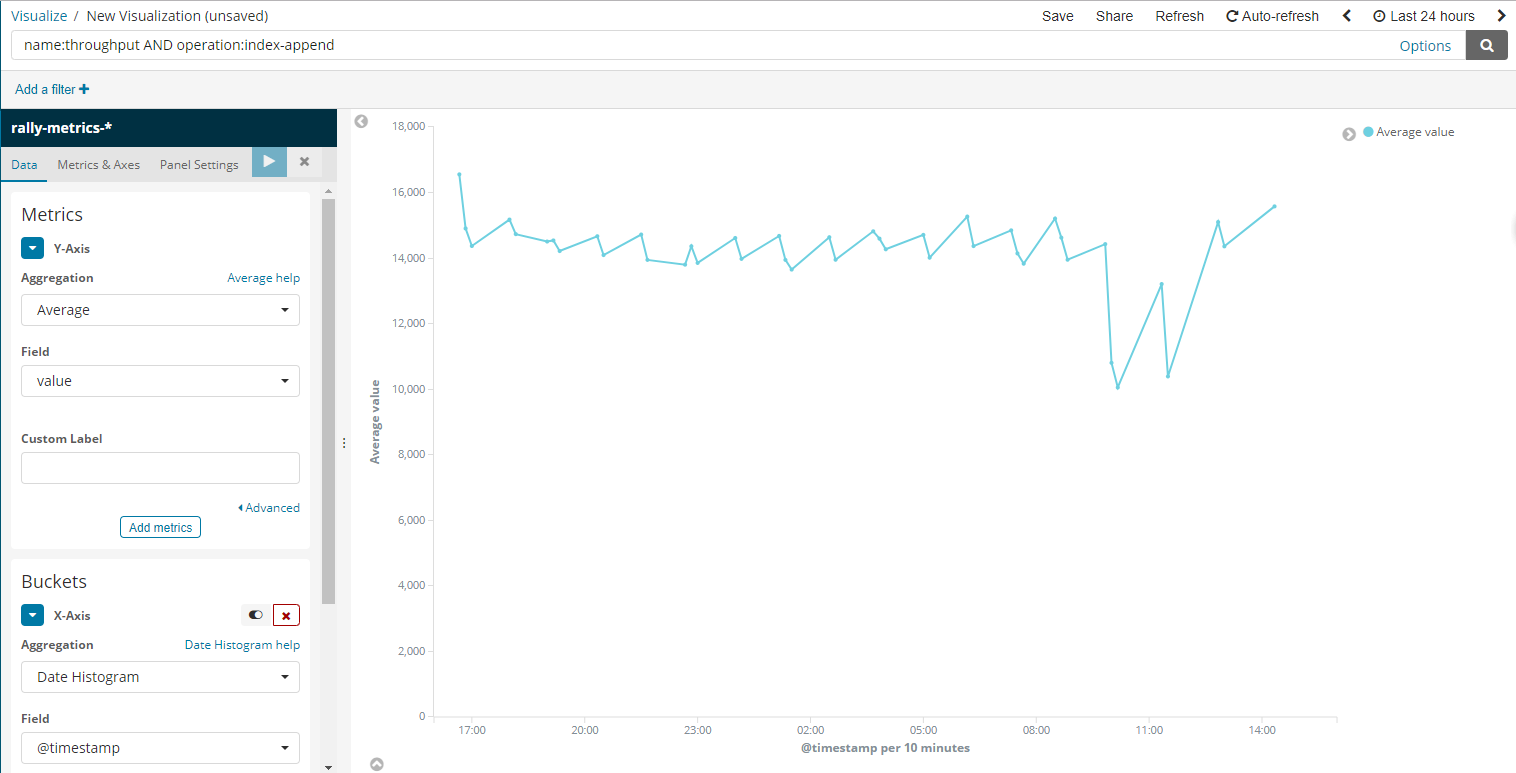


图 5-2 通过kibana查看rally-metrices-\* 中某个指标

## Rally-result-\*

rally-result-\* 是将每次race的最终汇总结果记录

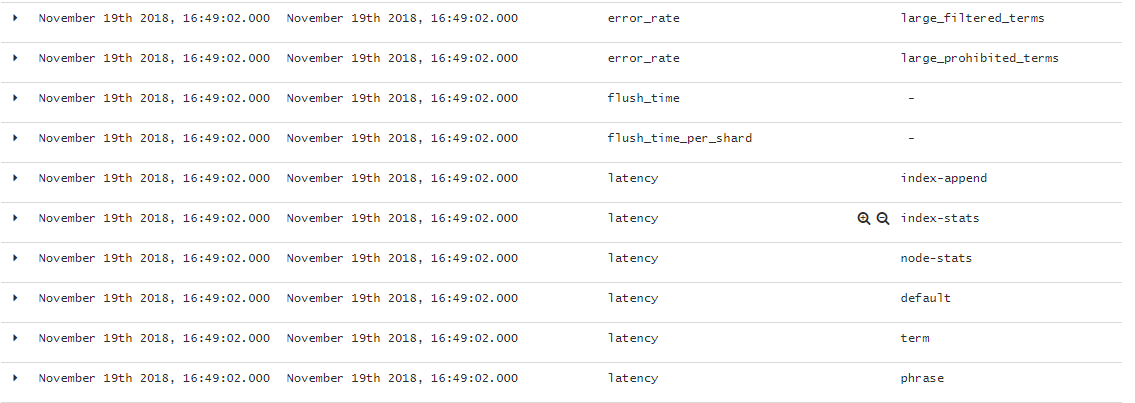


图 5-3 rally-result-\* 部分展示

查看其中某个指标详情



图 5-4rally-result-\* 某个指标详情

这条记录记录了term operation 的latency指标数据，汇总值以percentile（百分位数）的形式展示。基于该数值，我们可以绘制针对某个指标的多race对比

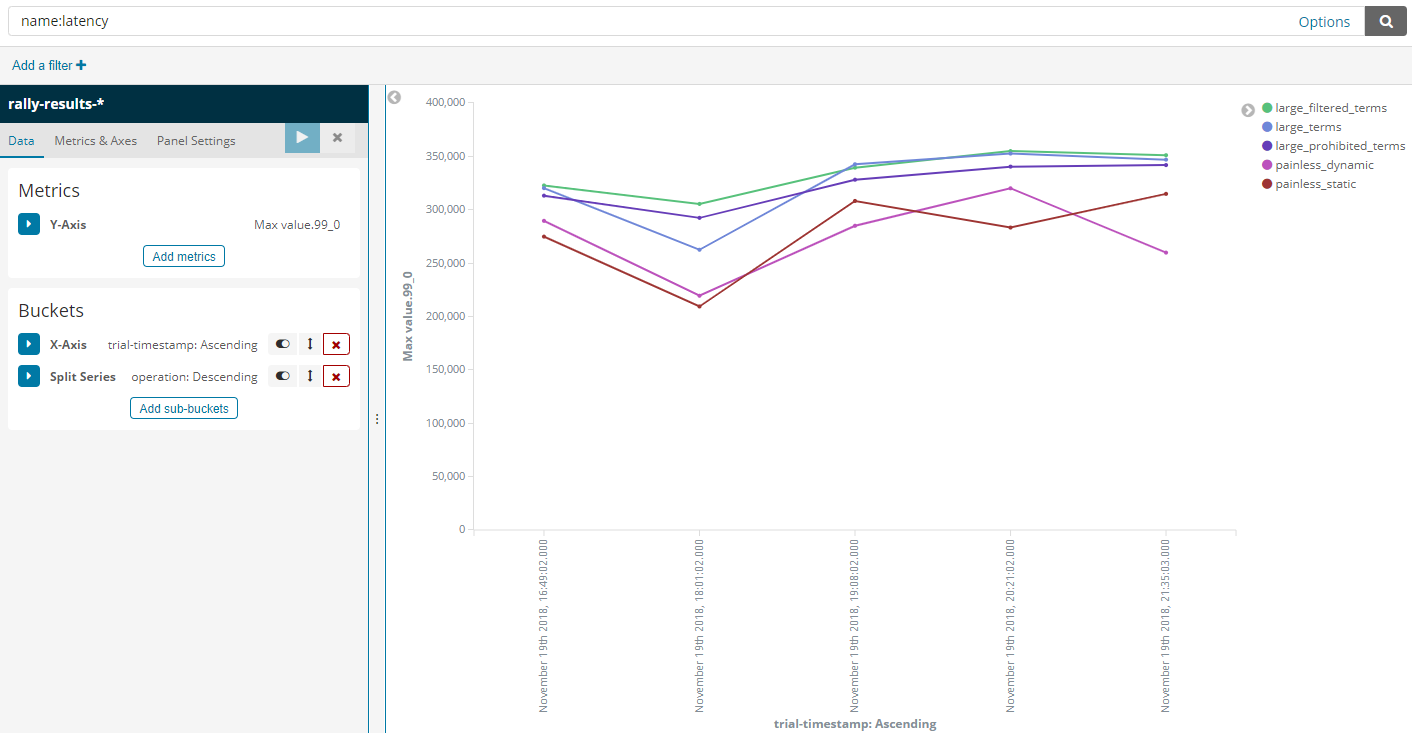


图 5-5 多次 race的latency比较

## rally-races-\*

rally-races-\* 是记录所有race的最终结果，即命令执行结束的输出结果

# 常用指令

使用本地ES集群测试:

--pipeline=benchmark-only

测试数据集，默认是geonames:

--track=geonames

使用离线数据（第一次使用会报错）:

--offline

## 常用组合指令

第一次测试从远端下载数据

|  |
| --- |
| esrally race --pipeline=benchmark-only --target-hosts=localhost:9200 |

第二次测试，数据集不变的话，直接使用本地数据

|  |
| --- |
| esrally race --pipeline=benchmark-only --target-hosts=localhost:9200 --offline |