时序数据库 CTSDB

开发指南

产品文档

腾讯云

文档目录

开发指南
HTTP API 参考
简介

最近更新时间：2019-10-18 16:34:45

CTSDB 支持 HTTP 协议进行数据写入和查询等操作。提供的 HTTP API 是 RESTful API，对资源的请求方式是通过向资源对应的 URI 发送标准的 HTTP 请求，例如 GET 用来获取资源，POST 用来新建资源（也可以用于更新资源），PUT 用来更新资源，DELETE 用来删除资源等。

用户通过 HTTP API 几乎可以实现所有的数据操作。CTSDB 通过提供 VPC 网络隔离和访问时提供用户名和密码的身份认证方式来保证数据的安全性，通过 JSON 格式的结构体进行数据交换，每个请求都会返回一个标准的 HTTP 响应状态码和响应内容。若操作失败，用户可以根据响应内容获取到具体错误信息。

通过 RESTful API 接口连接实例方式请参见 连接实例。

命名规则

CTSDB 中 metric、tag、field 的命名应简明扼要。

- **metric 命名规则**：允许使用小写英文字母、数字、 _（下划线）、 -（短划线）、 .（圆点）的组合，且不能以 .（圆点）、_（下划线） 或 -（短划线）开头。长度为1到200个字符。
- **metric 中 tags 和 fields 命名规则**：允许大小写英文字母、数字、_（下划线）、-（短划线）、 .（圆点）的任意组合，且不能以 .（圆点）开头。长度为1到255个字符。

系统限制

- **metric 中 tags 和 fields 限制**：metric 中字段总数不超过1000。
- **批量写入 metric 时写入点限制**：建议单次 bulk 条数在1000 - 5000之间，物理大小在1MB - 15MB大小之间。

系统默认规则

- **写入数据时新增字段处理**：如果您往 metric 表写入数据时有未定义的新增字段，CTSDB 默认会将新增字段存储为 tags。您也可通过修改 metric 的 options 选项的 default_type 字段来修改，修改 metric 请参考 更新 metric。
  - 若新增字段为整型，则 CTSDB 将其存储为 long 类型。
  - 若新增字段为小数，则 CTSDB 将其存储为 float 类型。
  - 若新增字段为字符串，则 CTSDB 将其存储为 string 类型，其长度为 max_string_length，超过部分会被系统丢弃，max_string_length 的长度可自定义，默认为256个字符，修改请参考 更新 metric。
- **日期与时间处理**：日期与时间在 CTSDB 中以 UTC 格式存储，因此，查询数据时涉及时间范围的查询请通过 time_zone 参数来指定时区，其格式为 ISO 8601 UTC 偏移（例如 +01:00 或 -08:00），具体时区需根据实例所处的地域来确定，一般国内地域是东八区。具体请参考 常用查询示例。

连接实例

CTSDB 实例目前只提供 VPC 网络下的连接方式。您可以通过控制台连接实例，也可以通过 RESTful API 接口连接实例，通过 API 接口连接实例时需要提供 root 帐号的密码，以确保安全性。

CURL 连接实例创建表示例如下，其中 ${user:password} 是实例的用户名和密码， ${vip}:${vport} 是实例的 IP 和 Port， ${metric_name} 是新建的表名称。

```
curl -u ${user:password} -XPUT ${vip}:${vport}/_metric/${metric_name} -d'
{
"tags": {
"region": "string",
"set": "long",
"host": "string"
},
"time": {
"name": "timestamp",
"format": "epoch_second"
```

```
},
"fields": {
"cpu_usage": "float"
},
"options": {
"expire_day": 7,
"refresh_interval": "10s",
"number_of_shards": 5,
"number_of_replicas": 1,
"rolling_period": 1
}
}'
```

新建 metric

最近更新时间：2019-10-22 09:28:09

请求地址

地址为实例的 IP 和 PORT，可从控制台获取到，例如10.13.20.15:9200。

请求路径和方法

- 路径：`/_metric/${metric_name}`，`${metric_name}` 为新建的 metric 的名称。
- 方法：PUT

> 说明：
> `metric` 命名限制请参见 系统限制。

请求参数

无

请求内容

参数名称	必选	类型	描述
tags	是	map	维度列，至少包含一个维度，支持的数据类型：text（带有分词、全文索引的字符串）、string（不分词的字符串）、long、integer、short、byte、double、float、date、boolean。 格式如 {"region": "string","set": "long","host": "string"}
time	否	map	时间列相关配置，例如 {"name": "timestamp", "format": "epoch_second"}，若不填则系统默认格式为 {"name": "timestamp", "format": "epoch_millis"}
fields	否	map	指标列，为了节省空间，建议使用最适合实际业务使用的类型，支持的数据类型：string（字符串）、long、integer、short、byte、double、float、date、boolean。 例如 {"cpu_usage":"float"}
options	否	map	常用的调优配置信息 例 如 {"expire_day":7,"refresh_interval":"10s","number_of_shards":5,"number_of_replicas":1,"rolling_period":1,"max_string_length":256,"default_date_format":"strict_date_optional_time","indexed_fields":["host"]}

- time 字段的 name 默认为 timestamp，时间格式（format）完全兼容 Elasticsearch 的时间格式，如 epoch_millis（以毫秒为单位的 Unix 时间戳）、epoch_second（以秒为单位的 Unix 时间戳）、basic_date（格式如 yyyyMMdd）、basic_date_time（格式如 yyyyMMdd'T'HHmmss.SSSZ）等。
- options 选项及解释如下：
  - expire_day：数据过期时间（单位：天），取值范围为非零整数，过期后数据自动清理，缺省情况下为最小值-1（代表永不过期）。
  - refresh_interval：数据刷新频率，写入的数据从内存刷新到磁盘后可查询。默认为10秒。
  - number_of_shards：表分片数，取值范围为正整数，小表可以忽略，大表按照一个分片至多25G设置分片数，默认为3。
  - number_of_replicas：副本数，取值范围为非负整数，例如一主一副为 1，缺省为 1。
  - rolling_period：子表时长（单位：天），取值范围为非零整数，CTSDB 存储数据时，为了方便做数据过期和提高查询效率，根据特定时间间隔划分子表，缺省情况下由数据过期时间决定，下面具体说明缺省子表时长和过期时间的关系。
  - max_string_length：自定义字符串类型的值最大可支持的长度，取值范围为正整数，最大为2^31 - 1，默认为256。
  - default_date_format：自定义维度列和指标列 date类型的格式，默认为 strict_date_optional_time 或 epoch_millis。
  - indexed_fields：指定指标列中需要保留索引的字段，可指定多个，以数组形式指定。
  - default_type：指定新增字段的默认类型。可选项为 tag、field，系统默认值为 tag。

过期时间	子表时长

过期时间	子表时长
≤ 7天	1天
> 7天，≤ 20天	3天
> 20天，≤ 49天	7天
> 49天，≤ 3个月	15天
> 3个月	30天
永不过期	30天

返回内容

需要通过 error 字段判断请求是否成功，若返回内容有 error 字段则请求失败，具体错误详情请参照 error 字段描述。

CURL 示例说明

请求：

```
curl -u root:le201909 -X PUT 172.16.345.14:9201/_metric/ctsdb_test -d'
{
"tags":
{
"region":"string"
},
"time":
{
"name":"timestamp",
"format":"epoch_second"
},
"fields":
{
"cpuUsage":"float"
},
"options":
{
"expire_day":7,
"refresh_interval":"10s",
"number_of_shards":5
}
}
'
```

成功的返回：

```
{
"acknowledged": true,
"message": "create ctsdb metric ctsdb_test success!"
}
```

失败的返回：

```
{
"error": {
"reason": "table ctsdb_test already exist",
"type": "metric_exception"
},
"status": 201
}
```

查询 metric

最近更新时间：2019-10-18 16:30:39

获取所有 metric

请求地址

地址为实例的 IP 和 PORT，可从控制台获取到，例如10.13.20.15:9200。

请求路径和方法

路径：`/_metrics`
方法：GET

请求参数

无

请求内容

无

返回内容

需要通过 error 字段判断请求是否成功，若返回内容有 error 字段则请求失败，具体错误详情请参照 error 字段描述。

CURL 示例说明

请求：
`curl -u root:le201909 -X GET 172.16.345.14:9201/_metrics`

返回：

```
{
"result":
{
"metrics":
[
"ctsdb_test",
"ctsdb_test1"
]
},
"status": 200
}
```

获取特定 metric

请求地址

地址为实例的 IP 和 PORT，可从控制台获取到，例如10.13.20.15:9200。

请求路径和方法

路径：`/_metric/${metric_name}`，`${metric_name}` 为metric的名称。
方法：GET

请求参数

无

请求内容

无

返回内容

需要通过 error 字段判断请求是否成功，若返回内容有 error 字段则请求失败，具体错误详情请参照 error 字段描述。

CURL 示例说明

请求：

```
curl -u root:le201909 -X GET 172.16.345.14:9201/_metric/ctsdb_test
```

返回：


```
{
"result":
{
"ctsdb_test":
{
"tags":
{
"region": "string"
},
"time":
{
"name": "timestamp",
"format": "epoch_second"
},
"fields":
{
"cpuUsage": "float"
},
"options":
{
"expire_day": 7,
"refresh_interval": "10s",
"number_of_shards": 5
}
}
},
"status": 200
}
```

更新 metric

最近更新时间：2019-10-18 16:30:51

请求地址

地址为实例的 IP 和 PORT，可从控制台获取到，例如10.13.20.15:9200。

请求路径和方法

路径：`/_metric/${metric_name}/update`，`${metric_name}` 为 metric 的名称。
方法：PUT

请求参数

无

请求内容

字段 tags、time、fields、options 均为 map 类型的，选填，格式请参考 API 新建 metric。具体要求如下：
tags：允许新增维度字段和修改已存在的维度字段类型，不会删除原有字段。
time：name 不能修改，format 允许修改。
fields：允许新增指标字段和修改已存在的指标字段类型，不会删除原有字段。
options 属性如下：

属性名称	必选	类型	描述
expire_day	否	integer	数据过期时间，取值范围为非零整数，过期后数据自动清理，缺省情况下永不过期
refresh_interval	否	string	数据刷新频率，写入的数据从内存刷新到磁盘后可查询，默认为10秒
number_of_shards	否	integer	表分片数，取值范围为正整数，小表可忽略，大表按照一个分片至多25G设置分片数，默认为3
number_of_replicas	否	integer	副本数，取值范围为非负整数，例如一主一副为1，默认为1
rolling_period	否	integer	子表时长（单位：天），取值范围为非零整数，为了方便做数据过期清理和提高查询效率，根据特定时间间隔划分子表，缺省情况下由数据过期时间决定
max_string_length	否	integer	自定义字符串类型的值最大可支持的长度，取值范围为正整数，最大为2^31 - 1，默认为256
default_date_format	否	string	自定义维度列和指标列 date 类型的格式，默认为 strict_date_optional_time 或 epoch_millis
indexed_fields	否	array	指定指标列中需要保留索引的字段，可指定多个，以数组形式指定
default_type	否	string	指定新增字段的默认类型。可选项为 tag、field，系统默认值为 tag

说明：

- 由于历史数据不可被修改，更新字段后 metric 信息不会立即变更，需要等待下一个子表产生。如果需要确认更新操作是否成功，可通过 `GET /_metric/${metric_name}?v` 接口进行确认。
- 只有为 short、integer、float 的字段才允许修改类型。其中 short 类型可被修改为 integer 和 long 类型；integer 类型可被修改为 long 类型；float 类型可被修改为 double 类型。

返回内容

需要通过 error 字段判断请求是否成功，若返回内容有 error 字段则请求失败，具体错误详情请参照 error 字段描述。

CURL 示例说明

请求：

```
curl -u root:le201909 -X PUT 172.16.345.14:9201/_metric/ctsdb_test/update -d'
{
"tags":{
"set":"string"
},
"time":{
"name":"timestamp",
"format":"epoch_second"
},
"fields":{
"diskUsage":"float"
},
"options":{
"expire_day":15,
"number_of_shards":10
}
}'
```

返回：

```
{
"acknowledged": true,
"message": "update ctsdb metric test111 success!"
}
```

删除 metric 字段

最近更新时间：2019-10-18 16:31:10

请求地址

地址为实例的 IP 和 PORT，可从控制台获取到，例如10.13.20.15:9200。

请求路径和方法

路径： `/_metric/${metric_name}/delete` ， `${metric_name}` 为 metric 的名称。
方法：PUT

请求参数

无

请求内容

参数名称	必选	类型	描述
tags	否	Array	枚举需要删除的维度字段，如 `"tags": ["ip"]`
fields	否	Array	枚举需要删除的指标字段，如 `"fields": ["diskUsage"]`

说明：
由于历史数据不可被修改，删除字段后 metric 信息不会立即变更，需要等待下一个子表产生。如果需要确认删除操作是否成功，可通过 `GET /_metric/${metric_name}?v` 接口进行确认。

返回内容

需要通过 error 字段判断请求是否成功，若返回内容有 error 字段则请求失败，具体错误详情请参照 error 字段描述。

CURL 示例说明

请求：

```
curl -u root:le201909 -X PUT 172.16.345.14:9201/_metric/ctsdb_test1/delete -d'
{
"tags": ["ip"],
"fields": ["cpu"]
}'
```

返回：

```
{
"acknowledged": true,
"message": "update ctsdb_test1 metric success!"
}
```

删除 metric

最近更新时间：2019-10-18 16:31:20

请求地址

地址为实例的 IP 和 PORT，可从控制台获取到，例如10.13.20.15:9200。

请求路径和方法

路径：`/_metric/${metric_name}`，`${metric_name}` 为需要删除的 metric 的名称。
方法：DELETE

请求参数

无

请求内容

无

返回内容

需要通过 error 字段判断请求是否成功，若返回内容有 error 字段则请求失败，具体错误详情请参照 error 字段描述。

CURL 示例说明

请求：
`DELETE /_metric/ctsdb_test1`
请求：
`curl -u root:le201909 -X DELETE 172.16.345.14:9201/_metric/ctsdb_test1`

返回：

```
{
"acknowledged": true,
"message": "delete metric ctsdb_test1 success!"
}
```

查询数据

最近更新时间：2019-10-18 16:31:28

请求地址

地址为实例的 IP 和 PORT，可从控制台获取到，例如10.13.20.15:9200。

请求路径和方法

路径： `${metric_name}/_search` ， `${metric_name}` 为 metric 的名称。
方法：GET

请求参数

可设置写入数据时的 _routing 参数值来提高查询效率，具体请参见示例。

请求内容

查询主要有普通查询和聚合查询，查询请求完全兼容 Elasticsearch api，具体请求内容请参照示例。

返回内容

需要通过 error 字段判断请求是否成功，若返回内容有 error 字段则请求失败，具体错误详情请参照 error 字段描述。

CURL 示例说明

具体查询示例请参考 常用查询示例。

批量查询数据

最近更新时间：2019-10-18 16:31:40

请求地址

地址为实例的 IP 和 PORT，可从控制台获取到，例如10.13.20.15:9200。

请求路径和方法

路径： `/_msearch`
方法：GET

请求参数

可通过设置写入数据时的 _routing 参数值来提高查询效率，具体请参照示例。也可通过设置 filter_path 参数来过滤并简化返回结果，其参数值为按照逗号分隔的多个 json 路径，其中单个 json 路径用点号连接，可以使用 * 通配符来适配任意字符， ** 通配符可以用来匹配任意路径，可以使用 - 前缀来去除某些返回字段。例如，responses.*shards.f* 表示 responses 下 _shards 中的以 f 开头的字段； responses.***.score 表示 responses 中所有 _score 的字段。

请求内容

查询主要有普通查询和聚合查询，查询请求完全兼容 Elasticsearch api，具体请求内容请参照示例。

返回内容

需要通过 error 字段判断请求是否成功，若返回内容有 error 字段则请求失败，具体错误详情请参照 error 字段描述。这里列举几个查询通用返回字段，详情见下表，若需要简化查询返回结果，可设置 filter_path 参数。

字段名称	描述
hits	返回匹配的查询结果，其中 total 字段表示匹配到的数据条数。里面的 hits 字段为数组结构，若无指定，则包含所有查询结果的前十条。hits 数组里的每个结果中包含 _index（查询涉及到的子表），若查询中指定了 docvalue_fields，则会返回 fields 字段具体指明每个字段的值。
took	整个查询耗费的毫秒数。
_shards	参与查询的分片数。其中 total 标识总分片数，successful 标识执行成功的数量，failed 标识执行失败的数量，skipped 标识跳过执行的分片数。
timed_out	查询是否超时，取值有 false 和 true。
status	查询返回的状态码，2XX 代表成功。

CURL 示例说明

无 filter_path 参数的请求

请求：

```
curl -u root:le201909 -X GET 172.16.345.14:9201/_msearch -d'
{"index" : "amyli_weather","routing":"host_20"}
{"query" : {"match_all" : {}}, "from" : 0, "size" : 10}
{"index" : "ctsdb_test2017","routing":"host_100"}
{"query" : {"match_all" : {}}, "from" : 0, "size" : 10}
'
```

返回：

```
{
"responses":
[
{
"took": 0,
"timed_out": false,
"_shards": {
"total": 0,
"successful": 0,
"skipped": 0,
"failed": 0
},
"hits": {
"total": 0,
"max_score": 0,
"hits": []
},
"status": 200
},
{
"took": 1,
"timed_out": false,
"_shards": {
"total": 3,
"successful": 3,
"skipped": 0,
"failed": 0
},
"hits": {
"total": 1,
"max_score": 1,
"hits": [
{
"_index": "ctsdb_test2017@-979200000_30",
"_type": "doc",
"_id": "AV_8fBhlUAkC9PF9L-2t",
"_score": 1
}
]
},
"status": 200
}
]
}
```


有 filter_path 参数的请求

请求：

```
curl -u root:le201909 -X GET 172.16.345.14:9201/_msearch?filter_path=responses.\*\*.hits,responses._shards.f\*,-responses.\*\*._score -d'
{"index" : "amyli_weather","routing":"host_20"}
{"query" : {"match_all" : {}}, "from" : 0, "size" : 10}
{"index" : "ctsdb_test2017","routing":"host_100"}
{"query" : {"match_all" : {}}, "from" : 0, "size" : 10}
'
```

返回：


```
{
"responses": [
{
"_shards": {
"failed": 0
}
},
{
"_shards": {
```

```
"failed": 0
},
"hits": {
"hits": [
{
"_index": "ctsdb_test2017@-979200000_30",
"_type": "doc",
"_id": "AV_8fBhIUAkC9PF9L-2t"
}
]
}
}
]
}
```


说明：
如果 filter_path 参数同时设置了匹配条件和去除条件，则返回结果先执行去除条件再执行匹配条件。

常用查询示例

最近更新时间：2019-10-18 16:32:51

此篇文章主要介绍 CTSDB 中常用查询及关键字的使用，并给出简单 CURL 示例。所有示例使用的 metric 结构如下所示：

```
{
"ctsdb_test" : {
"tags" : {
"region" : "string"
},
"time" : {
"name" : "timestamp",
"format" : "epoch_millis"
},
"fields" : {
"cpuUsage" : "float",
"diskUsage" : "string",
"dcpuUsage" : "integer"
},
"options" : {
"expire_day" : 7,
"refresh_interval" : "10s",
"number_of_shards" : 5
}
}
}
```

组合查询

这里的组合查询即可以用于单查询也可用于复合查询。查询体中的 query 关键字通过 Query DSL(Domain Specific Language) 来定义查询条件。下文会先介绍如何构造过滤条件再介绍怎样组合过滤条件和如何处理返回结果集。

常用过滤条件

1. Range

Range 即区间查询，Range 查询支持的字段类型包括 string、long、integer、short、double、float、date。Range 查询可包含的参数如下表所示：

参数名称	描述
gte	大于或等于
gt	大于
lte	小于或等于
lt	小于

> 说明：
> 当 Range 查询涉及时间类型时，可通过 format 参数来指定时间格式。具体时间格式请参考 新建 metric。

时间范围查询的 CURL 示例：

```
curl -u root:le201909 -X GET 172.16.345.14:9201/ctsdb_test/_search -d'
{
"query": {
"range": {
"timestamp": {
"gte": "01/01/2018",
"lte": "03/01/2018",
"format": "MM/dd/yyyy",
```

```
"time_zone":"+08:00"
}
}
}
}'
```

数字范围查询 CURL 示例：

```
curl -u root:le201909 -X GET 172.16.345.14:9201/ctsdb_test/_search -d'
{
"query": {
"range": {
"cpuUsage": {
"gte": 1.0,
"lte": 10.0
}
}
}
}'
```

2. Terms

Terms 关键字用于查询时匹配特定字段，其值需要用中括号包裹。

CURL 示例说明：

```
curl -u root:le201909 -X GET 172.16.345.14:9201/ctsdb_test/_search -d'
{
"query": {
"terms": {
"region": ["sh", "bj"]
}
}
}'
```

过滤条件组合

复合查询常使用 Bool 关键字来组合多个查询条件，常见的用于 Bool 查询的组合关键字有 filter（类似于 AND）、must_not（类似于 NOT）、should（类似于 OR）。为提高查询性能，请务必加上 time 字段的 range 查询，且 time 字段，不论查询时以何种方式写入，返回值统一为 epoch_millis 格式。query-bool-filter 的组合形式可显著提升查询性能，请务必采用这种查询方式。

1. AND 条件 CURL 示例说明

```
curl -u root:le201909 -X GET 172.16.345.14:9201/ctsdb_test/_search -d'
{
"query": {
"bool": {
"filter": [
{
"range": {
"timestamp": {
"format": "yyyy-MM-dd HH:mm:ss",
"gte": "2017-11-06 23:00:00",
"lt": "2018-03-06 23:05:00",
"time_zone":"+08:00"
}

}
},
{
"terms": {
"region": ["sh"]
}
},
{
"terms": {
"cpuUsage": ["2.0"]
```

```
}
}
]
}
},
"docvalue_fields": [
"cpuUsage",
"timestamp"
]
}'
```

说明：
此查询条件类似于 timestamp>='2017-11-06 23:00:00' AND timestamp<'2018-03-06 23:05:00' AND region=sh AND cpuUsage=2.0。

2. OR 条件 CURL 示例说明

```
curl -u root:le201909 -X GET 172.16.345.14:9201/ctsdb_test/_search -d'
{
"query": {
"bool": {
"filter": [
{
"range": {
"timestamp": {
"format": "yyyy-MM-dd HH:mm:ss",
"gte": "2017-11-06 23:00:00",
"lt": "2018-11-06 23:05:00",
"time_zone":"+08:00"
}
}
},
{
"term": {
"region": "gz"
}
}
],
"should": [
{
"terms": {
"cpuUsage": ["2.0"]
}
},
{
"terms": {
"cpuUsage": ["2.5"]
}
}
],
"minimum_should_match": 1
}
},
"docvalue_fields": [
"cpuUsage",
"timestamp"
]
}'
```

说明：
此查询条件类似于 timestamp>='2017-11-06 23:00:00' AND timestamp<'2018-11-06 23:05:00' AND region='gz' AND (cpuUsage=2.0 or cpuUsage=2.5)。
minimum_should_match 参数的意义在于设置 cpuUsage=2.0 和 cpuUsage=2.5 至少匹配的个数，系统默认 minimum_should_match 为0。

3.NOT 条件 CURL 示例说明

```
curl -u root:le201909 -X GET 172.16.345.14:9201/ctsdb_test/_search -d'
{
"query": {
"bool": {
"filter": [
{
"range": {
"timestamp": {
"format": "yyyy-MM-dd HH:mm:ss",
"gte": "2017-11-06 23:00:00",
"lt": "2018-11-06 23:05:00",
"time_zone":"+08:00"
}

}
},
{
"terms": {
"region": ["gz"]
}
}
],
"must_not": [
{
"terms": {
"cpuUsage": ["2.0"]
}
}
]
}
},
"docvalue_fields": [
"cpuUsage",
"timestamp"
]
}'
```

说明：
此查询条件类似于 timestamp>=2017-11-06 23:00:00 AND timestamp<2018-11-06 23:05:00 AND region='gz' AND cpuUsage !=2.0。

返回结果集处理

1. From/Size

通过设置 From 和 Size 关键字可对查询结果进行分页。From 关键字定义了查询结果中第一条数据的偏移量，Size 关键字配置返回结果的最大条数。From 系统默认为 0，Size 系统默认为 10，From 与 Size 的总和系统默认不能超过 65536。若想要更大的返回结果，请参考本文的 scroll 关键字查询。

CURL 示例说明：

```
curl -u root:le201909 -X GET 172.16.345.14:9201/ctsdb_test/_search -d'
{
"from": 0,
"size": 5,
"query": {
"bool": {
"filter": {
"range": {
"timestamp": {
"gte": "01/01/2018",
"lte": "03/01/2018",
"format": "dd/MM/yyyy",
"time_zone":"+08:00"
}
```

```
}
},
"must_not": {
"terms": {
"region": ["bj"]
}
}
}
}
}'
```

2. Scroll

Scroll 可以理解为关系型数据库里的 cursor，因此 scroll 并不适合用来做实时搜索，而更适用于后台批处理任务。
可以把 scroll 分为初始化和遍历两个阶段，初始化时将所有符合搜索条件的搜索结果缓存起来，类似于快照，在遍历时，从这个快照里取数据。注意，在 Scroll 初始化后对 metric 的插入、删除、更新都不会影响遍历结果。
初始化阶段可通过 size 关键字指定返回结果集的大小，size 默认值为10，最大值为65536；初始化阶段可通过 query 关键字指定查询条件；初始化阶段可通过 docvalue_fields 关键字指定返回字段。初始化阶段和遍历阶段都返回 `_scroll_id` 字段，后续遍历可以通过指定前一次遍历返回的 `_scroll_id` 来进行新的遍历，直到返回结果为空；初始化和遍历两个阶段都可以通过指定 scroll 参数来设定遍历的上下文环境保留时间，过期后 scroll_id 将变得无效。时间格式如下表所示：

格式	描述
d	days
h	hours
m	minutes
s	seconds
ms	milliseconds
micros	microseconds
nanos	nanoseconds

CURL 示例说明：

scroll 初始化：

```
curl -u root:le201909 -X POST 172.16.345.14:9201/ctsdb_test/_search?scroll=1m -d'
{
"size":5,
"query": {
"bool": {
"filter": [
{
"terms": {
"region": ["gz"]
}
}
]
}
},
"docvalue_fields": [
"cpuUsage",
"region",
"timestamp"
]
}'
```

scroll 初始化返回：

```
{
"_scroll_id": "DnF1ZXJ5VGhlbkZldGNoAwAAAAADrOFFm5YSEhnMjdnUWNPcndHS1k5Wjc3bHcAAAAAAAz_1RZiRkZTcGp4dFRXR18xMGtzSmhEUFJRAA
AAAAAP5vQWOXFOR29lc0hROHFWMmFGTkVmSkxmZw==",
"took": 10641,
```

```
"timed_out": false,
"_shards": {
"total": 3,
"successful": 3,
"skipped": 0,
"failed": 0
},
"hits": {
"total": 1592072666,
"max_score": 0.65708643,
"hits": [
{
"_index": "ctsdb_test@0_-1",
"_type": "doc",
"_id": "oyylNU0U65cZjByyt7sW_JmPPgAACY4Bh0",
"_score": 0.65708643,
"_routing": "354d14eb",
"fields": {
"region": [
"gz"
],
"cpuUsage": [
"2.0"
],
"timestamp": [
1509909300000
]
}
},
{
"_index": "ctsdb_test@0_-1",
"_type": "doc",
"_id": "oyykFN0yd1d9NDPfzjRdrJ8whQAACYsBqc",
"_score": 0.65708643,
"_routing": "14dd3277",
"fields": {
"region": [
"gz"
],
"cpuUsage": [
"1.8"
],
"timestamp": [
1509908340000
]
}
},
{
"_index": "ctsdb_test@0_-1",
"_type": "doc",
"_id": "oyylHLp9jl_3sF4N2rnh67h4SgAABHIBso",
"_score": 0.65708643,
"_routing": "1cba7d8e",
"fields": {
"region": [
"gz"
],
"cpuUsage": [
"2.5"
],
"timestamp": [
1509909720000
]
}
},
{
"_index": "ctsdb_test@0_-1",
"_type": "doc",
```

```
"_id": "oyylH2JsOKnFHGUinQ7jM-ZwkgAAAvcBso",
"_score": 0.65708643,
"_routing": "1f626c38",
"fields": {
"region": [
"gz"
],
"cpuUsage": [
"2.1"
],
"timestamp": [
1509909720000
]
}
},
{
"_index": "ctsdb_test@0_-1",
"_type": "doc",
"_id": "oyylHLp9jl_3sF4N2rnh67h4SgAABGsBso",
"_score": 0.65708643,
"_routing": "1cba7d8e",
"fields": {
"region": [
"gz"
],
"cpuUsage": [
"2.0"
],
"timestamp": [
1509909720000
]
}
}
]
}
}
```


scroll 遍历：

```
curl -u root:le201909 -X POST 172.16.345.14:9201/_search/scroll -d'
{
"scroll" : "1m",
"scroll_id" : "DnF1ZXJ5VGhlbkZldGNoAwAAAAAADrOFFm5YSEhnMjdnUWNPcndHS1k5Wjc3bHcAAAAAAAAz_1RZiRkZTcGp4dFRXR18xMGtzSmhEUFJRAAAAAAAAP5vQWOXFOR29lc0hROHFWMmFGTkVmSkxmZw=="
}'
```

> 说明：
> 此请求中的 scroll_id 是 scroll 初始化返回的 *scroll*id 值。而下一次遍历则需要将 scroll_id 参数调整为上一次遍历返回的 *scroll*id 值，即每次请求中的 scroll_id 参数是上一次请求返回的 *scroll*id 值，直到返回结果为空，遍历结束。

3. Sort

Sort 关键字主要用于对查询结果进行排序。排序方式有 asc 和 desc 两种，CTSDB 对于用户自定义字段的默认排序方式为 asc。排序模式有 min、max、sum、avg、median。其中 sum、avg、median 只适用于类型为 array 并存储数字的字段。

CURL 示例说明：

```
curl -u root:le201909 -X GET 172.16.345.14:9201/ctsdb_test/_search -d'
{
"query": {
"bool": {
"must": {
"range": {
"timestamp": {
```

```
"gte": "01/01/2018",
"lte": "03/01/2018",
"format": "MM/dd/yyyy",
"time_zone":"+08:00"
}
}
}
}
},
"sort": [
{
"cpuUsage": {
"order": "asc",
"mode": "min"
}
},
{
"timestamp": {
"order": "asc"
}
},
"diskUsage"
]
}'
```

4. docvalue_fields

docvalue_fields 关键词指定需要返回的字段名称，需要以数组的形式指定。

CURL 示例说明：

```
curl -u root:le201909 -X GET 172.16.345.14:9201/ctsdb_test/_search -d'
{
"query": {
"terms": {
"region": ["sh", "bj"]
}
},
"docvalue_fields": ["timestamp", "cpuUsage"]
}'
```

聚合查询

agg 关键字主要用于构造聚合查询。用户可到返回的 aggregations 字段取聚合结果。聚合返回字段说明详见下表。如果只关注聚合结果，请在查询时设置 size 参数为0。

字段名称	描述
hits	返回匹配的查询结果，其中 total 字段表示参与聚合的数据条数。里面的 hits 字段为数组结构，若无指定，则包含所有查询结果的前十条。hits 数组里的每个结果中包含 _index（查询涉及到的子表），若查询中指定了 docvalue_fields，则会返回 fields字段具体指明每个字段的值。
took	整个查询耗费的毫秒数。
_shards	参与查询的分片数。其中 total 标识总分片数，successful 标识执行成功的数量，failed 标识执行失败的数量，skipped标识跳过执行的分片数。
timed_out	查询是否超时，取值有false和true。
aggregations	聚合返回结果。

下面列举几种常用的聚合方式。

普通聚合

普通聚合需要指定聚合名称、聚合方式（常用的聚合方式有 min、max、avg、value_count、sum 等）和聚合所作用的字段。

CURL 示例说明：

```
curl -u root:le201909 -X GET 172.16.345.14:9201/ctsdb_test/_search -d'
{
"size":0,
"query": {
"terms": {
"region": ["sh", "bj"]
}
},
"aggs": {
"myname": {
"max": {
"field": "cpuUsage"
}
}
}
}'
```

说明：
上例中对字段 cpuUsage 做 max 聚合（用户也可指定 min、avg 等），返回结果取别名 myname（用户可任意指定该名称）。

返回结果：

```
{
"took": 1,
"timed_out": false,
"_shards": {
"total": 20,
"successful": 20,
"skipped": 0,
"failed": 0
},
"hits": {
"total": 7,
"max_score": 0,
"hits": []
},
"aggregations": {
"myname": {
"value": 4
}
}
}
```

terms 聚合

terms 聚合主要用于查询某个字段的所有唯一值以及该值的个数，用户可指定唯一值返回时的排序规则，以及返回结果数，并且可对参与聚合的数据字段进行模糊或者精确匹配，详情请参考如下示例,使用 filter_path 参数可定制返回结果字段，使用方法请参考 批量查询数据。

CURL 示例说明：

请求：

```
curl -u root:le201909 -X GET 172.16.345.14:9201/ctsdb_test/_search?filter_path=aggregations -d'
{
"aggs": {
"myname": {
"terms": {"field":"region"}
}
}
}'
```

返回：

```
{
"aggregations": {
"myname": {
"doc_count_error_upper_bound": 0,
"sum_other_doc_count": 0,
"buckets": [
{
"key": "sh",
"doc_count": 10
},
{
"key": "Motor_sports",
"doc_count": 6
},
{
"key": "gz",
"doc_count": 3
},
{
"key": "bj",
"doc_count": 2
},
{
"key": "cd",
"doc_count": 2
},
{
"key": "Winter_sports",
"doc_count": 1
},
{
"key": "water_sports",
"doc_count": 1
}
]
}
}
}
```

说明：
上例表示查看 ctsdb_test 的 region 字段中所有的唯一值及其出现的次数。分析返回字段 aggregations 中的 buckets 字段发现，region 字段共有7种类型的值，分别是 sh、Motor_sports、gz、bj、cd、Winter_sports、water_sports， 并且返回字段通过 doc_count 指明了每种值的个数。

用户可通过 size 字段指定返回的唯一值的个数，例如某字段名为 region，其唯一值有7个，可通过设置 size 字段为5指定只返回前5个，具体请参考示例。

请求：

```
curl -u root:le201909 -X GET 172.16.345.14:9201/ctsdb_test/_search?filter_path=aggregations -d'
{
"aggs": {
"myname": {
"terms": {
"field":"region",
"size":5
}
}
}
}'
```

返回：

```
{
"aggregations": {
```

```
"myname": {
"doc_count_error_upper_bound": 0,
"sum_other_doc_count": 2,
"buckets": [
{
"key": "sh",
"doc_count": 10
},
{
"key": "Motor_sports",
"doc_count": 6
},
{
"key": "gz",
"doc_count": 3
},
{
"key": "bj",
"doc_count": 2
},
{
"key": "cd",
"doc_count": 2
}
]
}
}
}
```

用户可对返回结果进行排序，详情请参考如下示例。

1.请求（返回结果按照唯一值的个数降序排列）：

```
curl -u root:le201909 -X GET 172.16.345.14:9201/ctsdb_test/_search?filter_path=aggregations -d'
{
"aggs": {
"myname": {
"terms": {
"field":"region",
"order":{"_count":"desc"}
}
}
}
}'
```

返回：

```
{
"aggregations": {
"myname": {
"doc_count_error_upper_bound": 0,
"sum_other_doc_count": 0,
"buckets": [
{
"key": "sh",
"doc_count": 10
},
{
"key": "Motor_sports",
"doc_count": 6
},
{
"key": "gz",
"doc_count": 3
},
{
"key": "bj",
"doc_count": 2
```

```
},
{
"key": "cd",
"doc_count": 2
},
{
"key": "Winter_sports",
"doc_count": 1
},
{
"key": "water_sports",
"doc_count": 1
}
]
}
}
}
```

2.请求（返回结果按照唯一值的字母升序进行排列）：

```
curl -u root:le201909 -X GET 172.16.345.14:9201/ctsdb_test/_search?filter_path=aggregations -d'
{
"aggs": {
"myname": {
"terms": {
"field":"region",
"order":{"_term":"asc"}
}
}
}
}'
```

返回：

```
{
"aggregations": {
"myname": {
"doc_count_error_upper_bound": 0,
"sum_other_doc_count": 0,
"buckets": [
{
"key": "Motor_sports",
"doc_count": 6
},
{
"key": "Winter_sports",
"doc_count": 1
},
{
"key": "bj",
"doc_count": 2
},
{
"key": "cd",
"doc_count": 2
},
{
"key": "gz",
"doc_count": 3
},
{
"key": "sh",
"doc_count": 10
},
{
"key": "water_sports",
"doc_count": 1
```

```
}
]
}
}
}
```

用户可设置正则表达式对参与 terms 聚合的数据字段进行模糊匹配或者指定字段进行精确匹配，详情请参考示例。

模糊匹配示例：
请求（只针对 region 字段中有 sport 并且不是以 water_ 开头的数据进行聚合并返回结果）：

```
curl -u root:le201909 -X GET 172.16.345.14:9201/ctsdb_test/_search?filter_path=aggregations -d'
{
"aggs": {
"myname": {
"terms": {
"field":"region",
"include" : ".*sport.*",
"exclude" : "water_.*"
}
}
}
}'
```

返回：

```
{
"aggregations": {
"myname": {
"doc_count_error_upper_bound": 0,
"sum_other_doc_count": 0,
"buckets": [
{
"key": "Motor_sports",
"doc_count": 6
},
{
"key": "Winter_sports",
"doc_count": 1
}
]
}
}
}
```

精确匹配示例
请求（region_zone 指定值为"sh", "bj","cd","gz"的 region 字段进行聚合，而 region_sports 指定值不为"sh", "bj","cd","gz"的 region 字段进行聚合）

```
curl -u root:le201909 -X GET 172.16.345.14:9201/ctsdb_test/_search?filter_path=aggregations -d'
{
"aggs" : {
"region_zone" : {
"terms" : {
"field" : "region",
"include" : ["sh", "bj","cd","gz"]
}
},
"region_sports" : {
"terms" : {
"field" : "region",
"exclude" : ["sh", "bj","cd","gz"]
}
}
}
}'
```

返回：

```
{
"aggregations": {
"region_sport": {
"doc_count_error_upper_bound": 0,
"sum_other_doc_count": 0,
"buckets": [
{
"key": "Motor_sports",
"doc_count": 6
},
{
"key": "Winter_sports",
"doc_count": 1
},
{
"key": "water_sports",
"doc_count": 1
}
]
},
"region_zone": {
"doc_count_error_upper_bound": 0,
"sum_other_doc_count": 0,
"buckets": [
{
"key": "sh",
"doc_count": 10
},
{
"key": "gz",
"doc_count": 3
},
{
"key": "bj",
"doc_count": 2
},
{
"key": "cd",
"doc_count": 2
}
]
}
}
}
```

Date Histogram 聚合

Date Histogram 主要对日期做直方图聚合。
CURL 示例说明：

```
curl -u root:le201909 -X GET 172.16.345.14:9201/ctsdb_test/_search -d'
{
"query": {
"terms": {
"region": ["sh", "bj"]
}
},
"aggs": {
"time_1h_agg": {
"date_histogram": {
"field": "timestamp",
"interval": "1h"
},
"aggs": {
"avgCpuUsage": {
"avg": {
```

```
"field": "cpuUsage"
}
}
}
}
}
}'
```

返回：


```
{
"took": 5,
"timed_out": false,
"_shards": {
"total": 20,
"successful": 20,
"skipped": 0,
"failed": 0
},
"hits": {
"total": 6,
"max_score": 0.074107975,
"hits": [
{
"_index": "ctsdb_test@1520092800000_3",
"_type": "doc",
"_id": "AWH2QtGR5xcjRaw2ETf-",
"_score": 0.074107975,
"_routing": "sh"
},
{
"_index": "ctsdb_test@1520092800000_3",
"_type": "doc",
"_id": "AWH2QtGR5xcjRaw2ETf_",
"_score": 0.074107975,
"_routing": "sh"
},
{
"_index": "ctsdb_test@1520092800000_3",
"_type": "doc",
"_id": "AWH2Q5Xr5xcjRaw2ETgA",
"_score": 0.074107975,
"_routing": "sh"
},
{
"_index": "ctsdb_test@1520092800000_3",
"_type": "doc",
"_id": "AWH2Q5Xr5xcjRaw2ETgB",
"_score": 0.074107975,
"_routing": "sh"
},
{
"_index": "ctsdb_test@1520092800000_3",
"_type": "doc",
"_id": "AWH2RGfF5xcjRaw2ETgC",
"_score": 0.074107975,
"_routing": "sh"
},
{
"_index": "ctsdb_test@1520092800000_3",
"_type": "doc",
"_id": "AWH2RGfF5xcjRaw2ETgD",
"_score": 0.074107975,
"_routing": "sh"
}
]
},
```

```
"aggregations": {
"time_1h_agg": {
"buckets": [
{
"key_as_string": "1520222400",
"key": 1520222400000,
"doc_count": 1,
"avgCpuUsage": {
"value": 2.5
}
},
{
"key_as_string": "1520226000",
"key": 1520226000000,
"doc_count": 0,
"avgCpuUsage": {
"value": null
}
},
{
"key_as_string": "1520229600",
"key": 1520229600000,
"doc_count": 0,
"avgCpuUsage": {
"value": null
}
},
{
"key_as_string": "1520233200",
"key": 1520233200000,
"doc_count": 0,
"avgCpuUsage": {
"value": null
}
},
{
"key_as_string": "1520236800",
"key": 1520236800000,
"doc_count": 0,
"avgCpuUsage": {
"value": null
}
},
{
"key_as_string": "1520240400",
"key": 1520240400000,
"doc_count": 0,
"avgCpuUsage": {
"value": null
}
},
{
"key_as_string": "1520244000",
"key": 1520244000000,
"doc_count": 0,
"avgCpuUsage": {
"value": null
}
},
{
"key_as_string": "1520247600",
"key": 1520247600000,
"doc_count": 1,
"avgCpuUsage": {
"value": 2
}
},
{
```

```
"key_as_string": "1520251200",
"key": 1520251200000,
"doc_count": 4,
"avgCpuUsage": {
"value": 2.25
}
}
]
}
}
}
```

说明：
上例是对字段 cpuUsage 进行粒度为1小时的 date_histogram 聚合。返回结果中总的聚合名称为 time_1h_agg（用户可任意指定该名称），每个时间分段内的聚合名称为 avgCpuUsage（用户可任意指定该名称）。interval 字段可选的时间粒度有 year、quarter、 month、week、day、hour、minute、second，用户也可将时间粒度用时间单位来表示，例如1y代表1year,1h代表1hour。系统不支持小数粒度的时间单位，例如1.5h您需要转换为90m。

Percentiles 聚合

Percentiles 聚合即百分位聚合。百分位可以任意指定，系统默认的百分位是 1、5、25、50、75、95、99。

CURL 示例说明：

```
curl -u root:le201909 -X GET 172.16.345.14:9201/ctsdb_test/_search -d'
{
"query": {
"terms": {
"region": ["sh", "bj"]
}
},
"aggs":
{
"myname":
{
"percentiles":
{
"field": "cpuUsage",
"percents": [1,25,50,70,99]
}
}
}
}'
```

返回：

```
{
"took": 18,
"timed_out": false,
"_shards": {
"total": 20,
"successful": 20,
"skipped": 0,
"failed": 0
},
"hits": {
"total": 6,
"max_score": 0.074107975,
"hits": [
{
"_index": "ctsdb_test@1520092800000_3",
"_type": "doc",
"_id": "AWH2QtGR5xcjRaw2ETf-",
"_score": 0.074107975,
"_routing": "sh"
```

```
},
{
"_index": "ctsdb_test@1520092800000_3",
"_type": "doc",
"_id": "AWH2QtGR5xcjRaw2ETf_",
"_score": 0.074107975,
"_routing": "sh"
},
{
"_index": "ctsdb_test@1520092800000_3",
"_type": "doc",
"_id": "AWH2Q5Xr5xcjRaw2ETgA",
"_score": 0.074107975,
"_routing": "sh"
},
{
"_index": "ctsdb_test@1520092800000_3",
"_type": "doc",
"_id": "AWH2Q5Xr5xcjRaw2ETgB",
"_score": 0.074107975,
"_routing": "sh"
},
{
"_index": "ctsdb_test@1520092800000_3",
"_type": "doc",
"_id": "AWH2RGfF5xcjRaw2ETgC",
"_score": 0.074107975,
"_routing": "sh"
},
{
"_index": "ctsdb_test@1520092800000_3",
"_type": "doc",
"_id": "AWH2RGfF5xcjRaw2ETgD",
"_score": 0.074107975,
"_routing": "sh"
}
]
},
"aggregations": {
"myname": {
"values": {
"1.0": 2,
"25.0": 2,
"50.0": 2.25,
"70.0": 2.5,
"99.0": 2.5
}
}
}
}
```


说明：
上例中是对字段 cpuUsage 做 percentiles 聚合，选取的百分位为 1,25,50,70,99。聚合返回结果的别名是 myname（用户可任意指定该名称）。

Cardinality 聚合

Cardinality 聚合主要用于统计去重后的数量。默认情况下，当聚合结果小于等于3000时，Cardinality 返回的结果是精确值，大于3000时，为近似值。

CURL 示例说明：

```
curl -u root:le201909 -X GET 172.16.345.14:9201/ctsdb_test/_search -d'
{
"query": {
"terms": {
"region": ["sh", "bj"]
```

```
}
},
"aggs":
{
"myname":
{
"cardinality":
{
"field": "cpuUsage"
}
}
}
}'
```

返回：

```
{
"took": 15,
"timed_out": false,
"_shards": {
"total": 20,
"successful": 20,
"skipped": 0,
"failed": 0
},
"hits": {
"total": 6,
"max_score": 0.074107975,
"hits": [
{
"_index": "ctsdb_test@1520092800000_3",
"_type": "doc",
"_id": "AWH2QtGR5xcjRaw2ETf-",
"_score": 0.074107975,
"_routing": "sh"
},
{
"_index": "ctsdb_test@1520092800000_3",
"_type": "doc",
"_id": "AWH2QtGR5xcjRaw2ETf_",
"_score": 0.074107975,
"_routing": "sh"
},
{
"_index": "ctsdb_test@1520092800000_3",
"_type": "doc",
"_id": "AWH2Q5Xr5xcjRaw2ETgA",
"_score": 0.074107975,
"_routing": "sh"
},
{
"_index": "ctsdb_test@1520092800000_3",
"_type": "doc",
"_id": "AWH2Q5Xr5xcjRaw2ETgB",
"_score": 0.074107975,
"_routing": "sh"
},
{
"_index": "ctsdb_test@1520092800000_3",
"_type": "doc",
"_id": "AWH2RGfF5xcjRaw2ETgC",
"_score": 0.074107975,
"_routing": "sh"
},
{
"_index": "ctsdb_test@1520092800000_3",
"_type": "doc",
```

```
"_id": "AWH2RGfF5xcjRaw2ETgD",
"_score": 0.074107975,
"_routing": "sh"
}
]
},
"aggregations": {
"myname": {
"value": 2
}
}
}
```


说明：
上例的聚合是对字段 cpuUsage 做 cardinality 聚合，返回的聚合结果取别名 myname（用户可任意指定该名称）。

批量写入数据

最近更新时间：2019-10-18 16:33:13

往单个 metric 批量写入数据

此接口同时适用于批量和非批量写入数据至单个 metric 的场景，为提高写入性能，建议批量写入数据。

请求地址

地址为实例的 IP 和 PORT，可从控制台获取到，例如10.13.20.15:9200。

请求路径和方法

请求路径：`/${metric_name}/doc/_bulk` ，${metric_name}为 metric 的名称。
方法：POST

> 说明：
> doc 关键字为写入数据的 _type，为了便于以后系统做解析和升级，请务必加上 doc 关键字。

请求参数

可通过设置 filter_path 参数来过滤并简化返回结果，具体请参考 批量查询数据。

请求内容

批量写入 metric 需要一种 NDJSON 格式的结构数据，类似于如下：

```
元数据\n
需要写入的数据\n
....
元数据\n
需要写入的数据\n
```

元数据的格式如下所示：

```
{
"index" :
{
"_id" : "1", #文档 ID(选填)
"_routing": "sh" #路由值(选填)
}
}
```

写入数据的格式类似于：

```
{
"field1" : "value1",
"field2" : "value2"
}
```

> 注意：
> 写入数据时可通过设置 _routing 参数来路由数据写入的分片，该参数选填，可指定任意值。指定相同的 _routing 值可将不同数据路由至同一分片上。另外，查询时指定已设置的 _routing 参数值，系统会根据路由到指定分片查询数据，可以极大优化查询速度。请求体中最后一行需要加换行符。

返回内容

需要注意的是，批量写入数据接口与其他接口返回结果稍有不同。请优先关注 json 结果的 errors（注意不是 error）字段，如果为 false，代表所有数据写入成功；如果为 true，代表有部分写入失败，具体失败的详情可以通过 items 字段得到。

items 字段为一个数组，数组中每一个元素与写入请求一一对应，可通过每个元素是否有 error 字段判断请求是否成功，若有 error 字段则表示请求失败，具体错误内容在 error 字段内，若无 error 字段表示请求成功。

CURL 示例说明

返回成功的示例：

请求：

```
curl -u root:le201909 -X POST 172.16.345.14:9201/ctsdb_test/doc/_bulk -d'
{"index":{"_routing": "sh" }}
{"region":"sh","cpuUsage":2.5,"timestamp":1505294654}
{"index":{"_routing": "sh" }}
{"region":"sh","cpuUsage":2.0,"timestamp":1505294654}
'
```

返回：

```
{
"took": 134,
"errors": false,
"items":
[
{
"index":
{
"_index": "ctsdb_test@1505232000000_1",
"_type": "doc",
"_id": "AV_8eeo_UAkC9PF9L-2q",
"_version": 1,
"result": "created",
"_shards":
{
"total": 2,
"successful": 2,
"failed": 0
},
"created": true,
"status": 201
}
},
{
"index":
{
"_index": "ctsdb_test2@1505232000000_1",
"_type": "doc",
"_id": "AV_8eeo_UAkC9PF9L-2r",
"_version": 1,
"result": "created",
"_shards":
{
"total": 2,
"successful": 2,
"failed": 0
},
"created": true,
"status": 201
}
}
]
}
```

说明：

上面返回中的 errors 为 false，代表所有数据写入成功。items 数组标识每一条记录写入结果，与 bulk 请求中的每一条写入顺序对应。items 的单条记录中，status 为 2XX 代表此条记录写入成功，_index 标识了写入的 metric 子表，_shards 记录副本写入情况，上例中 total 表示两副本，successful 代表两副本均写入成功。

返回失败的示例：

请求：

```
curl -u root:le201909 -X POST 172.16.345.14:9201/hcbs_client_trace/doc/_bulk -d'
{"index":{"_type":"type"}}
{"vol_id":"c57e008c-0ae0-41cd-8da8-6989d0522fc6","io_type":2,"data_len":4096,"latency":3,"try_times":1,"errcode":0,"start_time":1503404266}
{"index":{"_type":"type"}}
{"vol_id":"c57e008c-0ae0-41cd-8da8-6989d0522fc6","io_type":"abc","data_len":4096,"latency":1,"try_times":1,"errcode":0,"start_time":1503404266}
'
```

返回：


```
{
"took": 7,
"errors": true,
"items": [
{
"index": {
"_index": "hcbs_client_trace",
"_type": "type",
"_id": "AWMe9r9lNifptzlWMVPT",
"_version": 1,
"result": "created",
"_shards": {
"total": 2,
"successful": 2,
"failed": 0
},
"created": true,
"status": 201
}
},
{
"index": {
"_index": "hcbs_client_trace",
"_type": "type",
"_id": "AWMe9r9lNifptzlWMVPU",
"status": 400,
"error": {
"type": "mapper_parsing_exception",
"reason": "failed to parse [io_type]",
"caused_by": {
"type": "number_format_exception",
"reason": "For input string: \"abc\""
}
}
}
}
]
}
```


说明：

上面返回中的 errors 为 true，代表部分数据写入失败。items 数组标识每一条记录写入结果，与 bulk 请求中的每一条写入顺序对应。items 的单条记录中，status 为 2XX 代表此条记录写入成功，error 字段详细给出了错误内容，_index 标识了写入的 metric 子表，_shards 记录副本写入情况，上例中 total 表示两副本，successful 代表两副本均写入成功。

往多个 metric 批量写入数据

此接口同时适用于批量和非批量写入单个 metric 或者多个 metric 的场景。为提高写入性能，建议批量写入数据。

请求地址

地址为实例的 IP 和 PORT，可从控制台获取到，例如10.13.20.15:9200。

请求路径和方法

请求路径： `_bulk`
方法：PUT

> 说明：
> doc 关键字为写入数据的 _type，为了便于以后系统做解析和升级，请务必加上 doc 关键字。

请求参数

可通过设置 filter_path 参数来过滤并简化返回结果，具体请参考 批量查询数据。

请求内容

批量写入 metric 需要一种 NDJSON 格式的结构数据，类似于如下：

```
元数据\n
需要写入的数据\n
....
元数据\n
需要写入的数据\n
```

元数据的格式如下所示：

```
{
"index" :
{
"_index" :"metric_name", #要写入的 metric
"_type" : "doc", #写入的文档类型
"_id" : "1", #文档 ID(选填)
"_routing": "sh" #路由值(选填)
}
}
```

写入数据的格式类似于：

```
{
"field1" : "value1",
"field2" : "value2"
}
```

> 说明：
> 写入数据时可通过设置 _routing 参数来路由数据写入的分片，该参数选填，可指定任意值。指定相同的 _routing 值可将不同数据路由至同一分片上。另外，查询时指定已设置的 _routing 参数值，系统会根据路由到指定分片查询数据，可以极大优化查询速度。注意，请求体中最后一行需要加换行符。

返回内容

需要通过 error 字段判断请求是否成功，若返回内容有 error 字段则请求失败，具体错误内容在 error 字段内。注意：若请求成功，但是 errors（注意不是 error）字段非等于 false，则该 errors 字段具体指出写入失败的具体数据。

CURL 示例说明

请求：

```
curl -u root:le201909 -X PUT 172.16.345.14:9201/_bulk -d'
{"index":{"_index" : "ctsdb_test", "_type" : "doc", "_routing": "sh" }}
{"region":"sh","cpuUsage":2.5,"timestamp":1505294654}
{"index":{"_index" : "ctsdb_test2", "_type" : "doc", "_routing": "sh" }}
{"region":"sh","cpuUsage":2.0,"timestamp":1505294654}
'
```

返回：


```
{
"took": 134,
"errors": false,
"items":
[
{
"index":
{
"_index": "ctsdb_test@1505232000000_1",
"_type": "doc",
"_id": "AV_8eeo_UAkC9PF9L-2q",
"_version": 1,
"result": "created",
"_shards":
{
"total": 2,
"successful": 2,
"failed": 0
},
"created": true,
"status": 201
}
},
{
"index":
{
"_index": "ctsdb_test2@1505232000000_1",
"_type": "doc",
"_id": "AV_8eeo_UAkC9PF9L-2r",
"_version": 1,
"result": "created",
"_shards":
{
"total": 2,
"successful": 2,
"failed": 0
},
"created": true,
"status": 201
}
}
]
}
```


说明：
上面返回中的 errors 为 false，代表所有数据写入成功。items 数组标识每一条记录写入结果，与 bulk 请求中的每一条写入顺序对应。items 的单条记录中，status 为 2XX 代表此条记录写入成功，_index 标识了写入的 metric 子表，_shards 记录副本写入情况，上例中 total 表示两副本，successful 代表两副本均写入成功。

删除数据

最近更新时间：2019-10-18 16:33:38

请求地址

地址为实例的 IP 和 PORT，可从控制台获取到，例如10.13.20.15:9200。

请求路径和方法

请求路径： `/${metric_name}/_delete_by_query` ， `${metric_name}` 为 metric 的名称。
方法：POST

请求参数

无

请求内容

删除 metric 时的查询条件，具体请参考示例。

返回内容

需要通过 error 字段判断请求是否成功，若返回内容有 error 字段则请求失败，具体错误内容在 error 字段内。注意：若请求成功，但是 errors（注意不是 error）字段非等于 false，则该 errors 字段具体指出写入失败的具体数据。

CURL 示例说明

请求：

```
curl -u root:le201909 -X POST 172.16.345.14:9201/ctsdb_test/_delete_by_query -d'
{
"query": {
"bool": {
"filter": {
"match_all": {}
}
}
}
}'
```

返回：

```
{
"took": 43,
"timed_out": false,
"total": 1,
"deleted": 1,
"batches": 1,
"version_conflicts": 0,
"noops": 0,
"retries": {
"bulk": 0,
"search": 0
}
}
```

修改数据

最近更新时间：2019-07-22 15:34:54

CTSDB 修改数据可使用先删除数据再写入数据的方式。数据的删除和写入，请参见 删除数据 和 批量写入数据。

Rollup 相关操作

最近更新时间：2019-10-18 16:33:56

建立 Rollup 任务

在海量数据场景下，业务系统每天甚至每小时会产生 PB 级别数据。而时序数据的最主要的特点就是海量性、时效性和趋势性，因此通常情况下，使用数据的系统（例如监控系统或数据分析系统）通常只需要最近时间段内的高精度数据，而历史数据只需降精度（Downsampling）保存即可。用户可通过配置 Rollup 任务定时聚合历史数据保存至新的数据表。Rollup 任务不仅能降精度保存历史数据，也能提高查询性能，降低存储成本。需要注意的是，Rollup 任务会自动根据 base_metric 建立子表，继承父表的所有配置，如果指定 options，会覆盖父表配置。

1. 请求地址

地址为实例的 IP 和 PORT，可从控制台获取到，例如：10.13.20.15:9200。

2. 请求路径和方法

路径：`/_rollup/${rollup_task_name}`，`${rollup_task_name}` 为 Rollup 任务的名称。
方法：PUT

3. 请求参数

无

4. 请求内容

参数名称	必选	类型	描述
base_metric	是	string	Rollup 依赖的 metric 名称（父表）
rollup_metric	是	string	Rollup 产生的 metric 名称（子表）
base_rollup	否	string	依赖的 Rollup 任务，任务执行前会检查相应时间段的依赖任务是否完成执行
query	否	string	过滤数据的查询条件，由很多个元素和操作对组成，例如：`name:host AND type:max OR region:gz`
group_by	是	Array	进行聚合的维度列，可以包含多列
function	是	Map	指定聚合的名称、方法和字段，其字段只能选自 base_metric 里的 fields 字段，如果 base_metric 的 fields 为空，则无法设置 rollup，function 有 sum、avg、min、max、set、any、first、last、percentiles 等。例如：`{"cost_total":{"sum": {"field":"cost"}},"cpu_usage_avg":{ "avg": { "field":"cpu_usage"}}}`
interval	是	string	聚合粒度（rollup 产生数据的时间精度），例如1s、5m（5分钟）、1h、1d等
frequency	否	string	调度频率，例如5m、1h、1d等，默认等于 interval
delay	否	string	延迟执行时间，写入数据通常有一定的延时，避免丢失数据，例如5m、1h等
start_time	否	string	开始时间，从该时间开始周期性执行 Rollup，默认为当前时间
end_time	否	string	结束时间，到达该时间后不再调度 ，默认为时间戳最大值
options	否	map	rollup_metric 选项，跟新建 metric 选项一致

5. 返回内容

需要通过 error 字段判断请求是否成功，若返回内容有 error 字段则请求失败，具体错误详情请参照 error 字段描述。

6. CURL 示例说明

请求：

```
curl -u root:le201909 -X POST 172.16.345.14:9201/_rollup/ctsdb_rollup_task_test -d'
{
"base_metric": %{base_metric_name},
"rollup_metric": %{rollup_metric_name},
"base_rollup": %{base_rollup_name},
"query" : "name:host AND type:max",
```

```
"group_by": ["host"],
"function": {
"cost_total": {
"sum": {
"field": "cost"
}
},
"cpu_usage_avg": {
"avg": {
"field": "cpu_usage"
}
},
"value": {
"percentiles": {
"field": "value",
"percents": [
95
]
}
},
"metricName": {
"set": {
"value": "cpu_usage"
}
},
"appid": {
"any": {
"field": "appid"
}
},
"first_value": {
"first": {
"field": "value"
}
},
"last_value": {
"last": {
"field": "value"
}
}
},
"interval": "1m",
"frequency": "5m",
"delay": "1m",
"start_time": "1502892000",
"end_time": "2147483647",
"options": {
"expire_day": 365
}
}
'
```

返回：

```
{
"acknowledged": true,
"message": "create rollup success"
}
```

获取所有 Rollup 任务的名称

1. 请求地址

地址为实例的 IP 和 PORT，可从控制台获取到，例如：10.13.20.15:9200。

2. 请求路径和方法

路径：`/_rollups`
方法：GET

3. 请求参数

无

4. 请求内容

无

5. 返回内容

需要通过 error 字段判断请求是否成功，若返回内容有 error 字段则请求失败，具体错误详情请参照 error 字段描述。

6. CURL 示例说明

请求：
`curl -u root:le201909 -X GET 172.16.345.14:9201/_rollups`

返回：

```
{
"result":
{
"rollups":
[
"rollup_jgq_6",
"rollup_jgq_60"
]
},
"status": 200
}
```

获取某个 Rollup 任务的详细信息

1. 请求地址

地址为实例的 IP 和 PORT，例如：10.13.20.15:9200。

2. 请求路径和方法

路径：`/_rollup/${rollup_task_name}`，`${rollup_task_name}` 为 Rollup 任务的名称。
方法：GET

3. 请求参数

指定 v 参数可以查看 rollup 的具体进度，返回结构中的 @last_end_time 为 rollup 最新进度。

4. 请求内容

无

5. 返回内容

需要通过 error 字段判断请求是否成功，若返回内容有 error 字段则请求失败，具体错误详情请参照 error 字段描述。

6. CURL 示例说明

请求：
`curl -u root:le201909 -X GET 172.16.345.14:9201/_rollup/rollup_jgq_6?v`

返回：

```
{
"result": {
"rollup_jgq_6": {
"base_metric": "cvm_device-300",
```

```
"rollup_metric": "cvm_device-86400",
"query": "metricName:cpu_usage AND statType:max",
"group_by": [
"vm_uuid"
],
"function": {
"value": {
"percentiles": {
"field": "value",
"percents": [
95
]
}
},
"metricName": {
"set": {
"value": "cpu_usage"
}
},
"appid": {
"any": {
"field": "appid"
}
}
},
"interval": "1d",
"delay": "5m",
"options": {
"expire_day": 186
},
"frequency": "1d",
"start_time": 1534003200,
"end_time": 2147483647,
"@state": "running", // 运行状态
"@timestamp": 1550766085000, // rollup 任务信息更新的时间点
"@last_end_time": 1550764800 // rollup 任务正确执行到的时间点

}
},
"status": 200
}
```

删除 Rollup 任务

1. 请求地址

地址为实例的 IP 和 PORT，例如：10.13.20.15:9200。

2. 请求路径和方法

路径： `/_rollup/${rollup_task_name}` ， `${rollup_task_name}` ，为 Rollup 任务的名称。
方法：DELETE

3. 请求参数

无

4. 请求内容

无

5. 返回内容

需要通过 error 字段判断请求是否成功，若返回内容有 error 字段则请求失败，具体错误详情请参照 error 字段描述。

6. CURL 示例说明

请求：

```
curl -u root:le201909 -X DELETE 172.16.345.14:9201/_rollup/ctsdb_rollup_task_test
```

返回：

```
{
"acknowledged": true,
"message": "delete rollup success"
}
```

更新 Rollup 任务

1. 请求地址

地址为实例的 IP 和 PORT，例如：10.13.20.15:9200。

2. 请求路径和方法

路径：`/_rollup/${rollup_task_name}/update` ，`${rollup_task_name}` 为 Rollup 任务的名称。
方法：POST

3. 请求参数

无

4. 请求内容

参数名称	必选	类型	描述
state	是	string	running/pause
start_time	否	string	开始时间，从该时间开始周期性执行 Rollup，默认为当前时间
end_time	否	string	结束时间，到达改时间后不再调度，默认为时间戳最大值
options	否	map	聚合选项，跟新建 metric 选项一致

5. 返回内容

需要通过 error 字段判断请求是否成功，若返回内容有 error 字段则请求失败，具体错误详情请参照 error 字段描述。

6. CURL 示例说明

请求：

```
curl -u root:le201909 -X POST 172.16.345.14:9201/_rollup/ctsdb_rollup_task_test/update -d'
{
"state":"running",
"start_time": "1511918989",
"end_time": "1512019765",
"options":
{
"expire_day": 365
}
}'
```

返回：

```
{
"acknowledged": true,
"message": "update rollup success"
}
```