时序数据库 CTSDB

实践教程

腾讯云

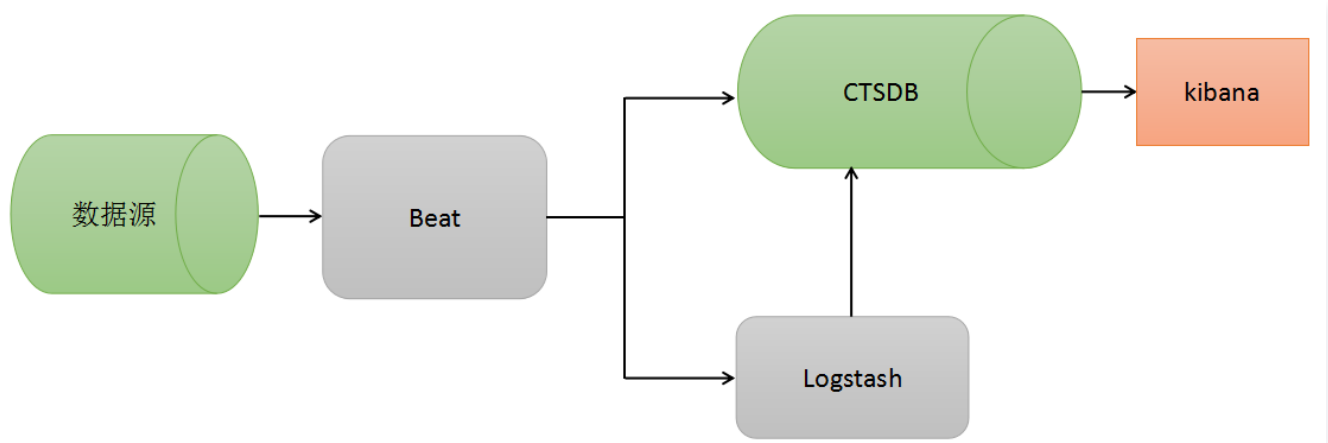
文档目录

实践教程
CTSDB 对接 ELK 组件及 Grafana

最近更新时间：2024-05-15 11:23:16

概述

云数据库 CTSDB 是一款分布式、可扩展、支持近实时数据搜索与分析的时序数据库，且兼容 ELK 生态组件，您可以非常方便的使用 ELK 组件与 CTSDB 对接。
ELK 组件提供了丰富的数据处理功能，包括数据采集、数据清洗、可视化图形展示等。常用的 ELK 生态组件包括 Filebeat、Logstash、Kibana。同时，CTSDB 也支持 Grafana 作为可视化平台。常见架构图如下：



组件的使用

Filebeat

Filebeat 是一个轻量级开源日志文件数据搜集器，作为 agent 安装到服务器上，Filebeat 读取文件内容，发送到 Logstash 进行解析后进入 CTSDB，或直接发送到 CTSDB 进行集中式存储和分析。

Filebeat 的使用流程

1. **安装**
   Filebeat 安装介绍请参见 该地址 。
2. **配置**
   Filebeat 的配置采用 YAML 格式文件，主要配置为全局配置、输入配置、输出配置，下节会给出使用样例。
3. **启动**
   Filebeat 启动时可以指定配置文件路径，若不指定则默认使用 filebeat.yml。

Filebeat 使用示例

1. 首先，将 Filebeat 的安装包解压缩到某一目录，如下所示：

```
[user_02@TENCENT64 ~/filebeat]$ ls
NOTICE  README.md  data  filebeat  filebeat.full.yml  filebeat.template-es2x.json  filebeat.template-es6x.json  filebeat.template.json  filebeat.yml  logs  module  scripts
```

2. 然后，配置 filebeat.yml，配置参考如下：

```
filebeat.shutdown_timeout: 5 # How long filebeat waits on shutdown for the publisher to finish.
max_procs: 4 # 可同时执行的最大cpu数，默认为操作系统可用的逻辑cpu数
filebeat.spool_size: 102400
filebeat.idle_timeout: 2s
processors:
- drop_fields: # 需要drop掉的字段
  fields: ["beat","input_type","source","offset"]
filebeat.prospectors:
- paths: ["/data/log/filebeat-tutorial.log"]    # 样例数据所在的路径
```

```
  fields:
    metricname: metric1
  harvester_buffer_size: 1638400
  close_timeout: 0.5h
  scan_frequency: 2s
- paths: ["/mylog/*.log","/mylog1/*.log"]
  fields:
    metricname: table2
  harvester_buffer_size: 1638401
  close_timeout: 0.5h
  scan_frequency: 2s
output.elasticsearch:
  hosts: ["127.0.0.1:9200"]
  index: "%{[fields.indexname]}"  # 通配，可以达到不同类别的数据写入不同index的目的
  username: "root" # 对于有权限的CTSDB这里需要填用户名和密码
  password: "changeme"
  worker: 2 # 工作线程数
  loadbalance: true # 是否开启负载均衡
  bulk_max_size: 512 # 一次bulk的最大文档数
  flush_interval: 2s
  template:
    enabled: false  # 注意：Filebeat启动后会put一个默认的template，对接CTSDB时，需要禁用Filebeat的template
```

部分样例数据如下：

```
83.149.9.216 - - [04/Jan/2015:05:13:42 +0000] "GET /presentations/logstash-monitorama-
2013/images/kibana-search.png HTTP/1.1" 200 203023 "http://semicomplete.com/presentations/logstash-
monitorama-2013/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_1) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/32.0.1700.77 Safari/537.36"
83.149.9.216 - - [04/Jan/2015:05:13:42 +0000] "GET /presentations/logstash-monitorama-
2013/images/kibana-dashboard3.png HTTP/1.1" 200 171717 "http://semicomplete.com/presentations/logstash-
monitorama-2013/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_1) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/32.0.1700.77 Safari/537.36"
83.149.9.216 - - [04/Jan/2015:05:13:44 +0000] "GET /presentations/logstash-monitorama-
2013/plugin/highlight/highlight.js HTTP/1.1" 200 26185 "http://semicomplete.com/presentations/logstash-
monitorama-2013/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_1) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/32.0.1700.77 Safari/537.36"
```

3. **启动 Filebeat，并观察 CTSDB 中对应表的数据：**

```
nohup ./filebeat &
    less logs/filebeat # 查看部分日志，通过日志libbeat.es.published_and_acked_events=100可以看出我们的100条日志都成功写入到es中
    2018-05-25T14:32:24+08:00 INFO Non-zero metrics in the last 30s: filebeat.harvester.open_files=1
filebeat.harvester.running=1 filebeat.harvester.started=1 libbeat.es.call_count.PublishEvents=1
libbeat.es.publish.read_bytes=1535 libbeat.es.publish.write_bytes=40172
libbeat.es.published_and_acked_events=100 libbeat.publisher.published_events=100 publish.events=101
registrar.states.current=1 registrar.states.update=101 registrar.writes=2

    # 通过kibana或curl查看es中是否有数据写入到metric1
    # 命令：
    GET metric1/_search
    {
      "sort": [
    {
      "@timestamp": {
    "order": "desc"
      }
    }
```

```
        ],
        "docvalue_fields": ["@timestamp", "message"]
      }
      # 结果:
      {
        "took": 2,
        "timed_out": false,
        "_shards": {
      "total": 3,
      "successful": 3,
      "skipped": 0,
      "failed": 0
        },
        "hits": {
      "total": 100,
      "max_score": null,
      "hits": [
        {
      "_index": "metric1@1525536000000_30",
      "_type": "doc",
      "_id": "AWOV_o1wBzkw2jsSfrLN",
      "_score": null,
      "fields": {
        "@timestamp": [
      1527229914629
        ],
        "message": [
      "218.30.103.62 - - [04/Jan/2015:05:27:57 +0000] \"GET /blog/geekery/c-vs-python-bdb.html
HTTP/1.1\" 200 11388 \"-\" \"Sogou web spider/4.0(+http://www.sogou.com/docs/help/webmasters.htm#07)\""
        ]
      },
      "sort": [
        1527229914629
      ]
        },
      # 内容太多，这里省略，通过hits.total可以看出，查询命中了100条文档，证明100条log都成功写入CTSDB`
```

上述示例是直接通过 Filebeat 将原始日志数据写入到 CTSDB 中，并没有做字段的解析，下节将会介绍通过 Logstash 解析数据，然后写入 CTSDB。

Logstash

Logstash 是一款具有实时数据解析功能的开源数据收集引擎。Logstash 能够搜集各种数据源，并对数据进行过滤、分析、格式化等操作，然后存储到 CTSDB。

Logstash 使用流程

1. **安装**
   Logstash 安装请参见 该地址 。
2. **配置**
   Logstash 的主要配置包含三个模块，分别为数据源输入，数据解析规则，数据输出。下节会给出使用样例。
3. **启动**
   Logstash 启动时，可以指定配置文件，否则，默认使用 logstash.yml 作为配置，解析规则默认使用 pipelines.yml 中的配置。

Logstash使用示例

1. 首先，将 Logstash 的安装包解压缩到某一目录，如下所示：

```
[user_00@TENCENT64 ~/luckie/logstash-6.2.4]$ ls
bin  config  CONTRIBUTORS  data  first-pipeline.conf  Gemfile  Gemfile.lock  lib  LICENSE  logs  logstash-core  logstash-core-plugin-api  modules  nohup.out  NOTICE.TXT  tools  vendor
```

2. 然后，创建一个配置文件，当然也可以在 logstash.yml 和 pipelines.yml 中进行配置。这里创建一个配置文件名为 first-pipeline.conf，配置如下：

```
# 输入源
input {
 beats {
     port => "5044"
 }
}
# 解析过滤
filter {
 grok {
     match => {
         "message" => "%{COMBINEDAPACHELOG}"
     }
 }
}
# 输出
output {
 elasticsearch {
     action => "index"
     hosts => ["localhost:9200"]
     index => "logstash_metric" # CTSDB中创建的metric名
     document_type => "doc"
     user => "root" # 对于有权限的CTSDB需要指定用户名和密码
     password => "changeme"
 }
}
```

grok filter 插件在 Logstash 默认可用的，其能够将非结构化的数据解析为结构化的数据，具体使用参考 文档 。

3. 启动 Logstash、Filebeat，并观察 CTSDB 中对应表的数据：

```
  # 这里需要注意的是，Filebeat的输出是Logstash，因此Filebeat的输出项配置改为：
 output.logstash:
   hosts: ["localhost:5044"]
 # 清空Filebeat的data目录，启动Filebeat
 rm data/registry
 nohup ./filebeat &
 # 启动Logstash
 nohup bin/logstash -f first-pipeline.conf --config.reload.automatic &
 # 通过kibana或curl查看CTSDB中是否有数据写入到metric1
 # 命令：
 GET logstash_metric/_search
 {
   "sort": [
     {
       "@timestamp": {
         "order": "desc"
       }
     }
   ],
   "docvalue_fields": ["@timestamp","request", "response", "type", "bytes",  "verb", "agent",
"clientip"]
 }
# 结果：
 {
   "took": 0,
   "timed_out": false,
   "_shards": {
     "total": 0,
     "successful": 0,
     "skipped": 0,
```

```
      "failed": 0
    },
    "hits": {
      "total": 0,
      "max_score": 0,
      "hits": []
    }
  }
  # 这里发现结果是空的，说明没有数据没有写入到CTSDB，查看Logstash日志：
  [2018-05-25T21:00:07,081][ERROR][logstash.outputs.elasticsearch] Encountered a retryable error. Will
Retry with exponential backoff  {:code=>403, :url=>"http://127.0.0.1:9200/_bulk"}
  [2018-05-25T21:00:07,081][ERROR][logstash.outputs.elasticsearch] Encountered a retryable error. Will
Retry with exponential backoff  {:code=>403, :url=>"http://127.0.0.1:9200/_bulk"}
  # 发现bulk出错，并返回403权限错误，仔细验证 用户名和密码，发现并无问题，继续查看es日志：
  [2018-05-25T20:59:27,545][WARN ][o.e.p.o.OPackActionFilter] [1505480279000001609] process index
failed: Invalid format: "2018-05-25T12:51:18.905Z"
  [2018-05-25T20:59:27,547][WARN ][o.e.p.o.OPackActionFilter] [1505480279000001609] process index
failed: Invalid format: "2018-05-25T12:51:18.905Z"
  # 从es的日志可以看出 ，时间格式解析出错。出错的原因是，笔者建metric时没有指定时间字段的格式，那么CTSDB默认为
epoch_millis，因此需要修改下时间格式：
  POST /_metric/logstash_metric/update?pretty
  {
    "time": {
           "name": "@timestamp",
           "format": "strict_date_optional_time"
      }
  }
  # 重启Logstash、Filebeat重新写入数据，再查看CTSDB：
  # 结果
  {
    "took": 2,
    "timed_out": false,
    "_shards": {
      "total": 3,
      "successful": 3,
      "skipped": 0,
      "failed": 0
    },
    "hits": {
      "total": 100,
      "max_score": null,
      "hits": [
        {
          "_index" : "logstash_metric@1527004800000_30",
          "_type" : "doc",
          "_id" : "AWOvG0YgQoCkIV2BLcov",
          "_score" : null,
          "fields" : {
            "request" : [
              "/blog/tags/puppet?flav=rss20"
            ],
            "agent" : [
              "\"UniversalFeedParser/4.2-pre-314-svn +http://feedparser.org/\""
            ],
            "@timestamp" : [
              1527651156725
            ],
            "response" : [
              "200"
            ],
            "bytes" : [
```

```
            14872
          ],
          "clientip" : [
            "46.105.14.53"
          ],
          "verb" : [
            "GET"
          ],
          "type" : [
            "log"
          ]
        },
        "sort" : [
          1527651156725
        ]
      },
    ... ...
    # 内容太多，这里省略，通过hits.total可以看出，查询命中了100条文档，证明100条log都成功写入CTSDB
```

从上述示例，我们可以看出，通过 Filebeat 采集数据到 Logstash，然后利用 Logstash 的数据解析功能，将日志解析为多个字段，然后写入 CTSDB。

Kibana

Kibana 是一个旨在为 Elasticsearch 设计的开源的分析和可视化平台。可以使用 Kibana 来搜索，查看存储在 CTSDB metric 中的数据并与其进行交互。可以利用 Kibana 中丰富的图表、表格、曲线等功能来可视化数据并进行数据分析。

Kibana 使用流程

1. **安装**
   下载与 Elasticseach 对应的 Kibana 版本，并解压到某一目录，兼容的具体版本可参考 查看实例详情 在控制台查看。
2. **配置**
   Kibana 的配置很简单，下节会给出样例。具体配置项含义参考 该地址 。
3. **运行**
   Kibana 运行时，默认使用 config/kibana.yml 作为配置。

Kibana使用示例

1. 首先，将 Kibana 的安装包解压到某一目录，如下所示

```
[user_00@TENCENT64 ~/repository/kibana]$ ls
bin  config  data  kibana.log  kibana.nohup  LICENSE.txt  logs  node  node_modules  nohup.out  optimize  package.json  plugins  plugins.bk  README.txt  src  webpackShims
```

2. 然后，修改 config 下的配置文件。主要配置如下：

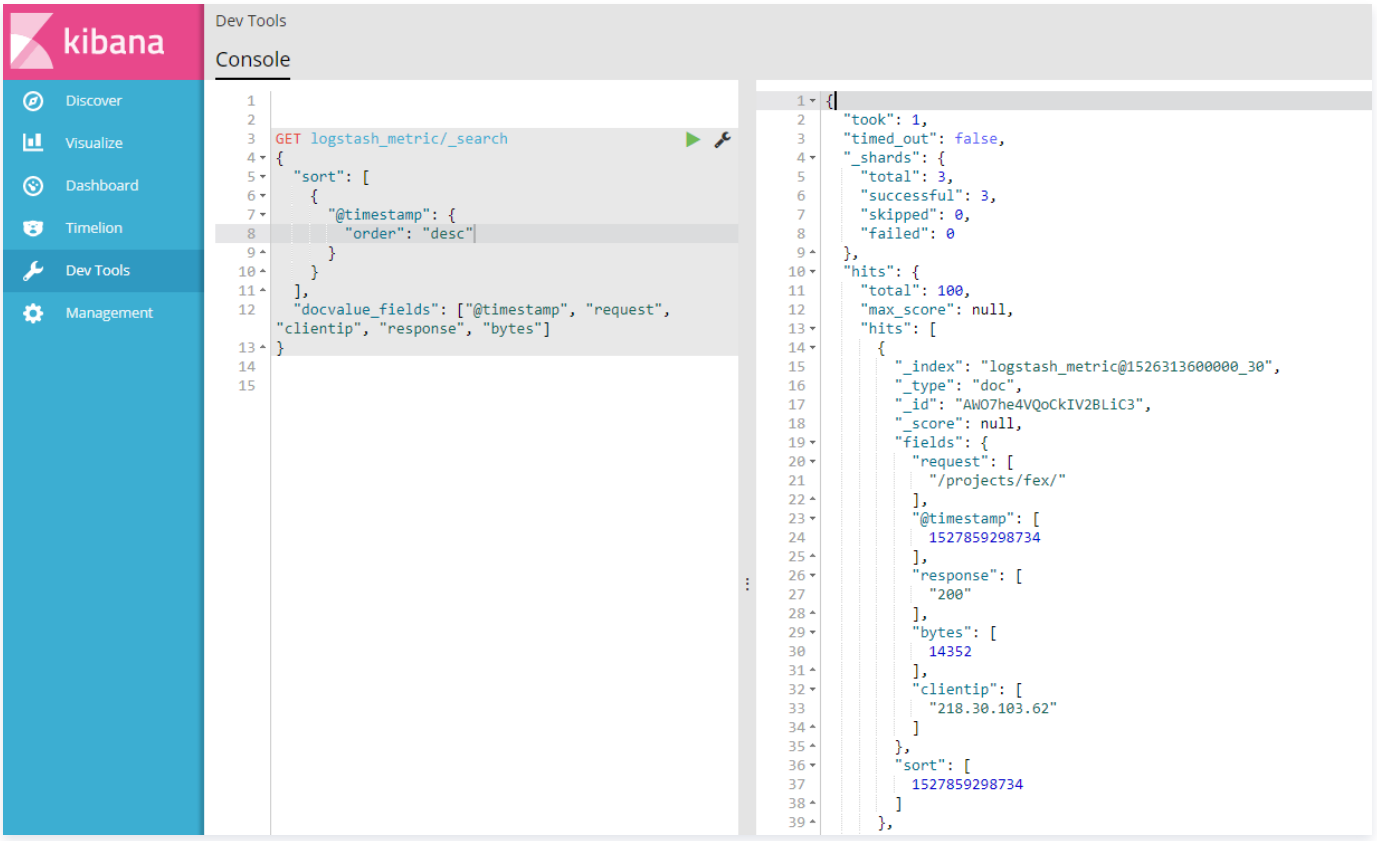
```
# config/kibana.yml
# Kibana server监听的端口
server.port: 5601
# Kibana server所绑定的服务器ip
server.host: 127.0.0.1
# 所要连接的CTSDB的url
elasticsearch.url: "http://127.0.0.1:9200"
# 若使用带权限的CTSDB，则需要指定用户名和密码
elasticsearch.username: "root"
elasticsearch.password: "changeme"
```

- 启动，并通过浏览器访问 kibana

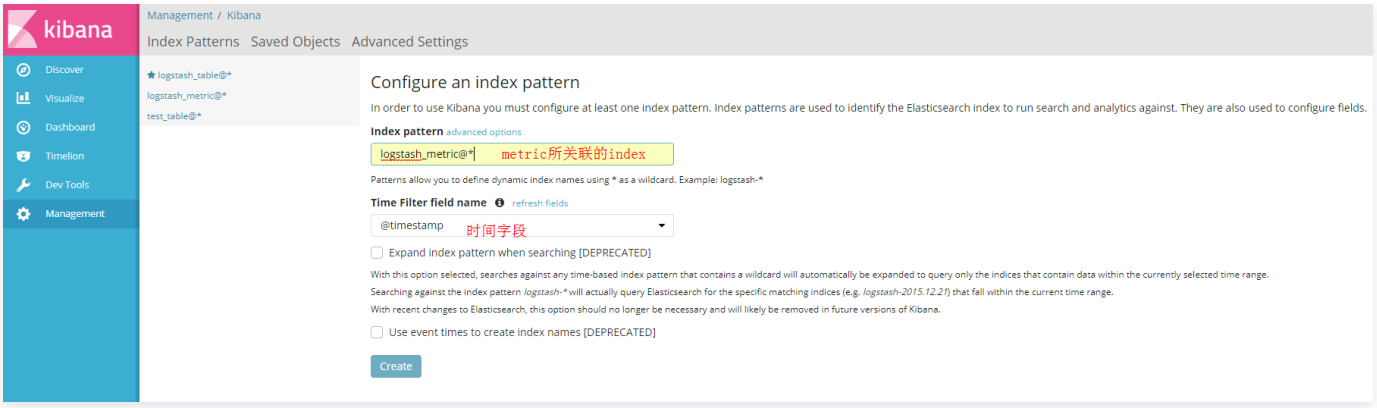
```
nohup bin/kibana &
```

利用 ip:port 或者域名访问 kibana server，如下：

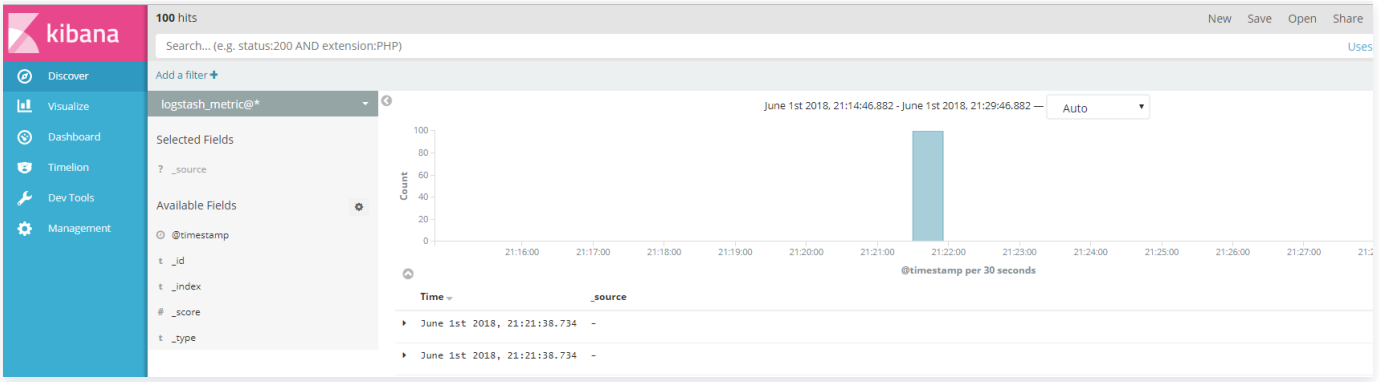
利用开发工具，我们可以很方便的访问 CTSDB，如下图所示：



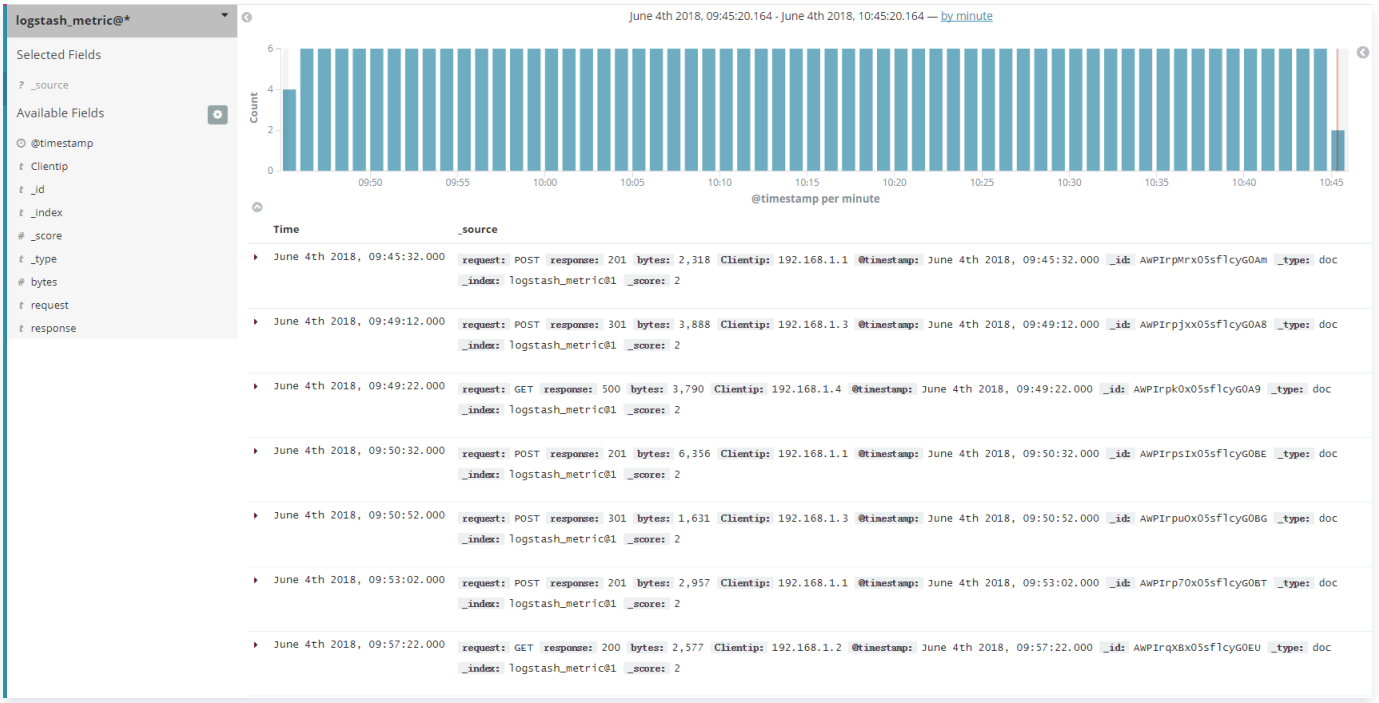
在管理页面创建需要访问的索引，如下所示：



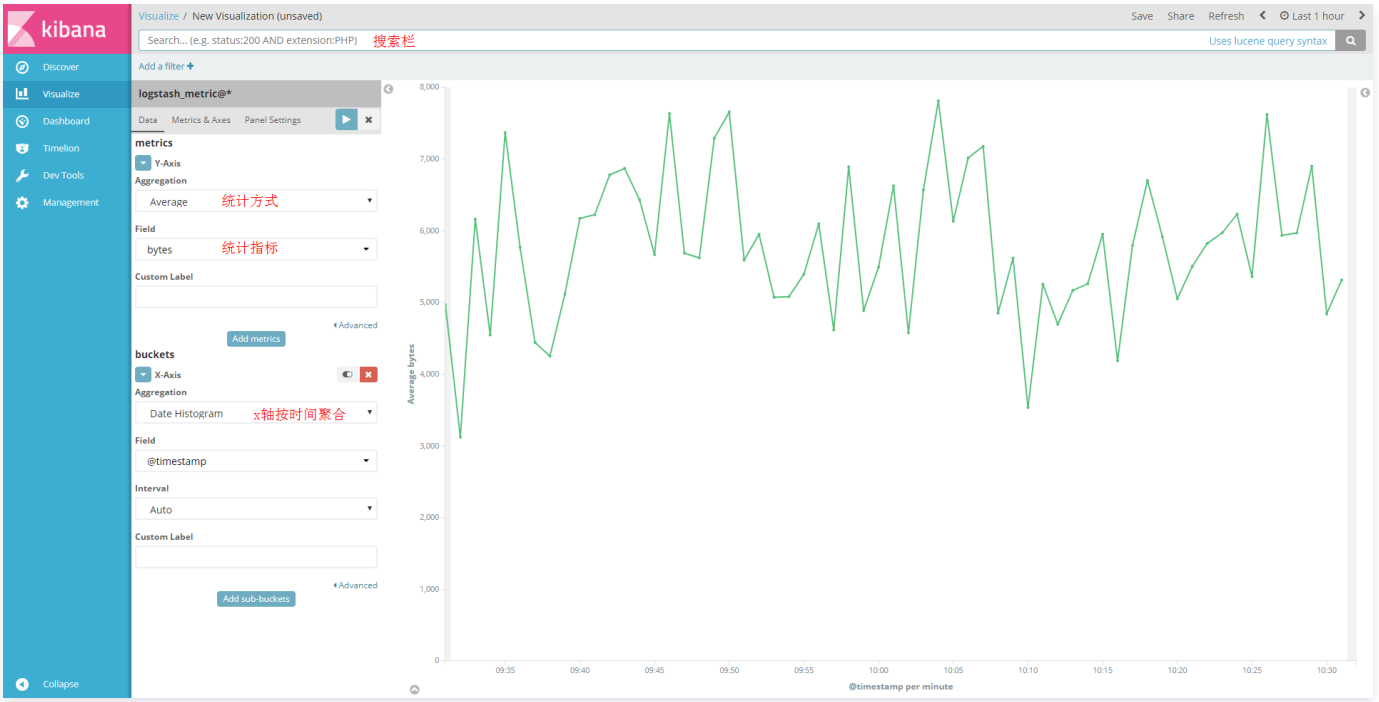
如果您有日志搜索的需求，可能会使用到搜索功能，如下图所示：



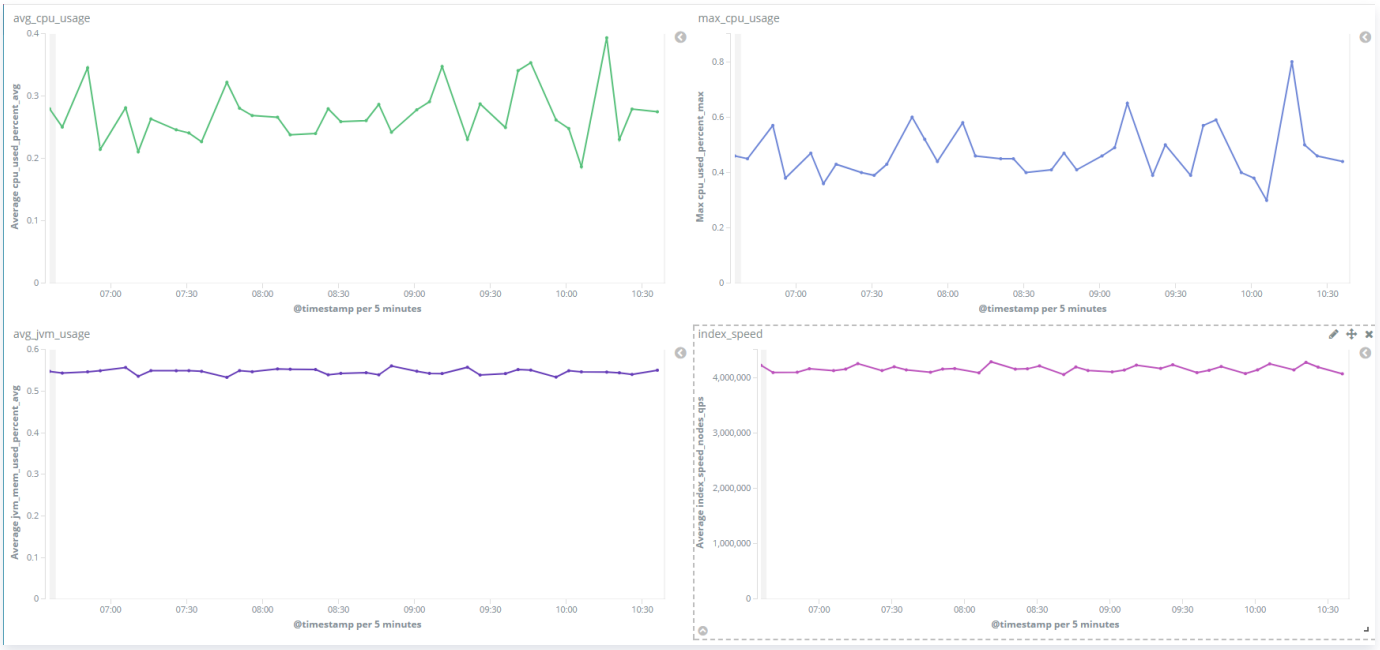
从上图，我们发现，搜索的结果出了时间以外，没有其他，原因是 CTSDB 为了节省存储空间默认没有开启 source 功能，如果您有日志搜索的需求，请联系我们开启 source。开启 source 后的效果如下图所示：



利用可视化，我们可以构建各种图表，这里以创建 Line Chart 为例，展示 Kibana 的图表效果，如下所示：



Kibana 不仅提供了丰富的可视化图表功能，而且还可以利用仪表盘将我们保存的可视化图表统一展示都一个页面上，非常方便地查看多个指标的变化状态。这里为了展示效果，贴一张我们内部真实的监控数据的仪表盘视图，如下所示：



Grafana

Grafana 是一款开源的仪表盘工具，它提供了丰富的图表功能，类似 Kibana，利用 Grafana 精细的展示效果，可以帮助您有效地进行数据分析。和 Kibana 不同的是，Grafana 支持的数据源种类更多，包含 influxdb、opentsdb、Elasticsearch，下面演示利用 Grafana 来可视化的分析 CTSDB 中的数据。

Grafana

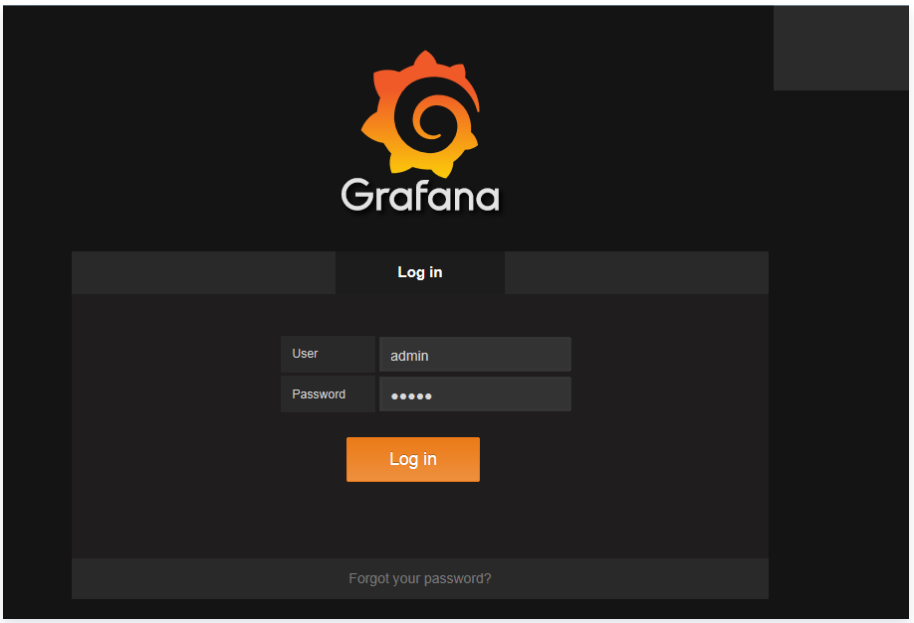
1. 安装
   Grafana 的安装请参见 官方文档 。
2. 配置
   Grafana 的配置项较多，可以使用默认的配置，具体配置说明参考 该地址 。
3. 运行
   Grafana 运行时，默认使用 /etc/grafana/grafana.ini 作为配置。

Grafana 使用示例

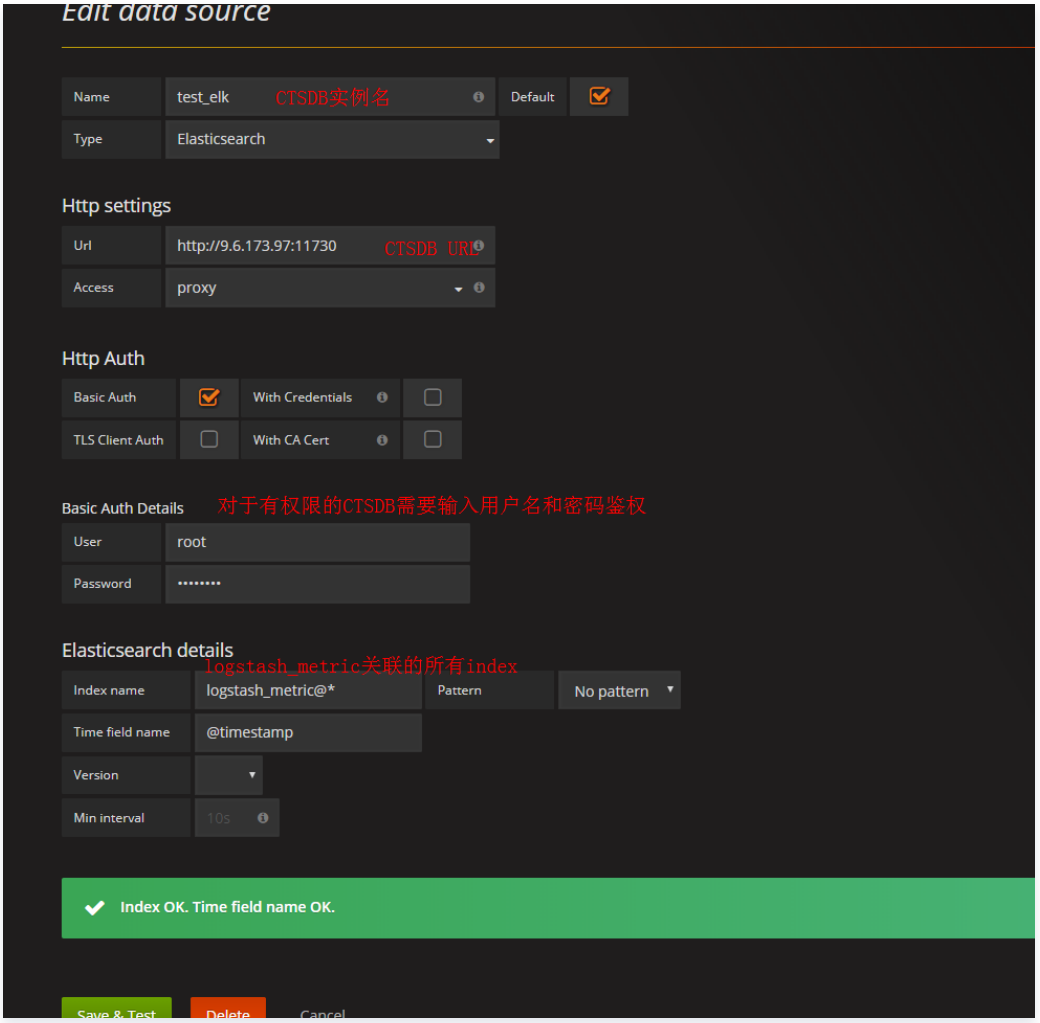
1. 首先，启动 grafana 服务。

```
sudo service grafana-server start
```

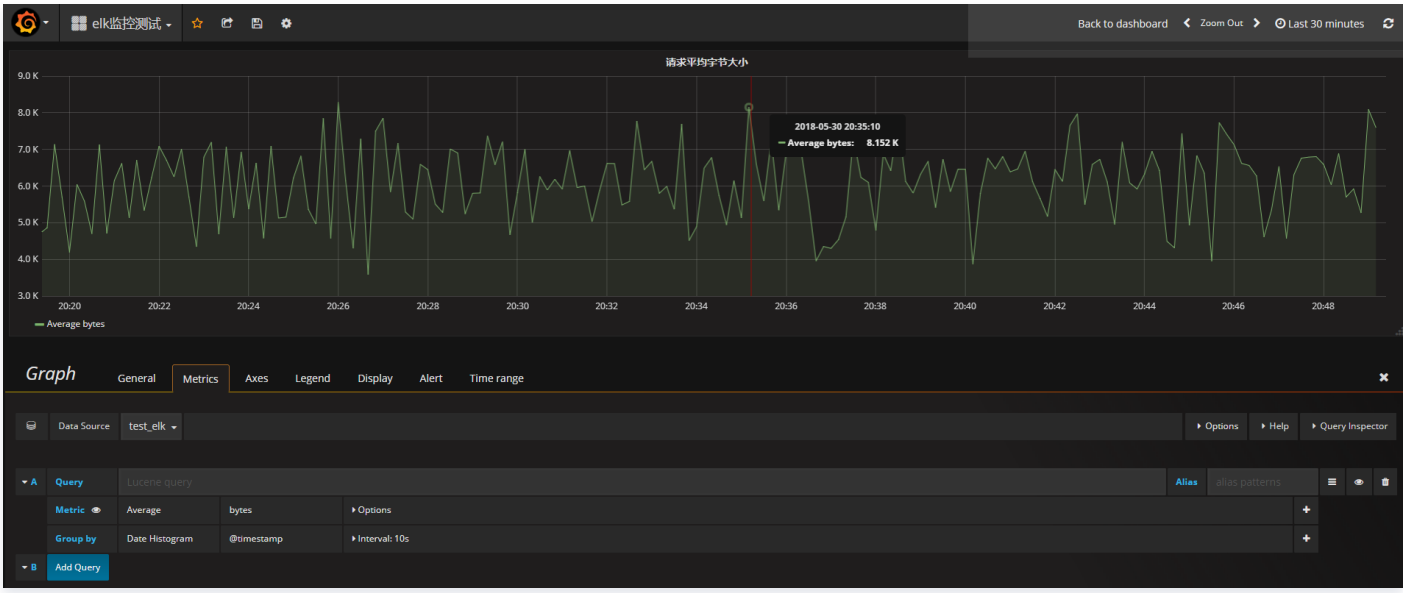
2. 然后，通过浏览器访问 Grafana 服务：



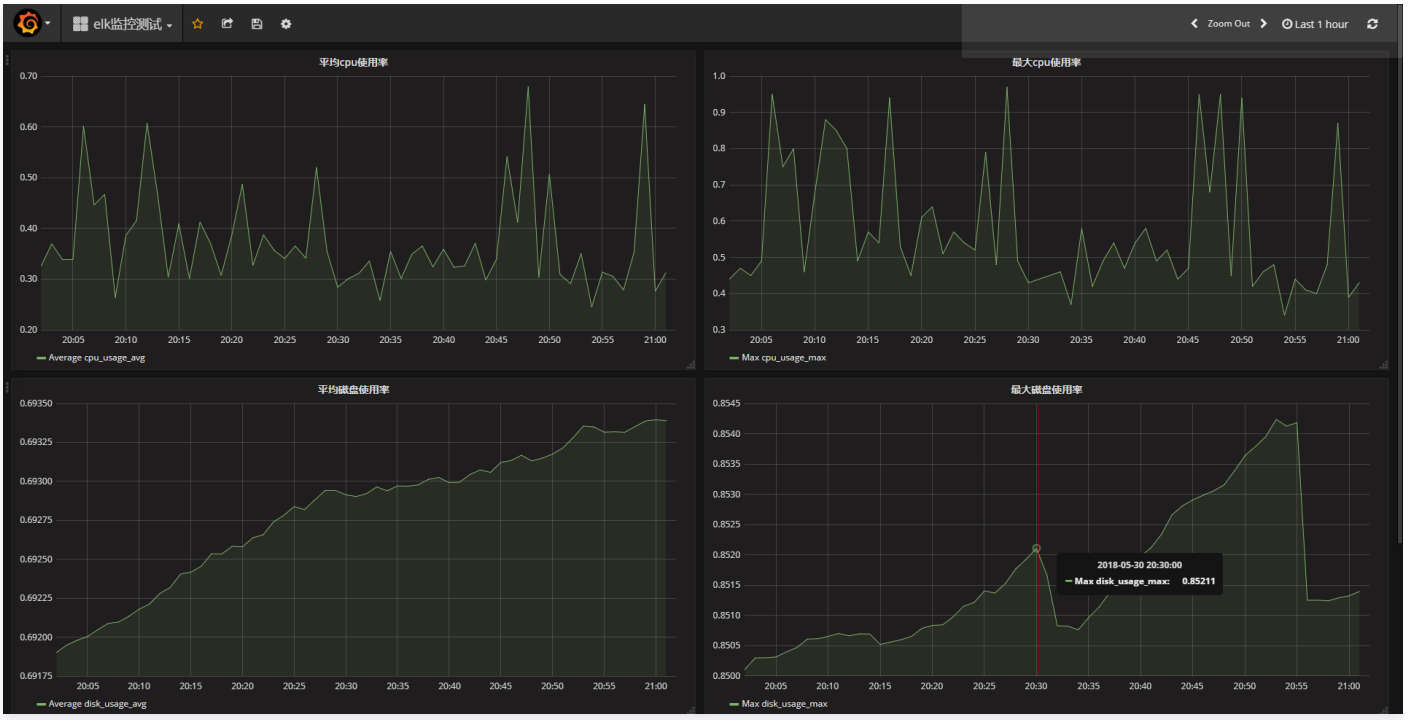
3. 创建数据源，建立 dashboard，如下所示：



4. 利用 dashboard 创建可视化图表，如下图所示：



5. 从上图可以看出，Grafana 的图表展示效果和 Kibana 略有区别，但是功能本质上是一样的，这个看您的个人使用习惯和爱好。同样，Grafana 的 dashboard 也能同时展示多个可视化图表，如下图所示：



小结

以上为 ELK 生态组件及 Grafana 对接 CTSDB 的详细使用过程，如在使用过程中遇到问题，可通过 在线支持 解决。

MySQL 数据导入 CTSDB

最近更新时间：2022-08-22 15:46:32

前言

CTSDB 是一款分布式、可扩展、支持近实时数据搜索与分析的时序数据库，且兼容 Elasticsearch 常用的 API 接口。对于很多用户，想要将 MySQL 中的数据导入到 CTSDB 中，而又找不到一种较好的方法，这里给出一种简单快捷的方式，轻松将 MySQL 中的数据同步到 CTSDB。

工具介绍

go-mysql-elasticsearch 是一款开源的高性能的 MySQL 数据同步 Elasticsearch 的工具，其由 go 语言开发，编译及使用都非常简单。go-mysql-elasticsearch 的原理也很简单，首先使用 mysqldump 获取当前 MySQL 的数据，然后在通过此时 binlog 的 name 和 position 获取增量数据，再根据 binlog 构建 restful api 写入数据到 Elasticsearch 中。由于 CTSDB 基于 Elasticsearch 开发，因此，可以完美对接 go-mysql-elasticsearch，导入 MySQL 数据。

MySQL 数据同步 CTSDB 步骤

MySQL 样例数据构建

既然读者有 MySQL 导入 CTSDB 的需求，那 MySQL 的安装就不用多说了。这里为了整个流程的完整性，就从样例数据的灌入开始，用 go 写了一个小工具，生成一些样例数据并灌入到 MySQL 中，表结构如下：

```
mysql> desc test_table;
+-----------+-------------+------+-----+---------+----------------+
| Field     | Type        | Null | Key | Default | Extra          |
+-----------+-------------+------+-----+---------+----------------+
| id        | int(11)     | NO   | PRI | NULL    | auto_increment |
| timestamp | bigint(20)  | YES  |     | NULL    |                |
| cpu_usage | float       | YES  |     | NULL    |                |
| host_ip   | varchar(20) | YES  |     | NULL    |                |
| region    | varchar(20) | YES  |     | NULL    |                |
+-----------+-------------+------+-----+---------+----------------+
```

以上创建了一个名为 test_table 的表，然后向该表灌入2000条样例数据，部分数据如下所示：

```
mysql> select * from test_table;
+------+------------+-----------+-------------+-----------+
| id   | timestamp  | cpu_usage | host_ip     | region|
+------+------------+-----------+-------------+-----------+
|1 | 1527676339 |  0.23 | 192.168.1.1 | beijing   |
|2 | 1527676399 |  0.78 | 192.168.1.2 | shanghai  |
|3 | 1527676459 |   0.2 | 192.168.1.3 | guangzhou |
|4 | 1527676519 |  0.47 | 192.168.1.4 | shanghai  |
|5 | 1527676579 |  0.13 | 192.168.1.5 | beijing   |
|6 | 1527676639 |  0.15 | 192.168.1.1 | beijing   |
|7 | 1527676699 |  0.07 | 192.168.1.2 | shanghai  |
|8 | 1527676759 |  0.17 | 192.168.1.3 | guangzhou |
|9 | 1527676819 |  0.94 | 192.168.1.4 | shanghai  |
|10| 1527676879 |  0.06 | 192.168.1.5 | beijing   |
```

至此，MySQL 端的样例数据准备完毕。

CTSDB metric 创建

现在，我们在 CTSDB 上创建一个和 MySQL 一样的表结构，用于存储对应的数据，创建接口如下所示：

```
POST /_metric/test_metric
{
  "time": {
```

```
        "name": "timestamp",    # 与 MySQL 表中的 timestamp 对应，CTSDB 常用的时间域
        "format": "strict_date_optional_time || epoch_second"
      },
      "tags": {
        "region": "string",
        "host_ip": "string"
      },
      "fields": {
        "cpu_usage": "float"    # fields 域代表指标列，很明显 cpu_usage 代表需要监控 CPU 使用率指标
      }
    }
```

至此，CTSDB 中的表结构也准备好了，下面我们使用 go-mysql-elasticsearch 来同步数据。

go-mysql-elasticsearch 使用

由于 go-mysql-elasticsearch 是用 go 语言开发，因此首先安装 go，官方要求的版本是 1.6 以上，go 的安装非常简单，参考官方文档 下载、安装，然后开始安装 go-mysql-elasticsearch，整个步骤如下：

```
$ go get github.com/siddontang/go-mysql-elasticsearch
$ cd $GOPATH/src/github.com/siddontang/go-mysql-elasticsearch
$ make
```

工具安装好后，需要进行一些合理地配置我们才能愉快地使用，下面将会给出一个配置范例，并给予相应地注释说明：

```
# 注意：go-mysql-elasticsearch 的默认配置文件在 go-mysql-elasticsearch/etc/river.toml
    # MySQL address, user and password
    # user must have replication privilege in MySQL.
    my_addr = "127.0.0.1:3306"
    my_user = "root"
    my_pass = "123456"
    my_charset = "utf8"
    # Set true when elasticsearch use https
    #es_https = false
    # CTSDB 地址
    es_addr = "9.6.174.42:13982"
    # 如果使用的是带权限的 CTSDB，需要设置用户名和密码
    es_user = "root"
    es_pass = "changeme"
    # Path to store data, like master.info, if not set or empty,
    # we must use this to support breakpoint resume syncing.
    # TODO: support other storage, like etcd.
    data_dir = "./var"  # 存储的是 binlog 的名字及位置
    # Inner http status address
    stat_addr = "127.0.0.1:12800"
    # pseudo server id like a slave
    server_id = 1001
    # mysql or mariadb
    flavor = "mysql"
    # mysqldump execution path
    # if not set or empty, ignore mysqldump.
    mysqldump = "mysqldump"
    # minimal items to be inserted in one bulk
    bulk_size = 512
    # force flush the pending requests if we don't have enough items >= bulk_size
    flush_bulk_time = "200ms"
    # Ignore table without primary key
    skip_no_pk_table = true
    # MySQL data source
    [[source]]
```

```
    schema = "mysql_es"
    # Only below tables will be synced into Elasticsearch.
    # "t_[0-9]{4}" is a wildcard table format, you can use it if you have many sub tables, like table_0000
- table_1023
    # I don't think it is necessary to sync all tables in a database.
    tables = ["test_*"]
    [[rule]]
    schema = "mysql_es"    # MySQL 数据库名
    table = "test_table"   # MySQL 表名
    index = "test_metric"  # CTSDB 中 metric 名
    type = "doc"           # 文档类型
```

以上配置，为测试所使用的配置，如果您有更高级的需求可以参考官方文档，合理进行配置。配置 ok 后，我们来运行 go-mysql-elasticsearch，如下所示：

```
    $  ./bin/go-mysql-elasticsearch -config=./etc/river.toml
    2018/05/31 21:43:44 INFO  create BinlogSyncer with config {1001 mysql 127.0.0.1 3306 root   utf8 false
false <nil> false false 0 0s 0s 0}
    2018/05/31 21:43:44 INFO  run status http server 127.0.0.1:12800
    2018/05/31 21:43:44 INFO  skip dump, use last binlog replication pos (mysql-bin.000002, 194296) or
GTID %!s(<nil>)
    2018/05/31 21:43:44 INFO  begin to sync binlog from position (mysql-bin.000002, 194296)
    2018/05/31 21:43:44 INFO  register slave for master server 127.0.0.1:3306
    2018/05/31 21:43:44 INFO  start sync binlog at binlog file (mysql-bin.000002, 194296)
    2018/05/31 21:43:44 INFO  rotate to (mysql-bin.000002, 194296)
    2018/05/31 21:43:44 INFO  rotate binlog to (mysql-bin.000002, 194296)
    2018/05/31 21:43:44 INFO  save position (mysql-bin.000002, 194296)
```

这里需要注意，由于 go-mysql-elasticsearch 需要利用 binlog，而且 binlog 一定要变成 row-based format 格式，因此在 MySQL 必须配置如下参数：

```
# binlog 参数必须要配置如下：
log_bin=mysql-bin
binlog_format = ROW
server-id=1
```

现在，我们来看一下 CTSDB 中是否成功导入了 MySQL 中的数据：

```
    GET test_metric/_search?size=1000
    {
      "sort": [
    {
      "timestamp": {
    "order": "desc"
      }
    }
      ],
      "docvalue_fields": ["timestamp", "host_ip", "region", "cpu_usage"]
    }
    #结果：
    {
      "took": 8,
      "timed_out": false,
      "_shards": {
    "total": 3,
    "successful": 3,
    "skipped": 0,
    "failed": 0
      },
```

```
    "hits": {
  "total": 2000,
  "max_score": null,
  "hits": [
    {
  "_index": "test_metric@1525363200000_30",
  "_type": "doc",
  "_id": "2000",
  "_score": null,
  "fields": {
    "host_ip": [
  "192.168.1.5"
    ],
    "region": [
  "beijing"
    ],
    "cpu_usage": [
  0.05000000074505806
    ],
    "timestamp": [
  1527807286000
    ]
  },
  "sort": [
    1527807286000
  ]
    },
    ......
  }
```

小结

可以看到，使用 go-mysql-elasticsearch，仅需要在配置文件里面写规则，就能非常方便的将数据从 MySQL 同步给 ES。上面仅举了一些简单的例子，如果有更多的需求可以参考 go-mysql-elasticsearch的官方文档。
除了本文所介绍的工具外，这里再推荐两种工具：

- py-mysql-elasticsearch-sync，该工具是使用 Python语言编写，与 go-mysql-elasticsearch 的原理类似，都是使用 binlog 来实现数据的同步，安装及使用见 官方文档 。
- logstash，使用 logstash 同步数据时需要安装 logstash-input-jdbc、logstash-output-elasticsearch 两个插件，具体使用参考 官方文档 、elastic 官方文档 。
  如果您在使用上述工具中遇到问题，可通过 在线支持 解决。

快速选择实例

最近更新时间：2024-01-18 10:32:32

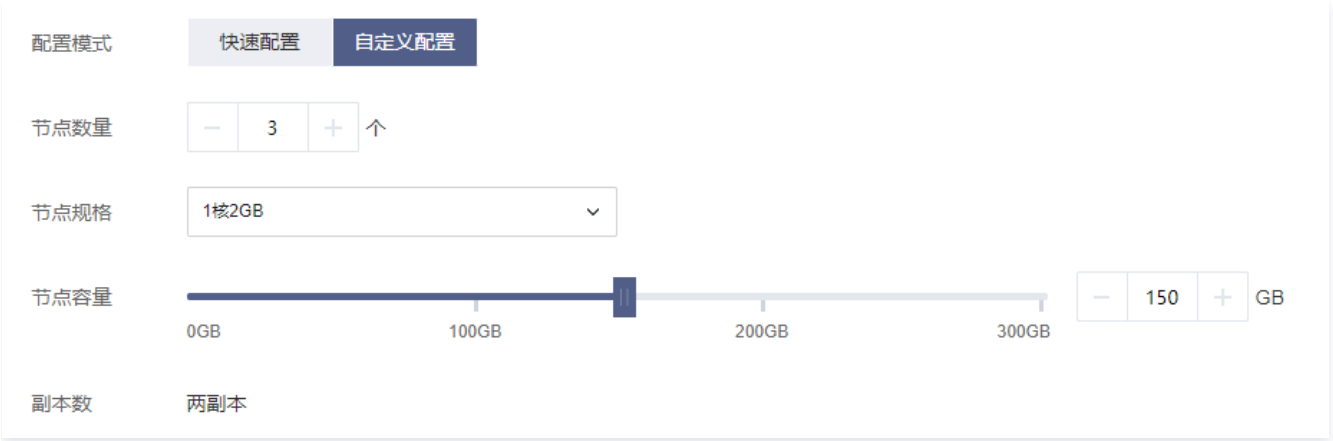
时序数据库 CTSDB 支持快速配置和自定义配置两种选择模式，下面分别详细叙述。建议您主要从三个方面需求来考虑如何选择合适的实例规格：性能需求、容量需求、其他要求。实例性能请参考 产品性能 。
进入 CTSDB 购买页，在选择模式处可按需选择对应模式。

快速配置

时序数据库 CTSDB 提供快速配置模式，该模式下，您只需要选择所存储的数据量，系统自动根据3节点两副本的模式来创建实例。另外，系统自动选择节点配置时，在节点存储容量相同的前提下，会优先选择 CPU 和内存较小的节点配置来分配实例。

自定义配置

在自定义配置模式下，您可自主选择节点数量，节点规格和节点容量。具体如下图所示：



注意：
在生产环境中，选择节点数量与节点规格，建议如下：

- 节点数量：配置至少3个节点。仅有2个节点的实例，无法完全保证数据的高可靠性，仅适用于测试环境。
- 节点规格：配置至少2核9GB。节点规格为1核2GB，内存过小无法保证高可用性，仅适用于测试环境。