时序数据库 CTSDB

快速入门

腾讯云

【联系我们】

我们致力于为您提供个性化的售前购买咨询服务，及相应的技术售后服务，任何问题请联系 4009100100或95716。

文档目录

快速入门
新建试用实例

最近更新时间：2024-08-30 17:24:41

操作场景

配置一台免费试用的时序数据库 CTSDB 3.0 版标准型实例集群。

地域

当前支持**广州、北京、上海**地域新建实例，后续地域在规划准备中。

前提条件

- 已注册腾讯云账号并完成实名认证，且时序数据库 CTSDB 3.0 版内测申请成功。
  - 如需注册腾讯云账号：请单击 注册腾讯云账号 。
  - 如需完成实名认证：请单击 实名认证 。
  - 如需内测试用申请，请单击 立即申请 ，填写相应产品测试申请单并提交。时序数据库 CTSDB 3.0版收到试用申请评估后，腾讯云数据库团队将邀请您进行产品体验、测试和商务洽谈。
- 已规划数据库实例需满足的规格。
- 已规划数据库实例的私有网络与安全组，请参见 私有网络 与 安全组 。

操作步骤

1. 使用腾讯云账号登录 时序数据库控制台 。
2. 在控制台页面上方，选择 **3.0版**。
3. 单击**新建试用实例**，进入**新建时序数据库 CTSDB 3.0版新建试用实例**页面。
4. 请参见下表，配置如下参数，购买实例。

地域 广州 清远 上海 北京 新加坡

处于不同地域的云产品内网不通，新建后不能切换地域，建议选择最靠近业务的地域。

可用区 多可用区

多可用区部署，数据副本自动分布在多个可用区，具有更高的可用性和容灾能力。
可与相同地域和私有网络下各个可用区的云产品内网互通；例如，相同私有网络下广州二区的云服务器可通过内网访问广州地域的时序数据库。

配置模式 按资源配置

时序计费模式 包年包月

时序节点规格 2核 8GB

时序节点数量 3个

存储计费模式 包年包月

预购存储容量，建议根据业务数据量的预估值填写

存储容量 100 GB 12288 GB 500 GB

数据副本 3 副本

存储类型 SSD本地盘

冷热数据分层 支持数据分层，可按需降冷到对象存储，冷数据按时间使用量计费。新建实例后可以在库管理中配置。

网络 redis

CIDR: 10.0.0.0/24，子网IP/可用IP: 253个/247个
当前网络选择下，仅"ruidaxu_vpc"网络的主机可访问数据库 新建私有网络 新建子网

标签 标签键 标签值

+ 添加 键值粘贴板

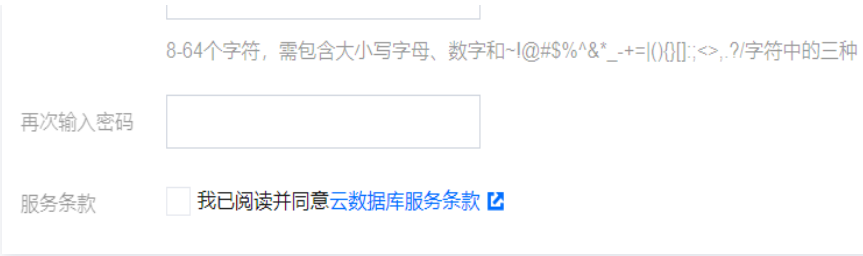
如现有的标签不合适，您可以去控制台新建标签键/标签值

安全组 请选择

如您业务需要放通其他端口，您可以自定义安全组

实例名 创建后命名 立即命名

密码



分类	界面参数	参数含义	配置方法
地域信息	**地域**	实例所属地域。	选择产品已经支持的地域。
	可用区	实例所属可用区。	固定为**多可用区**。
计算	**配置模式**	**按资源配置**：计算资源将根据计算规格及其节点数量配置。	适用于需要明确计算节点 CPU 核数和内存容量的业务场景。如何选择，请参见 实例规格 。
	时序计费模式	配置计算资源的计费方式。	仅支持**包年包月**。
	时序节点规格	**配置模式**选择为**按资源配置**，该参数配置计费资源的 CPU 核数与内存规格。	在下拉列表选择需要的计算规格。计算规格具体信息，请参见 实例规格 。
	时序节点数量	配置计算节点的数量。	取值范围：3 ~ 30。
存储	**存储计费模式**	选择存储资源的计费方式。	仅支持**包年包月**。
	存储容量	若**存储计费模式**选择**包年包月**，显示该参数。配置存储磁盘容量。	在滑轴上选择所需的磁盘容量。取值范围：[100GB,12288GB]。
	数据副本	选择存储数据的副本数量。	当前仅支持**3副本**。
	存储类型	选择存储的磁盘类型。	当前仅支持 **SSD 本地盘**。
网络	**网络**	选择具体的私有网络及其子网。 • 使用云服务器 CVM 连接自动分配给云数据库的内网地址，这种连接方式使用内网高速网络，延迟低。云服务器和数据库须是同一账号，且同一个 VPC 内（保障同一个地域）。 • 私有网络具有地域（Region）属性（如广州），而子网具有可用区（Zone）属性（如广州一区），私有网络可划分一个	• 在下拉列表分别选择已配置的私有网络及子网。 • 若现有网络不满足需求，请单击**新建私有网络**或**新建子网**，创建所需的网络环境。

		或多个子网，同一私有网络下不同子网默认内网互通，不同私有网络间（无论是否在同一地域）默认内网隔离。	
	标签	给实例设定标签。您可以根据标签归类管理实例。	• 在下拉列表选择已配置的标签。 • 若现有标签不合适，请单击添加，可以选择标签键与标签值。
	安全组	给实例设置安全组规则，以控制访问数据库的入流量。	您可以在**选择已有安全组**下拉框中选择已有的安全组，也可以单击**自定义安全组**，设置新的安全组入站规则。具体操作，请参见 安全组 。
实例	**实例名**	设置实例名称。	• **创建后命名**：实例名称默认将于实例 ID 保持一致，实例创建后，可根据需求修改。 • **立即命名**：请在下面输入框输入实例名称。名称要求：1-60个字符，可包含中文/大小写字母/数字/"-"/"_"/"."
	密码	设置实例访问密码。	密码复杂度要求：8-64个字符，需包含大小写字母、数字和~!@#$%^&*_-+=|(){}[]:;<>,.?/字符中的三种。
	再次输入密码	再次输入密码，保证密码准确性。	-
协议	**服务条款**	说明使用云数据库服务内容、服务费用、使用规则、知识产权等相关服务条款。	勾选**我已阅读并同意云数据库服务条款**。

5. 单击**立即申请**，自动返回实例列表页面，当前实例**状态**为**创建中**，等待实例状态更新为**运行中**即可使用。

连接并写入数据

最近更新时间：2024-08-30 10:43:51

本章节以 Linux 操作系统为例，协助您通过云服务器 CVM 内网，使用 HTTP 方式连接时序数据库 CTSDB 3.0 版实例。

前提条件

- 注册腾讯云账号，并完成 实名认证。
- 申请与时序数据库 CTSDB 3.0 版在同一地域同一个 VPC 内的 Linux 云服务器 CVM。
- 已 新建数据库实例，且状态为**运行中**。
- 获取实例的内网 IP 地址与网络端口。具体操作，请参见 查看实例。
- 获取实例的访问账号及密码信息。具体操作，请参见 账号管理。

操作步骤

步骤1：准备数据

如下为一些关于车辆的时序数据，每行代表一个车辆的信息，包括车辆的 id、所在城市、类型、速度和温度等。时间戳位于每行末尾，表示这些信息是在对应的时间戳时刻记录的。现将这些数据写入已创建的数据库实例中。

```
car,id=0,city=city_0,type=type_0 speed=100,temp=20 1675236656000000000
car,id=1,city=city_1,type=type_1 speed=101,temp=21 1675236656000000000
car,id=2,city=city_0,type=type_2 speed=102,temp=22 1675236656000000000
car,id=3,city=city_1,type=type_0 speed=103,temp=23 1675236656000000000
car,id=4,city=city_0,type=type_1 speed=104,temp=24 1675236656000000000
car,id=5,city=city_1,type=type_2 speed=105,temp=25 1675236656000000000
car,id=0,city=city_0,type=type_0 speed=106,temp=26 1675236657000000000
car,id=1,city=city_1,type=type_1 speed=107,temp=27 1675236657000000000
car,id=2,city=city_0,type=type_2 speed=108,temp=28 1675236657000000000
car,id=3,city=city_1,type=type_0 speed=100,temp=29 1675236657000000000
car,id=4,city=city_0,type=type_1 speed=101,temp=30 1675236657000000000
car,id=5,city=city_1,type=type_2 speed=102,temp=20 1675236657000000000
car,id=0,city=city_0,type=type_0 speed=103,temp=21 1675236658000000000
car,id=1,city=city_1,type=type_1 speed=104,temp=22 1675236658000000000
car,id=2,city=city_0,type=type_2 speed=105,temp=23 1675236658000000000
car,id=3,city=city_1,type=type_0 speed=106,temp=24 1675236658000000000
car,id=4,city=city_0,type=type_1 speed=107,temp=25 1675236658000000000
car,id=5,city=city_1,type=type_2 speed=108,temp=26 1675236658000000000
car,id=0,city=city_0,type=type_0 speed=100,temp=27 1675236659000000000
car,id=1,city=city_1,type=type_1 speed=101,temp=28 1675236659000000000
```

```
car,id=2,city=city_0,type=type_2 speed=102,temp=29 1675236659000000000
car,id=3,city=city_1,type=type_0 speed=103,temp=30 1675236659000000000
car,id=4,city=city_0,type=type_1 speed=104,temp=20 1675236659000000000
car,id=5,city=city_1,type=type_2 speed=105,temp=21 1675236659000000000
car,id=0,city=city_0,type=type_0 speed=106,temp=22 1675236660000000000
car,id=1,city=city_1,type=type_1 speed=107,temp=23 1675236660000000000
car,id=2,city=city_0,type=type_2 speed=108,temp=24 1675236660000000000
car,id=3,city=city_1,type=type_0 speed=100,temp=25 1675236660000000000
car,id=4,city=city_0,type=type_1 speed=101,temp=26 1675236660000000000
car,id=5,city=city_1,type=type_2 speed=102,temp=27 1675236660000000000
car,id=0,city=city_0,type=type_0 speed=103,temp=28 1675236661000000000
car,id=1,city=city_1,type=type_1 speed=104,temp=29 1675236661000000000
car,id=2,city=city_0,type=type_2 speed=105,temp=30 1675236661000000000
car,id=3,city=city_1,type=type_0 speed=106,temp=20 1675236661000000000
car,id=4,city=city_0,type=type_1 speed=107,temp=21 1675236661000000000
car,id=5,city=city_1,type=type_2 speed=108,temp=22 1675236661000000000
```

步骤2：登录云服务器 CVM

1. 登录 云服务器控制台 。
2. 在左侧导航栏，选择**实例**。
3. 在实例管理页面上方，选择地域。
4. 在实例列表中找到已申请的 CVM ，单击右侧**操作**列中的**登录** 。
5. 输入申请 CVM 时设置的用户名密码即可登录云服务器。

步骤3：创建数据库

CREATE DATABASE 的语法格式如下所示：

```
CREATE DATABASE <database_name> [WITH [DURATION <duration [默认无限制] >] ]
```

- **database_name**：指数据库库名。
- **DURATION**：数据过期删除时间（TTL，Time To Live），单位：天，取值为非零整数，选填参数。如果不启用数据过期，不配置该参数，如需启用数据过期，该数据库中的数据达到过期时间后将被自动删除清理。例如 DURATION 180d，表示该数据库中的数据达到180天即过期自动删除。

创建数据库 `time_series_test_db` ，执行格式如下所示：

```
curl --get http://${influxdb_ip}:8086/query \
     --user "time_series_test_account":"test_password" \
     --data-urlencode "pretty=true" \
```

```
    --data-urlencode 'q=CREATE DATABASE time_series_test_db'
```

- **${influxdb_ip}**：指数据库实例的内网 IP 地址。其端口默认为8086，请以实际情况进行替换。
- **user**：指访问实例的账号与密码信息，默认账号以实例 ID 命名。您可以在 控制台 的**账号管理**页面查看账号信息。具体操作，请参见 账号管理 。
- **data-urlencode "pretty=true"**： pretty 为 true，将以 JSON Pretty Print 格式返回信息。
- **data-urlencode**：即 CREATE DATABASE 的语法格式 ，如'q=CREATE DATABASE time_series_test_db'、'q=CREATE DATABASE time_series_test_db WITH DURATION 180d'。

返回如下信息，说明在创建中。

```
{
    "code": "invalid",
    "message": "Tsdb create database operation is executing, please try
again later(maybe failture)."
}
```

返回如下信息，说明创建成功。

```
{
    "results": [
        {
            "statement_id": 0
        }
    ]
}
```

执行 `show database` ，查看已经创建的数据库，执行格式如下所示：

```
curl --get http://${influxdb_ip}:8086/query \
    --user "time_series_test_account":"test_password" \
    --data-urlencode "pretty=true" \
    --data-urlencode 'q=show databases'
```

执行示例，如下所示：

```
[root@VM-16-41-centos ~]# curl --get http://10.16.12.8:8086/query \
>       --user "ctsdbi-8ym3****":"test@123" \
>       --data-urlencode "pretty=true" \
```

```
>       --data-urlencode 'q=show databases'
{
    "results": [
        {
            "statement_id": 0,
            "series": [
                {
                    "name": "databases",
                    "columns": [
                        "name"
                    ],
                    "columns_types": [
                        "string"
                    ],
                    "values": [
                        [
                            "test0407"
                        ],
                        [
                            "time_series_test_db"
                        ],
                        [
                            "_internal"
                        ],
                        [
                            "TestAlvinDatabase"
                        ],
                        [
                            "time_series_test_db0"
                        ]
                    ]
                }
            ]
        }
    ]
}
```

步骤4：写入数据

给数据库 time_series_test_db 写入数据。

```
curl -v --data-binary @/tmp/line_protocol_sample.txt \
     --user "time_series_test_account":"test_password" \
```

```
        -H "Content-Type: text/plain" \
        -H "Accept: application/json" -X POST \
        "http://${influxdb_ip}:8086/write?
db=time_series_test_db&precision=ns"
```

- **@/tmp/line_protocol_sample.txt：指写入数据的文件的相对路径。**
- **user：配置实例的访问账号与密码。**
- **influxdb_ip：指实例内网 IP 地址。**
- **db：指数据库库名。**
- **precision：时间数据的精度，可以使用的精度包括 ns（纳秒）、u（微秒）、ms（毫秒）、s（秒）、m（分钟）、h（小时）。**

执行示例如下，将 `/tmp/test.txt` 文档中的时序数据写入数据库 `time_series_test_db` 。

```
[root@VM-16-41-centos tmp]# curl -v --data-binary @/tmp/test.txt \
>       --user "ctsdbi-8ym3****":"test@123" \
>         -H "Content-Type: text/plain" \
>         -H "Accept: application/json" -X POST \
>         "http://10.16.12.8:8086/write?
db=time_series_test_db&precision=ns"
* About to connect() to 10.16.12.8 port 8086 (#0)
*   Trying 10.16.12.8...
* Connected to 10.16.12.8 (10.16.12.8) port 8086 (#0)
* Server auth using Basic with user 'ctsdbi-8ym3****'
> POST /write?db=time_series_test_db&precision=ns HTTP/1.1
> Authorization: Basic Y3RzZGJpLTh5********************
> User-Agent: curl/7.29.0
> Host: 10.16.12.8:8086
> Content-Type: text/plain
> Accept: application/json
> Content-Length: 2552
> Expect: 100-continue
>
* Done waiting for 100-continue
< HTTP/1.1 204 No Content
< Date: Mon, 10 Apr 2023 03:48:51 GMT
< Content-Type: application/json
< Content-Length: 0
<
* Connection #0 to host 10.16.12.8 left intact
```

写入数据返回信息中，`Date: Mon, 10 Apr 2023 03:48:51 GMT` 显示写入完成的时间点，无任何错误提示信息，说明写入成功。常见的错误信息，请参见下表。

错误信息	错误含义	处理方式
`{ "error": "NotFoundCollection cannot find collections" }`	连接失败	排查连接数据库的内网 IP 地址与端口是否正确。
`{ "error": "BadParam ERR wrong password.\r\n" }`	访问数据库的账户密码错误	请检查密码信息是否正确。
`{ "error": "NotFoundAccount ERR account id or name not exists: 1nB3pmg==" }`	访问数据的账户不存在	请检查账户信息是否输入错误。

<pre>{
 	
"error": "InvalidFieldFormat ParsePoint() failed|status=InvalidFieldFormat InvalidBooleanFormat not start with tTfF|line=727|col=14\ncar id=0,city=city_0,type=type_0 speed=100,temp=20 0\n
| start here"
 	
}</pre>
	数据写入失败	请检查
		`car id=0,city=city_0,type=type_0 speed=100,temp=20`
		这一行数据的第14个字符是否存在 Line Protocol 格式问题。

使用 `show measurements` 查看写入的 measurements，执行方式如下所示：

```
curl --get http://${influxdb_ip}:8086/query \
     --user "time_series_test_account":"test_password" \
     --data-urlencode 'db=time_series_test_db' \
     --data-urlencode "pretty=true" \
     --data-urlencode "q=show measurements"
```

执行示例，如下所示：

```
[root@VM-16-41-centos ~]# curl --get http://10.16.12.8:8086/query \
>       --user "ctsdbi-8ym3****":"test@123" \
>       --data-urlencode 'db=time_series_test_db' \
>       --data-urlencode "pretty=true" \
>       --data-urlencode "q=show measurements"
{
    "results": [
        {
            "statement_id": 0,
            "series": [
                {
                    "name": "measurements",
                    "columns": [
                        "name"
                    ],
                    "columns_types": [
                        "string"
```

```
                    ],
                    "values": [
                        [
                            "car"
                        ],
                        [
                            "d=0"
                        ]
                    ]
                }
            ]
        }
    ]
}
```

步骤5：查询数据

Select 语法格式如下所示：

```
select_stmt = "SELECT" fields from_clause [ where_clause ] [
group_by_clause ] [ order_by_clause ] [ limit_clause ] [ offset_clause ]
[ slimit_clause ] [ soffset_clause ] [ sql_order_by_clause ]
```

参数含义，请参见下表。

参数	参数含义
fields	指定需要查询的字段列表，可以是一个或多个字段，用逗号分隔。如果需要查询所有字段，可以使用通配符“*”。
from_clause	指定需要查询的 measurement 名称。
where_clause	指定查询的条件，可以是一个或多个条件，可以使用 AND、OR、NOT 等逻辑运算符组合多个条件。条件包括 tag 和 field，可以使用比较运算符（如=、!=、>、<等）进行比较。
group_by_clause	指定查询结果需要按照哪个 tag 进行分组，可以是一个或多个 tag，用逗号分隔。
order_by_clause	指定查询结果需要按照哪个字段进行排序，可以是一个或多个字段，用逗号分隔。默认情况下，查询结果按照时间戳排序。
limit_clause	指定查询结果的最大行数。
offset_clause	指定查询结果的起始行数。

slimit_clause	类似于 limit_clause，但仅应用于每个分组内部的结果集。
soffset_clause	类似于 offset_clause，但仅应用于每个分组内部的结果集。
sql_order_by_clause	类似于 order_by_clause，但是使用 SQL 语法进行排序。

查询全部数据执行方式，如下所示：

```
curl --get http://${influxdb_ip}:8086/query \
     --user "time_series_test_account":"test_password" \
     --data-urlencode 'db=time_series_test_db' \
     --data-urlencode "pretty=true" \
     --data-urlencode 'q=select * from car'
```

分页查询执行方式，如下所示：

```
curl --get http://${influxdb_ip}:8086/query \
     --user "time_series_test_account":"test_password" \
     --data-urlencode 'db=time_series_test_db' \
     --data-urlencode "pretty=true" \
     --data-urlencode 'q=select * from car limit 3 offset 2'
```

聚合函数查询执行方式，如下所示：

```
curl --get http://${influxdb_ip}:8086/query \
     --user "time_series_test_account":"test_password" \
     --data-urlencode 'db=time_series_test_db' \
     --data-urlencode "pretty=true" \
     --data-urlencode 'q=select max(speed) as fun1, min(speed) as fun2
from car group by time(3s), type fill(none)'
```

聚合函数执行返回结果，如下所示：

```
{
    "results": [
        {
            "statement_id": 0,
            "series": [
                {
                    "name": "car",
```

```
            "tags": {
                "type": "type_0"
            },
            "columns": [
                "time",
                "fun1",
                "fun2"
            ],
            "columns_types": [
                "time",
                "float",
                "float"
            ],
            "values": [
                [
                    "2023-02-01T07:30:54Z",
                    103.0,
                    100.0
                ],
                [
                    "2023-02-01T07:30:57Z",
                    106.0,
                    100.0
                ],
                [
                    "2023-02-01T07:31:00Z",
                    106.0,
                    100.0
                ]
            ]
        },
        {
            "name": "car",
            "tags": {
                "type": "type_1"
            },
            "columns": [
                "time",
                "fun1",
                "fun2"
            ],
            "columns_types": [
                "time",
```

```
                    "float",
                    "float"
                ],
                "values": [
                    [
                        "2023-02-01T07:30:54Z",
                        104.0,
                        101.0
                    ],
                    [
                        "2023-02-01T07:30:57Z",
                        107.0,
                        101.0
                    ],
                    [
                        "2023-02-01T07:31:00Z",
                        107.0,
                        101.0
                    ]
                ]
            },
            {
                "name": "car",
                "tags": {
                    "type": "type_2"
                },
                "columns": [
                    "time",
                    "fun1",
                    "fun2"
                ],
                "columns_types": [
                    "time",
                    "float",
                    "float"
                ],
                "values": [
                    [
                        "2023-02-01T07:30:54Z",
                        105.0,
                        102.0
                    ],
                    [
```

```
                            "2023-02-01T07:30:57Z",
                            108.0,
                            102.0
                        ],
                        [
                            "2023-02-01T07:31:00Z",
                            108.0,
                            102.0
                        ]
                    ]
                }
            ]
        }
    ]
}
```

步骤6：删除数据库

DELETE

删除具体的数据，语法格式如下所示：

```
DELETE FROM <measurement_name> WHERE [ <tag_key> =' <tag_value> '] |
[ <time interval> ]
```

- **measurement_name**：指定需要删除数据的 measurement 名称。
- **tag_key、tag_value**：可选参数，指定需要删除的数据点对应的 tag 键值对。如果指定了 tag_key 和 tag_value，则只会删除符合条件的数据点。如果不指定 tag_key 和 tag_value，则会删除所有 measurement_name 中的数据点。
- **time interval**：可选参数，指定需要删除的时间范围。可以使用以下格式进行指定：
  - 时间戳：删除指定时间戳的数据点，例如："time >= 1434059627 and time <= 1434060027"
  - 相对时间：删除相对于当前时间一定时间范围内的数据点，例如："time >= now() - 1h"
  - 时间段：删除指定时间段内的数据点，例如："time >= '2018-01-01T00:00:00Z' and time <= '2018-01-02T00:00:00Z'"

执行方式，如下所示：

```
curl --get http://${influxdb_ip}:8086/query \
     --user "time_series_test_account":"test_password" \
     --data-urlencode 'db=time_series_test_db' \
     --data-urlencode "pretty=true" \
```

```
      --data-urlencode "q=delete from car where city = 'city_0'"
```

DROP MEASUREMENT

删除 measurement 语法格式，如下所示：

```
DROP MEASUREMENT <measurement_name>
```

执行方式如下所示：

```
curl --get http://${influxdb_ip}:8086/query \
     --user "time_series_test_account":"test_password" \
     --data-urlencode 'db=time_series_test_db' \
     --data-urlencode "pretty=true" \
     --data-urlencode 'q=drop measurement car'
```

DROP DATABASE

删除数据库，语法格式如下所示：

```
DROP DATABASE <database_name>
```

执行方式，如下所示：

```
curl --get http://${influxdb_ip}:8086/query \
     --user "time_series_test_account":"test_password" \
     --data-urlencode "pretty=true" \
     --data-urlencode 'q=DROP DATABASE time_series_test_db'
```

返回如下信息，说明删除成功。

```
{
    "results": [
        {
            "statement_id": 0
        }
    ]
}
```

更多参考

- 时序数据库 CTSDB 3.0 版与原生的 InfluxDB 使用相同。具体如何管理数据库，请参见 InfluxDB®官方文档。
- 时序数据库CTSDB 3.0 版使用了 InfluxDB 原生 SDK。多语言 SDK 的示例，请参见 InfluxDB®客户端文档。