云数据库 Tendis

操作指南

产品文档

腾讯云

文档目录

操作指南
- 多语言连接
  - .Net 连接示例
  - C 连接示例
  - Go 连接示例
  - Java 连接示例
  - Node.js 连接示例
  - PHP 连接示例
  - Python 连接示例
- 维护管理实例
  - 为实例指定项目
  - 扩容实例规格
  - 销毁实例
- 监控功能
- 配置安全组
- 禁用命令

操作指南
多语言连接
.Net 连接示例

最近更新时间：2021-01-15 16:35:28

运行前必备：

下载并安装 ServiceStack.Redis。

示例代码：

- 不使用连接池

```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using ServiceStack.Redis;
using System;

namespace ConsoleApplication1
{
class Program
{
static void Main(string[] args)
{
string host = "10.xx.xx.46";//实例访问 host 地址
int port = 6379;// 端口信息
string instanceId = "bd87dadc-8xx1-4xx1-86dd-021xxxcde96";//实例 ID
string pass = "1234567q";//密码

RedisClient redisClient = new RedisClient(host, port, instanceId + ":" + pass);
string key = "name";
string value = "QcloudV5!";
redisClient.Set(key, value); //设置值
System.Console.WriteLine("set key:[" + key + "]value:[" + value + "]");
string getValue = System.Text.Encoding.Default.GetString(redisClient.Get(key));
//读取值
System.Console.WriteLine("value:" + getValue);
System.Console.Read();
}
}
}
```

- 使用 ServiceStack 4.0 连接池

```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using ServiceStack.Redis;
using System;

namespace ConsoleApplication2
{
class Program
{
static void Main(string[] args)
{
string[] testReadWriteHosts = new[] {
"redis://:fb92bxxxabf11e5:1234xx8a1A@10.x.x.1:6379"/*redis://:实例ID:密码@访问地址:端口*/
};
RedisConfig.VerifyMasterConnections = false;//需要设置
PooledRedisClientManager redisPoolManager = new PooledRedisClientManager(10/*连接池个数*/,
10/*连接池超时时间*/, testReadWriteHosts);
for (int i = 0; i < 100; i++)
{
IRedisClient redisClient = redisPoolManager.GetClient();//获取连接
RedisNativeClient redisNativeClient = (RedisNativeClient)redisClient;
redisNativeClient.Client = null;//需要设置
try
{
string key = "test1111";
string value = "test1111";
redisClient.Set(key, value);
redisClient.Dispose();//
}
catch (Exception e)
{
System.Console.WriteLine(e.Message);
}
}
System.Console.Read();
}
}
}
```

- 使用ServiceStack 3.0 连接池

```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using ServiceStack.Redis;
using System;

namespace ConsoleApplication3
{
class Program
{
static void Main(string[] args)
{
string[] testReadWriteHosts = new[] {
"fb92bfxxbf11e5:123456xx1A@10.x.x.1:6379" /*实例ID:密码@访问地址:端口*/
};
PooledRedisClientManager redisPoolManager = new PooledRedisClientManager(10/*连接池个
数*/, 10/*连接池超时时间*/, testReadWriteHosts);
for (int i = 0; i < 100; i++)
{
IRedisClient redisClient = redisPoolManager.GetClient();//获取连接
try
{
string key = "test1111";
string value = "test1111";
redisClient.Set(key, value);
redisClient.Dispose();//
}
catch (Exception e)
{
System.Console.WriteLine(e.Message);
}
}
System.Console.Read();
}
}
}
```

运行结果：

C 连接示例

最近更新时间：2021-01-15 16:35:28

运行前必备:

下载并安装 hiredis。

示例代码：

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include <hiredis.h>

int main(int argc, char **argv) {
unsigned int j;
redisContext *c;
redisReply *reply;

if (argc < 4) {
printf("Usage: 192.xx.xx.195 6379 instance_id password\n");
exit(0);
}
const char *hostname = argv[1];
const int port = atoi(argv[2]);
const char *instance_id = argv[3];
const char *password = argv[4];


struct timeval timeout = { 1, 500000 }; // 1.5 seconds
c = redisConnectWithTimeout(hostname, port, timeout);
if (c == NULL || c->err) {
if (c) {
printf("Connection error: %s\n", c->errstr);
redisFree(c);
} else {
printf("Connection error: can't allocate redis context\n");
}
exit(1);
}

/* AUTH */
reply = redisCommand(c, "AUTH %s", password);
printf("AUTH: %s\n", reply->str);
freeReplyObject(reply);
```

```
/* PING server */
reply = redisCommand(c,"PING");
printf("PING: %s\n", reply->str);
freeReplyObject(reply);

/* Set a key */
reply = redisCommand(c,"SET %s %s", "name", "credis_test");
printf("SET: %s\n", reply->str);
freeReplyObject(reply);

/* Try a GET */
reply = redisCommand(c,"GET name");
printf("GET name: %s\n", reply->str);
freeReplyObject(reply);

/* Disconnects and frees the context */
redisFree(c);

return 0;
}
```

运行结果：

```
[root@VM_0_194_centos hiredis]# ./example  192 [redacted] 195 6379 84ffd722-b506-4934
-9025-645bb2a0997b 1234567q
AUTH: OK
PING: PONG
SET: OK
GET name: credis_test
[root@VM_0_194_centos hiredis]#
```

Go 连接示例

最近更新时间：2021-01-15 16:35:28

运行前必备：

下载客户端 Go-redis。

示例代码：

```
package main

import(

"fmt"

"redis"

"log"

)

func main() {

const host=192.xx.xx.195
const port=6379
const instanceId="84ffd722-b506-4934-9025-64xxx997b"
const pass="123d7sq"
// 连接 Tendis 服务器 192.xx.xx.195:6379 并授权 instanceId 密码
spec := redis.DefaultSpec().Host(host).Port(port).Password(instanceId+":"+pass);
client, err := redis.NewSynchClientWithSpec(spec)

if err != nil { // 是否连接出错

log.Println("error on connect redis server")

return

}

newvalue :=[]byte("QcloudV5!");

err=client.Set("name",newvalue);

if err != nil { // 设置值出错

log.Println(err)
```

```
return

}

value, err := client.Get("name") // 取值

if err != nil {

log.Println(err)

return

}

fmt.Println("name value is:",fmt.Sprintf("%s", value)) //输出

}
```

运行结果：

```
[root@VM_0_194_centos go_src]# go run testRedis.go
name value is: QcloudV5!
[root@VM_0_194_centos go_src]#
```

Java 连接示例

最近更新时间：2021-01-15 16:35:28

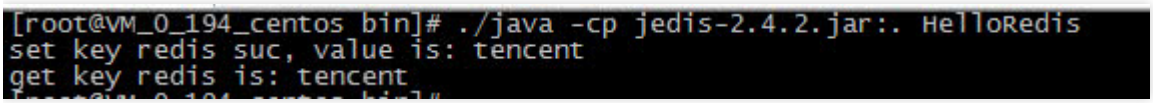
运行前必备：

下载客户端 Jedis。

示例代码：

```
import redis.clients.jedis.Jedis;

public class HelloRedis {

public static void main(String[] args) {
try {
/**以下参数分别填写您的 Tendis 实例内网 IP、端口号、实例 ID 和密码*/
String host = "192.xx.xx.195";
int port = 6379;
String instanceid = "crs-09xxxqv";
String password = "123ad6aq";
//连接Tendis
Jedis jedis = new Jedis(host, port);
//鉴权
jedis.auth(instanceid + ":" + password);

/**接下来可以开始操作 Tendis 实例，可以参考 https://github.com/xetorthio/jedis */
//设置 Key
jedis.set("redis", "tencent");
System.out.println("set key redis suc, value is: tencent");
//获取 Key
String value = jedis.get("redis");
System.out.println("get key redis is: " + value);

//关闭退出
jedis.quit();
jedis.close();
} catch (Exception e) {
e.printStackTrace();
}
}
}
```

运行结果：


[root@VM_0_194_centos bin]# ./java -cp jedis-2.4.2.jar:. HelloRedis
set key redis suc, value is: tencent
get key redis is: tencent

Node.js 连接示例

最近更新时间：2021-01-15 16:35:28

运行前必备：

执行以下命令，安装 node-redis：

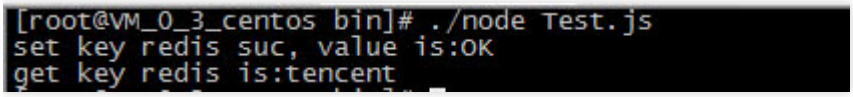
```
npm install hiredis redis
```

示例代码：

```
var redis = require("redis");

/**以下参数分别填写您的 Redis 实例内网 IP、端口号、实例 ID 和密码*/
var host = "192.xx.xx.2",
port = "6379",
instanceid = "c53xx52f-55dc-4c22-a941-630xxx88",
pwd = "12as6zb";
//连接 Tendis
var client = redis.createClient(port, host, {detect_buffers: true});
// Tendis 连接错误
client.on("error", function(error) {
console.log(error);
});
//鉴权
client.auth(instanceid + ":" + pwd);

/**接下来可以开始操作 Tendis 实例 */
//设置 Key
client.set("redis", "tencent", function(err, reply){
if (err) {
console.log(err);
return;
}
console.log("set key redis " + reply.toString() + ", value is tencent");
});

//获取 Key
client.get("redis", function (err, reply) {
if (err) {
console.log(err);
return;
}
console.log("get key redis is:" + reply.toString());
//程序结束关闭客户端
```

```
client.end();
});
```

运行结果：



PHP 连接示例

最近更新时间：2021-01-15 16:35:29

运行前必备：

下载客户端 phpredis。

示例代码：

```
<?php
/**以下参数分别填写您的 Tendis 实例内网 IP、端口号、实例 ID 和密码*/
$host = "192.xx.xx.2";
$port = 6379;
$pwd = "123tj6na";

$redis = new Redis();
//连接 Tendis
if ($redis->connect($host, $port) == false) {
die($redis->getLastError());
}
//鉴权
if ($redis->auth($pwd) == false) {
die($redis->getLastError());
}

/**接下来可以开始操作 Tendis 实例，可以参考 https://github.com/phpredis/phpredis */

//设置 Key
if ($redis->set("redis", "tencent") == false) {
die($redis->getLastError());
}
echo "set key redis suc, value is:tencent\n";

//获取 Key
$value = $redis->get("redis");
echo "get key redis is:".$value."\n";
?>
```

运行结果：

```
[root@VM_0_3_centos bin]# ./php Test.php
set key redis suc, value is:tencent
get key redis is:tencent
```

Python 连接示例

最近更新时间：2021-07-01 15:52:10

运行前必备：

下载并安装 redis-py。

示例代码：

```
#!/usr/bin/env python
#-*- coding: utf-8 -*-
import redis
#这里替换为连接的实例 host 和 port
host = '192.xx.xx.195'
port = 6379
#这里替换为实例 password
pwd='password'
#连接时通过 password 参数指定 AUTH 信息
r= redis.StrictRedis(host=host, port=port, password=pwd)
#连接建立后就可以进行数据库操作，请参见 https://github.com/andymccurdy/redis-py
r.set('name', 'python_test');
print r.get('name')
```

运行结果：

```
[root@VM_0_194_centos fasterquan]# python redis-python.py
python_test
[root@VM_0_194_centos fasterquan]#
```

维护管理实例
为实例指定项目

最近更新时间：2023-12-21 20:17:38

本文为您介绍如何通过控制台将实例分配至不同的项目进行管理。

操作场景

云数据库 Tendis 支持将实例分配至不同的项目进行管理，已指定项目的实例也可重新指定到其他项目。

说明：

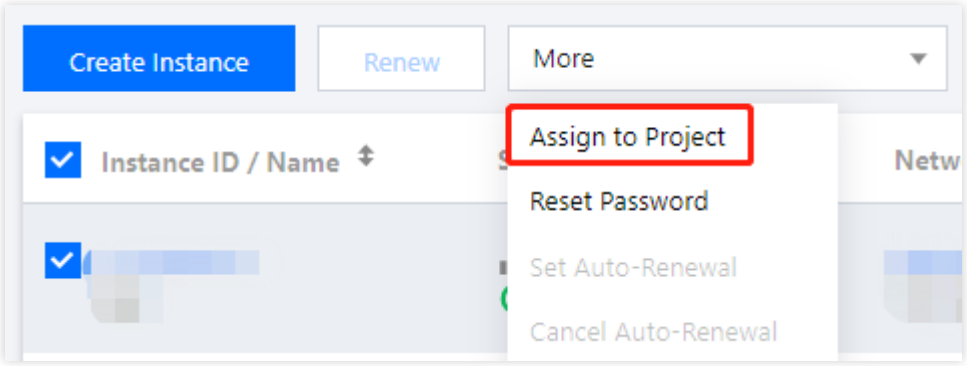
数据库实例在项目间进行分配和移动，不会影响实例对外提供的服务。

操作步骤

1. 登录 Tendis 控制台，选择对应地域，在实例列表选择所需实例，在上方选择**更多操作>分配至项目**。

说明：

或单击实例名，在实例详情页的**所属项目**处，单击**分配至项目**来调整项目。



2. 在弹出的对话框，选择对应项目，单击**确定**。

扩容实例规格

最近更新时间：2023-12-21 20:17:54

本文为您介绍如何通过 Tendis 控制台扩容实例节点规格和分片磁盘容量。

操作场景

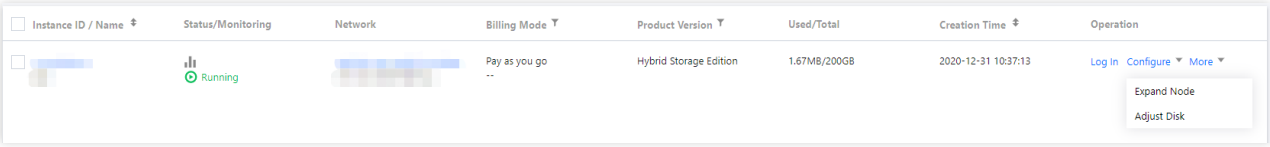
云数据库 Tendis 支持扩容实例的规格，提供灵活的扩容操作。您可根据业务所处的实际情况灵活调整 Tendis 实例的规格，从而更好满足资源充分调整优化等需求。

说明：

Tendis 实例暂不支持缩容，存储版暂不支持扩容。

操作步骤

1. 登录 Tendis 控制台，选择对应地域，在实例列表选择所需实例，在**操作**列选择**配置变更** > **扩容缓存容量**或**扩容磁盘容量**。



2. 在弹出的对话框，选择需更改的配置，单击**确定**。

说明：

配置变更后，实例将按照新的规格计费。

为了避免缩容失败，缩容后的实例容量要求大于或等于现有数据量的1.3倍。

分片的新增和删除操作，系统将自动均衡 Slot 配置，并且迁移数据。

阻塞命令 BLPOP、BRPOP、BRPOPLPUSH、SUBSCRIBE 在扩缩容期间会存在1次或者多次命令失败（影响次数和分片数量相关），请在操作请评估好对业务的影响。

开启副本只读功能的实例，在扩缩容期间，会有1次或者多次的命令失败（影响次数和分片数量相关），请在操作请评估好对业务的影响。

3. 返回实例列表，待实例状态变为**运行中**，即可正常使用。

销毁实例

最近更新时间：2023-12-21 20:18:07

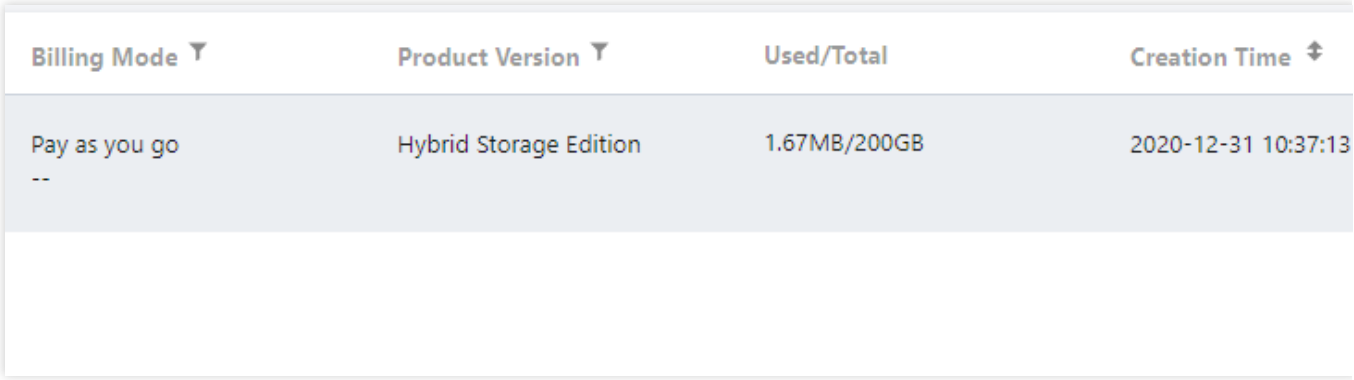
本文为您介绍如何通过 Tendis 控制台销毁实例。

操作场景

根据业务需求，您可以在控制台自助退还按量计费实例。
按量计费实例退还后，实例被移入云数据库回收站保留24小时，期间实例无法访问。如您想恢复该实例，可在回收站进行开机恢复。
自助退还后，实例的状态一旦变为“隔离中”时，就不再产生与该实例相关的费用。
注意：
实例销毁后数据将无法找回，备份文件会同步销毁，无法在云上进行数据恢复，请提前做好备份文件的转存。
实例销毁后 IP 资源同时释放。

操作步骤

1. 登录 Tendis 控制台，选择对应地域，在实例列表选择所需实例，在**操作**列选择**更多>销毁**。



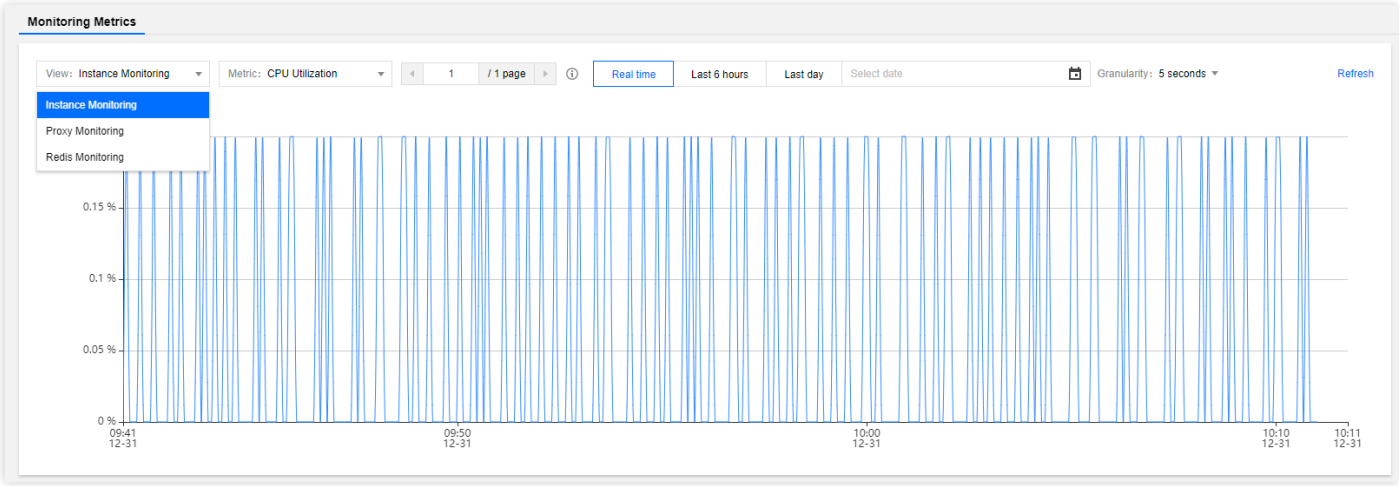
2. 在弹出的对话框，确认无误后，单击**销毁**。
注意：
销毁后所有数据将被清除且不可恢复，请谨慎操作。

监控功能

最近更新时间：2022-01-18 15:28:15

云数据库 Tendis 提供完整透明的监控服务，监控提供了包括 Proxy 监控、Redis 监控和 Tendis 监控汇总，详情如下：

- Proxy 监控：云数据库 Tendis 标准架构和集群架构都包含 Proxy，监控服务提供实例所有 Proxy 节点的监控信息。
- Redis 监控：包含了 Tendis 主节点和副本节点的监控信息。
- Tendis 监控：汇总了整个实例的监控数据，包括 Proxy 节点和 Tendis 节点的监控数据，通过 SUM、AVG、MAX、LAST 等聚合算法聚合而成。



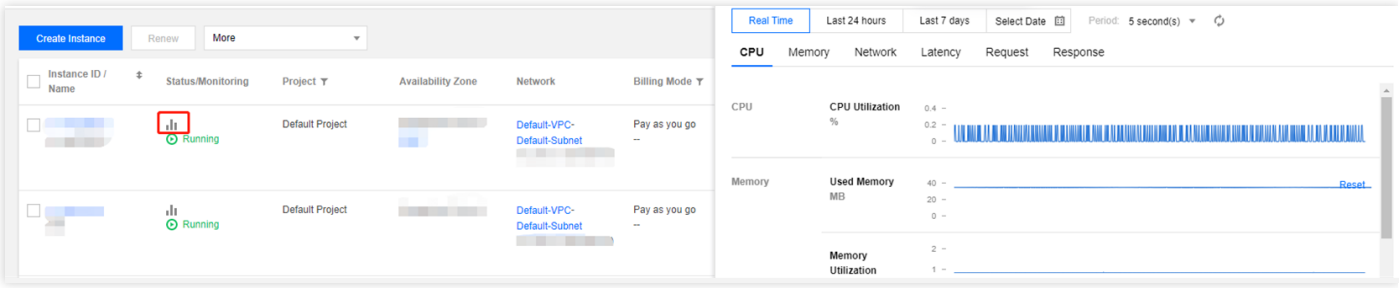
监控粒度/保留时长

Tendis 目前支持5秒、1分钟、5分钟、1小时、1天的粒度的指标监控，各粒度监控数据保留时长请参见 使用约束。

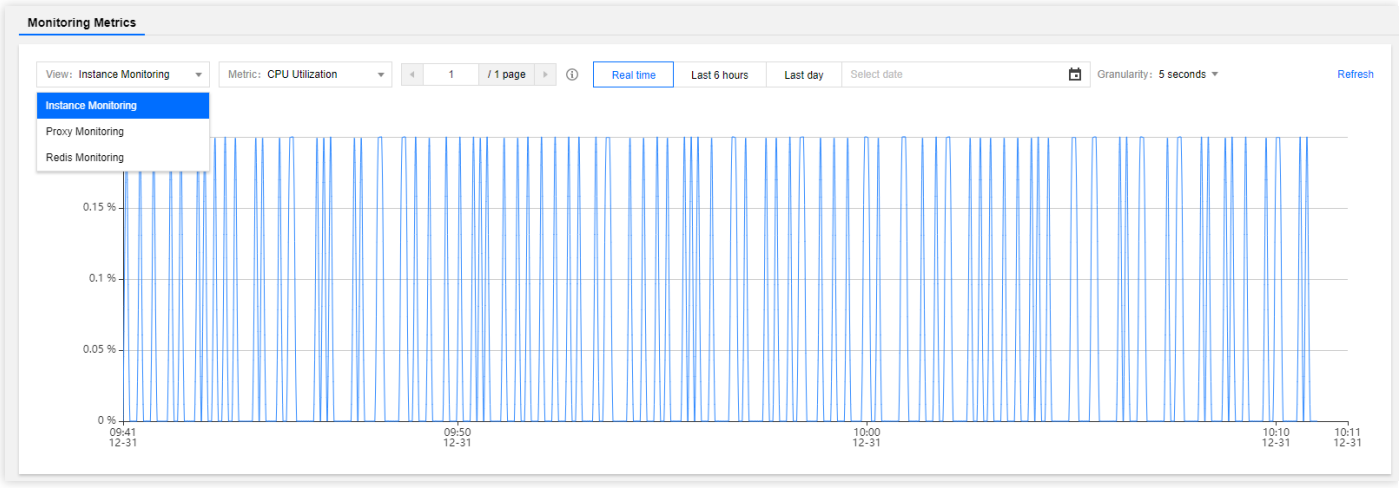
查看监控

您可以通过 Tendis 实例列表、Tendis 实例监控页面、云监控控制台3个地方查看云数据库 Tendis 的监控信息：

- 实例列表：登录 Tendis 控制台，在实例列表，单击如下监控图标，可快速浏览监控指标。



- 实例监控页面：登录 Tendis 控制台，单击实例 ID 进入实例管理页面，选择**系统监控>监控指标**，可查看实例监控信息详情。



监控指标说明

Proxy 监控

每个 Tendis 实例包含了至少3个 Proxy 节点，通常 Proxy 节点数是 Tendis 节点数量的1.5倍，Proxy 节点提供以下监控信息：

分组	指标	指标名称	单位	TIPS
CPU	CPU 使用率	cpu_util	%	Proxy CPU 使用率
请求	总请求	proxy_commands	次/秒	Proxy 执行的命令数
	Key 请求数	cmd_key_count	个/秒	命令访问的 Key 个数

分组	指标	指标名称	单位	TIPS
	Mget 请求数	cmd_mget	次/秒	Mget 命令执行次数
	执行错误	cmd_err	次/秒	Proxy 命令执行错误的次数，例如，命令不存在、参数错误等情况
	大 Value 请求	cmd_big_value	次/秒	请求命令大小超过32KB的执行次数
网络监控	连接数量	connections	个	连接到实例的 TCP 连接数量
	连接使用率	connections_util	%	实际 TCP 连接数量和最大连接数比
	入流量	in_flow	Mb/s	内网入流量
	入流量使用率	in_bandwidth_util	%	内网入流量实际使用和最大流量比
	入流量限流触发	in_flow_limit	次	入流量触发限流的次数
	出流量	out_flow	Mb/s	内网出流量
	出流量使用率	out_bandwidth_util	%	内网出流量实际使用和最大流量比
	出流量限流触发	out_flow_limit	次	出流量触发限流的次数
时延监控	平均执行时延	latency_avg	ms	Proxy 到 Redis Server 的执行时延平均值
	最大执行时延	latency_max	ms	Proxy 到 Redis Server 的执行时延最大值
	读平均时延	latency_read	ms	Proxy 到 Redis Server 的读命令平均执行时延，读命令分类，请参见 命令分类
	写平均时延	latency_write	ms	Proxy 到 Redis Server 的写命令平均执行时延，写命令分类，请参见 命令分类
	其他命令平均时延	latency_other	ms	Proxy 到 Redis Server 的读写命令之外的命令平均执行时延

Redis 监控

Redis 监控提供整个实例/集群所有主节点和从节点的监控信息，提供以下监控指标：

分组	指标	指标名称	单位	TIPS
CPU 监控	CPU 使用率	cpu_util	%	平均 CPU 使用率
网络	连接数量	connections	个	Proxy 连接到节点的连接数
	连接使用率	connections_util	%	节点连接数使用率
内存监控	内存使用量	mem_used	MB	实际使用内存容量，包含数据和缓存部分
	内存使用率	mem_util	%	实际使用内存和申请总内存之比
	Key 总个数	keys	个	实例存储的总 Key 个数（一级 Key）
	key 过期数	expired	个	时间窗内被淘汰的 Key 个数，对应 info 命令输出的 expired_keys
	key 驱逐数	evicted	个	时间窗内被驱逐的 Key 个数，对应 info 命令输出的 evicted_keys
	复制延迟	repl_delay	Byte	副本节点的相对主节点命令延迟长度
请求监控	总请求	commands	次/秒	QPS，命令执行次数
	读请求	cmd_read	次/秒	读命令执行次数，读命令分类，请参见 命令分类
	写请求	cmd_write	次/秒	写命令执行次数，写命令分类，请参见 命令分类
	其他请求	cmd_other	次/秒	读写命令之外的命令执行次数
响应监控	慢查询	cmd_slow	次	执行时延大于 slowlog-log-slower-than 配置的命令次数
	读请求命中	cmd_hits	次	读请求 Key 存在的个数，对应 info 命令输出的 keyspace_hits 指标
	读请求 Miss	cmd_miss	次	读请求 Key 不存在的个数，对应 info 命令输出的 keyspace_misses 指标

分组	指标	指标名称	单位	TIPS
	读请求命中率	cmd_hits_ratio	%	Key 命中 \ (Key命中 + KeyMiss)，该指标可以反应 Cache Miss 的情况

Tendis 监控

Tendis 监控汇总了整个实例的监控数据，由 Proxy 节点和 Redis 的监控数据，通过 SUM、AVG、MAX、LAST 等聚合算法聚合而成。

分组	指标中文名	关联视图	指标英文名	单位	指标说
CPU 监控	CPU 使用率	Tendis 节点	cpu_util	%	平均 CPU 使用率
	节点最大 CPU 使用率	Tendis 节点	cpu_max_util	%	实例中节点（分片或者副本）最大 CPU 使用率
内存监控	内存使用量	Tendis 节点	mem_used	MB	实际使用内存容量，包含数据和缓存部分
	内存使用率	Tendis 节点	mem_util	%	实际使用内存和申请总内存之比
	节点最大内存使用率	Tendis 节点	mem_max_util	%	实例中节点（分片或者副本）最大内存使用率
	Key 总个数	Tendis 节点	keys	个	实例存储的总 Key 个数（一级 Key）
	Key 过期数	Tendis 节点	expired	个	时间窗内被淘汰的 Key 个数，对应 info 命令输出的 expired_keys
	Key 驱逐数	Tendis 节点	evicted	个	时间窗内被驱逐的 Key 个数，对应 info 命令输出的 evicted_keys
网络监控	连接数量	Proxy 节点	connections	个	连接到实例的 TCP 连接数量
	连接使用率	Proxy 节点	connections_util	%	实际 TCP 连接数量和最大连接数比
	入流量	Proxy 节点	in_flow	Mb/s	内网入流量
	入流量使用率	Proxy 节点	in_bandwidth_util	%	内网入流量实际使用和最大流量比

分组	指标中文名	关联视图	指标英文名	单位	指标说
	入流量限流触发	Proxy 节点	in_flow_limit	次	入流量触发限流的次数
	出流量	Proxy 节点	out_flow	Mb/s	内网出流量
	出流量使用率	Proxy 节点	out_bandwidth_util	%	内网出流量实际使用和最大流量比
	出流量限流触发	Proxy 节点	out_flow_limit	次	出流量触发限流的次数
	平均执行时延	Proxy 节点	latency_avg	ms	Proxy 到 Redis Server 的执行时延平均值
	最大执行时延	Proxy 节点	latency_max	ms	Proxy 到 Redis Server 的执行时延最大值
	读平均时延	Proxy 节点	latency_read	ms	Proxy 到 Redis Server 的读命令平均执行时延，读命令分类，请参见 命令分类
	写平均时延	Proxy 节点	latency_write	ms	Proxy 到 Redis Server 的写命令平均执行时延，写命令分类，请参见 命令分类
	其他命令平均时延	Proxy 节点	latency_other	ms	Proxy 到 Redis Server 的读写命令之外的命令平均执行时延
请求监控	总请求	Tendis 节点	commands	次/秒	QPS，命令执行次数
	读请求	Tendis 节点	cmd_read	次/秒	读命令执行次数，读命令分类，请参见 命令分类
	写请求	Tendis 节点	cmd_write	次/秒	写命令执行次数，写命令分类，请参见 命令分类
	其他请求	Tendis 节点	cmd_other	次/秒	读写命令之外的命令执行次数
	大 Value 请求	Proxy 节点	cmd_big_value	次/秒	请求命令大小超过32KB的执行次数

分组	指标中文名	关联视图	指标英文名	单位	指标说
	Key 请求数	Proxy 节点	cmd_key_count	个/秒	命令访问的 Key 个数
	Mget 请求数	Proxy 节点	cmd_mget	个/秒	Mget 命令执行次数
	慢查询	Tendis 节点	cmd_slow	次	执行时延大于 slowlog - log - slower - than 配置的命令次数
	读请求命中	Tendis 节点	cmd_hits	次	读请求 Key 存在的个数，对应 info 命令输出的 keyspace_hits 指标
	读请求Miss	Tendis 节点	cmd_miss	次	读请求 Key 不存在的个数，对应 info 命令输出的 keyspace_misses 指标
	执行错误	Proxy 节点	cmd_err	次	命令执行错误的次数，例如，命令不存在、参数错误等情况
	读请求命中率	Tendis 节点	cmd_hits_ratio	%	Key 命中 / (Key 命中 + KeyMiss)，该指标可以反应 Cache Miss 的情况

命令分类

命令分类	列表
读命令	get,strlen,exists,getbit,getrange,substr,mget,llen,lindex,lrange,sismember,scard,srandmember,sinter,sunion,sdiff,smembers,sscan,zrange,zrangebyscore,zrevrangebyscore,zrangebylex,zrevrangebylex,zcount,zlexcount,zrevrange,zcard,zscore,zrank,zrevrank,zscan,hget,hmget,hlen,hstrlen,hkeys,hvals,hgetall,hexists,hscan,randomkey,keys,scan,dbsize,type,ttl,touch,pttl,dump,object,memory,bitcount,bitpos,georadius_ro,georadiusbymember_ro,geohash,geopos,geodist,p
写命令	set,setnx,setex,psetex,append,del,unlink,setbit,bitfield,setrange,incr,decr,rpush,lpush,rpushx,lpushx,linsert,rpop,lpop,brpop,brpoplpush,blpop,lset,ltrim,lrem,rpoplpush,sadd,srem,smove,spop,sinterstore,sunionstore,sdiffstore,zadd,zincrby,zrem,zremrangebyscore,zremrangebyrank,zremrangebylex,zunionstore,zinterstore,hset,hsetnx,hmset,hincrby,hincrbyfloat,hdel,incrby,decrby,incrbyfloat,getset,mset,msetnx,swapdb,move,rename,renamenx,expire,expireat,pexpire,pexpireat,flushdb,flushall,sort,persist,restore,restore-asking,migrate,bitop,geoadd,georadius,georadiusbymembe pfadd,pfmerge,pfdebug

配置安全组

最近更新时间：2023-12-21 20:20:56

操作场景

安全组 是一种有状态的包含过滤功能的虚拟防火墙，用于设置单台或多台云数据库的网络访问控制，是腾讯云提供的重要的网络安全隔离手段。安全组是一个逻辑上的分组，您可以将同一地域内具有相同网络安全隔离需求的云数据库实例加到同一个安全组内。云数据库与云服务器等共享安全组列表，安全组内基于规则匹配，具体规则与限制请参见 安全组详细说明。

注意：

云数据库安全组目前仅支持私有网络 VPC 内网访问和外网访问的网络控制，暂不支持对基础网络的网络控制。
仅广州、上海、北京、成都地域支持数据库外网访问的安全组，其他地域暂不支持。
由于云数据库没有主动出站流量，因此出站规则对云数据库不生效。

为云数据库配置安全组

步骤一：创建安全组

1. 登录 云服务器控制台。
2. 在左侧导航选择**安全组**页，选择地域，单击**新建**。
3. 在弹出来的对话框中，完成如下配置，确认后单击**确定**即可。

模板：根据安全组中的数据库实例需要部署的服务，选择合适的模板，简化安全组规则配置。如下表所示：

模板	说明	说明
放通全部端口	默认放通全部端口到公网和内网，具有一定安全风险。	-
放通22，80，443，3389端口和ICMP协议	默认放通22，80，443，3389端口和 ICMP 协议，内网全放通。	此模板对云数据库不生效。
自定义	安全组创建成功后，按需自行添加安全组规则。具体操作请参见下文**添加安全组规则**。	-

名称：自定义设置安全组名称。
所属项目：默认选择**默认项目**，可指定为其他项目，便于后期管理。
备注：自定义，简短地描述安全组，便于后期管理。

步骤二：添加安全组规则

1. 在 安全组页，在需要设置规则的安全组行中，单击**操作**列的**修改规则**。
2. 在安全组规则页面，单击**入站规则>添加规则**。
3. 在弹出的对话框中，设置规则。

类型：默认选择**自定义**，您也可以选择其他系统规则模板。

来源：流量的源（入站规则） 或目标（出站规则），请指定以下选项之一：

指定的源/目标	说明
单个 IPv4 地址或 IPv4 地址范围	用 CIDR 表示法（如203.0.113.0、203.0.113.0/24或者0.0.0.0/0，其中0.0.0.0/0代表匹配所有 IPv4 地址）。
单个 IPv6 地址或 IPv6 地址范围	用 CIDR 表示法（如FF05::B5、FF05:B5::/60、::/0或者0::0/0，其中::/0或者0::0/0代表匹配所有 IPv6 地址）。
引用安全组 ID，您可以引用以下安全组的 ID： 安全组 ID 其他安全组	当前安全组表示与安全组关联的云服务器。 其他安全组表示同一区域中同一项目下的另一个安全组 ID。
引用 参数模板 中的 IP 地址对象或 IP 地址组对象	-

协议端口：填写协议类型和端口范围，您也可以引用 参数模板 中的协议端口或协议端口组。

说明：

连接云数据库 Tendis 须开通　6379协议端口。

策略：默认选择**允许**。

允许：放行该端口相应的访问请求。

拒绝：直接丢弃数据包，不返回任何回应信息。

备注：自定义，简短地描述规则，便于后期管理。

4. 单击**完成**，完成安全组入站规则的添加。

步骤三：配置安全组

安全组是腾讯云提供的实例级别防火墙，可以对云数据库进行入流量控制。您可以在购买实例时绑定安全组，也可以购买实例后在控制台绑定安全组。

注意：

目前云数据库 Tendis 安全组仅支持私有网络云数据库配置。

1. 登录 Tendis 控制台，在实例列表，单击实例 ID，进入实例管理页面。
2. 在实例管理页面，选择**安全组**页，单击**配置安全组**。
3. 在弹出的对话框，选择需要绑定的安全组，单击**确定**，即可完成安全组绑定云数据库的操作。

导入安全组规则

1. 在 安全组页，选择需要的安全组，单击安全组 ID。
2. 在入/出站规则页签上，单击**导入规则**。
3. 在弹出的对话框，选择已编辑好的入站/出站规则模板文件，单击**开始导入**。

说明：

如果需要导入规则的安全组下已存在安全组规则，建议您先导出现有规则，否则导入新规则时，将覆盖原有规则。

克隆安全组

1. 在 安全组页，在列表的**操作**列选择**更多>克隆**。
2. 在弹出的对话框，选定目标地域、目标项目后，单击**确定**。若新安全组需关联 CVM，请重新进行管理安全组内云服务器。

删除安全组

1. 在 安全组页，选择需要删除的安全组，在**操作**列选择**更多>删除**。
2. 在弹出的对话框，单击**确定**。若当前安全组有关联的 CVM 则需要先解除安全组才能进行删除。

禁用命令

最近更新时间：2023-12-21 20:22:57

本文为您介绍如何通过 Tendis 控制台禁用部分命令。

操作场景

Tendis 部分命令的使用可能会导致服务不稳定、或者数据误删除，因此云数据库 Redis 提供了禁用部分命令的功能，支持通过配置 `disable-command-list` 参数来禁用部分命令。

操作步骤

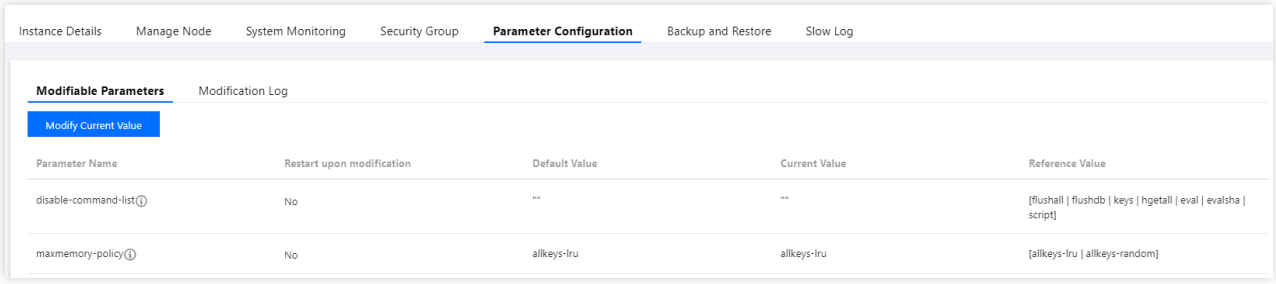
禁用命令

1. 登录 Tendis 控制台，选择对应地域，单击实例 ID，进入实例管理页面。
2. 在实例管理页面，选择**参数配置>可修改参数**页，在 `disable-command-list` 参数行，可配置禁用命令名单。

说明：

支持禁用的命令包括 flushall、flushdb、keys、hgetall、eval、evalsha、script。

配置禁用参数后会在2分钟内生效，禁用命令参数不会重启 Tendis 服务，对存量链接生效。



取消禁用命令

1. 登录 Tendis 控制台，选择对应地域，单击实例 ID，进入实例管理页面。
2. 在实例管理页面，选择**参数配置>可修改参数**页，在**当前运行参数值**禁用列表，删除相应的命令即可取消命令禁用。

参数修改历史

1. 登录 Tendis 控制台，选择对应地域，单击实例 ID，进入实例管理页面。
2. 在实例管理页面，选择**参数配置>修改历史**页，可查看参数修改历史记录。