

分布式缓存服务

用户指南

文档版本 01
发布日期 2024-12-16



版权所有 © 华为云计算技术有限公司 2024。保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

商标声明



HUAWEI和其他华为商标均为华为技术有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为云计算技术有限公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为云计算技术有限公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

华为云计算技术有限公司

地址：贵州省贵安新区黔中大道交兴功路华为云数据中心 邮编：550029

网址：<https://www.huaweicloud.com/>

目录

1 DCS 业务使用流程	1
2 创建用户并授权使用 DCS	3
3 购买 Redis 实例	7
4 连接 Redis 实例	14
4.1 配置 Redis 网络连接	14
4.1.1 连接 Redis 网络要求	14
4.1.2 开启 Redis 4.0/5.0/6.0 公网访问并获取公网访问地址	15
4.2 配置 Redis 访问控制	18
4.2.1 配置 Redis 访问白名单	18
4.2.2 配置 Redis 访问密码	19
4.2.3 配置 Redis SSL 数据加密传输	21
4.2.4 配置 Redis ACL 访问账号	22
4.3 使用客户端连接 Redis	24
4.3.1 Redis-cli 客户端连接 Redis	24
4.3.2 Jedis 客户端连接 Redis (Java)	27
4.3.3 Lettuce 客户端连接 Redis (Java)	35
4.3.4 Redisson 客户端连接 Redis (Java)	47
4.3.5 Redis-py 客户端连接 Redis (Python)	57
4.3.6 Go-redis 客户端连接 Redis (Go)	59
4.3.7 HireRedis 客户端连接 Redis (C++)	61
4.3.8 StackExchange.Redis 客户端连接 Redis (C#)	63
4.3.9 Phpredis 客户端连接 Redis (PHP)	65
4.3.10 Predis 客户端连接 Redis (PHP)	67
4.3.11 Ioredis 客户端连接 Redis (Node.js)	68
4.4 控制台连接 Redis	70
4.5 公网连接 Redis 3.0 (Redis 3.0 已停售)	71
4.5.1 开启 Redis 3.0 实例的公网访问	71
4.5.2 Redis-cli 客户端公网连接 Redis 3.0	72
5 连接 Memcached 实例 (已停售)	82
5.1 配置 Memcached 访问密码	82
5.2 使用客户端连接 Memcached	83
5.2.1 Telnet 客户端连接 Memcached	83

5.2.2 Spymemcached 客户端连接 Memcached (Java)	84
5.2.3 Python-binary-memcached 客户端连接 Memcached (Python)	87
5.2.4 Libmemcached 客户端连接 Memcached (C++)	88
5.2.5 Libmemcached 客户端连接 Memcached (PHP)	91
6 管理实例.....	96
6.1 查看和修改 DCS 实例基本信息.....	96
6.2 查看 DCS 实例后台任务.....	100
6.3 查看 DCS 实例的客户端连接信息.....	100
6.4 修改 DCS 实例配置参数.....	102
6.5 配置 DCS 实例参数模板.....	115
6.5.1 查看 DCS 实例参数模板信息.....	115
6.5.2 创建 DCS 实例自定义参数模板.....	127
6.6 配置 DCS 实例标签.....	139
6.7 重命名 DCS 实例高危命令.....	141
6.8 导出 DCS 实例列表.....	142
6.9 切换 DCS 实例的主备节点.....	142
6.10 管理 DCS 实例分片与副本.....	143
7 备份恢复实例数据.....	146
7.1 DCS 备份恢复概述.....	146
7.2 自动备份 DCS 实例数据.....	148
7.3 手动备份 DCS 实例数据.....	149
7.4 恢复 DCS 实例数据.....	150
7.5 下载 DCS 实例备份文件.....	151
8 变更实例.....	156
8.1 变更 DCS 实例规格.....	156
8.2 调整 DCS 实例带宽.....	164
8.3 变更 DCS 集群实例为多可用区.....	166
9 管理实例生命周期.....	168
9.1 重启 DCS 实例.....	168
9.2 关闭/启动 DCS 实例.....	169
9.3 删除 DCS 实例.....	169
9.4 清空 DCS 实例数据.....	171
10 分析诊断实例.....	172
10.1 查找 Redis 实例大 Key 和热 Key.....	172
10.2 扫描并删除 Redis 实例的过期 Key.....	177
10.3 诊断 Redis 实例.....	180
10.4 查看 Redis 实例的慢查询记录.....	181
10.5 查询 Redis 实例运行日志.....	182
10.6 查看 Redis 实例的命令审计日志.....	183
11 迁移实例数据.....	186

11.1 DCS 数据迁移概述.....	186
11.2 迁移方案说明.....	191
11.3 DCS 实例间迁移.....	193
11.3.1 使用迁移任务在线迁移 DCS Redis 实例.....	193
11.3.2 使用备份文件离线迁移 DCS Redis 实例.....	198
11.4 自建 Redis 迁移至 DCS.....	202
11.4.1 使用迁移任务在线迁移自建 Redis.....	202
11.4.2 使用备份文件离线迁移自建 Redis.....	206
11.4.3 使用 Redis-cli 离线迁移自建 Redis (AOF 文件)	209
11.4.4 使用 Redis-cli 离线迁移自建 Redis (RDB 文件)	211
11.4.5 使用 RedisShake 工具在线迁移自建 Redis Cluster 集群.....	212
11.4.6 使用 RedisShake 工具离线迁移自建 Redis Cluster 集群.....	215
11.5 其他云厂商 Redis 迁移至 DCS.....	217
11.5.1 使用迁移任务在线迁移其他云厂商 Redis.....	217
11.5.2 使用备份文件离线迁移其他云厂商 Redis.....	221
11.5.3 使用 Rump 在线迁移其他云厂商 Redis.....	223
11.5.4 使用 RedisShake 在线迁移其他云厂商 Redis.....	225
11.5.5 使用 RedisShake 离线迁移其他云厂商 Redis.....	230
12 测试实例性能.....	232
12.1 使用 memtier_benchmark 测试 Redis 性能.....	232
12.2 使用 redis-benchmark 测试 Redis 性能.....	234
12.3 redis-benchmark 与 memtier_benchmark 的差异.....	237
12.4 Redis 性能测试数据参考.....	237
12.4.1 Redis 3.0 主备实例测试数据.....	237
12.4.2 Redis 3.0 Proxy 集群实例测试数据.....	239
12.4.3 Redis 4.0/5.0 主备实例测试数据.....	240
12.4.4 Redis 4.0/5.0 Proxy 实例测试数据.....	241
12.4.5 Redis 4.0/5.0 Cluster 集群实例测试数据.....	243
12.4.6 Redis 6.0 主备实例测试数据.....	244
12.4.7 Redis 6.0 Cluster 集群实例测试数据.....	247
12.4.8 Redis 备份恢复迁移性能测试数据.....	249
13 申请扩大 DCS 配额.....	252
14 查看监控指标与配置告警.....	254
14.1 DCS 支持的监控指标.....	254
14.2 DCS 常用的监控指标.....	281
14.3 查看 DCS 性能监控.....	283
14.4 配置 DCS 监控告警.....	283
15 查看 DCS 审计日志.....	292

1 DCS 业务使用流程

DCS 实例管理方式

DCS提供了Web化的服务管理平台，即管理控制台，还提供了基于HTTPS请求的RESTful API（Application programming interface）管理方式。

- Web管理控制台方式

您注册后登录[管理控制台](#)，单击主页左上角服务列表，选择“应用中间件 > 分布式缓存服务 Redis版”，进入DCS的管理界面。

Web管理控制台DCS缓存实例的相关管理功能，具体操作，请参见本手册[购买Redis实例](#)到[迁移实例数据](#)。

DCS各项指标的监控数据会记录在云监控服务中，如果您需要查看相关监控数据或配置监控告警规则，请登录云监控控制台查看，具体操作，请参考[查看DCS性能监控](#)。

如果您开启了云审计服务，系统会将DCS实例的操作记录到云审计服务，您可以登录云审计服务控制台查看，具体查看操作，请参考[查看DCS审计日志](#)。

- API方式

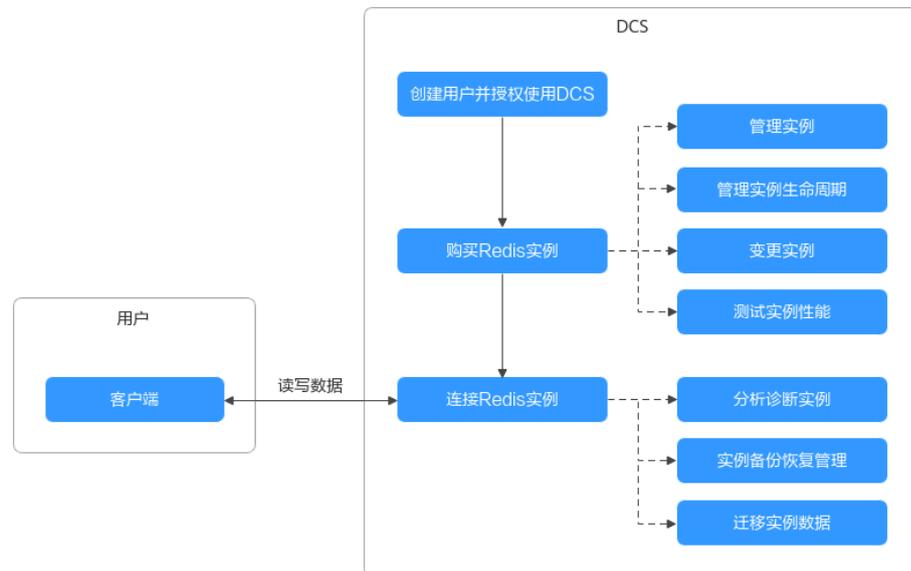
DCS提供了基于RESTful的API接口，支持您将分布式缓存服务集成到自己的应用系统，实现自动化统一管理，有关API的调用说明与具体的API接口内容，请参考《[分布式缓存服务API参考](#)》。

须知

1. 对于已经开放API的功能，用户可以选择通过Web管理控制台或调用API的操作方式，对于未开放API的功能，请通过Web管理控制台进行操作。
2. 监控与审计的API请参考[云监控服务](#)以及[云审计服务](#)的帮助手册。

DCS 使用流程

图 1-1 DCS 使用流程图



1. **创建用户并授权使用DCS。**

2. **购买Redis实例。**

3. **连接Redis实例。**

DCS实例创建后，您可以通过客户端连接Redis实例，同时，DCS也支持通过控制台连接Redis。

4. **管理DCS缓存实例及数据。**

DCS提供了**管理实例**、**管理实例生命周期**、**变更实例**、**测试实例性能**、**分析诊断实例**、**实例备份恢复管理**、**迁移实例数据**等操作指导，用户可根据业务需要管理您的缓存实例及数据。

2 创建用户并授权使用 DCS

如果您需要对您所拥有的DCS服务进行精细的权限管理，您可以使用[统一身份认证服务](#)（Identity and Access Management，简称IAM），通过IAM，您可以：

- 根据企业的业务组织，在您的华为云账号中，给企业中不同职能部门的员工创建IAM用户，让员工拥有唯一安全凭证，并使用DCS资源。
- 根据企业用户的职能，设置不同的访问权限，以达到用户之间的权限隔离。
- 将DCS资源委托给更专业、高效的其他华为云账号或者云服务，这些账号或者云服务可以根据权限进行代运维。

如果华为云账号已经能满足您的要求，不需要创建独立的IAM用户，您可以跳过本章节，不影响您使用DCS服务的其它功能。

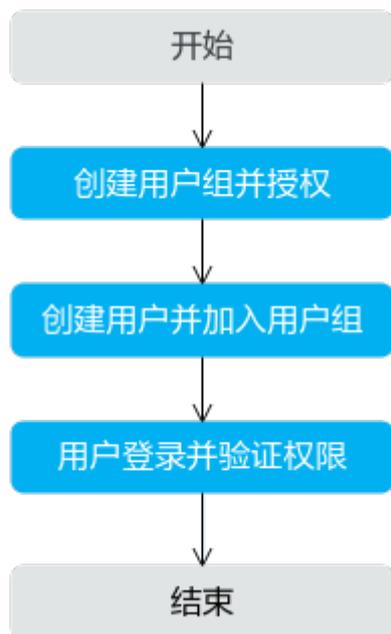
本章节以创建用户并授予“DCS ReadOnlyAccess”权限为例，为您介绍对用户授权的方法，操作流程如[图2-1](#)所示。

前提条件

给用户组授权之前，请您了解用户组可以添加的DCS系统策略，并结合实际需求进行选择，DCS支持的系统策略及策略间的对比，请参见[DCS系统策略](#)。若您需要对除DCS之外的其它服务授权，IAM支持服务的所有策略请参见[权限策略](#)。

示例流程

图 2-1 给用户授权 DCS 权限流程



1. 创建用户组并授权

在IAM控制台创建用户组，并授予分布式缓存服务的只读权限“DCS ReadOnlyAccess”。

2. 创建用户并加入用户组

在IAM控制台创建用户，并将其加入1中创建的用户组。

3. 用户登录并验证权限

新创建的用户登录控制台，验证分布式缓存服务的只读权限。

- 在“服务列表”中选择分布式缓存服务，进入DCS主界面，单击右上角“购买缓存实例”，尝试购买缓存实例，如果无法购买缓存实例（假设当前权限仅包含DCS ReadOnlyAccess），表示“DCS ReadOnlyAccess”已生效。
- 在“服务列表”中选择除分布式缓存服务外（假设当前策略仅包含DCS ReadOnlyAccess）的任一服务，若提示权限不足，表示“DCS ReadOnlyAccess”已生效。

DCS 自定义策略

如果系统预置的DCS权限，不满足您的授权要求，可以创建自定义策略。自定义策略中可以添加的授权项（Action）请参考[细粒度策略支持的授权项](#)。

目前华为云支持以下两种方式创建自定义策略：

- 可视化视图创建自定义策略：无需了解策略语法，按可视化视图导航栏选择云服务、操作、资源、条件等策略内容，可自动生成策略。
- JSON视图创建自定义策略：可以在选择策略模板后，根据具体需求编辑策略内容；也可以直接在编辑框内编写JSON格式的策略内容。

具体创建步骤请参见：[创建自定义策略](#)。本章为您介绍常用的DCS自定义策略样例。

📖 说明

由于缓存的存在，对用户、用户组以及企业项目授予OBS相关的细粒度策略后，大概需要等待5分钟细粒度策略才能生效。

DCS 自定义策略样例

- 示例1：授权用户删除缓存实例、重启实例及清空实例数据。

```
{
  "Version": "1.1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dcs:instance:delete",
        "dcs:instance:modifyStatus"
      ]
    }
  ]
}
```

- 示例2：拒绝用户删除缓存实例

拒绝策略需要同时配合其他策略使用，否则没有实际作用。用户被授予的策略中，一个授权项的作用如果同时存在Allow和Deny，则遵循Deny优先。

如果您给用户授予DCS FullAccess的系统策略，但不希望用户拥有DCS FullAccess中定义的删除缓存实例权限，您可以创建一条拒绝删除缓存实例的自定义策略，然后同时将DCS FullAccess和拒绝策略授予用户，根据Deny优先原则，则用户可以对DCS执行除了删除缓存实例外的所有操作。拒绝策略示例如下：

```
{
  "Version": "1.1",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "dcs:instance:delete"
      ]
    }
  ]
}
```

DCS 资源

资源是服务中存在的对象。在DCS中，资源包括：instance，您可以在创建自定义策略时，通过指定资源路径来选择特定资源。

表 2-1 DCS 的指定资源与对应路径

指定资源	资源名称	资源路径
instance	实例	<p>【格式】 DCS:*:*:instance:实例ID</p> <p>【说明】 对于实例资源，DCS自动生成资源路径前缀 DCS:*:*:instance: 通过实例ID指定具体的资源路径，支持通配符*。例如： DCS:*:*:instance:*表示任意DCS实例。</p>

3 购买 Redis 实例

您可以根据业务需要购买相应计算能力和存储空间的Redis实例，可同时购买多个Redis实例。

准备实例依赖资源

DCS的Redis实例部署于虚拟私有云（VPC）中，且需要绑定具体的子网，通过这样的方式为Redis提供隔离的、用户可自主配置管理的虚拟网络环境。如果是企业版Redis，还需要配置安全组，以提升实例的安全性。

因此，在创建Redis实例前您需要提前准备Redis实例的相关依赖资源：VPC、子网和安全组。如果已有VPC、子网和安全组，可重复使用，不需要重新创建。

仅创建企业版Redis实例时需要准备安全组。

表 3-1 DCS 依赖资源

准备资源	要求	创建指导
VPC和子网	<p>不同的Redis实例可以重复使用相同的VPC和子网，也可以使用不同的VPC和子网，请根据实际需要进行配置。在创建VPC和子网时应注意如下要求：</p> <ul style="list-style-type: none">• 创建的VPC与使用的DCS服务应在相同的区域。• 创建VPC和子网时，如无特殊需求，配置参数使用默认配置即可。	<p>创建VPC和子网的操作指导，请参考创建虚拟私有云和子网，若需要在已有VPC上创建和使用新的子网，请参考为虚拟私有云创建新的子网。</p>

准备资源	要求	创建指导
安全组	<p>不同的Redis实例可以重复使用相同的安全组，也可以使用不同的安全组，请根据实际需要进行选择。在创建安全组时应注意如下要求：</p> <ul style="list-style-type: none"> 创建安全组时，“模板”选择“自定义”。 安全组创建后，请保留系统默认添加的入方向“允许安全组内的弹性云服务器彼此通信”规则和出方向“放通全部流量”规则。 Redis 6.0企业版实例必须添加如表3-2所示安全组规则，其他规则请根据实际需要添加。 	<p>创建安全组的操作指导，请参考创建安全组，为安全组添加规则的操作指导，请参考添加安全组规则。</p>

表 3-2 安全组规则

方向	协议	端口	源地址
入方向	TCP	6379	0.0.0.0/0

购买 Redis 实例

步骤1 进入[购买缓存实例](#)页面。

步骤2 选择实例购买方式，支持“快速购买”和“自定义”两种购买方式。

“快速购买”的方式中提供了几种常用的“规格配置”，如[图3-1](#)，如果其中有适合您业务需求的实例类型及配置，您可以通过此方式快速选择并购买。如果您需要灵活的选择其他实例类型、实例规格等信息，请通过“自定义”的方式进行购买，如[图3-2](#)。

说明

“快速购买”和“自定义”两种购买方式只有规格配置的选择方式不同，其他配置方式相同。

图 3-1 快速购买规格配置



图 3-2 自定义购买规格配置



步骤3 选择“计费模式”。

步骤4 在“区域”下拉列表中，选择靠近您应用程序的区域，可降低网络延时、提高访问速度。

步骤5 选择“项目”，每个区域默认对应一个项目。

步骤6 规格配置。如果您选择的是“快速购买”的方式，请从常见的几种规格配置中进行选择，配置说明请参考表3-3。如果您选择的是“自定义”的方式，请参考表3-4配置实例规格。

表 3-3 规格配置说明（快速购买）

规格配置项	配置说明
产品类型-内存规格	产品类型包含“基础版”和“企业版”。 Redis企业版与基础版差异 例如：基础版-16GB，为基础版产品类型，内存规格为16GB的实例。
版本号	Redis实例的版本号。不同Redis版本差异请参考 Redis版本差异 。 实例创建后，Redis的版本不支持变更或升级。如需使用更高版本的Redis实例，需重新创建高版本Redis实例，然后将原有Redis实例的数据迁移到高版本实例上。
实例类型	快速购买的实例类型包含：主备实例、Cluster集群、或Proxy集群实例。不同实例类型的介绍请参考 DCS实例类型 。
CPU架构	快速购买方式下的CPU架构为x86类型。

规格配置项	配置说明
副本数	副本数为2，即实例的节点数为2（一个主节点，一个备节点）。
可用区	快速购买的实例主节点和备节点所在的可用区。

表 3-4 规格配置说明（自定义购买）

规格配置项	配置说明
缓存类型	DCS仅支持购买Redis缓存类型。
产品类型	产品类型包含“基础版”和“企业版”。 Redis企业版与基础版差异 仅当“版本号”选择6.0时，支持选择“基础版”和“企业版”，其他版本仅支持“基础版”。
版本类型	企业版实例的版本类型包含“高性能型”和“存储型”。 仅当“产品类型”选择“企业版”时，需要选择版本类型。
CPU架构	CPU架构包含“x86计算”和“Arm计算”两种。 推荐使用“x86计算”类型，部分Region已停售“Arm计算”类型。
版本号	当前DCS支持的Redis版本有：4.0、5.0和6.0。不同Redis版本差异请参考 Redis版本差异 。 实例创建后，Redis的版本不支持变更或升级。如需使用更高版本的Redis实例，需重新创建高版本Redis实例，然后将原有Redis实例的数据迁移到高版本实例上。
实例类型	DCS支持的实例类型有：单机、主备、Proxy集群、Cluster集群、和读写分离。不同实例类型的特点和架构，请参考 DCS实例类型 。 目前Redis 6.0企业版仅支持主备实例类型。 不同Region支持购买的Redis版本和实例类型有所不同，购买时请以控制台实际界面为准。

规格配置项	配置说明
可用区	<p>选择“可用区”。当“实例类型”为主备、读写分离、Proxy 集群、Cluster 集群时，可用区会分为“可用区”和“备可用区”，您需要分别为主、备节点设置可用区（AZ）。</p> <p>每个区域包含多个可用区，当前DCS支持将主备/读写分离/集群实例的主、备节点分别部署在不同的AZ内，使节点间电力与网络均物理隔离。您可以将应用程序也进行跨AZ部署，从而达到数据与应用全部高可用。</p> <p>说明</p> <ul style="list-style-type: none"> 当Redis主备/读写分离/集群实例进行跨可用区部署时，如果其中一个可用区故障，另一个可用区的节点不受影响。备节点会自动升级为主节点，对外提供服务，从而提供更高的容灾能力。 由于实例跨可用区部署时网络访问效率略低于部署在同一可用区内，因此Redis实例跨可用区部署时，主备节点之间同步效率会略有降低。 如果需要提高访问速度，可选择和应用同一个可用区。 在“华南-广州”区域购买DCS实例时，因为该区域可用区1-5与可用区6-7的物理距离较远，为了防止内部访问时延较大，主备可用区不支持分别选择可用区1-5及可用区6-7。例如，主可用区选择了可用区1-5中的一个时，备可用区不支持选择可用区6-7。在其他区域下购买DCS实例时无此限制。
副本数	<p>配置实例副本数。副本指缓存实例的节点。副本数为1表示实例没有备节点，副本数为2表示实例有备节点（一个主节点，一个备节点），副本数为3即实例有一个主节点，2个备节点。</p> <p>不同版本和实例类型，支持的副本数范围不同，请以控制台显示为准。单机实例不支持设置“副本数”。</p>
规格选择模式	<p>仅当实例类型为集群实例时，支持选择“规格选择模式”，只能选择一种模式，不支持同时定义分片容量和分片数。</p> <ul style="list-style-type: none"> 快速选择：无需定义单分片容量和分片数，从默认的实例规格中进行选择。 自定义分片容量：先定义单分片的容量，再配置实例规格。 自定义分片数：先定义分片数，再配置实例规格。
实例规格	<p>在“实例规格”区域，选择符合您需求的内存规格。如需了解更多实例性能请参考实例规格。实例规格默认配额请以控制台显示为准。</p> <p>您如需增加配额，单击规格下方的“申请扩大配额”，即可跳转到工单管理界面提交工单，增加配额。</p>

步骤7 配置实例网络环境信息。

- 选择已经创建好的“虚拟私有云”和“子网”。
 - 如需通过弹性云服务器访问实例，请选择与弹性云服务器相同的虚拟私有云。
 - 目前DCS实例创建完成后不支持切换虚拟私有云和子网，请谨慎选择。

- 您可以通过[共享VPC](#)功能，使用其他账号共享的VPC和子网，以实现网络资源的共享和统一管理，提升资源管控效率、降低运维成本。
- 2. 在“IPv4地址”区域，设置实例IP地址（即内网IP）。
Cluster集群实例和企业版实例仅支持自动分配地址，其他实例类型都支持自动分配IP地址和手动分配IP地址，如果您需要自定义实例IP地址，请选择手动分配IP地址。
- 3. 配置实例“端口”。购买的基础版Redis实例，支持自定义端口，自定义端口范围为1~65535的任意数字；如果未自定义，则使用默认端口6379。
如果购买的Redis版本为6.0企业版，不支持自定义端口，端口为6379。
- 4. 在“安全组”下拉列表，选择已经创建好的安全组。
安全组是一组对弹性云服务器的访问规则的集合，为同一个VPC内具有相同安全保护需求并相互信任的弹性云服务器提供访问策略。
仅企业版Redis实例需要配置安全组。Redis基础版实例是基于VPC Endpoint，暂不支持安全组，建议基础版实例创建完成后[配置实例白名单](#)。
如果选择的安全组没有开放6379端口，页面增加显示“开放6379端口”的复选框并默认勾选，表示实例创建后，会默认放开该实例安全组的6379端口。如果所选的安全组没有放开6379端口，将无法连接到该实例。

步骤8 设置实例的“名称”。

创建单个实例时，名称长度为4到64位的字符串。批量创建实例时，名称长度为4到56位的字符串，实例名称格式为“自定义名称-*n*”，其中*n*从000开始，依次递增。例如，批量创建两个实例，自定义名称为dcs_demo，则两个实例的名称为dcs_demo-000和dcs_demo-001。

步骤9 选择“企业项目”。通过选择企业项目可以帮助您将相关的资源集中在一起，按企业项目的方式来管理云资源。

如果用户无法选择企业项目，建议检查用户权限，参考[创建DCS时选择不到需要的企业项目](#)处理。

步骤10 设置实例密码。

- “访问方式”：支持“密码访问”和“免密访问”，您可以设置访问实例时是否要进行密码验证。

📖 说明

- 选择免密访问方式时，存在安全风险，请谨慎使用。
- 若申请免密模式的Redis实例，申请成功后，可以通过重置密码进行密码设置，具体可参考[修改Redis实例的访问方式](#)。
- “密码”和“确认密码”：只有“访问方式”为“密码访问”时，才会显示该参数，请在输入框中配置连接Redis实例的密码。

📖 说明

- DCS服务出于安全考虑，在密码访问模式下，连接使用Redis实例时，需要先进行密码认证。
- 请妥善保存密码，并定期更新密码。系统无法获取您设置的密码内容。

步骤11 单击“高级配置”，根据需要选择是否设置以下信息。

1. 设置“参数配置”。请根据需要选择参数模板为“系统默认”或“使用自定义模板”。

当选择“使用自定义模板”时，请从选择框中选择一个您需要的自定义参数模板。单击“查看参数”可以查看或修改所选参数模板的参数配置。如果没有提前创建该实例版本和实例类型的自定义参数模板，此时选择框为空，单击“查看参数模板列表”可以进入模板创建页面进行创建，创建方式请参考[创建DCS实例自定义参数模板](#)。



2. 设置实例备份策略，根据需要选择是否开启“自动备份”。
只有当实例类型为主备、读写分离或者集群实例时显示该参数。关于实例备份的说明及备份策略的设置请参考[备份与恢复说明](#)。
3. 当前暂不支持创建实例时配置“公网访问”，请在实例创建完成后前往详情页进行配置，具体请参考[开启Redis 4.0/5.0/6.0公网访问并获取公网访问地址](#)。
4. 设置“标签”。
标签用于标识云资源，当您拥有相同类型的许多云资源时，可以使用标签按各种维度（例如用途、所有者或环境）对云资源进行分类。
如您的组织已经设定分布式缓存服务的相关标签策略，需要按照标签策略规则为缓存实例资源添加标签。如果标签不符合标签策略的规则，可能会导致缓存实例资源创建失败，请联系组织管理员了解标签策略详情。
 - 如果您已经预定义了标签，在“标签键”和“标签值”中选择已经定义的标签键值对。另外，您可以单击右侧的“查看预定义标签”，系统会跳转到标签管理服务页面，查看已经预定义的标签，或者创建新的标签。
 - 您也可以通过输入标签键和标签值，添加标签。标签的命名规格，请参考[管理标签](#)章节。
5. 重命名实例高危命令。
当前支持的高危命令有command、keys、flushdb、flushall、hgetall、scan、hscan、sscan、和zscan，Proxy集群实例还支持dbsize和dbstats命令重命名，其他命令暂时不支持重命名。
6. 设置实例维护时间窗。
设置DCS服务运维对实例进行维护的时间，在维护前，服务运维会提前和您沟通确认。
7. 设置实例的“描述”。

步骤12 设置实例购买时长。仅当Redis为包年/包月计费模式时，需要选择购买时长及是否需要自动续费。

步骤13 设置实例购买数量。

步骤14 实例信息配置完成后，单击“立即购买”，进入实例信息确认页面。

页面显示申请的分布式缓存服务的实例名称、缓存版本和实例规格等信息。

步骤15 确认实例信息无误后，提交请求。

步骤16 任务提交成功后，自动返回缓存管理页面，当新建实例的状态显示“运行中”时，实例创建成功。

----结束

4 连接 Redis 实例

4.1 配置 Redis 网络连接

4.1.1 连接 Redis 网络要求

任何兼容Redis协议的客户端都可以访问DCS的Redis实例，您可以根据自身应用特点选用任何Redis客户端，Redis支持的客户端列表请参见[Redis客户端](#)。

客户端连接Redis在不同的连接场景下，需要满足不同的连接约束：

- 使用同一VPC内客户端访问Redis实例。
安装了客户端的弹性云服务器必须与Redis实例属于同一个VPC。Redis 3.0/6.0企业版实例，弹性云服务器与Redis实例需配置为相同的安全组，或者安全组不同时配置安全组连通规则。Redis 4.0/5.0实例，如果实例配置了IP白名单，需将弹性云服务器的IP地址加入实例IP白名单，以确保弹性云服务器与Redis实例的网络是连通的。
安全组配置，请参考[如何选择和配置安全组](#)。白名单配置，请参考[管理实例白名单](#)。
- 客户端与Redis实例所在VPC为相同Region下的不同VPC。
如果客户端与Redis实例不在相同VPC中，可以通过建立VPC对等连接方式连通网络，具体请参考：[缓存实例是否支持跨VPC访问？](#)。
- 客户端与Redis实例所在VPC不在相同Region。
如果客户端服务器和Redis实例不在同一Region，支持通过云专线打通网络，请参考[云专线](#)。
在跨Region访问Redis实例时，实例域名无法跨Region解析，无法通过域名访问。可以通过在hosts中手动配置域名与IP绑定关系或使用IP进行访问。
- 公网访问
客户端公网访问Redis 4.0/5.0/6.0实例时，请参考[开启Redis 4.0/5.0/6.0公网访问并获取公网访问地址](#)开启实例公网访问。
客户端公网访问Redis 3.0实例时，Redis缓存实例需要配置正确的安全组规则。当SSL加密功能关闭时，Redis实例的安全组入方向规则，必须允许外部地址访问6379端口；当SSL加密功能开启时，则必须允许外部地址访问36379端口。具体配置请参考常见问题：[如何选择和配置安全组？](#)

4.1.2 开启 Redis 4.0/5.0/6.0 公网访问并获取公网访问地址

Redis 4.0/5.0/6.0实例支持通过绑定弹性负载均衡（ELB）实现Redis公网访问，本章节介绍开启Redis 4.0/5.0/6.0实例公网访问、获取公网访问连接地址和端口、以及Redis实例或ELB添加IP白名单的操作指导。Redis 3.0实例开启公网访问，请参考[公网连接 Redis 3.0（Redis 3.0已停售）](#)。

约束与限制

- **Redis 4.0/5.0/6.0实例开启公网访问功能暂未完全开放，必须先联系客服开通公网访问白名单后，才支持该功能。**
- 仅单机、主备、读写分离、Proxy集群实例支持开启公网访问，Cluster集群实例暂不支持。
- 客户端公网访问Redis，与处于同一VPC下的客户端访问Redis相比，网络时延会增加。
- 因公网性能问题造成的客户端访问异常不计入SLA。

前提条件

- 准备弹性负载均衡ELB，创建ELB的操作，请参考[创建独享型负载均衡器](#)。创建的ELB必须符合以下条件。如果已有符合条件的ELB，无需重复创建。
 - ELB的实例类型必须选择“独享型”，且必须开启“IP类型后端（跨VPC后端）”。
 - ELB的规格中必须包含“网络型(TCP/UDP)”。
 - ELB必须选择与Redis实例相同的VPC。
 - ELB必须绑定弹性公网IP（EIP）。
 - ELB必须有可用的端口。
 - 单个ELB挂载多个实例时，Redis的性能会受限于ELB的规格。
- 为保护Redis实例的网络安全性，Redis实例必须配置访问密码，免密访问的实例不支持开启公网访问。如需修改免密访问的实例为密码访问，请参考[重置缓存实例密码](#)。

开启公网访问并获取公网访问地址

步骤1 登录[分布式缓存服务管理控制台](#)。

步骤2 在管理控制台左上角单击 ，选择实例所在的区域。

步骤3 单击左侧菜单栏的“缓存管理”，进入缓存管理页面。

步骤4 单击需要开启公网访问的实例名称，进入该实例的基本信息页面。

步骤5 单击“公网访问”后的“开启”。

步骤6 在开启公网访问弹窗中勾选需要绑定的ELB，单击“确定”。

如果没有可选的ELB，单击页面提示的“弹性负载均衡”跳转链接，可前往ELB控制台页面进行创建。如果已经创建了ELB，未在ELB选择列表中，请参考[前提条件](#)中的说明排查ELB是否符合绑定条件。

说明

- Redis实例绑定ELB期间，请勿删除绑定的ELB和监听器，并保证ELB可用，否则会影响Redis的正常公网连接。
- 如果需要删除ELB实例，请先在Redis实例详情页面解除绑定（关闭公网连接），再在ELB控制台删除ELB实例。

图 4-1 绑定 ELB



步骤7 开启公网状态显示“成功”后，表示开启公网访问成功。

步骤8 单击左侧菜单栏的“概览”，返回实例基本信息页面查看公网访问信息。如需关闭公网访问，单击“关闭”。

- 下图中的“EIP”地址为Redis实例的公网访问地址，“监听器”后的端口号为公网访问的端口。

图 4-2 公网访问连接地址



- 主备实例开启公网访问后，会生成两个监听器。一个主节点监听器（以listener-master开头）和一个备节点监听器（以listener-slave开头），分别用于监听实例的主节点和备节点。公网连接主备实例时请使用主监听器后的端口用于连接主备实例的主节点。仅当需要配置主备实例读写分离时，需要同时使用主、备监听器端口，分别连接主、备节点。

图 4-3 主备实例公网连接地址



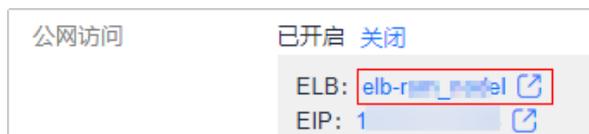
- 连接信息中的“连接地址”及“IP地址”为相同VPC内客户端访问Redis时的“域名地址:端口”和“IP地址:端口”。

----结束

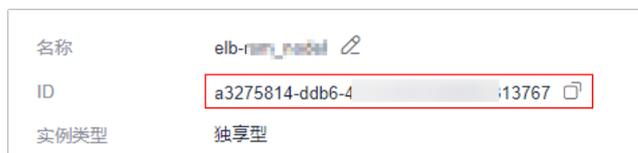
Redis 实例配置 ELB 内网 IP 白名单（可选）

如果Redis开启了IP白名单，需要将ELB内网IP地址添加到Redis实例的IP白名单中，以确保ELB可以访问Redis实例：

1. 单击“公网访问”中ELB后的链接，跳转到负载均衡器页面。



2. 复制页面中的ELB“ID”。



3. 单击页面中“IPv4私有地址”后的地址，跳转到对应子网页面。
4. 选择“IP地址管理”页签，在第二个搜索框中筛选资源ID（已复制的ELB ID），获取ELB内网IP地址。



5. 将ELB的全部内网IP地址添加到Redis的IP白名单中，添加方式请参考[配置Redis访问白名单](#)。

配置 ELB IP 地址组（可选）

如果ELB需要配置IP地址组（IP白名单），请将Redis实例的公网IP地址（EIP地址）添加到ELB的IP地址组。开启ELB IP地址组的操作请参考[访问控制IP地址组](#)。

4.2 配置 Redis 访问控制

4.2.1 配置 Redis 访问白名单

Redis 3.0/Memcached和Redis 4.0/Redis 5.0实例及Redis 6.0的部署模式不一样，DCS在控制访问缓存实例的方式也不一样，差别如下：

- Redis 3.0/Memcached/Redis 6.0企业版：通过配置安全组访问规则控制，不支持白名单功能。安全组配置操作，具体请参考[如何选择和配置安全组](#)。
- Redis 4.0/Redis 5.0/Redis 6.0基础版：不支持安全组，只支持通过白名单控制。

本章节主要介绍如何管理Redis 4.0、Redis 5.0和Redis 6.0基础版实例白名单，如果需要指定的IP地址才能访问Redis 4.0、Redis 5.0和Redis 6.0基础版实例，您需要将指定的IP地址加入到实例白名单中。白名单配置生效后，在IP白名单外的IP无法新建连接到实例，存量的连接断开前不受影响。

如果实例没有添加任何白名单或停用白名单功能，所有与实例所在VPC互通的IP地址都可以访问该实例。

创建白名单分组

- 步骤1** 登录[分布式缓存服务管理控制台](#)。
- 步骤2** 在管理控制台左上角单击 ，选择实例所在的区域。
- 步骤3** 单击左侧菜单栏的“缓存管理”，进入实例信息页面。
- 步骤4** 单击需要创建白名单的DCS缓存实例名称，进入该实例的基本信息页面。
- 步骤5** 选择“实例配置 > 白名单配置”进入白名单配置页面，单击“创建白名单分组”。
- 步骤6** 在弹出的“创建白名单分组”页面，设置“分组名”和“IP地址/地址段”。

表 4-1 创建白名单参数说明

参数名称	参数说明	示例
分组名	实例的白名单分组名称。每个实例支持创建4组白名单。 分组名的命名规范： <ul style="list-style-type: none"> • 需以字母开头。 • 长度范围在4到64位之间。 • 只能包含字母、数字、中划线、下划线。 	DCS-test

参数名称	参数说明	示例
IP地址/地址段	添加允许访问实例的IP地址/地址段。每个实例最多可以添加100个IP地址/地址段。多个IP地址/地址段之间用“,”分隔。 不支持的IP和地址段：0.0.0.0和0.0.0.0/0	10.10.10.1,10.10.10.10,192.168.0.0/16

步骤7 设置完成后，单击“确定”。

创建成功后，白名单功能立即生效，只有该分组白名单中的IP地址才允许访问实例。对于已有的长连接，需重新连接后生效。

📖 说明

- 修改白名单：在白名单列表，单击“编辑”，可以修改该白名单分组下的IP地址/地址段。
- 删除白名单：在白名单列表，单击“删除”，可以删除该白名单分组。
- 停用白名单：单击白名单列表左上角的“停用白名单”。停用白名单后，所有与实例VPC相通的IP都能访问该实例，单击“启用白名单”，可以重启白名单。

----结束

4.2.2 配置 Redis 访问密码

DCS提供了通过密码访问Redis的功能，确保缓存数据足够安全。同时也支持免密访问Redis的方式，您可以综合安全与便利的考虑，选择适合的Redis访问方式。

对于生产环境的系统，特别是含有用户等重要信息的缓存实例，建议配置实例密码访问Redis。

- 如果您需要修改缓存实例密码，可以参考[修改缓存实例密码](#)，输入旧密码并设置新密码。
- 如果您需要切换密码访问和免密访问两种Redis访问方式，或者忘记密码需要重置新密码，请参考[重置缓存实例密码](#)。

实例密码安全使用建议

1. redis-cli连接时隐藏密码。

Linux操作系统中，如果对redis-cli指定-a选项并携带密码，则在系统日志以及history记录中会保留密码信息，容易被他人获取。建议执行redis-cli命令时不指定-a选项，等连接上Redis后，输入auth命令完成鉴权。如下示例：

```
$ redis-cli -h 192.168.0.148 -p 6379
redis 192.168.0.148:6379>auth yourPassword
OK
redis 192.168.0.148:6379>
```

2. 脚本使用交互式输入密码鉴权，或使用不同权限的用户管理与执行。

脚本涉及到缓存实例连接，则采用交互式输入密码。如果需要自动化执行脚本，可使用其他用户管理脚本，以sudo方式授权执行。

3. 应用程序中使用加密模块对Redis密码加密配置。

修改缓存实例密码

📖 说明

- 免密访问模式的实例不支持修改密码操作。
- 只有处于“运行中”状态的Redis缓存实例支持修改密码。
- 更改密码后，服务端无需重启，立即生效。客户端需使用更新后的密码才能连接（长连接断开重连时需要使用新密码，断开前还可以继续使用旧密码）。

步骤1 登录[分布式缓存服务管理控制台](#)。

步骤2 在管理控制台左上角单击 ，选择实例所在的区域。

步骤3 单击左侧菜单栏的“缓存管理”。

步骤4 在需要修改密码的DCS缓存实例右侧，单击“操作”列下的“更多 > 修改密码”。

步骤5 系统弹出修改密码对话框。输入“旧密码”、“新密码”和“确认密码”。

📖 说明

修改DCS缓存实例密码时，如果重复5次输入错误的旧密码，该实例账户将被锁定5分钟，锁定期间不允许修改密码。

DCS账号密码必须满足以下复杂度要求：

- 密码不能为空。
- 新密码与旧密码不能相同。
- 密码长度在8到64位之间。
- 至少必须包含如下四种字符中的三种：
 - 小写字母
 - 大写字母
 - 数字
 - 特殊字符包括（`~!@#\$%^&*()-_+=\|{}<.>/?`）

步骤6 单击“确定”完成密码修改。

----结束

重置缓存实例密码

📖 说明

- 只有处于“运行中”状态的Redis实例支持重置缓存实例密码。
- 免密模式存在安全风险，重置为免密模式后，您也可以通过重置密码再次设置密码。
- 为保护Redis实例的网络安全性，已开启公网访问的Redis实例不支持开启免密访问。
- 只有所有节点都重置密码成功，系统才会提示重置密码成功，否则会提示重置失败。重置失败可能会造成实例重启，将缓存实例密码还原。
- 重置新的密码后，服务端无需重启，立即生效。客户端需使用新的密码才能连接（长连接断开重连时需要使用新密码，断开前还可以继续使用旧密码）。

步骤1 登录[分布式缓存服务管理控制台](#)。

步骤2 在管理控制台左上角单击 ，选择实例所在的区域。

- 步骤3** 单击左侧菜单栏的“缓存管理”。
- 步骤4** 在需要修改访问方式的Redis实例右侧，单击“操作”列下的“更多 > 重置密码”。
- 步骤5** 系统弹出“重置密码”对话框，请根据实际情况选择以下操作。
- 密码访问修改为免密访问。
打开“免密访问”开关，并单击“确定”，完成免密访问设置。
 - 免密访问修改为密码访问，或者重置新密码。
在弹出的“重置密码”对话框，输入“新密码”和“确认密码”，并单击“确定”，完成密码设置。
DCS账号密码必须满足以下复杂度要求：
 - 密码不能为空。
 - 新密码与旧密码不能相同。
 - 密码长度在8到64位之间。
 - 至少必须包含如下四种字符中的三种：
 - 小写字母
 - 大写字母
 - 数字
 - 特殊字符包括 (`~!@#\$%^&*()-_+=+\\{|},<.>/?)

----结束

4.2.3 配置 Redis SSL 数据加密传输

Redis 6.0基础版的单机、主备、Cluster集群实例支持开启SSL链路加密传输，确保数据传输过程的安全性。其他版本的实例暂不支持该功能，Redis的传输协议RESP在Redis 6.0之前的版本仅支持明文传输。

说明

因SSL加密传输的原因，SSL和客户端IP透传功能无法同时开启，加密链路下不支持携带客户端IP。

开启或关闭 SSL

- 步骤1** 登录[分布式缓存服务管理控制台](#)。
- 步骤2** 在管理控制台左上角单击 ，选择实例所在的区域。
- 步骤3** 单击左侧菜单栏的“缓存管理”。
- 步骤4** 在“缓存管理”页面，单击需要执行操作的缓存实例名称。
- 步骤5** 单击左侧菜单栏的“SSL设置”，进入SSL设置页面。
- 步骤6** 单击“SSL证书”后的 ，可以开启或关闭SSL。

须知

- 开通或关闭SSL将会重启您的实例。实例会出现秒级的连接闪断，请在业务低峰期执行该操作并确保应用具备重连机制。
- 重启操作无法撤销，单机实例及其他关闭了AOF持久化（参数配置appendonly为no）的实例，数据将清空，实例正在执行的备份任务会被停止，请谨慎操作。
- 开启SSL后，实例的读写性能会有所下降。

步骤7 开启SSL功能后，单击“下载证书”，下载SSL证书。

步骤8 解压SSL证书，将解压后的“ca.crt”文件上传到Redis客户端所在的服务器上。

步骤9 在连接实例的命令中配置“ca.crt”文件所在的路径。例如，使用redis-cli连接实例时，请参考[使用redis-cli连接Redis实例](#)。

----结束

4.2.4 配置 Redis ACL 访问账号

如需为Redis缓存实例创建多个账号，可以通过DCS的账号管理（ACL）功能，创建只读或读写账号，实现不同用户对缓存实例的只读或读写访问控制。

前提条件

目前仅Redis 4.0和Redis 5.0版本的Redis实例，默认支持账号管理功能。Redis 6.0版本的实例如需使用该功能，请联系客服开启白名单及升级实例内核版本。

企业版Redis如需使用账号管理，需要设置实例密码，免密访问的企业版Redis实例不支持该功能。

配置 Redis ACL 访问账号

步骤1 登录[分布式缓存服务管理控制台](#)。

步骤2 在管理控制台左上角单击，选择实例所在的区域。

步骤3 单击左侧菜单栏的“缓存管理”，进入实例信息页面。

步骤4 单击DCS缓存实例名称，进入该实例的基本信息页面。

步骤5 选择“账号管理”进入账号管理页面。

“账号名称”为“default”的账号为实例的“默认账号”，默认账号的权限为读写权限，该账号的密码即缓存实例的访问密码。

步骤6 单击“创建账号”，可以创建普通账号。

说明

- 每个实例支持最多18个普通账号。
- 如果Redis实例开启了免密访问，创建的普通账号不生效，仅支持默认账号。
- 如需使用普通账号请先关闭默认账号的免密访问：单击默认账号对应“操作”列的“重置密码”即可为实例设置密码。

步骤7 在弹出的创建账号窗口中，设置“账号名称”、选择账号权限为“只读”或“读写”、设置账号密码及备注。

步骤8 单击“确定”，完成账号创建。

须知

当使用创建的ACL普通账号连接实例时，需要配置实例密码为 **`{username:password}`**。

- 以**使用redis-cli连接Redis实例**为例，当使用默认账号连接实例时命令如下：
`./redis-cli -h {dcs_instance_address} -p 6379 -a {password}`
- 如果使用实例创建的ACL普通账号连接，实例密码需要配置为“账号名称:账号密码”：
`./redis-cli -h {dcs_instance_address} -p 6379 -a {username:password}`

图 4-4 账号管理



----结束

更多操作

普通账号创建后，支持以下操作：

表 4-2 账号操作

操作	操作说明
修改密码	单击普通账号对应“操作”列的“修改密码”，修改该账号的访问密码。
重置密码	如果忘记原账号密码，单击普通账号对应“操作”列的“重置密码”，可以重置该账号的访问密码。
修改权限	单击普通账号对应“操作”列的“更多>修改权限”，可修改账号为“只读”或“读写”权限。
修改备注	单击普通账号对应“操作”列的“更多>修改备注”，可修改账号的“备注”。
删除账号	单击普通账号对应“操作”列的“更多>删除”，删除该账号。
批量删除账号	勾选需要删除的普通账号，单击上方的“删除”，删除所选账号。 默认账号无法删除。

4.3 使用客户端连接 Redis

4.3.1 Redis-cli 客户端连接 Redis

本章节介绍使用redis-cli客户端连接Redis实例的方法，更多客户端的使用方法请参考[Redis客户端](#)。

以下操作以通过弹性云服务器上的客户端连接Redis实例为例进行说明。

如果是公网访问Redis 3.0实例，请参考[Redis-cli客户端公网连接Redis 3.0](#)。

前提条件

- 已成功创建Redis实例，且状态为“运行中”。创建Redis实例的操作请参考[购买Redis实例](#)。
- 已创建弹性云服务器，创建弹性云服务器的方法，请参见[购买弹性云服务器](#)。
- 如果弹性云服务器为Linux系统，该弹性云服务器必须已经安装gcc编译环境。

如果ECS未安装gcc编译环境，请执行以下命令进行安装：

```
yum install -y make
yum install -y pcre-devel
yum install -y zlib-devel
yum install -y libevent-devel
yum install -y openssl-devel
yum install -y gcc-c++
```

- 连接实例前确保客户端与Redis实例之间网络互通，具体请参考[连接Redis网络要求](#)。

Redis-cli 客户端连接 Redis (Linux 版)

步骤1 查看并获取待连接Redis实例的IP地址/域名和端口。

请参考[查看实例信息](#)，获取实例信息中的“连接地址”或“IP地址”，如[图4-5](#)。关于使用域名地址（连接地址）还是IP地址连接实例，请参考[应该选择域名还是IP地址连接Redis实例?](#)。

当客户端公网访问Redis时，请参考[开启Redis 4.0/5.0/6.0公网访问并获取公网访问地址](#)，获取实例公网访问地址及端口。

图 4-5 获取实例连接地址



📖 说明

- 本文操作步骤涉及实例端口时，统一以6379为例，连接实例时请根据实际情况替换。
- Redis 4.0及以上版本的Proxy集群实例，“连接地址”和“IP地址”为负载均衡器地址，系统将请求分发到不同的Proxy节点上。
- Redis 3.0 Proxy集群使用“后端服务地址”，可以连接到指定的Proxy节点。

步骤2 安装redis-cli客户端。

以下步骤以客户端安装在Linux系统上为例进行描述。

1. 登录弹性云服务器。
2. 执行以下命令，获取Redis客户端源码，下载路径为<https://download.redis.io/releases/redis-6.2.13.tar.gz>。

```
wget http://download.redis.io/releases/redis-6.2.13.tar.gz
```

📖 说明

此处以安装redis-6.2.13版本为例，您也可以安装其他版本。具体操作，请参见[Redis官网](#)。

3. 执行如下命令，解压Redis客户端源码包。

```
tar -xzf redis-6.2.13.tar.gz
```

4. 进入Redis目录并编译Redis客户端源码。

```
cd redis-6.2.13  
make  
cd src
```

📖 说明

如果编译的Redis源码为6.0及以上版本，且需要使用支持TLS/SSL的redis-cli，请将以上命令中的**make**替换为**make BUILD_TLS=yes**开启TLS。

步骤3 连接Redis实例。

- **连接Redis非Cluster集群实例。**

如果是单机/主备/Proxy集群/读写分离实例，请执行以下操作。

- a. 执行以下命令连接Redis实例。

```
./redis-cli -h {dcs_instance_address} -p {port}
```

其中**{dcs_instance_address}**为Redis实例的IP地址/域名，**{port}**为连接Redis实例的端口。IP地址/域名和端口获取见[步骤1](#)。

以下以使用Redis实例的域名连接地址为示例，具体连接地址和端口请根据实际情况修改：

```
[root@ecs-redis src]# ./redis-cli -h redis-069949a-dcs-lxy.dcs.huaweicloud.com -p 6379  
redis-069949a-dcs-lxy.dcs.huaweicloud.com:6379>
```

- b. 如果Redis实例设置了密码访问，则执行本步骤输入密码，校验通过后才可进行缓存数据读写，免密访问的实例无需该步骤。

```
auth {password}
```

其中**{password}**为创建Redis实例时自定义的密码，请按实际情况修改后执行。

密码访问成功的回显示例如下：

```
redis-069949a-dcs-lxy.dcs.huaweicloud.com:6379> auth *****  
OK  
redis-069949a-dcs-lxy.dcs.huaweicloud.com:6379>
```

- **连接Redis Cluster集群实例。**

如果是Cluster集群实例，请执行以下操作。

- a. 执行以下命令连接Redis实例。

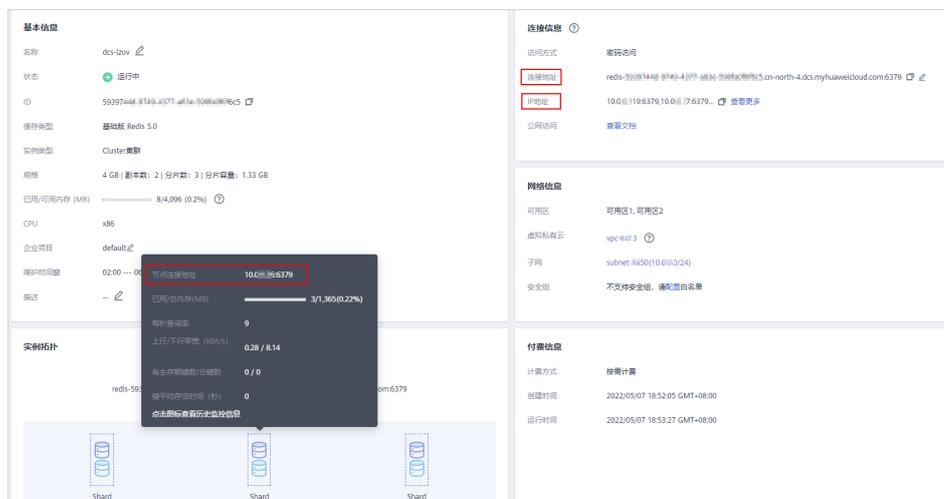
```
./redis-cli -h {dcs_instance_address} -p {port} -a {password} -c
```

其中，**{dcs_instance_address}**为Redis实例的IP地址/域名，**{port}**为Redis实例的端口，**{password}**为Cluster集群实例的密码，**-c**为连接集群节点时使用。IP地址/域名和端口获取见**步骤1**。

说明

- 在使用redis-cli连接Cluster集群时，请注意连接命令是否已加上-c。在连接Cluster集群节点时务必正确使用连接命令，否则会出现连接失败的问题。
- 客户端通过内网连接Redis时，**{dcs_instance_address}**参数的值可以为Redis实例“连接地址”、“IP地址”、及“实例拓扑”图中分片上的“节点连接地址”。获取方法，在控制台单击实例进入实例详情页面即可查看，如图4-6所示。
- 如果实例已开启免密访问，则无需输入实例的访问密码 **-a {password}**，如果忘记密码或需要重置密码请参考**重置缓存实例密码**。
- “IP地址”字段中提供了多个IP，任意一个IP地址均可用于连接实例，在连接时，任选其中一个IP连接，都表示可以连接上集群实例，在进行数据读写时，key存储在哪个slot中，由 $Crc16(key) \bmod 16384$ 的值决定，推荐配置全部IP地址，可靠性更强。
- 实例拓扑图中分片上的“节点连接地址”，可以直接连接到指定的分片。

图 4-6 获取 Cluster 集群实例连接地址



- 以下以使用Redis实例的IP连接为示例，具体请根据实际情况修改：

```
root@ecs-redis:~/redis-6.2.13/src# ./redis-cli -h 192.168.0.85 -p 6379 -a ***** -c
192.168.0.85:6379>
```

- 以下以使用Redis实例的域名连接为示例，具体请根据实际情况修改：

```
root@ecs-redis:~/redis-6.2.13/src# ./redis-cli -h redis-51e463c-dcs-lxy.dcs.huaweicloud.com -p 6379 -a ***** -c
redis-51e463c-dcs-lxy.dcs.huaweicloud.com:6379>
```

- b. 执行**cluster nodes**查看Cluster集群节点信息。

Cluster集群每一个分片都是一主一从的双副本结构，执行该命令可以查看该实例的所有节点信息，如下所示。

```
192.168.0.85:6379> cluster nodes
0988ae8fd3686074c9afdccce73d7878c81a33ddc 192.168.0.231:6379@16379 slave
```

```
f0141816260ca5029c56333095f015c7a058f113 0 1568084030
000 3 connected
1a32d809c0b743bd83b5e1c277d5d201d0140b75 192.168.0.85:6379@16379 myself,master - 0
1568084030000 2 connected 5461-10922
c8ad7af9a12cce3c8e416fb67bd6ec9207f0082d 192.168.0.130:6379@16379 slave
1a32d809c0b743bd83b5e1c277d5d201d0140b75 0 1568084031
000 2 connected
7ca218299c254b5da939f8e60a940ac8171adc27 192.168.0.22:6379@16379 master - 0
1568084030000 1 connected 0-5460
f0141816260ca5029c56333095f015c7a058f113 192.168.0.170:6379@16379 master - 0
1568084031992 3 connected 10923-16383
19b1a400815396c6223963b013ec934a657bdc52 192.168.0.161:6379@16379 slave
7ca218299c254b5da939f8e60a940ac8171adc27 0 1568084031
000 1 connected
```

各节点只能进行只读操作，不能进行写操作。在进行数据写入时，key存储在哪个slot中，由 $\text{CRC16}(\text{key}) \bmod 16384$ 的值决定。

如下所示，数据写入时，根据 $\text{CRC16}(\text{key}) \bmod 16384$ 的值决定key存储位置，并跳转到该slot所在的节点上。

```
192.168.0.170:6379> set hello world
-> Redirected to slot [866] located at 192.168.0.22:6379
OK
192.168.0.22:6379> set happy day
OK
192.168.0.22:6379> set abc 123
-> Redirected to slot [7638] located at 192.168.0.85:6379
OK
192.168.0.85:6379> get hello
-> Redirected to slot [866] located at 192.168.0.22:6379
"world"
192.168.0.22:6379> get abc
-> Redirected to slot [7638] located at 192.168.0.85:6379
"123"
192.168.0.85:6379>
```

📖 说明

当Redis 6.0基础版实例开启了SSL时，需要配置SSL证书路径：

- Cluster集群连接命令：
`./redis-cli -h {dcs_instance_address} -p {port} -a {password} -c --tls --cacert {certification file path}`
- 单机、主备连接命令：
`./redis-cli -h {dcs_instance_address} -p {port} -a {password} --tls --cacert {certification file path}`

（通过SSL加密连接Redis时，请使用6.x及以上版本的redis-cli客户端）

----结束

Redis-cli 客户端连接 Redis (Windows 版)

Windows版本的Redis客户端安装包，请单击[这里](#)下载编译包。下载后直接解压安装到自定义目录，然后使用cmd工具进入该目录，执行以下命令连接redis实例：

```
redis-cli.exe -h XXX -p 6379
```

其中：“XXX”为Redis实例的IP地址/域名，“6379”为Redis实例的端口。IP地址/域名和端口获取见[查看实例信息](#)，请按实际情况修改后执行。

4.3.2 Jedis 客户端连接 Redis (Java)

本章节介绍使用Jedis客户端连接Redis实例的方法。更多的客户端的使用方法请参考[Redis客户端](#)。

在springboot类型的项目中，spring-data-redis中已提供了对jedis、lettuce的集成适配。另外，在springboot1.x中默认集成的是jedis，springboot2.x中改为了lettuce，因此在springboot2.x及更高版本中集成使用jedis，需要对已集成的lettuce组件依赖进行排包。

📖 说明

Springboot版本不得低于2.3.12.RELEASE，Jedis版本不得低于3.10.0。

前提条件

- 已成功创建Redis实例，且状态为“运行中”。创建Redis实例的操作请参考[购买Redis实例](#)。
- 查看并获取待连接Redis实例的IP地址/域名和端口。具体步骤请参见[查看和修改DCS实例基本信息](#)。
- 连接实例前确保客户端与Redis实例之间网络互通，具体请参考[连接Redis网络要求](#)。

Pom 配置

```
<!-- 引入spring-data-redis组件 -->
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-data-redis</artifactId>
  <!--spring boot 2.0之后默认lettuce客户端, 使用jedis时需要排包-->
  <exclusions>
    <exclusion>
      <groupId>io.lettuce</groupId>
      <artifactId>lettuce-core</artifactId>
    </exclusion>
  </exclusions>
</dependency>
<!-- 引入jedis依赖包 -->
<dependency>
  <groupId>redis.clients</groupId>
  <artifactId>jedis</artifactId>
  <version>${jedis.version}</version>
</dependency>
```

基于 application.properties 配置

- 单机、主备、读写分离、Proxy集群实例配置

```
#redis host
spring.redis.host=<host>
#redis 端口号
spring.redis.port=<port>
#redis 数据库下标
spring.redis.database=0
#redis 密码
spring.redis.password=<password>
#redis 读写超时
spring.redis.timeout=2000
#是否开启连接池
spring.redis.jedis.pool.enabled=true
#连接池的最小连接数
spring.redis.jedis.pool.min-idle=50
#连接池的最大空闲连接数
spring.redis.jedis.pool.max-idle=200
#连接池的最大连接数
spring.redis.jedis.pool.max-active=200
#连接池耗尽后获取连接的最大等待时间，默认-1表示一直等待
spring.redis.jedis.pool.max-wait=3000
```

```
#空闲连接逐出的检测周期，默认为60S
spring.redis.jedis.pool.time-between-eviction-runs=60S
```

- **Cluster集群实例配置**

```
#redis cluster节点连接信息
spring.redis.cluster.nodes=<ip:port>,<ip:port>,<ip:port>
#redis cluster密码
spring.redis.password=<password>
#redis cluster访问最大重定向次数
spring.redis.cluster.max-redirects=3
#redis 读写超时
spring.redis.timeout=2000
#是否开启连接池
spring.redis.jedis.pool.enabled=true
#连接池的最小连接数
spring.redis.jedis.pool.min-idle=50
#连接池的最大空闲连接数
spring.redis.jedis.pool.max-idle=200
#连接池的最大连接数
spring.redis.jedis.pool.max-active=200
#连接池耗尽后获取连接的最大等待时间，默认-1表示一直等待
spring.redis.jedis.pool.max-wait=3000
#空闲连接逐出的检测周期，默认为60S
spring.redis.jedis.pool.time-between-eviction-runs=60S
```

基于 Bean 方式配置

- **单机、主备、读写分离、Proxy集群实例配置**

```
import java.time.Duration;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.data.redis.connection.RedisConnectionFactory;
import org.springframework.data.redis.connection.RedisStandaloneConfiguration;
import org.springframework.data.redis.connection.jedis.JedisClientConfiguration;
import org.springframework.data.redis.connection.jedis.JedisConnectionFactory;

import redis.clients.jedis.JedisPoolConfig;

@Configuration
public class RedisConfiguration {

    @Value("${redis.host}")
    private String redisHost;

    @Value("${redis.port:6379}")
    private Integer redisPort = 6379;

    @Value("${redis.database:0}")
    private Integer redisDatabase = 0;

    @Value("${redis.password}")
    private String redisPassword;

    @Value("${redis.connect.timeout:3000}")
    private Integer redisConnectTimeout = 3000;

    @Value("${redis.read.timeout:2000}")
    private Integer redisReadTimeout = 2000;

    @Value("${redis.pool.minSize:50}")
    private Integer redisPoolMinSize = 50;

    @Value("${redis.pool.maxSize:200}")
    private Integer redisPoolMaxSize = 200;

    @Value("${redis.pool.maxWaitMillis:3000}")
    private Integer redisPoolMaxWaitMillis = 3000;
```

```

@Value("${redis.pool.softMinEvictableIdleTimeMillis:1800000}")
private Integer redisPoolSoftMinEvictableIdleTimeMillis = 30 * 60 * 1000;

@Value("${redis.pool.timeBetweenEvictionRunsMillis:60000}")
private Integer redisPoolBetweenEvictionRunsMillis = 60 * 1000;

@Bean
public RedisConnectionFactory redisConnectionFactory(JedisClientConfiguration
clientConfiguration) {

    RedisStandaloneConfiguration standaloneConfiguration = new RedisStandaloneConfiguration();
    standaloneConfiguration.setHostName(redisHost);
    standaloneConfiguration.setPort(redisPort);
    standaloneConfiguration.setDatabase(redisDatabase);
    standaloneConfiguration.setPassword(redisPassword);

    return new JedisConnectionFactory(standaloneConfiguration, clientConfiguration);
}

@Bean
public JedisClientConfiguration clientConfiguration() {

    JedisClientConfiguration clientConfiguration = JedisClientConfiguration.builder()
        .connectTimeout(Duration.ofMillis(redisConnectTimeout))
        .readTimeout(Duration.ofMillis(redisReadTimeout))
        .usePooling().poolConfig(redisPoolConfig())
        .build();

    return clientConfiguration;
}

private JedisPoolConfig redisPoolConfig() {

    JedisPoolConfig poolConfig = new JedisPoolConfig();
    //连接池的最小连接数
    poolConfig.setMinIdle(redisPoolMinSize);
    //连接池的最大空闲连接数
    poolConfig.setMaxIdle(redisPoolMaxSize);
    //连接池的最大连接数
    poolConfig.setMaxTotal(redisPoolMaxSize);
    //连接池耗尽后是否需要等待，默认true表示等待。当值为true时，setMaxWait才会生效
    poolConfig.setBlockWhenExhausted(true);
    //连接池耗尽后获取连接的最大等待时间，默认-1表示一直等待
    poolConfig.setMaxWaitMillis(redisPoolMaxWaitMillis);
    //创建连接时校验有效性(ping)，默认false
    poolConfig.setTestOnCreate(false);
    //获取连接时校验有效性(ping)，默认false，业务量大时建议设置为false减少开销
    poolConfig.setTestOnBorrow(true);
    //归还连接时校验有效性(ping)，默认false，业务量大时建议设置为false减少开销
    poolConfig.setTestOnReturn(false);
    //是否开启空闲连接检测，如为false，则不剔除空闲连接
    poolConfig.setTestWhileIdle(true);
    //连接空闲多久后逐出，空闲时间>该值，并且空闲连接>最大空闲数时直接逐出
    poolConfig.setSoftMinEvictableIdleTimeMillis(redisPoolSoftMinEvictableIdleTimeMillis);
    //关闭根据MinEvictableIdleTimeMillis判断逐出
    poolConfig.setMinEvictableIdleTimeMillis(-1);
    //空闲连接逐出的检测周期，默认为60S
    poolConfig.setTimeBetweenEvictionRunsMillis(redisPoolBetweenEvictionRunsMillis);
    return poolConfig;
}
}

```

- **Cluster集群实例配置**

```

import java.time.Duration;
import java.util.ArrayList;
import java.util.List;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;

```

```
import org.springframework.context.annotation.Configuration;
import org.springframework.data.redis.connection.RedisClusterConfiguration;
import org.springframework.data.redis.connection.RedisConnectionFactory;
import org.springframework.data.redis.connection.RedisNode;
import org.springframework.data.redis.connection.jedis.JedisClientConfiguration;
import org.springframework.data.redis.connection.jedis.JedisConnectionFactory;

import redis.clients.jedis.JedisPoolConfig;

@Configuration
public class RedisConfiguration {

    @Value("${redis.cluster.nodes}")
    private String redisClusterNodes;

    @Value("${redis.password}")
    private String redisPassword;

    @Value("${redis.connect.timeout:3000}")
    private Integer redisConnectTimeout = 3000;

    @Value("${redis.read.timeout:2000}")
    private Integer redisReadTimeout = 2000;

    @Value("${redis.pool.minSize:50}")
    private Integer redisPoolMinSize = 50;

    @Value("${redis.pool.maxSize:200}")
    private Integer redisPoolMaxSize = 200;

    @Value("${redis.pool.maxWaitMillis:3000}")
    private Integer redisPoolMaxWaitMillis = 3000;

    @Value("${redis.pool.softMinEvictableIdleTimeMillis:1800000}")
    private Integer redisPoolSoftMinEvictableIdleTimeMillis = 30 * 60 * 1000;

    @Value("${redis.pool.timeBetweenEvictionRunsMillis:60000}")
    private Integer redisPoolBetweenEvictionRunsMillis = 60 * 1000;

    @Bean
    public RedisConnectionFactory redisConnectionFactory(JedisClientConfiguration
clientConfiguration) {

        RedisClusterConfiguration clusterConfiguration = new RedisClusterConfiguration();

        List<RedisNode> clusterNodes = new ArrayList<>();
        for (String clusterNodeStr : redisClusterNodes.split(",")) {
            String[] nodeInfo = clusterNodeStr.split(":");
            clusterNodes.add(new RedisNode(nodeInfo[0], Integer.valueOf(nodeInfo[1])));
        }
        clusterConfiguration.setClusterNodes(clusterNodes);

        clusterConfiguration.setPassword(redisPassword);
        clusterConfiguration.setMaxRedirects(3);

        return new JedisConnectionFactory(clusterConfiguration, clientConfiguration);
    }

    @Bean
    public JedisClientConfiguration clientConfiguration() {

        JedisClientConfiguration clientConfiguration = JedisClientConfiguration.builder()
            .connectTimeout(Duration.ofMillis(redisConnectTimeout))
            .readTimeout(Duration.ofMillis(redisReadTimeout))
            .usePooling().poolConfig(redisPoolConfig())
            .build();

        return clientConfiguration;
    }
}
```

```
private JedisPoolConfig redisPoolConfig() {

    JedisPoolConfig poolConfig = new JedisPoolConfig();
    //连接池的最小连接数
    poolConfig.setMinIdle(redisPoolMinSize);
    //连接池的最大空闲连接数
    poolConfig.setMaxIdle(redisPoolMaxSize);
    //连接池的最大连接数
    poolConfig.setMaxTotal(redisPoolMaxSize);
    //连接池耗尽后是否需要等待，默认true表示等待。当值为true时，setMaxWait才会生效
    poolConfig.setBlockWhenExhausted(true);
    //连接池耗尽后最大等待时间，默认-1表示一直等待
    poolConfig.setMaxWaitMillis(redisPoolMaxWaitMillis);
    //创建连接时校验有效性(ping)，默认false
    poolConfig.setTestOnCreate(false);
    //获取连接时校验有效性(ping)，默认false，业务量大时建议设置为false减少开销
    poolConfig.setTestOnBorrow(true);
    //归还连接时校验有效性(ping)，默认false，业务量大时建议设置为false减少开销
    poolConfig.setTestOnReturn(false);
    //是否开启空闲连接检测，如为false，则不剔除空闲连接
    poolConfig.setTestWhileIdle(true);
    //连接空闲多久后逐出，当空闲时间>该值，并且空闲连接>最大空闲数时直接逐出
    poolConfig.setSoftMinEvictableIdleTimeMillis(redisPoolSoftMinEvictableIdleTimeMillis);
    //关闭根据MinEvictableIdleTimeMillis判断逐出
    poolConfig.setMinEvictableIdleTimeMillis(-1);
    //空闲连接逐出的检测周期，默认为60s
    poolConfig.setTimeBetweenEvictionRunsMillis(redisPoolBetweenEvictionRunsMillis);
    return poolConfig;
}
```

SSL 连接配置（可选配置）

当实例开启了SSL，通过SSL连接实例时，请使用以下内容替换[基于Bean方式配置](#)中的JedisClientConfiguration构造方法clientConfiguration()。Redis实例支持SSL的情况请参考[配置Redis SSL数据加密传输](#)。

```
@Bean
public JedisClientConfiguration clientConfiguration() throws Exception {
    JedisClientConfiguration.JedisClientConfigurationBuilder configurationBuilder
        = JedisClientConfiguration.builder()
        .connectTimeout(Duration.ofMillis(redisConnectTimeout))
        .readTimeout(Duration.ofMillis(redisReadTimeout));

    configurationBuilder.usePooling().poolConfig(redisPoolConfig());
    configurationBuilder.useSsl().sslSocketFactory(getTrustStoreSslSocketFactory());
    return configurationBuilder.build();
}

private SSLSocketFactory getTrustStoreSslSocketFactory() throws Exception{
    //加载自定义路径下的ca证书,可结合具体业务配置
    CertificateFactory cf = CertificateFactory.getInstance("X.509");
    Certificate ca;
    try (InputStream is = new FileInputStream("./ca.crt")) {
        ca = cf.generateCertificate(is);
    }

    //创建keystore
    String keyStoreType = KeyStore.getDefaultType();
    KeyStore keyStore = KeyStore.getInstance(keyStoreType);
    keyStore.load(null, null);
    keyStore.setCertificateEntry("ca", ca);

    //创建TrustManager
    TrustManagerFactory trustManagerFactory = TrustManagerFactory.getInstance(
        TrustManagerFactory.getDefaultAlgorithm());
    trustManagerFactory.init(keyStore);
}
```

```
//创建SSLContext
SSLContext context = SSLContext.getInstance("TLS");
context.init(null, trustManagerFactory.getTrustManagers(), new SecureRandom());
return context.getSocketFactory();
}
```

参数明细

表 4-3 RedisStandaloneConfiguration 参数

参数	默认值	说明
hostName	localhost	连接Redis实例的IP地址/域名。
port	6379	连接端口号。
database	0	数据库下标，默认0。
password	-	连接Redis实例的密码。

表 4-4 RedisClusterConfiguration 参数

参数	说明
clusterNodes	cluster节点连接信息，需节点IP、Port。
maxRedirects	cluster访问最大重定向次数。
password	连接密码。

表 4-5 JedisPoolConfig 参数

参数	默认值	说明
minIdle	-	连接池的最小连接数。
maxIdle	-	连接池的最大空闲连接数。
maxTotal	-	连接池的最大连接数。
blockWhenExhausted	true	连接池耗尽后是否需要等待，默认true表示等待，false表示不等待。当值为true时，设置maxWaitMillis才会生效。
maxWaitMillis	-1	连接池耗尽后获取连接的最大等待时间，单位：毫秒。默认-1表示一直等待。
testOnCreate	false	创建连接时校验有效性(ping)，false：不校验，true：校验。
testOnBorrow	false	获取连接时校验有效性(ping)，false：不校验，true：校验。业务量大时建议设置为false减少开销。

参数	默认值	说明
testOnReturn	false	归还连接时校验有效性(ping), false: 不校验, true: 校验。业务量大时建议设置为false减少开销。
testWhileIdle	false	是否开启空闲连接检测, 如为false, 则不剔除空闲连接, 建议值: true 。
softMinEvictableIdleTimeMillis	1800000	连接空闲多久后逐出, (空闲时间>该值 && 空闲连接>最大空闲数)时直接逐出, 单位: 毫秒。
minEvictableIdleTimeMillis	60000	根据minEvictableIdleTimeMillis时间判断逐出, 单位: 毫秒。 建议值: -1 , 表示关闭该策略, 改用softMinEvictableIdleTimeMillis策略。
timeBetweenEvictionRunsMillis	60000	空闲连接逐出的检测周期, 单位: 毫秒。

表 4-6 JedisClientConfiguration 参数

参数	默认值	说明
connectTimeout	2000	连接超时时间, 单位: 毫秒。
readTimeout	2000	请求等待响应的超时时间, 单位: 毫秒。
poolConfig	-	池化配置, 具体请参见 JedisPoolConfig 。

DCS 实例配置建议

- 连接池配置

📖 说明

以下计算方式只适用于一般业务场景, 建议根据业务情况做适当调整适配。

连接池的大小没有绝对的标准, 建议根据业务流量进行合理配置, 一般连接池大小的参数计算公式如下:

- 最小连接数 = (单机访问Redis QPS) / (1000ms / 单命令平均耗时)
- 最大连接数 = (单机访问Redis QPS) / (1000ms / 单命令平均耗时) * 150%

举例: 某个业务应用的QPS为10000左右, 每个请求需访问Redis10次, 即每秒对Redis的访问次数为100000次, 同时该业务应用有10台机器, 计算如下:

单机访问Redis QPS = 100000 / 10 = 10000

单命令平均耗时 = 20ms (Redis处理单命令耗时为5~10ms, 遇到网络抖动按照15~20ms来估算)

最小连接数 = (10000) / (1000ms / 20ms) = 200

最大连接数 = (10000) / (1000ms / 20ms) * 150% = 300

4.3.3 Lettuce 客户端连接 Redis (Java)

本章节介绍使用Lettuce客户端连接Redis实例的方法。更多的客户端的使用方法请参考[Redis客户端](#)。

在springboot类型的项目中，spring-data-redis中已提供了对[jedis](#)、[lettuce](#)的集成适配。另外，在springboot1.x中默认集成的是jedis，springboot2.x中改为了lettuce，因此在springboot2.x及更高版本中想集成使用lettuce，无需手动引入lettuce依赖包。

📖 说明

Springboot版本不得低于2.3.12.RELEASE，Lettuce版本不得低于[6.3.0.RELEASE](#)。

前提条件

- 已成功创建Redis实例，且状态为“运行中”。创建Redis实例的操作请参考[购买Redis实例](#)。
- 查看并获取待连接Redis实例的IP地址/域名和端口。具体步骤请参见[查看和修改DCS实例基本信息](#)。
- 连接实例前确保客户端与Redis实例之间网络互通，具体请参考[连接Redis网络要求](#)。

Pom 配置

```
<!-- 引入spring-data-redis组件，默认已集成lettuce依赖SDK -->
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-data-redis</artifactId>
</dependency>
<dependency>
  <groupId>io.lettuce</groupId>
  <artifactId>lettuce-core</artifactId>
  <version>${lettuce.version}</version>
</dependency>
```

基于 application.properties 配置

- 单机、主备、读写分离、Proxy集群实例配置

```
#redis host
spring.redis.host=<host>
#redis 端口号
spring.redis.port=<port>
#redis 数据库下标
spring.redis.database=0
#redis 密码
spring.redis.password=<password>
#redis 读写超时
spring.redis.timeout=2000
```

- Cluster集群实例配置

```
# redis cluster节点信息
spring.redis.cluster.nodes=<ip:port>,<ip:port>,<ip:port>
# redis cluster 最大重定向次数
spring.redis.cluster.max-redirects=3
# redis cluster 节点密码
spring.redis.password=<password>
# redis cluster 超时配置
spring.redis.timeout=2000
# 开启自适应拓扑刷新
spring.redis.lettuce.cluster.refresh.adaptive=true
# 开启每10S定时刷新拓扑结构
spring.redis.lettuce.cluster.refresh.period=10S
```

基于 Bean 方式配置

- 单机、主备、读写分离、Proxy 集群实例配置

```
import java.time.Duration;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.data.redis.connection.RedisConnectionFactory;
import org.springframework.data.redis.connection.RedisStandaloneConfiguration;
import org.springframework.data.redis.connection.lettuce.LettuceClientConfiguration;
import org.springframework.data.redis.connection.lettuce.LettuceConnectionFactory;

import io.lettuce.core.ClientOptions;
import io.lettuce.core.SocketOptions;

/**
 * Lettuce 非池化配置，与 application.properties 配置方式二选一
 */
@Configuration
public class RedisConfiguration {

    @Value("${redis.host}")
    private String redisHost;

    @Value("${redis.port:6379}")
    private Integer redisPort = 6379;

    @Value("${redis.database:0}")
    private Integer redisDatabase = 0;

    @Value("${redis.password:}")
    private String redisPassword;

    @Value("${redis.connect.timeout:2000}")
    private Integer redisConnectTimeout = 2000;

    @Value("${redis.read.timeout:2000}")
    private Integer redisReadTimeout = 2000;

    @Bean
    public RedisConnectionFactory redisConnectionFactory(LettuceClientConfiguration
clientConfiguration) {

        RedisStandaloneConfiguration standaloneConfiguration = new RedisStandaloneConfiguration();
        standaloneConfiguration.setHostName(redisHost);
        standaloneConfiguration.setPort(redisPort);
        standaloneConfiguration.setDatabase(redisDatabase);
        standaloneConfiguration.setPassword(redisPassword);

        LettuceConnectionFactory connectionFactory = new
LettuceConnectionFactory(standaloneConfiguration, clientConfiguration);
        connectionFactory.setDatabase(redisDatabase);
        return connectionFactory;
    }

    @Bean
    public LettuceClientConfiguration clientConfiguration() {

        SocketOptions socketOptions =
SocketOptions.builder().connectTimeout(Duration.ofMillis(redisConnectTimeout)).build();

        ClientOptions clientOptions = ClientOptions.builder()
            .autoReconnect(true)
            .pingBeforeActivateConnection(true)
            .cancelCommandsOnReconnectFailure(false)
            .disconnectedBehavior(ClientOptions.DisconnectedBehavior.ACCEPT_COMMANDS)
            .socketOptions(socketOptions)
            .build();
    }
}
```

```

        LettuceClientConfiguration clientConfiguration = LettuceClientConfiguration.builder()
            .commandTimeout(Duration.ofMillis(redisReadTimeout))
            .readFrom(ReadFrom.MASTER)
            .clientOptions(clientOptions)
            .build();

        return clientConfiguration;
    }
}

```

- 单机、主备、读写分离、Proxy集群实例池化配置

引入池化组件

```

<dependency>
  <groupId>org.apache.commons</groupId>
  <artifactId>commons-pool2</artifactId>
  <version>2.11.1</version>
</dependency>

```

代码配置

```

import java.time.Duration;

import org.apache.commons.pool2.impl.GenericObjectPoolConfig;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.data.redis.connection.RedisConnectionFactory;
import org.springframework.data.redis.connection.RedisStandaloneConfiguration;
import org.springframework.data.redis.connection.lettuce.LettuceClientConfiguration;
import org.springframework.data.redis.connection.lettuce.LettuceConnectionFactory;
import org.springframework.data.redis.connection.lettuce.LettucePoolingClientConfiguration;

import io.lettuce.core.ClientOptions;
import io.lettuce.core.SocketOptions;

/**
 * Lettuce 池化配置
 */
@Configuration
public class RedisPoolConfiguration {
    @Value("${redis.host}")
    private String redisHost;

    @Value("${redis.port:6379}")
    private Integer redisPort = 6379;

    @Value("${redis.database:0}")
    private Integer redisDatabase = 0;

    @Value("${redis.password}")
    private String redisPassword;

    @Value("${redis.connect.timeout:2000}")
    private Integer redisConnectTimeout = 2000;

    @Value("${redis.read.timeout:2000}")
    private Integer redisReadTimeout = 2000;

    @Value("${redis.pool.minSize:50}")
    private Integer redisPoolMinSize = 50;

    @Value("${redis.pool.maxSize:200}")
    private Integer redisPoolMaxSize = 200;

    @Value("${redis.pool.maxWaitMillis:2000}")
    private Integer redisPoolMaxWaitMillis = 2000;

    @Value("${redis.pool.softMinEvictableIdleTimeMillis:1800000}")

```

```
private Integer redisPoolSoftMinEvictableIdleTimeMillis = 30 * 60 * 1000;

@Value("${redis.pool.timeBetweenEvictionRunsMillis:60000}")
private Integer redisPoolBetweenEvictionRunsMillis = 60 * 1000;

@Bean
public RedisConnectionFactory redisConnectionFactory(LettuceClientConfiguration
clientConfiguration) {

    RedisStandaloneConfiguration standaloneConfiguration = new RedisStandaloneConfiguration();
    standaloneConfiguration.setHostName(redisHost);
    standaloneConfiguration.setPort(redisPort);
    standaloneConfiguration.setDatabase(redisDatabase);
    standaloneConfiguration.setPassword(redisPassword);

    LettuceConnectionFactory connectionFactory = new
LettuceConnectionFactory(standaloneConfiguration, clientConfiguration);
    connectionFactory.setDatabase(redisDatabase);
    //关闭共享链接, 才能池化生效
    connectionFactory.setShareNativeConnection(false);
    return connectionFactory;
}

@Bean
public LettuceClientConfiguration clientConfiguration() {

    SocketOptions socketOptions =
SocketOptions.builder().connectTimeout(Duration.ofMillis(redisConnectTimeout)).build();

    ClientOptions clientOptions = ClientOptions.builder()
        .autoReconnect(true)
        .pingBeforeActivateConnection(true)
        .cancelCommandsOnReconnectFailure(false)
        .disconnectedBehavior(ClientOptions.DisconnectedBehavior.ACCEPT_COMMANDS)
        .socketOptions(socketOptions)
        .build();

    LettucePoolingClientConfiguration clientConfiguration =
LettucePoolingClientConfiguration.builder()
        .poolConfig(poolConfig())
        .commandTimeout(Duration.ofMillis(redisReadTimeout))
        .clientOptions(clientOptions)
        .readFrom(ReadFrom.MASTER)
        .build();
    return poolingClientConfiguration;
}

private GenericObjectPoolConfig redisPoolConfig() {
    GenericObjectPoolConfig poolConfig = new GenericObjectPoolConfig();
    //连接池的最小连接数
    poolConfig.setMinIdle(redisPoolMinSize);
    //连接池的最大空闲连接数
    poolConfig.setMaxIdle(redisPoolMaxSize);
    //连接池的最大连接数
    poolConfig.setMaxTotal(redisPoolMaxSize);
    //连接池耗尽后是否需要等待, 默认true表示等待。当值为true时, setMaxWait才会生效
    poolConfig.setBlockWhenExhausted(true);
    //连接池耗尽后获取连接的最大等待时间, 默认-1表示一直等待
    poolConfig.setMaxWait(Duration.ofMillis(redisPoolMaxWaitMillis));
    //创建连接时校验有效性(ping), 默认false
    poolConfig.setTestOnCreate(false);
    //获取连接时校验有效性(ping), 默认false, 业务量大时建议设置为false减少开销
    poolConfig.setTestOnBorrow(true);
    //归还连接时校验有效性(ping), 默认false, 业务量大时建议设置为false减少开销
    poolConfig.setTestOnReturn(false);
    //是否开启空闲连接检测, 如为false, 则不剔除空闲连接
    poolConfig.setTestWhileIdle(true);
    //连接空闲多久后逐出, 当空闲时间>该值, 并且空闲连接>最大空闲数时直接逐出
```

```
poolConfig.setSoftMinEvictableIdleTime(Duration.ofMillis(redisPoolSoftMinEvictableIdleTimeMillis));
//关闭根据MinEvictableIdleTimeMillis判断逐出
poolConfig.setMinEvictableIdleTime(Duration.ofMillis(-1));
//空闲连接逐出的检测周期，默认为60s
poolConfig.setTimeBetweenEvictionRuns(Duration.ofMillis(redisPoolBetweenEvictionRunsMillis));
return poolConfig;
}
}
```

- **Cluster 集群实例配置**

```
import java.time.Duration;
import java.util.ArrayList;
import java.util.List;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.data.redis.connection.RedisClusterConfiguration;
import org.springframework.data.redis.connection.RedisConnectionFactory;
import org.springframework.data.redis.connection.RedisNode;
import org.springframework.data.redis.connection.lettuce.LettuceClientConfiguration;
import org.springframework.data.redis.connection.lettuce.LettuceConnectionFactory;

import io.lettuce.core.ClientOptions;
import io.lettuce.core.SocketOptions;
import io.lettuce.core.cluster.ClusterClientOptions;
import io.lettuce.core.cluster.ClusterTopologyRefreshOptions;

/**
 * Lettuce Cluster 非池化配置，与 application.properties 配置方式二选一
 */
@Configuration
public class RedisConfiguration {

    @Value("${redis.cluster.nodes}")
    private String redisClusterNodes;

    @Value("${redis.cluster.maxRedirects:3}")
    private Integer redisClusterMaxRedirects;

    @Value("${redis.password}")
    private String redisPassword;

    @Value("${redis.connect.timeout:2000}")
    private Integer redisConnectTimeout = 2000;

    @Value("${redis.read.timeout:2000}")
    private Integer redisReadTimeout = 2000;

    @Value("${redis.cluster.topology.refresh.period.millis:10000}")
    private Integer redisClusterTopologyRefreshPeriodMillis = 10000;

    @Bean
    public RedisConnectionFactory redisConnectionFactory(LettuceClientConfiguration
clientConfiguration) {

        RedisClusterConfiguration clusterConfiguration = new RedisClusterConfiguration();

        List<RedisNode> clusterNodes = new ArrayList<>();
        for (String clusterNodeStr : redisClusterNodes.split(",")) {
            String[] nodeInfo = clusterNodeStr.split(":");
            clusterNodes.add(new RedisNode(nodeInfo[0], Integer.valueOf(nodeInfo[1])));
        }
        clusterConfiguration.setClusterNodes(clusterNodes);

        clusterConfiguration.setPassword(redisPassword);
        clusterConfiguration.setMaxRedirects(redisClusterMaxRedirects);

        LettuceConnectionFactory connectionFactory = new
```

```

LettuceConnectionFactory(clusterConfiguration, clientConfiguration);
    return connectionFactory;
}

@Bean
public LettuceClientConfiguration clientConfiguration() {

    SocketOptions socketOptions =
SocketOptions.builder().connectTimeout(Duration.ofMillis(redisConnectTimeout)).build();

    ClusterTopologyRefreshOptions topologyRefreshOptions =
ClusterTopologyRefreshOptions.builder()
        .enableAllAdaptiveRefreshTriggers()
        .enablePeriodicRefresh(Duration.ofMillis(redisClusterTopologyRefreshPeriodMillis))
        .build();

    ClusterClientOptions clientOptions = ClusterClientOptions.builder()
        .autoReconnect(true)
        .pingBeforeActivateConnection(true)
        .cancelCommandsOnReconnectFailure(false)
        .disconnectedBehavior(ClientOptions.DisconnectedBehavior.ACCEPT_COMMANDS)
        .socketOptions(socketOptions)
        .topologyRefreshOptions(topologyRefreshOptions)
        .build();

    LettuceClientConfiguration clientConfiguration = LettuceClientConfiguration.builder()
        .commandTimeout(Duration.ofMillis(redisReadTimeout))
        .readFrom(ReadFrom.MASTER)
        .clientOptions(clientOptions)
        .build();
    return clientConfiguration;
}
}

```

- Cluster实例池化配置

引入池化组件

```

<dependency>
  <groupId>org.apache.commons</groupId>
  <artifactId>commons-pool2</artifactId>
  <version>2.11.1</version>
</dependency>

```

代码配置

```

import java.time.Duration;
import java.util.ArrayList;
import java.util.List;

import org.apache.commons.pool2.impl.GenericObjectPoolConfig;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.data.redis.connection.RedisClusterConfiguration;
import org.springframework.data.redis.connection.RedisConnectionFactory;
import org.springframework.data.redis.connection.RedisNode;
import org.springframework.data.redis.connection.lettuce.LettuceClientConfiguration;
import org.springframework.data.redis.connection.lettuce.LettuceConnectionFactory;
import org.springframework.data.redis.connection.lettuce.LettucePoolingClientConfiguration;

import io.lettuce.core.ClientOptions;
import io.lettuce.core.SocketOptions;
import io.lettuce.core.cluster.ClusterClientOptions;
import io.lettuce.core.cluster.ClusterTopologyRefreshOptions;

/**
 * Lettuce 池化配置
 */
@Configuration
public class RedisPoolConfiguration {

```

```
@Value("${redis.cluster.nodes}")
private String redisClusterNodes;

@Value("${redis.cluster.maxDirects:3}")
private Integer redisClusterMaxDirects;

@Value("${redis.password}")
private String redisPassword;

@Value("${redis.connect.timeout:2000}")
private Integer redisConnectTimeout = 2000;

@Value("${redis.read.timeout:2000}")
private Integer redisReadTimeout = 2000;

@Value("${redis.cluster.topology.refresh.period.millis:10000}")
private Integer redisClusterTopologyRefreshPeriodMillis = 10000;

@Value("${redis.pool.minSize:50}")
private Integer redisPoolMinSize = 50;

@Value("${redis.pool.maxSize:200}")
private Integer redisPoolMaxSize = 200;

@Value("${redis.pool.maxWaitMillis:2000}")
private Integer redisPoolMaxWaitMillis = 2000;

@Value("${redis.pool.softMinEvictableIdleTimeMillis:1800000}")
private Integer redisPoolSoftMinEvictableIdleTimeMillis = 30 * 60 * 1000;

@Value("${redis.pool.timeBetweenEvictionRunsMillis:60000}")
private Integer redisPoolBetweenEvictionRunsMillis = 60 * 1000;

@Bean
public RedisConnectionFactory redisConnectionFactory(LettuceClientConfiguration
clientConfiguration) {

    RedisClusterConfiguration clusterConfiguration = new RedisClusterConfiguration();

    List<RedisNode> clusterNodes = new ArrayList<>();
    for (String clusterNodeStr : redisClusterNodes.split(",")) {
        String[] nodeInfo = clusterNodeStr.split(":");
        clusterNodes.add(new RedisNode(nodeInfo[0], Integer.valueOf(nodeInfo[1])));
    }
    clusterConfiguration.setClusterNodes(clusterNodes);

    clusterConfiguration.setPassword(redisPassword);
    clusterConfiguration.setMaxRedirects(redisClusterMaxDirects);

    LettuceConnectionFactory connectionFactory = new
LettuceConnectionFactory(clusterConfiguration, clientConfiguration);
    //一定要关闭共享连接, 否则连接池将不会生效
    connectionFactory.setShareNativeConnection(false);
    return connectionFactory;
}

@Bean
public LettuceClientConfiguration clientConfiguration() {

    SocketOptions socketOptions =
SocketOptions.builder().connectTimeout(Duration.ofMillis(redisConnectTimeout)).build();

    ClusterTopologyRefreshOptions topologyRefreshOptions =
ClusterTopologyRefreshOptions.builder()
        .enableAllAdaptiveRefreshTriggers()
        .enablePeriodicRefresh(Duration.ofMillis(redisClusterTopologyRefreshPeriodMillis))
        .build();
```

```
ClusterClientOptions clientOptions = ClusterClientOptions.builder()
    .autoReconnect(true)
    .pingBeforeActivateConnection(true)
    .cancelCommandsOnReconnectFailure(false)
    .disconnectedBehavior(ClientOptions.DisconnectedBehavior.ACCEPT_COMMANDS)
    .socketOptions(socketOptions)
    .topologyRefreshOptions(topologyRefreshOptions)
    .build();

LettucePoolingClientConfiguration clientConfiguration =
LettucePoolingClientConfiguration.builder()
    .poolConfig(poolConfig())
    .commandTimeout(Duration.ofMillis(redisReadTimeout))
    .clientOptions(clientOptions)
    .readFrom(ReadFrom.MASTER)
    .build();
return clientConfiguration;
}

private GenericObjectPoolConfig poolConfig() {
    GenericObjectPoolConfig poolConfig = new GenericObjectPoolConfig();
    //连接池的最小连接数
    poolConfig.setMinIdle(redisPoolMinSize);
    //连接池的最大空闲连接数
    poolConfig.setMaxIdle(redisPoolMaxSize);
    //连接池的最大连接数
    poolConfig.setMaxTotal(redisPoolMaxSize);
    //连接池耗尽后是否需要等待, 默认true表示等待. 当值为true时, setMaxWait才会生效
    poolConfig.setBlockWhenExhausted(true);
    //连接池耗尽后获取连接的最大等待时间, 默认-1表示一直等待
    poolConfig.setMaxWait(Duration.ofMillis(redisPoolMaxWaitMillis));
    //创建连接时校验有效性(ping), 默认false
    poolConfig.setTestOnCreate(false);
    //获取连接时校验有效性(ping), 默认false, 业务量大时建议设置为false减少开销
    poolConfig.setTestOnBorrow(true);
    //归还连接时校验有效性(ping), 默认false, 业务量大时建议设置为false减少开销
    poolConfig.setTestOnReturn(false);
    //是否开启空闲连接检测, 如为false, 则不剔除空闲连接
    poolConfig.setTestWhileIdle(true);
    //禁止最小空闲时间关闭连接
    poolConfig.setMinEvictableIdleTime(Duration.ofMillis(-1));
    //连接空闲多久后逐出, 当空闲时间>该值, 并且空闲连接>最大空闲数时直接逐出, 不再根据
    MinEvictableIdleTimeMillis判断 (默认逐出策略)

    poolConfig.setSoftMinEvictableIdleTime(Duration.ofMillis(redisPoolSoftMinEvictableIdleTimeMillis));
    //空闲连接逐出的检测周期, 默认为60s
    poolConfig.setTimeBetweenEvictionRuns(Duration.ofMillis(redisPoolBetweenEvictionRunsMillis));

    return poolConfig;
}
}
```

SSL 连接配置 (可选配置)

当实例开启了SSL, 通过SSL连接实例时, 请使用以下内容替换[基于Bean方式配置](#)中的LettuceClientConfiguration构造方法clientConfiguration()。Redis实例支持SSL的情况请参考[配置Redis SSL数据加密传输](#)。

- 单机、主备、读写分离、Proxy集群实例配置

```
@Bean
public LettuceClientConfiguration clientConfiguration() {

    SocketOptions socketOptions =
    SocketOptions.builder().connectTimeout(Duration.ofMillis(redisConnectTimeout)).build();

    SslOptions sslOptions = SslOptions.builder()
```

```
.trustManager(new File(certificationPath))
.build();

ClientOptions clientOptions = ClientOptions.builder()
    .sslOptions(sslOptions)
    .autoReconnect(true)
    .pingBeforeActivateConnection(true)
    .cancelCommandsOnReconnectFailure(false)
    .disconnectedBehavior(ClientOptions.DisconnectedBehavior.ACCEPT_COMMANDS)
    .socketOptions(socketOptions)
    .build();
LettuceClientConfiguration clientConfiguration = LettuceClientConfiguration.builder()
    .commandTimeout(Duration.ofMillis(redisReadTimeout))
    .readFrom(ReadFrom.MASTER)
    .clientOptions(clientOptions)
    .useSsl()
    .build();

return clientConfiguration;
}
```

- **Cluster 集群实例配置**

```
@Bean
public LettuceClientConfiguration clientConfiguration() {

    SocketOptions socketOptions =
    SocketOptions.builder().connectTimeout(Duration.ofMillis(redisConnectTimeout)).build();

    SslOptions sslOptions = SslOptions.builder()
        .trustManager(new File(certificationPath))
        .build();

    ClusterTopologyRefreshOptions topologyRefreshOptions = ClusterTopologyRefreshOptions.builder()
        .enableAllAdaptiveRefreshTriggers()
        .enablePeriodicRefresh(Duration.ofMillis(redisClusterTopologyRefreshPeriodMillis))
        .build();

    ClusterClientOptions clientOptions = ClusterClientOptions.builder()
        .sslOptions(sslOptions)
        .autoReconnect(true)
        .pingBeforeActivateConnection(true)
        .cancelCommandsOnReconnectFailure(false)
        .disconnectedBehavior(ClientOptions.DisconnectedBehavior.ACCEPT_COMMANDS)
        .socketOptions(socketOptions)
        .topologyRefreshOptions(topologyRefreshOptions)
        .build();

    LettuceClientConfiguration clientConfiguration = LettuceClientConfiguration.builder()
        .commandTimeout(Duration.ofMillis(redisReadTimeout))
        .readFrom(ReadFrom.MASTER)
        .clientOptions(clientOptions)
        .useSsl()
        .build();

    return clientConfiguration;
}
```

参数明细

表 4-7 LettuceConnectionFactory 参数

参数	类型	默认值	说明
configuration	RedisConfigura tion	-	redis连接配置，常用两个子类： <ul style="list-style-type: none"> • RedisStandaloneConfiguration • RedisClusterConfiguration
clientConfigur ation	LettuceClientCo nfiguration	-	客户端配置参数，常用子类： LettucePoolingClientConfiguration (用于池化)
shareNativeCo nnection	boolean	true	是否采用共享连接，默认true，采用 连接池时必须设置为false

表 4-8 RedisStandaloneConfiguration 参数

参数	默认值	说明
hostName	localhost	连接Redis实例的IP地址/域名
port	6379	连接端口号
database	0	数据库下标
password	-	连接密码

表 4-9 RedisClusterConfiguration 参数

参数	说明
clusterNodes	cluster节点连接信息，需节点IP、Port
maxRedirects	cluster访问最大重定向次数， 建议值：3
password	连接密码

表 4-10 LettuceClientConfiguration 参数

参数	类型	默认值	说明
timeout	Duration	60s	命令超时时间配置， 建议值：2s
clientOptions	ClientOptions	-	配置项

参数	类型	默认值	说明
readFrom	readFrom	MASTER	读取模式，建议值：MASTER，其余配置在发生故障切换场景下，均存在访问失败风险

表 4-11 LettucePoolingClientConfiguration 参数

参数	类型	默认值	说明
timeout	Duration	60s	命令超时时间配置， 建议值：2s
clientOptions	ClientOptions	-	配置项
poolConfig	GenericObjectPoolConfig	-	连接池配置
readFrom	readFrom	MASTER	读取模式，建议值：MASTER，其余配置在发生故障切换场景下，均存在访问失败风险

表 4-12 ClientOptions 参数

参数	类型	默认值	说明
autoReconnect	boolean	true	连接断开后，是否自动发起重连， 建议值：true
pingBeforeActivateConnection	boolean	true	连接创建后，是否通过ping/pong校验连接可用性， 建议值：true
cancelCommandsOnReconnectFailure	boolean	true	连接重连失败时，是否取消队列中的命令， 建议值：false
disconnectedBehavior	DisconnectedBehavior	DisconnectedBehavior.DEFAULT	连接断开时的行为， 建议值：ACCEPT_COMMANDS <ul style="list-style-type: none"> • DEFAULT：当autoReconnect为true时，允许命令进入队列等待，当autoReconnect为false时，禁止命令进入队列等待 • ACCEPT_COMMANDS：允许命令进入队列等待 • REJECT_COMMANDS：禁止命令进入队列等待
socketOptions	SocketOptions	-	网络配置项

表 4-13 SocketOptions 参数

参数	默认值	说明
connectTimeout	10s	连接超时时间配置， 建议值：2s

表 4-14 GenericObjectPoolConfig 参数

参数	默认值	说明
minIdle	-	连接池的最小连接数
maxIdle	-	连接池的最大空闲连接数
maxTotal	-	连接池的最大连接数
blockWhenExhausted	true	连接池耗尽后是否需要等待，默认true表示等待。当值为true时，设置maxWaitMillis才会生效
maxWaitMillis	-1	连接池耗尽后获取连接的最大等待时间，默认-1表示一直等待
testOnCreate	false	创建连接时校验有效性(ping)，默认false
testOnBorrow	false	获取连接时校验有效性(ping)，默认false，业务量大时建议设置为false减少开销
testOnReturn	false	归还连接时校验有效性(ping)，默认false，业务量大时建议设置为false减少开销
testWhileIdle	false	是否开启空闲连接检测，如为false，则不剔除空闲连接， 建议值：true
softMinEvictableIdleTimeMillis	1800000	连接空闲多久后逐出，（空闲时间>该值 && 空闲连接>最大空闲数）时直接逐出
minEvictableIdleTimeMillis	60000	根据minEvictableIdleTimeMillis判断逐出， 建议值：-1 ，关闭该策略，改用softMinEvictableIdleTimeMillis策略
timeBetweenEvictionRunsMillis	60000	空闲连接逐出的检测周期，单位：毫秒

DCS 实例配置建议

- 连接池化

因lettuce底层采用基于netty的NIO模式，和redis server进行通信，不同于jedis的BIO模式。底层采用长连接 + 队列的组合模式，借助TCP顺序发、顺序收的特性，来实现同时处理多请求发送和多响应接收，单条连接可支撑的QPS在3K~5K不等，线上系统建议不要超过3K。lettuce本身不支持池化，且在springboot中默认

不开启池化，如需开启池化，需通过手动引入commons-pool2组件，并关闭LettuceConnectionFactory.shareNativeConnection（共享连接）来实现池化。

因每条lettuce连接默认需要配置两个线程池-I/O thread pools、computation thread pool，用于支撑IO事件读取和异步event处理，如配置成连接池形式使用，每个连接都将会创建两个线程池，对内存资源的占用偏高。**鉴于lettuce的底层模型实现，及单连接突出的处理能力，不建议通过池化的方式使用lettuce。**

- 拓扑刷新

在连接cluster类型实例中，lettuce会在初始化时，向配置的节点列表随机发送cluster nodes来获取集群slot的分布信息。如后续cluster扩/缩容、主备切换等，会导致集群拓扑结构发生变化，**lettuce默认是不感知的，需手动开启主动感知拓扑结构变化**，如下：

- **基于application.properties配置**

```
# 开启自适应拓扑刷新
spring.redis.lettuce.cluster.refresh.adaptive=true
# 开启每10s定时刷新拓扑结构
spring.redis.lettuce.cluster.refresh.period=10S
```

- **基于API配置**

```
ClusterTopologyRefreshOptions topologyRefreshOptions =
ClusterTopologyRefreshOptions.builder()
    .enableAllAdaptiveRefreshTriggers()
    .enablePeriodicRefresh(Duration.ofMillis(redisClusterTopologyRefreshPeriodMillis))
    .build();

ClusterClientOptions clientOptions = ClusterClientOptions.builder()
    ...
    ...
    .topologyRefreshOptions(topologyRefreshOptions)
    .build();
```

- 爆炸半径

因lettuce底层采用的是单长连接 + 请求队列的组合模式，一旦遇到网络抖动/闪断，或连接失活，将影响所有请求，尤其是在连接失活场景中，将尝试tcp重传，直至重传超时关闭连接，待连接重建后才能恢复。在重传期间请求队列会不断堆积请求，上层业务非常容易出现批量超时，甚至在部分操作系统内核中的重传超时配置过长，致使业务系统长时间处于不可用状态。因此，**不推荐使用lettuce组件，建议用jedis组件替换。**

4.3.4 Redisson 客户端连接 Redis（Java）

本章节介绍使用Redisson客户端连接Redis实例的方法。更多的客户端的使用方法请参考[Redis客户端](#)。

在springboot类型的项目中，spring-data-redis中提供了对jedis、lettuce的适配，但没有提供对redisson组件的适配。为了能够在springboot中集成redisson，redisson侧主动提供了适配springboot的组件：redisson-spring-boot-starter（请参考：<https://mvnrepository.com/artifact/org.redisson/redisson>）。

注意：在springboot1.x中默认集成的是jedis，springboot2.x中改为了lettuce。

📖 说明

- 如果创建Redis实例时设置了密码，使用Redisson客户端连接Redis时，需要配置密码进行连接，建议不要将明文密码硬编码在代码中。
- 连接单机、读写分离、Proxy集群实例需要使用Redisson的SingleServerConfig配置对象中的useSingleServer方法，连接主备实例需要使用Redisson的MasterSlaveServersConfig配置对象中的useMasterSlaveServers方法，Cluster集群实例需要使用ClusterServersConfig对象中的useClusterServers方法。
- Springboot版本不得低于2.3.12.RELEASE，Redisson版本不得低于3.37.0。

前提条件

- 已成功创建Redis实例，且状态为“运行中”。创建Redis实例的操作请参考[购买Redis实例](#)。
- 查看并获取待连接Redis实例的IP地址/域名和端口。具体步骤请参见[查看和修改DCS实例基本信息](#)。

Pom 配置

```
<!-- 引入spring-data-redis组件 -->
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-data-redis</artifactId>
  <exclusions>
    <!-- 因springboot2.x中默认集成了lettuce，因此需要排掉该依赖 -->
    <exclusion>
      <artifactId>lettuce-core</artifactId>
      <groupId>io.lettuce</groupId>
    </exclusion>
  </exclusions>
</dependency>
<!-- 引入redisson对springboot的集成适配包 -->
<dependency>
  <groupId>org.redisson</groupId>
  <artifactId>redisson-spring-boot-starter</artifactId>
  <version>${redisson.version}</version>
</dependency>
```

基于 Bean 方式配置

因springboot中没有提供对redisson的适配，在application.properties配置文件自然也没有对应的配置项，只能通过基于Bean的方式注入。

- 单机、读写分离、Proxy集群实例配置

```
import org.redisson.Redisson;
import org.redisson.api.RedissonClient;
import org.redisson.codec.JsonJacksonCodec;
import org.redisson.config.Config;
import org.redisson.config.SingleServerConfig;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class SingleConfig {

    @Value("${redis.address}")
    private String redisAddress;

    @Value("${redis.password}")
    private String redisPassword;
```

```
@Value("${redis.database:0}")
private Integer redisDatabase = 0;

@Value("${redis.connect.timeout:3000}")
private Integer redisConnectTimeout = 3000;

@Value("${redis.connection.idle.timeout:10000}")
private Integer redisConnectionIdleTimeout = 10000;

@Value("${redis.connection.ping.interval:1000}")
private Integer redisConnectionPingInterval = 1000;

@Value("${redis.timeout:2000}")
private Integer timeout = 2000;

@Value("${redis.connection.pool.min.size:50}")
private Integer redisConnectionPoolMinSize;

@Value("${redis.connection.pool.max.size:200}")
private Integer redisConnectionPoolMaxSize;

@Value("${redis.retry.attempts:3}")
private Integer redisRetryAttempts = 3;

@Value("${redis.retry.interval:200}")
private Integer redisRetryInterval = 200;

@Bean
public RedissonClient redissonClient(){
    Config redissonConfig = new Config();

    SingleServerConfig serverConfig = redissonConfig.useSingleServer();
    serverConfig.setAddress(redisAddress);
    serverConfig.setConnectionMinimumIdleSize(redisConnectionPoolMinSize);
    serverConfig.setConnectionPoolSize(redisConnectionPoolMaxSize);

    serverConfig.setDatabase(redisDatabase);
    serverConfig.setPassword(redisPassword);
    serverConfig.setConnectTimeout(redisConnectTimeout);
    serverConfig.setIdleConnectionTimeout(redisConnectionIdleTimeout);
    serverConfig.setPingConnectionInterval(redisConnectionPingInterval);
    serverConfig.setTimeout(timeout);
    serverConfig.setRetryAttempts(redisRetryAttempts);
    serverConfig.setRetryInterval(redisRetryInterval);

    redissonConfig.setCodec(new JsonJacksonCodec());
    return Redisson.create(redissonConfig);
}
}
```

- **主备实例配置**

```
import org.redisson.Redisson;
import org.redisson.api.RedissonClient;
import org.redisson.codec.JsonJacksonCodec;
import org.redisson.config.Config;
import org.redisson.config.MasterSlaveServersConfig;
import org.redisson.config.ReadMode;
import org.redisson.config.SubscriptionMode;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

import java.util.HashSet;

@Configuration
public class MasterStandbyConfig {
    @Value("${redis.master.address}")
    private String redisMasterAddress;

    @Value("${redis.slave.address}")
```

```

private String redisSlaveAddress;

@Value("${redis.database:0}")
private Integer redisDatabase = 0;

@Value("${redis.password:}")
private String redisPassword;

@Value("${redis.connect.timeout:3000}")
private Integer redisConnectTimeout = 3000;

@Value("${redis.connection.idle.timeout:10000}")
private Integer redisConnectionIdleTimeout = 10000;

@Value("${redis.connection.ping.interval:1000}")
private Integer redisConnectionPingInterval = 1000;

@Value("${redis.timeout:2000}")
private Integer timeout = 2000;

@Value("${redis.master.connection.pool.min.size:50}")
private Integer redisMasterConnectionPoolMinSize = 50;

@Value("${redis.master.connection.pool.max.size:200}")
private Integer redisMasterConnectionPoolMaxSize = 200;

@Value("${redis.retry.attempts:3}")
private Integer redisRetryAttempts = 3;

@Value("${redis.retry.interval:200}")
private Integer redisRetryInterval = 200;

@Bean
public RedissonClient redissonClient() {
    Config redissonConfig = new Config();

    MasterSlaveServersConfig serverConfig = redissonConfig.useMasterSlaveServers();
    serverConfig.setMasterAddress(redisMasterAddress);
    HashSet<String> slaveSet = new HashSet<>();
    slaveSet.add(redisSlaveAddress);
    serverConfig.setSlaveAddresses(slaveSet);

    serverConfig.setDatabase(redisDatabase);
    serverConfig.setPassword(redisPassword);

    serverConfig.setMasterConnectionMinimumIdleSize(redisMasterConnectionPoolMinSize);
    serverConfig.setMasterConnectionPoolSize(redisMasterConnectionPoolMaxSize);

    serverConfig.setReadMode(ReadMode.MASTER);
    serverConfig.setSubscriptionMode(SubscriptionMode.MASTER);

    serverConfig.setConnectTimeout(redisConnectTimeout);
    serverConfig.setIdleConnectionTimeout(redisConnectionIdleTimeout);
    serverConfig.setPingConnectionInterval(redisConnectionPingInterval);
    serverConfig.setTimeout(timeout);
    serverConfig.setRetryAttempts(redisRetryAttempts);
    serverConfig.setRetryInterval(redisRetryInterval);

    redissonConfig.setCodec(new JsonJacksonCodec());
    return Redisson.create(redissonConfig);
}
}

```

- **Cluster 集群实例配置**

```

import org.redisson.Redisson;
import org.redisson.api.RedissonClient;
import org.redisson.codec.JsonJacksonCodec;
import org.redisson.config.ClusterServersConfig;
import org.redisson.config.Config;
import org.redisson.config.ReadMode;

```

```
import org.redisson.config.SubscriptionMode;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

import java.util.List;

@Configuration
public class ClusterConfig {

    @Value("${redis.cluster.address}")
    private List<String> redisClusterAddress;

    @Value("${redis.cluster.scan.interval:5000}")
    private Integer redisClusterScanInterval = 5000;

    @Value("${redis.password}")
    private String redisPassword;

    @Value("${redis.connect.timeout:3000}")
    private Integer redisConnectTimeout = 3000;

    @Value("${redis.connection.idle.timeout:10000}")
    private Integer redisConnectionIdleTimeout = 10000;

    @Value("${redis.connection.ping.interval:1000}")
    private Integer redisConnectionPingInterval = 1000;

    @Value("${redis.timeout:2000}")
    private Integer timeout = 2000;

    @Value("${redis.retry.attempts:3}")
    private Integer redisRetryAttempts = 3;

    @Value("${redis.retry.interval:200}")
    private Integer redisRetryInterval = 200;

    @Value("${redis.master.connection.pool.min.size:50}")
    private Integer redisMasterConnectionPoolMinSize = 50;

    @Value("${redis.master.connection.pool.max.size:200}")
    private Integer redisMasterConnectionPoolMaxSize = 200;

    @Bean
    public RedissonClient redissonClient() {
        Config redissonConfig = new Config();

        ClusterServersConfig serverConfig = redissonConfig.useClusterServers();
        serverConfig.setNodeAddresses(redisClusterAddress);
        serverConfig.setScanInterval(redisClusterScanInterval);

        serverConfig.setPassword(redisPassword);

        serverConfig.setMasterConnectionMinimumIdleSize(redisMasterConnectionPoolMinSize);
        serverConfig.setMasterConnectionPoolSize(redisMasterConnectionPoolMaxSize);

        serverConfig.setReadMode(ReadMode.MASTER);
        serverConfig.setSubscriptionMode(SubscriptionMode.MASTER);

        serverConfig.setConnectTimeout(redisConnectTimeout);
        serverConfig.setIdleConnectionTimeout(redisConnectionIdleTimeout);
        serverConfig.setPingConnectionInterval(redisConnectionPingInterval);
        serverConfig.setTimeout(timeout);
        serverConfig.setRetryAttempts(redisRetryAttempts);
        serverConfig.setRetryInterval(redisRetryInterval);

        redissonConfig.setCodec(new JsonJacksonCodec());
        return Redisson.create(redissonConfig);
    }
}
```

```
}  
}
```

SSL 连接配置（可选配置）

当实例开启了SSL，通过SSL连接实例时，请将[基于Bean方式配置](#)中的RedissonClient构造方法clientConfiguration()中添加如下configRedissonSSL(serverConfig)逻辑，同时将redis的连接地址从redis://ip:port改为rediss://ip:port格式。Redis实例支持SSL的情况请参考[配置Redis SSL数据加密传输](#)。

```
private void configRedissonSSL(BaseConfig serverConfig) {  
    TrustManagerFactory trustManagerFactory = null;  
    try {  
        //加载自定义路径下的ca证书,可结合具体业务配置  
        CertificateFactory cf = CertificateFactory.getInstance("X.509");  
        Certificate ca;  
        try (InputStream is = new FileInputStream(certificationPath)) {  
            ca = cf.generateCertificate(is);  
        }  
  
        //创建keystore  
        String keyStoreType = KeyStore.getDefaultType();  
        KeyStore keyStore = KeyStore.getInstance(keyStoreType);  
        keyStore.load(null, null);  
        keyStore.setCertificateEntry("ca", ca);  
  
        //创建TrustManager  
        trustManagerFactory = TrustManagerFactory.getInstance(TrustManagerFactory.getDefaultAlgorithm());  
        trustManagerFactory.init(keyStore);  
    } catch (CertificateException | IOException | KeyStoreException | NoSuchAlgorithmException e) {  
        e.printStackTrace();  
        return;  
    }  
    serverConfig.setSslTrustManagerFactory(trustManagerFactory);  
}
```

参数明细

表 4-15 Config 参数

参数	默认值	说明
codec	org.redisson.codec.JsonJacksonCodec	编码格式，内置了JSON/Avro/Smile/CBOR/MsgPack等编码格式
threads	cpu核数 * 2	RTopic Listener、RRemoteService和RExecutorService执行使用的线程池
executor	null	功能同上，不设置该参数时，会根据threads参数初始化一个线程池
nettyThreads	cpu核数 * 2	连接redis-server的tcp channel使用的线程池，所有channel共享该连接池，映射到netty即Bootstrap.group(...)
eventLoopGroup	null	功能同上，不设置该参数时，会根据nettyThreads参数初始化一个EventLoopGroup，用于底层tcpchannel使用

参数	默认值	说明
transportMode	TransportMode.NIO	传输模式，可选有NIO、EPOLL（需额外引包）、KQUEUE（需额外引包）
lockWatchdogTimeout	30000	监控锁的看门狗超时时间，单位：毫秒。用于分布式锁场景下未指定leaseTimeout参数时，采用该值为默认值
keepPubSubOrder	true	是否按照订阅发布消息的顺序来接收， 如能接受并行处理消息，建议设置为false

表 4-16 单机、读写分离、Proxy 集群实例 SingleServerConfig 参数

参数	默认值	说明
address	-	节点连接信息，redis://ip:port
database	0	选择使用的数据库编号
connectionMinimumIdleSize	32	连接每个分片主节点的最小连接数
connectionPoolSize	64	连接每个分片主节点的最大连接数
subscriptionConnectionMinimumIdleSize	1	连接目标节点的用于发布订阅的最小连接数
subscriptionConnectionPoolSize	50	连接目标节点的用于发布订阅的最大连接数
subscriptionPerConnection	5	每个订阅连接上的最大订阅数量
connectionTimeout	10000	连接超时时间，单位：毫秒
idleConnectionTimeout	10000	空闲连接的最大回收时间，单位：毫秒
pingConnectionInterval	30000	检测连接可用心跳，单位：毫秒， 建议值：3000ms
timeout	3000	请求等待响应的超时时间，单位：毫秒
retryAttempts	3	发送失败的最大重试次数
retryInterval	1500	每次重试的时间间隔，单位：毫秒， 建议值：200ms
clientName	null	客户端名称

表 4-17 主备实例 MasterSlaveServersConfig 参数

参数	默认值	说明
masterAddress	-	主节点连接信息，redis://ip:port。
slaveAddresses	-	从节点连接信息列表，Set<redis://ip:port>。
readMode	SLAVE	读取模式，默认读流量分发到从节点，可选值：MASTER、SLAVE、MASTER_SLAVE； 建议MASTER ，其余配置在故障切换场景下，均存在访问失败风险。
loadBalancer	RoundRobinLoad Balancer	负载均衡算法，在readMode为SLAVE、MASTER_SLAVE时生效，均衡读流量分发。
masterConnectionMinimumIdleSize	32	连接每个分片主节点的最小连接数。
masterConnectionPoolSize	64	连接每个分片主节点的最大连接数。
slaveConnectionMinimumIdleSize	32	连接每个分片每个从节点的最小连接数，如readMode=MASTER，该配置值将失效。
slaveConnectionPoolSize	64	连接每个分片每个从节点的最大连接数，如readMode=MASTER，该配置值将失效。
subscriptionMode	SLAVE	订阅模式，默认只在从节点订阅，可选值：SLAVE、MASTER； 建议采用MASTER 。
subscriptionConnectionMinimumIdleSize	1	连接目标节点的用于发布订阅的最小连接数。
subscriptionConnectionPoolSize	50	连接目标节点的用于发布订阅的最大连接数。
subscriptionPerConnection	5	每个订阅连接上的最大订阅数量。
connectionTimeout	10000	连接超时时间，单位：毫秒。
idleConnectionTimeout	10000	空闲连接的最大回收时间，单位：毫秒。
pingConnectionInterval	30000	检测连接可用心跳，单位：毫秒，建议值： 3000ms 。
timeout	3000	请求等待响应的超时时间，单位：毫秒。
retryAttempts	3	发送失败的最大重试次数。

参数	默认值	说明
retryInterval	1500	每次重试的时间间隔，单位：毫秒， 建议值：200ms 。
clientName	null	客户端名称。

表 4-18 Cluster 集群实例 ClusterServersConfig 参数

参数	默认值	说明
nodeAddress	-	集群节点的地址连接信息，每个节点采用 redis://ip:port 方式，多个节点连接信息用英文逗号隔开。
password	null	集群登录密码。
scanInterval	1000	定时检测集群节点状态的时间间隔，单位：毫秒。
readMode	SLAVE	读取模式，默认读流量分发到从节点，可选值：MASTER、SLAVE、MASTER_SLAVE； 建议修改为MASTER ，其余配置在故障切换场景下，均存在访问失败风险。
loadBalancer	RoundRobinLoadBalancer	负载均衡算法，在readMode为SLAVE、MASTER_SLAVE时生效，均衡读流量分发。
masterConnectionMinimumIdleSize	32	连接每个分片主节点的最小连接数。
masterConnectionPoolSize	64	连接每个分片主节点的最大连接数。
slaveConnectionMinimumIdleSize	32	连接每个分片每个从节点的最小连接数，如readMode=MASTER，该配置值将失效。
slaveConnectionPoolSize	64	连接每个分片每个从节点的最大连接数，如readMode=MASTER，该配置值将失效。
subscriptionMode	SLAVE	订阅模式，默认只在从节点订阅，可选值：SLAVE、MASTER； 建议采用MASTER 。
subscriptionConnectionMinimumIdleSize	1	连接目标节点的用于发布订阅的最小连接数。
subscriptionConnectionPoolSize	50	连接目标节点的用于发布订阅的最大连接数。
subscriptionPerConnection	5	每个订阅连接上的最大订阅数量。

参数	默认值	说明
connectionTimeout	10000	连接超时时间，单位：毫秒。
idleConnectionTimeout	10000	空闲连接的最大回收时间，单位：毫秒。
pingConnectionInterval	30000	检测连接可用心跳，单位：毫秒， 建议值：3000 。
timeout	3000	请求等待响应的超时时间，单位：毫秒。
retryAttempts	3	发送失败的最大重试次数。
retryInterval	1500	每次重试的时间间隔，单位：毫秒， 建议值：200 。
clientName	null	客户端名称。

DCS 实例配置建议

- 读取模式（readMode）**
 建议采用MASTER，即Master节点承担所有的读写流量，一方面避免数据因主从同步时延带来的一致性问题；另一方面，如果从节点故障，配置值=SLAVE，所有读请求会触发报错；配置值=MASTER_SLAVE，部分读请求会触发异常。读报错会持续failedSlaveCheckInterval（默认180s）时间，直至从可用节点列表中移除。
 如需读写流量分流处理，DCS服务提供了针对读写流量分流的读写分离实例类型，通过在中间架设代理节点实现读写流量分发，遇到从节点故障时，自动切流至主节点，对业务应用无感知，且故障感知时间窗口远小于redisson内部的时间窗口。
- 订阅模式（subscriptionMode）**
 建议采用MASTER，原理同**读取模式（readMode）**。
- 连接池配置**

说明

以下计算方式只适用于一般业务场景，建议根据业务情况做适当调整适配。

连接池的大小没有绝对的标准，建议根据业务流量进行合理配置，一般连接池大小的参数计算公式如下：

- 最小连接数 = (单机访问Redis QPS) / (1000ms / 单命令平均耗时)
- 最大连接数 = (单机访问Redis QPS) / (1000ms / 单命令平均耗时) * 150%

举例：某个业务应用的QPS为10000左右，每个请求需访问Redis10次，即每秒对Redis的访问次数为100000次，同时该业务应用有10台机器，计算如下：

单机访问Redis QPS = 100000 / 10 = 10000

单命令平均耗时 = 20ms（Redis处理单命令耗时为5~10ms，遇到网络抖动按照15~20ms来估算）

最小连接数 = (10000) / (1000ms / 20ms) = 200

最大连接数 = (10000) / (1000ms / 20ms) * 150% = 300

- **重试配置**
redisson中支持重试配置，主要是如下两个参数，建议根据业务情况配置合理值，一般重试次数为3，重试间隔为200ms左右。
 - retryAttempts: 配置重试次数
 - retryInterval: 配置重试时间间隔

📖 说明

在redisson中，部分API通过借助LUA的方式实现，性能表现上偏低，建议使用jedis客户端替换redisson。

4.3.5 Redis-py 客户端连接 Redis (Python)

本章节介绍使用Python Redis客户端redis-py连接Redis实例的方法。更多的客户端的使用方法请参考[Redis客户端](#)。

以下操作以通过弹性云服务器上的客户端连接Redis实例为例进行说明。

📖 说明

连接单机、主备、Proxy集群实例建议使用redis-py，Cluster集群实例建议使用redis-py-cluster。

前提条件

- 已成功创建Redis实例，且状态为“运行中”。创建Redis实例的操作请参考[购买Redis实例](#)。
- 已创建弹性云服务器，创建弹性云服务器的方法，请参见[购买弹性云服务器](#)。
- 如果弹性云服务器为Linux系统，该弹性云服务器必须已经安装Python编译环境。
- 连接实例前确保客户端与Redis实例之间网络互通，具体请参考[连接Redis网络要求](#)。

Redis-py 客户端连接 Redis

步骤1 查看并获取待连接Redis实例的IP地址/域名和端口。

具体步骤请参见[查看和修改DCS实例基本信息](#)。

步骤2 登录弹性云服务器。

本章节以弹性云服务器操作系统为centos为例介绍通过Python Redis客户端连接实例。

步骤3 连接Redis实例。

如果弹性云服务器操作系统没有自带Python，可以使用yum方式安装。

```
yum install python
```

📖 说明

要求系统Python版本为3.6+，当默认Python版本小于3.6时，可通过以下操作修改Python默认版本。

1. 删除Python软链接文件： `rm -rf python`
 2. 重新创建新指向Python： `ln -s pythonX.X.X python`，其中X为Python具体版本号。
- **连接单机、主备、proxy集群实例。**

- a. 安装Python和Python Redis客户端redis-py。
 - i. 如果系统没有自带Python，可以使用yum方式安装。
 - ii. 下载并解压redis-py。
`wget https://github.com/andymccurdy/redis-py/archive/master.zip`
`unzip master.zip`
 - iii. 进入到解压目录后安装Python Redis客户端redis-py。
`python setup.py install`

安装后执行python命令，返回如下信息说明成功安装redis-py：

图 4-7 执行 python

```
[root@ecs-... redis-py-master]# python
Python 3.6.8 (default, Nov 16 2020, 16:55:22)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-44)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import redis
>>>
```

- b. 使用redis-py客户端连接实例。以下步骤以命令行模式进行示例（也可以将命令写入python脚本中再执行）：
 - i. 执行python命令，进入命令行模式。返回如下信息说明已进入命令行模式：

图 4-8 进入命令行模式

```
[root@ecs-... redis-py-master]# python
Python 3.6.8 (default, Nov 16 2020, 16:55:22)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-44)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import redis
>>>
```

- ii. 在命令行中执行以下命令，连接Redis实例。
`r = redis.StrictRedis(host='XXX.XXX.XXX.XXX', port=6379, password='*****');`

其中，XXX.XXX.XXX.XXX为Redis实例的IP地址/域名，“6379”为Redis实例的端口。IP地址/域名和端口获取见[步骤1](#)，请按实际情况修改后执行。*****为创建Redis实例时自定义的密码，请按实际情况修改后执行。

界面显示一行新的命令行，说明连接Redis实例成功。可以输入命令对数据库进行读写操作。

图 4-9 连接 redis 成功

```
>>> r = redis.StrictRedis(host='...', port=6379, password='...');
>>> r.set("foo", "bar")
True
>>> print(r.get("foo"))
b'bar'
>>> _
```

● 连接Cluster集群实例。

- a. 安装redis-py-cluster客户端。
 - i. 执行以下命令下载released版本。
`wget https://github.com/Grokzen/redis-py-cluster/releases/download/2.1.3/redis-py-cluster-2.1.3.tar.gz`
 - ii. 解压压缩包。
`tar -xvf redis-py-cluster-2.1.3.tar.gz`
 - iii. 进入到解压目录后安装Python Redis客户端redis-py-cluster。
`python setup.py install`

- b. 使用redis-py-cluster客户端连接Redis实例。

以下步骤以命令行模式进行示例（也可以将命令写入python脚本中再执行）：

- i. 执行python命令，进入命令行模式。
ii. 在命令行中执行以下命令，连接Redis实例。如果实例为免密访问，则省略命令中的, password='*****'

```
>>> from rediscluster import RedisCluster

>>> startup_nodes = [{"host": "192.168.0.143", "port": "6379"}, {"host": "192.168.0.144", "port": "6379"}, {"host": "192.168.0.145", "port": "6379"}, {"host": "192.168.0.146", "port": "6379"}]

>>> rc = RedisCluster(startup_nodes=startup_nodes, decode_responses=True, password='*****')

>>> rc.set("foo", "bar")
True
>>> print(rc.get("foo"))
'bar'
```

---结束

4.3.6 Go-redis 客户端连接 Redis (Go)

本章节介绍使用go-redis客户端连接Redis实例的方法。更多的客户端的使用方法请参考[Redis客户端](#)。

以下操作以通过弹性云服务器上的客户端连接Redis实例为例进行说明。

前提条件

- 已成功创建Redis实例，且状态为“运行中”。创建Redis实例的操作请参考[购买Redis实例](#)。
- 查看并获取待连接Redis实例的IP地址/域名和端口。具体步骤请参见[查看和修改DCS实例基本信息](#)。
- 已创建弹性云服务器，创建弹性云服务器的方法，请参见[购买弹性云服务器](#)。
- 连接实例前确保客户端与Redis实例之间网络互通，具体请参考[连接Redis网络要求](#)。

Go-redis 客户端连接 Redis

步骤1 登录弹性云服务器。

弹性云服务器操作系统，这里以Window为例。

步骤2 在弹性云服务器安装VS 2017社区版。

步骤3 启动VS 2017，新建一个工程，工程名自定义，这里设置为“redisdemo”。

步骤4 导入go-redis的依赖包，在终端输入go get github.com/go-redis/redis。

图 4-10 终端输入

```

25
26 //集群
27 rdbCluster := redis.NewClusterClient(&redis.ClusterOptions{
28     Addr: []string{"host:port"},
29     Password: "*****",
30 })
31 val1, err1 := rdbCluster.Get("key").Result()
32 if err1 != nil {
33     if err == redis.Nil {
34         fmt.Println("key does not exists")
35     }
36     return
37 }
38
39 go mod tidy
40
41 PS C:\Users\... \go\src\testProject> go get github.com/go-redis/redis
42 go: downloading github.com/go-redis/redis v6.15.9+incompatible
43 go get: added github.com/go-redis/redis v6.15.9+incompatible
44 PS C:\Users\... \go\src\testProject> git build -o goDemo main.go

```

步骤5 编写如下代码：

```

package main

import (
    "fmt"
    "github.com/go-redis/redis"
)

func main() {
    // 单机
    rdb := redis.NewClient(&redis.Options{
        Addr: "host:port",
        Password: "*****", // no password set
        DB: 0, // use default DB
    })

    val, err := rdb.Get("key").Result()
    if err != nil {
        if err == redis.Nil {
            fmt.Println("key does not exists")
            return
        }
        panic(err)
    }
    fmt.Println(val)

    // 集群
    rdbCluster := redis.NewClusterClient(&redis.ClusterOptions{
        Addr: []string{"host:port"},
        Password: "*****",
    })
    val1, err1 := rdbCluster.Get("key").Result()
    if err1 != nil {
        if err == redis.Nil {
            fmt.Println("key does not exists")
            return
        }
        panic(err)
    }
    fmt.Println(val1)
}

```

其中，**host:port**分别为Redis实例的IP地址/域名以及端口。IP地址/域名和端口获取见[前提条件](#)，请按实际情况修改后执行。*****为创建Redis实例时自定义的密码，请按实际情况修改后执行。

步骤6 执行go build -o test main.go命令进行打包，如打包名为test可执行文件。

⚠ 注意

若打包后需要在Linux系统下运行则需要在打包前设置：

```
set GOARCH=amd64
```

```
set GOOS=linux
```

步骤7 执行./test连接实例。

----结束

4.3.7 Hiredis 客户端连接 Redis (C++)

本章节介绍使用C++ hiredis连接Redis实例的方法。更多的客户端的使用方法请参考[Redis客户端](#)。

以下操作以通过弹性云服务器上的客户端连接Redis实例为例进行说明。

📖 说明

本章节操作，仅适用于连接单机、主备、Proxy集群实例，如果是使用C++ Redis客户端连接Cluster集群，请参考[C++ Redis客户端](#)。

前提条件

- 已成功创建Redis实例，且状态为“运行中”。创建Redis实例的操作请参考[购买Redis实例](#)。
- 已创建弹性云服务器，创建弹性云服务器的方法，请参见[购买弹性云服务器](#)。
- 如果弹性云服务器为Linux系统，该弹性云服务器必须已经安装gcc编译环境。
- 连接实例前确保客户端与Redis实例之间网络互通，具体请参考[连接Redis网络要求](#)。

Hiredis 客户端连接 Redis

步骤1 查看并获取待连接Redis实例的IP地址/域名和端口。

具体步骤请参见[查看和修改DCS实例基本信息](#)。

步骤2 登录弹性云服务器。

本章节以弹性云服务器操作系统为centos为例介绍通过C++ redis客户端连接实例。

步骤3 安装gcc、make和hiredis。

如果系统没有自带编译环境，可以使用yum方式安装。

```
yum install gcc make
```

步骤4 下载并解压hiredis。

```
wget https://github.com/redis/hiredis/archive/master.zip  
unzip master.zip
```

步骤5 进入到解压目录后编译安装。

```
make  
make install
```

步骤6 使用hiredis客户端连接Redis实例。

关于hiredis的使用，请参考redis官网的使用介绍。这里举一个简单的例子，介绍连接、密码鉴权等的使用。

1. 编辑连接Redis实例的demo示例，然后保存退出。

```
vim connRedis.c
```

示例内容如下：

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <hiredis.h>
int main(int argc, char **argv) {
    unsigned int j;
    redisContext *conn;
    redisReply *reply;
    if (argc < 3) {
        printf("Usage: example {instance_ip_address} 6379 {password}\n");
        exit(0);
    }
    const char *hostname = argv[1];
    const int port = atoi(argv[2]);
    const char *password = argv[3];
    struct timeval timeout = { 1, 500000 }; // 1.5 seconds
    conn = redisConnectWithTimeout(hostname, port, timeout);
    if (conn == NULL || conn->err) {
        if (conn) {
            printf("Connection error: %s\n", conn->errstr);
            redisFree(conn);
        } else {
            printf("Connection error: can't allocate redis context\n");
        }
    }
    exit(1);
}
/* AUTH */
reply = redisCommand(conn, "AUTH %s", password);
printf("AUTH: %s\n", reply->str);
freeReplyObject(reply);

/* Set */
reply = redisCommand(conn, "SET %s %s", "welcome", "Hello, DCS for Redis!");
printf("SET: %s\n", reply->str);
freeReplyObject(reply);

/* Get */
reply = redisCommand(conn, "GET welcome");
printf("GET welcome: %s\n", reply->str);
freeReplyObject(reply);

/* Disconnects and frees the context */
redisFree(conn);
return 0;
}
```

2. 执行以下命令进行编译。

```
gcc connRedis.c -o connRedis -I /usr/local/include/hiredis -lhiredis
```

如果有报错，可查找hiredis.h文件路径，并修改编译命令。

编译完后得到一个可执行文件connRedis。

3. 执行以下命令，连接Redis实例。

```
./connRedis {redis_instance_address} 6379 {password}
```

其中，*{redis_instance_address}*为Redis实例的IP地址/域名，“6379”为Redis实例的端口。IP地址/域名和端口获取见**步骤1**，请按实际情况修改后执行。

*{password}*为创建Redis实例时自定义的密码，请按实际情况修改后执行。

返回以下回显信息，表示成功连接Redis实例。

```
AUTH: OK
SET: OK
GET welcome: Hello, DCS for Redis!
```

须知

如果运行报错找不到hiredis库文件，可参考如下命令，将相关文件复制到系统目录，并增加动态链接。

```
mkdir /usr/lib/hiredis
cp /usr/local/lib/libhiredis.so.0.13 /usr/lib/hiredis/
mkdir /usr/include/hiredis
cp /usr/local/include/hiredis/hiredis.h /usr/include/hiredis/
echo '/usr/local/lib' >> /etc/ld.so.conf
ldconfig
```

以上so文件与.h文件的位置，需要替换成实际文件位置。

---结束

4.3.8 StackExchange.Redis 客户端连接 Redis (C#)

本章节介绍使用StackExchange.Redis客户端连接Redis实例的方法。更多的客户端的使用方法请参考[Redis客户端](#)。

以下操作以通过弹性云服务器上的客户端连接Redis实例为例进行说明。

说明

使用StackExchange客户端连接Proxy集群实例时，目前无法使用Proxy集群的多DB功能。

前提条件

- 已成功创建Redis实例，且状态为“运行中”。创建Redis实例的操作请参考[购买Redis实例](#)。
- 已创建弹性云服务器，创建弹性云服务器的方法，请参见[购买弹性云服务器](#)。
- 如果弹性云服务器为Linux系统，该弹性云服务器必须已经安装gcc编译环境。
- 连接实例前确保客户端与Redis实例之间网络互通，具体请参考[连接Redis网络要求](#)。

StackExchange.Redis 客户端连接 Redis

步骤1 查看并获取待连接Redis实例的IP地址/域名和端口。

具体步骤请参见[查看和修改DCS实例基本信息](#)。

步骤2 登录弹性云服务器。

弹性云服务器操作系统，这里以Window为例。

步骤3 在弹性云服务器安装VS 2017社区版。

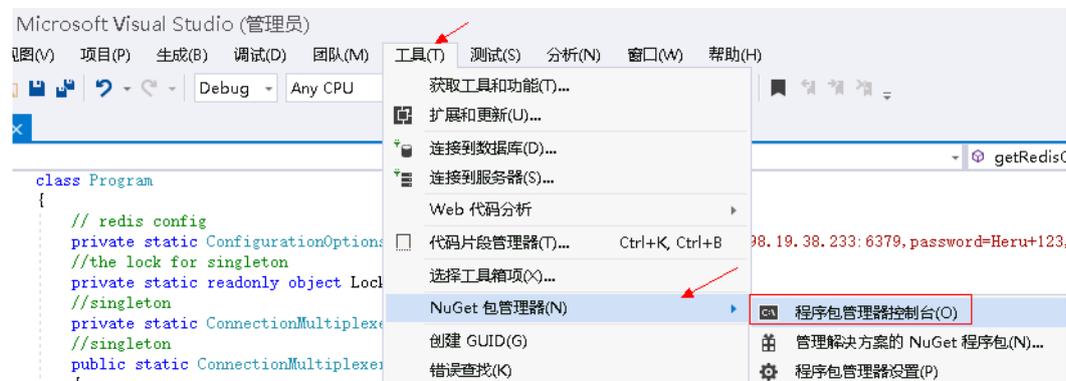
步骤4 启动VS 2017，新建一个工程。

工程名自定义，这里设置为“redisdemo”。

步骤5 使用VS的nuget管理工具安装C# Redis客户端StackExchange.Redis。

按照如图4-11操作，进入程序包管理器控制台，在nuget控制台输入：**Install-Package StackExchange.Redis -Version 2.2.79**。（版本号可以不指定）

图 4-11 进入程序包管理器控制台



步骤6 编写如下代码，并使用String的set和get测试连接。

```
using System;
using StackExchange.Redis;

namespace redisdemo
{
    class Program
    {
        // redis config
        private static ConfigurationOptions connDCS = ConfigurationOptions.Parse("{instance_ip_address}:
{port},password=*****,connectTimeout=2000");
        //the lock for singleton
        private static readonly object Locker = new object();
        //singleton
        private static ConnectionMultiplexer redisConn;
        //singleton
        public static ConnectionMultiplexer getRedisConn()
        {
            if (redisConn == null)
            {
                lock (Locker)
                {
                    if (redisConn == null || !redisConn.IsConnected)
                    {
                        redisConn = ConnectionMultiplexer.Connect(connDCS);
                    }
                }
            }
            return redisConn;
        }
        static void Main(string[] args)
        {
            redisConn = getRedisConn();
            var db = redisConn.GetDatabase();
            //set get
            string strKey = "Hello";
            string strValue = "DCS for Redis!";
            Console.WriteLine( strKey + ", " + db.StringGet(strKey));

            Console.ReadLine();
        }
    }
}
```

其中，{instance_ip_address}和{port}分别为Redis实例的IP地址/域名以及端口。IP地址/域名和端口获取见步骤1，请按实际情况修改后执行。*****为创建Redis实例时自定义的密码，请按实际情况修改后执行。

步骤7 运行代码，控制台界面输出如下，表示连接成功。

```
Hello, DCS for Redis!
```

关于客户端的其他命令，可以参考[StackExchange.Redis](#)。

----结束

4.3.9 Phpreid 客户端连接 Redis (PHP)

本章节介绍使用phpredis客户端连接Redis的方法。更多的客户端的使用方法请参考[Redis客户端](#)。

以下操作以通过弹性云服务器上的客户端连接Redis实例为例进行说明。

说明

本章节操作，仅适用于连接单机、主备、Proxy集群实例，如果是使用phpredis客户端连接Cluster集群，请参考[phpredis客户端使用说明](#)。

前提条件

- 已成功创建Redis实例，且状态为“运行中”。创建Redis实例的操作请参考[购买Redis实例](#)。
- 已创建弹性云服务器，创建弹性云服务器的方法，请参见[购买弹性云服务器](#)。
- 如果弹性云服务器为Linux系统，该弹性云服务器必须已经安装gcc编译环境。
- 连接实例前确保客户端与Redis实例之间网络互通，具体请参考[连接Redis网络要求](#)。

Phpreid 客户端连接 Redis

步骤1 查看并获取待连接Redis实例的IP地址/域名和端口。

具体步骤请参见[查看和修改DCS实例基本信息](#)。

步骤2 登录弹性云服务器。

本章节以弹性云服务器操作系统为centos为例介绍通过phpredis redis客户端连接实例。

步骤3 安装gcc-c++及make等编译组件。

```
yum install gcc-c++ make
```

步骤4 安装php开发包与命令行工具。

执行如下命令，使用yum方式直接安装。

```
yum install php-devel php-common php-cli
```

安装完后可查看版本号，确认成功安装：

```
php --version
```

步骤5 安装php redis客户端。

1. 下载php redis源文件。

```
wget http://pecl.php.net/get/redis-5.3.7.tgz
```

仅以该版本作为示例，您还可以去redis官网或者php官网下载其他版本的phpredis客户端。

2. 解压php redis源文件包。

```
tar -zxvf redis-5.3.7.tgz
```

```
cd redis-5.3.7
```

3. 编译前先执行扩展命令。

```
phpize
```

4. 配置php-config文件。

```
./configure --with-php-config=/usr/bin/php-config
```

不同操作系统，不同的php安装方式，该文件位置不一样。建议在配置前，先查找和确认该文件的目录：

```
find / -name php-config
```

5. 编译和安装php redis客户端。

```
make && make install
```

6. 安装完后在php.ini文件中增加extension配置项，用于增加redis模块的引用配置。

```
vim /etc/php.ini
```

增加如下配置项：

```
extension = "/usr/lib64/php/modules/redis.so"
```

说明

php.ini和redis.so两个文件的目录可能不同，需要先查找确认。

例如：**find / -name php.ini**

7. 保存退出后确认扩展生效。

```
php -m |grep redis
```

如果以上命令返回了redis，表示php redis客户端环境搭建好了。

步骤6 使用php redis客户端连接Redis实例。

1. 编辑一个redis.php文件：

```
<?php
    $redis_host = "{redis_instance_address}";
    $redis_port = {port};
    $user_pwd = "{password}";
    $redis = new Redis();
    if ($redis->connect($redis_host, $redis_port) == false) {
        die($redis->getLastError());
    }
    if ($redis->auth($user_pwd) == false) {
        die($redis->getLastError());
    }
    if ($redis->set("welcome", "Hello, DCS for Redis!") == false) {
        die($redis->getLastError());
    }
    $value = $redis->get("welcome");
    echo $value;
    $redis->close();
?>
```

其中，{redis_instance_address}为Redis实例的IP地址/域名，{port}为Redis实例的端口。IP地址/域名和端口获取见[步骤1](#)，请按实际情况修改后执行。{password}为创建Redis实例时自定义的密码，请按实际情况修改后执行。如果免密访问，请将密码认证的if语句屏蔽。

2. 执行php redis.php，连接Redis实例。

----结束

4.3.10 Predis 客户端连接 Redis (PHP)

本章节介绍使用Predis客户端连接Redis的方法。更多的客户端的使用方法请参考[Redis 客户端](#)。

以下操作以通过弹性云服务器上的客户端连接Redis实例为例进行说明。

前提条件

- 已成功创建Redis实例，且状态为“运行中”。创建Redis实例的操作请参考[购买Redis实例](#)。
- 已创建弹性云服务器，创建弹性云服务器的方法，请参见[购买弹性云服务器](#)。
- 如果弹性云服务器为Linux系统，该弹性云服务器必须已经安装php编译环境。
- 连接实例前确保客户端与Redis实例之间网络互通，具体请参考[连接Redis网络要求](#)。

Predis 客户端连接 Redis

步骤1 查看并获取待连接Redis实例的IP地址/域名和端口。

具体步骤请参见[查看和修改DCS实例基本信息](#)。

步骤2 登录弹性云服务器。

步骤3 安装php开发包与命令行工具。执行如下命令，使用yum方式直接安装。

```
yum install php-devel php-common php-cli
```

步骤4 安装完后可查看版本号，确认成功安装。

```
php --version
```

步骤5 将Predis包下载到/usr/share/php目录下。

1. 通过以下命令下载Predis源文件。

```
wget https://github.com/predis/predis/archive/refs/tags/v2.2.2.tar.gz
```

说明

仅以该版本作为示例，您还可以去redis官网或者php官网下载其他版本的predis客户端。

2. 解压Predis源文件包。

```
tar -zxvf predis-2.2.2.tar.gz
```

3. 将解压好的predis目录重命名为“predis”，并移动到/usr/share/php/下。

```
mv predis-2.2.2 predis
```

步骤6 编辑一个文件连接redis。

- 使用redis.php文件连接Redis单机/主备/Proxy集群示例：

```
<?php
require 'predis/autoload.php';
Predis\Autoloader::register();
$client = new Predis\Client([
    'scheme' => 'tcp',
    'host' => '{redis_instance_address}',
    'port' => {port},
    'password' => '{password}'
]);
```

```
$client->set('foo', 'bar');  
$value = $client->get('foo');  
echo $value;  
?>
```

- 使用redis-cluster.php连接Redis Cluster集群代码示例:

```
<?php  
require 'premis/autoload.php';  
$servers = array(  
    'tcp://{redis_instance_address}:{port}'  
);  
$options = array('cluster' => 'redis');  
$client = new Premis\Client($servers, $options);  
$client->set('foo', 'bar');  
$value = $client->get('foo');  
echo $value;  
?>
```

其中, {redis_instance_address}为Redis实例真实的IP地址/域名, {port}为Redis实例真实的端口。IP地址/域名和端口获取见[步骤1](#), 请按实际情况修改后执行。{password}为创建Redis实例时自定义的密码, 请按实际情况修改后执行。如果免密访问, 请将password行去掉。

步骤7 执行php redis.php连接Redis实例。

----结束

4.3.11 Ioredis 客户端连接 Redis (Node.js)

本章节介绍使用ioRedis客户端连接Redis实例的方法。更多的客户端的使用方法请参考[Redis客户端](#)。

以下操作以通过弹性云服务器上的客户端连接Redis实例为例进行说明。

说明

本章节操作, 仅适用于连接单机、主备、Proxy集群实例, 如果是使用ioRedis客户端连接Cluster集群, 请参考[NodeJs Redis客户端使用](#)。

前提条件

- 已成功创建Redis实例, 且状态为“运行中”。创建Redis实例的操作请参考[购买Redis实例](#)。
- 已创建弹性云服务器, 创建弹性云服务器的方法, 请参见[购买弹性云服务器](#)。
- 如果弹性云服务器为Linux系统, 该弹性云服务器必须已经安装gcc编译环境。
- 连接实例前确保客户端与Redis实例之间网络互通, 具体请参考[连接Redis网络要求](#)。

Ioredis 客户端连接 Redis

- **客户端服务器为Ubuntu(debian系列)**

步骤1 查看并获取待连接Redis实例的IP地址/域名和端口。

具体步骤请参见[查看和修改DCS实例基本信息](#)。

步骤2 登录弹性云服务器。

步骤3 安装Node.js。

```
apt install nodejs-legacy
```

如果以上命令安装不了，备选方式如下：

```
wget https://nodejs.org/dist/v0.12.4/node-v0.12.4.tar.gz --no-check-certificate
tar -xvf node-v4.28.5.tar.gz
cd node-v4.28.5
./configure
make
make install
```

📖 说明

安装完成后，可执行 `node --version` 查看 Node.js 的版本号，确认 Node.js 已安装成功。

步骤4 安装js包管理工具npm。

```
apt install npm
```

步骤5 安装NodeJs redis客户端ioredis。

```
npm install ioredis
```

步骤6 编辑连接Redis实例的示例脚本。

编辑连接示例脚本 `ioredisdemo.js`。示例脚本中增加以下内容，包括连接以及数据读取。

```
var Redis = require('ioredis');
var redis = new Redis({
  port: 6379, // Redis port
  host: '192.168.0.196', // Redis host
  family: 4, // 4 (IPv4) or 6 (IPv6)
  password: '*****',
  db: 0
});
redis.set('foo', 'bar');
redis.get('foo', function (err, result) {
  console.log(result);
});
// Or using a promise if the last argument isn't a function
redis.get('foo').then(function (result) {
  console.log(result);
});
// Arguments to commands are flattened, so the following are the same:
redis.sadd('set', 1, 3, 5, 7);
redis.sadd('set', [1, 3, 5, 7]);
// All arguments are passed directly to the redis server:
redis.set('key', 100, 'EX', 10);
```

其中，**host**为Redis实例的IP地址/域名，**port**为Redis实例的端口。IP地址/域名和端口获取见[步骤1](#)，请按实际情况修改后执行。*********为创建Redis实例时自定义的密码，请按实际情况修改后执行。

步骤7 运行示例脚本，连接Redis实例。

```
node ioredisdemo.js
```

----结束

● 客户端服务器为centos(redhat系列)

步骤1 查看并获取待连接Redis实例的IP地址/域名和端口。

具体步骤请参见[查看和修改DCS实例基本信息](#)。

步骤2 登录弹性云服务器。

步骤3 安装Node.js。

```
yum install nodejs
```

如果以上命令安装不了，备选方式如下：

```
wget https://nodejs.org/dist/v0.12.4/node-v0.12.4.tar.gz --no-check-certificate
tar -xvf node-v0.12.4.tar.gz
cd node-v0.12.4
./configure
make
make install
```

📖 说明

安装完成后，可执行**node -v**查看Node.js的版本号，确认Node.js已安装成功。

步骤4 安装js包管理工具npm。

```
yum install npm
```

步骤5 安装Node.js redis客户端ioredis。

```
npm install ioredis
```

步骤6 编辑连接Redis实例的示例脚本。

编辑连接示例脚本ioredisdemo.js。示例脚本中增加以下内容，包括连接以及数据读取。

```
var Redis = require('ioredis');
var redis = new Redis({
  port: 6379, // Redis port
  host: '192.168.0.196', // Redis host
  family: 4, // 4 (IPv4) or 6 (IPv6)
  password: '*****',
  db: 0
});
redis.set('foo', 'bar');
redis.get('foo', function (err, result) {
  console.log(result);
});
// Or using a promise if the last argument isn't a function
redis.get('foo').then(function (result) {
  console.log(result);
});
// Arguments to commands are flattened, so the following are the same:
redis.sadd('set', 1, 3, 5, 7);
redis.sadd('set', [1, 3, 5, 7]);
// All arguments are passed directly to the redis server:
redis.set('key', 100, 'EX', 10);
```

其中，**host**为Redis实例的IP地址/域名，**port**为Redis实例的端口。IP地址/域名和端口获取见[步骤1](#)，请按实际情况修改后执行。*********为创建Redis实例时自定义的密码，请按实际情况修改后执行。

步骤7 运行示例脚本，连接Redis实例。

```
node ioredisdemo.js
```

----结束

4.4 控制台连接 Redis

DCS支持通过管理控制台的Web CLI功能连接Redis实例。

说明

- 请勿通过Web CLI输入敏感信息，以免敏感信息泄露。
- 当前在Web CLI下所有命令参数暂不支持中文且key和value不支持空格。
- 当value值为空时，执行get命令返回nil。
- 在Web CLI中，部分命令被禁用，详情请参考[Web CLI中支持和禁用命令](#)。
- 只有Redis 4.0、Redis 5.0和Redis 6.0实例支持该操作，Redis 3.0不支持该功能。

前提条件

只有当实例处于“运行中”状态，才能执行此操作。

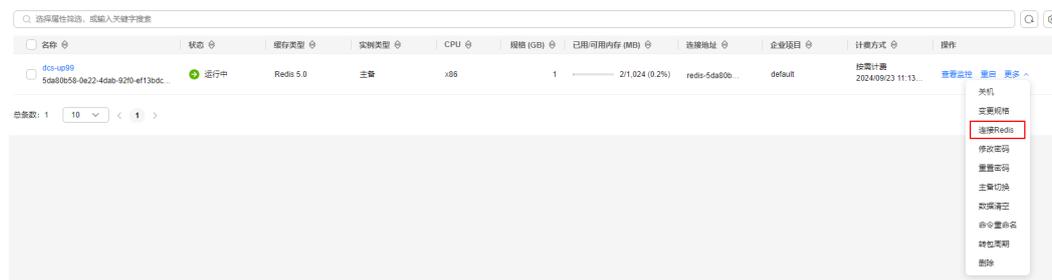
控制台连接 Redis

步骤1 登录[分布式缓存服务管理控制台](#)。

步骤2 在管理控制台左上角单击，选择实例所在的区域。

步骤3 单击左侧菜单栏的“缓存管理”，然后单击“操作”列下的“更多 > 连接Redis”，进入Web CLI登录界面，如下图所示。

图 4-12 进入 Web CLI 登录界面



步骤4 输入实例的访问密码进入Web CLI，然后选择当前操作的Redis数据库，在命令输入框输入Redis命令，按Enter键执行。

说明

- 控制台连接实例空闲超过5分钟会连接超时，再次登录需要重新输入访问密码。
- 免密访问的Redis实例无需输入密码。

---结束

4.5 公网连接 Redis 3.0（Redis 3.0 已停售）

4.5.1 开启 Redis 3.0 实例的公网访问

如果实例已经开启了公网访问，不需要执行本章节。

如果实例没有开启公网访问，可参考本章节，开启公网访问开关，在开启公网访问时，您可以选择是否使用SSL加密传输。

📖 说明

- 通过公网SSL加密方式访问Redis实例时，建议使用前下载CA证书，并使用CA证书来验证DCS缓存实例的证书，以提高安全性。
- 通过公网直接（未开启SSL加密）访问实例时，请直接访问Redis实例的弹性公网IP与6379端口，不需下载证书，也不需要客户端安装Stunnel工具。
- 推荐使用SSL加密Redis客户端与DCS实例之间的传输通道，确保数据传输安全。

前提条件

- 缓存类型：必须为Redis 3.0版本。如果不是，则不支持开启公网访问。
- 访问方式：必须为密码访问。如果是免密方式，请先参考[重置缓存实例密码](#)修改为密码访问方式。

开启 Redis 实例的公网访问

步骤1 登录[分布式缓存服务管理控制台](#)。

步骤2 在管理控制台左上角单击 ，选择实例所在的区域。

步骤3 单击左侧菜单栏的“缓存管理”。

步骤4 单击对应Redis实例的名称，进入该实例的基本信息页面。

步骤5 单击“公网访问”右侧的 ，弹出“修改公网访问”页面。

步骤6 单击 ，打开公网访问开关。

步骤7 从“弹性IP地址”下拉列表中选择一个弹性IP。

如果没有弹性IP，可单击右侧的“查看弹性IP”，系统会跳转到网络控制台的弹性公网IP页面，创建弹性公网IP。创建完之后，单击“弹性IP地址”右侧的刷新按钮，即可选择新建的公网IP。

步骤8 （可选）根据需要选择是否开启SSL加密功能。

建议使用SSL加密Redis客户端与DCS实例之间的传输通道，确保数据传输安全。

步骤9 单击“确定”，开启公网访问功能。

开启公网访问功能大约需要1~2分钟，请耐心等待。

当前页面会自动跳转到“后台任务”页签，并显示当前任务的操作进度。任务状态为“成功”，表示开启公网访问成功。

----结束

4.5.2 Redis-cli 客户端公网连接 Redis 3.0

本文介绍使用redis-cli客户端通过公网连接Redis 3.0实例的具体操作。

公网访问功能便于开发人员在本地搭建开发或测试环境，提高开发效率。在生产环境（正式环境）中，请通过VPC内连接方式访问Redis实例，保障访问效率。

前提条件

使用redis-cli客户端通过公网访问Redis实例时：

- 实例必须为Redis 3.0实例，且已经开启了公网访问功能。
- 如果访问Redis实例需要使用证书，可进入到缓存实例详情页面提前下载该证书。

公网连接 Redis 3.0（Linux 环境，开启 SSL 加密时）

步骤1 确认Redis实例的安全组入方向规则是否配置正确，即是否允许外部地址访问36379端口。

当SSL加密功能开启时，必须允许36379端口被外部地址访问。需要安装Stunnel客户端，然后访问Redis的公网地址。

图 4-13 安全组规则（端口配置为 36379）



步骤2 获取待连接Redis实例的公网访问地址、证书。

- 公网访问地址：单击实例，进入实例详情页即可查看，如下所示。
- 证书：单击“证书下载”，下载解压之后，获得dcs-ca.cer和dcs-ca-bundle.pem文件，**dcs-ca.cer**为公网访问时需要的二进制格式公钥证书，dcs-ca-bundle.pem为文本格式证书文件。

图 4-14 查看公网访问地址（开启 SSL，端口为 36379）



步骤3 登录本地Linux设备。

步骤4 安装Stunnel客户端。

这里主要介绍Stunnel客户端的几种常见安装方法，您可以选择其中一种方式进行操作。

📖 说明

推荐使用apt和yum两种安装方式，常见Linux系统，一般至少支持其中一种。

- apt-get方式安装。

apt-get管理deb格式的软件包，适用于Debian类操作系统，如Ubuntu。命令如下：

apt install stunnel或apt-get install stunnel

如果命令执行后提示找不到Stunnel，可以尝试执行**apt update**，更新配置后再安装Stunnel。

- yum方式安装。

管理rpm格式的软件包，适用于Fedora、CentOS、Red Hat等操作系统。命令如下：

yum install stunnel

步骤5 打开Stunnel配置文件stunnel.conf。

- 如果是apt-get安装方式，默认路径为/etc/stunnel/stunnel.conf。
如果路径不存在或者路径下无配置文件，可新增。
- 如果是yum安装方式，默认路径为/usr/local/stunnel/stunnel.conf。
如果路径不存在或者路径下无配置文件，可新增。

📖 说明

- 如果不确定配置文件应该存储在哪，可以在安装后直接输入stunnel命令，获取文件路径提示。
- 配置文件可以存储在任何路径，在Stunnel启动的时候指定该配置文件即可。

步骤6 在配置文件stunnel.conf中新增如下内容，然后保存退出。

```
debug = 4
output = /var/log/stunnel.log
sslVersion = all
[redis-client]
client = yes
accept = 8000
connect = {公网访问地址}
CAfile = /etc/stunnel/dcs-ca.cer
```

参数需要根据以下说明修改，其他参数不用修改：

- **client**值固定填**yes**，表示为Stunnel客户端。
- **CAfile**为CA证书，为可选参数。如果需要，根据**步骤2**的操作，使用下载之后解压得到dcs-ca.cer证书；如果不需要，可不配置，删除此参数。
- **accept**为Stunnel监听端口，可以自定义。Redis客户端访问缓存实例时填写此端口。
- **connect**为Stunnel转发地址与端口，此处填Redis缓存实例的弹性IP与端口，替换为**步骤2**获取的公网访问地址即可。

配置示例如下：

```
[redis-client]
client = yes
CAfile = D:\tmp\dc\dc-ca.cer
accept = 8000
connect = 49.**.**.211:36379
```

步骤7 执行以下命令，启动stunnel服务。

```
stunnel /{customdir}/stunnel.conf
```

其中{customdir}为**步骤5**中stunnel.conf配置文件的存储路径。命令示例如下：

```
stunnel /etc/stunnel/stunnel.conf
```

说明

Ubuntu环境下，启动命令可以使用/etc/init.d/stunnel4 start。Stunnel4.x的版本，服务/进程名为stunnel4。

启动后可执行ps -ef|grep stunnel确认进程是否正常运行。

步骤8 执行以下命令，查看Stunnel是否已经被监听。

```
netstat -plunt |grep 8000|grep "LISTEN"
```

其中，8000替换为**步骤6**中accept字段配置的Stunnel监听端口。

返回信息有包含8000的端口的记录行，表示stunnel客户端正常运行。Redis客户端连接“127.0.0.1:8000”，Stunnel会将请求转发给DCS的Redis实例。

步骤9 连接Redis实例。

1. 登录本地Linux设备。
2. 获取Redis客户端源码，下载路径为：<http://download.redis.io/releases/redis-5.0.8.tar.gz>。

```
wget http://download.redis.io/releases/redis-5.0.8.tar.gz
```

说明

您也可以使用yum、apt方式安装Redis客户端。

- yum方式，执行命令：**yum install redis**
 - apt方式，执行命令：**apt install redis-server**
3. 执行如下命令，解压Redis客户端源码包。

```
tar -xzf redis-5.0.8.tar.gz
```
 4. 进入Redis目录并编译Redis客户端源码。

```
cd redis-5.0.8
make
```
 5. 执行以下命令连接Redis实例。

```
cd src
./redis-cli -h 127.0.0.1 -p 8000
```

注意

连接命令中-h后的连接地址应该为Stunnel客户端地址，-p后的端口为Stunnel客户端监控端口，不要使用控制台展示的公网连接地址和端口，连接地址保持127.0.0.1即可；连接端口为步骤6中accept字段配置的Stunnel监听端口，本文示例定义的是8000。

6. 输入密码，校验通过后才可进行缓存数据读写。

auth {password}

其中，{password}为创建Redis实例时自定义的密码，请按实际情况修改后执行。连接成功后，回显信息如下：

```
OK
127.0.0.1:8000>
```

----结束

公网连接 Redis 3.0 (Linux 环境, 关闭 SSL 加密时)

- 步骤1 确认Redis实例的安全组入方向规则是否配置正确，即是否允许外部地址访问6379端口。

当SSL加密功能关闭时，必须允许6379端口被外部地址访问。放开后，即可直接访问Redis的公网地址。

图 4-15 安全组规则 (端口配置为 6379)



- 步骤2 查看并获取待连接Redis实例的公网访问地址。

单击实例，进入实例详情页即可查看，如下所示。

图 4-16 查看公网访问地址 (关闭 SSL, 端口为 6379)



- 步骤3 登录本地Linux设备。

- 步骤4 获取Redis客户端源码，下载路径为http://download.redis.io/releases/redis-5.0.8.tar.gz。

wget http://download.redis.io/releases/redis-5.0.8.tar.gz

📖 说明

您也可以使用yum、apt方式安装Redis客户端。

- yum方式，执行命令：**yum install redis**
- apt方式，执行命令：**apt install redis-server**

步骤5 执行如下命令，解压Redis客户端源码包。

```
tar -xzf redis-5.0.8.tar.gz
```

步骤6 进入Redis目录并编译Redis客户端源码。

```
cd redis-5.0.8
```

```
make
```

步骤7 执行以下命令连接Redis实例。

```
cd src
```

```
./redis-cli -h {公网访问IP} -p 6379
```

其中，{公网访问IP}替换为**步骤2**中获取的Redis实例的IP即可。示例如下：

```
./redis-cli -h 49.**.**.211 -p 6379
```

步骤8 输入密码，校验通过后才可进行缓存数据读写。

```
auth {password}
```

其中，**{password}**为创建Redis实例时自定义的密码，请按实际情况修改后执行。

连接成功后，回显信息如下：

```
OK
49.**.**.211:6379>
```

----结束

公网连接 Redis 3.0（Windows 环境，开启实例 SSL 加密时）

步骤1 确认Redis实例的安全组入方向规则是否配置正确，即是否允许外部地址访问36379端口。

当SSL加密功能开启时，必须允许36379端口被外部地址访问。放开后，需要安装Stunnel客户端，然后访问Redis的公网地址。

图 4-17 安全组规则（端口配置为 36379）



步骤2 获取待连接Redis实例的公网访问地址、证书。

- 公网访问地址：单击实例，进入实例详情页即可查看，如下所示。
- 证书：单击“证书下载”，下载解压之后，获得dcs-ca.cer和dcs-ca-bundle.pem文件，**dcs-ca.cer**为公网访问时需要的二进制格式公钥证书，dcs-ca-bundle.pem为文本格式证书文件。

图 4-18 查看公网访问地址（开启 SSL，端口为 36379）



步骤3 下载Stunnel安装包。从<https://www.stunnel.org/downloads.html>下载最新版本的Windows版Stunnel安装包（以exe结尾的安装包，例如，stunnel-5.44-win32-installer.exe）到本地Windows设备。

步骤4 运行Stunnel安装程序，安装Stunnel客户端。

步骤5 配置Stunnel客户端。在任务栏单击图标右键，选择“Edit Configuration”，新增如下配置内容，然后保存退出。

```
[redis-client]
client = yes
CAfile = D:\tmp\dcs\dcs-ca.cer
accept = 8000
connect = {公网访问地址}
```

参数需要根据以下说明修改，其他参数不用修改：

- **client**值固定填yes，表示为Stunnel客户端。
- **CAfile**为CA证书，为可选参数。如果需要，根据步骤2的操作，使用下载之后解压得到dcs-ca.cer证书；如果不需要，可不配置，删除此参数。

- **accept**为Stunnel监听端口，可以自定义。Redis客户端访问实例时填写此端口。
- **connect**为服务端连接地址，此处填Redis实例的弹性公网IP与端口，替换为[步骤2](#)获取的公网访问地址即可。

已开启了SSL加密功能的配置为例，如下所示：

```
[redis-client]
client = yes
CAfile = D:\tmp\dcs\dcs-ca.cer
accept = 8000
connect = 49.**.**.211:36379
```

步骤6 在任务栏单击图标右键，选择“Reload Configuration”。

步骤7 打开命令提示符工具cmd.exe，执行以下命令，查看127.0.0.1:8000是否已经被监听。

```
netstat -an |find "8000"
```

假设客户端的监听端口配置为“8000”。

返回列表行中显示有“127.0.0.1:8000”，状态为“LISTENING”，表示stunnel客户端正常运行。Redis客户端连接“127.0.0.1:8000”，stunnel会将请求转发给DCS的Redis实例。

步骤8 连接Redis实例。

1. 获取Redis客户端安装包到本地Windows设备，并解压安装包。
Windows版本的Redis客户端安装包，下载请单击[这里](#)。
2. 打开命令提示符工具cmd.exe，并执行以下命令，进入Redis客户端安装包的解压目录。

以解压目录D:\redis-64.3.0.503为例，命令如下：

D:

```
cd D:\redis-64.3.0.503
```

3. 执行以下命令连接Redis实例。

```
redis-cli -h 127.0.0.1 -p 8000 -a <password>
```

注意

连接命令中-h后的连接地址应该为Stunnel客户端地址，-p后的端口为Stunnel客户端监听端口，不要使用控制台展示的公网连接地址和端口，连接地址保持127.0.0.1即可；连接端口为[步骤5](#)中accept字段配置的Stunnel监听端口，本文示例定义的是8000。

<password>为创建Redis实例时自定义的密码，请按实际情况修改后执行。

连接成功后，回显信息如下：

```
127.0.0.1:8000>
```

输入“info”可正常返回Redis实例信息。如果不能正常返回，或者连接异常断开，可在任务栏找到Stunnel图标，并右键单击，选择“Show Log Window”，打开Stunnel客户端的日志查看原因。

----结束

公网连接 Redis 3.0（Windows 环境，关闭实例 SSL 加密时）

步骤1 确认Redis实例的安全组入方向规则是否配置正确，即是否允许外部地址访问6379端口。

当SSL加密功能关闭时，必须允许6379端口被外部地址访问。放开后，即可直接访问Redis的公网地址。

图 4-19 安全组规则（端口配置为 6379）



步骤2 查看并获取待连接Redis实例的公网访问地址。

单击实例，进入实例详情页即可查看，如下所示。

图 4-20 查看公网访问地址（关闭 SSL，端口为 6379）



步骤3 获取Redis客户端安装包到本地Windows设备，并解压安装包。

Windows版本的Redis客户端安装包，下载请单击[这里](#)。

步骤4 打开命令提示符工具cmd.exe，并执行以下命令，进入Redis客户端安装包的解压目录。

以解压目录D:\redis-64.3.0.503为例，命令如下：

D:

```
cd D:\redis-64.3.0.503
```

步骤5 执行以下命令连接Redis实例。

```
redis-cli -h {公网访问IP} -p 6379 -a {password}
```

其中，{公网访问IP}替换为**步骤2**中获取的Redis实例的IP，{password}为创建Redis实例时自定义的密码，请按实际情况修改后执行。

连接成功后，回显信息如下：

```
139.**.**.175:6379>
```

输入“info”可正常返回Redis实例信息。

----结束

故障排除

- Error: Connection reset by peer
原因：安全组没有配置正确，需要参考[放通36379端口](#)或[放通6379端口](#)。
- 使用redis-cli连接实例，会出现：远程主机强迫关闭一个现有的连接。
原因：开启了SSL加密传输，连接时没有配置Stunnel，直接使用界面提示的IP地址进行连接。开启SSL加密时，您需要按照[公网连接Redis 3.0（Linux环境，开启SSL加密时）](#)的操作步骤执行。
- 更多连接失败的问题，请查看[Redis实例连接失败的原因排查](#)。

5 连接 Memcached 实例（已停售）

5.1 配置 Memcached 访问密码

使用场景

Memcached实例的访问方式支持免密访问和密码访问两种模式，在实例创建之后支持修改，主要使用场景如下：

- 对于密码访问模式的Memcached实例，当您需要通过免用户名和密码访问模式连接Memcached实例，可通过开启Memcached实例的免密访问功能，清空Memcached实例的用户名和密码。
Memcached文本协议不支持用户名密码认证，若需要使用文本协议连接Memcached实例，必须开启实例的免密访问。
- 对于免密访问模式的Memcached实例，当您需要通过用户名和密码访问模式连接时，您可以通过重置密码，修改为密码访问模式。

配置 Memcached 访问密码

步骤1 登录[分布式缓存服务管理控制台](#)。

步骤2 在管理控制台左上角单击 ，选择实例所在的区域。

步骤3 单击左侧菜单栏的“缓存管理”。

步骤4 在需要修改访问方式的Memcached实例右侧，单击“操作”列下的“更多 > 重置密码”。

步骤5 系统弹出“重置密码”对话框，请根据实际情况选择以下操作。

- 如果是密码模式修改为免密模式
打开“免密访问”开关，并单击“确定”，完成免密访问设置。
- 如果是免密模式修改为密码访问模式
在弹出的“重置密码”对话框，输入“新密码”和“确认密码”，并单击“确定”，完成密码设置。

----结束

5.2 使用客户端连接 Memcached

5.2.1 Telnet 客户端连接 Memcached

介绍使用同一VPC内弹性云服务器ECS上的Telnet客户端连接Memcached实例的方法。

前提条件

- 已成功申请Memcached实例，且状态为“运行中”。
- 已创建弹性云服务器，并已安装好客户端。创建弹性云服务器的方法，请参见《弹性云服务器用户指南》。

📖 说明

您创建的弹性云服务器必须与Memcached实例属于同一个VPC，并配置相同的安全组，以确保弹性云服务器与缓存实例的网络是连通的。

- 如果弹性云服务器与Memcached实例不在相同VPC中，可以通过建立VPC对等连接方式连通网络，具体请参考常见问题：[缓存实例是否支持跨VPC访问？](#)
- 如果弹性云服务器与Memcached实例配置了不同的安全组，可以通过设置安全组规则连通网络，具体请参考常见问题：[如何选择和配置安全组？](#)
- 建议在使用本手册时删除示例代码中的所有注释信息。
- 请确保所有命令行、代码块输入格式都是UTF-8，否则会出现编译出错或者运行失败的情况。

Telnet 客户端连接 Memcached

步骤1 登录[分布式缓存服务管理控制台](#)。

步骤2 在管理控制台左上角单击 ，选择实例所在的区域。

步骤3 单击左侧菜单栏的“缓存管理”，进入缓存实例信息页面。

步骤4 单击需要使用的其中一个Memcached实例的名称，进入该Memcached实例的基本信息页面。查看并获取该Memcached实例的IP地址/域名和端口。

步骤5 连接Memcached实例。

1. 登录已创建的弹性云服务器。
2. 执行如下命令，确认是否已安装Telnet客户端。

which telnet

若界面显示Telnet客户端所在目录，表示当前云服务器已安装Telnet客户端。否则请自行安装Telnet客户端。

📖 说明

- Linux系统中，如果没有安装Telnet客户端，可执行**yum -y install telnet**安装。
 - 在Windows系统中，可通过“控制面板 > 程序 > 打开或关闭Windows功能”，找到并打开“Telnet客户端”功能。
3. 执行如下命令，连接并使用Memcached实例。

```
telnet {ip or domain name} {port}
```

其中{ip or domain name} 为Memcached实例的IP地址/域名，{port}为Memcached实例的端口。IP地址/域名和端口获取方法请参考[步骤4](#)，请按实际情况修改后执行。

界面提示如下表示连接缓存实例成功。

```
Trying XXX.XXX.XXX.XXX...
Connected to XXX.XXX.XXX.XXX.
Escape character is '^'.
```

📖 说明

- 如果Memcached实例为**免密访问**模式，连接后可直接执行以下操作，输入命令。
- 如果Memcached实例为**密码访问**模式，连接后执行以下操作，会提示“ERROR authentication required”，表示没有权限，需要先执行**auth 用户名@密码**进行认证，其中，用户名和密码，表示设置连接Memcached实例的用户名和密码。

输入命令，示例如下（其中加粗部分内容为输入的命令，其他为命令返回内容）：

```
set hello 0 0 6
world!
STORED
get hello
VALUE hello 0 6
world!
END
```

----结束

5.2.2 Spymemcached 客户端连接 Memcached（Java）

介绍使用同一VPC内弹性云服务器ECS上的Java客户端连接Memcached实例的方法。

前提条件

- 已成功申请Memcached实例，且状态为“运行中”。
- 已创建弹性云服务器，并已安装好客户端。创建弹性云服务器的方法，请参见《弹性云服务器用户指南》。

📖 说明

您创建的弹性云服务器必须与Memcached实例属于同一个VPC，并配置相同的安全组，以确保弹性云服务器与缓存实例的网络是连通的。

- 如果弹性云服务器与Memcached实例不在相同VPC中，可以通过建立VPC对等连接方式连通网络，具体请参考常见问题：[缓存实例是否支持跨VPC访问？](#)
- 如果弹性云服务器与Memcached实例配置了不同的安全组，可以通过设置安全组规则连通网络，具体请参考常见问题：[如何选择和配置安全组？](#)
- 您的弹性云服务器已安装好Java JDK和常用的IDE（如Eclipse）。
- 已获取spymemcached-x.y.z.jar依赖包。

📖 说明

其中x.y.z为依赖包的版本号，建议获取最新版本。

Spymemcached 客户端连接 Memcached

步骤1 登录[分布式缓存服务管理控制台](#)。

- 步骤2** 在管理控制台左上角单击 ，选择实例所在的区域。
- 步骤3** 单击左侧菜单栏的“缓存管理”，进入缓存实例信息页面。
- 步骤4** 单击需要使用的其中一个Memcached实例的名称，进入该Memcached实例的基本信息页面。查看并获取该Memcached实例的IP地址/域名和端口。
- 步骤5** 将已获取的spymemcached-x.y.z.jar依赖包上传到已创建的弹性云服务器。
- 步骤6** 登录弹性云服务器。
- 步骤7** 在Eclipse中创建一个Java工程，并将spymemcached-x.y.z.jar依赖包导入，工程名可自定义。
- 步骤8** 新建一个ConnectMemcached1类，将如下Java代码复制到类中并修改代码。

- 密码模式代码示例

其中 *ip or domain name:port* 需要修改为 **步骤4** 获取的 Memcached 实例 IP 地址/域名和端口。*userName* 和 *password* 需要修改为 Memcached 实例的用户名和密码。

```
//java 连接加密的Memcached代码
import java.io.IOException;
import java.util.concurrent.ExecutionException;

import net.spy.memcached.AddrUtil;
import net.spy.memcached.ConnectionFactoryBuilder;
import net.spy.memcached.ConnectionFactoryBuilder.Protocol;
import net.spy.memcached.MemcachedClient;
import net.spy.memcached.auth.AuthDescriptor;
import net.spy.memcached.auth.PlainCallbackHandler;
import net.spy.memcached.internal.OperationFuture;

public class ConnectMemcached1
{
    public static void main(String[] args)
    {
        final String connectionaddress = "ip or domain name:port";
        final String username = "userName";//用户名
        final String password = "password";//密码
        MemcachedClient client = null;
        try
        {
            AuthDescriptor authDescriptor =
                new AuthDescriptor(new String[] {"PLAIN"}, new PlainCallbackHandler(username,
                    password));
            client = new MemcachedClient(
                new ConnectionFactoryBuilder().setProtocol(Protocol.BINARY)
                    .setAuthDescriptor(authDescriptor)
                    .build(),
                AddrUtil.getAddresses(connectionaddress));
            String key = "memcached";//向Memcached中存一个key为"memcached"的数据
            String value = "Hello World";//value为Hello World
            int expireTime = 5; // 过期时间，单位s; 从写入时刻开始计时，超过expireTime s后，该数据过期
            失效，无法再读出;
            doExcute(client, key, value, expireTime);//执行操作
        }
        catch (IOException e)
        {
            e.printStackTrace();
        }
    }

    /**
     *向Memcached写数据方法
     */
    private static void doExcute(MemcachedClient client, String key, String value, int expireTime)
    {
```

```
try
{
    OperationFuture<Boolean> future = client.set(key, expireTime, value);
    future.get();// spymemcached set()是异步的, future.get() 等待cache.set()操作结束, 也可以不
    等待, 用户根据自己需求选择;
    System.out.println("Set操作成功");
    System.out.println("Get操作:" + client.get(key));
    Thread.sleep(6000);//等待6000毫秒, 即6秒, 该数据将会过期失效, 无法再读出
    System.out.println("6秒后再执行Get操作:" + client.get(key));
}
catch (InterruptedException e)
{
    e.printStackTrace();
}
catch (ExecutionException e)
{
    e.printStackTrace();
}
if (client != null)
{
    client.shutdown();
}
}
```

- 免密模式代码示例

其中ip or domain name:port需要修改为[步骤4](#)获取的Memcached实例IP地址/域名和端口。

```
//java 连接免密的Memcached代码
import java.io.IOException;
import java.util.concurrent.ExecutionException;

import net.spy.memcached.AddrUtil;
import net.spy.memcached.BinaryConnectionFactory;
import net.spy.memcached.MemcachedClient;
import net.spy.memcached.internal.OperationFuture;

public class ConnectMemcached
{
    public static void main(String[] args)
    {
        final String connectionaddress = "ip or domain name:port";
        MemcachedClient client = null;
        try
        {
            client = new MemcachedClient(new BinaryConnectionFactory(),
            AddrUtil.getAddresses(connectionaddress));
            String key = "memcached";//向Memcached中存一个key为"memcached"的数据
            String value = "Hello World";//value为Hello World
            int expireTime = 5; // 过期时间, 单位; 从写入时刻开始计时, 超过 expireTime s后, 该数据过期
            失效, 无法再读出;
            doExcute(client, key, value, expireTime);//执行操作
        }
        catch (IOException e)
        {
            e.printStackTrace();
        }
    }

    /**
     *向Memcached写数据方法
     */
    private static void doExcute(MemcachedClient client, String key, String value, int expireTime)
    {
        try
        {
            OperationFuture<Boolean> future = client.set(key, expireTime, value);
            future.get();// spymemcached set()是异步的, future.get() 等待cache.set()操作结束, 也可以不
```

```
等待，用户根据自己需求选择；
System.out.println("Set操作成功");
System.out.println("Get操作:" + client.get(key));
Thread.sleep(6000);//等待6000毫秒，即6秒，该数据将会过期失效，无法再读出
System.out.println("6秒后再执行Get操作:" + client.get(key));

}
catch (InterruptedException e)
{
    e.printStackTrace();
}
catch (ExecutionException e)
{
    e.printStackTrace();
}
if (client != null)
{
    client.shutdown();
}
}
```

步骤9 运行main方法，在Eclipse下的Console窗口可以看到如下结果。

```
Set操作成功
Get操作:Hello World
6秒后再执行Get操作:null
```

----结束

5.2.3 Python-binary-memcached 客户端连接 Memcached (Python)

介绍使用同一VPC内弹性云服务器ECS上的Python客户端连接Memcached实例的方法。

前提条件

- 已成功申请Memcached实例，且状态为“运行中”。
- 已创建弹性云服务器。创建弹性云服务器的方法，请参见《弹性云服务器用户指南》。

说明

您创建的弹性云服务器必须与Memcached实例属于同一个VPC，并配置相同的安全组，以确保弹性云服务器与缓存实例的网络是连通的。

- 如果弹性云服务器与Memcached实例不在相同VPC中，可以通过建立VPC对等连接方式连通网络，具体请参考常见问题：[缓存实例是否支持跨VPC访问？](#)
- 如果弹性云服务器与Memcached实例配置了不同的安全组，可以通过设置安全组规则连通网络，具体请参考常见问题：[如何选择和配置安全组？](#)
- 弹性云服务器已安装好Python，建议为2.7.6或更高版本。
- 已获取[python-binary-memcached-x.y.z.zip](#)依赖包。

说明

其中x.y.z为依赖包的版本号，建议获取最新版本。

Python-binary-memcached 客户端连接 Memcached

步骤1 登录[分布式缓存服务管理控制台](#)。

- 步骤2** 在管理控制台左上角单击 ，选择实例所在的区域。
- 步骤3** 单击左侧菜单栏的“缓存管理”，进入缓存实例信息页面。
- 步骤4** 单击需要使用的其中一个Memcached实例的名称，进入该Memcached实例的基本信息页面。查看并获取该Memcached实例的IP地址/域名和端口。
- 步骤5** 将已获取的python-binary-memcached-x.y.z.zip依赖包上传到已创建的弹性云服务器，假设下载得到的包名为python-binary-memcached-x.y.z.zip。
- 步骤6** 登录弹性云服务器。
- 步骤7** 执行如下命令安装依赖包。

```
unzip -xzvf python-binary-memcached-x.y.z.zip
cd python-binary-memcached-x.y.z
python setup.py install
```

说明

如以上步骤安装报错，可使用apt或yum方式安装依赖包，如apt方式安装的具体命令如下：

```
apt install python-pip;
pip install python-binary-memcached;
```

- 步骤8** 新建Python文件如dcs_test.py，将如下Python代码复制到dcs_test.py文件并修改代码。

- 密码模式代码示例

其中 *ip or domain name:port* 需要修改为 **步骤4** 获取的Memcached实例IP地址/域名和端口。*userName* 和 *password* 需要修改为Memcached实例的用户名和密码。

```
import bmemcached
client = bmemcached.Client(('ip or domain name:port'), 'userName', 'password')
print "set('key', 'hello world!)"
print client.set('key', 'hello world!)"
print "get('key)"
print client.get('key')
```

- 免密模式代码示例

其中 *ip or domain name:port* 需要修改为 **步骤4** 获取的Memcached实例IP地址/域名和端口。

```
import bmemcached
client = bmemcached.Client('ip or domain name:port')
print "set('key', 'hello world!)"
print client.set('key', 'hello world!)"
print "get('key)"
print client.get('key')
```

- 步骤9** 运行dcs_test.py文件，可以看到如下结果。

```
# python test.py
set('key', 'hello world!)"
True
get('key)"
hello world!
```

----结束

5.2.4 Libmemcached 客户端连接 Memcached（C++）

介绍使用同一VPC内弹性云服务器ECS上的C++客户端连接Memcached实例的方法。

前提条件

- 已成功申请Memcached实例，且状态为“运行中”。
- 已创建弹性云服务器。创建弹性云服务器的方法，请参见《弹性云服务器用户指南》。

📖 说明

您创建的弹性云服务器必须与Memcached实例属于同一个VPC，并配置相同的安全组，以确保弹性云服务器与缓存实例的网络是连通的。

- 如果弹性云服务器与Memcached实例不在相同VPC中，可以通过建立VPC对等连接方式连通网络，具体请参考常见问题：[缓存实例是否支持跨VPC访问？](#)
- 如果弹性云服务器与Memcached实例配置了不同的安全组，可以通过设置安全组规则连通网络，具体请参考常见问题：[如何选择和配置安全组？](#)
- 您的弹性云服务器已安装好GCC，建议为4.8.4或更高版本。
- 已获取libmemcached-x.y.z.tar.gz依赖包。

📖 说明

其中x.y.z为依赖包的版本号，建议获取最新版本。

Libmemcached 客户端连接 Memcached（C++）

步骤1 登录[分布式缓存服务管理控制台](#)。

步骤2 在管理控制台左上角单击📍，选择实例所在的区域。

步骤3 单击左侧菜单栏的“缓存管理”，进入缓存实例信息页面。

步骤4 单击需要使用的其中一个Memcached实例的名称，进入该Memcached实例的基本信息页面。查看并获取该Memcached实例的IP地址/域名和端口。

步骤5 将已获取的libmemcached-x.y.z.tar.gz依赖包上传到已创建的弹性云服务器。

步骤6 登录弹性云服务器。

步骤7 安装sasl相关依赖包。

debian类系统：**apt install libsasl2-dev cloog.ppl**

Redhat类系统：**yum install cyrus-sasl***

步骤8 执行如下命令安装依赖包。

```
tar -xzvf libmemcached-x.y.z.tar.gz
```

```
cd libmemcached-x.y.z
```

```
./configure --enable-sasl
```

```
make
```

```
make install
```

步骤9 新建build.sh文件，将如下代码复制到build.sh文件。

```
g++ -o dcs_sample dcs_sample.cpp -lmemcached -std=c++0x -pthread -lsasl2
```

 说明

编译如果报错找不到libmemcached.so.11文件，请用find命令找到此文件，并将其复制到/usr/lib目录下。

步骤10 新建dcs_sample.cpp文件，将如下C++代码复制到dcs_sample.cpp文件并修改代码。

- 密码模式代码示例

其中`ip or domain name`和`port`需要修改为**步骤4**获取的Memcached实例IP地址/域名和端口。`userName`和`password`需要修改为Memcached实例的用户名和密码。

```
#include <iostream>
#include <string>
#include <libmemcached/memcached.h>
using namespace std;

#define IP "ip"
#define PORT "port"
#define USERNAME "userName"
#define PASSWORD "password"
memcached_return result;

memcached_st * init()
{
    memcached_st *memcached = NULL;
    memcached_server_st *cache;
    memcached = memcached_create(NULL);
    cache = memcached_server_list_append(NULL, IP, PORT, &result);

    sasl_client_init(NULL);
    memcached_set_sasl_auth_data(memcached, USERNAME, PASSWORD);
    memcached_behavior_set(memcached, MEMCACHED_BEHAVIOR_BINARY_PROTOCOL, 1);
    memcached_server_push(memcached, cache);
    memcached_server_list_free(cache);
    return memcached;
}

int main(int argc, char *argv[])
{
    memcached_st *memcached=init();
    string key = "memcached";
    string value = "hello world!";
    size_t value_length = value.length();
    int expire_time = 0;
    uint32_t flag = 0;

    result =
    memcached_set(memcached, key.c_str(), key.length(), value.c_str(), value.length(), expire_time, flag);
    if (result != MEMCACHED_SUCCESS){
        cout << "set data failed: " << result << endl;
        return -1;
    }
    cout << "set succeed, key: " << key << ", value: " << value << endl;
    cout << "get key:" << key << endl;
    char* result = memcached_get(memcached, key.c_str(), key.length(), &value_length, &flag, &result);
    cout << "value:" << result << endl;

    memcached_free(memcached);
    return 0;
}
```

- 免密模式代码示例

其中`ip`和`port`需要修改为**步骤4**获取的Memcached实例IP地址/域名和端口。

```
#include <iostream>
#include <string>
#include <libmemcached/memcached.h>
using namespace std;
```

```
#define IP "ip or domain name"
#define PORT port
memcached_return result;

memcached_st * init()
{
    memcached_st *memcached = NULL;
    memcached_server_st *cache;
    memcached = memcached_create(NULL);
    cache = memcached_server_list_append(NULL, IP, PORT, &result);
    memcached_server_push(memcached,cache);
    memcached_server_list_free(cache);
    return memcached;
}

int main(int argc, char *argv[])
{
    memcached_st *memcached=init();
    string key = "memcached";
    string value = "hello world!";
    size_t value_length = value.length();
    int expire_time = 0;
    uint32_t flag = 0;

    result =
    memcached_set(memcached,key.c_str(),key.length(),value.c_str(),value.length(),expire_time,flag);
    if (result != MEMCACHED_SUCCESS){
        cout <<"set data failed: " << result << endl;
        return -1;
    }
    cout << "set succeed, key: " << key << " ,value: " << value << endl;
    cout << "get key:" << key << endl;
    char* result = memcached_get(memcached,key.c_str(),key.length(),&value_length,&flag,&result);
    cout << "value:" << result << endl;

    memcached_free(memcached);
    return 0;
}
```

步骤11 执行如下命令编译源码。

```
chmod 700 build.sh
```

```
./build.sh
```

生成dcs_sample二进制文件。

步骤12 执行如下命令连接使用Memcached实例。

```
./dcs_sample
```

```
set succeed, key: memcached ,value: hello world!
get key:memcached
value:hello world!
```

----结束

5.2.5 Libmemcached 客户端连接 Memcached（PHP）

介绍使用同一VPC内弹性云服务器ECS上的PHP客户端连接Memcached实例的方法。

前提条件

- 已成功申请Memcached实例，且状态为“运行中”。
- 已创建弹性云服务器。创建弹性云服务器的方法，请参见《弹性云服务器用户指南》。

📖 说明

您创建的弹性云服务器必须与Memcached实例属于同一个VPC，并配置相同的安全组，以确保弹性云服务器与缓存实例的网络是连通的。

- 如果弹性云服务器与Memcached实例不在相同VPC中，可以通过建立VPC对等连接方式连通网络，具体请参考常见问题：[缓存实例是否支持跨VPC访问？](#)
- 如果弹性云服务器与Memcached实例配置了不同的安全组，可以通过设置安全组规则连通网络，具体请参考常见问题：[如何选择和配置安全组？](#)

RedHat 系列

以CentOS7为例介绍PHP版本客户端的安装。Redhat、Fedora等系统也适用。

步骤1 安装gcc-c++及make等编译组件。

```
yum install gcc-c++ make
```

步骤2 安装sasI相关包。

```
yum install cyrus-sasl*
```

步骤3 安装libmemcached。

由于libmemcached库需要增加sasI认证参数，因此不能直接使用yum命令安装。

```
wget https://launchpad.net/libmemcached/1.0/1.0.18/+download/libmemcached-1.0.18.tar.gz
```

```
tar -xvf libmemcached-1.0.18.tar.gz
```

```
cd libmemcached-1.0.18
```

```
./configure --prefix=/usr/local/libmemcached --enable-sasl
```

```
make && make install
```

📖 说明

安装libmemcached之前要完成gcc-c++、sasI相关组件的安装。否则会在编译过程中报错，报错问题解决后请执行make clean之后重新make。

步骤4 安装php。

```
yum install php-devel php-common php-cli
```

须知

php7.x对SASL认证存在兼容问题，建议使用php 5.6版本。如果yum源提供的php不是5.6版本，请自行在网上查找下载源。

步骤5 安装memcached客户端。

注意运行configure配置安装时，增加开启sasI参数。

```
wget http://pecl.php.net/get/memcached-2.1.0.tgz
```

```
tar zxvf memcached-2.1.0.tgz
```

```
cd memcached-2.1.0
```

phpize

```
./configure --with-libmemcached-dir=/usr/local/libmemcached --enable-memcached-sasl
```

```
make && make install
```

步骤6 增加php.ini配置。

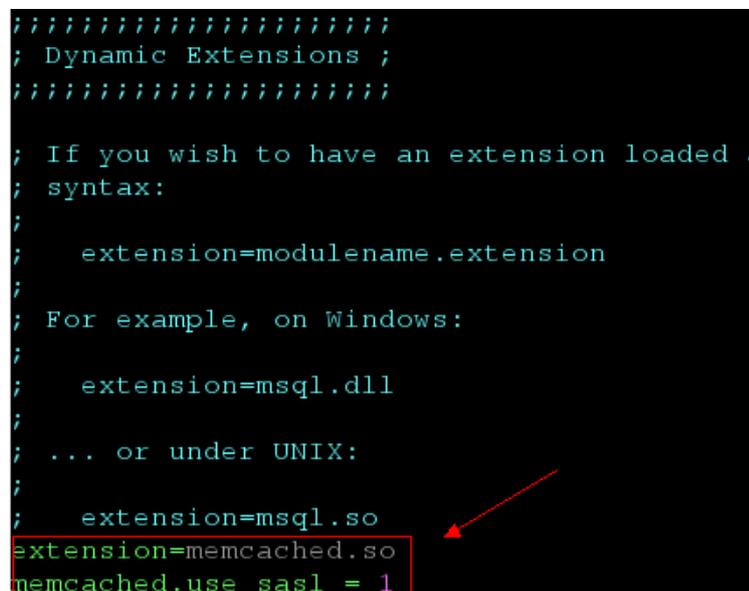
使用find或者locate命令找到php.ini文件。

```
find / -name php.ini
```

编辑该文件，增加以下两行：

```
extension=memcached.so  
memcached.use_sasl = 1
```

图 5-1 增加 php.ini 配置



步骤7 测试连接。

新增一个memcached.php文件，增加如下内容：

```
<?php  
$connect = new Memcached; //声明一个Memcached连接  
$connect->setOption(Memcached::OPT_COMPRESSION, false); //关闭压缩  
$connect->setOption(Memcached::OPT_BINARY_PROTOCOL, true); //使用二进制协议  
$connect->setOption(Memcached::OPT_TCP_NODELAY, true); //关闭TCP网络延迟策略  
$connect->addServer('{memcached_instance_ip}', 11211); //此处填写实例ip和端口  
$connect->setSaslAuthData('{username}', '{password}'); //如果实例开启免密访问，则删除或者注释此行  
$connect->set("DCS", "Come on!");  
echo "DCS: '$connect->get("DCS")";  
echo "\n";  
$connect->quit();  
?>
```

保存后运行情况如下：

```
[root@testphpmemcached ~]# php memcached.php  
DCS: Come on!  
[root@testphpmemcached ~]#
```

----结束

debian 系列

以Ubuntu为例，安装步骤如下：

步骤1 安装gcc及make等编译组件。

```
apt install gcc make
```

步骤2 安装php。

推荐使用php5.x的版本，对sasI认证兼容性较好。

按照如下步骤先添加php低版本的镜像源，然后安装php5.6以及php5.6-dev。

```
apt-get install -y language-pack-en-base;
```

```
LC_ALL=en_US.UTF-8;
```

```
add-apt-repository ppa:ondrej/php;
```

```
apt-get update;
```

```
apt-get install php5.6 php5.6-dev;
```

安装完成后，使用php -version，查看版本号为5.6，说明安装成功。

```
root@dcs-nodelete:/etc/apt# php -version
PHP 5.6.36-1+ubuntu16.04.1+deb.sury.org+1 (cli)
Copyright (c) 1997-2016 The PHP Group
```

📖 说明

如果需要卸载php，可使用如下命令：

```
apt install aptitude -y
```

```
aptitude purge `dpkg -l | grep php| awk '{print $2}' |tr "\n" " "`
```

步骤3 安装sasI组件。

```
apt install libsasl2-dev cloog.ppl
```

步骤4 安装libmemcached。

```
wget https://launchpad.net/libmemcached/1.0/1.0.18/+download/
libmemcached-1.0.18.tar.gz
```

```
tar -xvf libmemcached-1.0.18.tar.gz
```

```
cd libmemcached-1.0.18
```

```
./configure --prefix=/usr/local/libmemcached
```

```
make && make install
```

📖 说明

安装libmemcached之前要完成gcc、sasI相关组件的安装。否则会在编译过程中报错，报错问题解决后请执行make clean之后重新make。

步骤5 安装memcached客户端。

首先安装zlib组件。

```
apt install zlib1g.dev
```

安装注意运行configure配置安装时，增加开启sasl参数。

```
wget http://pecl.php.net/get/memcached-2.2.0.tgz;
```

```
tar zxvf memcached-2.2.0.tgz;
```

```
cd memcached-2.2.0;
```

```
phpize5.6;
```

```
./configure --with-libmemcached-dir=/usr/local/libmemcached --enable-memcached-sasl;
```

```
make && make install;
```

步骤6 增加pdo.ini配置。

使用find命令找到pdo.ini文件。

```
find / -name pdo.ini
```

默认应该在/etc/php/5.6/mods-available路径下。编辑该文件，增加以下两行：

```
extension=memcached.so  
memcached.use_sasl = 1
```

图 5-2 增加 pdo.ini 配置

```
; configuration for php common module  
; priority=10  
extension=pdo.so  
extension=memcached.so  
memcached.use_sasl=1
```

步骤7 测试连接。

新增一个memcached.php文件，增加如下内容：

```
<?php  
$connect = new Memcached; //声明一个Memcached连接  
$connect->setOption(Memcached::OPT_COMPRESSION, false); //关闭压缩  
$connect->setOption(Memcached::OPT_BINARY_PROTOCOL, true); //使用二进制协议  
$connect->setOption(Memcached::OPT_TCP_NODELAY, true); //关闭TCP网络延迟策略  
$connect->addServer('{memcached_instance_ip}', 11211); //此处填写实例ip和端口  
$connect->setSaslAuthData('{username}', '{password}'); //如果实例开启免密访问，则删除或者注释此行  
$connect->set("DCS", "Come on!");  
echo 'DCS: ', $connect->get("DCS");  
echo "\n";  
$connect->quit();  
?>
```

保存后运行情况如下：

```
[root@dcs-nodelete ~]# php memcached.php  
DCS: Come on!  
[root@dcs-nodelete ~]#
```

----结束

6 管理实例

6.1 查看和修改 DCS 实例基本信息

本节介绍如何在DCS管理控制台查看和修改DCS缓存实例的基本信息。

说明

DCS Redis 3.0已下线，暂停售卖，建议使用Redis 4.0及以上版本。

查看和修改 DCS 实例基本信息

步骤1 登录[分布式缓存服务管理控制台](#)。

步骤2 在管理控制台左上角单击 ，选择实例所在的区域。

步骤3 单击左侧菜单栏的“缓存管理”，进入缓存管理页面。

步骤4 查询DCS缓存实例。

- 支持通过关键字搜索对应的DCS缓存实例。
直接在搜索栏输入关键字即可。
- 支持通过指定属性的关键字查询对应的DCS缓存实例。
当前支持的属性有“名称”、“规格”、“ID”、“IP地址”、“可用区”、“状态”、“实例类型”、“缓存类型”等。
例如，单击搜索栏，选择过滤属性“缓存类型”，可选择缓存类型（Redis 3.0、Redis 4.0、Redis 5.0和Redis 6.0）过滤DCS缓存实例。

更多的搜索设置帮助，请单击搜索栏右侧的搜索帮助。

步骤5 在需要查看的DCS缓存实例左侧，单击该实例的名称，进入实例的基本信息页面。参数说明如[表6-1](#)所示。

表 6-1 参数说明

信息类型	参数	说明
基本信息	名称	DCS缓存实例的名称。单击“名称”后的  可以修改实例名称。
	状态	DCS缓存实例状态。
	ID	DCS缓存实例的ID。
	缓存类型	DCS的缓存版本。例如，基础版Redis 4.0。 实例创建后，不支持修改缓存版本，如需使用其他版本，建议重新创建实例并进行数据迁移。
	小版本	DCS缓存实例的小版本号。DCS会不定期升级产品小版本，优化产品功能和修复产品缺陷，单击“升级小版本”可以升级实例的小版本到最新版本。 说明 <ul style="list-style-type: none"> 默认不支持查看“小版本”，如需查看或升级小版本，请联系客服开通白名单。 建议在业务低峰期执行实例升级。业务高峰期（如实例内存利用率、CPU利用率达到90%以上或写入流量过大）升级可能会失败，若升级失败，请在业务低峰期再次尝试升级。 实例升级采用节点迁移的方式，在数据迁移过程中，访问时延会增大，每迁移一个分片会发生一次秒级闪断和一分钟以内的只读，请确保客户端应用具备重连机制和处理异常的能力。 Cluster集群请确保客户端能正常处理MOVED和ASK命令，否则会导致请求失败。 仅Redis 4.0及以上基础版实例显示该参数，企业版实例不支持升级小版本。 如果实例已经为最新小版本，不支持升级。 升级小版本不会改变实例的连接地址、访问密码、白名单配置、参数及告警配置等信息。
代理版本	DCS缓存实例的Proxy代理节点版本号，仅Proxy集群和读写分离实例显示代理版本，其他实例类型不涉及。DCS会不定期升级产品的代理版本，优化产品功能和修复产品缺陷，单击“升级代理版本”可以升级实例的代理版本到最新版本。 说明 <ul style="list-style-type: none"> 默认不支持查看“代理版本”，如需查看或升级代理版本，请联系客服开通白名单。 实例升级过程中会发生秒级连接闪断，请确保客户端应用具备重连机制和处理异常的能力，建议在业务低峰期升级。 如果实例已经为最新代理版本，不支持升级。 升级代理版本不会改变实例的连接地址、访问密码、白名单配置、参数及告警配置等信息。 	

信息类型	参数	说明
	实例类型	DCS缓存实例类型，支持“单机”、“主备”、“Proxy集群”、“Cluster集群”和“读写分离”。 如需变更实例类型，请参考 变更DCS实例规格 了解支持变更的实例类型、变更实例类型的注意事项和操作步骤。
	规格（GB）	DCS缓存实例规格。 如需变更实例规格，请参考 变更DCS实例规格 了解变更实例规格的注意事项和操作步骤。
	带宽	DCS缓存实例的带宽。 单击“调整带宽”可以调整实例的带宽值，详情请参考 调整实例带宽 。
	已用/可用内存（MB）	DCS缓存实例已经使用的内存量和您可以使用的最大内存量。 已使用的内存量包括两部分： <ul style="list-style-type: none"> • 用户存储的数据； • Redis-server内部的buffer（如client buffer、repl-backlog等），以及内部的数据结构。
	CPU	DCS缓存实例的CPU架构。只有Redis实例才会显示该字段。实例创建后，不支持变更CPU架构。
	企业项目	实例的企业项目。单击参数后的  可以切换实例的企业项目。 企业项目可提供资源/人员/财务等隔离，修改企业项目会修改以上内容的隔离目标。
	维护时间窗	运维操作时间。单击参数后的  可以修改时间窗。 在下拉框中选择新的维护时间窗，单击  保存修改，单击  取消修改。 修改成功后立即生效。
	描述	DCS缓存实例的描述信息。单击“描述”后的  可以修改描述信息。
连接信息	访问方式	根据是否配置访问密码，访问方式分为“密码访问”和“免密访问”。 如需变更密码访问方式，请参考 配置Redis访问密码 。

信息类型	参数	说明
	连接地址	<p>同一VPC内客户端连接Redis缓存实例时的域名地址和端口。</p> <p>单击“连接地址”后的可以修改端口。实例创建后，不支持修改连接地址。</p> <p>说明</p> <ul style="list-style-type: none"> 如果是Redis 4.0、Redis 5.0和Redis 6.0主备实例，“连接地址”表示主节点的域名和端口号，“只读地址”，表示备节点的域名和端口号。客户端连接时，可选择主节点或备节点的域名和端口号，详情请参考主备实例架构。 仅Redis 4.0/5.0/6.0基础版实例支持修改端口，Redis 3.0实例、Redis 6.0企业版和Memcached实例不支持。
	IP地址	<p>同一VPC内客户端连接DCS缓存实例的IP地址和端口。</p> <p>单击IP地址后的可以修改实例的端口。实例创建后，不支持修改IP地址。在连接实例时，建议优先使用域名地址进行连接。</p>
	公网访问	<p>目前仅Redis 3.0实例默认支持开启公网访问。开启Redis 3.0实例公网访问，请参考公网连接Redis 3.0 (Redis 3.0已停售)。其他版本暂不支持公网访问。</p> <p>如果是Redis 4.0、Redis 5.0和Redis 6.0版本的实例，需要先联系客服开通公网访问白名单后，才支持开启公网访问。开启Redis 4.0/5.0/6.0实例公网访问、及获取公网访问地址的操作，请参考开启Redis 4.0/5.0/6.0公网访问并获取公网访问地址。</p> <p>Memcached（已停售）不支持公网访问。</p>
网络信息	可用区	<p>缓存实例节点所属的可用区。</p> <p>对于单可用区的多副本集群实例，支持变更备节点可用区，具体请参考变更DCS集群实例为多可用区。其他场景均不支持变更可用区。</p>
	虚拟私有云	DCS缓存实例所在的私有网络。实例创建后，不支持修改虚拟私有云。
	子网	DCS缓存实例所属子网。实例创建后，不支持修改子网。
	安全组	<p>DCS缓存实例所关联的安全组。单击“安全组”后的可以修改安全组。</p> <p>在下拉框选择新的安全组，单击保存修改，单击取消修改。修改成功后立即生效。</p> <p>当前仅Redis 3.0、Redis 6.0企业版和Memcached支持安全组访问控制，Redis 4.0、Redis 5.0和Redis 6.0基础版实例是基于VPC Endpoint，暂不支持安全组，建议配置实例白名单。</p> <p>说明</p> <p>此处只能下拉选择已创建的安全组，若需要重新配置安全组，可参考如何选择和配置安全组。</p>

信息类型	参数	说明
实例拓扑	-	查看实例拓扑图，将鼠标移动到具体节点图标，可以查看该节点的总体监控信息，或者单击节点图标，可以查看节点历史监控信息。 单机实例不显示实例的拓扑图。
付费信息	计费方式	DCS缓存实例的计费方式。如需变更计费方式，请参考 变更计费模式概述 。
	创建时间	DCS缓存实例开始创建时间。
	运行时间	DCS缓存实例完成创建时间。

---结束

6.2 查看 DCS 实例后台任务

对实例的一些操作，如扩容、修改密码、重置密码等，会启动一个后台任务，您可以在DCS管理控制台的后台任务页，查看该操作的状态等信息，同时可通过删除操作，清理任务信息。

查看 DCS 实例后台任务

步骤1 登录[分布式缓存服务管理控制台](#)。

步骤2 在管理控制台左上角单击 ，选择实例所在的区域。

步骤3 单击左侧菜单栏的“缓存管理”，进入实例信息页面。

“缓存管理”支持通过筛选来查询对应的缓存实例。支持的筛选条件有“名称”、“规格”、“ID”、“IP地址”、“可用区”、“状态”、“实例类型”、“缓存类型”等。

步骤4 在需要查看的DCS缓存实例左侧，单击该实例的名称，进入该实例的基本信息页面。

步骤5 单击“后台任务”，查看后台任务列表。

可以通过选择任务的时间段，输入任务属性或关键字筛选任务。

- 单击 ，刷新任务状态。
- 单击“操作”列下的“删除”，清理任务信息。

说明

您只能在任务已经执行完成，即任务状态为成功或者失败时，才能执行删除操作。

---结束

6.3 查看 DCS 实例的客户端连接信息

DCS的会话管理功能可用于查询实例的客户端连接信息，也支持通过会话管理断开客户端连接。

约束与限制

- 会话管理仅显示外部客户端连接信息，不显示通过Web-Cli连接的信息。
- 会话管理目前仅在“北京一”、“北京四”、“上海一”、“上海二”、“广州”、“贵阳一”区域支持。
- Redis 4.0及以上版本的实例，支持会话管理功能。Redis 3.0版本实例可以通过执行Client List命令，查询客户端IP信息。

查看 DCS 实例的客户端连接信息

步骤1 登录[分布式缓存服务管理控制台](#)。

步骤2 在管理控制台左上角单击 ，选择实例所在的区域。

步骤3 单击左侧菜单栏的“缓存管理”，进入实例信息页面。

步骤4 单击需要查看的DCS缓存实例名称，进入该实例的基本信息页面。

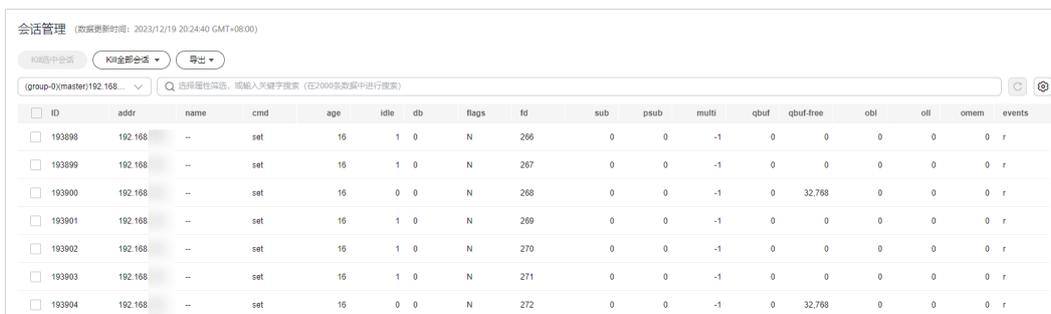
步骤5 单击“会话管理”。

步骤6 在会话管理页面，可以显示当前连接该实例的客户端会话信息。

说明

- Proxy集群和读写分离实例查询的是连接单proxy节点的会话信息，单机、主备和Cluster集群实例查询的是连接单数据节点的会话信息。
- 在页面中，可以选择需要查询的数据节点或Proxy节点、输入并查询指定的会话地址、更新查询信息、及设置会话的显示项。

图 6-1 会话管理



ID	addr	name	cmd	age	idle	db	flags	fd	sub	psub	multi	qbuf	qbuf-free	obl	oll	omem	events
193898	192.168...	-	set	16	1	0	N	266	0	0	-1	0	0	0	0	0	r
193899	192.168...	-	set	16	1	0	N	267	0	0	-1	0	0	0	0	0	r
193900	192.168...	-	set	16	0	0	N	268	0	0	-1	0	32,768	0	0	0	r
193901	192.168...	-	set	16	1	0	N	269	0	0	-1	0	0	0	0	0	r
193902	192.168...	-	set	16	1	0	N	270	0	0	-1	0	0	0	0	0	r
193903	192.168...	-	set	16	1	0	N	271	0	0	-1	0	0	0	0	0	r
193904	192.168...	-	set	16	0	0	N	272	0	0	-1	0	32,768	0	0	0	r

表 6-2 会话字段说明

字段	描述
ID	会话的唯一ID标识。
addr	会话地址。如果Redis开启了IP透传，该地址为客户端IP地址；如果Redis未开启IP透传，该地址为内部私网占用IP。
name	客户端名称，可通过代码中的 setClientName(...) 配置，如果未配置则该字段为空。
cmd	最近一次执行的命令。

字段	描述
age	连接的时长，单位：秒。
idle	连接空闲的时间，单位：秒。
db	最近一次执行命令的DB标识，例如DB0，则该字段显示0。
flags	连接标志位，M表示来自主节点的连接，S表示来自从节点的连接，其余标志请参考： https://redis.io/docs/latest/commands/client-list/
fd	连接FD。
sub	当前连接普通订阅的Channel数量。
psub	当前连接模式订阅的Channel数量。
multi	通过事务/LUA方式执行的命令数量，如果未执行过该字段显示-1。
qbuf	输入缓冲区的空间大小（字节数）。
qbuf-free	输入缓冲区的剩余空间大小（字节数）。
obl	输出缓冲区的长度。
oll	输出缓冲区的列表长度。
omem	输出缓冲区的空间大小（字节数）。
events	连接FD上产生的可读、可写事件。读事件：r，写事件：w。

步骤7 选择需要kill的会话，单击“kill选中会话”可断开选择的客户端连接，也可以选择“kill全部会话”。

如果所kill的客户端具备重连机制，断开后会自动重连。

步骤8 如果需要导出客户端连接数据，单击“导出”，可以选择导出全部或部分已选中的连接数据。

----结束

6.4 修改 DCS 实例配置参数

为了确保分布式缓存服务发挥出最优性能，您可以根据自己的业务情况对DCS缓存实例的运行参数进行调整。

例如，如需关闭实例的持久化功能，需要将“appendonly”修改为“no”。更多缓存实例的参数说明请参考[缓存实例配置参数说明](#)。

说明

实例配置参数修改之后，参数会立即生效（不需要手动重启实例），如果是集群，会在所有分片生效。

修改单实例配置参数

- 步骤1 登录[分布式缓存服务管理控制台](#)。
- 步骤2 在管理控制台左上角单击 ，选择实例所在的区域。
- 步骤3 单击左侧菜单栏的“缓存管理”。
- 步骤4 在“缓存管理”页面，单击DCS缓存实例的名称。
- 步骤5 单击“实例配置 > 参数配置”进入配置界面。
- 步骤6 单击需要修改的参数右侧的“修改”，修改该参数。如果需要同时修改多个参数，单击参数列表上方的“修改”，可以同时修改多个参数。

图 6-2 修改参数



- 步骤7 根据需要修改配置参数。
各参数的详细介绍见[缓存实例配置参数说明](#)，一般情况下，按照系统默认值设置参数即可。
- 步骤8 单击“保存”。
- 步骤9 在弹出的修改确认对话框中，单击“是”，确认修改参数。
参数修改的任务状态“成功”后，修改参数成功。

----结束

缓存实例配置参数说明

说明

- [表6-3](https://redis.io/topics/memory-optimization)中的内存优化相关参数可以参考Redis官网说明，链接：<https://redis.io/topics/memory-optimization>。
- 不同实例类型和版本支持配置的参数和取值可能略有不同，控制台参数配置页面中未显示的参数不支持修改。

表 6-3 Redis 缓存实例配置参数说明

参数名	参数解释	例外场景	取值范围	默认值
active-expire-num	定期删除过期键时随机检查key的数量。 如果增加该参数值，短时间内可能出现CPU使用率升高，命令时延增大，随后恢复正常。如果减小该参数值，内存中过期key数量可能上升，占用更多的内存。	Redis 3.0和Redis 6.0企业版实例不支持该参数。 说明 该参数为2021年9月新增，在此之前创建的实例，如果无法正常修改参数值，请联系客服处理。	1 ~ 1,000	20
timeout	客户端空闲N秒（timeout参数的取值）后将断开连接。参数值为0表示禁用该功能，即客户端空闲不会断开连接。	-	0~7200 单位：秒	0
appendfsync	操作系统的fsync函数刷新缓冲区数据到磁盘，有些操作系统会真正刷新磁盘上的数据，其他一些操作系统只会尝试尽快完成。	单机实例不支持该参数。	<ul style="list-style-type: none"> no：不调用fsync，由操作系统决定何时刷新数据到磁盘，性能最高。 always：每次写AOF文件都调用fsync，性能最差，但数据最安全。 everysec：每秒调用一次fsync。兼具数据安全和性能。 	no

参数名	参数解释	例外场景	取值范围	默认值
appendonly	指定是否在每次更新操作后进行日志记录，Redis在默认情况下是异步的把数据写入磁盘，如果不开启，可能会在断电时导致一段时间内的数据丢失。	单机实例不支持该参数。	<ul style="list-style-type: none"> • yes: 开启日志记录，即开启持久化功能。 • no: 关闭日志记录，即关闭持久化功能。 • only-replica: 仅开启从节点持久化功能。 <p>说明 仅Redis 4.0及以上版本的基础版主备或集群实例，或企业版高性能型主备实例支持only-replica配置项。 如果以上版本及实例类型的实例，控制台中无only-replica配置项，可以联系客服放通该配置项的白名单。</p>	yes
client-output-buffer-limit-slave-soft-seconds	主从同步缓冲区大小超过client-output-buffer-slave-soft-limit主从同步缓冲区大小软限制，并持续时间超过此参数值时，服务端会主动断开连接。该参数值配置的越小，越容易断开断连。	-	0~60 单位：秒	60

参数名	参数解释	例外场景	取值范围	默认值
client-output-buffer-slave-hard-limit	主从同步缓冲区大小硬限制。如果同步缓冲区大小超过这个值，则主从同步连接立即断开。该参数值配置的越小，越容易断开断连。	-	0 ~ 17,179,869,184 单位：字节	1,717,986,918
client-output-buffer-slave-soft-limit	主从同步缓冲区大小软限制。如果同步缓冲区大小超过这个值，并持续时间达到client-output-buffer-limit-slave-soft-seconds参数配置的秒数，则主从同步连接断开。该参数值配置的越小，越容易断开断连。	-	0 ~ 17,179,869,184 单位：字节	1,717,986,918

参数名	参数解释	例外场景	取值范围	默认值
maxmemory-policy	内存使用达到上限（maxmemory）时，对缓存数据的逐出策略，有8个取值供选择。	-	<ul style="list-style-type: none"> volatile-lru：根据LRU算法删除设置了过期时间的键值。 allkeys-lru：根据LRU算法删除任一键值。 volatile-random：删除设置了过期时间的随机键值。 allkeys-random：删除一个随机键值。 volatile-ttl：删除即将过期的键值，即TTL值最小的键值。 noeviction：不删除任何键值，只是返回一个写错误。 volatile-lfu：根据LFU算法删除设置了过期时间的键值。 allkeys-lfu：根据LFU算法删除任一键值。 <p>详情可参考Redis官网的逐出策略。</p>	volatile-lru 说明 如果是2020年7月之前创建的Redis实例，且没有修改过该参数，则默认值为noeviction。如果是2020年7月之后创建的实例，默认值都为volatile-lru。
lua-time-limit	Lua脚本的最长执行时间。	-	100 ~ 5,000 单位：毫秒	5,000

参数名	参数解释	例外场景	取值范围	默认值
master-read-only	设置实例为只读状态。设置只读后，所有写入命令将返回失败。	Proxy集群实例不支持该参数。	<ul style="list-style-type: none"> • yes • no 	no
maxclients	<p>最大同时连接的客户端个数。客户端数量超过该配置值时，会发生连接失败。该参数值配置过大时，会增加服务器处理连接的开销，影响服务器性能，命令时延增大；该参数值配置过小，可能无法完全发挥服务器的性能。</p> <p>该参数为单节点（单分片）连接数上限：</p> <ul style="list-style-type: none"> • 集群实例单节点连接数上限=实例连接数上限/分片数。 • 单机/主备实例，单节点连接数上限=实例最大连接数。 	读写分离实例暂不支持该参数。	1,000 ~ 50,000	10,000
proto-max-bulk-len	Redis协议中的最大的请求大小。该参数的配置值需要大于客户请求的长度，否则请求将无法执行。	-	1,048,576 ~ 536,870,912 单位：字节	536,870,912

参数名	参数解释	例外场景	取值范围	默认值
repl-backlog-size	用于增量同步的复制积压缓冲区大小。这是一个用来在从节点断开连接时，存放从节点数据的缓冲区，当从节点重新连接时，如果丢失的数据少于缓冲区的大小，可以用缓冲区中的数据开始增量同步。	-	16,384 ~ 1,073,741,824 单位：字节	1,048,576
repl-backlog-ttl	从节点断开后，主节点释放复制积压缓冲区内存的秒数。值为0时表示永不释放复制积压缓冲区内存。	-	0 ~ 604,800 单位：秒	3,600
repl-timeout	主从同步超时时间。	单机实例不支持该参数。	30 ~ 3,600 单位：秒	60
hash-max-ziplist-entries	当hash表中记录数少于参数值，使用ziplist编码格式，节约内存。	-	1~10000	512
hash-max-ziplist-value	当hash表中各字段长度的最大值小于参数值时，使用ziplist编码格式，节约内存。	-	1~10000	64
set-max-intset-entries	当一个集合仅包含字符串且整型元素数量少于参数值时，使用intset编码格式，节约内存。	-	1~10000	512

参数名	参数解释	例外场景	取值范围	默认值
zset-max-ziplist-entries	当有序集合中记录数少于参数值，使用ziplist编码格式，节约内存。	-	1~10000	128
zset-max-ziplist-value	当有序集合中各字段长度的最大值小于参数值时，使用ziplist编码格式，节约内存。	-	1~10000	64
latency-monitor-threshold	<p>延时监控的采样时间阈值（最小值）。当阈值设置为0时，不做监控，也不采样；当阈值设置为大于0时，将监控并记录执行耗时大于阈值的操作。</p> <p>您可以通过LATENCY等命令获取统计数据 and 配置、执行采样监控。</p> <p>说明 latency-monitor-threshold参数一般在定位问题时使用。采集完latency信息，定位问题后，建议重新将latency-monitor-threshold设置为0，以免引起不必要的延迟。</p>	Proxy集群实例不支持该参数。	0~86400000 单位：毫秒	0

参数名	参数解释	例外场景	取值范围	默认值
notify-keyspace-events	键空间通知，配置该参数后客户端可以通过Redis的订阅与发布功能，来接收那些以某种方式改动了Redis数据集的事件。该参数配置为空时，功能关闭。当参数不是空字符串时，功能开启。	Proxy集群实例不支持该参数。	可配置为以下字符的任意组合，指定了服务器该发送哪些类型的通知： K：键空间通知，所有通知以__keyspace@__为前缀。 E：键事件通知，所有通知以__keyevent@__为前缀。 g：DEL、EXPIRE、RENAME等类型无关的通用命令的通知。 \$：字符串命令的通知。 l：列表命令的通知。 s：集合命令的通知。 h：哈希命令的通知。 z：有序集合命令的通知。 x：过期事件：每当有过期键被删除时发送。 e：驱逐(evict)事件：每当有键因为maxmemory政策而被删除时发送。 A：参数g \$lshzxe的别名。 输入的参数中至少有一个K或者E，A不能	Ex

参数名	参数解释	例外场景	取值范围	默认值
			与g\$shzxe同时出现，不能出现相同字母。例如，如果只想订阅键空间中和列表相关的通知，那么参数就应该设为Kl。将参数设为字符串"AKE"表示发送所有类型的通知。	
slowlog-log-slower-than	Redis慢查询会记录超过指定执行时间的命令。slowlog-log-slower-than用于配置记录到redis慢查询命令执行时间阈值。	-	0~1,000,000 单位：微秒	10,000
proxy-slowlog-log-slower-than	Proxy节点慢查询会记录超过指定执行时间的命令。proxy-slowlog-log-slower-than用于配置记录到proxy节点慢查询命令的执行时间阈值。	目前仅“华东-上海二”和“华南-广州”区域的Proxy集群和读写分离实例支持该参数。	30,000~2,000,000 单位：微秒	256,000
slowlog-max-len	Redis慢查询记录的条数。注意慢查询记录会消耗额外的内存。可以通过执行SLOWLOG RESET命令清除慢查询记录。	-	0~1,000	128

参数名	参数解释	例外场景	取值范围	默认值
proxy-slowlog-max-len	Proxy节点慢查询记录的条数。注意慢查询记录会消耗额外的内存。可以通过执行SLOWLOG RESET命令清除慢查询记录。	目前仅“华东-上海二”和“华南-广州”区域的Proxy集群和读写分离实例支持该参数。	0 ~ 1,000	128
auto-kill-timeout-lua-process	开启该参数时，执行超时的lua脚本进程会被自动kill。如果lua脚本执行了写操作无法kill，并且实例开启了持久化时，则该lua脚本所在的节点会自动重启，lua脚本执行的写操作将不被保存。	单机实例和Redis 3.0实例不支持该参数。	<ul style="list-style-type: none"> • yes: 开启 • no: 关闭 	no
audit-log-customer-command-list	审计日志需要额外记录的命令。审计日志默认只记录写命令，该参数可以指定用户额外需要记录的命令。只有在开启了审计日志的情况下才有效。	审计日志 目前为白名单特性，只有开启了审计日志白名单的用户，并且仅当实例类型为Proxy集群实例时，才显示该参数。	该参数配置的值必须由英文字母或与.-_组成的命令，且必须以英文字母开头和结尾，每个命令最大长度为10，最多支持配置10个命令。命令之间用空格隔开，配置的内容最后必须以空格结尾。	-

参数名	参数解释	例外场景	取值范围	默认值
dispatch-pubsub-to-fixed-shard	该参数用于指定发布订阅的channel是否集中在0号槽位对应的分片上。开启开关时，发布订阅的处理逻辑与单机版一致。用户轻度使用发布订阅时，推荐开启开关。用户重度使用发布订阅功能时，请使用默认配置，让订阅分担到全部分片中。	仅Proxy集群实例支持该参数。	<ul style="list-style-type: none"> • yes: 开启开关。订阅的channel都集中在0号槽位对应的集群分片上。 • no: 关闭开关。根据订阅channel值计算出hash槽位值，再将channel分散到各个槽位对应的分片中。 	no
readonly-lua-route-to-slave-enabled	是否把只读账号的只读lua路由到从节点，如果开启，只读账号的只读lua可以执行，且会路由到从节点。	仅读写分离实例支持该参数。	<ul style="list-style-type: none"> • yes: 开启 • no: 关闭 	no
cluster-sentinel-enabled	实例兼容哨兵（Sentinel）模式。	仅Proxy集群实例支持该参数。	<ul style="list-style-type: none"> • yes: 开启 • no: 关闭 	no
scan-support-wr-split	该参数关闭时，scan命令在master节点执行，该参数开启后，scan命令将在slave节点执行。开启该参数后，可以降低scan命令对主节点的压力，但对于新写入主节点的数据，可能无法第一时间在从节点查询到。	<p>仅Proxy集群实例支持该参数。</p> <p>如果历史创建的Proxy集群实例不支持该参数，请联系客服升级实例版本。</p>	<ul style="list-style-type: none"> • yes: 开启 • no: 关闭 	no

表 6-4 Memcached 缓存实例配置参数说明

参数名	参数解释	取值范围	默认值
timeout	客户端与服务端连接空闲超时断开时间，参数设为0表示连接永不断开。	0~7200 单位：秒	0
maxclients	最大同时连接的客户端个数。	1,000 ~ 10,000	10,000
maxmemory-policy	内存使用达到上限时对缓存数据管理策略。	<ul style="list-style-type: none"> volatile-lru：根据LRU算法删除设置了过期时间的键值。 allkeys-lru：根据LRU算法删除任一键值。 volatile-random：删除设置了过期时间的随机键值。 allkeys-random：删除一个随机键值。 volatile-ttl：删除即将过期的键值，即TTL值最小的键值。 noeviction：不删除任何键值，只是返回一个写错误。 	noeviction
reserved-memory-percent	预留给后台进程用于持久化、主从同步等内部处理的内存百分比（相对于最大可用内存的百分比）。	0-80	30

6.5 配置 DCS 实例参数模板

6.5.1 查看 DCS 实例参数模板信息

不同Redis版本及类型的实例对应了不同的系统默认参数模板，系统默认参数模板中包含实例的默认参数配置。用户也可以根据业务需求创建不同参数配置的自定义参数模板，在创建实例时，会根据选择的参数模板，创建对应参数配置的实例。

本节介绍如何在分布式缓存服务管理控制台查看实例参数模板的参数信息。

查看参数模板信息

步骤1 登录分布式缓存服务管理控制台。

步骤2 在管理控制台左上角单击 ，选择实例所在的区域。

步骤3 单击左侧菜单栏的“参数模板”。

步骤4 在“参数模板”页面，选择“系统默认”或者“自定义”。

步骤5 查询参数模板。

当前支持通过模板名称搜索对应的参数模板，直接在搜索栏输入关键字即可。

步骤6 在需要查看的参数模板左侧，单击该模板名称，进入模板的参数页面。各参数的详细介绍见[表6-5](#)。

表 6-5 Redis 缓存实例配置参数说明

参数名	参数解释	例外场景	取值范围	默认值
active-expire-num	定期删除过期键时随机检查key的数量。 如果增加该参数值，短时间内可能出现CPU使用率升高，命令时延增大，随后恢复正常。如果减小该参数值，内存中过期key数量可能上升，占用更多的内存。	Redis 3.0和Redis 6.0企业版实例不支持该参数。 说明 该参数为2021年9月新增，在此之前创建的实例，如果无法正常修改参数值，请联系客服处理。	1~1,000	20
timeout	客户端空闲N秒（timeout参数的取值）后将断开连接。参数值为0表示禁用该功能，即客户端空闲不会断开连接。	-	0~7200 单位：秒	0

参数名	参数解释	例外场景	取值范围	默认值
appendfsync	操作系统的fsync函数刷新缓冲区数据到磁盘，有些操作系统会真正刷新磁盘上的数据，其他一些操作系统只会尝试尽快完成。	单机实例不支持该参数。	<ul style="list-style-type: none"> no: 不调用fsync，由操作系统决定何时刷新数据到磁盘，性能最高。 always: 每次写AOF文件都调用fsync，性能最差，但数据最安全。 everysec: 每秒调用一次fsync。兼具数据安全和性能。 	no
appendonly	指定是否在每次更新操作后进行日志记录，Redis在默认情况下是异步的把数据写入磁盘，如果不开启，可能会在断电时导致一段时间内的数据丢失。	单机实例不支持该参数。	<ul style="list-style-type: none"> yes: 开启日志记录，即开启持久化功能。 no: 关闭日志记录，即关闭持久化功能。 only-replica: 仅开启从节点持久化功能。 <p>说明 仅Redis 4.0及以上版本的基础版主备或集群实例，或企业版高性能型主备实例支持only-replica配置项。 如果以上版本及实例类型的实例，控制台中无only-replica配置项，可以联系客服放通该配置项的白名单。</p>	yes

参数名	参数解释	例外场景	取值范围	默认值
client-output-buffer-limit-slave-soft-seconds	主从同步缓冲区大小超过client-output-buffer-slave-soft-limit主从同步缓冲区大小软限制，并持续时间超过此参数值时，服务端会主动断开连接。该参数值配置的越小，越容易断开断连。	-	0~60 单位：秒	60
client-output-buffer-slave-hard-limit	主从同步缓冲区大小硬限制。如果同步缓冲区大小超过这个值，则主从同步连接立即断开。该参数值配置的越小，越容易断开断连。	-	0~17,179,869,184 单位：字节	1,717,986,918
client-output-buffer-slave-soft-limit	主从同步缓冲区大小软限制。如果同步缓冲区大小超过这个值，并持续时间达到client-output-buffer-limit-slave-soft-seconds参数配置的秒数，则主从同步连接断开。该参数值配置的越小，越容易断开断连。	-	0~17,179,869,184 单位：字节	1,717,986,918

参数名	参数解释	例外场景	取值范围	默认值
maxmemory-policy	内存使用达到上限（maxmemory）时，对缓存数据的逐出策略，有8个取值供选择。	-	<ul style="list-style-type: none"> volatile-lru：根据LRU算法删除设置了过期时间的键值。 allkeys-lru：根据LRU算法删除任一键值。 volatile-random：删除设置了过期时间的随机键值。 allkeys-random：删除一个随机键值。 volatile-ttl：删除即将过期的键值，即TTL值最小的键值。 noeviction：不删除任何键值，只是返回一个写错误。 volatile-lfu：根据LFU算法删除设置了过期时间的键值。 allkeys-lfu：根据LFU算法删除任一键值。 <p>详情可参考Redis官网的逐出策略。</p>	volatile-lru 说明 如果是2020年7月之前创建的Redis实例，且没有修改过该参数，则默认值为noeviction。如果是2020年7月之后创建的实例，默认值都为volatile-lru。
lua-time-limit	Lua脚本的最长执行时间。	-	100 ~ 5,000 单位：毫秒	5,000

参数名	参数解释	例外场景	取值范围	默认值
master-read-only	设置实例为只读状态。设置只读后，所有写入命令将返回失败。	Proxy集群实例不支持该参数。	<ul style="list-style-type: none"> • yes • no 	no
maxclients	<p>最大同时连接的客户端个数。客户端数量超过该配置值时，会发生连接失败。该参数值配置过大时，会增加服务器处理连接的开销，影响服务器性能，命令时延增大；该参数值配置过小，可能无法完全发挥服务器的性能。</p> <p>该参数为单节点（单分片）连接数上限：</p> <ul style="list-style-type: none"> • 集群实例单节点连接数上限=实例连接数上限/分片数。 • 单机/主备实例，单节点连接数上限=实例最大连接数。 	读写分离实例暂不支持该参数。	1,000 ~ 50,000	10,000
proto-max-bulk-len	Redis协议中的最大的请求大小。该参数的配置值需要大于客户请求的长度，否则请求将无法执行。	-	1,048,576 ~ 536,870,912 单位：字节	536,870,912

参数名	参数解释	例外场景	取值范围	默认值
repl-backlog-size	用于增量同步的复制积压缓冲区大小。这是一个用来在从节点断开连接时，存放从节点数据的缓冲区，当从节点重新连接时，如果丢失的数据少于缓冲区的大小，可以用缓冲区中的数据开始增量同步。	-	16,384~1,073,741,824 单位：字节	1,048,576
repl-backlog-ttl	从节点断开后，主节点释放复制积压缓冲区内存的秒数。值为0时表示永不释放复制积压缓冲区内存。	-	0~604,800 单位：秒	3,600
repl-timeout	主从同步超时时间。	单机实例不支持该参数。	30~3,600 单位：秒	60
hash-max-ziplist-entries	当hash表中记录数少于参数值，使用ziplist编码格式，节约内存。	-	1~10000	512
hash-max-ziplist-value	当hash表中各字段长度的最大值小于参数值时，使用ziplist编码格式，节约内存。	-	1~10000	64
set-max-intset-entries	当一个集合仅包含字符串且整型元素数量少于参数值时，使用intset编码格式，节约内存。	-	1~10000	512

参数名	参数解释	例外场景	取值范围	默认值
zset-max-ziplist-entries	当有序集合中记录数少于参数值，使用ziplist编码格式，节约内存。	-	1~10000	128
zset-max-ziplist-value	当有序集合中各字段长度的最大值小于参数值时，使用ziplist编码格式，节约内存。	-	1~10000	64
latency-monitor-threshold	<p>延时监控的采样时间阈值（最小值）。当阈值设置为0时，不做监控，也不采样；当阈值设置为大于0时，将监控并记录执行耗时大于阈值的操作。</p> <p>您可以通过LATENCY等命令获取统计数据 and 配置、执行采样监控。</p> <p>说明 latency-monitor-threshold参数一般在定位问题时使用。采集完latency信息，定位问题后，建议重新将latency-monitor-threshold设置为0，以免引起不必要的延迟。</p>	Proxy集群实例不支持该参数。	0~86400000 单位：毫秒	0

参数名	参数解释	例外场景	取值范围	默认值
notify-keyspace-events	键空间通知，配置该参数后客户端可以通过Redis的订阅与发布功能，来接收那些以某种方式改动了Redis数据集的事件。该参数配置为空时，功能关闭。当参数不是空字符串时，功能开启。	Proxy集群实例不支持该参数。	可配置为以下字符的任意组合，指定了服务器该发送哪些类型的通知： K: 键空间通知，所有通知以__keyspace@__为前缀。 E: 键事件通知，所有通知以__keyevent@__为前缀。 g: DEL、EXPIRE、RENAME等类型无关的通用命令的通知。 \$: 字符串命令的通知。 l: 列表命令的通知。 s: 集合命令的通知。 h: 哈希命令的通知。 z: 有序集合命令的通知。 x: 过期事件：每当有过期键被删除时发送。 e: 驱逐(evict)事件：每当有键因为maxmemory政策而被删除时发送。 A: 参数g \$lshzxe的别名。 输入的参数中至少有一个K或者E，A不能	Ex

参数名	参数解释	例外场景	取值范围	默认值
			与g\$shzxe同时出现，不能出现相同字母。例如，如果只想订阅键空间中和列表相关的通知，那么参数就应该设为Kl。将参数设为字符串"AKE"表示发送所有类型的通知。	
slowlog-log-slower-than	Redis慢查询会记录超过指定执行时间的命令。slowlog-log-slower-than用于配置记录到redis慢查询命令执行时间阈值。	-	0~1,000,000 单位：微秒	10,000
proxy-slowlog-log-slower-than	Proxy节点慢查询会记录超过指定执行时间的命令。proxy-slowlog-log-slower-than用于配置记录到proxy节点慢查询命令的执行时间阈值。	目前仅“华东-上海二”和“华南-广州”区域的Proxy集群和读写分离实例支持该参数。	30,000~2,000,000 单位：微秒	256,000
slowlog-max-len	Redis慢查询记录的条数。注意慢查询记录会消耗额外的内存。可以通过执行SLOWLOG RESET命令清除慢查询记录。	-	0~1,000	128

参数名	参数解释	例外场景	取值范围	默认值
proxy-slowlog-max-len	Proxy节点慢查询记录的条数。注意慢查询记录会消耗额外的内存。可以通过执行SLOWLOG RESET命令清除慢查询记录。	目前仅“华东-上海二”和“华南-广州”区域的Proxy集群和读写分离实例支持该参数。	0~1,000	128
auto-kill-timeout-lua-process	开启该参数时，执行超时的lua脚本进程会被自动kill。如果lua脚本执行了写操作无法kill，并且实例开启了持久化时，则该lua脚本所在的节点会自动重启，lua脚本执行的写操作将不被保存。	单机实例和Redis 3.0实例不支持该参数。	<ul style="list-style-type: none"> • yes: 开启 • no: 关闭 	no
audit-log-customer-command-list	审计日志需要额外记录的命令。审计日志默认只记录写命令，该参数可以指定用户额外需要记录的命令。只有在开启了审计日志的情况下才有效。	审计日志 目前为白名单特性，只有开启了审计日志白名单的用户，并且仅当实例类型为Proxy集群实例时，才显示该参数。	该参数配置的值必须由英文字母或与._组成的命令，且必须以英文字母开头和结尾，每个命令最大长度为10，最多支持配置10个命令。命令之间用空格隔开，配置的内容最后必须以空格结尾。	-

参数名	参数解释	例外场景	取值范围	默认值
dispatch-pubsub-to-fixed-shard	该参数用于指定发布订阅的channel是否集中在0号槽位对应的分片上。开启开关时，发布订阅的处理逻辑与单机版一致。用户轻度使用发布订阅时，推荐开启开关。用户重度使用发布订阅功能时，请使用默认配置，让订阅分担到全部分片中。	仅Proxy集群实例支持该参数。	<ul style="list-style-type: none"> • yes: 开启开关。订阅的channel都集中在0号槽位对应的集群分片上。 • no: 关闭开关。根据订阅channel值计算出hash槽位值，再将channel分散到各个槽位对应的分片中。 	no
readonly-lua-route-to-slave-enabled	是否把只读账号的只读lua路由到从节点，如果开启，只读账号的只读lua可以执行，且会路由到从节点。	仅读写分离实例支持该参数。	<ul style="list-style-type: none"> • yes: 开启 • no: 关闭 	no
cluster-sentinel-enabled	实例兼容哨兵（Sentinel）模式。	仅Proxy集群实例支持该参数。	<ul style="list-style-type: none"> • yes: 开启 • no: 关闭 	no
scan-support-wr-split	该参数关闭时，scan命令在master节点执行，该参数开启后，scan命令将在slave节点执行。开启该参数后，可以降低scan命令对主节点的压力，但对于新写入主节点的数据，可能无法第一时间在从节点查询到。	<p>仅Proxy集群实例支持该参数。</p> <p>如果历史创建的Proxy集群实例不支持该参数，请联系客服升级实例版本。</p>	<ul style="list-style-type: none"> • yes: 开启 • no: 关闭 	no

📖 说明

1. maxclients、reserved-memory-percent、client-output-buffer-slave-soft-limit、client-output-buffer-slave-hard-limit参数的默认值和取值范围与实例规格有关，因此参数模板不显示该四个参数。
2. 表6-5中的内存优化相关参数可以参考Redis官网说明，链接：<https://redis.io/topics/memory-optimization>。

----结束

6.5.2 创建 DCS 实例自定义参数模板

不同Redis版本及类型的实例对应了不同的系统默认参数模板，系统默认参数模板中包含实例的默认参数配置。用户也可以根据业务需求创建不同参数配置的自定义参数模板，在创建实例时，会根据选择的参数模板，创建对应参数配置的实例。

本节介绍如何在分布式缓存服务管理控制台创建和修改自定义参数模板。

创建自定义参数模板

步骤1 登录分布式缓存服务管理控制台。

步骤2 在管理控制台左上角单击📍，选择实例所在的区域。

步骤3 单击左侧菜单栏的“参数模板”，进入“参数模板”页面。

步骤4 选择“系统默认”或者“自定义”页签，可针对系统默认模板或已经创建好的自定义模板进行新的自定义模板创建。

- 如果选择“系统默认”，则单击需要创建实例类型的系统默认模板右侧“操作”列下的“创建为自定义模板”。
- 如果选择“自定义”，则单击需要复制的自定义模板右侧“操作”列下的“复制”。

步骤5 设置“模板名称”和“描述”。

📖 说明

模板名称长度为4到64位的字符串，以字母或者数字开头，模板名称只能包含字母、数字、中划线、下划线和点号。描述内容可以为空。

步骤6 配置参数选择“可修改参数”。

当前支持通过参数名称搜索对应的参数，直接在搜索栏输入关键字即可。

步骤7 在需要修改的配置参数对应的“参数运行值”列输入修改值。

各参数的详细介绍见表6-6，一般情况下，按照系统默认值设置参数即可。

表 6-6 Redis 缓存实例配置参数说明

参数名	参数解释	例外场景	取值范围	默认值
active-expire-num	定期删除过期键时随机检查key的数量。 如果增加该参数值，短时间内可能出现CPU使用率升高，命令时延增大，随后恢复正常。如果减小该参数值，内存中过期key数量可能上升，占用更多的内存。	Redis 3.0和Redis 6.0企业版实例不支持该参数。 说明 该参数为2021年9月新增，在此之前创建的实例，如果无法正常修改参数值，请联系客服处理。	1 ~ 1,000	20
timeout	客户端空闲N秒（timeout参数的取值）后将断开连接。参数值为0表示禁用该功能，即客户端空闲不会断开连接。	-	0~7200 单位：秒	0
appendfsync	操作系统的fsync函数刷新缓冲区数据到磁盘，有些操作系统会真正刷新磁盘上的数据，其他一些操作系统只会尝试尽快完成。	单机实例不支持该参数。	<ul style="list-style-type: none"> no：不调用fsync，由操作系统决定何时刷新数据到磁盘，性能最高。 always：每次写AOF文件都调用fsync，性能最差，但数据最安全。 everysec：每秒调用一次fsync。兼具数据安全和性能。 	no

参数名	参数解释	例外场景	取值范围	默认值
appendonly	指定是否在每次更新操作后进行日志记录，Redis在默认情况下是异步的把数据写入磁盘，如果不开启，可能会在断电时导致一段时间内的数据丢失。	单机实例不支持该参数。	<ul style="list-style-type: none"> • yes: 开启日志记录，即开启持久化功能。 • no: 关闭日志记录，即关闭持久化功能。 • only-replica: 仅开启从节点持久化功能。 <p>说明 仅Redis 4.0及以上版本的基础版主备或集群实例，或企业版高性能型主备实例支持only-replica配置项。 如果以上版本及实例类型的实例，控制台中无only-replica配置项，可以联系客服放通该配置项的白名单。</p>	yes
client-output-buffer-limit-slave-soft-seconds	主从同步缓冲区大小超过client-output-buffer-slave-soft-limit主从同步缓冲区大小软限制，并持续时间超过此参数值时，服务端会主动断开连接。该参数值配置的越小，越容易断开断连。	-	0~60 单位：秒	60

参数名	参数解释	例外场景	取值范围	默认值
client-output-buffer-slave-hard-limit	主从同步缓冲区大小硬限制。如果同步缓冲区大小超过这个值，则主从同步连接立即断开。该参数值配置的越小，越容易断开断连。	-	0 ~ 17,179,869,184 单位：字节	1,717,986,918
client-output-buffer-slave-soft-limit	主从同步缓冲区大小软限制。如果同步缓冲区大小超过这个值，并持续时间达到client-output-buffer-limit-slave-soft-seconds参数配置的秒数，则主从同步连接断开。该参数值配置的越小，越容易断开断连。	-	0 ~ 17,179,869,184 单位：字节	1,717,986,918

参数名	参数解释	例外场景	取值范围	默认值
maxmemory-policy	内存使用达到上限（maxmemory）时，对缓存数据的逐出策略，有8个取值供选择。	-	<ul style="list-style-type: none"> volatile-lru：根据LRU算法删除设置了过期时间的键值。 allkeys-lru：根据LRU算法删除任一键值。 volatile-random：删除设置了过期时间的随机键值。 allkeys-random：删除一个随机键值。 volatile-ttl：删除即将过期的键值，即TTL值最小的键值。 noeviction：不删除任何键值，只是返回一个写错误。 volatile-lfu：根据LFU算法删除设置了过期时间的键值。 allkeys-lfu：根据LFU算法删除任一键值。 <p>详情可参考Redis官网的逐出策略。</p>	volatile-lru 说明 如果是2020年7月之前创建的Redis实例，且没有修改过该参数，则默认值为noeviction。如果是2020年7月之后创建的实例，默认值都为volatile-lru。
lua-time-limit	Lua脚本的最长执行时间。	-	100 ~ 5,000 单位：毫秒	5,000

参数名	参数解释	例外场景	取值范围	默认值
master-read-only	设置实例为只读状态。设置只读后，所有写入命令将返回失败。	Proxy集群实例不支持该参数。	<ul style="list-style-type: none"> • yes • no 	no
maxclients	<p>最大同时连接的客户端个数。客户端数量超过该配置值时，会发生连接失败。该参数值配置过大时，会增加服务器处理连接的开销，影响服务器性能，命令时延增大；该参数值配置过小，可能无法完全发挥服务器的性能。</p> <p>该参数为单节点（单分片）连接数上限：</p> <ul style="list-style-type: none"> • 集群实例单节点连接数上限=实例连接数上限/分片数。 • 单机/主备实例，单节点连接数上限=实例最大连接数。 	读写分离实例暂不支持该参数。	1,000 ~ 50,000	10,000
proto-max-bulk-len	Redis协议中的最大的请求大小。该参数的配置值需要大于客户请求的长度，否则请求将无法执行。	-	1,048,576 ~ 536,870,912 单位：字节	536,870,912

参数名	参数解释	例外场景	取值范围	默认值
repl-backlog-size	用于增量同步的复制积压缓冲区大小。这是一个用来在从节点断开连接时，存放从节点数据的缓冲区，当从节点重新连接时，如果丢失的数据少于缓冲区的大小，可以用缓冲区中的数据开始增量同步。	-	16,384 ~ 1,073,741,824 单位：字节	1,048,576
repl-backlog-ttl	从节点断开后，主节点释放复制积压缓冲区内存的秒数。值为0时表示永不释放复制积压缓冲区内存。	-	0 ~ 604,800 单位：秒	3,600
repl-timeout	主从同步超时时间。	单机实例不支持该参数。	30 ~ 3,600 单位：秒	60
hash-max-ziplist-entries	当hash表中记录数少于参数值，使用ziplist编码格式，节约内存。	-	1~10000	512
hash-max-ziplist-value	当hash表中各字段长度的最大值小于参数值时，使用ziplist编码格式，节约内存。	-	1~10000	64
set-max-intset-entries	当一个集合仅包含字符串且整型元素数量少于参数值时，使用intset编码格式，节约内存。	-	1~10000	512

参数名	参数解释	例外场景	取值范围	默认值
zset-max-ziplist-entries	当有序集合中记录数少于参数值，使用ziplist编码格式，节约内存。	-	1~10000	128
zset-max-ziplist-value	当有序集合中各字段长度的最大值小于参数值时，使用ziplist编码格式，节约内存。	-	1~10000	64
latency-monitor-threshold	<p>延时监控的采样时间阈值（最小值）。当阈值设置为0时，不做监控，也不采样；当阈值设置为大于0时，将监控并记录执行耗时大于阈值的操作。</p> <p>您可以通过LATENCY等命令获取统计数据 and 配置、执行采样监控。</p> <p>说明 latency-monitor-threshold参数一般在定位问题时使用。采集完latency信息，定位问题后，建议重新将latency-monitor-threshold设置为0，以免引起不必要的延迟。</p>	Proxy集群实例不支持该参数。	0~86400000 单位：毫秒	0

参数名	参数解释	例外场景	取值范围	默认值
notify-keyspace-events	键空间通知，配置该参数后客户端可以通过Redis的订阅与发布功能，来接收那些以某种方式改动了Redis数据集的事件。该参数配置为空时，功能关闭。当参数不是空字符串时，功能开启。	Proxy集群实例不支持该参数。	可配置为以下字符的任意组合，指定了服务器该发送哪些类型的通知： K: 键空间通知，所有通知以__keyspace@__为前缀。 E: 键事件通知，所有通知以__keyevent@__为前缀。 g: DEL、EXPIRE、RENAME等类型无关的通用命令的通知。 \$: 字符串命令的通知。 l: 列表命令的通知。 s: 集合命令的通知。 h: 哈希命令的通知。 z: 有序集合命令的通知。 x: 过期事件：每当有过期键被删除时发送。 e: 驱逐(evict)事件：每当有键因为maxmemory政策而被删除时发送。 A: 参数g \$lshzxe的别名。 输入的参数中至少有一个K或者E，A不能	Ex

参数名	参数解释	例外场景	取值范围	默认值
			与g\$shzxe同时出现，不能出现相同字母。例如，如果只想订阅键空间中和列表相关的通知，那么参数就应该设为Kl。将参数设为字符串"AKE"表示发送所有类型的通知。	
slowlog-log-slower-than	Redis慢查询会记录超过指定执行时间的命令。slowlog-log-slower-than用于配置记录到redis慢查询命令执行时间阈值。	-	0~1,000,000 单位：微秒	10,000
proxy-slowlog-log-slower-than	Proxy节点慢查询会记录超过指定执行时间的命令。proxy-slowlog-log-slower-than用于配置记录到proxy节点慢查询命令的执行时间阈值。	目前仅“华东-上海二”和“华南-广州”区域的Proxy集群和读写分离实例支持该参数。	30,000~2,000,000 单位：微秒	256,000
slowlog-max-len	Redis慢查询记录的条数。注意慢查询记录会消耗额外的内存。可以通过执行SLOWLOG RESET命令清除慢查询记录。	-	0~1,000	128

参数名	参数解释	例外场景	取值范围	默认值
proxy-slowlog-max-len	Proxy节点慢查询记录的条数。注意慢查询记录会消耗额外的内存。可以通过执行SLOWLOG RESET命令清除慢查询记录。	目前仅“华东-上海二”和“华南-广州”区域的Proxy集群和读写分离实例支持该参数。	0~1,000	128
auto-kill-timeout-lua-process	开启该参数时，执行超时的lua脚本进程会被自动kill。如果lua脚本执行了写操作无法kill，并且实例开启了持久化时，则该lua脚本所在的节点会自动重启，lua脚本执行的写操作将不被保存。	单机实例和Redis 3.0实例不支持该参数。	<ul style="list-style-type: none"> • yes: 开启 • no: 关闭 	no
audit-log-customer-command-list	审计日志需要额外记录的命令。审计日志默认只记录写命令，该参数可以指定用户额外需要记录的命令。只有在开启了审计日志的情况下才有效。	审计日志 目前为白名单特性，只有开启了审计日志白名单的用户，并且仅当实例类型为Proxy集群实例时，才显示该参数。	该参数配置的值必须由英文字母或与.-_组成的命令，且必须以英文字母开头和结尾，每个命令最大长度为10，最多支持配置10个命令。命令之间用空格隔开，配置的内容最后必须以空格结尾。	-

参数名	参数解释	例外场景	取值范围	默认值
dispatch-pubsub-to-fixed-shard	该参数用于指定发布订阅的channel是否集中在0号槽位对应的分片上。开启开关时，发布订阅的处理逻辑与单机版一致。用户轻度使用发布订阅时，推荐开启开关。用户重度使用发布订阅功能时，请使用默认配置，让订阅分担到全部分片中。	仅Proxy集群实例支持该参数。	<ul style="list-style-type: none"> • yes: 开启开关。订阅的channel都集中在0号槽位对应的集群分片上。 • no: 关闭开关。根据订阅channel值计算出hash槽位值，再将channel分散到各个槽位对应的分片中。 	no
readonly-lua-route-to-slave-enabled	是否把只读账号的只读lua路由到从节点，如果开启，只读账号的只读lua可以执行，且会路由到从节点。	仅读写分离实例支持该参数。	<ul style="list-style-type: none"> • yes: 开启 • no: 关闭 	no
cluster-sentinel-enabled	实例兼容哨兵（Sentinel）模式。	仅Proxy集群实例支持该参数。	<ul style="list-style-type: none"> • yes: 开启 • no: 关闭 	no
scan-support-wr-split	该参数关闭时，scan命令在master节点执行，该参数开启后，scan命令将在slave节点执行。开启该参数后，可以降低scan命令对主节点的压力，但对于新写入主节点的数据，可能无法第一时间在从节点查询到。	<p>仅Proxy集群实例支持该参数。</p> <p>如果历史创建的Proxy集群实例不支持该参数，请联系客服升级实例版本。</p>	<ul style="list-style-type: none"> • yes: 开启 • no: 关闭 	no

📖 说明

1. maxclients、reserved-memory-percent、client-output-buffer-slave-soft-limit、client-output-buffer-slave-hard-limit参数的默认值和取值范围与实例规格有关，因此参数模板不显示该四个参数。
2. 表6-5中的内存优化相关参数可以参考Redis官网说明，链接：<https://redis.io/topics/memory-optimization>。

步骤8 单击“确定”，完成创建自定义参数模板。

----结束

修改或删除自定义模板

步骤1 登录分布式缓存服务管理控制台。

步骤2 在管理控制台左上角单击 ，选择实例所在的区域。

步骤3 单击左侧菜单栏的“参数模板”，进入“参数模板”页面。

步骤4 选择“自定义”。

步骤5 如果需要修改自定义参数模板，可以通过以下两种方式进行修改。

- 单击需要修改的自定义模板右侧“操作”列下的“编辑”。
 - a. 修改模板名称和描述。
 - b. 在“配置参数”区域，在选项框选择“可修改参数”，在需要修改的配置参数对应的“参数运行值”列输入修改值。各参数的详细介绍见表6-6，一般情况下，按照系统默认值设置参数即可。
 - c. 单击“确定”，完成修改配置参数。
- 单击自定义模板名称，进入模板的参数页面，可修改配置参数。
 - a. 配置参数选择“可修改参数”。支持通过参数名称搜索对应的参数，直接在搜索栏输入关键字即可。
 - b. 单击“修改”。
 - c. 在需要修改的配置参数对应的“参数运行值”列输入修改值。各参数的详细介绍见表6-6，一般情况下，按照系统默认值设置参数即可。
 - d. 单击“保存”，完成修改配置参数。

步骤6 如果需要删除自定义模板，单击需要删除的自定义模板右侧“操作”列下的“删除”。

单击“是”，完成删除自定义参数模板。

----结束

6.6 配置 DCS 实例标签

标签是DCS实例的标识，为DCS实例添加标签，可以方便用户识别和管理拥有的DCS实例资源。您可以在创建DCS实例时添加标签，也可以在DCS实例创建完成后，在实例的详情页添加标签。每个DCS实例最多添加20个标签。

说明

如果您的组织已经设定分布式缓存服务的相关标签策略，则需按照标签策略规则为缓存实例资源添加标签。标签不符合标签策略的规则，则可能会导致添加标签失败，请联系组织管理员了解标签策略详情。

每个标签由“标签键”和“标签值”两部分组成，“标签键”和“标签值”的命名规则如表6-7所示。

表 6-7 标签命名规则

参数名称	规则
标签键	<ul style="list-style-type: none"> 不能为空。 对于同一个实例，Key值唯一。 长度不超过128个字符。 可以包含任意语种的文字、字母、数字、空格和_ . : = + - @。 首尾字符不能为空格。 不能以_sys_开头。
标签值	<ul style="list-style-type: none"> 长度不超过255个字符。 可以包含任意语种的文字、字母、数字、空格和_ . : / = + - @。 首尾字符不能为空格。

配置实例标签

步骤1 登录[分布式缓存服务管理控制台](#)。

步骤2 在管理控制台左上角单击 ，选择实例所在的区域。

步骤3 单击左侧菜单栏的“缓存管理”，进入实例信息页面。

步骤4 在需要查看的DCS缓存实例左侧，单击该实例的名称，进入该实例的基本信息页面。

步骤5 选择“实例配置 > 标签”，进入标签页面，界面显示该实例的标签列表。

步骤6 您可以根据实际需要，执行以下操作：

- **添加标签**

a. 单击“添加/编辑标签”，弹出“添加/编辑标签”窗口。

如果您已经预定义了标签，在“标签键”和“标签值”中选择已经定义的标签键值对。另外，您可以单击的“查看预定义标签”，系统会跳转到标签管理服务页面，查看已经预定义的标签，或者创建新的标签。

您也可以直接在“标签键”和“标签值”中输入设置标签。

b. 单击“确定”。为实例添加标签成功。

- **修改标签**

单击“添加/编辑标签”，弹出“添加/编辑标签”窗口，若想修改某标签键，在“添加/编辑标签”窗口中先删除该键，再添加该键，输入想要修改的标签值，单击“添加”。

- **删除标签**

单击标签所在行“操作”列下的“删除”，如果确认删除，在弹出的“删除标签”窗口，单击“是”。

----结束

6.7 重命名 DCS 实例高危命令

Redis实例创建之后，支持修改部分高危命令，命令修改后，仅修改者知晓修改后的命令，其他用户执行原始命令时会被拦截报错，以此可以限制高危命令的使用。

当前支持重命名的高危命令有command、keys、flushdb、flushall、hgetall、scan、hscan、sscan、和zscan，Proxy集群实例还支持dbsize和dbstats命令重命名，其他命令暂时不支持。

📖 说明

- 仅Redis 4.0及以上版本的实例，支持命令重命名功能。
- 重命名的过程中会自动重启实例，请谨慎操作。
- 重命名操作完成后立即生效。因为涉及安全性，页面不会显示重命名后的命令。
- 同一个命令支持多次重命名，每次新的重命名操作都会覆盖之前的重命名命令，如果需要恢复高危命令或忘记重命名后的命令，重新执行命令重命名即可。
- 命令不能重命名为除本命令外的其他原始命令，例如，keys命令可以命名为keys本身或非命令abc123，但不可以重命名为scan等其他原始命令。
- 重命名的命令必须以字母开头，长度范围为4~64个字符，且只能包含字母、数字、中划线和下划线。

重命名高危命令

步骤1 登录[分布式缓存服务管理控制台](#)。

步骤2 在管理控制台左上角单击 ，选择实例所在的区域。

步骤3 单击左侧菜单栏的“缓存管理”。

步骤4 在需要进行重命名命令的缓存实例右侧，单击“操作”列下的“更多 > 命令重命名”。

步骤5 在“命令重命名”对话框中，选择需要重命名的高危命令，并输入重命名名称，单击“确定”。

在“命令重命名”对话框中单击“添加重命名命令”，可以同时为多个高危命令进行重命名。重命名后，您可以在“后台任务”页面查看重命名的操作记录。

图 6-3 命令重命名

<input type="checkbox"/>	序号	任务名	ID	用户名	状态	启动时间	结束时间	详细信息	操作
<input type="checkbox"/>	1	命令重命名			成功	2023/12/19 17:08:37 GMT+08:00	2023/12/19 17:08:54 GMT+08:00	变更前: command	删除

----结束

6.8 导出 DCS 实例列表

DCS管理控制台支持导出并下载DCS实例的信息列表，提供用户查看或对比DCS实例信息，实例列表的导出形式为Excel。

导出 DCS 实例列表

- 步骤1** 登录[分布式缓存服务管理控制台](#)。
- 步骤2** 在管理控制台左上角单击 ，选择实例所在的区域。
- 步骤3** 单击左侧菜单栏的“缓存管理”。
- 步骤4** 在“缓存管理”页面单击“导出”，默认导出全部实例列表。如需导出部分实例列表，先勾选需要导出的实例，再单击“导出”。
- 步骤5** 页面跳转至“任务中心”，当“导出缓存实例列表”任务执行成功后，单击任务右侧的“下载”即可下载缓存实例列表。

图 6-4 导出缓存实例列表

名称	ID	状态	可用区	缓存类型	实例类型	规格 (GB)	已用/可用连接地址	创建时间	计费方式	虚拟私有	虚拟私有	企业项目
dcx-trpt	5e4f4c58-	运行中	可用区1	Redis 5.0	单机	0.125	0/128 (0% 198.19.32.	2019/05/2	免费	null	null	default
dcx-APITe	693491b0-	运行中		Redis 3.0	主备	2	2/1,536 (0 172.16.14	2019/05/0	包年/包月	null	52267da0-	default

----结束

6.9 切换 DCS 实例的主备节点

DCS管理控制台支持手动切换DCS主备实例和读写分离实例的主备节点，该操作用于特殊场景。例如，释放所有业务连接或终止当前正在执行的业务操作。

集群实例不支持此操作，如果需要手动切换Proxy集群和Cluster集群分片的主备节点，请在实例详情的“节点管理”（或“分片与副本”）页签中操作，具体操作可参考[管理DCS实例分片与副本](#)。

须知

- 主备节点切换期间，业务会发生少于10秒的连接闪断，请在操作前确保应用具备断连重建能力。
- 主备节点切换时，新的主备关系同步需要消耗较多资源，请不要在业务繁忙时执行该操作。
- 由于主备之间数据同步采用异步机制，主备节点切换期间可能丢失少量正在操作的数据。
- 实例主备切换后，IP地址不变，不需要切换客户端连接地址。

前提条件

只有当DCS缓存实例处于“运行中”状态，才能执行此操作。

切换 DCS 实例的主备节点

- 步骤1 登录[分布式缓存服务管理控制台](#)。
 - 步骤2 在管理控制台左上角单击 ，选择实例所在的区域。
 - 步骤3 单击左侧菜单栏的“缓存管理”。
 - 步骤4 在需要进行主备切换的缓存实例右侧，单击“操作”列下的“更多 > 主备切换”。
 - 步骤5 当主备切换任务显示“成功”时，主备切换完成。
- 结束

6.10 管理 DCS 实例分片与副本

本节主要介绍如何查询主备、集群、读写分离实例的分片与副本信息，以及将集群实例的从节点手动升级为主节点的操作。

- 主备和读写分离实例，分片数为1，默认是一个一主一从的双副本架构，支持通过“节点管理”查看分片信息，如果需要手动切换主从节点，请执行[切换DCS实例的主备节点](#)操作。
- 对于有多个从副本的主备实例，还可以通过“节点管理”设置主备切换优先级，或摘除从副本的域名IP（仅当包含多个从副本时支持该操作，摘除域名IP后，通过只读域名访问实例时，返回的信息中不包括摘除的副本IP）。
- Proxy集群、Cluster集群实例，每个集群是由多个分片组成，每个分片默认是一个双副本架构，您可以通过“节点管理”查看分片信息，还可以根据业务需要，手动切换分片主从节点。

说明

- 该特性目前在控制台上最新名称为“节点管理”，在部分区域为“分片与副本”，请以控制台实际显示为准。
- 仅Redis 4.0及以上版本的实例支持该特性。
- 如果是单机实例，仅更新为“节点管理”的区域支持该特性。
- 不同实例规格对应的分片数，具体请参考[Redis Cluster集群实例介绍](#)和[Redis Proxy集群介绍](#)。
- 创建集群实例后，如需调整分片数，可通过[变更规格](#)，选择需要的实例规格。

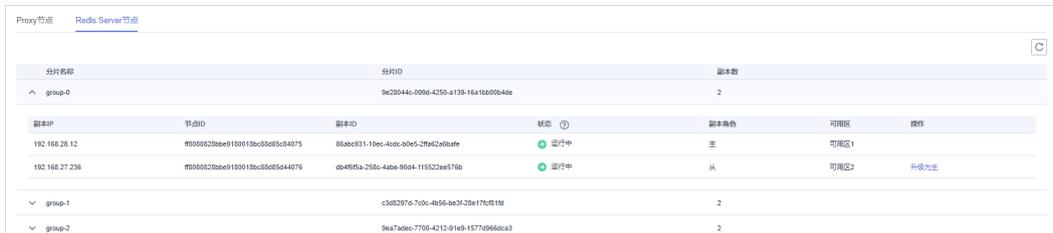
管理 DCS 实例分片与副本

- 步骤1 登录[分布式缓存服务管理控制台](#)。
- 步骤2 在管理控制台左上角单击 ，选择实例所在的区域。
- 步骤3 单击左侧菜单栏的“缓存管理”，进入实例信息页面。
- 步骤4 单击缓存实例名称，进入该实例的基本信息页面。
- 步骤5 单击“节点管理”或“分片与副本”，进入实例的节点管理页面。

界面显示该实例的所有分片列表，包含“分片名称”、“分片ID”和该分片下的“副本数”。

步骤6 单击分片名称前面的  图标，展开当前分片下的所有副本。包含“副本IP”、“节点ID”、“副本ID”、“状态”“副本角色”和副本所在的“可用区”。

图 6-5 节点管理（集群实例）



分片名称	分片ID	副本数				
group-0	9c28044c-0990-4250-a139-16a1b000040e	2				
副本IP	节点ID	副本ID	状态	副本角色	可用区	操作
192.168.28.12	#000023bee910001b0c585d5c34075	66abc931-11ec-4c0c-b0e5-2f6a2a0a0afe	运行中	主	可用区1	
192.168.27.236	#000023bee910001b0c585d5d44076	db4f65a-250c-4a8e-90d4-11552ae571b	运行中	从	可用区2	升级为主
group-1	c3d82976-70bc-4b56-ba3f-23e17c50110	2				
group-2	9ea7a6ec-7700-4212-91a9-157f49966ca3	2				

图 6-6 节点管理（主备实例）



分片名称	分片ID	副本数					
group-0	f5165904-62ba-4ca1-aa99-0a9746079208	2					
副本IP	节点ID	副本ID	状态	副本角色	可用区	主备切换优先级	操作
10.0.0.179	#0000023bee910001b0c582a56c400a	6cece0f0-9054-40be-9ca8-0736011afe0c	运行中	主	可用区1		
10.0.0.186	#0000023bee910001b0c582a678400b	c5a2a74-3dc0-4d9c-8c29-d5ed1d967ae	运行中	从	可用区2	100	摘除域名IP

图 6-7 节点管理（单机实例）



分片名称	分片ID	副本数			
group-0	7b9a0e38-0b60-45cf-ae21-d63e18400504	1			
副本IP	节点ID	副本ID	状态	副本角色	可用区
10.0.0.36	#0000023bee910007901b0a0b0594b4c2d		运行中	主	可用区1

您还可以对副本执行以下操作：

● 集群实例

展开集群实例单分片信息后，单击从副本右侧的“升级为主”，可以将该分片下的从副本升级为主副本。

 说明

如果是Proxy集群实例，您还可以在节点管理的“Proxy节点”页签查看实例的Proxy节点信息（节点IP、节点ID、节点名称）。其他实例类型不涉及“Proxy节点”。

● 主备/读写分离实例

a. 如果主备实例的从副本数多于1个，单击“摘除域名IP”，可以摘除对应从副本（只读副本）的IP，摘除成功后，只读域名不会再解析到该副本IP。

如果主备实例只有1个从副本，则不支持摘除域名。

b. 对于有多个从副本的主备或读写分离实例，单击“主备切换优先级”列对应的 ，可以设置主备切换优先级。

当主节点故障以后，系统会按照您指定的优先级，自动切换到优先级最高的从节点上。如果优先级相同，则系统会内部进行选择 and 切换。优先级为0-100，1-100优先级逐步降低，1为最高，100为最低，0为禁止倒换。

● 单机实例

单机实例仅一个副本，在“节点管理”页面可以查看该实例的节点信息，不支持其他操作。

----结束

7 备份恢复实例数据

7.1 DCS 备份恢复概述

介绍如何通过管理控制台对DCS缓存实例进行数据备份，以及备份数据恢复。

备份缓存数据的必要性

业务系统日常运行中可能出现一些小概率的异常事件，比如异常导致缓存实例出现大量脏数据，或者在实例出现故障后持久化文件不能重新加载。部分可靠性要求非常高的业务系统，除了要求缓存实例高可用，还要求缓存数据安全、可恢复，甚至永久保存。

DCS支持将当前时间点的实例缓存数据备份并存储到对象存储服务（OBS）中，以便在缓存实例发生异常后能够使用备份数据进行恢复，保障业务正常运行。

备份方式

DCS缓存实例支持自动和手动两种备份方式。

- 自动备份

您可以通过管理控制台设置一个定时自动备份策略，在指定时间点将实例的缓存数据自动备份存储。

自动备份频率以天为单位，您根据需要，选择每周备份一次或多次。备份数据保留最多7天，过期后系统自动删除。

自动备份主要目的在于让实例始终拥有一个完整的数据副本，在必要时可以及时恢复实例数据，保证业务稳定，实例数据安全多一重保障。

- 手动备份

除了定时备份，DCS还支持由用户手动发起备份请求，将实例当前缓存数据进行备份，并存储到OBS服务中。

您在执行业务系统维护、升级等高危操作前，可以先行备份实例缓存数据。

备份过程对实例的影响

备份操作是在备节点执行，备份期间不影响实例正常对外提供服务。

在主备节点全量数据同步或者实例高负载的场景下，数据同步需要一定的时间，在数据同步没有完成的情况下开始备份，备份数据与主节点最新数据相比，有一定延迟。

在实例备节点进行备份期间，如果主节点有新的数据写入，备份文件不会包含备份期间的数据变化。

备份的其他说明

- 支持备份的实例类型
 - Redis的“主备”、“Proxy集群”、“Cluster集群”和“读写分离”实例支持数据备份与恢复功能，“单机”实例暂不支持。单机实例若需要备份，可参考[Redis单机实例使用Redis-cli工具备份](#)，使用redis-cli工具导出RDB文件。
 - Memcached的“主备”实例支持数据备份与恢复功能，“单机”实例暂不支持。

- 备份原理

Redis 3.0实例采用的是AOF文件进行持久化，Redis 4.0及以上基础版实例，手动备份支持选择RDB格式和AOF格式进行持久化；自动备份仅支持RDB格式进行持久化。

如果需要导出Redis 3.0的RDB备份文件，可以通过redis-cli导出，使用命令：
`redis-cli -h {redis_address} -p 6379 -a {password} --rdb {output.rdb}`。

说明

- DCS Redis 3.0已下线，暂停售卖，建议使用Redis 4.0及以上版本。
- 放通了SYNC命令的Redis 3.0单机实例可以通过执行此命令将RDB文件导出；Redis 3.0 proxy实例由于架构的原因，不支持放通SYNC命令，因此不能导出RDB文件。

备份任务在备节点执行，DCS通过将备节点的数据持久化文件压缩并转移到OBS服务中存储，从而实现实例数据备份。

DCS以小时为单位，定期检查所有实例的备份策略，对于需要执行备份的实例，启动备份任务。

- 备份时间点的选择
建议选择业务量少的时间段进行备份。
- 备份文件的存储
备份文件存储在对象存储服务（OBS）中。
- 自动备份异常的处理
自动备份任务触发后，如果实例当前正在进行重启、扩容等操作，则定时任务顺延到下一时间段处理。
实例备份失败或者其他任务正在进行而推迟备份，DCS会在下一时间段继续尝试备份，一天最多会尝试三次。
- 备份数据保存期限
自动备份产生的备份文件根据您设置的策略保留1-7天，超期由系统自动删除，但至少会保留最近一次的数据备份记录。
如果备份数量没有超出最大限制（24个），手动备份数据不会自动清除，您可根据需要手动删除备份数据。

📖 说明

- 自动和手动备份记录总数最多不超过24个，当备份记录超过24个时，自动删除最早的备份记录。
- 当删除实例时，备份数据会随实例删除，如果需要保存备份数据，请提前将备份数据下载保存。
- 删除所有备份文件，会影响备份文件相关能力，如故障时执行备份恢复，请谨慎操作。

数据恢复

- 数据恢复流程
 - a. 您通过控制台发起数据恢复请求。
 - b. DCS从对象存储服务（OBS）获取数据备份文件。
 - c. 暂停实例数据读写服务。
 - d. 替换主实例的持久化文件。
 - e. 重新加载新的持久化文件。
 - f. 完成数据恢复，对外提供数据读写服务。
- 数据恢复对业务系统的影响
恢复操作是将备份文件在主节点执行，实例数据恢复期间需暂停数据读写服务，直到主实例完成数据恢复。
- 数据恢复异常处理
数据恢复文件如果被损坏，DCS在恢复过程中会尝试修复。修复成功则继续进行数据恢复，修复失败，DCS主备实例会将实例还原到执行恢复前的状态。

7.2 自动备份 DCS 实例数据

本节介绍如何在DCS管理控制台设置自动备份策略。设置完成后，系统将根据备份策略定时备份实例数据。

DCS的“自动备份”默认为关闭状态，如需开启自动备份，请参考本章节进行操作。单机实例不支持“备份与恢复”功能。

如果不需要自动备份，可以修改备份策略设置，关闭自动备份。

前提条件

已成功申请DCS主备、集群或读写分离缓存实例，且实例处于运行中状态。

配置自动备份策略

步骤1 登录[分布式缓存服务管理控制台](#)。

步骤2 在管理控制台左上角单击 ，选择实例所在的区域。

步骤3 单击左侧菜单栏的“缓存管理”，进入实例信息页面。

“缓存管理”支持通过筛选来查询对应的缓存实例。支持的筛选条件有“名称”、“规格”、“ID”、“IP地址”、“可用区”、“状态”、“实例类型”、“缓存类型”等。

步骤4 在需要备份的DCS缓存实例左侧，单击实例名称，进入实例的基本信息页面。

步骤5 单击“备份与恢复”页签，进入备份恢复管理页面。

步骤6 单击“自动备份”右侧的 ，打开自动备份开关，显示备份策略信息。

表 7-1 备份策略参数说明

参数	说明
备份周期	自动备份频率。 可设置为每周的某一天或者某几天，按实际需要适当增加备份频率。
保留天数	备份数据保存期限。 保存天数可选1~7，超过期限后，备份数据将被永久删除，无法用来恢复实例。
开始时间	自动备份任务执行时间。时间格式：00:00~23:00间的任意整点时间。 每小时检查一次备份策略，如果符合备份策略设置的开始时间，则执行备份操作。 说明 实例备份大约耗时5~30分钟，备份期间发生的数据新增或修改记录，将不会保存到备份数据中。为了尽量减少备份对业务的影响，备份开始时间建议设置在业务交易较少的时间段。 实例只有处于“运行中”状态时，系统才对其执行数据备份。

步骤7 设置好备份参数，单击“确定”，完成备份策略设置。

开启自动备份后，也支持关闭自动备份开关，或单击“修改”，修改备份策略。

步骤8 实例将在设置的备份时间自动执行备份，并在该页面查看备份记录。

备份完成后，单击备份记录后的“下载”，“恢复”，或“删除”，即可执行相关操作。

----结束

7.3 手动备份 DCS 实例数据

当您需要即时备份DCS缓存实例中的数据，可以通过手动备份功能完成实例数据备份。本节介绍如何在DCS管理控制台手动备份主备缓存实例的数据。

如果备份数量没有超出最大限制（24个），手动备份数据不会自动清除，您可根据需要手动删除备份数据。

📖 说明

- 自动和手动备份记录总数最多不超过24个，当备份记录超过24个时，自动删除最早的备份记录。
- 当删除实例时，备份数据会随实例删除，如果需要保存备份数据，请提前将备份数据下载保存。
- 删除所有备份文件，会影响备份文件相关能力，如故障时执行备份恢复，请谨慎操作。

前提条件

已成功申请DCS主备、集群或读写分离缓存实例，且实例处于运行中状态。

手动备份 DCS 实例数据

步骤1 登录[分布式缓存服务管理控制台](#)。

步骤2 在管理控制台左上角单击 ，选择实例所在的区域。

步骤3 单击左侧菜单栏的“缓存管理”，进入实例信息页面。

“缓存管理”支持通过筛选来查询对应的缓存实例。支持的筛选条件有“名称”、“规格”、“ID”、“IP地址”、“可用区”、“状态”、“实例类型”、“缓存类型”等。

步骤4 在需要备份的DCS缓存实例左侧，单击实例名称，进入实例的基本信息页面。

步骤5 单击“备份与恢复”页签，进入备份与恢复管理页面。

步骤6 单击“手动备份”，弹出手动备份窗口。

步骤7 选择备份格式，支持选择RDB格式或AOF格式。

📖 说明

Redis 4.0、5.0、6.0基础版和企业版高性能型实例支持选择RDB或AOF备份格式，企业版存储型实例仅支持RDB格式，Redis 3.0实例仅支持RDB格式。

AOF格式的备份会优先在从节点备份，从节点AOF会被重写。

步骤8 单击“确定”，开始执行手工备份任务。

备注说明最长不能超过128个字节。

📖 说明

实例备份需耗时10~15分钟，备份期间发生的数据新增或修改记录，将不会保存到备份数据中。

----结束

7.4 恢复 DCS 实例数据

本节介绍如何在DCS管理控制台将实例已备份的记录恢复到本实例中。例如在实例数据误删除的场景，您可以通过该操作恢复实例数据。

如果需要将备份数据迁移到其他DCS实例中，请参考[使用备份文件离线迁移DCS Redis实例](#)。

约束与限制

- Proxy集群支持**开启或关闭多DB**，开启多DB期间的备份数据，不支持恢复到关闭多DB后的Proxy集群中。
- 如果实例进行了如下内容的变更操作，则实例变更前的备份文件无法恢复到变更后的实例。
 - 实例缩容
 - 集群实例水平扩容
 - 实例类型变更（主备实例变更为读写分离实例除外）

前提条件

- 已成功申请DCS主备、集群或读写分离缓存实例，且实例处于运行中状态。
- 实例已有历史数据备份，且备份状态为“成功”。

恢复 DCS 实例数据

步骤1 登录[分布式缓存服务管理控制台](#)。

步骤2 在管理控制台左上角单击，选择实例所在的区域。

步骤3 单击左侧菜单栏的“缓存管理”，进入实例信息页面。

“缓存管理”支持通过筛选来查询对应的缓存实例。支持的筛选条件有“名称”、“规格”、“ID”、“IP地址”、“可用区”、“状态”、“实例类型”、“缓存类型”等。

步骤4 在需要备份的DCS缓存实例左侧，单击实例名称，进入实例的基本信息页面。

步骤5 单击“备份与恢复”页签，进入备份恢复管理页面。

页面下方显示历史备份数据列表。

步骤6 选择需要恢复的历史备份数据，单击右侧的“恢复”，弹出实例恢复窗口。

步骤7 单击“确定”，开始执行实例恢复任务。

备注说明最长不能超过128个字节。

您可以在“恢复记录”页签查询当前实例恢复任务执行结果，该页签会展示实例的所有恢复记录，不支持删除恢复记录。

说明

- 实例恢复需耗时1~30分钟。
- 恢复过程中，实例会有一段时间不能处理客户端的数据操作请求，当前数据将被删除，待恢复完成后存储原有备份数据。

----结束

7.5 下载 DCS 实例备份文件

由于自动备份和手动备份实例有一定的限制性（自动备份的文件在系统最大保留天数为7天，手动备份会占用OBS空间），您可将实例的rdb和aof备份文件下载，本地永久保存。

当前仅支持将主备、读写分离、或者集群实例的备份文件下载，单机实例不支持备份恢复功能。单机实例若需要下载备份文件，可参考[导出单机实例rdb备份文件](#)，使用redis-cli工具导出rdb文件。

以下仅针对主备、读写分离和集群实例：

- 如果是Redis 3.0，支持aof格式持久化，支持在控制台下载导出aof格式的备份文件，如果需要导出rdb，可以通过redis-cli导出，使用命令：`redis-cli -h {redis_address} -p 6379 -a {password} --rdb {output.rdb}`。

说明

DCS Redis 3.0已下线，暂停售卖，建议使用Redis 4.0及以上版本。

- 如果是Redis 4.0及以上版本，支持aof和rdb格式持久化，可以在控制台下载导出aof和rdb格式的备份文件。
- 对于多分片的集群实例，下载的备份文件为多个，每个分片对应一个备份文件。

前提条件

实例已做备份且没有过期。

下载 DCS 实例备份文件

步骤1 登录[分布式缓存服务管理控制台](#)。

步骤2 在管理控制台左上角单击，选择实例所在的区域。

步骤3 单击左侧菜单栏的“缓存管理”，进入DCS缓存实例信息页面。

“缓存管理”页面支持通过搜索栏筛选对应的缓存实例。

步骤4 在需要查看的DCS缓存实例左侧，单击实例名称，进入实例的基本信息页面。

步骤5 单击“备份与恢复”页签，进入备份恢复管理页面。

页面下方显示历史备份数据列表。

步骤6 选择需要下载的历史备份数据，单击右侧的“下载”，弹出下载备份文件窗口。

步骤7 选择通过URL或OBS方式下载备份文件。

- URL下载

图 7-1 URL 下载



- a. 设置URL有效期并单击“查询”按钮，系统将生成临时的备份文件URL，URL超过有效期将会失效，需要重新生成新的URL。
URL有效期设置范围为5分钟到12小时之间。
- b. 在URL列表，单击“下载”，下载备份文件。
如果您的账号开启了**敏感操作保护**，备份文件下载前会弹出“操作确认”窗口进行验证，如**图7-3**。如果未开启敏感操作保护，将直接下载备份文件到本地。

📖 说明

- 单击“复制全部URL”或单击URL下载链接后的复制图标，可以复制URL下载链接。
 - 如果通过URL的下载链接，在Linux系统中使用wget命令获取备份文件，则需要将下载链接使用**英文引号**括起来。如：

```
wget 'https://obsEndpoint.com:443/redisdemo.rdb?parm01=value01&parm02=value02'
```

原因是URL中携带符号：&，wget命令识别URL参数会出现异常，需要使用英文引号辅助识别完整URL。
 - 请妥善保管备份文件和URL下载链接，避免数据泄露。
- OBS下载

图 7-2 OBS 下载



- 选择OBS下载页签，单击“准备下载”下的“OBS Browser+”链接，根据指导下载OBS Browser+。
- 安装OBS Browser+客户端，使用实例所在华为云账号登录OBS Browser+，登录方式请参考[登录OBS Browser+](#)。
- 在OBS Browser+客户挂载外部桶，挂载外部桶的操作请参考[配置挂载外部桶](#)。
添加挂载时，请使用“挂载外部桶”下的桶名称。
- 单击添加挂载的外部桶名称，进入该外部桶。在外部桶中查找备份文件，查找方式请参考[搜索文件或文件夹](#)。
“下载备份文件”下的备份文件路径即备份文件所在的文件夹名称和备份文件名。
- 单击备份文件右侧的下载图标，下载备份文件。

----结束

开启敏感操作保护（可选）

下载备份文件支持敏感操作保护。开启操作保护后，下载备份文件时，需要输入验证码进行验证，从而提升客户数据安全性。

📖 说明

- 只有管理员可以设置敏感操作，普通IAM用户只有查看权限，不能对其进行设置，如需修改，请联系管理员为您操作或添加权限。
- 关于敏感操作的更多说明请参考[敏感操作](#)。

步骤1 登录华为云控制台，鼠标移动至右上方的用户名，在下拉列表中选择“安全设置”。

步骤2 在安全设置页面选择“敏感操作”页签。

步骤3 单击“操作保护”项对应的“立即启用”，即可开启操作保护。

步骤4 开启操作保护后，在下载DCS备份文件时，会弹出“操作确认”窗口，需要完成身份验证后，才可以下载备份文件。

图 7-3 操作确认

操作确认

您已开启操作保护。为了防止您的资源被无意操作，请在下方输入您的验证码。
如需关闭操作保护，请在账号安全设置“敏感操作”中关闭。[关闭操作保护](#)

验证方式 手机 邮箱 虚拟MFA ?

邮箱名 s@...com [修改](#)

验证码 [获取验证码](#)

[确定](#) [取消](#)

----结束

8 变更实例

8.1 变更 DCS 实例规格

DCS管理控制台支持变更Redis和Memcached缓存实例规格，即变更实例的类型、内存规格或副本数，您可以根据实际需要，选择合适的实例规格。

说明

- **执行实例变更规格，建议在业务低峰期操作。**业务高峰期（如实例在内存利用率、CPU利用率达到90%以上或写入流量过大）变更规格可能会失败，若变更失败，请在业务低峰期再次尝试变更。
- 如果实例创建时间非常早，由于实例版本没有升级而无法兼容规格变更（扩容/缩容）功能，请联系客服将缓存实例升级到最新版本，升级后就可以支持规格变更（扩容/缩容）功能。
- 实例变更规格，不会影响实例的连接地址、访问密码、数据、及安全组/白名单配置等信息，也不需要重启实例。

实例类型变更须知

表 8-1 DCS 实例类型变更明细

实例版本	支持的实例变更类型	变更须知及影响
Redis 3.0	单机实例变更为主备实例	连接会有秒级中断，大约1分钟左右的只读。
	主备实例变更为Proxy集群实例	1. 如果Redis 3.0主备实例数据存储在多DB上，或数据存储在非DB0上，不支持变更为Proxy集群；数据必须是只存储在DB0上的主备实例才支持变更为Proxy集群。 2. 连接会中断，5~30分钟只读。
Memcached	单机实例变更为主备实例	会有秒级业务中断、大约1分钟只读。

实例版本	支持的实例变更类型	变更须知及影响
Redis 4.0/5.0/6.0	主备实例或读写分离实例变更为Proxy集群实例	<ol style="list-style-type: none"> 1. 变更为proxy集群时，需要评估proxy集群的多DB使用限制和命令使用限制对业务的影响。具体请参考proxy集群使用多DB限制，实例受限使用命令。 2. 变更前实例的已用内存必须小于变更后最大内存的70%，否则将不允许变更。 3. 如果变更前实例的已用内存超过总内存的90%，变更的过程中可能会导致部分key逐出。 4. 变更完成后需要对实例重新创建告警规则。 5. 如果原实例是主备实例，请确保应用中没有直接引用只读IP或只读域名。 6. 请确保您的客户端应用具备重连机制和处理异常的能力，否则在变更规格后有可能需要重启客户端应用。 7. 变更规格过程中会有秒级业务中断、大约1分钟只读，建议在业务低峰时进行变更。
	Proxy集群实例变更为主备实例或读写分离实例	
Redis 4.0/5.0/6.0	主备实例变更为读写分离实例 说明 读写分离实例暂不支持直接变更为主备实例。	<ol style="list-style-type: none"> 1. 目前只支持主备实例变更为相同容量的读写分离实例，小于4G规格的主备实例不支持变更为读写分离实例。 2. 如果变更前实例的已用内存超过总内存的90%，变更的过程中可能会导致部分key逐出。 3. 变更完成后需要对实例重新创建告警规则。 4. 请确保主备实例的应用中没有直接引用只读IP或只读域名。 5. 请确保您的客户端应用具备重连机制和处理异常的能力，否则在变更规格后有可能需要重启客户端应用。 6. 变更规格过程中会有秒级业务中断，建议在业务低峰时进行变更。 7. 主备实例如果创建了ACL账号，不支持变更为读写分离实例。 8. Redis 6.0如果开启了SSL链路加密传输，不支持变更为读写分离实例。

除了上表中提到的实例外，其他实例类型目前不支持实例类型的变更。若您想实现跨实例类型的规格变更，建议您创建新的实例后进行数据迁移并交换实例IP，迁移操作请参考[使用迁移任务在线迁移Redis实例](#)进行操作。

实例类型变更后支持的命令，请参考对应的[开源命令兼容性](#)。

实例规格变更须知

- 支持实例规格变更明细如下：

表 8-2 实例规格变更明细

缓存类型	单机实例	主备实例	Cluster集群实例	Proxy集群实例	读写分离实例
Redis 3.0	支持扩容和缩容	支持扩容和缩容	-	仅支持扩容	-
Redis 4.0	支持扩容和缩容	支持扩容、缩容和副本数变更	支持扩容、缩容和副本数变更	支持扩容和缩容	支持扩容、缩容和副本数变更
Redis 5.0	支持扩容和缩容	支持扩容、缩容和副本数变更	支持扩容、缩容和副本数变更	支持扩容和缩容	支持扩容、缩容和副本数变更
Redis 6.0 基础版	支持扩容和缩容	支持扩容、缩容和副本数变更	支持扩容、缩容和副本数变更	支持扩容和缩容	支持扩容、缩容和副本数变更
Redis 6.0 企业版	-	支持扩容和缩容	-	-	-
Memcached	支持扩容和缩容	支持扩容和缩容	-	-	-

📖 说明

- Redis 3.0和Memcached实例在预留内存不足的情况下，内存用满可能会导致扩容失败，具体可参考[预留内存](#)。
- 副本数变更和容量变更不支持同时进行，需分开两次执行变更。
- 删除副本时，每次操作仅支持删除一个副本。
- **实例规格变更的影响：**

表 8-3 实例规格变更的影响

实例类型	规格变更类型	实例规格变更的影响
单机、主备和读写分离实例	扩容/缩容	<ul style="list-style-type: none"> • Redis 4.0及以上版本基础版实例，扩容期间连接会有秒级中断，大约1分钟的只读，缩容期间连接不会中断。 • Redis 3.0实例，规格变更期间连接会有秒级中断，5~30分钟只读。 • Redis企业版实例，规格变更期间连接会有秒级中断，大约1分钟的只读。 • 如果是扩容，只扩大实例的内存，不会提升CPU处理能力。 • 单机实例不支持持久化，变更规格不能保证数据可靠性。在实例变更后，需要确认数据完整性以及是否需要再次填充数据。如果有重要数据，建议先把数据用迁移工具迁移到其他实例备份。 • 主备和读写分离实例缩容前的备份记录，缩容后不能使用。如有需要请提前下载备份文件，或缩容后重新备份。

实例类型	规格变更类型	实例规格变更的影响
Proxy和Cluster集群实例	扩容/缩容	<ul style="list-style-type: none"> ● 水平扩容（分片数增加）： <ul style="list-style-type: none"> - 连接不中断，但会占用CPU，导致性能有20%以内的下降。 - 分片数增加时，会新增数据节点，数据自动负载均衡到新的数据节点，访问时延会增大。 ● 水平缩容（分片数减少）： <ul style="list-style-type: none"> - 分片数减少时，会删除节点。Cluster集群实例缩容前，请确保应用中没有直接引用这些删除的节点，否则可能导致业务访问异常。 - 删除节点会导致连接闪断，请确保您的客户端应用具备重连机制和处理异常的能力，否则在变更规格后可能需要重启客户端应用。 ● 垂直扩容（分片数不变，分片容量增加）： <ul style="list-style-type: none"> - 如果节点所在的虚拟机内存容量不足，会发生节点迁移，迁移时业务连接会有闪断和只读。 - 如果虚拟机内存容量充足，则直接扩大节点容量，对业务无影响。 <p>说明 Redis 3.0版本集群实例不支持垂直扩缩容。</p> <ul style="list-style-type: none"> ● 垂直缩容（分片数不变，分片容量减少）：无影响。 ● 实例缩容前，每个节点的已用内存要小于缩容后节点最大内存的70%，否则将不允许变更。 ● 实例规格变更期间，可能会进行数据迁移，访问时延会增大。Cluster集群请确保客户端能正常处理MOVED和ASK命令，否则会导致请求失败。 ● 实例规格变更期间，如果有大批量数据写入导致节点内存写满，将会导致变更失败。 ● 在实例规格变更前，请先使用缓存分析中的大key分析，确保实例中没有大key存在，否则在规格改变后，节点间进行数据迁移的过程中，单个key过大（≥512MB）会触发Redis内核对于单key的迁移限制，造成数据迁移超时失败，进而导致规格变更失败，key越大失败的概率越高。 ● Cluster集群实例扩容或缩容时，请确保客户端开启集群拓扑自动刷新配置，否则在变更后需要重启客户端。Lettuce客户端开启集群拓扑自动刷新配置请参考Lettuce客户端连接Cluster集群实例中的示例。 ● 实例规格变更前的备份记录，变更后不能使用。如有需要请提前下载备份文件，或变更后重新备份。

实例类型	规格变更类型	实例规格变更的影响
主备、读写分离和Cluster集群实例	副本数变更	<ul style="list-style-type: none"> Cluster集群实例增加或删除副本时，请确保客户端开启集群拓扑自动刷新配置，否则在变更后需要重启客户端。Lettuce客户端开启集群拓扑自动刷新配置请参考Lettuce客户端连接Cluster集群实例中的示例。 删除副本会导致连接中断，需确保您的客户端应用具备重连机制和处理异常的能力，否则在删除副本后需要重启客户端应用。增加副本不会连接中断。 当副本数已经为实例支持的最小副本数时，不支持删除副本。

变更实例

步骤1 登录[分布式缓存服务管理控制台](#)。

步骤2 在管理控制台左上角单击，选择实例所在的区域。

步骤3 单击左侧菜单栏的“缓存管理”。

步骤4 在需要规格变更的实例右侧，单击“操作”列下的“更多 > 变更规格”，进入到变更实例规格页面。

步骤5 在变更实例规格页面中，选择您需要变更的目标规格。

集群实例如需扩容单分片容量，请参考[集群实例单分片扩容](#)。

步骤6 选择变更时间为“立即变更”或“可维护时间窗内进行变更”。

“可维护时间窗内进行变更”适用于如下**变更规格时存在客户端连接中断的场景**。

表 8-4 变更规格时存在客户端连接中断的场景

变更规格任务	客户端连接中断的场景
单机或主备实例扩容	从8G以下扩容到8G或8G以上时
Proxy或Cluster集群实例缩容	分片数减少时
变更实例类型	主备/读写分离与Proxy集群之间实例类型变更
删除副本	主备/Cluster集群/读写分离实例删除副本

说明

- 不涉及客户端连接中断的场景，选择在可维护时间窗内变更，也会立即变更。
- 提交变更规格后，不支持取消变更，可以修改“维护时间窗”时间推迟变更（变更过程中，维护时间窗可修改次数不超过3次）。
- Redis 3.0和Memcached变更实例时，仅支持“立即变更”。
- 在“维护时间窗”内变更的实例，变更的起始时间点是在维护时间窗时段内的随机时间，不是维护时间窗的起始时间。
- 集群实例缩容需要迁移的数据量过大时，缩容完成的时间可能会超出可维护时间窗。

步骤7 单击“下一步”，确认变更详情，并查看风险检查结果。

当实例风险检查提示异常时，实例有变更失败的风险，请参考表8-5进行处理。

表 8-5 风险检查项说明

风险检查项	风险检查原因	检查结果异常的处理建议
非标配置检查 说明 <ul style="list-style-type: none"> • 目前仅北京四、上海一、上海二等部分区域支持非标配置检查，具体区域以控制台实际检查结果为准。 • 支持检查的非标场景： <ul style="list-style-type: none"> - 实例单节点带宽非标 - 实例单节点内存非标 - Cluster集群副本数非标 - Proxy集群Proxy节点数量非标 - Proxy集群最大连接数maxclients非标（超出最大可配连接数） 	当检查实例存在非标配置项时，会提示您当前实例存在非标配置，变更规格会转换为标准的DCS实例配置。 其中，只有当非标检查结果为非标准带宽或非标准Proxy节点数时，您可以选择实例变更后保留原非标带宽或Proxy节点数，其他非标配置不支持保留。	<ul style="list-style-type: none"> • 如果实例不存在非标配置，检查正常，无需处理。 • 如果实例存在非标配置，请根据提示选择是否继续变更，或是否需要保留非标带宽/非标Proxy集群Proxy节点数。
节点状态	实例节点状态异常会导致实例变更失败。	如果提示节点状态异常，请联系客服。
数据集内存分布检查 说明 该检查项只针对Proxy集群和Cluster集群实例。	Redis集群实例变更规格过程中会进行节点间的数据迁移，如果存在大Key（大于512MB），会触发Redis内核对于单Key的迁移限制，造成数据迁移超时失败。 当节点间实例数据集内存分布不均，且差值大于512MB，表示实例有大Key，有变更失败的风险。	当提示节点可能存在大Key时，建议先 处理大Key 后，再进行实例变更。

风险检查项	风险检查原因	检查结果异常的处理建议
内存利用率检查	当节点内存利用率过高 (>90%) 时, 在变更规格过程中可能导致Key逐出或变更失败。	提示内存利用过高时, 建议通过优化大Key、过期Key扫描、或删除部分Key等方式优化内存。
网络输入流量检查 说明 该检查项只针对单机、主备、读写分离实例。	网络输入流量过高, 写缓冲区溢出, 可能导致规格变更失败。	如提示网络输入流量过高, 请在业务低峰期进行变更。
CPU利用率检查	检查五分钟内的节点CPU利用率是否过高 (>90%)。节点CPU利用率过高时, 可能导致规格变更失败。	如提示CPU利用率是否过高, 建议在业务低峰期进行变更。 Redis实例CPU利用率高问题排查和解决
资源容量 说明 该检查项只针对集群实例扩容分片容量的场景。	集群实例扩容分片容量时, 如果实例所在虚拟机资源容量不足, 变更过程中需要节点迁移, 迁移节点时, 业务连接会有闪断和只读。	如果资源容量检查存在风险, 请确保您的客户端应用具备重连机制和处理异常的能力, 否则在变更规格后可能需要重启客户端应用。

📖 说明

- 检查结果正常, 说明所检查项不存在变更失败的风险。
- 当提示检查失败时, 可能是以下两方面原因:
 - 连接实例主节点获取信息失败, 建议检查实例状态是否正常。
 - 系统异常, 建议稍后“重新检查”。
- 检查过程中单击“停止检查”, 检查会终止, 如需重新检查, 请单击“重新检查”。
- 手动停止检查或检查结果提示异常时, 如需继续执行规格变更, 需要勾选“我已知晓风险”。

步骤8 风险检查正常后, 单击“提交订单”, 开始变更DCS缓存实例。提交变更后, 在界面上可以选择跳转到[后台任务](#)界面, 查看变更任务的状态。

单击后台任务页面中的任务名称, 可以查看任务详情。实例规格变更成功后, 实例状态切换为“运行中”。

图 8-1 查看后台任务详情



📖 说明

- 当单机实例规格变更失败时，实例暂不可用，实例规格仍然为变更前的规格，部分管理操作（如参数配置、规格变更等）暂不支持，待后台完成变更处理后，实例将自动恢复正常，实例规格将更新为变更后的规格。
- 当主备和集群实例规格变更失败时，实例规格仍然为变更前的规格，部分管理操作（如参数配置、备份恢复、规格变更等）暂不支持，请按照变更前的规格使用，避免因数据超过规格而被丢失。
- 当规格变更成功时，您可以按照新的规格使用DCS缓存实例。
- DCS单机、主备和读写分离缓存实例规格变更大约需要5到30分钟，集群实例规格变更所需时间稍长。

----结束

8.2 调整 DCS 实例带宽

Redis实例作为更靠近应用服务的数据层，通常会执行较多的数据存取并消耗网络带宽。当实例带宽不足时，可能会产生流控，导致业务延迟增大，客户端连接异常等问题。目前，Redis 4.0及以上版本的实例，支持通过控制台调整Redis实例带宽，用于适配业务对带宽值的不同需求。

约束与限制

- 企业版Redis暂不支持调整带宽。
- 只有在运行中的实例支持调整带宽，如果是变更中、故障中、重启中等其他状态下的实例不支持调整实例带宽。
- 带宽的调整范围在实例基准带宽到最大可调整的带宽之间，通常在实例节点所在物理机带宽资源充足的前提下，实例可调整的单分片最大带宽为2048 Mbit/s。
- 实例带宽与单分片带宽的关系如下：
 - 单机/主备实例带宽=单分片带宽。
 - 读写分离实例带宽=单分片带宽 * 副本数。
 - 集群实例带宽=单分片带宽 * 分片数，当各分片带宽值不同时，集群实例带宽值为各个分片带宽值之和。

调整 DCS 实例带宽

- 步骤1 登录[分布式缓存服务管理控制台](#)。
- 步骤2 在管理控制台左上角单击 ，选择实例所在的区域。
- 步骤3 单击左侧菜单栏的“缓存管理”。
- 步骤4 在“缓存管理”页面，单击DCS缓存实例的名称。
- 步骤5 在缓存实例的“基本信息”栏中单击带宽后的“调整带宽”。

图 8-2 调整带宽



步骤6 在“调整带宽”页面，设置带宽参数。

实例默认为手动调整带宽的方式，请根据需要设置目标带宽值。开启“自动弹性带宽调整”功能白名单后，实例带宽调整方式支持选择“手动调整”或“自动弹性带宽调整”（如果控制台不支持选择带宽调整方式，请联系客服开启“自动弹性调整带宽”功能白名单）。

说明

- 目标带宽值只支持设置为8的整数倍，并且在可设置的范围内。如果设置的带宽值不是8的整数倍，订单提交后会按照向下取整的方式调整带宽。例如输入的带宽值为801，则按照800 Mbit/s的目标带宽调整带宽。
- 变更页面显示的变更后费用为该实例额外购买的带宽计费金额，不包含原实例费用。
- 调整带宽的计费方式仅支持按需计费。
- 您可以根据需要多次调整带宽，单个计费周期（1小时）中如果有多次带宽变更，该计费周期以最大带宽费用收费。例如将一个Redis实例（默认带宽值为256 Mbit/s）的带宽变更为2048 Mbit/s后，在一个计费周期内再次将带宽值变更为512 Mbit/s，实例在该计费周期将按照2048 Mbit/s的带宽值扣费。
- 手动调整**
手动调整实例的目标带宽值。集群实例多个分片需要调整带宽时，可以对多个分片单独设置不同的目标带宽，也可以同时勾选多个分片后，单击页面左上角的“批量调整带宽”，统一设置带宽值。

图 8-3 手动设置新带宽值



- 自动弹性带宽调整
需要开启“自动带宽扩展”并根据需要设定自动带宽扩容策略，如表8-6。
系统会根据您设定的自动带宽扩容策略自动执行带宽扩展，如果触发带宽扩展，单分片最高可扩展至2048Mbit/s。如果您已对实例带宽进行过手动调整，自动弹性带宽调整结果会覆盖已手动调整的结果。

表 8-6 设定自动带宽扩容策略

带宽扩容策略	说明
瞬时带宽使用率不小于	触发带宽自动扩容的瞬时带宽使用率阈值，单位：%。 例如该阈值设置为70%时，则当分片的瞬时带宽使用率达到70%时，会触发该分片带宽自动扩容。
观测窗口	带宽弹性的观测窗口，单位：分钟。默认值：1。 例如观测窗口时间设置为1分钟时，则带宽监控数据取值为1分钟内的监控数据。
静默时间	扩容操作的静默时间，单位：秒。默认值：0。 当带宽自动扩容后，如果再次监测到瞬时带宽使用率超过阈值，实例在设置的静默时间内不会立即扩容，设置静默时间可以避免实例连续进行带宽自动扩容。

步骤7 确认新的带宽值及带宽费用，在“带宽调整确认”处勾选确认后，单击“提交订单”。

调整带宽任务的状态为“成功”后，新的带宽值立即生效。

----结束

8.3 变更 DCS 集群实例为多可用区

为满足单可用区（主、备节点在相同可用区）集群实例实现跨可用区容灾部署的需求，DCS支持将单可用区集群实例通过迁移备节点可用区的方式，变更为多可用区（主、备节点在不同可用区）的集群实例。

约束与限制

- 仅副本数 ≥ 2 的单可用区集群实例支持该功能，其他场景均不支持变更实例可用区。
- 升级Proxy集群实例可用区须知：
 - 迁移可用区过程中会有秒级业务中断，建议在业务低峰时进行变更。
 - 请确保您的客户端应用具备重连机制和处理异常的能力，否则在迁移可用区后有可能需要重启客户端应用。
- 升级Cluster集群实例可用区须知：
 - 迁移可用区过程不会影响主节点，业务不会中断，但是性能会略有下降，建议在业务低峰时进行。
 - 迁移可用区过程会导致部分副本连接中断，需确保您的客户端应用具备重连机制和处理异常的能力。

变更集群实例为多可用区

步骤1 登录[分布式缓存服务管理控制台](#)。

步骤2 在管理控制台左上角单击 ，选择实例所在的区域。

步骤3 单击左侧菜单栏的“缓存管理”。

步骤4 在“缓存管理”页面，单击需要升级为多可用区的DCS缓存实例的名称。

步骤5 在缓存实例的“网络信息”栏中单击“可用区”后的“升级实例支持多可用区”。

图 8-4 升级实例支持多可用区



步骤6 在弹出的“迁移可用区”窗口，选择“备可用区”。

图 8-5 选择备可用区



步骤7 选择“变更时间”为“立即变更”或“可维护时间窗内进行变更”。

步骤8 单击“确定”提交变更。

当变更状态显示为“成功”时，变更完成。

----结束

9 管理实例生命周期

9.1 重启 DCS 实例

在使用DCS实例过程中，如果需要手动重启实例，例如实例内存碎片率较高或实例故障，希望通过重启实例尝试恢复时，您可以使用DCS管理控制台“重启”功能。该功能支持批量重启DCS实例。

约束与限制

- 只有当DCS缓存实例处于“运行中”或“故障”状态，才能执行此操作。
- 单机实例或关闭了AOF持久化（参数appendonly配置为no）的主备、集群、读写分离实例，在重启实例后，实例中原有的数据将被清空，请谨慎操作。
- 在重启DCS缓存实例过程中，您无法对实例进行读写操作。
- 在重启DCS缓存实例过程中，如果有正在进行的备份操作，备份任务会终止，也可能会重启失败。
- 重启DCS缓存实例会断开原有客户端连接，建议在应用中配置自动重连。

重启 DCS 实例

步骤1 登录[分布式缓存服务管理控制台](#)。

步骤2 在管理控制台左上角单击 ，选择实例所在的区域。

步骤3 单击左侧菜单栏的“缓存管理”。

步骤4 勾选“名称”列下需要重启的DCS缓存实例左侧的方框（可选择一个或多个缓存实例），单击信息栏左上侧的“重启”。

如果重启单个实例，也可以在需要重启的DCS缓存实例右侧，单击“操作”列下的“重启”。

步骤5 单击“是”，重启DCS缓存实例。DCS缓存实例重启成功后，缓存实例状态切换为“运行中”。

重启DCS缓存实例大约需要**10秒到30分钟**，具体需要时长与实例的缓存大小有关。

📖 说明

默认情况只会重启实例进程。如果勾选“强制重启”，Redis 3.0、Memcached实例将会重启虚拟机，Redis 4.0及以上版本实例不支持强制重启，将会重启实例进程。

----结束

9.2 关闭/启动 DCS 实例

Redis 4.0及以上版本的实例，支持实例关机操作，实例关机后将停止数据读写，并且无法进行规格变更、参数配置、密码修改、缓存分析、实例备份、和在线迁移等操作。

📖 说明

- Redis实例默认状态为“运行中”，当实例关机后，运行状态为“已关闭”。
- Redis实例关机后，计费不受影响，与运行中的实例计费一致。

关闭/启动 DCS 实例

步骤1 登录[分布式缓存服务管理控制台](#)。

步骤2 在管理控制台左上角单击 ，选择实例所在的区域。

步骤3 单击左侧菜单栏的“缓存管理”。

步骤4 关闭或启动DCS实例。

- 关闭实例
 - a. 在需要关机的缓存实例右侧，单击“操作”列下的“更多>关机”，或勾选需要关机的缓存实例后，单击实例列表上方的“更多>关机”。
 - b. 在弹出的“关机”窗口确认无误后单击“是”。当实例状态为“已关闭”时，实例关机完成。
- 开启实例
 - a. 状态为“已关闭”的实例如需开机，单击对应实例“操作”列下的“启动”，或勾选需要开机的实例，单击实例列表上方的“更多 > 启动”。
 - b. 在弹出的“启动”窗口确认无误后单击“是”。当实例状态为“运行中”时，实例开启完成。

----结束

9.3 删除 DCS 实例

DCS管理控制台支持删除DCS缓存实例，且可批量删除DCS缓存实例、一键式删除创建失败的DCS缓存实例。

约束与限制

- DCS缓存实例已存在，且实例状态为运行中、故障、已关闭时才能执行删除操作。

- DCS缓存实例删除后，实例中原有的数据和备份数据将被删除，且删除的实例无法恢复，如需保留数据备份记录，请在删除实例前，将实例的备份文件下载到本地永久保存。
- 如果是集群实例，会将实例的所有节点删除。
- 包年/包月的实例，不支持直接删除。

删除 DCS 实例

删除创建成功的实例

步骤1 登录[分布式缓存服务管理控制台](#)。

步骤2 在管理控制台左上角单击 ，选择实例所在的区域。

步骤3 单击左侧菜单栏的“缓存管理”。

步骤4 勾选“名称”列下的需要删除的DCS缓存实例左侧的方框，可选一个或多个。

DCS缓存实例状态为创建中、启动中、停止中、重启中时不允许执行删除操作。

步骤5 单击实例上方信息栏的“更多 > 删除”。

步骤6 根据提示输入“DELETE”，并单击“是”，完成删除缓存实例。

删除DCS缓存实例大约需要1到30分钟。

说明

如果只需要删除单个DCS缓存实例，也可以在“缓存管理”界面，单击需要删除的DCS缓存实例右侧“操作”列下的“更多 > 删除”。

----结束

删除创建失败的实例

步骤1 登录[分布式缓存服务管理控制台](#)。

步骤2 在管理控制台左上角单击 ，选择实例所在的区域。

步骤3 单击左侧菜单栏的“缓存管理”。

若当前存在创建失败的DCS缓存实例，界面信息栏会显示“创建失败的实例”及失败数量信息。

步骤4 单击“创建失败的实例”后的图标或者数量。

弹出“创建失败的实例”界面。

步骤5 在“创建失败的实例”界面删除创建失败的DCS缓存实例。

- 单击“全部删除”按钮，一键式删除所有创建失败的DCS缓存实例。
- 单击需要删除的DCS缓存实例右侧的“删除”，依次删除创建失败的DCS缓存实例。

----结束

9.4 清空 DCS 实例数据

在使用DCS实例过程中，如果需要清空实例中的数据，除了可以在连接实例后通过flushdb或flushall命令清空数据，DCS管理控制台还提供了“数据清空”功能，本章节向您介绍如何通过该功能一键式清空实例数据。

须知

清空实例数据后，Redis无法提供数据访问加速，可能导致业务时延陡增。

约束与限制

- 只有Redis 4.0及以上版本的实例，且实例状态处于“运行中”时，才支持“数据清空”功能。
- 数据清空操作无法撤销，且数据被清空后将无法恢复，请谨慎操作。

清空 DCS 实例数据

步骤1 登录[分布式缓存服务管理控制台](#)。

步骤2 在管理控制台左上角单击 ，选择实例所在的区域。

步骤3 单击左侧菜单栏的“缓存管理”。

步骤4 勾选“名称”列下的相应实例名称左侧的方框，可选一个或多个。

步骤5 单击信息栏左上侧的“更多 > 数据清空”。

步骤6 单击“是”，清空实例数据。

----结束

10 分析诊断实例

10.1 查找 Redis 实例大 Key 和热 Key

大Key和热Key是Redis使用中经常遇到的问题，本章节主要介绍DCS管理控制台的大Key和热Key分析功能，通过该功能可以分别监控到Redis实例中占用空间最大的Key和存储数据中被访问最多的Key。

- 大Key可以分为两种情况：
 - Key的Value占用存储空间较大。一般单个String类型的Key大小达到10KB，或者集合类型的Key总大小达到50MB，则被定义为大Key。
 - Key的元素数量较多。一般集合类型的Key中元素超过5000个，则被定义为大Key。
- 热Key通常指当一个Key的访问频率或资源占用显著高于其他Key时，则称之为热Key。例如：
 - 某个集群实例一个分片每秒处理10000次请求，其中有3000次都是操作同一个Key。
 - 某个集群实例一个分片的总带宽使用（入带宽+出带宽）为100Mbits/s，其中80Mbits是由于对某个Hash类型的Key执行HGETALL所占用。

约束与限制

大Key分析使用限制和说明：

- 在大Key分析时，会遍历Redis实例中的所有Key，因此分析所需要时间取决于Key的数量。
- 如果要进行大key分析，建议在业务低谷期间进行，且不要与配置的自动备份时间重叠。
- 如果是主备、读写分离和集群实例，大Key分析是对备节点的分析，对实例性能影响较小。如果是单机实例，由于只有一个节点，是对主节点进行分析，客户访问性能会略有影响。
- 对于大Key分析任务，每个Redis实例默认最多保存100次分析任务的记录，当超过100条记录时会默认删除最老的分析记录，而存入最新的记录。同时，支持用户在控制台上手动删除无用的大Key分析记录。

热Key分析使用限制和说明：

- 只有Redis 4.0及以上版本的实例支持热Key分析。
- 实例maxmemory-policy参数必须配置为allkeys-lfu或者volatile-lfu。
- 在热Key分析时，会遍历Redis实例中的所有Key，因此分析所需要时间取决于Key的数量。
- 配置自动热key分析时，不要在业务高峰期进行，避免影响业务，同时也不要过了高峰期太久，避免分析结果不准确。
- 热Key分析是对实例主节点的分析，在进行分析时，客户访问性能会略有影响。
- 对于热Key分析任务，每个Redis实例默认最多保存100次分析任务的记录。当超过100条记录时会默认删除最老的分析记录，而存入最新的记录。同时，支持用户在控制台上手动删除无用的热Key分析记录。

说明

执行大Key和热Key分析会消耗CPU，建议在业务低峰时段执行大Key和热Key分析，降低CPU被用满的可能。

分析查找大 Key

步骤1 登录[分布式缓存服务管理控制台](#)。

步骤2 在管理控制台左上角单击  ，选择实例所在的区域。

步骤3 单击左侧菜单栏的“缓存管理”，进入实例信息页面。

步骤4 单击需要缓存分析的Redis实例名称，进入该实例的基本信息页面。

步骤5 选择“分析与诊断 > 缓存分析”进入“缓存分析”页面。

步骤6 在“大Key分析”页签下，您可以选择立即对实例进行大Key分析或者设置定时任务，每日自动分析。

步骤7 当分析任务结束后，单击分析列表“操作”列的“查看”，您可以查询当前实例不同类型的大Key分析结果。

单击分析列表“操作”列的“下载”或“删除”，也可以下载或删除分析结果。

说明

大Key分析结果显示数据大小最大的前100条Key记录（string类型显示20条，list/set/zset/hash类型显示80条）。

图 10-1 查看大 Key 分析结果（string 类型）

查看详情

任务ID 249d47bf-61f8-40df-9789-985645f390ef 开始时间 2024/07/08 11:22:26 GMT+08:00

状态 ✔ 成功 结束时间 2024/07/08 11:22:57 GMT+08:00

string类型 list/set/zset/hash类型

共20条记录 🔍

Key 名称	类型	大小(Bytes)	Database
normal-873	String	3,128	0
normal-2294	String	3,128	0
normal-1949	String	3,128	0
normal-1616	String	3,128	0
normal-45	String	3,128	0
normal-130	String	3,128	0
normal-1663	String	3,128	0
normal-1347	String	3,128	0
normal-2815	String	3,128	0
normal-796	String	3,128	0

图 10-2 查看大 Key 分析结果（list/set/zset/hash 类型）

查看详情

任务ID 249d47bf-61f8-40df-9789-985645f390ef 开始时间 2024/07/08 11:22:26 GMT+08:00

状态 ✔ 成功 结束时间 2024/07/08 11:22:57 GMT+08:00

string类型 list/set/zset/hash类型

共80条记录 🔍

Key 名称	类型	大小(Bytes)	数量(Counts)	Database
stream-858	Stream	88323	24	0
stream-476	Stream	73670	15	0
hash-858	Hash	68875	24	0
hash-476	Hash	66688	15	0
hash-1325	Hash	65694	12	0
hash-2210	Hash	65694	12	0
stream-322	Stream	64782	15	0
stream-1325	Stream	63143	12	0
stream-757	Stream	62456	18	0
858	Zset	59417	24	0

表 10-1 大 Key 分析结果参数说明

参数名称	参数说明
Key名称	大Key分析结果中Key的名称。
类型	Key的类型，包括String、Hash、List、Set、ZSet等数据类型。
大小	Key的Value大小，单位为：Bytes。
数量	Key中元素的数量，仅list/set/zset/hash类型的数据显示该参数。单位为：Counts。
Database	Key所在的Database。

----结束

分析查找热 Key

- 步骤1** 登录[分布式缓存服务管理控制台](#)。
- 步骤2** 在管理控制台左上角单击，选择实例所在的区域。
- 步骤3** 单击左侧菜单栏的“缓存管理”，进入实例信息页面。
- 步骤4** 单击需要缓存分析的Redis实例名称，进入该实例的基本信息页面。
- 步骤5** 选择“分析与诊断 > 缓存分析”进入“缓存分析”页面。
- 步骤6** 在“热Key分析”页签下，您可以选择立即对实例进行热Key分析或者设置定时任务，每日自动分析。

说明

如果是2020年7月之前创建的Redis实例，且没有修改过该参数，则maxmemory-policy默认值为noeviction，如果是2020年7月之后创建的实例，maxmemory-policy默认值为volatile-lru。您需要先将maxmemory-policy参数配置为allkeys-lfu或者volatile-lfu，才能执行热Key分析。您可以在“实例配置>参数配置”下修改参数。allkeys-lfu和volatile-lfu的具体介绍请参考[逐出策略](#)。

- 步骤7** 当分析任务结束后，单击分析列表“操作”列的“查看”，可以查询当前实例的热Key分析结果。

单击分析列表“操作”列的“下载”或“删除”，也可以下载或删除分析结果。

说明

热Key分析结果，显示在分析时间段内被访问频度最高的前100条Key记录。

图 10-3 查看热 Key 分析结果

查看详情

任务ID: 71200223-bada-4905-842f-9bfc3bea6ea 开始时间: 2024/07/08 11:21:45 GMT+08:00

状态: ✔ 成功 结束时间: 2024/07/08 11:21:47 GMT+08:00

共100条记录 🔍

Key 名称	类型	大小	频度 ?	Database
json350	Dcsjson	15,073 Bytes	8	0
json43	Dcsjson	1,998 Bytes	8	0
json214	Dcsjson	23,261 Bytes	8	0
json172	Dcsjson	28,178 Bytes	8	0
json20	Dcsjson	22,544 Bytes	8	0
json258	Dcsjson	388 Bytes	7	0
json16	Dcsjson	3,480 Bytes	7	0
json62	Dcsjson	2,867 Bytes	7	0
json104	Dcsjson	8,673 Bytes	7	0
json329	Dcsjson	11,244 Bytes	7	0
json191	Dcsjson	16,123 Bytes	7	0

表 10-2 热 Key 分析结果参数说明

参数名称	参数说明
Key名称	热Key分析结果中Key的名称。
类型	Key的类型，包括String、Hash、List、Set、Zset等数据类型。
大小	Key的Value的大小，单位为：Bytes。
频度	表示Key在一段时间（一般指1分钟）的访问频度，会随着访问的频率而变化。 该值并不是简单的访问频率值，而是一个基于概率的对数计数器结果，最大为255（可表示100万次访问），超过255后如果继续频繁访问该值并不会继续增大，同时默认如果每过一分钟没有访问，该值会衰减1。
分片	Key所在的分片。 说明 只有集群实例显示该参数。
Database	Key所在的DB。

----结束

大 Key 和热 Key 的常见问题

- 问题1: [Redis Cluster集群实例级别容量和性能未达到瓶颈，但某个分片容量或性能已过载?](#)
- 问题2: [单个Key负载过大（热Key），有什么影响?](#)
- 问题3: [为了减少大Key和热Key过大，有什么使用建议?](#)
- 问题4: [为什么Redis 3.0实例没有热Key功能?](#)
- 问题5: [如何提前发现大Key和热Key?](#)

10.2 扫描并删除 Redis 实例的过期 Key

在开源Redis的键空间中，有两种删除Key的方式。

- 使用DEL等命令直接对Key进行删除。
- 使用类似于EXPIRE等命令对Key设置过期时间，当达到过期时间时，Redis键空间中的Key将不可访问。对于设置了过期时间的Key，当达到过期时间时，Redis不会立即对Key进行删除，由于Redis当前主线程仍然为单线程，故Redis设计了几种机制对已经过期的Key进行内存释放：
 - 惰性删除：Redis的删除策略由主循环中的判断逻辑进行控制，所有Key读写命令执行之前都会调用函数对其进行检查，如果过期，则删除该键，然后返回Key不存在的结果；未过期则不做操作，继续执行原有的命令。
 - 定期删除：由Redis的定时任务函数实现，该函数以一定的频率运行，每次运行时，都从键空间中随机取出一定数量的Key进行检查，并删除其中的过期Key。开源Redis不是每次定时任务都会检查所有的Key，而是随机检查一定数量的Key（默认一次从设置过期时间的Key中随机检查20个，每秒10次），该机制旨在防止阻塞Redis主进程太久而造成业务阻塞，所以会造成已过期的Key释放内存速度较慢。

基于开源Redis以上机制，分布式缓存服务提供了一种通用的“过期Key扫描”的方式，来定时释放所有已经过期Key占用的内存，通过自行配置定时任务，在任务执行期间，会对所有缓存实例的主节点进行扫描操作，扫描操作会遍历整个实例的键空间，触发Redis引擎中对Key过期的判断，从而释放已过期的Key。

📖 说明

建议在业务低峰时段执行过期Key扫描，降低CPU被用满的可能。

约束与限制

只有Redis 4.0及以上版本实例支持过期key扫描，企业版实例不支持。

DCS不支持查询已释放的过期Key。

扫描并删除 Redis 实例的过期 Key

- 步骤1** 登录[分布式缓存服务管理控制台](#)。
- 步骤2** 在管理控制台左上角单击 ，选择实例所在的区域。
- 步骤3** 单击左侧菜单栏的“缓存管理”，进入实例信息页面。

步骤4 单击需要缓存分析的Redis实例名称，进入该实例的基本信息页面。

步骤5 选择“分析与诊断 > 缓存分析”进入缓存分析页面。

步骤6 在“过期key扫描”页签下，可以执行过期key扫描释放掉过期的key。

- 单击“立即扫描”，可立即对实例执行手动过期key扫描。
- 开启“自动扫描”，通过设置定时任务，到设定时间将会执行自动扫描。自动扫描的配置说明请参考[表10-3](#)和[自动扫描性能说明和配置建议](#)。

表 10-3 自动扫描配置参数

参数名称	参数说明
首次扫描时间	设定的第一次扫描时间，须设定在当前时间之后。 取值格式：YYYY/MM/DD hh:mm:ss
扫描间隔	从首次扫描时间开始，每隔一个时间间隔，便启动一次扫描。 <ul style="list-style-type: none"> • 如果到达启动时刻，上一次扫描还未结束，则本次轮空。 • 启动扫描的时间有五分钟冗余量，即超过本次启动时刻，不足五分钟，仍然会启动，不至于轮空。 说明 连续扫描可能使cpu占用率较高，建议根据实例中key总量以及key增长情况来配置，可参考 自动扫描性能说明和配置建议 。 取值范围：0~43,200 默认值：1440 单位：分
扫描超时	此参数的目的在于避免因不可知原因造成的扫描超时，导致后面的定时任务无法执行。设定此参数，超过超时时间后，返回失败，以便能继续进行下一轮扫描。 <ul style="list-style-type: none"> • 超时时间不少于扫描间隔时间的2倍。 • 可根据每次过期key扫描的时间，以及使用场景所能承受的最大超时时间，设定一个经验值。 取值范围：1~86,400 默认值：2880 单位：分
迭代扫描key数量	SCAN命令用于迭代当前数据库中的key集合。COUNT选项的作用就是让用户告知迭代命令，在每次迭代中应该从数据集里返回多少元素。具体参见 scan命令介绍 。迭代式扫描可降低一次扫描过多key而造成扫描时间过长，影响redis性能的问题。 举例：redis中有1000万个key，迭代扫描key数量设为1000，则迭代10000次可完成全库扫描。 取值范围：10~1,000 默认值：10 单位：个

步骤7 当过期key扫描任务提交后，每次过期key扫描都会生成一个任务记录，通过任务记录可以查看扫描的任务ID、状态、扫描方式、扫描开始和结束的时间。

图 10-4 过期 key 扫描任务列表

任务ID	状态	方式	开始时间	结束时间
82d31b-5d11-4788-b218-acc529284a45	成功	批量扫描	2021-09-06 10:00:08 GMT+08:00	2021-09-06 10:00:11 GMT+08:00
8f95275-22f1-43bc-8c5c-1a04827d7618	成功	批量扫描	2021-09-06 09:59:56 GMT+08:00	2021-09-06 09:59:57 GMT+08:00
c35859c-af7a-4121-895c-57d3ba998983	成功	批量扫描	2021-09-06 09:59:57 GMT+08:00	2021-09-06 09:59:58 GMT+08:00
9895881-ad41-4d7b-8244-18d4f18f1e0f	成功	批量扫描	2021-09-06 07:00:07 GMT+08:00	2021-09-06 07:00:09 GMT+08:00
43a28f3-5383-41e7-a20f-835c55a4a40c	成功	批量扫描	2021-09-06 06:00:08 GMT+08:00	2021-09-06 06:00:10 GMT+08:00
405a736-5831-4639-9115-4678a75a8853	成功	批量扫描	2021-09-06 05:01:06 GMT+08:00	2021-09-06 05:01:08 GMT+08:00
03848bc-c879-43d3-864a-a8d2761a3a30	成功	批量扫描	2021-09-06 04:00:07 GMT+08:00	2021-09-06 04:00:09 GMT+08:00
b0c5f02-d712-4270-96d7-478a33011311	成功	批量扫描	2021-09-06 03:00:07 GMT+08:00	2021-09-06 03:00:10 GMT+08:00
8f10606-6f8c-4413-8889-8e6684035112	成功	批量扫描	2021-09-06 02:00:07 GMT+08:00	2021-09-06 02:00:08 GMT+08:00
ad08f10-93ac-47bc-af0f-59888a752c27	成功	批量扫描	2021-09-06 01:00:08 GMT+08:00	2021-09-06 01:00:10 GMT+08:00

说明

扫描失败的两种情况：

- 出现异常导致扫描失败。
- 扫描超时导致失败，可能是key数量太多，未能在超时时间内扫描完，也会失败，但其实已经删除了部分key了。

---结束

自动扫描性能说明和配置建议

性能说明：

- 数据面底层SCAN扫描间隔5ms，相当于1秒钟扫描200次。迭代扫描key数量设为10/50/100/1000时，每秒钟扫描2000/10000/20000/200000个key。
- 每秒钟扫描key数量越大，cpu占用率也相应增加。

测试参考：

使用主备实例测试，在有1000万不过期和500万过期的key，过期时间为1-10秒的场景下，完成一次全库扫描，测试数据如下：

说明

以下测试结果仅供参考，不同局点环境和网络波动等客观条件可能产生差异。

- 自然删除，每秒删除1万条过期key，删除500万过期key，耗时约为8分钟，cpu占用率约为5%。
- “迭代扫描key数量”设为10，耗时约为 1500万/0.2万/60秒 = 125分，cpu占用率约为8%。
- “迭代扫描key数量”设为50，耗时约为 1500万/1万/60秒 = 25分，删除key时cpu占用率约10%。
- “迭代扫描key数量”设为100，耗时约为 1500万/2万/60秒 = 12.5分，删除key时cpu占用率约20%。
- “迭代扫描key数量”设为1000，耗时约为 1500万/20万/60秒 = 1.25分，删除key时cpu占用率约为25%。

配置建议：

- 您可根据实例中key总量以及key增长情况，来配置迭代扫描key数量和扫描间隔。

- 如测试参考中，1500万key总量，“迭代扫描key数量”设为10，扫完一遍需约125分，那么扫描间隔建议设置4小时以上。
- 如果希望提高扫描速度，那么可以将“迭代扫描key数量”设为100，扫完一遍需约12.5分，那么扫描间隔建议设置30分钟以上。
- 迭代扫描key数量越大，扫描速度越快，cpu占用率也相应增加，用户要平衡耗时和cpu占用率。
- 如果过期key数量增长速度不快，可以一天执行一次过期key扫描。

说明

建议将扫描开始时间设置为业务低峰期。将扫描间隔设为1天，超时时间设为2天。

10.3 诊断 Redis 实例

当您的Redis实例发现故障、性能有问题时，您可以通过实例诊断功能，及时获取实例诊断项目异常的原因、影响以及处理建议。

约束与限制

- Redis 3.0、Memcached实例不支持实例诊断。
- 新创建的缓存实例，请在实例创建成功10分钟后，再执行实例诊断，否则可能会诊断失败。

诊断 Redis 实例

步骤1 登录[分布式缓存服务管理控制台](#)。

步骤2 在管理控制台左上角单击 ，选择实例所在的区域。

步骤3 单击左侧菜单栏的“缓存管理”，进入实例信息页面。

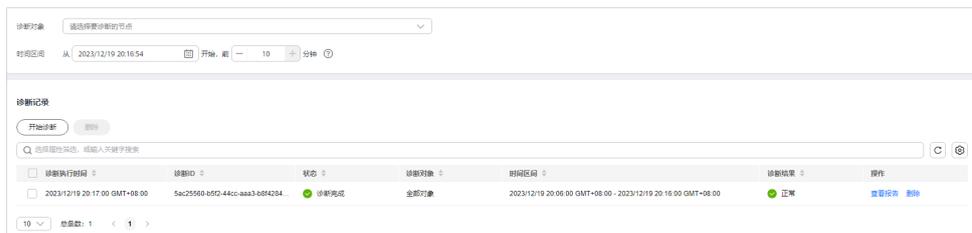
步骤4 单击需要实例诊断的DCS缓存实例名称，进入该实例的基本信息页面。

步骤5 选择“分析与诊断 > 实例诊断”进入实例诊断页面。

步骤6 设置诊断对象和诊断时间区间，单击“开始诊断”。

- 诊断对象：支持选择单节点、所有节点。
- 时间区间：支持诊断实例7天内的数据，每次诊断最长周期为10分钟。如下图设置，表示诊断实例在选定时间之前10分钟内的数据。

图 10-5 设置实例诊断对象和时间



📖 说明

实例在规格变更过程中执行实例诊断可能会诊断失败。

步骤7 诊断完成后，在诊断记录列表中可以查看诊断结果，如果出现异常，单击“查看报告”，查看具体异常的诊断项。

在异常的诊断项中，您可以查看产生异常的原因、异常的影响，以及处理异常的建议。

---结束

10.4 查看 Redis 实例的慢查询记录

慢查询是Redis用于记录命令执行时间过长的机制。您可以在DCS控制台查看慢请求日志，帮助解决性能问题。

查询结果中，涉及的慢语句命令详情，请前往[Redis官方网站](#)查看。

慢查询结果由以下实例配置参数决定：

- `slowlog-log-slower-than`：如果在Redis实例的数据节点中执行一个命令，执行时间超过了`slowlog-log-slower-than`参数设置的阈值（单位为微秒），则会被记录到慢查询中。该参数的默认值为10000，即10ms，当Redis命令执行时间超过10ms，则生成慢查询。
- `slowlog-max-len`：Redis实例记录的慢查询个数由`slowlog-max-len`参数的值决定，默认值为128个，即默认最多只显示最新的128个慢查询记录。

如下两个参数仅“华东-上海二”和“华南-广州”区域支持。

- `proxy-slowlog-log-slower-than`：如果在Proxy集群和读写分离实例的proxy节点中执行一个命令，执行时间超过了`proxy-slowlog-log-slower-than`参数设置的阈值（单位为微秒），则会被记录到慢查询中。该参数的默认值为256000，即256ms，当proxy节点的命令执行时间超过256ms，则生成慢查询。
- `proxy-slowlog-max-len`：Proxy集群和读写分离实例在proxy节点的慢查询个数由`proxy-slowlog-max-len`参数的值决定，默认值为128个，即默认最多只显示最新的128个慢查询记录。

实例配置参数的修改以及参数解释，请参考[修改DCS实例配置参数](#)。

约束与限制

- 如果是Redis 3.0 Proxy集群实例，必须是2019年10月14号后创建的Redis 3.0 Proxy集群实例才支持慢查询。
- 当前界面只能查询七天内的慢查询。
- 实例发生重启后，将无法查询到重启前的慢查询记录。

查看慢查询记录

步骤1 登录[分布式缓存服务管理控制台](#)。

步骤2 在管理控制台左上角单击 ，选择实例所在的区域。

步骤3 单击左侧菜单栏的“缓存管理”，进入实例信息页面。

步骤4 单击需要进行慢查询的DCS缓存实例名称，进入该实例的基本信息页面。

步骤5 选择“分析与诊断 > 慢查询”进入慢查询页面。

步骤6 设置查询时间，单击右侧的刷新图标，查看慢查询记录。如果您想了解返回查询结果中慢语句命令详情，请前往[Redis官方网站](#)查看。

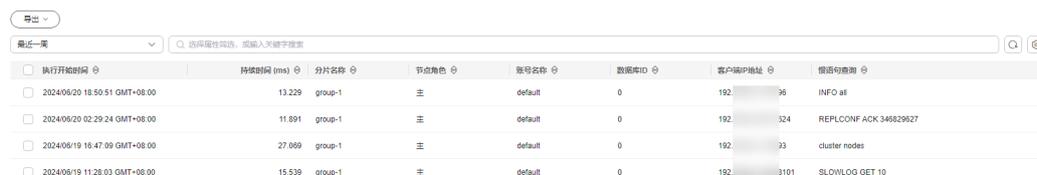
如果需要筛选慢查询记录，单击筛选属性栏，选择筛选属性并输入筛选条件。

📖 说明

目前仅在“华东-上海一”、“华东-上海二”和“华南-广州”区域，Proxy集群和读写分离类型的实例的慢查询结果包含“Proxy”和“Redis Server”类别，分别对应Proxy节点和Redis实例节点的慢查询记录。

- 如果Proxy集群与读写分离实例创建日期在2024/08之前，并且没有升级过Proxy节点，请联系客服升级Proxy节点。否则“Proxy”下慢查询记录始终为空。
- 只有实例开启客户端IP透传后，慢查询记录中的“客户端IP地址”才是真实的客户端IP地址。如果是Proxy集群和读写分离类型的实例，仅在“Proxy”下的客户端IP地址为真实的客户端IP。

图 10-6 实例慢查询记录



执行开始时间	持续时间 (ms)	分片名称	节点角色	实例名称	数据center ID	客户端IP地址	慢语句查询
2024/08/20 18:50:51 GMT+08:00	13.229	group-1	主	default	0	192.168.1.36	INFO all
2024/08/20 02:29:24 GMT+08:00	11.891	group-1	主	default	0	192.168.1.324	REPLCONF ACK 346829627
2024/08/19 16:47:09 GMT+08:00	27.069	group-1	主	default	0	192.168.1.33	cluster nodes
2024/08/19 11:28:03 GMT+08:00	15.539	group-1	主	default	0	192.168.1.3101	SLOWLOG GET 10

步骤7 如果需要下载慢查询记录，单击“导出”选择导出全部或选中的数据。

----结束

10.5 查询 Redis 实例运行日志

DCS管理控制台支持查询Redis实例运行日志，根据设定时间采集对应的redis.log日志，采集成功后，您可以将运行日志下载到本地查看。

通常在实例运行异常时，用户可以通过运行日志查看实例是否发生AOF重写、配置修改、高危命令执行、或主备倒换等记录，帮助排查和处理实例异常问题。

约束与限制

- Redis 4.0及以上版本的实例支持该功能。
- 采集的运行日志文件只保留7天，超期后会自动删除。
- Redis实例最长支持查询7天内的运行日志。

查询 Redis 实例运行日志

步骤1 登录[分布式缓存服务管理控制台](#)。

步骤2 在管理控制台左上角单击📍，选择实例所在的区域。

步骤3 单击左侧菜单栏的“缓存管理”，进入实例信息页面。

步骤4 单击需要查看运行日志的DCS缓存实例名称，进入该实例的基本信息页面。

步骤5 单击“运行日志”页签。

步骤6 单击“采集日志文件”，选择采集时长后，单击“确定”。

如果是主备、读写分离和集群实例，支持根据分片和副本进行日志采集，您需要选择指定分片和副本。如果是单机实例，实例只有一个节点，默认采集该节点的日志。

采集的日志文件以一个自然天为单位，例如选择采集的时长为前3天，则会根据日期生成成为3个不同的日志文件。

步骤7 采集日志文件成功后，单击运行日志文件后的“下载”，可以下载并查看该文件。

📖 说明

Redis内核产生日志较少，您选择的时段内可能日志为空。

----结束

10.6 查看 Redis 实例的命令审计日志

命令审计日志，是记录客户端访问DCS操作的一种日志，由华为云的云日志服务（LTS）提供存储、查询和分析等功能。

目前仅Redis 4.0及以上版本的Proxy集群实例，并且仅在北京四、上海一、上海二和广州Region支持命令审计日志，其他实例类型和其他区域的实例暂不支持。

如果符合以上条件，控制台仍未显示审计日志，请联系客服人员升级您的proxy节点。

审计日志开启后会在LTS侧创建对应的日志组、日志流和仪表盘。使用期间按照日志量收费，收费标准请参照[价格详情](#)。

须知

- 审计日志开启会重启所有proxy节点，请确保客户端具备断连重试功能。
如果实例开启过审计日志，且没有操作过实例扩容、节点迁移，再次开启审计日志不会重启proxy节点。
- 开启审计日志后，若写入流量或QPS过大，可能会导致DCS实例出现性能下降以及审计日志的部分丢失。

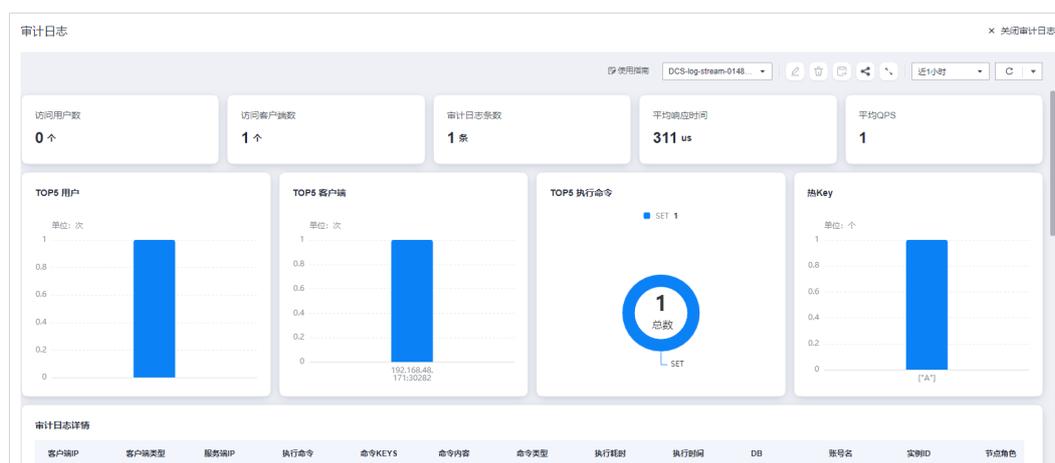
约束与限制

- 开启审计日志需要用户账号拥有LTS服务创建日志组和日志流的权限。
- 新创建的缓存实例，请在实例创建成功10分钟后，再开启审计日志，否则可能会操作失败。
- 实例扩容、缩容以及节点迁移的场景下，审计日志会自动关闭。再次使用需重新开启。
- DCS的审计日志默认存储时间为一天，开启审计日志后，可以在LTS控制台修改日志组的存储天数。

查看 Redis 实例的命令审计日志

- 步骤1** 登录[分布式缓存服务管理控制台](#)。
- 步骤2** 在管理控制台左上角单击 ，选择实例所在的区域。
- 步骤3** 单击左侧菜单栏的“缓存管理”，进入实例信息页面。
- 步骤4** 单击需要查看操作审计日志的DCS缓存实例名称，进入该实例的基本信息页面。
- 步骤5** 单击“日志管理>审计日志”，进入审计日志页面。
- 步骤6** 如未开启审计日志，请单击“开启审计日志”。
- 步骤7** 开启该功能后会在LTS侧创建对应的日志组和日志流，写命令会通过proxy节点上报到LTS。
- 步骤8** 开启后，即可通过审计日志查看如下内容。

图 10-7 审计日志



----结束

更多操作

- **关闭审计日志**
如需关闭审计日志功能，单击审计日志页面右上角的“关闭审计日志”，关闭后不再记录命令。

📖 说明

- 关闭审计日志后，若还有部分日志未上报完毕，这部分日志会继续上报完。
- 关闭审计日志后，在LTS侧的日志组和日志流资源不会删除，该资源本身不产生费用。
- 如果需要删除日志组和日志流，请到LTS控制台手动操作，参考[删除日志组](#)、[删除日志流](#)。
- **修改日志存储时间**
在LTS控制台的日志组列表页面，单击对应日志组操作列的“修改”，修改日志存储时间。

图 10-8 修改日志存储时间或删除日志组



日志组列表

创建日志组 点击此处添加高级条件

日志组名称	备注	企业项目	日志条数	标签	操作
DC-S-log-group-1731897660774		default	1		修改 删除 <input type="button" value="更多"/>

11 迁移实例数据

11.1 DCS 数据迁移概述

DCS控制台界面支持的数据迁移方式有在线迁移和离线迁移（备份文件导入）两种方式，其中，在线迁移支持增量数据迁移。

- 在线迁移，涉及到SYNC/PSYNC命令，适用于源Redis放通了SYNC/PSYNC命令的场景。支持将源Redis中的数据全量迁移或增量迁移到目标Redis中。
进行在线迁移时，迁移执行机会向源端地址发送PSYNC命令，其原理可参考[Replication介绍](#)，该命令会引起源端执行fork系统调用，对时延产生影响，其影响范围可参考[Redis官网](#)。
- 离线迁移，适用于源Redis和目标Redis网络不连通、源Redis不支持SYNC/PSYNC命令的场景。需要将数据备份文件导入到OBS，DCS从OBS桶中读取数据，将数据迁移到华为云Redis中；或直接将备份文件导入到DCS实例中。

由于用户对Redis的使用环境和场景各有差异，具体的迁移方案需要用户根据实际需求完善与细化。迁移耗时也与数据量大小、源Redis部署出处、网络带宽等相关，具体耗时需要在演练过程中记录与评估。

在迁移实例前需要分析业务系统使用到的缓存相关命令（附：[DCS命令兼容性说明参考](#)），在演练阶段对命令逐一验证。如有需要，可联系客服。

须知

- 当前数据迁移功能为公测免费，开始收费时间会另行通知。
- 数据迁移是一项重要且严肃的工作，准确性与时效性要求非常高，且与具体业务和操作环境相关。
- 本文提供的案例仅供参考，实际迁移应考虑具体的业务场景和需求，请勿直接套用。
- 本文提供的迁移操作，部分命令中包含了实例密码，这会导致密码记录到操作系统中，请注意保护密码不被泄露，并及时清除历史操作记录。
- DCS Redis 3.0已下线，暂停售卖，建议使用Redis 4.0及以上版本。

DCS 支持的迁移能力

说明

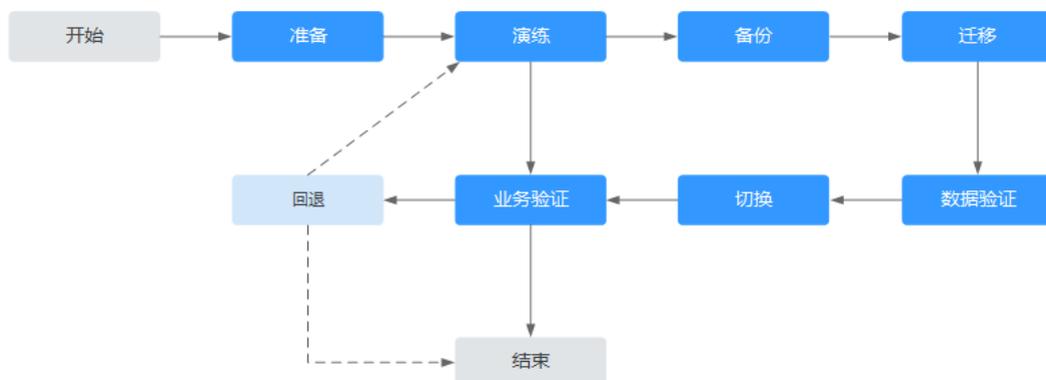
- **DCS Redis**: 指的是华为云分布式缓存服务的Redis。
- **自建Redis**: 指的是在云上、其他云厂商、本地数据中心自行搭建Redis。
- √表示支持, ×表示不支持。
- 源端为**其他云厂商Redis**时, 只有在满足和目标**DCS Redis**的网络相通、源Redis已放通SYNC和PSYNC命令这两个前提下, 才可以使用在线迁移的方式, 将源Redis中的数据全量迁移或增量迁移到目标Redis中, 但其他云厂商的部分实例可能存在无法在线迁移的问题, 可以采用离线或其它迁移方案, 具体请参考[迁移方案概览](#)。

表 11-1 DCS 支持的迁移能力

迁移类型	源端	目标端: DCS服务		
		单机/主备/读写分离	Proxy集群	Cluster集群
备份文件导入	AOF文件	√	√	√
	RDB文件	√	√	√
在线迁移	DCS Redis: 单机/主备/读写分离	√	√	√
	DCS Redis: Proxy集群	√	√	√
	DCS Redis: Cluster集群	√	√	√
	自建Redis	√	√	√
	其他云厂商Redis	√	√	√

迁移流程

图 11-1 迁移流程示意图



1. 迁移评估。

获取当前待迁移的缓存数据信息（可参考表11-2记录以下信息），包括：

- 实例数量
- 各实例配置的数据库数量
- 各数据库的key数量
- 业务用到的数据库
- 各实例数据占用空间
- Redis版本
- Redis实例类型
- 业务与各实例的连接关系

📖 说明

- 连接待迁移Redis后，执行**info keypace**命令可以查看数据分布情况，确认有数据的数据库编号以及数据库中的key数量。记录各数据库存储的key的数量可供迁移验证对比。
- 连接待迁移Redis后，执行**info memory**命令，参考“used_memory_human”的返回值可以查看源Redis数据占用空间。请确认用于中转的ECS可用磁盘空间和目标实例规格与剩余可用内存是否足够（不小于源Redis数据大小）。

根据获取到的信息规划DCS目标缓存实例，包括：

- 申请缓存实例数量
- 缓存实例的内存规格（不小于源Redis内存规格）
- 缓存实例的版本（不低于源Redis版本）
- 缓存实例类型
- 缓存实例与业务所属网络规划（VPC/子网/安全组）

2. 迁移准备。

当完成迁移评估后，需要准备以下内容：

- a. 移动存储介质
用于在网络不通（自建数据中心场景）的情况下以复制方式传输数据。
- b. 网络资源
按照业务规划创建虚拟私有云与子网。
- c. 服务器资源
申请弹性云服务器，承载Redis客户端。用于导出或导入缓存数据。
弹性云服务器的规格建议不低于8C16G。
- d. DCS缓存实例
按照迁移规划**申请缓存实例**，如果实例数量超过用户默认配额，请联系客服。
- e. 相关工具安装
包括SSH工具、FTP工具等。
- f. 信息收集
信息收集包括参与人员联系方式，弹性云服务器登录信息，缓存实例信息与数据库信息等。
- g. 整体迁移方案
制定总体迁移计划，包括人员安排、演练方案、迁移方案、验证方案、业务切换方案、回退方案。

每一份方案需要有细化到可执行的操作步骤，以及可标记任务结束的里程碑。

3. 迁移演练。

迁移演练的主要目的如下：

- a. 验证迁移工具与过程的可行性。
- b. 发掘迁移过程中遇到的问题，并做出有效的改进。
- c. 评估迁移耗时。
- d. 优化迁移步骤，验证部分工作并行的可行性，提高迁移效率。

4. 备份。

在迁移前，需要先行备份，包括但不限于缓存数据、Redis配置文件，用于应急。

5. 迁移。

在完成一到两轮的迁移演练，并根据演练过程中发现的问题进行优化后，正式开始数据迁移。

迁移过程应该细化到每一步可执行的步骤，有明确的开始与结束确认动作。

6. 数据验证。

数据验证可以包括以下几方面：

- 各数据库的key分布是否与原来或者迁移预期一致。
- 关键Key的检查。
- Key的过期时间检查。
- 实例是否能够正常备份和恢复。

7. 业务切换。

- a. 当缓存数据完成迁移，且验证无误后，业务可以正式切换缓存数据的连接，恢复对外。
- b. 如果涉及到缓存数据库编号的变化，业务还需修改编号的选择配置。
- c. 如果业务整体由数据中心或其他云厂商迁移到华为云，业务和缓存数据的迁移可并行。

8. 业务验证。

- a. 业务应用与DCS缓存实例的连通。
- b. 通过业务操作对缓存数据的增删改查。
- c. 如果条件满足，进行压测，确认性能满足业务峰值压力。

9. 回退。

当遇到演练中没有及时发现的问题，导致数据迁移后无法供业务使用，且短期无法解决，则涉及到业务回退。

由于源Redis数据仍然存在，因此只需业务完成回退，重新接入源Redis实例即可。

在完成回退后，可继续从演练甚至准备阶段重新开始，解决问题。

迁移信息收集表

评估和准备阶段收集的信息填写参考下表：

表 11-2 迁移信息收集

迁移源	信息项	说明
源Redis (列出所有待迁移的实例)	源Redis实例的IP地址	-
	Redis访问密码(如有)	-
	总数据量大小	执行 info memory 命令, 参考 used_memory_human 的值得到总数据量大小。用于评估迁移方案、DCS缓存实例规格、ECS可用磁盘空间等是否满足, 以及预估迁移耗时(业务中断时间)。
	不为空的数据库编号	info keyspaces 命令查询得到。用于确认迁移是否涉及多数据库, 非AOF文件方式迁移, 部分开源工具可能须逐库处理导出和导入。DCS缓存实例中, 单机和主备实例支持0-255共256个数据库, 集群默认只提供一个数据库。
	各数据库的key数量	用于迁移后进行数据完整性验证。
	数据类型	CDM迁移服务当前支持Hash和String两种数据格式, 如果源数据含有list、set之类数据, 请采用第三方迁移工具。
华为云ECS (弹性云服务器) 如果待迁移实例较多, 可准备多台ECS并行迁移	弹性IP地址	选择与DCS缓存实例网络互通的弹性云服务器进行数据导入, 确保导入过程网络稳定。带宽建议选取高配, 提升数据传输效率。
	系统登录用户/密码	-
	CPU/内存	部分迁移工具支持多线程并行导入, 使用高规格ECS, 能提升导入速度。
	可用磁盘空间	ECS需要预留足够的可用磁盘空间, 存储压缩文件以及解压后的缓存数据文件。 注: 为提高数据传输效率, 对于较大的数据文件, 建议压缩后再传输到弹性云服务器。
DCS缓存实例 (根据源Redis实例数与数据量情况选择合适的规格与实例数)	实例连接地址	-
	实例连接端口	-
	实例访问密码	-
	实例类型	-
	实例规格/可用内存	-

迁移源	信息项	说明
网络配置	VPC	提前规划VPC，确保应用服务、DCS缓存实例等处于相同VPC中。
	子网	-
	安全组或白名单	由于Redis 3.0和Redis 4.0/5.0/6.0实例部署模式不一样，控制访问方式也不一样，需要制定相应的安全组或白名单规则，确保网络连通。具体请根据目标Redis实例参考 配置安全组 或者 配置白名单
...	...	<i>其他配置信息。</i>

11.2 迁移方案说明

迁移工具

表 11-3 Redis 迁移工具对比

工具/命令/服务	特点	说明
DCS控制台界面一键式迁移	操作简单，同时支持在线迁移和离线迁移（备份文件导入）两种方式，其中在线迁移支持增量数据迁移。	<ul style="list-style-type: none"> 离线迁移，适用于源Redis和目标Redis网络不连通、源Redis不支持SYNC/PSYNC命令的场景。需要将数据备份文件导入到OBS，DCS从OBS桶中读取数据，将数据迁移到DCS的Redis中。 在线迁移，涉及到SYNC/PSYNC命令，适用于源Redis放通了SYNC/PSYNC命令的场景。支持将源Redis中的数据全量迁移或增量迁移到目标Redis中。
Redis-cli	<ul style="list-style-type: none"> Redis自带命令行工具，支持导出RDB文件，也支持将持久化的AOF文件整库导入。 AOF文件为所有数据更改命令的全量集合，数据文件稍大。 	-
Rump	支持在线迁移，支持在同一个实例的不同数据库之间，以及不同实例的数据库之间迁移。	不支持增量迁移。 建议停业务后迁移，避免出现Key丢失。详情参考 使用Rump在线迁移其他云厂商Redis 。
Redis-shake	在线迁移和离线迁移均支持的一款开源工具。	适用于Cluster集群的数据迁移。

工具/命令/服务	特点	说明
自行开发迁移脚本	灵活，根据实际情况适配。	-

迁移方案

表 11-4 迁移方案

迁移场景	工具	迁移案例	迁移说明
华为云DCS实例间迁移	DCS控制台界面一键式迁移	<ul style="list-style-type: none"> 如果是相同账号下相同Region之间的实例进行迁移，推荐使用迁移任务在线迁移DCS Redis实例。 如果是不同账号或不同Region之间的实例进行迁移，推荐使用备份文件离线迁移DCS Redis实例。 	由于Redis不同版本存在数据兼容问题，建议低版本Redis迁移到同版本或高版本Redis，如果是高版本Redis迁移到低版本Redis，可能会导致迁移失败。
自建Redis迁移至DCS 说明 自建Redis，指的是在华为云、其他云厂商、本地数据中心自行搭建的Redis。	DCS控制台界面一键式迁移	<ul style="list-style-type: none"> 如果自建Redis和DCS Redis实例网络连通，推荐使用迁移任务在线迁移DCS Redis实例。 如果自建Redis和DCS Redis实例网络不通，推荐使用备份文件离线迁移自建Redis。 	-
	Redis-cli	<ul style="list-style-type: none"> 使用Redis-cli离线迁移自建Redis (AOF文件) 使用Redis-cli离线迁移自建Redis (RDB文件) 	-
	Redis-Shake	<ul style="list-style-type: none"> 使用RedisShake工具在线迁移自建Redis Cluster集群 使用RedisShake工具离线迁移自建Redis Cluster集群 	-
其他云厂商Redis服务迁移至DCS	DCS控制台界面一键式迁移	<ul style="list-style-type: none"> 如果其他云厂商Redis服务，没有禁用SYNC和PSYNC命令，推荐使用迁移任务在线迁移其他云厂商Redis。 如果其他云厂商Redis服务，禁用了SYNC和PSYNC命令，推荐使用备份文件离线迁移其他云厂商Redis。 	如果需要使用在线迁移，建议联系其他云厂商运维人员放通SYNC和PSYNC命令。

迁移场景	工具	迁移案例	迁移说明
	Rump	使用Rump在线迁移其他云厂商Redis	-
	Redis-shake	使用RedisShake离线迁移其他云厂商Redis 使用RedisShake在线迁移其他云厂商Redis	-

11.3 DCS 实例间迁移

11.3.1 使用迁移任务在线迁移 DCS Redis 实例

在满足DCS源Redis和目标Redis的网络相通、源Redis放通SYNC和PSYNC命令这两个前提下，支持使用在线迁移的方式，将DCS源Redis中的数据全量或增量迁移到目标Redis中。

📖 说明

在线迁移，相当于临时给源端Redis增加一个从节点并且做一次全量同步到迁移机，建议在业务低峰期执行迁移，否则可能导致源端实例CPU瞬时冲高，时延增大。

约束与限制

- 在线迁移不支持公网方式直接迁移。
- 在迁移之前，请先阅读[迁移方案说明](#)，选择正确的迁移方案，了解当前DCS支持的在线迁移能力，选择适当的目标实例。
- 将高版本Redis实例数据迁移到低版本Redis实例可能失败。
- 较早建立的实例如果密码中包含单引号（'），则该实例不支持进行在线迁移，建议修改实例密码或使用其他迁移方式。
- 如果是单机/主备等多DB的源实例迁移到Proxy集群实例，Proxy集群默认仅有一个DB0，请先确保源实例DB0以外的DB是否有数据，如果有，请先参考[Proxy集群开启多DB的使用限制及操作方式](#)开启Proxy集群多DB。
- 如果是单机/主备等多DB的源端实例迁移到Cluster集群实例，Cluster集群不支持多DB，仅有一个DB0，请先确保源端实例DB0以外的DB是否有数据，如果有，请将数据转存到DB0，否则会出现迁移失败，将数据转存到DB0的操作请参考[使用Rump在线迁移](#)。
- 进行在线迁移时，建议将源端实例的参数repl-timeout配置为300秒，client-output-buffer-limit配置为实例最大内存的20%。
- 开启了SSL的目标实例不支持数据迁移，需要关闭目标实例SSL后再进行迁移，开启或关闭SSL的操作请参考[配置Redis SSL数据加密传输](#)。

前提条件

- 如果您还没有目标Redis，请先创建，创建操作，请参考[购买Redis实例](#)。

- 如果您已有目标Redis，则不需要重复创建，为了对比迁移前后数据及预留足够的内存空间，建议在数据迁移之前清空目标实例数据，清空操作请参考[清空Redis实例数据](#)。

如果没有清空实例数据，数据迁移后，目标Redis与源Redis实例重复的数据会被覆盖，源Redis没有、目标Redis有的数据会保留。

创建在线迁移任务

步骤1 登录[分布式缓存服务管理控制台](#)。

如果源Redis与目标Redis所属不同账号下，请使用源Redis所在的账号登录DCS。

步骤2 在管理控制台左上角单击，选择源Redis所在的区域。

说明

仅当在线迁移任务与源Redis为相同账号下相同区域时，在线迁移时选择的源端Redis会自动放通源Redis的SYNC和PSYNC命令。否则无法进行在线迁移。

步骤3 单击左侧菜单栏的“数据迁移”。页面显示迁移任务列表页面。

步骤4 单击右上角的“创建在线迁移任务”。

步骤5 设置迁移任务名称和描述。

任务名称请以字母开头，长度不小于4位且不超过64位。任务名称只能包含字母、数字、中划线、下划线。

步骤6 配置在线迁移任务虚拟机资源的虚拟私有云（VPC）、子网和安全组。

说明

- 请选择与源Redis或目标Redis相同的VPC。
- 创建的在线迁移任务会占用一个租户侧IP，即控制台上迁移任务对应的“迁移IP”。如果源端Redis或目标端Redis配置了白名单，需确保配置了迁移IP或关闭白名单限制。
- 迁移任务所选安全组的“出方向规则”需放通源端Redis和目标端Redis的IP和端口（安全组默认情况下为全部放通，则无需单独放通），以便迁移任务的虚拟机资源能访问源Redis和目标Redis。

----结束

检查网络

步骤1 检查源Redis、目标Redis、迁移任务资源所在VPC是否在同一个VPC内。

如果是，则执行[配置在线迁移任务](#)；如果不是，执行[步骤2](#)。

步骤2 检查源Redis的VPC、目标Redis的VPC、迁移任务资源所在VPC的网络是否打通，确保迁移任务的虚拟机资源能访问源Redis和目标Redis。

如果已打通，则执行[配置在线迁移任务](#)；如果没打通，则执行[步骤3](#)。

步骤3 执行相应操作，打通网络。

- 当源Redis和目标Redis都属于DCS同一Region，请参考[VPC对等连接说明](#)创建对等连接，打通网络。

- 当源Redis和目标Redis属于DCS不同Region，请参考[云连接](#)创建云连接，打通网络。

---结束

配置在线迁移任务

步骤1 创建完在线迁移任务之后，单击该迁移任务右侧的“配置”，配置在线迁移的源Redis、目标Redis等信息。

步骤2 选择迁移方法。

支持“全量迁移”和“全量迁移+增量迁移”两种，“全量迁移”和“全量迁移+增量迁移”的功能及限制如表11-5所示。

如果实例迁移后需要[交换DCS实例IP](#)，迁移方法必须选择“全量迁移+增量迁移”。

表 11-5 在线迁移方法说明

迁移类型	描述
全量迁移	该模式为Redis的一次性迁移，适用于可中断业务的迁移场景。全量迁移过程中， 如果源Redis有数据更新，这部分更新数据不会被迁移到目标Redis。
全量迁移+增量迁移	该模式为Redis的持续性迁移，适用于对业务中断敏感的迁移场景。增量迁移阶段通过解析日志等技术，持续保持源Redis和目标端Redis的数据一致。 增量迁移，迁移任务会在迁移开始后，一直保持迁移中状态，不会自动停止。 需要您在合适时间，在“操作”列单击“停止”，手动停止迁移。停止后，源端数据不会丢失，只是目标端不再写入数据。增量迁移在传输链路网络稳定情况下是秒级时延，具体的时延情况依赖于网络链路的传输质量。

图 11-2 选择迁移方法



步骤3 仅当迁移方法选择“全量迁移+增量迁移”时，支持选择是否启用“带宽限制”。

如果启用带宽限制功能，当数据同步速度达到带宽限制时，将限制同步速度的继续增长。

步骤4 选择是否“自动重连”。如开启自动重连模式，迁移过程中在遇到网络等异常情况时，会无限自动重连。

自动重连模式在无法进行增量同步时，会触发全量同步，增加带宽占用，请谨慎选择。

步骤5 分别配置“源Redis”和“目标Redis”。

1. 配置“源Redis类型”和“源Redis实例”：“源Redis类型”请选择“云服务Redis”，并在“源Redis实例”处选择需要迁移的源Redis。
2. 配置“目标Redis类型”和“目标Redis实例”：
 - 如果目标Redis与迁移任务处于相同VPC，或处于同一区域下已打通网络的不同VPC，“目标Redis类型”请选择“云服务Redis”，并在“目标Redis实例”处选择需要迁移的目标Redis。
 - 如果目标Redis与迁移任务处于不同区域，“目标Redis类型”请选择“自建Redis”，并在“目标Redis实例”处输入目标Redis的IP地址和端口。如果目标Redis为Cluster集群，需要输入集群所有主节点的IP端口，用英文逗号隔开。例如：192.168.1.1:6379,192.168.0.0:6379
3. 分别配置“源Redis实例密码”和“目标Redis实例密码”：如果是密码访问模式实例，在输入连接实例的密码后，单击密码右侧的“测试连接”，检查实例密码是否正确、网络是否连通。如果是免密访问的实例，请直接单击“测试连接”。此处暂不支持使用Redis实例[账号管理](#)功能中创建的ACL账号及密码。
4. 在“源DB”或“目标DB”中，您可以选择是否需要指定具体迁移的DB。例如源端输入5，目标端输入6时，表示迁移源Redis DB5中的数据到目标Redis的DB6。当源端不指定DB，目标端指定DB时，表示默认迁移源端的全部数据，到目标端指定的DB；当目标端不指定DB时，表示默认迁移到与源端对应的DB。

说明

- 当源端为多DB，目标端为单DB的DCS实例时（单DB的实例只有DB0），需要源端的所有数据都在DB0，或者指定仅迁移源端某一DB中的数据并将目标端DB指定为0，否则会迁移失败。
- DCS Redis的DB数请参见[Redis实例是否支持多DB方式？](#)。

步骤6 单击“下一步”。**步骤7** 确认迁移信息，然后单击“提交”，开始执行迁移任务。

可返回迁移任务列表中，观察对应的迁移任务的状态，迁移成功后，任务状态显示“成功”。

如果出现迁移失败，建议单击迁移任务名称，进入迁移任务详情页面，通过“迁移日志”排查迁移失败的原因。

说明

- 如果是全量迁移+增量迁移，全量迁移后会一直处于增量迁移中的状态。
- 如需手动停止迁移中的任务，勾选迁移任务左侧的方框，单击迁移任务上方的“停止”，即可停止迁移。
- 勾选停止迁移或迁移失败的迁移任务，单击迁移任务上方的“重启”，可重新进行迁移。如果重启迁移任务后迁移失败，建议单击“配置”，重新配置在线迁移任务后重试。
- 单次最多支持勾选50个在线迁移任务，批量停止、删除、或重启迁移任务。

----结束

迁移后验证

迁移完成后，可以通过以下方式确认数据的完整性：

1. 连接源Redis和目标Redis，连接Redis的方式请参考[Redis-cli客户端连接Redis](#)。

2. 分别对源Redis和目标Redis执行**info keypace**，查看keys参数和expires参数的值。

图 11-3 查看实例数据

```
192.168.1.217:6379> info keypace
# Keyspace
db0:keys=81869,expires=0,avg_ttl=0
192.168.1.217:6379>
```

3. 对比源Redis和目标Redis的keys参数分别减去expires参数的差值。如果差值一致，则表示数据完整，迁移正常。

说明

如果是全量迁移，迁移过程中源Redis更新的数据不会迁移到目标实例。

交换 DCS 实例 IP（可选）

当DCS源Redis与目标Redis满足以下条件时，支持交换源Redis与目标Redis的IP地址。交换实例IP后，客户端代码无需修改源端实例的访问地址，即可自动连接到目标Redis。

满足交换实例IP的前提条件：

- Redis 4.0及以上版本的基础版实例支持实例交换IP，**企业版实例不支持实例交换IP**。
- 如果是Redis 3.0实例，需要先联系客服开通Redis 3.0实例交换IP的白名单，并且仅当Redis 3.0作为源端实例，目标端为Redis 4.0/5.0/6.0基础版实例时，支持交换实例IP。
- 开启公网访问后的源实例或目标实例，不支持交换IP。
- 源Redis与目标Redis必须是单机、主备、读写分离或Proxy集群实例，**Cluster集群实例不支持交换实例IP**。
- 实例**选择迁移方法**步骤时，必须选择的是“**全量迁移 + 增量迁移**”，如果是“全量迁移”的方式，则不支持交换实例IP。
- 源Redis与目标Redis实例的端口需要一致。

须知

1. 交换IP过程中，会自动停止在线迁移任务。
2. 交换实例IP地址时，会有一分钟内只读和秒级的闪断。如果源端实例为Redis 3.0，交换IP地址时，会有一分钟内只读和30秒左右的中断。
3. 请确保您的客户端应用具备重连机制和处理异常的能力，否则在交换IP后有可能需要重启客户端应用。
4. 源实例和目标实例不在同一子网时，交换IP地址后，会更新实例的子网信息。
5. 如果源端是主备实例，交换IP时不会交换备节点IP，请确保应用中直接引用备节点IP。
6. 如果应用中有直接引用域名，请选择交换域名，否则域名会挂在源实例中。
7. 请确保目标Redis和源Redis密码一致，否则交换IP后，客户端会出现密码验证错误。
8. 当源实例配置了白名单时，则在进行IP交换前，保证目标实例也配置同样的白名单。
9. Redis3.0实例交换IP地址后，需要将源实例的安全组配置同步至目标实例的白名单配置中。

步骤1 在“数据迁移>在线迁移”页面，当迁移任务状态显示为“增量迁移中”时，单击操作列的“更多 > 交换IP”打开交换IP弹框。

步骤2 在交换IP弹框中，在交换域名区域，选择是否交换域名。

说明

- 如果客户端使用域名连接Redis，必须选择交换域名，否则客户端应用需要修改使用的域名。
- 如果没有选择交换域名，则只交换实例的IP地址。

步骤3 单击“确定”，交换IP任务提交成功，当迁移任务的状态显示为“IP交换成功”，表示交换IP任务完成。

IP交换成功后，如果需要切换为原IP，单击操作列的“更多 > 回滚IP”，当任务状态显示为“IP回滚成功”表示IP交换回滚完成。

----结束

11.3.2 使用备份文件离线迁移 DCS Redis 实例

DCS支持通过备份文件导入的方式离线迁移DCS实例间的数据。

- 如果您需要迁移的源Redis与目标Redis为DCS相同账号下相同Region的实例，且源Redis实例非单机实例时，离线迁移请参考[通过Redis实例离线迁移](#)。
- 如果您需要迁移的源Redis与目标Redis为DCS不同账号或不同Region的实例，或源Redis实例为单机实例时，离线迁移请参考[通过OBS桶离线迁移](#)。

约束与限制

开启了SSL的目标实例不支持数据迁移，需要关闭目标实例SSL后再进行迁移，开启或关闭SSL的操作请参考[配置Redis SSL数据加密传输](#)。

前提条件

- 源Redis实例已执行数据备份并备份成功。
 - 如果是[通过Redis实例离线迁移](#)，无需下载备份文件到本地，备份数据请参考[手动备份DCS实例数据](#)。
 - 如果是[通过OBS桶离线迁移](#)，需要下载备份文件到本地，请参考[下载实例备份文件](#)。
- 已准备目标Redis实例。如果您还没有目标Redis实例，请先创建Redis，参考[购买Redis实例](#)。

目前Redis高版本支持兼容低版本，因此，同版本或低版本可以迁移到高版本Redis，目标端创建的实例版本不要低于源端Redis版本。
- 请确保目标Redis有足够的存储空间，迁移实例前建议清空目标Redis中的数据，清空实例数据的操作请参考[清空Redis实例数据](#)。如果没有清空实例数据，数据迁移后，目标Redis与源Redis实例重复的数据会被覆盖，源Redis没有、目标Redis有的数据会保留。

通过 Redis 实例离线迁移

步骤1 登录[分布式缓存服务管理控制台](#)。

步骤2 在管理控制台左上角单击 ，选择源Redis和目标Redis所在的区域。

步骤3 单击左侧菜单栏的“数据迁移”。页面显示迁移任务列表页面。

步骤4 单击右上角的“创建备份导入任务”，进入创建备份导入任务页面。

步骤5 设置迁移任务名称和描述。

任务名称请以字母开头，长度不小于4位且不超过64位。任务名称只能包含字母、数字、中划线、下划线。

步骤6 源Redis“数据来源”选择“Redis实例”。

图 11-4 选择数据来源（Redis 实例）



步骤7 在“源Redis实例”中选择需要迁移的源端实例。

步骤8 根据需要选择“源DB”，您可以指定源Redis备份文件某一个DB中的数据，例如输入5时，则只迁移DB5中的数据；无需指定DB时，请保持置空，即迁移全部DB。

步骤9 选择“是否多DB Proxy集群”，只有当源Redis数据为DCS Proxy集群实例，且开启了多DB（Proxy实例multi-db参数值为yes）时选择。

步骤10 在“备份记录”中选择需要迁移的备份文件。

步骤11 “目标Redis实例”请选择[前提条件](#)中准备的目标Redis。

步骤12 如果目标Redis是密码访问模式，请输入密码后，单击“测试连接”，检查密码是否正确。免密访问的实例，请直接单击“测试连接”。

步骤13 根据需要选择“目标DB”，您可以指定迁移数据到目标Redis的某一个DB中，例如输入5时，则迁移到目标Redis的DB5；不填表示不指定，默认迁移到与源端相同的DB中。

📖 说明

- 当源端为多DB，目标端为单DB的DCS实例时（单DB的实例只有DB0），需要源端的所有数据都在DB0，或者指定仅迁移源端某一DB中的数据并将目标端DB指定为0，否则会迁移失败。
- DCS Redis的DB数请参见[Redis实例是否支持多DB方式?](#)。

步骤14 单击“立即创建”。

步骤15 确认迁移信息，然后单击“提交”，开始创建迁移任务。

返回迁移任务列表，观察对应的迁移任务的状态，任务状态显示“成功”时，迁移成功。

----结束

通过 OBS 桶离线迁移

通过OBS桶离线迁移，您需要先将源Redis的数据备份下载到本地，然后将备份文件上传到与DCS目标Redis同一账号相同区域下的OBS桶中，再创建备份导入迁移任务，从OBS桶中读取数据并将数据迁移到目标Redis中。

📖 说明

- 上传OBS桶的备份文件支持.aof、.rdb、.zip、.tar.gz四种格式，您可以直接上传.aof和.rdb文件，也可以将.aof和.rdb文件压缩成.zip或.tar.gz文件，然后将压缩后的文件上传到OBS桶。
- 如果源端实例是集群Redis，每个备份文件对应集群中的一个分片，需要下载所有的备份文件，然后逐个上传到OBS桶。在迁移时，需要把所有分片的备份文件选中。

步骤1 在目标Redis实例所在的账号和区域下创建OBS桶。如果已有符合条件的OBS桶，无需重新创建。

在创建过程中，以下两个参数请按要求设置，其他详细的创建步骤，请参考[创建桶](#)章节。

- 选择“区域”。
OBS桶所在区域必须跟Redis目标实例所在区域相同。
- 设置“默认存储类别”，存储类别支持“标准存储”和“低频访问存储”。
请不要选择“归档存储”，否则会导致备份文件迁移失败。

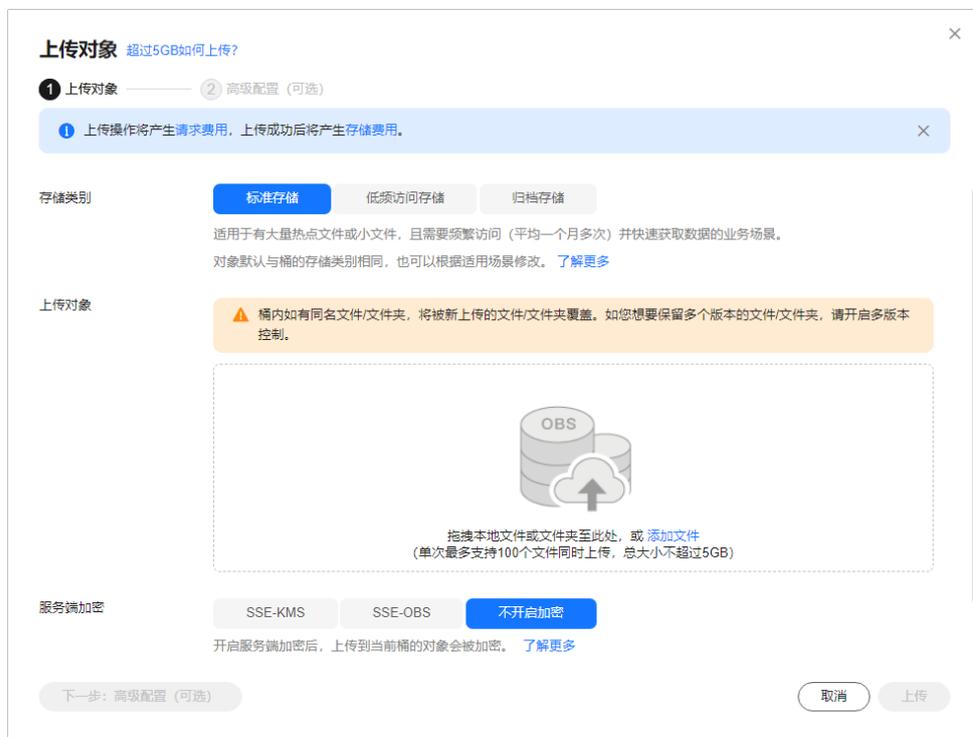
步骤2 上传备份文件到OBS桶。

1. 在OBS管理控制台的桶列表中，单击创建的桶名称，进入“概览”页面。
2. 在左侧导航栏，单击“对象”。
3. 在“对象”页签下，单击“上传对象”，系统弹出“上传对象”窗口。
4. 指定对象的存储类别。
请不要选择“归档存储”，否则会导致备份文件迁移失败。
5. 上传对象。

您可以拖拽本地文件或文件夹至“上传对象”区域框内添加待上传的文件，也可以通过单击“上传对象”区域框内的“添加文件”，选择本地文件添加。

单次最多支持100个文件同时上传，总大小不超过5GB。

图 11-5 上传对象



6. 选择服务端加密方式，支持选择“SSE-KMS”、“SSE-OBS”或“不开启加密”，详情请参见[服务端加密](#)。
7. 单击“上传”。

步骤3 单击控制台左上角服务列表，选择“应用中间件 > 分布式缓存服务 Redis版”，进入分布式缓存服务。

步骤4 单击左侧菜单栏的“数据迁移”进入数据迁移页面。

步骤5 单击右上角的“创建备份导入任务”。

步骤6 设置迁移任务名称和描述。

任务名称请以字母开头，长度不小于4位且不超过64位。任务名称只能包含字母、数字、中划线、下划线。

步骤7 在“源Redis”区域，“数据来源”选择“OBS桶”，在“OBS桶名”中选择已上传备份文件的OBS桶。

图 11-6 选择数据来源（OBS 桶）

步骤8 根据需要选择“源DB”，您可以指定源端备份文件某一个DB中的数据，例如输入5时，则只迁移DB5中的数据；无需指定DB时，请保持置空，即迁移全部DB。

步骤9 选择“是否多DB Proxy集群”，只有当源Redis数据为DCS Proxy集群实例，且开启了多DB（Proxy实例multi-db参数值为yes）时选择。

步骤10 单击“添加备份文件”，选择需要迁移的备份文件。

步骤11 在“目标Redis”区域，选择**前提条件**中准备的“目标Redis实例”。

步骤12 如果目标Redis是密码访问模式，请输入密码后，单击“测试连接”，检查密码是否正确。免密访问的实例，请直接单击“测试连接”。

步骤13 根据需要选择“目标DB”，您可以指定迁移数据到目标Redis的某一个DB中，例如输入5时，则迁移到目标Redis的DB5；不填表示不指定，默认迁移到与源端相同的DB中。

📖 说明

- 当源端为多DB，目标端为单DB的DCS实例时（单DB的实例只有DB0），需要源端的所有数据都在DB0，或者指定仅迁移源端某一DB中的数据并将目标端DB指定为0，否则会迁移失败。
- DCS Redis的DB数请参见[Redis实例是否支持多DB方式？](#)。

步骤14 单击“立即创建”。

步骤15 确认迁移信息，然后单击“提交”，开始创建迁移任务。

可返回迁移任务列表中，观察对应的迁移任务的状态，迁移成功后，任务状态显示“成功”。

----结束

11.4 自建 Redis 迁移至 DCS

11.4.1 使用迁移任务在线迁移自建 Redis

在满足源端自建Redis和目标Redis的网络相通、源Redis放通SYNC和PSYNC命令这两个前提下，DCS支持通过在线迁移的方式，将源端的自建Redis数据全量或增量迁移到DCS目标Redis中。

📖 说明

在线迁移，相当于临时给源端Redis增加一个从节点并且做一次全量同步到迁移机，建议在业务低峰期执行迁移，否则可能导致源端实例CPU瞬时冲高，时延增大。

约束与限制

- 如果源Redis禁用了SYNC和PSYNC命令，请务必放通后再执行在线迁移，否则会导致在线迁移失败。
- 在线迁移不支持公网方式直接迁移。
- 进行在线迁移时，建议将源端实例的参数repl-timeout配置为300秒，client-output-buffer-limit配置为实例最大内存的20%。
- 源端仅支持Redis 3.0及3.0以上的Redis版本。
- 较早建立的实例如果密码中包含单引号（'），则该实例不支持进行在线迁移，建议修改实例密码或使用其他迁移方式。
- 开启了SSL的目标实例不支持数据迁移，需要关闭目标实例SSL后再进行迁移，开启或关闭SSL的操作请参考[配置Redis SSL数据加密传输](#)。

前提条件

- 在迁移之前，请先阅读[迁移方案概览](#)，选择正确的迁移方案，了解当前DCS支持的在线迁移能力，选择适当的目标实例。
- 如果是单机/主备等多DB的源端实例迁移到Proxy集群实例，Proxy集群默认不开启多DB，仅有一个DB0，请先确保源端实例DB0以外的DB是否有数据，如果有，请先参考[开启多DB操作](#)开启Proxy集群多DB设置。
- 如果是单机/主备等多DB的源端实例迁移到Cluster集群实例，Cluster集群不支持多DB，仅有一个DB0，请先确保源端实例DB0以外的DB是否有数据，如果有，请将数据转存到DB0，否则会出现迁移失败，将数据转存到DB0的操作请参考[使用Rump在线迁移](#)。
- 已获取准备迁移的源Redis实例的IP地址和端口。
- 如果您还没有目标Redis，请先创建目标Redis，具体操作请参考[购买Redis实例](#)。
- 如果您已有目标Redis，则不需要重复创建，为了对比迁移前后数据及预留足够的内存空间，建议在数据迁移之前清空目标实例数据，清空操作请参考[清空Redis实例数据](#)。如果没有清空实例数据，数据迁移后，目标Redis与源Redis实例重复的数据迁移后会被覆盖，源Redis没有、目标Redis有的数据会保留。

创建在线迁移任务

步骤1 请使用DCS目标Redis所在的账号登录分布式缓存服务控制台。

步骤2 在管理控制台左上角单击 ，选择DCS目标Redis所在的区域。

步骤3 单击左侧菜单栏的“数据迁移”。页面显示迁移任务列表页面。

步骤4 单击右上角的“创建在线迁移任务”。

步骤5 设置迁移任务名称和描述。

任务名称请以字母开头，长度不小于4位且不超过64位。任务名称只能包含字母、数字、中划线、下划线。

步骤6 配置在线迁移任务虚拟机资源的VPC、子网和安全组。

说明

- 请选择与目标Redis相同的VPC，确保迁移资源能访问目标Redis实例。
- 创建的在线迁移任务会占用一个租户侧IP，即控制台上迁移任务对应的“迁移IP”。如果源端Redis或目标端Redis配置了白名单，需确保配置了迁移IP或关闭白名单限制。
- 迁移任务所选安全组的“出方向规则”需放通源端Redis和目标端Redis的IP和端口（安全组默认情况下为全部放通，则无需单独放通），以便迁移任务的虚拟机资源能访问源Redis和目标Redis。

----结束

检查网络

步骤1 检查源Redis、目标Redis、迁移任务资源所在VPC是否为同一个VPC。

如果是，请执行[配置在线迁移任务](#)；如果不是，请执行[步骤2](#)。

步骤2 检查源Redis的VPC、目标Redis的VPC、迁移任务资源所在VPC的网络是否打通，确保迁移任务的虚拟机资源能访问源Redis和目标Redis。

如果已打通，则执行[配置在线迁移任务](#)；如果没打通，则执行[步骤3](#)。

步骤3 执行相应操作，打通网络。

- 当源Redis和目标Redis所在的VPC属于同一云厂商的同一Region，请参考[VPC对等连接说明](#)，查看和创建对等连接，打通网络。
- 当源Redis和目标Redis所在的VPC属于同一云厂商的不同Region，请参考[云连接](#)，查看和创建云连接，打通网络。
- 当源Redis和目标Redis属于不同的云厂商，仅支持云专线打通网络，请参考[云专线](#)。

----结束

配置在线迁移任务

步骤1 创建完在线迁移任务之后，单击该迁移任务右侧的“配置”，配置在线迁移的源Redis、目标Redis等信息。

步骤2 选择迁移方法。

支持“全量迁移”和“全量迁移+增量迁移”两种，“全量迁移”和“全量迁移+增量迁移”的功能及限制如[表11-6](#)所示。

表 11-6 在线迁移方法说明

迁移类型	描述
全量迁移	该模式为Redis的一次性迁移，适用于可中断业务的迁移场景。全量迁移过程中，如果源Redis有数据更新，这部分更新数据不会被迁移到目标Redis。

迁移类型	描述
全量迁移 + 增量迁移	<p>该模式为Redis的持续性迁移，适用于对业务中断敏感的迁移场景。增量迁移阶段通过解析日志等技术，持续保持源Redis和目标端Redis的数据一致。</p> <p>增量迁移，迁移任务会在迁移开始后，一直保持迁移中状态，不会自动停止。需要您在合适时间，在“操作”列单击“停止”，手动停止迁移。停止后，源端数据不会造成丢失，只是目标端不再写入数据。增量迁移在传输链路网络稳定情况下是秒级时延，具体的时延情况依赖于网络链路的传输质量。</p>

图 11-7 选择迁移方法



步骤3 仅当迁移方法选择“全量迁移+增量迁移”时，支持选择是否启用“带宽限制”。

如果启用带宽限制功能，当数据同步速度达到带宽限制时，将限制同步速度的继续增长。

步骤4 选择是否“自动重连”。如开启自动重连模式，迁移过程中在遇到网络等异常情况时，会无限自动重连。

自动重连模式在无法进行增量同步时，会触发全量同步，增加带宽占用，请谨慎选择。

步骤5 分别配置“源Redis”和“目标Redis”。

- 配置“源Redis类型”和“源Redis实例”：

“源Redis类型”请选择“自建Redis”，并在“源Redis实例”处输入源Redis的IP地址和端口。

如果源Redis为Cluster集群，需要输入集群所有主节点的IP端口，用英文逗号隔开。例如：192.168.1.1:6379,192.168.0.0:6379
- 配置“目标Redis类型”和“目标Redis实例”：

“目标Redis类型”请选择“云服务Redis”，并在“目标Redis实例”处选择需要迁移的目标Redis。
- 分别配置“源Redis实例密码”和“目标Redis实例密码”：如果是密码访问模式实例，在输入连接实例密码后，单击密码右侧的“测试连接”，检查实例密码是否正确、网络是否连通。如果是免密访问的实例，请直接单击“测试连接”。如果测试连接失败，请检查输入的实例密码是否正确、Redis实例与迁移任务网络是否打通。

如果是DCS服务的Redis实例，此处暂不支持使用[账号管理](#)功能中创建的ACL账号及密码。
- 在“源DB”或“目标DB”中，您可以选择是否需要指定具体迁移的DB。例如源端输入5，目标端输入6时，表示迁移源Redis DB5中的数据到目标Redis的DB6；

当源端不指定DB，目标端指定DB时，表示默认迁移源端的全部数据，到目标端指定的DB，当目标端不指定DB时，表示默认迁移到与源端对应的DB。

📖 说明

- 当源端为多DB，目标端为单DB的DCS实例时（单DB的实例只有DB0），需要源端的所有数据都在DB0，或者指定仅迁移源端某一DB中的数据并将目标端DB指定为0，否则会迁移失败。
- DCS Redis的DB数请参见[Redis实例是否支持多DB方式?](#)。

步骤6 单击“下一步”。

步骤7 确认迁移信息，然后单击“提交”，开始创建迁移任务。

可返回迁移任务列表中，观察对应的迁移任务的状态，迁移成功后，任务状态显示“成功”。

如果出现迁移失败，建议单击迁移任务名称，进入迁移任务详情页面，通过“迁移日志”排查迁移失败的原因。

📖 说明

- 如果是增量迁移，全量迁移后会一直处于增量迁移中的状态。
- 如需手动停止迁移中的任务，勾选迁移任务左侧的方框，单击迁移任务上方的“停止”，即可停止迁移。
- 勾选停止迁移或迁移失败的迁移任务，单击迁移任务上方的“重启”，可重新进行迁移。如果重启迁移任务后迁移失败，建议单击“配置”，重新配置在线迁移任务后重试。
- 单次最多支持勾选50个在线迁移任务，批量停止、删除、或重启迁移任务。

----结束

迁移后验证

数据迁移前如果目标Redis中数据为空，迁移完成后，可以通过以下方式确认数据的完整性：

1. 连接源Redis和目标Redis。连接Redis的方法请参考[Redis-cli客户端连接Redis](#)。
2. 输入info keyspace，查看keys参数和expires参数的值。

```
192.168.1.217:6379> info keyspace
# Keyspace
db0:keys=81869,expires=0,avg_ttl=0
192.168.1.217:6379>
```

3. 对比源Redis和目标Redis的keys参数分别减去expires参数的差值。如果差值一致，则表示数据完整，迁移正常。

注意：如果是全量迁移，迁移过程中源Redis更新的数据不会迁移到目标实例。

11.4.2 使用备份文件离线迁移自建 Redis

本文档介绍如何通过备份文件导入的方式，将自建Redis离线迁移至DCS。

您需要先将自建Redis的数据备份下载到本地，然后将备份数据文件上传到华为云与DCS目标Redis实例同一账号下相同Region下的OBS桶中，最后在DCS控制台创建备份迁移任务，DCS从OBS桶中读取数据，将数据迁移到DCS的Redis中。

约束与限制

开启了SSL的目标实例不支持数据迁移，需要关闭目标实例SSL后再进行迁移，开启或关闭SSL的操作请参考[配置Redis SSL数据加密传输](#)。

前提条件

- 在迁移之前，请先阅读[迁移方案概览](#)，选择正确的迁移方案，了解当前DCS支持的在线迁移能力，选择适当的目标实例。
- 如果是单机/主备等多DB的源端实例迁移到Proxy集群实例，Proxy集群默认不开启多DB，仅有一个DB0，请先确保源端实例DB0以外的DB是否有数据，如果有，请先参考[开启多DB操作](#)开启Proxy集群多DB设置。
- 如果是单机/主备等多DB的源端实例迁移到Cluster集群实例，Cluster集群不支持多DB，仅有一个DB0，请先确保源端实例DB0以外的DB是否有数据，如果有，请将数据转存到DB0，否则会出现迁移失败，将数据转存到DB0的操作请参考[使用Rump在线迁移](#)。
- 准备源Redis的备份文件，备份文件的格式必须为.aof、.rdb、.zip或.tar.gz。
- 如果您还没有目标Redis，请先创建目标Redis，具体操作请参考[购买Redis实例](#)。
- 如果您已有目标Redis，则不需要重复创建，为了对比迁移前后数据及预留足够的内存空间，建议在数据迁移之前清空目标实例数据，清空操作请参考[清空Redis实例数据](#)。如果没有清空实例数据，数据迁移后，目标Redis与源Redis实例重复的数据迁移后会被覆盖，源Redis没有、目标Redis有的数据会保留。

创建 OBS 桶并上传备份文件

如果上传的源Redis备份文件小于5GB，请执行以下步骤，通过OBS控制台创建OBS桶并上传备份文件。如果上传的源Redis备份文件大于5GB，请参考[超过5GB如何上传上传备份文件](#)。

步骤1 通过OBS控制台，创建OBS桶。

在创建过程中，以下两个参数请按要求设置，其他详细的创建步骤，请参考[创建桶](#)章节。

- 选择“区域”。
OBS桶所在区域必须跟DCS目标Redis实例所在区域相同。
- 设置“存储类别”，当前支持“标准存储”、“低频访问存储”和“归档存储”。
请不要选择“归档存储”，否则会导致备份文件迁移失败。

步骤2 在OBS管理控制台的桶列表中，单击**步骤1**中创建的桶名称，进入“概览”页面。

步骤3 在左侧导航栏，单击“对象”。

步骤4 在“对象”页签下，单击“上传对象”，系统弹出“上传对象”对话框。

步骤5 指定对象的存储类别。

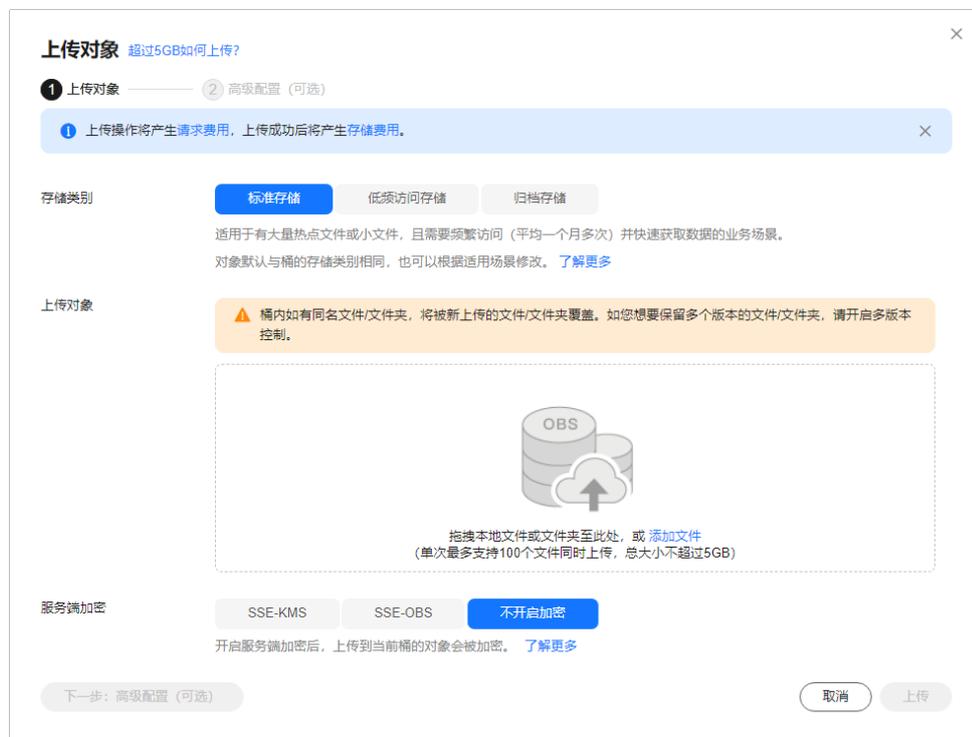
请不要选择“归档存储”，否则会导致备份文件迁移失败。

步骤6 上传对象。

您可以拖拽本地文件或文件夹至“上传对象”区域框内添加待上传的文件，也可以通过单击“上传对象”区域框内的“添加文件”，选择本地文件添加。

单次最多支持100个文件同时上传，总大小不超过5GB。

图 11-8 上传对象



步骤7 选择服务端加密方式，支持选择“SSE-KMS”、“SSE-OBS”或“不开启加密”，详情请参见**服务端加密**。

步骤8 单击“上传”。

----结束

创建迁移任务

步骤1 单击控制台左上角服务列表，选择“应用中间件 > 分布式缓存服务 Redis版”，进入分布式缓存服务。

步骤2 单击左侧菜单栏的“数据迁移”，进入数据迁移页面。

步骤3 单击右上角的“创建备份导入任务”。

步骤4 设置迁移任务名称和描述。

任务名称请以字母开头，长度不小于4位且不超过64位。任务名称只能包含字母、数字、中划线、下划线。

步骤5 “源Redis”区域中，“数据来源”选择“OBS桶”，在“OBS桶名”中选择已上传备份文件的OBS桶。

步骤6 根据需要选择“源DB”，您可以指定源端备份文件某一个DB中的数据，例如输入5时，则只迁移DB5中的数据；无需指定DB时，请保持置空，即迁移全部DB。

步骤7 选择“是否多DB Proxy集群”，只有当源Redis数据为DCS Proxy集群实例，且开启了多DB（Proxy实例multi-db参数值为yes）时选择。

步骤8 单击“添加备份文件”，选择需要迁移的备份文件。

步骤9 在“目标Redis”区域，选择[前提条件](#)中准备的“目标Redis实例”。

步骤10 如果目标Redis是密码访问模式，请输入密码后，单击“测试连接”，检查密码是否正确。免密访问的实例，请直接单击“测试连接”。

步骤11 根据需要选择“目标DB”，您可以指定迁移数据到目标Redis的某一个DB中，例如输入5时，则迁移到目标Redis的DB5；不填表示不指定，默认迁移到与源端相同的DB中。

说明

- 当源端为多DB，目标端为单DB的DCS实例时（单DB的实例只有DB0），需要源端的所有数据都在DB0，或者指定仅迁移源端某一DB中的数据并将目标端DB指定为0，否则会迁移失败。
- DCS Redis的DB数请参见[Redis实例是否支持多DB方式?](#)。

步骤12 单击“立即创建”。

步骤13 确认迁移信息，然后单击“提交”，开始创建迁移任务。

可返回迁移任务列表中，观察对应的迁移任务的状态，迁移成功后，任务状态显示“成功”。

---结束

迁移后验证

数据导入成功后，请连接DCS缓存实例，通过info命令，确认数据是否已按要求成功导入。连接Redis的方法请参考[Redis-cli客户端连接Redis](#)。

如果导入不成功，请检查操作步骤，如果是导入命令不正确，建议使用flushall或者flushdb命令清理目标实例中的缓存数据，修改导入命令后重新导入。

11.4.3 使用 Redis-cli 离线迁移自建 Redis（AOF 文件）

Redis-cli是Redis自带的一个命令行工具，安装Redis后即可直接使用Redis-cli工具。本文档主要介绍如何使用Redis-cli将自建Redis迁移到DCS缓存实例。

AOF文件的生成较快，适用于可以进入Redis服务器并修改配置的场景，如用户自建的Redis服务。

说明

- 建议选择业务量较少的时间段进行迁移。
- 正式进行迁移操作前，建议先暂停业务，确保不会在迁移过程中丢失新产生的数据变动。

约束与限制

开启了SSL的目标实例不支持数据迁移，需要关闭目标实例SSL后再进行迁移，开启或关闭SSL的操作请参考[配置Redis SSL数据加密传输](#)。

前提条件

- 如果您还没有目标Redis，请先创建目标Redis，具体操作请参考[购买Redis实例](#)。
- 如果您已有目标Redis，则不需要重复创建，为了对比迁移前后数据及预留足够的内存空间，建议在数据迁移之前清空目标实例数据，清空操作请参考[清空Redis实例](#)。

例数据。如果没有清空实例数据，数据迁移后，目标Redis与源Redis实例重复的数据迁移后会被覆盖，源Redis没有、目标Redis有的数据会保留。

- 已创建弹性云服务器ECS，创建弹性云服务器的方法，请参见[创建弹性云服务器](#)。

生成 AOF 文件

1. 登录弹性云服务器。
2. 安装Redis-cli客户端。该操作以客户端安装在Linux系统上为例进行说明。
 - a. 执行如下命令下载Redis。您也可以安装其他Redis版本。具体操作，请参见[Redis官网](#)。

```
wget http://download.redis.io/releases/redis-5.0.8.tar.gz
```
 - b. 执行如下命令，解压Redis客户端源码包。

```
tar -xzf redis-5.0.8.tar.gz
```
 - c. 进入Redis目录并编译Redis客户端源码。

```
cd redis-5.0.8
make
cd src
```
3. 执行如下命令开启缓存持久化，得到AOF持久化文件。

```
redis-cli -h {source_redis_address} -p {port} -a {password} config set appendonly yes
```

{source_redis_address}为源Redis的连接地址，{port}为源Redis的端口，{password}为源Redis的连接密码。
开启持久化之后，如果AOF文件大小不再变化，说明AOF文件为全量缓存数据。

📖 说明

- 使用redis-cli登录Redis实例，输入命令“**config get dir**”可以查找生成的AOF文件保存路径，文件名如果没有特殊指定，默认为：appendonly.aof。
- 生成AOF文件后如需关闭同步，可使用redis-cli登录redis实例，输入命令“**config set appendonly no**”进行关闭。

上传 AOF 文件至华为云 ECS

为节省传输时间，建议先压缩AOF文件，再将压缩文件（如以SFTP方式）上传到华为云ECS。

📖 说明

ECS需保证有足够的磁盘空间，供数据文件解压缩，同时要与缓存实例网络互通，通常要求相同VPC和相同子网，且安全组规则不限制访问端口。安全组设置请参考[如何选择和配置安全组](#)。

导入数据

登录弹性云服务器并执行如下命令导入数据。

```
redis-cli -h {dcs_instance_address} -p {port} -a {password} --pipe < appendonly.aof
```

{dcs_instance_address}为目标Redis连接地址，{port}为目标Redis的端口，{password}为目标Redis的连接密码。

VPC内导入AOF文件，平均100w数据（每条数据20字节），大概4~10秒完成。

须知

如果使用公网SSL加密时，连接地址和端口请替换为实际的配置信息。

迁移后验证

数据导入成功后，请连接DCS缓存实例，通过info命令，确认数据是否已按要求成功导入。连接Redis的方法请参考[Redis-cli客户端连接Redis](#)。

如果导入不成功，请检查操作步骤，如果是导入命令不正确，建议使用flushall或者flushdb命令清理目标实例中的缓存数据，修改导入命令后重新导入。

11.4.4 使用 Redis-cli 离线迁移自建 Redis（RDB 文件）

Redis-cli是Redis自带的一个命令行工具，安装Redis后即可直接使用Redis-cli工具。Redis-cli提供了RDB文件导出功能，如果Redis服务不支持获取AOF文件，可以尝试通过Redis-cli获取RDB文件。然后再通过其他工具（如RedisShake）导入到DCS的缓存实例中。

📖 说明

建议选择业务量较少的时间段进行迁移。

约束与限制

- 源端为Redis原生集群的数据时，需要针对集群的每个节点分别导出数据，然后逐一导入数据。
- 开启了SSL的目标实例不支持数据迁移，需要关闭目标实例SSL后再进行迁移，开启或关闭SSL的操作请参考[配置Redis SSL数据加密传输](#)。

前提条件

- 如果您还没有目标Redis，请先创建目标Redis，具体操作请参考[购买Redis实例](#)。
- 如果您已有目标Redis，则不需要重复创建，为了对比迁移前后数据及预留足够的内存空间，建议在数据迁移之前清空目标实例数据，清空操作请参考[清空Redis实例数据](#)。如果没有清空实例数据，数据迁移后，目标Redis与源Redis实例重复的数据迁移后会被覆盖，源Redis没有、目标Redis有的数据会保留。
- 已创建弹性云服务器ECS，创建弹性云服务器的方法，请参见[创建弹性云服务器](#)。
- 自建的源Redis实例必须放通SYNC命令，否则无法使用Redis-cli导出RDB文件。

导出 RDB 文件

1. 导出前准备。

对于主备或集群实例，数据写入RDB文件有一定的时延，时延策略配置在redis.conf文件中。建议先了解待迁移Redis实例的RDB策略配置，然后暂停业务系统并向Redis实例写入满足数量条件的测试数据，确保RDB文件为最新生成。

例如，redis.conf中对RDB的默认策略配置如下：

```
save 900 1 //900秒内有数据变更则写入RDB文件
save 300 10 //300秒内有10条以上数据变更则写入RDB文件
save 60 10000 //60秒内有10000条以上数据变更则写入RDB文件
```

因此，可以参考以上数据写入RDB的策略，在停止业务系统后，向Redis实例写入一定数量的测试数据，触发策略并写入RDB文件，确保业务数据均已同步到RDB文件中。

测试数据可以在导入后删除。

📖 说明

- 如果有某个数据库没有被业务系统使用，可以将测试数据写入该数据库，待导入DCS后，使用flushdb命令清空该数据库。
 - 单机实例如果不做持久化配置，则RDB文件需要临时生成，导出耗时较主备实例相比稍多一些。
2. 登录弹性云服务器。
 3. 安装Redis-cli客户端。该操作以客户端安装在Linux系统上为例进行说明。
 - a. 执行如下命令下载Redis。您也可以安装其他Redis版本。具体操作，请参见[Redis官网](#)。

```
wget http://download.redis.io/releases/redis-5.0.8.tar.gz
```
 - b. 执行如下命令，解压Redis客户端源码包。

```
tar -xzf redis-5.0.8.tar.gz
```
 - c. 进入Redis目录并编译Redis客户端源码。

```
cd redis-5.0.8
make
cd src
```
 4. 使用如下命令导出RDB文件：

```
redis-cli -h {source_redis_address} -p {port} -a {password} --rdb {output.rdb}
```

{source_redis_address}为源Redis的连接地址，{port}为源Redis的端口，{password}为源Redis的连接密码，{output.rdb}为RDB文件名。

执行命令后回显"Transfer finished with success."，表示文件导出成功。

上传 RDB 文件至华为云 ECS

为节省传输时间，建议先压缩RDB文件，再将压缩文件（如以SFTP方式）上传到华为云ECS。

📖 说明

ECS需保证有足够的磁盘空间，供数据文件解压缩，同时要与缓存实例网络互通，通常要求相同VPC和相同子网，且安全组规则不限制访问端口。安全组设置请参考[如何选择和配置安全组](#)。

导入数据

可借助RedisShake工具完成数据导入。

VPC内导入RDB文件，平均100w数据（每条数据20字节），大概4~10秒完成。

迁移后验证

数据导入成功后，请连接DCS缓存实例，通过info命令，确认数据是否已按要求成功导入。连接Redis的方法请参考[Redis-cli客户端连接Redis](#)。

如果导入不成功，请检查操作步骤，如果是导入命令不正确，建议使用flushall或者flushdb命令清理目标实例中的缓存数据，修改导入命令后重新导入。

11.4.5 使用 RedisShake 工具在线迁移自建 Redis Cluster 集群

RedisShake是一款开源的Redis迁移工具，支持Cluster集群的在线迁移与离线迁移（备份文件导入）。DCS Cluster集群与Redis Cluster集群设计一致，数据可平滑迁移。

本文以Linux系统环境为例，介绍如何使用RedisShake工具将自建的Redis Cluster集群在线迁移到DCS Cluster集群。

约束与限制

- 使用RedisShake工具将自建的Redis Cluster在线迁移到DCS Cluster集群，需要源Redis与目标Redis网络连通，或者通过一台中转云服务器连通两端的Cluster集群实例。
- 开启了SSL的目标实例不支持数据迁移，需要关闭目标实例SSL后再进行迁移，开启或关闭SSL的操作请参考[配置Redis SSL数据加密传输](#)。

前提条件

- 如果您还没有目标Redis，请先创建目标Redis，具体操作请参考[购买Redis实例](#)。
- 如果您已有目标Redis，则不需要重复创建，为了对比迁移前后数据及预留足够的内存空间，建议在数据迁移之前清空目标实例数据，清空操作请参考[清空Redis实例数据](#)。如果没有清空实例数据，数据迁移后，目标Redis与源Redis实例重复的数据迁移后会被覆盖，源Redis没有、目标Redis有的数据会保留。
- 已创建弹性云服务器ECS，创建弹性云服务器的方法，请参见[创建弹性云服务器](#)。ECS请选择与DCS Cluster集群实例相同虚拟私有云、子网和安全组，并且需要绑定弹性公网IP。
- 自建的源Redis Cluster集群如果是在本地或者其他云厂商的服务器上自建，需要允许被公网访问。

获取源 Redis 和目标 Redis 节点信息

1. 分别连接源端和目标端Redis。连接Redis的方法请参考[Redis-cli客户端连接Redis](#)。
2. 在线迁移Cluster集群时需要将Cluster集群各个节点数据分别迁移。执行如下命令分别查询源端和目标Cluster集群的所有节点的IP地址与端口：

```
redis-cli -h {redis_address} -p {redis_port} -a {redis_password} cluster nodes
```

{redis_address}为Redis的连接地址，{redis_port}为Redis的端口，
{redis_password}为Redis的连接密码。

在命令返回的结果中，获取所有master节点的IP端口，如下如所示：

```
[root@ecs-54-centos ~]# redis-cli -h 192.168.0.140 -p 6379 -a 23 cluster nodes
fb75f0743af4695a3d241ff7790b2f508e4985ff 192.168.0.140:6379@16379 myself,master - 0 1562144170000 3 connected
d112bae791b2bbd9602fe32963536b8a0db9eb79 192.168.0.61:6379@16379 master - 0 1562144171524 1 connected 0-5460
73e2f8fe196166f9ad1283361867d24c136413f0 192.168.0.194:6379@16379 master - 0 1562144170000 2 connected 5461-10
40d72299fde6045de0f79ee4b97910b505acbc6a 192.168.0.231:6379@16379 slave 73e2f8fe196166f9ad1283361867d24c136413
0e6c07faa64d724323e0d7cedc3f38346dcbd212 192.168.0.80:6379@16379 slave fb75f0743af4695a3d241ff7790b2f508e4985f
c16b9acaee7dd0721f129596cd43bd499c0e396 192.168.0.169:6379@16379 slave d112bae791b2bbd9602fe32963536b8a0db9eb
```

配置 RedisShake 工具

1. 登录弹性云服务器ECS。
2. 在ECS中执行以下命令下载RedisShake。本文以下载2.1.2版本为例，您可以根据实际需要下载其他[RedisShake版本](#)。

```
wget https://github.com/tair-opensource/RedisShake/releases/download/release-v2.1.2-20220329/release-v2.1.2-20220329.tar.gz
```

3. 执行命令解压RedisShake文件。

```
tar -xvf redis-shake-v2.1.2.tar.gz
```

```
[root@ecs-p-4-centos redisshake]# ll
total 16972
-rw-r--r-- 1 1320024 users 2749 Jun 24 16:15 ChangeLog
-rwxr-xr-x 1 1320024 users 14225 Jun 24 16:14 hypervisor
-rwxr-xr-x 1 1320024 users 13000971 Jun 24 16:14 redis-shake
-rw-r--r-- 1 1320024 users 8875 Jun 24 16:15 redis-shake.conf
-rw-r--r-- 1 root root 4326892 Jun 24 16:17 redis-shake.tar.gz
-rwxr-xr-x 1 1320024 users 458 Jun 24 16:14 start.sh
-rwxr-xr-x 1 1320024 users 374 Jun 24 16:14 stop.sh
```

4. 执行命令进入解压后的RedisShake文件目录。
cd redis-shake-v2.0.3
5. 编辑RedisShake工具配置文件redis-shake.conf，补充源端与目标端所有master节点信息。

vim redis-shake.conf

修改内容如下：

```
source.type = cluster
#若无密码，本项不填
source.password_raw = {source_redis_password}
#源Cluster集群所有master节点的IP地址与端口，以分号分隔
source.address = {master1_ip}:{master1_port};{master2_ip}:{master2_port}...{masterN_ip}:
{masterN_port}
target.type = cluster
#若无密码，本项不填
target.password_raw = {target_redis_password}
#目标Cluster集群所有master节点的IP地址与端口，以分号分隔
target.address = {master1_ip}:{master1_port};{master2_ip}:{master2_port}...{masterN_ip}:
{masterN_port}
```

修改后按下Esc键退出编辑模式，输入:wq! 按回车键保存配置并退出编辑界面。

在线迁移数据

使用如下命令同步源Redis集群和目标Redis集群数据：

```
./redis-shake -type sync -conf redis-shake.conf
```

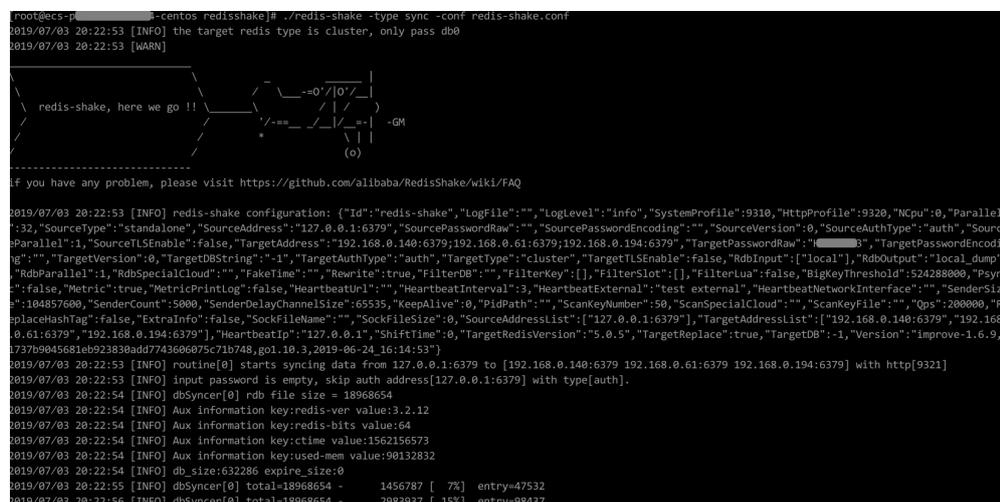
执行日志中出现如下信息，代表全量数据同步完成，进入增量同步阶段：

```
sync rdb done.
```

执行日志出现如下信息时，代表增量同步无新增内容，可手动停止同步（Ctrl + C）：

```
sync: +forwardCommands=0 +filterCommands=0 +writeBytes=0
```

图 11-9 RedisShake 在线迁移示意图



```
root@ecs-p-4-centos redisshake]# ./redis-shake -type sync -conf redis-shake.conf
2019/07/03 20:22:53 [INFO] the target redis type is cluster, only pass db0
2019/07/03 20:22:53 [WARN]
redis-shake, here we go !!
-----
if you have any problem, please visit https://github.com/alibaba/RedisShake/wiki/FAQ
2019/07/03 20:22:53 [INFO] redis-shake configuration: {"id":"redis-shake","logFile":"","logLevel":"info","systemProfile":"9310","httpProfile":"9320","ncpu":0,"parallel":32,"sourceType":"standalone","sourceAddress":"127.0.0.1:6379","sourcePasswordRaw":"","sourcePasswordEncoding":"","sourceAuthType":"auth","sourceParallel":1,"sourceTLSEnable":false,"targetAddress":"192.168.0.140:6379;192.168.0.61:6379;192.168.0.194:6379","targetPasswordRaw":"","targetPasswordEncoding":"","targetVersion":0,"targetDBString":-1,"targetAuthType":"auth","targetType":"cluster","targetTLSEnable":false,"rdbInput":{"local":"","rdbOutput":{"local_dump":"","rdbParallel":1,"rdbSpecialCloud":"","fakeTime":"","rewrite":true,"filterDB":"","filterKey":[],"filterslot":[],"filterlua":false,"bigkeyThreshold":524288000,"psync":false,"metric":true,"metricPrintLog":false,"heartbeatUrl":"","heartbeatInterval":3,"heartbeatExternal":"test external","heartbeatNetworkInterface":"","senderSize":104857600,"senderCount":5000,"senderDelayChannelSize":65535,"keepAlive":0,"pidPath":"","scanKeyNumber":50,"scanSpecialCloud":"","scanKeyFile":"","ops":200000,"replaceHashTag":false,"extraInfo":false,"sockFileName":"","sockFileSize":0,"sourceAddressList":["127.0.0.1:6379"],"targetAddressList":["192.168.0.140:6379","192.168.0.61:6379","192.168.0.194:6379"],"heartbeatIp":"127.0.0.1","shiftTime":0,"targetRedisVersion":"5.0.5","targetReplace":true,"targetDB":-1,"version":"improve-1.6.9,1737b9045681eb923830add7743606075c71b748.g01.10.3,2019-06-24 16:14:53"}
2019/07/03 20:22:53 [INFO] routine[0] starts syncing data from 127.0.0.1:6379 to [192.168.0.140:6379 192.168.0.61:6379 192.168.0.194:6379] with http[9321]
2019/07/03 20:22:53 [INFO] input password is empty, skip auth address[127.0.0.1:6379] with type[auth].
2019/07/03 20:22:54 [INFO] dbSyncer[0] rdb file size = 18968654
2019/07/03 20:22:54 [INFO] Aux information key:redis-ver value:3.2.12
2019/07/03 20:22:54 [INFO] Aux information key:redis-bits value:64
2019/07/03 20:22:54 [INFO] Aux information key:ctime value:1562156573
2019/07/03 20:22:54 [INFO] Aux information key:used-mem value:90132832
2019/07/03 20:22:54 [INFO] db_size:632286 expire_size:0
2019/07/03 20:22:55 [INFO] dbSyncer[0] total:18968654 - 1456787 [ 7%] entry=47532
2019/07/03 20:22:56 [INFO] dbSyncer[0] total:18968654 - 2983937 [ 15%] entry:98437
```


安装 RedisShake

1. 登录弹性云服务器ECS。
2. 在ECS中执行以下命令下载RedisShake。本文以下载2.1.2版本为例，您可以根据实际需要下载其他RedisShake版本。

```
wget https://github.com/tair-opensource/RedisShake/releases/download/release-v2.1.2-20220329/release-v2.1.2-20220329.tar.gz
```

3. 执行命令解压RedisShake文件。

```
tar -xvf redis-shake-v2.1.2.tar.gz
```

```
[root@ecs-p4-centos redisshake]# ll
total 16972
-rw-r--r-- 1 1320024 users 2749 Jun 24 16:15 ChangeLog
-rwxr-xr-x 1 1320024 users 14225 Jun 24 16:14 hypervisor
-rwxr-xr-x 1 1320024 users 13000971 Jun 24 16:14 redis-shake
-rw-r--r-- 1 1320024 users 8875 Jun 24 16:15 redis-shake.conf
-rw-r--r-- 1 root root 4326892 Jun 24 16:17 redis-shake.tar.gz
-rwxr-xr-x 1 1320024 users 458 Jun 24 16:14 start.sh
-rwxr-xr-x 1 1320024 users 374 Jun 24 16:14 stop.sh
```

说明

如果源Cluster部署在数据中心内网，则需在内网服务器上安装RedisShake，并参考[导出备份文件](#)导出源Cluster备份文件，然后将备份文件上传到云服务器。

导出备份文件

1. 执行命令进入解压后的RedisShake文件目录。

```
cd redis-shake-v2.0.3
```
2. 编辑RedisShake工具配置文件redis-shake.conf，补充源端所有master节点的连接信息。

```
vim redis-shake.conf
```

修改内容如下：

```
source.type = cluster
#若无密码，本项不填
source.password_raw = {source_redis_password}
#源Cluster集群所有master节点的IP地址与端口，以分号分隔
source.address = {master1_ip}:{master1_port};{master2_ip}:{master2_port}...{masterN_ip}:
{masterN_port}
```

修改后按下Esc键退出编辑模式，输入:**wq!** 按回车键保存配置并退出编辑界面。

3. 执行以下命令导出源Redis集群的RDB格式备份文件。

```
./redis-shake -type dump -conf redis-shake.conf
```

执行日志中出现如下信息时导出备份文件完成：

```
execute runner[*run.CmdDump] finished!
```

导入备份文件

1. 将导出的RDB备份文件（含多个）上传到与云服务器上。云服务器与目标端DCS Cluster集群实例的网络连通。
2. 编辑RedisShake工具配置文件redis-shake.conf。补充目标端所有master节点的连接信息。

```
vim redis-shake.conf
```

修改内容如下：

```
target.type = cluster
#若无密码，本项不填
target.password_raw = {target_redis_password}
#目标Cluster集群所有master节点的IP地址与端口，以分号分隔
```

```
target.address = {master1_ip}:{master1_port};{master2_ip}:{master2_port}...{masterN_ip}:  
{masterN_port}  
#需要导入的rdb文件列表, 用分号分隔  
rdb.input = {local_dump.0};{local_dump.1};{local_dump.2};{local_dump.3}
```

修改后按下Esc键退出编辑模式, 输入:**wq!** 按回车键保存配置并退出编辑界面。

3. 使用如下命令导入RDB备份文件到目标Cluster集群:

```
./redis-shake -type restore -conf redis-shake.conf
```

执行日志中出现如下信息时导入备份文件完成:

```
Enabled http stats, set status (incr), and wait forever.
```

迁移后验证

1. 数据同步结束后, 连接DCS Cluster集群, 连接Redis的方法请参考[Redis-cli客户端连接Redis](#)。
2. 通过info命令查看Keyspace中的Key数量, 确认数据是否完整导入。
如果数据不完整, 可使用flushall或者flushdb命令清理目标实例中的缓存数据后重新迁移。
3. 迁移验证完成后, 删除RedisShake配置文件。

11.5 其他云厂商 Redis 迁移至 DCS

11.5.1 使用迁移任务在线迁移其他云厂商 Redis

在满足源Redis和目标Redis的网络相通、源Redis放通SYNC和PSYNC命令这两个前提下的前提下, DCS支持通过在线迁移的方式, 将其他云厂商的Redis数据全量或增量迁移到DCS目标Redis中。

说明

在线迁移, 相当于临时给源端Redis增加一个从节点并且做一次全量同步到迁移机, 建议在业务低峰期执行迁移, 否则可能导致源端实例CPU瞬时冲高, 时延增大。

约束与限制

- 如果源Redis禁用了SYNC和PSYNC命令, 请务必联系源端云厂商放通SYNC和PSYNC命令, 否则会导致在线迁移失败。
- 在线迁移不支持公网方式直接迁移。
- 进行在线迁移时, 建议将源端实例的参数repl-timeout配置为300秒, client-output-buffer-limit配置为实例最大内存的20%。
- 源端仅支持Redis 3.0及3.0以上的Redis版本。
- 较早建立的实例如果密码中包含单引号('), 则该实例不支持进行在线迁移, 建议修改实例密码或使用其他迁移方式。
- 开启了SSL的目标实例不支持数据迁移, 需要关闭目标实例SSL后再进行迁移, 开启或关闭SSL的操作请参考[配置Redis SSL数据加密传输](#)。

前提条件

- 在迁移之前, 请先阅读[迁移方案概览](#), 选择正确的迁移方案, 了解当前DCS支持的在线迁移能力, 选择适当的目标实例。

- 如果是单机/主备等多DB的源端实例迁移到Proxy集群实例，Proxy集群默认不开启多DB，仅有一个DB0，请先确保源端实例DB0以外的DB是否有数据，如果有，请先参考[开启多DB操作](#)开启Proxy集群多DB设置。
- 如果是单机/主备等多DB的源端实例迁移到Cluster集群实例，Cluster集群不支持多DB，仅有一个DB0，请先确保源端实例DB0以外的DB是否有数据，如果有，请将数据转存到DB0，否则会出现迁移失败，将数据转存到DB0的操作请参考[使用Rump在线迁移](#)。
- 已获取准备迁移的源Redis实例的IP地址和端口。
- 如果您还没有目标Redis，请先创建目标Redis，具体操作请参考[购买Redis实例](#)。
- 如果您已有目标Redis，则不需要重复创建，为了对比迁移前后数据及预留足够的内存空间，建议在数据迁移之前清空目标实例数据，清空操作请参考[清空Redis实例数据](#)。如果没有清空实例数据，数据迁移后，目标Redis与源Redis实例重复的数据迁移后会被覆盖，源Redis没有、目标Redis有的数据会保留。

创建在线迁移任务

步骤1 请使用DCS目标Redis所在的账号登录分布式缓存服务控制台。

步骤2 在管理控制台左上角单击 ，选择DCS目标Redis所在的区域。

步骤3 单击左侧菜单栏的“数据迁移”。页面显示迁移任务列表页面。

步骤4 单击右上角的“创建在线迁移任务”。

步骤5 设置迁移任务名称和描述。

任务名称请以字母开头，长度不小于4位且不超过64位。任务名称只能包含字母、数字、中划线、下划线。

步骤6 配置在线迁移任务虚拟机资源的VPC、子网和安全组。

说明

- 请选择与目标Redis相同的VPC，确保迁移资源能访问目标Redis实例。
- 创建的在线迁移任务会占用一个租户侧IP，即控制台上迁移任务对应的“迁移IP”。如果源端Redis或目标端Redis配置了白名单，需确保配置了迁移IP或关闭白名单限制。
- 迁移任务所选安全组的“出方向规则”需放通源端Redis和目标端Redis的IP和端口（安全组默认情况下为全部放通，则无需单独放通），以便迁移任务的虚拟机资源能访问源Redis和目标Redis。

----结束

检查网络

步骤1 检查源Redis、目标Redis、迁移任务资源所在VPC是否为同一个VPC。

如果是，请执行[配置在线迁移任务](#)；如果不是，请执行[步骤2](#)。

步骤2 检查源Redis的VPC、目标Redis的VPC、迁移任务资源所在VPC的网络是否打通，确保迁移任务的虚拟机资源能访问源Redis和目标Redis。

如果已打通，则执行[配置在线迁移任务](#)；如果没打通，则执行[步骤3](#)。

步骤3 源Redis和目标Redis属于不同的云厂商，仅支持云专线打通网络，请参考[云专线](#)。

----结束

配置在线迁移任务

步骤1 创建完在线迁移任务之后，单击该迁移任务右侧的“配置”，配置在线迁移的源Redis、目标Redis等信息。

步骤2 选择迁移方法。

支持“全量迁移”和“全量迁移+增量迁移”两种，“全量迁移”和“全量迁移+增量迁移”的功能及限制如表11-7所示。

表 11-7 在线迁移方法说明

迁移类型	描述
全量迁移	该模式为Redis的一次性迁移，适用于可中断业务的迁移场景。全量迁移过程中， 如果源Redis有数据更新，这部分更新数据不会被迁移到目标Redis。
全量迁移+增量迁移	该模式为Redis的持续性迁移，适用于对业务中断敏感的迁移场景。增量迁移阶段通过解析日志等技术，持续保持源Redis和目标端Redis的数据一致。 增量迁移， 迁移任务会在迁移开始后，一直保持迁移中状态，不会自动停止。 需要您在合适时间，在“操作”列单击“停止”，手动停止迁移。停止后，源端数据不会造成丢失，只是目标端不再写入数据。增量迁移在传输链路网络稳定情况下是秒级时延，具体的时延情况依赖于网络链路的传输质量。

图 11-10 选择迁移方法



步骤3 仅当迁移方法选择“全量迁移+增量迁移”时，支持选择是否启用“带宽限制”。

如果启用带宽限制功能，当数据同步速度达到带宽限制时，将限制同步速度的继续增长。

步骤4 选择是否“自动重连”。如开启自动重连模式，迁移过程中在遇到网络等异常情况时，会无限自动重连。

自动重连模式在无法进行增量同步时，会触发全量同步，增加带宽占用，请谨慎选择。

步骤5 分别配置“源Redis”和“目标Redis”。

1. 配置“源Redis类型”和“源Redis实例”：

“源Redis类型”请选择“自建Redis”，并在“源Redis实例”处输入源Redis的IP地址和端口。

如果源Redis为Cluster集群，需要输入集群所有主节点的IP端口，用英文逗号隔开。例如：192.168.1.1:6379,192.168.0.0:6379

2. 配置“目标Redis类型”和“目标Redis实例”：
“目标Redis类型”请选择“云服务Redis”，并在“目标Redis实例”处选择需要迁移的目标Redis。
3. 分别配置“源Redis实例密码”和“目标Redis实例密码”：如果是密码访问模式实例，在输入连接实例密码后，单击密码右侧的“测试连接”，检查实例密码是否正确、网络是否连通。如果是免密访问的实例，请直接单击“测试连接”。如果测试连接失败，请检查输入的实例密码是否正确、Redis实例与迁移任务网络是否打通。
如果是DCS服务的Redis实例，此处暂不支持使用[账号管理](#)功能中创建的ACL账号及密码。
4. 在“源DB”或“目标DB”中，您可以选择是否需要指定具体迁移的DB。例如源端输入5，目标端输入6时，表示迁移源Redis DB5中的数据到目标Redis的DB6；当源端不指定DB，目标端指定DB时，表示默认迁移源端的全部数据，到目标端指定的DB，当目标端不指定DB时，表示默认迁移到与源端对应的DB。

📖 说明

- 当源端为多DB，目标端为单DB的DCS实例时（单DB的实例只有DB0），需要源端的所有数据都在DB0，或者指定仅迁移源端某一DB中的数据并将目标端DB指定为0，否则会迁移失败。
- DCS Redis的DB数请参见[Redis实例是否支持多DB方式?](#)。

步骤6 单击“下一步”。

步骤7 确认迁移信息，然后单击“提交”，开始创建迁移任务。

可返回迁移任务列表中，观察对应的迁移任务的状态，迁移成功后，任务状态显示“成功”。

如果出现迁移失败，建议单击迁移任务名称，进入迁移任务详情页面，通过“迁移日志”排查迁移失败的原因。

📖 说明

- 如果是增量迁移，全量迁移后会一直处于增量迁移中的状态。
- 如需手动停止迁移中的任务，勾选迁移任务左侧的方框，单击迁移任务上方的“停止”，即可停止迁移。
- 勾选停止迁移或迁移失败的迁移任务，单击迁移任务上方的“重启”，可重新进行迁移。如果重启迁移任务后迁移失败，建议单击“配置”，重新配置在线迁移任务后重试。
- 单次最多支持勾选50个在线迁移任务，批量停止、删除、或重启迁移任务。

---结束

迁移后验证

数据迁移前如果目标Redis中数据为空，迁移完成后，可以通过以下方式确认数据的完整性：

1. 连接源Redis和目标Redis。连接Redis的方法请参考[Redis-cli客户端连接Redis](#)。
2. 输入info keyspace，查看keys参数和expires参数的值。

```
192.168.1.217:6379> info keyspace
# Keyspace
db0:keys=81869,expires=0,avg_ttl=0
192.168.1.217:6379>
```

3. 对比源Redis和目标Redis的keys参数分别减去expires参数的差值。如果差值一致，则表示数据完整，迁移正常。

注意：如果是全量迁移，迁移过程中源Redis更新的数据不会迁移到目标实例。

11.5.2 使用备份文件离线迁移其他云厂商 Redis

本文档介绍如何通过备份文件导入的方式，将其他云厂商Redis离线迁移至DCS。

您需要先将源Redis的数据备份下载到本地，然后将备份数据文件上传到华为云与DCS目标Redis实例同一账号下相同Region下的OBS桶中，最后在DCS控制台创建备份迁移任务，DCS从OBS桶中读取数据，将数据迁移到DCS的Redis中。

约束与限制

开启了SSL的目标实例不支持数据迁移，需要关闭目标实例SSL后再进行迁移，开启或关闭SSL的操作请参考[配置Redis SSL数据加密传输](#)。

前提条件

- 在迁移之前，请先阅读[迁移方案概览](#)，选择正确的迁移方案，了解当前DCS支持的在线迁移能力，选择适当的目标实例。
- 如果是单机/主备等多DB的源端实例迁移到Proxy集群实例，Proxy集群默认不开启多DB，仅有一个DB0，请先确保源端实例DB0以外的DB是否有数据，如果有，请先参考[开启多DB操作](#)开启Proxy集群多DB设置。
- 如果是单机/主备等多DB的源端实例迁移到Cluster集群实例，Cluster集群不支持多DB，仅有一个DB0，请先确保源端实例DB0以外的DB是否有数据，如果有，请将数据转存到DB0，否则会出现迁移失败，将数据转存到DB0的操作请参考[使用Rump在线迁移](#)。
- 准备源Redis的备份文件，备份文件的格式必须为.aof、.rdb、.zip或.tar.gz。
- 如果您还没有目标Redis，请先创建目标Redis，具体操作请参考[购买Redis实例](#)。
- 如果您已有目标Redis，则不需要重复创建，为了对比迁移前后数据及预留足够的内存空间，建议在数据迁移之前清空目标实例数据，清空操作请参考[清空Redis实例数据](#)。如果没有清空实例数据，数据迁移后，目标Redis与源Redis实例重复的数据迁移后会被覆盖，源Redis没有、目标Redis有的数据会保留。

创建 OBS 桶并上传备份文件

如果上传的源Redis备份文件小于5GB，请执行以下步骤，通过OBS控制台创建OBS桶并上传备份文件。如果上传的源Redis备份文件大于5GB，请参考[超过5GB如何上传](#)上传备份文件。

步骤1 通过OBS控制台，创建OBS桶。

在创建过程中，以下两个参数请按要求设置，其他详细的创建步骤，请参考[创建桶](#)章节。

1. 选择“区域”。
OBS桶所在区域必须跟DCS目标Redis实例所在区域相同。
2. 设置“存储类别”，当前支持“标准存储”、“低频访问存储”和“归档存储”。
请不要选择“归档存储”，否则会导致备份文件迁移失败。

步骤2 在OBS管理控制台的桶列表中，单击**步骤1**中创建的桶名称，进入“概览”页面。

步骤3 在左侧导航栏，单击“对象”。

步骤4 在“对象”页签下，单击“上传对象”，系统弹出“上传对象”对话框。

步骤5 指定对象的存储类别。

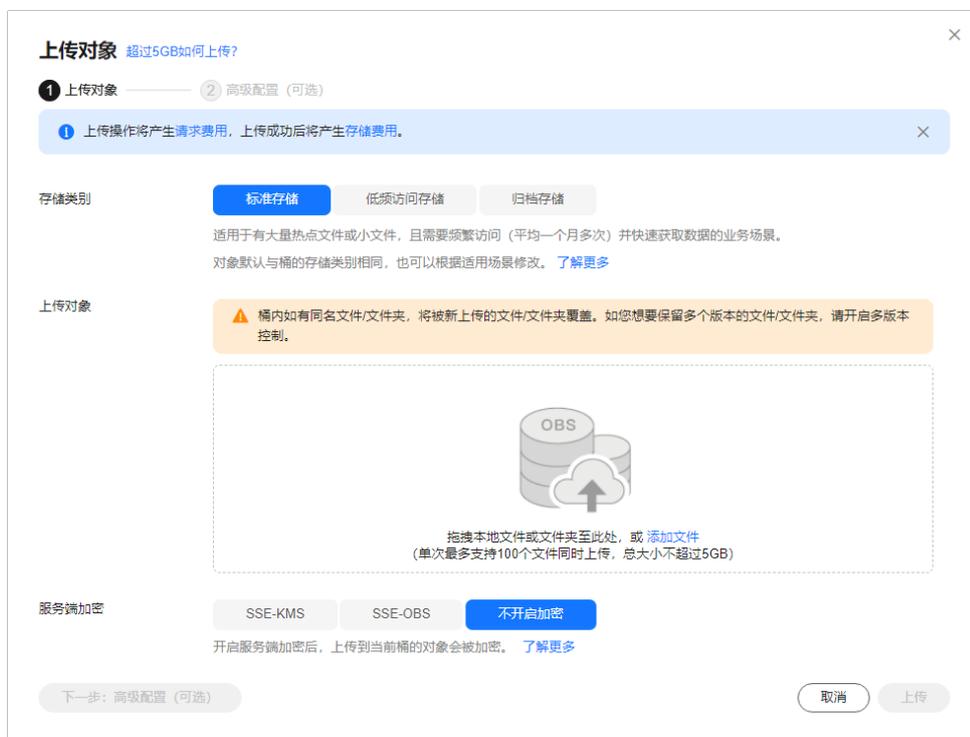
请不要选择“归档存储”，否则会导致备份文件迁移失败。

步骤6 上传对象。

您可以拖拽本地文件或文件夹至“上传对象”区域框内添加待上传的文件，也可以通过单击“上传对象”区域框内的“添加文件”，选择本地文件添加。

单次最多支持100个文件同时上传，总大小不超过5GB。

图 11-11 上传对象



步骤7 选择服务端加密方式，支持选择“SSE-KMS”、“SSE-OBS”或“不开启加密”，详情请参见**服务端加密**。

步骤8 单击“上传”。

----结束

创建迁移任务

步骤1 单击控制台左上角服务列表，选择“应用中间件 > 分布式缓存服务 Redis版”，进入分布式缓存服务。

步骤2 单击左侧菜单栏的“数据迁移”，进入数据迁移页面。

步骤3 单击右上角的“创建备份导入任务”。

步骤4 设置迁移任务名称和描述。

任务名称请以字母开头，长度不小于4位且不超过64位。任务名称只能包含字母、数字、中划线、下划线。

步骤5 “源Redis”区域中，“数据来源”选择“OBS桶”，在“OBS桶名”中选择已上传备份文件的OBS桶。**步骤6** 根据需要选择“源DB”，您可以指定源端备份文件某一个DB中的数据，例如输入5时，则只迁移DB5中的数据；无需指定DB时，请保持置空，即迁移全部DB。**步骤7** 选择“是否多DB Proxy集群”，只有当源Redis数据为DCS Proxy集群实例，且开启了多DB（Proxy实例multi-db参数值为yes）时选择。**步骤8** 单击“添加备份文件”，选择需要迁移的备份文件。**步骤9** 在“目标Redis”区域，选择**前提条件**中准备的“目标Redis实例”。**步骤10** 如果目标Redis是密码访问模式，请输入密码后，单击“测试连接”，检查密码是否正确。免密访问的实例，请直接单击“测试连接”。**步骤11** 根据需要选择“目标DB”，您可以指定迁移数据到目标Redis的某一个DB中，例如输入5时，则迁移到目标Redis的DB5；不填表示不指定，默认迁移到与源端相同的DB中。 **说明**

- 当源端为多DB，目标端为单DB的DCS实例时（单DB的实例只有DB0），需要源端的所有数据都在DB0，或者指定仅迁移源端某一DB中的数据并将目标端DB指定为0，否则会迁移失败。
- DCS Redis的DB数请参见[Redis实例是否支持多DB方式?](#)。

步骤12 单击“立即创建”。**步骤13** 确认迁移信息，然后单击“提交”，开始创建迁移任务。

可返回迁移任务列表中，观察对应的迁移任务的状态，迁移成功后，任务状态显示“成功”。

----**结束**

11.5.3 使用 Rump 在线迁移其他云厂商 Redis

部分云厂商的Redis实例禁止客户端发起SLAVEOF、BGSAVE、PSYNC等命令，无法使用Redis-cli、或RedisShake等工具快速导出数据。使用KEYS命令容易造成服务端阻塞。云厂商一般只提供备份文件下载，这种方式仅适宜离线迁移，且迁移过程对业务中断时间较长。

Rump是一款开源的Redis数据在线迁移工具，支持在同一个实例的不同数据库之间迁移数据，也支持不同实例的数据库之间迁移数据。本文档介绍如何通过Rump在线迁移其他云厂商Redis到DCS。

迁移原理

Rump使用**SCAN**来获取keys，用DUMP/RESTORE来get/set值。

SCAN是一个时间复杂度O(1)的命令，可以快速获得所有的key。DUMP/RESTORE使读/写值独立于键工作。

以下是Rump的主要特性：

- 通过SCAN非阻塞式的获取key，避免KEYS命令造成Redis服务阻塞。
- 支持所有数据类型的迁移。
- 把SCAN和DUMP/RESTORE操作放在同一个管道中，利用pipeline提升数据迁移过程中的网络效率。
- 不使用任何临时文件，不占用磁盘空间。
- 使用带缓冲区的channels，提升源服务器的性能。

约束与限制

- 使用Rump工具在线迁移，目标端不支持DCS集群类型实例。
- Redis实例的密码不能包含#@等特殊字符，避免迁移命令解析出错。
- 正式进行迁移操作前，建议先暂停业务。迁移过程中如果不断写入新的数据，可能会丢失少量Key。
- 开启了SSL的目标实例不支持数据迁移，需要关闭目标实例SSL后再进行迁移，开启或关闭SSL的操作请参考[配置Redis SSL数据加密传输](#)。

前提条件

- 如果您还没有目标Redis，请先创建目标Redis，具体操作请参考[购买Redis实例](#)。
- 如果您已有目标Redis，则不需要重复创建，为了对比迁移前后数据及预留足够的内存空间，建议在数据迁移之前清空目标实例数据，清空操作请参考[清空Redis实例数据](#)。如果没有清空实例数据，数据迁移后，目标Redis与源Redis实例重复的数据迁移后会被覆盖，源Redis没有、目标Redis有的数据会保留。
- 已创建弹性云服务器ECS，创建弹性云服务器的方法，请参见[创建弹性云服务器](#)。ECS请选择与DCS Cluster集群实例相同虚拟私有云、子网和安全组，并且需要绑定弹性公网IP。

安装 Rump

1. 登录弹性云服务器。
2. 下载Rump的[release版本](#)。

以64位Linux操作系统为例，执行以下命令：

```
wget https://github.com/stickermule/rump/releases/download/0.0.3/rump-0.0.3-linux-amd64;
```

3. 解压缩后，添加可执行权限。

```
mv rump-0.0.3-linux-amd64 rump;  
chmod +x rump;
```

迁移数据

执行如下命令迁移数据：

```
rump -from {source_redis_address} -to {target_redis_address}
```

- {source_redis_address}

源Redis实例地址，格式为：redis://[user:password@]host:port/db，中括号部分为可选项，实例设置了密码访问时需要填写密码，格式遵循RFC 3986规范。注意用户名可为空，但冒号不能省略，例如

```
redis://:mypassword@192.168.0.45:6379/1。
```

db为数据库编号，不传则默认为0。

- `{target_redis_address}`
目标Redis实例地址，格式与from相同。
以下示例表示将本地Redis数据库的第0个DB的数据迁移到192.168.0.153这台Redis数据库中，其中密码以*替代显示。

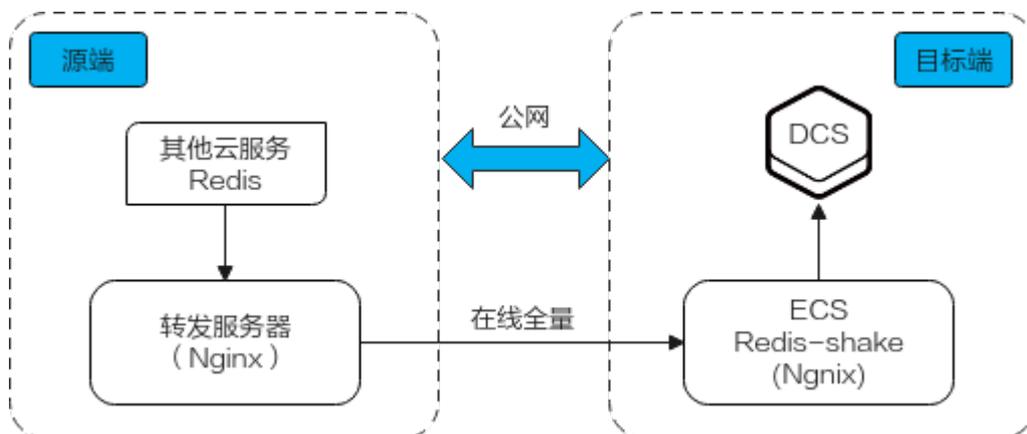
```
[root@ecs ~]# ./rump -from redis://127.0.0.1:6379/0 -to redis://:*@192.168.0.153:6379/0
.Sync done.
[root@ecs ~]#
```

11.5.4 使用 RedisShake 在线迁移其他云厂商 Redis

RedisShake是一款开源的Redis迁移工具，在Rump模式下，RedisShake可以以scan的方式从源端Redis获取全量数据，写入到目的端，实现数据迁移。这种迁移方式不依赖于SYNC和PSYNC，可以广泛应用于自建Redis、云Redis之间的迁移。

本文将介绍如何利用RedisShake的Rump模式，以在线全量的迁移方式，一次性将其他云厂商Redis迁移至华为云DCS中。

图 11-12 本方案数据流向示意图



约束与限制

- Rump模式不支持增量数据迁移，建议您先停止源端Redis的写入再进行迁移，防止数据不一致。
- 该方案配置只支持同DB映射迁移，异DB映射迁移该方案配置不适用。
- 源端为多DB使用（有非DB0的DB使用），华为云DCS为Proxy集群时，DCS需要开启多DB模式，否则会迁移失败（单DB0的Proxy集群不支持select命令）。
- 源端为多DB使用（有非DB0的DB使用），华为云DCS为Cluster集群时，该方案不支持（DCS Cluster集群只支持DB0模式）。
- 开启了SSL的目标实例不支持数据迁移，需要关闭目标实例SSL后再进行迁移，开启或关闭SSL的操作请参考[配置Redis SSL数据加密传输](#)。

前提条件

- 已创建[Redis实例](#)。
- 已创建用于运行RedisShake的[弹性云服务器\(ECS\)](#)，创建的ECS需要选择与Redis实例相同的VPC，并且需要绑定弹性公网IP。

迁移步骤

步骤1 分别在华为云ECS和源端转发服务器上安装Nginx，本文以ECS操作系统为Centos7.x为例进行安装，不同操作系统命令稍有不同。

1. 执行以下命令，添加Nginx到yum源。

```
sudo rpm -Uvh http://nginx.org/packages/centos/7/noarch/RPMS/nginx-release-centos-7-0.el7ngx.noarch.rpm
```
2. 添加完之后，执行以下命令，查看是否已经添加成功。

```
yum search nginx
```
3. 添加成功之后，执行以下命令，安装Nginx。

```
sudo yum install -y nginx
```
4. 执行以下命令安装stream模块。

```
yum install nginx-mod-stream --skip-broken
```
5. 启动Nginx并设置为开机自动运行。

```
sudo systemctl start nginx.service  
sudo systemctl enable nginx.service
```
6. 在本地浏览器中输入服务器地址（ECS公网IP地址），查看安装是否成功。如果出现下面页面，则表示安装成功。



步骤2 在源端Redis添加源端转发服务器的白名单。

步骤3 在源端转发服务器配置安全组。

1. 获取华为云ECS的公网IP地址。
2. 配置源端转发服务器安全组入方向，添加华为云ECS的公网IP地址，并放开来自华为云ECS访问请求的端口（以6379为例）。

步骤4 配置源端转发服务器的Nginx转发配置。

1. 登录Linux源端转发服务器，执行命令打开并修改配置文件。

```
cd /etc/nginx  
vi nginx.conf
```

2. 转发配置示例如下：

```
stream {  
    server {  
        listen 6379;  
        proxy_pass {source_instance_address}:{port};  
    }  
}
```

其中，6379为源端转发服务器本机监听端口，{source_instance_address}和{port}为源端Redis实例的连接地址和端口。

配置目的：通过访问源端转发服务器本机监听端口6379，访问源端Redis。

注意：以上配置必须配置在如下图所示的位置。

图 11-13 配置位置要求

```
# Load dynamic modules. See /usr/share/doc/nginx/README.dynamic.
include /usr/share/nginx/modules/*.conf;

events {
    worker_connections 1024;
}

stream {
    server {
```

3. 重启Nginx服务。
service nginx restart
4. 验证启动是否成功。
netstat -an|grep 6379
端口在监听状态，Nginx启动成功。

图 11-14 验证结果

```
tcp        0      0 0.0.0.0:6379          0.0.0.0:*           LISTEN
```

步骤5 配置华为云ECS的Nginx转发配置。

1. 登录Linux华为云ECS，执行命令打开并修改配置文件。
cd /etc/nginx
vi nginx.conf

2. 配置示例如下：

```
stream {
    server {
        listen 6666;
        proxy_pass {source_ecs_address}:6379;
    }
}
```

其中，6666为华为云ECS本机监听端口，{source_ecs_address}为源端转发服务器公网IP地址，6379为源端转发服务器Nginx的监听端口。

配置目的：通过访问华为云ECS本机监听端口6666，访问源端转发服务器。

注意：以上配置必须配置在如下图所示的位置。

图 11-15 配置位置要求

```
# Load dynamic modules. See /usr/share/doc/nginx/README.dynamic.
include /usr/share/nginx/modules/*.conf;

events {
    worker_connections 1024;
}

stream {
    server {
```

3. 重启Nginx服务。
service nginx restart
4. 验证启动是否成功。
netstat -an|grep 6666
端口在监听状态，Nginx启动成功。

图 11-16 验证结果

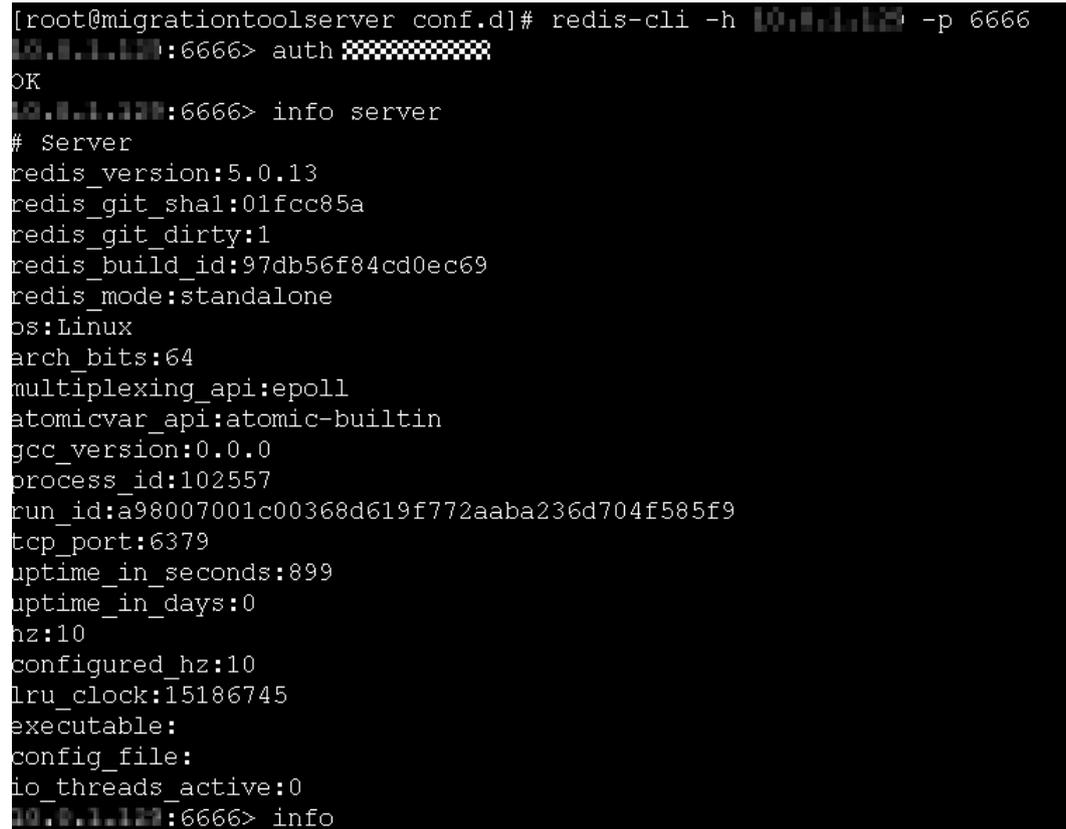


步骤6 在华为云ECS执行以下命令测试6666端口的网络连接。

```
redis-cli -h {target_ecs_address} -p 6666 -a {password}
```

其中，{target_ecs_address}为华为云ECS公网IP地址，6666为华为云ECS监听端口，{password}为源端Redis密码，如无密码可不填。

图 11-17 连接示例



步骤7 准备迁移工具RedisShake。

1. 登录华为云ECS。
2. 在华为云ECS中执行以下命令下载RedisShake，本文以下载2.0.3版本为例进行说明。您可以根据实际需要下载其他RedisShake版本。

```
wget https://github.com/tair-opensource/RedisShake/releases/download/release-v2.0.3-20200724/redis-shake-v2.0.3.tar.gz
```
3. 执行命令解压RedisShake文件。

```
tar -xvf redis-shake-v2.0.3.tar.gz
```

步骤8 配置RedisShake的配置文件。

1. 执行命令进入解压后的目录。

```
cd redis-shake-v2.0.3
```
2. 修改配置文件redis-shake.conf。

```
vim redis-shake.conf
```

修改源端Redis信息配置：

```
- source.type
```

源端redis实例类型，单机、主备、proxy集群实例都选择standalone，cluster实例选择cluster。

- source.address

华为云ECS公网IP地址和映射源端转发服务器的端口(华为云ECS监听端口6666)，用英文冒号隔开。

- source.password_raw

源端待迁移Redis实例的密码，如未设置密码，无需填写。

修改目标端DCS信息配置：

- target.type

华为云Redis实例类型，单机、主备、proxy集群实例都选择standalone，cluster实例选择cluster。

- target.address

华为云Redis实例的连接地址和端口，用英文冒号隔开。

- target.password_raw

华为云Redis实例的密码，如未设置密码，无需填写。

3. 按下Esc键退出编辑模式，输入:wq! 按回车键保存配置并退出编辑界面。

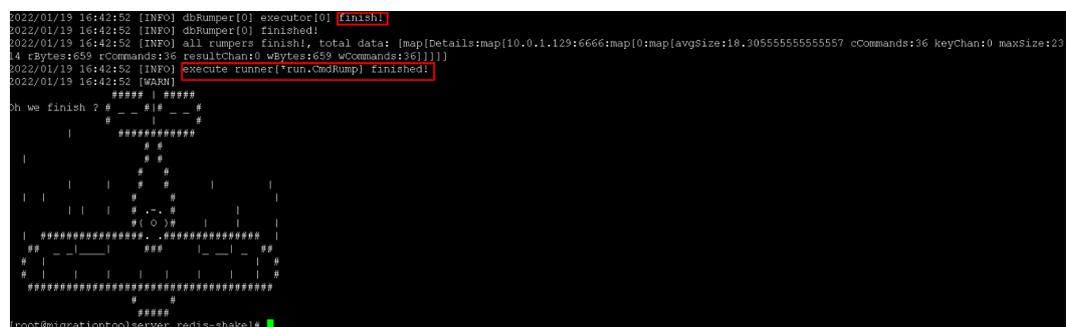
步骤9 执行命令启动RedisShake并使用rump（在线全量）模式开始数据迁移。

```
./redis-shake.linux -conf redis-shake.conf -type rump
```

图 11-18 迁移过程



图 11-19 迁移结果



步骤10 迁移完成后，请使用redis-cli工具连接源Redis和目标Redis，确认数据的完整性。

1. 分别连接源Redis和目标Redis。

连接操作请参考[redis-cli连接](#)。

2. 输入info keyspace，查看keys参数和expires参数的值。

- 对比源Redis和目标Redis的keys参数分别减去expires参数的差值。如果差值一致，表示数据完整，迁移正常。

步骤11 删除RedisShake配置文件。

----结束

11.5.5 使用 RedisShake 离线迁移其他云厂商 Redis

RedisShake是一款开源的Redis迁移工具，支持Cluster集群的在线迁移与离线迁移（备份文件导入）。当部署在其他云厂商Redis服务上的Cluster集群数据，无法在线迁移时，可以选择离线迁移。

与在线迁移相比，离线迁移适用于源实例与目标实例的网络无法连通的场景，或者源端实例部署在其他云厂商Redis服务中，无法实现在线迁移。

本文以Linux系统环境为例，介绍如何使用RedisShake工具进行Cluster集群数据离线迁移。

约束与限制

开启了SSL的目标实例不支持数据迁移，需要关闭目标实例SSL后再进行迁移，开启或关闭SSL的操作请参考[配置Redis SSL数据加密传输](#)。

前提条件

- 已创建[Redis实例](#)，注意创建的Cluster集群内存规格不能小于源端Cluster集群。
- 已创建用于运行RedisShake的[弹性云服务器\(ECS\)](#)，创建的ECS需要选择与Redis实例相同的VPC、子网和安全组。

迁移步骤

- 使用[Redis-cli](#)连接目标端Redis实例，获取目标端Cluster集群的Master节点IP地址与端口。

```
redis-cli -h {target_redis_address} -p {target_redis_port} -a {target_redis_password} cluster nodes
```

- {target_redis_address}：目标Redis实例的连接地址。
- {target_redis_port}：目标Redis实例的连接端口号。
- {target_redis_password}：目标Redis实例的连接密码。

在命令返回的结果中，获取所有master节点的IP端口，如下所示：

```
[root@ecs-54-centos ~]# redis-cli -h 192.168.0.140 -p 6379 -a 23 cluster nodes
fb75f0743af4695a3d241ff7790b2f508e4985ff 192.168.0.140:6379@16379 myself,master - 0 1562144170000 3 connected
d112bae791b2bbd9602fe32963536b8a0db9eb79 192.168.0.61:6379@16379 master - 0 1562144171524 1 connected 0-5460
73e2f8fe196166f9ad1283361867d24c136413f0 192.168.0.194:6379@16379 master - 0 1562144170000 2 connected 5461-10
40d72299fde6045de0f79ee4b97910b505acbc6a 192.168.0.231:6379@16379 slave 73e2f8fe196166f9ad1283361867d24c136413
be6c07faa64d724323e0d7cedc3f38346dcdb212 192.168.0.80:6379@16379 slave fb75f0743af4695a3d241ff7790b2f508e4985f
c16b9acaee7dd0721f129596cd43bd499c0e396 192.168.0.169:6379@16379 slave d112bae791b2bbd9602fe32963536b8a0db9eb
```

- 在准备好的华为云ECS上安装迁移工具RedisShake。
 - 登录华为云ECS。
 - 在华为云ECS中执行以下命令下载RedisShake，本文以下载2.0.3版本为例进行说明。您可以根据实际需要下载其他[RedisShake版本](#)。

```
wget https://github.com/tair-opensource/RedisShake/releases/download/release-v2.0.3-20200724/redis-shake-v2.0.3.tar.gz
```
 - 执行命令解压RedisShake文件。

```
tar -xvf redis-shake-v2.0.3.tar.gz
```

📖 说明

如果源端集群部署在数据中心内网，还需在内网服务器上安装RedisShake，并参考下述步骤进行数据导出，然后将数据文件上传到云服务器。

3. 从源端Redis控制台导出RDB文件，如果无法导出RDB文件，请联系源端客服获取。
4. 导入RDB文件。

- a. 将导出的RDB文件（含多个）上传到云服务器上。云服务器与目标端DCS Cluster集群实例的网络连通。

- b. 编辑RedisShake工具配置文件redis-shake.conf。

```
vim redis-shake.conf
```

补充目标端所有master节点的连接信息：

```
target.type = cluster
#若无密码，本项不填
target.password_raw = {target_redis_password}
#目标Cluster集群所有master节点的IP地址与端口，以分号分隔
target.address = {master1_ip};{master1_port};{master2_ip};{master2_port}...{masterN_ip};{masterN_port}
#需要导入的rdb文件列表，用分号分隔
rdb.input = {local_dump.0};{local_dump.1};{local_dump.2};{local_dump.3}
```

编辑完成后，按下Esc键退出编辑模式，输入:**wq!** 按回车键保存配置并退出编辑界面。

- c. 使用如下命令导入RDB文件到目标Cluster集群：

```
./redis-shake -type restore -conf redis-shake.conf
```

执行日志中出现如下信息时导入备份文件完成：

```
Enabled http stats, set status (incr), and wait forever.
```

5. 迁移后验证。

数据同步结束后，可使用redis-cli工具连接目标端DCS Cluster集群，通过**info**命令查看Keyspace中的Key数量，确认数据是否完整导入。

如果数据不完整，可使用**flushall**或者**flushdb**命令清理实例中的缓存数据后重新同步。

12 测试实例性能

12.1 使用 memtier_benchmark 测试 Redis 性能

Memtier_benchmark是Redis Labs推出的一款命令行工具，它能够产生多种流量模式，对Redis实例进行基准测试。该工具提供了丰富的自定义选项和报表功能，通过命令行界面就能够轻松地使用。了解memtier_benchmark更多详情，请访问https://github.com/RedisLabs/memtier_benchmark。

针对DCS Redis实例的性能评估，可以使用memtier_benchmark测试某种规格的实例在某个高并发场景下执行SET或GET时的性能。

测试步骤

步骤1 创建Redis缓存实例。

步骤2 创建3台弹性云服务器（ECS），ECS选择与实例相同可用区、VPC、子网和安全组。

说明

如果是测试单机或主备实例，创建1台ECS即可。

步骤3 在每台ECS上安装[memtier_benchmark](#)。以操作系统为CentOS 8.0为例进行安装。

1. 准备工作。

a. 安装编译所需的工具。

```
yum install -y autoconf
yum install -y automake
yum install -y make
yum install -y gcc-c++
yum install -y git
```

b. 启用PowerTools存储库的方法。

```
dnf config-manager --set-enabled PowerTools
```

c. 安装需要的依赖库。

```
yum install -y pcre-devel
yum install -y zlib-devel
```

```
yum install -y libmemcached-devel
yum install -y openssl-devel
```

2. 安装libevent库。

```
yum install -y libevent-devel
```

3. 下载、编译、安装memtier_benchmark的库。

- a. 根目录下创建文件夹，用于下载memtier_benchmark。

```
mkdir /env
```

- b. 下载memtier_benchmark源码。

```
cd /env
```

```
git clone https://github.com/RedisLabs/memtier_benchmark.git
```

- c. 进入源码目录。

```
cd memtier_benchmark
```

- d. 编译生成可执行文件memtier_benchmark。

```
autoreconf -ivf
```

```
./configure
```

```
make
```

- e. 将工具安装到系统中。

```
make install
```

4. 运行帮助命令查看是否安装成功。如果返回memtier_benchmark的参数说明，表示安装成功。

```
memtier_benchmark --help
```

步骤4 每台ECS上执行测试命令。

```
memtier_benchmark -s {IP} -n {nreqs} -c {connect_number} -t 4 -d {datasize}
```

📖 说明

如果实例类型为cluster集群，则命令为memtier_benchmark --cluster-mode -s {IP} -n {nreqs} -c {connect_number} -t 4 -d {datasize}

参数参考值：-c {connect_number}: 200, -n {nreqs}: 10000000, -d {datasize}: 32。

- -s表示实例的域名连接地址或IP地址。
- -t表示基准测试使用的线程数量
- -c表示客户端连接数
- -d表示单条数据大小，单位byte
- -n表示测试包数量

步骤5 不断调整客户端连接数，执行步骤4，得到最大的QPS（Query Per Second，表示每秒处理的读写操作数，单位：次/秒）。

步骤6 取3台测试ECS得到的每秒操作数总和，即为对应规格的性能数据。

如果测试Redis集群，建议每台测试ECS各开启两个benchmark客户端。

----结束

12.2 使用 redis-benchmark 测试 Redis 性能

Redis客户端源码包含一个名为redis-benchmark的性能测试工具，它可以模拟N个客户端同时向Redis发送M条查询命令的应用场景。

针对DCS Redis实例的性能评估，可以使用redis-benchmark测试某种规格的实例在某个高并发场景下执行SET或GET时的性能。

测试步骤

步骤1 创建Redis缓存实例。

步骤2 创建3台弹性云服务器（ECS），ECS选择与实例相同可用区、VPC、子网和安全组。

📖 说明

如果是测试单机或主备实例，创建1台ECS即可。

步骤3 在每台ECS上安装redis-benchmark。可通过以下两种方式安装Redis-server，安装Redis-server的同时，会同步安装benchmark。

- 安装方法一：

a. 下载redis客户端，此处以redis-6.0.9版本为例。

wget http://download.redis.io/releases/redis-6.0.9.tar.gz

b. 解压客户端压缩包。

tar xzf redis-6.0.9.tar.gz

c. 进入redis-6.0.9的src目录下。

cd redis-6.0.9/src

d. 编译源码。

make

编译完成后，工具一般在redis-x.x.x的src目录下。

e. 查看是否有redis-benchmark可执行文件。

ls

```
[root@ ~]# cd src# ls
adlist.c      config.h      geohash_helper.h  lzfp.h          rax.o          scripting.o    t_hash.c
adlist.h      config.o      geohash_helper.o  Makefile        rdb.c          sdsalloc.h    t_hash.o
adlist.o      crc16.c      geohash.o         memtest.c       rdb.h          sds.c         t_list.c
ae.c          crc16.o      geo.o             memtest.o       rdb.o          sds.h         t_list.o
ae_epoll.c    crc64.c      help.h            mkreleasestr.sh redisassert.h  sds.o         t_set.c
ae_evport.c   crc64.h      hyperloglog.c    module.c        redis-benchmark sentinel.c     t_set.o
ae.h          crc64.o      hyperloglog.o    module.o        redis-benchmark.c sentinel.o    t_stream.c
ae_kqueue.c   db.c         intset.c          multi.c         redis-check-aof server.c       t_stream.o
ae.o          db.o         intset.h          multi.o         redis-check-aof .o server.h      t_string.c
ae_select.c   debug.c      intset.o          multi.o         redis-check-aof.c server.o       t_string.o
anet.c        debugmacro.h latency.c         networking.c    redis-check-aof.o setproctitle.c t_zset.c
anet.h        debug.o      latency.h         networking.o    redis-check-rdb sha1.c         util.c
anet.o        defrag.c     latency.o         notify.c        redis-check-rdb.o sha1.h        util.h
aof.c         defrag.o     lazyfree.c       notify.o        redis-cli       sha1.o        util.o
aof.o         dict.c       listpack.c        object.c        redis-cli.c     siphash.c    valgrind.sup
asciilogo.h   dict.o       listpack.h        object.o        redis-cli.o     siphash.o    version.h
atomicvar.h   dict.o       listpack_malloc.h pgsort.c        redismodule.h  slowlog.c     ziplist.c
bio.c         endianconv.c listpack.o        pgsort.h        redis-sentinel slowlog.h     ziplist.h
bio.h         endianconv.h listpack.o        pgsort.o        redis-server    slowlog.o     ziplist.o
bio.o         endianconv.o localtime.c       pubsub.c        redis-trib.rb  solarisfixes.h zipmap.c
bitops.c     evict.c      localtime.o       pubsub.o        release.c       sort.o        zipmap.h
bitops.o     expire.c     lolwut5.c         quicklist.c     release.h       sort.o        zipmap.o
blocked.c    expire.o     lolwut5.o         quicklist.h     replication.c  sparkline.c   zmalloc.c
blocked.o    expire.o     lolwut.c          quicklist.o     replication.o  sparkline.o   zmalloc.h
childinfo.c  fmacros.h   lolwut.o          rand.c          rax.o          stream.h      zmalloc.o
childinfo.o  geo.c       lolwut.o          rand.h          rax.c          stream.o      zmalloc.o
cluster.c    geo.h       lolwut.o          rand.o          rax.h          syncio.c     zmalloc.o
cluster.h    geo.h       lolwut.o          rand.o          rax.o          syncio.o     zmalloc.o
cluster.o    geohash.c   lolwut.o          rand.o          rax.o          syncio.o     zmalloc.o
cluster.o    geohash.h   lolwut.o          rand.o          rax.o          syncio.o     zmalloc.o
config.c     geohash_helper.c lzfp.h           rax_malloc.h   scripting.c     testlib.h
```

f. 将工具安装到系统中。

make install

- 安装方法二：

根据ECS的不同的操作系统直接安装Redis-server，下面以ubuntu和CentOS系统为例：

- ubuntu系统

```
sudo apt update  
sudo apt install redis-server
```

- CentOS系统

```
sudo yum install epel-release  
sudo yum update  
sudo yum -y install redis
```

步骤4 每台ECS上执行测试命令。

```
redis-benchmark -h {IP} -p {Port} -a {password} -n {nreqs} -r {randomkeys} -c {connect_number} -d {datasize} -t {command}
```

参数参考值：-c {connect_number}: 200, -n {nreqs}: 10000000, -r {randomkeys}: 1000000, -d {datasize}: 32。

- -h表示实例的域名连接地址或IP地址。
- -p表示实例的端口，默认为6379。
- -a表示实例的连接密码，免密连接的实例无需输入-a {password}。
- -t表示执行具体测试命令合集。例如只测试set命令时，使用-t set；如果要测试ping、get、set命令，则使用 -t ping,set,get，命令间使用“,”分隔。
- -c表示客户端连接数。
- -d表示单条数据大小，单位Byte。
- -n表示测试包数量。
- -r表示使用随机key数量。

步骤5 不断调整客户端连接数，执行**步骤4**，得到最大的QPS（Query Per Second，表示每秒处理的读写操作数，单位：次/秒）。

步骤6 取3台测试ECS得到的每秒操作数总和，即为对应规格的性能数据。

如果测试Redis集群，建议每台测试ECS各开启两个benchmark客户端。

📖 说明

- redis-benchmark 测试cluster集群实例时需要加 --cluster 参数，其他实例类型不需要加。
- 如果想对cluster集群的最大连接数进行性能压测，但是压测到1万连接时程序退出，或者报错 Cannot assign requested address。这说明是测试用的ECS本机性能不足，请先检查自己是否只用了1台ECS进行压测。想要对集群压测，建议准备3台ECS，每台ECS起3个redis-benchmark来测试redis实例的最大连接数。

---结束

redis-cli 常用选项

- -h <hostname>：服务器的主机名，可以是IP或者域名
- -p <port>：服务器的端口，默认是6379
- -a <password>：连接服务器的密码，免密连接的实例无需输入-a {password}
- -r <repeat>：执行指定命令N次

- `-n <db>` : 数据库编号, 默认是0
- `-c` : 启用集群模式 (遵循-ASK和-MOVED重定向)
- `--latency` : 进入特殊模式连续采样延迟
- `--scan` : 非阻塞式的扫描键空间 (区别于`keys *`扫描键空间会导致redis-server阻塞)
- `--eval <file>` : 使用Lua脚本发送EVAL命令
- `-x` : 读取STDIN中的最后一个参数
- `--bigscan` : 扫描数据集的大key
- `--raw` : 如果显示数据是这样的'\xe4\xb8'十六进制编码, 可以尝试加`--raw`转为直接显示原始数据

了解redis-cli更多详情, 请访问 <https://redis.io/docs/connect/cli/>

redis-cli 常用命令举例

- 连接实例:
`./redis-cli -h {IP} -p 6379`
- 指定连接某个DB:
`./redis-cli -h {IP} -p 6379 -n 10`
- 连接cluster集群实例:
`./redis-cli -h {IP} -p 6379 -c`
- 测试时延 (原理是发ping命令):
`./redis-cli -h {IP} -p 6379 --latency`
- 执行scan扫描匹配指定模式的key:
`./redis-cli -h {IP} -p 6379 --scan --pattern '*:12345*'`

redis-benchmark (redis-6.0.9)常用选项

- `-h <hostname>` : 服务器的主机名, 可以是IP或者域名。
- `-p <port>` : 服务器的端口, 默认是6379。
- `-a <password>` : 连接服务器的密码, 免密连接的实例无需输入`-a {password}`。
- `-c <clients>` : 并发连接数, 默认50。
- `-n <requests>` : 请求总数 (默认为100000)。
- `-d <size>` : SET/GET值的数据大小 (以字节为单位, 默认值2)。
- `--dbnum <db>` : 选择指定的数据库编号 (默认值0)。
- `--threads <num>` : 启动多线程模式 (redis 6.0版本编译的redis-benchmark才支持, 多线程压测redis的性能优于单线程)。
- `--cluster` : 启动集群模式 (cluster集群才需要该参数)。
- `-k <boolean>` : 1=keep alive 0=reconnect (默认值1, 可以测试长短连接)。
- `-r <keyspacelen>` : 对SET/GET/INCR使用随机键, 对SADD使用随机值。参数中`keyspacelen`指的是添加键的数量。
- `-e` : 如果服务器回复错误, 请在stdout上显示它们。
- `-q` : 只展示query/sec的值。

- -l：循环测试。
- -t <tests>：可以对指定的命令进行测试。
- -l：空闲模式，仅打开N个空闲连接并等待。
- -P <numreq>：管道请求的并发数量（默认值为1）。

了解redis-benchmark更多详情，请访问：https://redis.io/docs/latest/operate/oss_and_stack/management/optimization/benchmarks/

redis-benchmark 常用命令举例

- 单机、主备、读写分离和proxy集群的测试命令：
./redis-benchmark -h {IP或域名} -p 6379 -a {password} --threads {num} -n {nreqs} -r {randomkeys} -c {clients} -d {datasize} -t {command}
- cluster集群测试命令：
./redis-benchmark -h {IP或域名} -p 6379 -a {password} --threads {num} -n {nreqs} -r {randomkeys} -c {clients} -d {datasize} --cluster -t {command}
- 测试短连接：
./redis-benchmark -h {IP或域名} -p 6379 -a {password} --threads {num} -n {nreqs} -r {randomkeys} -c {clients} -d {datasize} -k 0 -t {command}
- 测试空闲连接：
./redis-benchmark -h {IP或域名} -p 6379 -a {pwd} -c {clients} -l

12.3 redis-benchmark 与 memtier_benchmark 的差异

工具	支持 Memcached	支持读写比例设置	支持Payload大小随机	支持设置过期时间
memtier_benchmark	是	是	是	是
redis-benchmark	否	否	否	否

12.4 Redis 性能测试数据参考

12.4.1 Redis 3.0 主备实例测试数据

测试环境说明

- 测试实例规格
Redis 3.0 8G主备
Redis 3.0 32G主备
- 测试执行机规格
通用计算增强型 | c6.xlarge.2 | 4vCPUs | 8GB

测试命令

```
redis-benchmark -h {IP} -p {Port} -a {password} -n {nreqs} -r {randomkeys} -c {connection} -d {datasize} -t {command}
```

参数参考值：-c {connect_number}: 1000, -n {nreqs}: 10000000, -r {randomkeys}: 1000000, -d {datasize}: 32。

测试结果

表 12-1 SET 操作命令测试结果

实例规格	实例CPU类型	并发连接数 (个)	QPS	99.99%延迟 (ms)	第一个100%延迟(ms)	最后一个100%延迟 (ms)
8G	X86	1000	10765 7.69	20	23	27
		10000	72750. 55	362	366	371
32G	X86	1000	12108 8.83	9	12	12
		10000	79235. 53	203	204	267

表 12-2 GET 操作命令测试结果

实例规格	实例CPU类型	并发连接数 (个)	QPS	99.99%延迟 (ms)	第一个100%延迟(ms)	最后一个100%延迟 (ms)
8G	X86	1000	11935 0.25	6	24	27
		10000	77574. 7	152	358	365
32G	X86	1000	12465 0.98	16	17	17
		10000	81991. 41	195	196	199

说明

Redis 3.0实例暂不支持ARM CPU类型实例，仅提供X86类型的实例测试结果。

12.4.2 Redis 3.0 Proxy 集群实例测试数据

测试环境说明

- 测试实例规格
Redis 3.0 64G Proxy 集群
- 测试执行机规格
通用计算增强型 | c6.xlarge.2 | 4vCPUs | 8GB

测试命令

```
redis-benchmark -h {IP} -p {Port} -a {password} -n {nreqs} -r {randomkeys} -c {connection} -d {datasize} -t {command}
```

参数参考值：-c {connect_number}: 1000, -n {nreqs}: 10000000, -r {randomkeys}: 1000000, -d {datasize}: 32。

测试结果

表 12-3 SET 操作命令测试结果

实例规格	实例 CPU 类型	并发连接数 (个)	QPS	99.99%延迟 (ms)	第一个100%延迟(ms)	最后一个100%延迟 (ms)
64G	X86	1000	53496 0.92	24	34	209
		10000	51136 2.67	108	171	315

表 12-4 GET 操作命令测试结果

实例规格	实例 CPU 类型	并发连接数 (个)	QPS	99.99%延迟 (ms)	第一个100%延迟(ms)	最后一个100%延迟 (ms)
64G	X86	1000	58466 9.15	23	31	82
		10000	53317 8.04	144	184	370

说明

Redis 3.0实例暂不支持ARM CPU类型实例，仅提供X86类型的实例测试结果。

12.4.3 Redis 4.0/5.0 主备实例测试数据

测试环境说明

- 测试实例规格
Redis 4.0/5.0 8G主备
Redis 4.0/5.0 32G主备
- 测试执行机规格
通用计算增强型 | c6.2xlarge.2 | 8vCPUs | 16GB
- 测试执行机镜像
Ubuntu 18.04 server 64bit
- 测试工具
使用单台ECS测试，测试工具为redis-benchmark

测试命令

```
redis-benchmark -h {IP} -p {Port} -a {password} -n {nreqs} -r {randomkeys} -c {connection} -d {datasize} -t {command}
```

参数参考值：-c {connect_number}: 500, -n {nreqs}: 10000000, -r {randomkeys}: 1000000, -d {datasize}: 32, -t {command}: set。

测试结果

📖 说明

- 以下测试结果仅供参考，不同局点环境和网络波动等客观条件可能产生性能差异。
- QPS：即Query Per Second，表示每秒处理的读写操作数，单位是次/秒。
- 平均/最大时延：操作的平均/最大延迟时间，单位为毫秒（ms）。
- x%延迟：指x%操作的延迟时间，单位为毫秒（ms）。例如该指标的值为10ms，99.99%延迟表示99.99%的请求可以在10ms内被处理。

表 12-5 SET 操作命令测试结果

实例规格	实例CPU类型	并发连接数(个)	QPS	99.99%延迟(ms)	第一个100%延迟(ms)	最后一个100%延迟(ms)	平均时延(ms)
8G	X86	500	13206 8.98	11	18	205	3.298
		10000	82386 .58	171	178	263	69.275
8G	ARM	500	94811 .89	10	12	13	3.476
		10000	61264 .37	340	350	351	83.848
32G	X86	500	13138 5.33	9.5	16	17	3.333

实例规格	实例 CPU 类型	并发连接数 (个)	QPS	99.99% 延迟 (ms)	第一个 100%延迟 (ms)	最后一个 100%延迟 (ms)	平均时延 (ms)
		10000	82275 .41	157	162.18	162.43	62.105
32G	ARM	500	11755 3.02	8	21	22	3.875
		10000	76001 .7	175	386	387	99.362

表 12-6 GET 操作命令测试结果

实例规格	实例 CPU 类型	并发连接数 (个)	QPS	99.99% 延迟 (ms)	第一个 100%延迟 (ms)	最后一个 100%延迟 (ms)	平均时延 (ms)
8G	X86	500	13865 2.02	7	11	12	2.117
		10000	82710 .94	123.7	281.6	282.9	61.078
8G	ARM	500	95432 .59	8.8	10	214	3.186
		10000	60984 .16	217	337.15	337.92	83.321
32G	X86	500	13911 3.02	6.6	10	11	2.119
		10000	82489 .36	100	105.66	106	60.968
32G	ARM	500	13904 1.45	6	10	11	2.487
		10000	81563 .41	141	149	150	63

12.4.4 Redis 4.0/5.0 Proxy 实例测试数据

测试环境说明

- 测试实例规格
Redis 4.0/5.0 64G (8分片) Proxy集群
- 测试执行机规格
通用计算增强型 | c6.xlarge.2 | 4vCPUs | 8GB

- 测试工具
使用三台ECS并发测试，测试工具为memtier_benchmark

测试命令

```
memtier_benchmark --ratio=( 1:0 and 0:1 ) -s {IP} -n {nreqs} -c {connect_number} -t 4 -d {datasize}
```

参数参考值：-c {connect_number}: 1000, -n {nreqs}: 10000000, -d {datasize}: 32。

测试结果

说明

- 以下测试结果仅供参考，不同局点环境和网络波动等客观条件可能产生性能差异。
- QPS：即Query Per Second，表示每秒处理的读写操作数，单位是次/秒。
- 平均/最大时延：操作的平均/最大延迟时间，单位为毫秒（ms）。
- x%延迟：指x%操作的延迟时间，单位为毫秒（ms）。例如该指标的值为10ms，99.99%延迟表示99.99%的请求可以在10ms内被处理。

表 12-7 SET 操作命令测试结果

实例规格	实例CPU类型	并发连接数 (个)	QPS	95%左右延迟 (ms)	99.99%延迟 (ms)	最大时延 (ms)
64G	X86	3000	1,323,935.00	3.3	9.4	220
		5000	1,373,756.00	5.3	13	240
		10000	1,332,074.00	11	26	230
		80000	946,032.00	110	460	6800
64G	ARM	3000	837,864.92	5.8	16	78
		5000	763,609.69	10	29	240
		10000	703,808.39	20	47	250
		80000	625,841.69	170	410	940

表 12-8 GET 操作命令测试结果

实例规格	实例 CPU 类型	并发连接数 (个)	QPS	95%左右延迟 (ms)	99.99%延迟 (ms)	最大时延 (ms)
64G	X86	3000	1,366,153.00	3.3	9.3	230
		5000	1,458,451.00	5.1	13	220
		10000	1,376,399.00	11	29	440
		80000	953,837.00	120	1300	2200
64G	ARM	3000	764,114.55	6.1	17	100
		5000	765,187.74	10	27	230
		10000	731,310.95	20	47	250
		80000	631,373.33	170	1300	1900

12.4.5 Redis 4.0/5.0 Cluster 集群实例测试数据

测试环境说明

- 测试实例规格
Redis 4.0/5.0 32G Cluster 集群
- 测试执行机规格
通用计算增强型 | c6.xlarge.2 | 4vCPUs | 8GB
- 测试工具
使用三台ECS并发测试，测试工具为memtier_benchmark

测试命令

```
memtier_benchmark --cluster-mode --ratio=( 1:0 and 0:1 ) -s {IP} -n {nreqs} -c {connect_number} -t 4 -d {datasize}
```

参数参考值：-c {connect_number}: 1000, -n {nreqs}: 10000000, -d {datasize}: 32。

测试结果

说明

- 以下测试结果仅供参考，不同局点环境和网络波动等客观条件可能产生性能差异。
- QPS：即Query Per Second，表示每秒处理的读写操作数，单位是次/秒。
- 平均/最大时延：操作的平均/最大延迟时间，单位为毫秒（ms）。
- x%延迟：指x%操作的延迟时间，单位为毫秒（ms）。例如该指标的值为10ms，99.99%延迟表示99.99%的请求可以在10ms内被处理。

表 12-9 SET 操作命令测试结果

实例规格	实例CPU类型	并发连接数 (个)	QPS	99.99%延迟 (ms)	第一个100%延迟 (ms)	最后一个100%延迟 (ms)
32G	X86	1000	37178 0.2	5.6	6.3	44
		10000	25607 3.11	90	220	460
32G	ARM	1000	31705 3.78	17	34	230
		10000	24883 2.33	410	490	750

表 12-10 GET 操作命令测试结果

实例规格	实例CPU类型	并发连接数 (个)	QPS	99.99%延迟 (ms)	第一个100%延迟 (ms)	最后一个100%延迟 (ms)
32G	X86	1000	42700 0.04	5.0	5.3	78
		10000	30215 9.03	63	220	460
32G	ARM	1000	42140 2.06	13	14	65
		10000	30935 9.18	180	260	500

12.4.6 Redis 6.0 主备实例测试数据

Redis 6.0基础版实例支持开启SSL，本章节包含开启SSL前后的Redis实例性能测试数据。

测试环境说明

- 测试实例规格
Redis 6.0 基础版 8G主备
Redis 6.0 基础版 32G主备
- 测试执行机规格
通用计算增强型 | 8vCPUs | 16GiB | c7.2xlarge.2
- 测试执行机镜像
Ubuntu 18.04 server 64bit
- 测试工具
使用单台ECS测试，测试工具为mementier_benchmark

测试命令

未开启SSL场景：

```
./mementier_benchmark -s {IP} -p {port} -c {connect_number} -t {thread} -n allkeys --key-prefix="xxxx" --key-minimum=1 --key-maximum={max_key} --key-pattern=P:P --ratio=1:0 -d {datasize}
```

参数参考值：-c {connect_number}：1000，--key-maximum{max_key}：2000000，-d {datasize}：32。

开启SSL场景：

```
./mementier_benchmark -s {IP} -p {port} -c {connect_number} -t {thread} -n allkeys --key-prefix="xxxx" --key-minimum=1 --key-maximum={max_key} --key-pattern=P:P --ratio=1:0 -d {datasize} --tls --cacert ca.crt
```

参数参考值：-c {connect_number}：1000，--key-maximum{max_key}：2000000，-d {datasize}：32。

测试结果

📖 说明

- 以下测试结果仅供参考，不同局点环境和网络波动等客观条件可能产生性能差异。
- QPS：即Query Per Second，表示每秒处理的读写操作数，单位是次/秒。
- 平均/最大时延：操作的平均/最大延迟时间，单位为毫秒（ms）。
- x%延迟：指x%操作的延迟时间，单位为毫秒（ms）。例如该指标的值为10ms，99.99%延迟表示99.99%的请求可以在10ms内被处理。

表 12-11 SET 操作命令测试结果（未开启 SSL 场景）

实例规格	实例 CPU 类型	并发连接数 (个)	QPS	平均时延 (ms)	99%延迟 (ms)	99.9%延迟 (ms)
8G	X86	500	15104 7.41	3.355	6.175	12.223
		1000	14934 6.86	6.673	11.711	31.743
32G	X86	500	14364 8.1	3.476	5.215	13.055

实例规格	实例 CPU 类型	并发连接数 (个)	QPS	平均时延 (ms)	99%延迟 (ms)	99.9%延迟 (ms)
		4000	10451 7.03	37.881	139.263	175.103

表 12-12 SET 操作命令测试结果 (开启 SSL 场景)

实例规格	实例 CPU 类型	并发连接数 (个)	QPS	平均时延 (ms)	99%延迟 (ms)	99.9%延迟 (ms)
8G	X86	500	86827. 84	5.537	8.575	9.535
		1000	92413. 99	10.055	15.615	17.279
32G	X86	500	87385. 5	5.584	8.383	9.343
		4000	50813. 67	62.623	100.863	104.959

表 12-13 GET 操作命令测试结果 (未开启 SSL 场景)

实例规格	实例 CPU 类型	并发连接数 (个)	QPS	平均时延 (ms)	99%延迟 (ms)	99.9%延迟 (ms)
8G	X86	500	18041 3.66	2.764	4.287	11.583
		1000	17911 3.5	5.586	8.959	29.823
32G	X86	500	17526 8.86	2.848	4.079	11.839
		4000	13475 5.17	29.161	126.463	166.911

表 12-14 GET 操作命令测试结果（开启 SSL 场景）

实例规格	实例 CPU 类型	并发连接数 (个)	QPS	平均时延 (ms)	99%延迟 (ms)	99.9%延迟 (ms)
8G	X86	500	11363 7.22	4.316	6.239	7.359
		1000	10550 4.55	8.962	13.439	15.295
32G	X86	500	10030 9.99	4.603	6.559	6.943
		4000	57007. 69	55.052	85.503	89.087

12.4.7 Redis 6.0 Cluster 集群实例测试数据

Redis 6.0基础版实例支持开启SSL，本章节包含开启SSL前后的Redis实例性能测试数据。

测试环境说明

- 测试实例规格
Redis 6.0 基础版 32G Cluster集群
- 测试执行机规格
通用计算增强型 | 8vCPUs | 16GiB | c7.2xlarge.2
- 测试执行机镜像
Ubuntu 18.04 server 64bit
- 测试工具
使用三台ECS并发测试，测试工具为memtier_benchmark

测试命令

未开启SSL场景：

```
./memtier_benchmark -s {IP} -p {port} -c {connect_number} -t {thread} -n allkeys --key-prefix="xxxx" --key-minimum=1 --key-maximum={max_key} --key-pattern=P:P --ratio=1:0 -d {datasize} --cluster-mode
```

参数参考值：-c {connect_number}: 1000, --key-maximum{max_key}: 2000000, -d {datasize}: 32。

开启SSL场景：

```
./memtier_benchmark -s {IP} -p {port} -c {connect_number} -t {thread} -n allkeys --key-prefix="xxxx" --key-minimum=1 --key-maximum={max_key} --key-pattern=P:P --ratio=1:0 -d {datasize} --cluster-mode --tls --cacert ca.crt
```

参数参考值：-c {connect_number}: 1000, --key-maximum{max_key}: 2000000, -d {datasize}: 32。

测试结果

说明

- 以下测试结果仅供参考，不同局点环境和网络波动等客观条件可能产生性能差异。
- QPS：即Query Per Second，表示每秒处理的读写操作数，单位是次/秒。
- 平均/最大时延：操作的平均/最大延迟时间，单位为毫秒（ms）。
- x%延迟：指x%操作的延迟时间，单位为毫秒（ms）。例如该指标的值为10ms，99.99%延迟表示99.99%的请求可以在10ms内被处理。

表 12-15 SET 操作命令测试结果（未开启 SSL 场景）

实例规格	实例 CPU 类型	并发连接数 (个)	QPS	平均时延 (ms)	99%延迟 (ms)	99.9%延迟 (ms)
32G	X86	1000	32289 9.21	2.661	4.319	8.511
		3000	36033 6.14	7.757	13.055	29.439
		10000	33037 8.22	29.411	97.279	153.599

表 12-16 SET 操作命令测试结果（开启 SSL 场景）

实例规格	实例 CPU 类型	并发连接数 (个)	QPS	平均时延 (ms)	99%延迟 (ms)	99.9%延迟 (ms)
32G	X86	1000	23830 7.26	3.603	5.151	6.527
		3000	18545 5.62	13.196	20.607	352.255
		10000	11191 3.19	57.537	96.767	121.343

表 12-17 GET 操作命令测试结果（未开启 SSL 场景）

实例规格	实例 CPU 类型	并发连接数 (个)	QPS	平均时延 (ms)	99%延迟 (ms)	99.9%延迟 (ms)
32G	X86	1000	45042 2.66	1.875	2.767	6.879
		3000	43245 0.2	6.451	12.095	28.415

实例规格	实例 CPU 类型	并发连接数 (个)	QPS	平均时延 (ms)	99%延迟 (ms)	99.9%延迟 (ms)
		10000	50733 8.44	23.001	95.231	176.127

表 12-18 GET 操作命令测试结果 (开启 SSL 场景)

实例规格	实例 CPU 类型	并发连接数 (个)	QPS	平均时延 (ms)	99%延迟 (ms)	99.9%延迟 (ms)
32G	X86	1000	27406 6.16	3.076	4.255	7.071
		3000	20106 3.51	11.743	18.047	387.071
		10000	11602 6.38	51.284	84.479	136.191

12.4.8 Redis 备份恢复迁移性能测试数据

测试环境说明

- 测试实例规格：
 - Redis 5.0 8G主备
 - Redis 5.0 32G主备
 - Redis 5.0 64G Proxy集群 (副本数: 2 | 分片数: 8 | 分片容量: 8 GB)
 - Redis 5.0 256G Proxy集群 (副本数: 2 | 分片数: 32 | 分片容量: 8 GB)
 - Redis 5.0 64G Cluster集群 (副本数: 2 | 分片数: 8 | 分片容量: 8 GB)
 - Redis 5.0 256G Cluster集群 (副本数: 2 | 分片数: 32 | 分片容量: 8 GB)
- 测试执行机规格：
 - c6s.large.2 2vCPUs | 4GB

测试命令

256G proxy实例测试命令：

```
redis-benchmark -h {IP} -p{Port} -n 10000000 -r 10000000 -c 10000 -d 1024
```

256G cluster实例测试命令：

```
redis-benchmark -h {IP} -p{Port} -n 10000000 -r 10000000 -c 40000 -d 1024 -c
```

测试结果

表 12-19 迁移

源实例类型	源实例规格 (GB)	目标实例类型	目标实例规格 (GB)	迁移方式	数据量 (GB)	时间 (min)
Redis 5.0 主备	8	Redis 5.0 主备	8	全量迁移+增量迁移	7.78	3
Redis 5.0 主备	32	Redis 5.0 主备	32	全量迁移+增量迁移	31.9	17
Redis 5.0 proxy	64	Redis 5.0proxy	64	全量迁移+增量迁移	62.42	7
Redis 5.0 cluster	64	Redis 5.0cluster	64	全量迁移+增量迁移	57.69	6
Redis 5.0 proxy	256	Redis 5.0proxy	256	全量迁移+增量迁移	241.48	23
Redis 5.0 cluster	256	Redis 5.0cluster	256	全量迁移+增量迁移	240.21	22

表 12-20 备份

实例类型	实例规格 (GB)	备份方式	数据量 (GB)	时间 (min)
Redis 5.0主备	8	rdb	7.78	2
Redis 5.0主备	32	rdb	31.9	5
Redis 5.0 proxy	64	rdb	62.42	9
Redis 5.0 proxy	256	rdb	241.48	37
Redis 5.0 cluster	64	rdb	57.69	9
Redis 5.0 cluster	256	rdb	255	39
Redis 5.0主备	8	aof	7.9	2

实例类型	实例规格 (GB)	备份方式	数据量 (GB)	时间 (min)
Redis 5.0主备	32	aof	31.15	10
Redis 5.0 proxy	64	aof	62.42	20
Redis 5.0 proxy	256	aof	241.48	48
Redis 5.0 cluster	64	aof	57.69	19
Redis 5.0 cluster	256	aof	255	51

表 12-21 恢复

实例类型	实例规格 (GB)	恢复方式	数据量 (GB)	时间 (min)
Redis 5.0主备	8	rdb	7.9	2
Redis 5.0主备	32	rdb	31.15	6
Redis 5.0 proxy	64	rdb	62.42	10
Redis 5.0 proxy	256	rdb	246	42
Redis 5.0 cluster	64	rdb	57.69	10
Redis 5.0 cluster	256	rdb	255	40
Redis 5.0主备	8	aof	7.9	3
Redis 5.0主备	32	aof	31.15	10
Redis 5.0 proxy	64	aof	62.42	10
Redis 5.0 proxy	256	aof	246	46
Redis 5.0 cluster	64	aof	57.69	10
Redis 5.0 cluster	256	aof	255	43

13 申请扩大 DCS 配额

什么是配额？

为防止资源滥用，平台限制了各服务资源的配额，对用户的资源数量和容量做了限制。如您最多可以创建多少个DCS缓存实例，可使用多少内存等。

如果当前资源配额限制无法满足使用需要，您可以申请扩大配额。

怎样查看我的配额？

1. 登录管理控制台。
2. 在管理控制台左上角单击 ，选择实例所在的区域。
3. 在页面右上角，选择“资源 > 我的配额”。
系统进入“服务配额”页面。

图 13-1 我的配额



4. 您可以在“服务配额”页面，查看各项资源的总配额及使用情况。
如果当前配额不能满足业务要求，请参考后续操作，申请扩大配额。

如何申请扩大配额？

1. 登录管理控制台。
2. 在页面右上角，选择“资源 > 我的配额”。
系统进入“服务配额”页面。

图 13-2 我的配额



3. 单击“申请扩大配额”。
4. 在“新建工单”页面，根据您的需求，填写相关参数。
其中，“问题描述”项请填写需要调整的内容和申请原因。
5. 填写完毕后，勾选协议并单击“提交”。

14 查看监控指标与配置告警

云监控服务（CloudEye Service）是公有云提供的安全、可扩展的统一监控方案，通过云监控服务集中监控DCS的各种指标，基于云监控服务实现告警和事件通知。

14.1 DCS 支持的监控指标

功能说明

本节定义了DCS服务上报云监控服务的监控指标的命名空间，监控指标列表和维度定义，用户可以通过云监控服务提供管理控制台或[API接口](#)来检索DCS服务产生的监控指标和告警信息。

实例监控指标差异如下：

表 14-1 实例监控指标差异

实例类型	实例级监控	数据节点级监控	Proxy节点级监控
单机	支持 只有实例级别的监控指标，实例监控即为数据节点监控。	不涉及	不涉及
主备	支持 实例监控是指对主节点的监控。	支持 数据节点监控分别是对主节点和备节点的监控。	不涉及
读写分离	支持 实例监控是指对主节点的监控。	支持 数据节点监控分别是对主节点和备节点的监控。	支持 Proxy节点监控是对实例中每个Proxy节点的监控。

实例类型	实例级监控	数据节点级监控	Proxy节点级监控
Proxy集群	支持 实例监控是对集群所有主节点数据汇总后的监控。	支持 数据节点监控是对集群每个分片的监控。	支持 Proxy节点监控是对集群每个Proxy节点的监控。
Cluster集群	支持 实例监控是对集群所有主节点数据汇总后的监控。	支持 数据节点监控是对集群每个分片的监控。	不涉及

命名空间

SYS.DCS

Redis 3.0 实例监控指标

📖 说明

- DCS Redis 3.0已下线，暂停售卖，建议使用Redis 4.0及以上版本。
- 监控指标的维度请参考[维度](#)。

表 14-2 Redis 3.0 实例支持的监控指标

指标ID	指标名称	指标含义	取值范围	测量对象	监控周期（原始指标）
cpu_usage	CPU利用率	该指标对于统计周期内的测量对象的CPU使用率进行多次采样，表示多次采样的最高值。 单位：%。 如果是单机/主备实例，该指标为主节点的CPU值。 如果是Proxy集群实例，该指标为各个Proxy节点的平均值。	0-100%	Redis实例（单机/主备/集群）	1分钟

指标ID	指标名称	指标含义	取值范围	测量对象	监控周期(原始指标)
memory_usage	内存利用率	该指标用于统计测量对象的内存利用率。 单位: %。 须知 内存利用率统计是扣除预留内存的。	0-100 %	Redis实例(单机/主备/集群)	1分钟
net_in_throughput	网络输入吞吐量	该指标用于统计网口平均每秒的输入流量。 单位: byte/s。	>=0字节/秒	Redis实例(单机/主备/集群)	1分钟
net_out_throughput	网络输出吞吐量	该指标用于统计网口平均每秒的输出流量。 单位: byte/s。	>=0字节/秒	Redis实例(单机/主备/集群)	1分钟
connected_clients	活跃的客户数量	该指标用于统计已连接的客户端数量,包括系统监控、配置同步和业务相关的连接数,不包括来自从节点的连接。	>=0	Redis实例(单机/主备/集群)	1分钟
client_longest_output_list	客户端最长输出列表	该指标用于统计客户端所有现存连接的最长输出列表。	>=0	Redis实例(单机/主备/集群)	1分钟
client_biggest_input_buf	客户端最大输入缓冲	该指标用于统计客户端所有现存连接的最大输入数据长度。 单位: byte。	>=0byte	Redis实例(单机/主备/集群)	1分钟
blocked_clients	阻塞的客户端数量	该指标用于被阻塞操作挂起的客户端的数量。阻塞操作如BLPOP, BRPOP, BRPOPLPUSH。	>=0	Redis实例(单机/主备/集群)	1分钟
used_memory	已用内存	该指标用于统计Redis已使用的内存字节数。 单位: byte。	>=0byte	Redis实例(单机/主备/集群)	1分钟

指标ID	指标名称	指标含义	取值范围	测量对象	监控周期（原始指标）
used_memory_rss	已用内存RSS	该指标用于统计Redis已使用的RSS内存。即实际驻留“在内存中”的内存数。包含堆内存，但不包括换出的内存。 单位：byte。	>=0byte	Redis实例（单机/主备/集群）	1分钟
used_memory_peak	已用内存峰值	该指标用于统计Redis服务器启动以来使用内存的峰值。 单位：byte。	>=0byte	Redis实例（单机/主备/集群）	1分钟
used_memory_lua	Lua已用内存	该指标用于统计Lua引擎已使用的内存字节。 单位：byte。	>=0byte	Redis实例（单机/主备/集群）	1分钟
memory_fragment_ratio	内存碎片率	该指标用于统计当前的内存碎片率。其数值上等于used_memory_rss / used_memory。	>=0	Redis实例（单机/主备/集群）	1分钟
total_connections_received	新建连接数	该指标用于统计周期内新建的连接数。	>=0	Redis实例（单机/主备/集群）	1分钟
total_commands_processed	处理的命令数	该指标用于统计周期内处理的命令数。	>=0	Redis实例（单机/主备/集群）	1分钟
instantaneous_ops	每秒并发操作数	该指标用于统计每秒处理的命令数。	>=0	Redis实例（单机/主备/集群）	1分钟
total_net_input_bytes	网络收到字节数	该指标用于统计周期内收到的字节数。 单位：byte。	>=0byte	Redis实例（单机/主备/集群）	1分钟
total_net_output_bytes	网络发送字节数	该指标用于统计周期内发送的字节数。 单位：byte。	>=0byte	Redis实例（单机/主备/集群）	1分钟

指标ID	指标名称	指标含义	取值范围	测量对象	监控周期(原始指标)
instantaneous_input_kbps	网络瞬时输入流量	该指标用于统计瞬时的输入流量。 单位: KB/s。	>=0KB/s	Redis实例(单机/主备/集群)	1分钟
instantaneous_output_kbps	网络瞬时输出流量	该指标用于统计瞬时的输出流量。 单位: KB/s。	>=0KB/s	Redis实例(单机/主备/集群)	1分钟
rejected_connections	已拒绝的连接数	该指标用于统计周期内因为超过maxclients而拒绝的连接数量。	>=0	Redis实例(单机/主备/集群)	1分钟
expired_keys	已过期的键数量	该指标用于统计周期内因过期而被删除的键数量	>=0	Redis实例(单机/主备/集群)	1分钟
evicted_keys	已逐出的键数量	该指标用于统计周期内因为内存不足被删除的键数量。	>=0	Redis实例(单机/主备/集群)	1分钟
keyspace_hits	Keyspace命中次数	该指标用于统计周期内在主字典中查找命中次数。	>=0	Redis实例(单机/主备/集群)	1分钟
keyspace_misses	Keyspace错过次数	该指标用于统计周期内在主字典中查找不命中次数。	>=0	Redis实例(单机/主备/集群)	1分钟
pubsub_channels	Pubsub通道个数	该指标用于统计Pub/Sub通道个数。	>=0	Redis实例(单机/主备/集群)	1分钟
pubsub_patterns	Pubsub模式个数	该指标用于统计Pub/Sub模式个数。	>=0	Redis实例(单机/主备/集群)	1分钟
keyspace_hits_perc	缓存命中率	该指标用于统计Redis的缓存命中率, 其命中率算法为: keyspace_hits/(keyspace_hits+keyspace_misses) 单位: %。	0-100%	Redis实例(单机/主备/集群)	1分钟

指标ID	指标名称	指标含义	取值范围	测量对象	监控周期（原始指标）
command_max_delay	命令最大时延	统计实例的命令最大时延。 单位：ms。	>=0ms	Redis实例（单机/主备/集群）	1分钟
auth_errors	认证失败次数	统计实例的认证失败次数。	>=0	Redis实例（单机/主备）	1分钟
is_slow_log_exist	是否存在慢日志	统计实例是否存在慢日志。 说明 该监控不统计由 migrate、slaveof、config、bgsave、bgrewriteof命令导致的慢日志。	<ul style="list-style-type: none"> 1：表示存在 0：表示不存在。 	Redis实例（单机/主备）	1分钟
keys	缓存键总数	该指标用于统计Redis缓存中键总数。	>=0	Redis实例（单机/主备）	1分钟

Redis 4.0、Redis 5.0 和 Redis 6.0 实例监控指标

📖 说明

- 监控指标的维度请参考[维度](#)。
- 实例监控是对主节点数据汇总后的监控。
- 部分实例监控指标，是对从主节点和从节点汇聚的指标，请参考[表14-3](#)中“指标含义”的说明。

表 14-3 Redis 4.0、Redis 5.0 和 Redis 6.0 实例支持的监控指标

指标ID	指标名称	指标含义	取值范围	测量对象	监控周期（原始指标）
cpu_usage	CPU利用率	该指标对于统计周期内的测量对象的CPU使用率进行多次采样，表示多次采样的最高值。 单位：%。	0-100%	Redis实例（单机/主备/读写分离）	1分钟

指标ID	指标名称	指标含义	取值范围	测量对象	监控周期(原始指标)
cpu_avg_usage	CPU平均使用率	该指标对于统计周期内的测量对象的CPU使用率进行多次采样,表示多次采样的平均值。 单位: %。	0-100 %	Redis实例(单机/主备/读写分离)	1分钟
command_max_delay	命令最大时延	统计实例的命令最大时延。 单位为ms。	>=0ms	Redis实例	1分钟
total_connections_received	新建连接数	该指标用于统计周期内新建的连接数,包括从节点,系统监控、配置同步和业务相关的连接数。	>=0	Redis实例	1分钟
is_slow_log_exist	是否存在慢日志	统计实例是否存在慢日志。 说明 该监控不统计由migrate、slaveof、config、bgsave、bgrewriteaof命令导致的慢日志。	<ul style="list-style-type: none"> 1: 表示存在 0: 表示不存在。 	Redis实例	1分钟
memory_usage	内存利用率	该指标用于统计测量对象的内存利用率。 单位: %。	0-100 %	Redis实例	1分钟
expires	有过期时间的键总数	该指标用于统计Redis缓存中将会过期失效的键数目。	>=0	Redis实例	1分钟

指标ID	指标名称	指标含义	取值范围	测量对象	监控周期(原始指标)
keyspace_hits_perc	缓存命中率	该指标用于统计Redis的缓存命中率，其命中率算法为： keyspace_hits/(keyspace_hits+keyspace_misses) 从主节点和从节点汇聚的指标。如果单个统计周期内没有读命令的情况下，缓存命令率为0。 单位：%。	0-100%	Redis实例	1分钟
used_memory	已用内存	该指标用于统计Redis已使用的内存字节数。 单位：可在控制台进行选择，如KB、MB、byte等。	>=0	Redis实例	1分钟
used_memory_dataset	数据集使用内存	该指标用于统计Redis中数据集使用的内存。 单位：可在控制台进行选择，如KB、MB、byte等。	>=0	Redis实例	1分钟
used_memory_dataset_perc	数据集使用内存百分比	该指标用于统计Redis中数据集使用的内存所占总内存百分比。 从主节点和从节点汇聚的指标。 单位：%。	0-100%	Redis实例	1分钟

指标ID	指标名称	指标含义	取值范围	测量对象	监控周期（原始指标）
used_memory_rss	已用内存RSS	该指标用于统计Redis已使用的RSS内存。即实际驻留“在内存中”的内存数。包含堆内存，但不包括换出的内存。 单位：可在控制台进行选择，如KB、MB、byte等。	>=0	Redis实例	1分钟
instantaneous_ops	每秒并发操作数	该指标用于统计每秒处理的命令数。	>=0	Redis实例	1分钟
keyspace_misses	Keyspace错过次数	该指标用于统计周期内在主字典中查找不命中次数。 从主节点和从节点汇聚的指标。	>=0	Redis实例	1分钟
keys	缓存键总数	该指标用于统计Redis缓存中键总数。	>=0	Redis实例	1分钟
blocked_clients	阻塞的客户端数量	该指标用于被阻塞操作挂起的客户端的数量。	>=0	Redis实例	1分钟
connected_clients	活跃的客户端数量	该指标用于统计已连接的客户端数量，包括系统监控、配置同步和业务相关的连接数，不包括来自从节点连接。	>=0	Redis实例	1分钟
del	DEL	该指标用于统计平均每秒del操作数。 单位：Count/s	0-5000 00 Count/ s	Redis实例	1分钟
evicted_keys	已逐出的键数量	该指标用于统计周期内因为内存不足被删除的键数量。 从主节点和从节点汇聚的命令统计指标。	>=0	Redis实例	1分钟

指标ID	指标名称	指标含义	取值范围	测量对象	监控周期（原始指标）
expire	EXPIRE	该指标用于统计平均每秒expire操作数。 单位：Count/s	0-5000 00 Count/ s	Redis实例	1分钟
expired_keys	已过期的键数量	该指标用于统计周期内因过期而被删除的键数量。 从主节点和从节点汇聚的命令统计指标。	>=0	Redis实例	1分钟
get	GET	该指标用于统计平均每秒get操作数。 从主节点和从节点汇聚的命令统计指标。 单位：Count/s	0-5000 00 Count/ s	Redis实例	1分钟
hdel	HDEL	该指标用于统计平均每秒hdel操作数。 单位：Count/s	0-5000 00 Count/ s	Redis实例	1分钟
hget	HGET	该指标用于统计平均每秒hget操作数。 从主节点和从节点汇聚的命令统计指标。 单位：Count/s	0-5000 00 Count/ s	Redis实例	1分钟
hmget	HMGET	该指标用于统计平均每秒hmget操作数。 从主节点和从节点汇聚的命令统计指标。 单位：Count/s	0-5000 00 Count/ s	Redis实例	1分钟
hmset	HMSET	该指标用于统计平均每秒hmset操作数。 单位：Count/s	0-5000 00 Count/ s	Redis实例	1分钟

指标ID	指标名称	指标含义	取值范围	测量对象	监控周期（原始指标）
hset	HSET	该指标用于统计平均每秒hset操作数。 单位：Count/s	0-500000 Count/s	Redis实例	1分钟
instantaneous_input_kbps	网络瞬时输入流量	该指标用于统计瞬时的输入流量。 单位：KB/s。	>=0KB/s	Redis实例	1分钟
instantaneous_output_kbps	网络瞬时输出流量	该指标用于统计瞬时的输出流量。 单位：KB/s。	>=0KB/s	Redis实例	1分钟
memory_fragment_ratio	内存碎片率	该指标用于统计当前的内存碎片率。	>=0	Redis实例	1分钟
mget	MGET	该指标用于统计平均每秒mget操作数。 从主节点和从节点汇聚的命令统计指标。 单位：Count/s	0-500000 Count/s	Redis实例	1分钟
mset	MSET	该指标用于统计平均每秒mset操作数。 单位：Count/s	0-500000 Count/s	Redis实例	1分钟
pubsub_channels	Pubsub通道个数	该指标用于统计Pub/Sub通道个数。	>=0	Redis实例	1分钟
pubsub_patterns	Pubsub模式个数	该指标用于统计Pub/Sub模式个数。	>=0	Redis实例	1分钟
set	SET	该指标用于统计平均每秒set操作数。 单位：Count/s	0-500000 Count/s	Redis实例	1分钟
used_memory_lua	Lua已用内存	该指标用于统计Lua引擎已使用的内存字节。 单位：可在控制台进行选择，如KB、MB、Byte等。	>=0	Redis实例	1分钟

指标ID	指标名称	指标含义	取值范围	测量对象	监控周期（原始指标）
used_memory_peak	已用内存峰值	该指标用于统计Redis服务器启动以来使用内存的峰值。 单位：可在控制台进行选择，如KB、MB、Byte等。	>=0	Redis实例	1分钟
sadd	Sadd	该指标用于统计平均每秒sadd操作数。 单位：Count/s	0-5000 00 Count/s	Redis实例	1分钟
smembers	Smembers	该指标用于统计平均每秒smembers操作数。 从主节点和从节点汇聚的命令统计指标。 单位：Count/s	0-5000 00 Count/s	Redis实例	1分钟
scan	SCAN	每秒scan操作次数。 单位：Count/s	0-5000 00 Count/s	Redis实例	1分钟
setex	SETEX	每秒setex操作数。 单位：Count/s	0-5000 00 Count/s	Redis实例	1分钟
rx_controlled	流控次数	该指标用于统计周期内被流控的次数。 这个指标值大于0时，表示当前被使用带宽超过最大带宽限制被流控。 单位：Count。	>=0	Redis实例	1分钟
bandwidth_usage	带宽使用率	计算当前流量带宽与最大带宽限制的百分比。	0-200 %	Redis实例	1分钟

指标ID	指标名称	指标含义	取值范围	测量对象	监控周期（原始指标）
command_max_rt	最大时延	节点从接收命令到发出响应的时延最大值。 单位：us。	>=0us	Redis实例（单机）	1分钟
command_avg_rt	平均时延	节点从接收命令到发出响应的时延平均值。 单位：us。	>=0us	Redis实例（单机）	1分钟
used_storage	已使用的存储空间	已使用的存储空间。 单位：可在控制台进行选择，如KB、MB、Byte等。	>=0	Redis实例（企业版存储型）	1分钟
storage_usage	存储空间使用率	存储空间使用率。 单位：%。	0-100%	Redis实例（企业版存储型）	1分钟

Redis 实例数据节点监控指标

📖 说明

- Redis主备、读写分离、集群实例支持数据节点监控。
- 监控指标的维度请参考[维度](#)。

表 14-4 实例中数据节点支持的监控指标

指标ID	指标名称	指标含义	取值范围	测量对象	监控周期（原始指标）
cpu_usage	CPU利用率	该指标对于统计周期内的测量对象的CPU使用率进行多次采样，表示多次采样的最高值。 单位：%。	0-100%	Redis主备、读写分离、集群实例数据节点	1分钟

指标ID	指标名称	指标含义	取值范围	测量对象	监控周期（原始指标）
cpu_avg_usage	CPU平均使用率	该指标对于统计周期内的测量对象的CPU使用率进行多次采样，表示多次采样的平均值。 单位：%。	0-100%	Redis 4.0、5.0、6.0版本 主备、读写分离、集群实例数据节点	1分钟
memory_usage	内存利用率	该指标用于统计测量对象的内存利用率。 单位：%。	0-100%	Redis主备、读写分离、集群实例数据节点	1分钟
connected_clients	活跃的客户端数量	该指标用于统计已连接的客户端数量，包括系统监控、配置同步和业务相关的连接数，不包括来自从节点的连接。	>=0	Redis主备、读写分离、集群实例数据节点	1分钟
client_longest_out_list	客户端最长输出列表	该指标用于统计客户端所有现存连接的最长输出列表。	>=0	Redis 3.0、4.0版本 主备、读写分离、集群实例数据节点	1分钟
client_biggest_in_buf	客户端最大输入缓冲	该指标用于统计客户端所有现存连接的最大输入数据长度。 单位：可在控制台进行选择，如KB、MB、byte等。	>=0	Redis 3.0、4.0版本 主备、读写分离、集群实例数据节点	1分钟
blocked_clients	阻塞的客户端数量	该指标用于被阻塞操作挂起的客户端的数量。阻塞操作如BLPOP，BRPOP，BRPOPLPUSH。	>=0	Redis主备、读写分离、集群实例数据节点	1分钟
used_memory	已用内存	该指标用于统计Redis已使用的内存字节数。 单位：可在控制台进行选择，如KB、MB、byte等。	>=0	Redis主备、读写分离、集群实例数据节点	1分钟

指标ID	指标名称	指标含义	取值范围	测量对象	监控周期（原始指标）
used_memory_rss	已用内存RSS	该指标用于统计Redis已使用的RSS内存。即实际驻留“在内存中”的内存数，包含堆内存，但不包括换出的内存。 单位：可在控制台进行选择，如KB、MB、byte等。	>=0	Redis主备、读写分离、集群实例数据节点	1分钟
used_memory_peak	已用内存峰值	该指标用于统计Redis服务器启动以来使用内存的峰值。 单位：可在控制台进行选择，如KB、MB、byte等。	>=0	Redis主备、读写分离、集群实例数据节点	1分钟
used_memory_lua	Lua已用内存	该指标用于统计Lua引擎已使用的内存字节。 单位：可在控制台进行选择，如KB、MB、byte等。	>=0	Redis主备、读写分离、集群实例数据节点	1分钟
memory_fragment_ratio	内存碎片率	该指标用于统计当前的内存碎片率。其数值上等于used_memory_rss / used_memory。	>=0	Redis主备、读写分离、集群实例数据节点	1分钟
total_connections_received	新建连接数	该指标用于统计周期内新建的连接数，包括从节点，系统监控、配置同步和业务相关的连接数。	>=0	Redis主备、读写分离、集群实例数据节点	1分钟
total_commands_processed	处理的命令数	该指标用于统计周期内处理的命令数。	>=0	Redis主备、读写分离、集群实例数据节点	1分钟
instantaneous_ops	每秒并发操作数	该指标用于统计每秒处理的命令数。	>=0	Redis主备、读写分离、集群实例数据节点	1分钟

指标ID	指标名称	指标含义	取值范围	测量对象	监控周期（原始指标）
total_net_input_bytes	网络收到字节数	该指标用于统计周期内收到的字节数。 单位：可在控制台进行选择，如KB、MB、byte等。	>=0	Redis主备、读写分离、集群实例数据节点	1分钟
total_net_output_bytes	网络发送字节数	该指标用于统计周期内发送的字节数。 单位：可在控制台进行选择，如KB、MB、byte等。	>=0	Redis主备、读写分离、集群实例数据节点	1分钟
instantaneous_input_kbps	网络瞬时输入流量	该指标用于统计瞬时的输入流量。 单位：KB/s。	>=0KB/s	Redis主备、读写分离、集群实例数据节点	1分钟
instantaneous_output_kbps	网络瞬时输出流量	该指标用于统计瞬时的输出流量。 单位：KB/s。	>=0KB/s	Redis主备、读写分离、集群实例数据节点	1分钟
rejected_connections	已拒绝的连接数	该指标用于统计周期内因为超过maxclients而拒绝的连接数量。	>=0	Redis主备、读写分离、集群实例数据节点	1分钟
expired_keys	已过期的键数量	该指标用于统计周期内因过期而被删除的键数量。	>=0	Redis主备、读写分离、集群实例数据节点	1分钟
evicted_keys	已逐出的键数量	该指标用于统计周期内因为内存不足被删除的键数量。	>=0	Redis主备、读写分离、集群实例数据节点	1分钟
pubsub_channels	Pubsub通道个数	该指标用于统计Pub/Sub通道个数。	>=0	Redis主备、读写分离、集群实例数据节点	1分钟
pubsub_patterns	Pubsub模式个数	该指标用于统计Pub/Sub模式个数。	>=0	Redis主备、读写分离、集群实例数据节点	1分钟

指标ID	指标名称	指标含义	取值范围	测量对象	监控周期（原始指标）
keyspace_hits_perc	缓存命中率	该指标用于统计Redis的缓存命中率，其命中率算法为： keyspace_hits / (keyspace_hits + keyspace_misses) 如果单个统计周期内没有读命令的情况下，缓存命令率为0。 单位：%。	0-100%	Redis主备、读写分离、集群实例数据节点	1分钟
command_max_delay	命令最大时延	统计节点的命令最大时延。 单位：ms。	>=0ms	Redis主备、读写分离、集群实例数据节点	1分钟
is_slow_log_exist	是否存在慢日志	统计节点是否存在慢日志。 说明 该监控不统计由migrate、slaveof、config、bgsave、bgrewriteaof命令导致的慢查询。	<ul style="list-style-type: none"> 1：表示存在 0：表示不存在。 	Redis主备、读写分离、集群实例数据节点	1分钟
keys	缓存键总数	该指标用于统计Redis缓存中键总数。	>=0	Redis主备、读写分离、集群实例数据节点	1分钟
sadd	SADD	该指标用于统计平均每秒sadd操作数。 单位：Count/s	0-500000 Count/s	Redis 4.0、5.0、6.0版本 主备、读写分离、集群实例数据节点	1分钟
smembers	SMEMBERS	该指标用于统计平均每秒smembers操作数。 单位：Count/s	0-500000 Count/s	Redis 4.0、5.0、6.0版本 主备、读写分离、集群实例数据节点	1分钟

指标ID	指标名称	指标含义	取值范围	测量对象	监控周期（原始指标）
ms_repl_offset	主从数据同步差值	该指标用于统计主从节点之间的数据同步差值。	-	Redis 4.0、5.0、6.0版本 主备、读写分离、集群实例数据节点的 备节点	1分钟
del	DEL	该指标用于统计平均每秒del操作数。 单位：Count/s	0-5000 00 Count/s	Redis 4.0、5.0、6.0版本 主备、读写分离、集群实例数据节点	1分钟
expire	EXPIRE	该指标用于统计平均每秒expire操作数。 单位：Count/s	0-5000 00 Count/s	Redis 4.0、5.0、6.0版本 主备、读写分离、集群实例数据节点	1分钟
get	GET	该指标用于统计平均每秒get操作数。 单位：Count/s	0-5000 00 Count/s	Redis 4.0、5.0、6.0版本 主备、读写分离、集群实例数据节点	1分钟
hdel	HDEL	该指标用于统计平均每秒hdel操作数。 单位：Count/s	0-5000 00 Count/s	Redis 4.0、5.0、6.0版本 主备、读写分离、集群实例数据节点	1分钟
hget	HGET	该指标用于统计平均每秒hget操作数。 单位：Count/s	0-5000 00 Count/s	Redis 4.0、5.0、6.0版本 主备、读写分离、集群实例数据节点	1分钟
hmget	HMGET	该指标用于统计平均每秒hmget操作数。 单位：Count/s	0-5000 00 Count/s	Redis 4.0、5.0、6.0版本 主备、读写分离、集群实例数据节点	1分钟
hmset	HMSET	该指标用于统计平均每秒hmset操作数。 单位：Count/s	0-5000 00 Count/s	Redis 4.0、5.0、6.0版本 主备、集群实例数据节点	1分钟

指标ID	指标名称	指标含义	取值范围	测量对象	监控周期（原始指标）
hset	HSET	该指标用于统计平均每秒hset操作数。 单位：Count/s	0-5000 00 Count/s	Redis 4.0、5.0、6.0版本 主备、读写分离、集群实例 数据节点	1分钟
mget	MGET	该指标用于统计平均每秒mget操作数。 单位：Count/s	0-5000 00 Count/s	Redis 4.0、5.0、6.0版本 主备、读写分离、集群实例 数据节点	1分钟
mset	MSET	该指标用于统计平均每秒mset操作数。 单位：Count/s	0-5000 00 Count/s	Redis 4.0、5.0、6.0版本 主备、读写分离、集群实例 数据节点	1分钟
set	SET	该指标用于统计平均每秒set操作数。 单位：Count/s	0-5000 00 Count/s	Redis 4.0、5.0、6.0版本 主备、读写分离、集群实例 数据节点	1分钟
rx_controlled	流控次数	该指标用于统计周期内被流控的次数。 这个指标值大于0时，表示当前被使用带宽超过最大带宽限制被流控。 单位：Count。	>=0	Redis 4.0、5.0、6.0版本 主备、读写分离、集群实例 数据节点	1分钟
bandwidth_usage	带宽使用率	计算当前流量带宽与最大带宽限制的百分比。	0-200%	Redis 4.0、5.0、6.0版本 主备、读写分离、集群实例 数据节点	1分钟
connections_usage	连接数使用率	该指标用于统计当前连接数与最大连接数限制的百分比。 单位：%。	0-100%	Redis 4.0、5.0、6.0版本 主备、读写分离、集群实例 数据节点	1分钟

指标ID	指标名称	指标含义	取值范围	测量对象	监控周期（原始指标）
command_max_rt	最大时延	节点从接收命令到发出响应的时延最大值。 单位：us。	>=0us	Redis 4.0、5.0、6.0版本主备、读写分离、集群实例数据节点	1分钟
command_avg_rt	平均时延	节点从接收命令到发出响应的时延平均值。 单位：us。	>=0us	Redis 4.0、5.0、6.0版本主备、读写分离、集群实例数据节点	1分钟
sync_full	全量同步次数	该指标用于统计Redis服务器启动以来总共完成的全量同步次数。	>=0	Redis 4.0、5.0、6.0版本主备、读写分离、集群实例数据节点	1分钟
slow_log_counts	慢日志出现次数	该指标用于统计在一个统计周期内，慢日志出现的次数。	>=0	Redis 4.0、5.0、6.0版本主备、读写分离、集群实例数据节点	1分钟
scan	SCAN	每秒scan操作数。 单位：Count/s	0-500000 Count/s	Redis 4.0、5.0、6.0版本主备、读写分离、集群实例数据节点	1分钟
setex	SETEX	每秒setex操作数。 单位：Count/s	0-500000 Count/s	Redis 4.0、5.0、6.0版本主备、读写分离、集群实例数据节点	1分钟
used_storage	已使用的存储空间	已使用的存储空间。 单位：可在控制台进行选择，如KB、MB、Byte等。	>=0	Redis 6.0企业版存储型实例数据节点	1分钟
storage_usage	存储空间使用率	存储空间使用率。 单位：%。	0-100%	Redis 6.0企业版存储型实例数据节点	1分钟

Proxy 节点监控指标

说明

- Proxy集群和读写分离实例支持Proxy节点监控指标。
- 监控指标的维度请参考[维度](#)。

表 14-5 Redis 3.0 的 Proxy 集群实例 Proxy 节点支持的监控指标

指标ID	指标名称	指标含义	取值范围	测量对象	监控周期（原始指标）
cpu_usage	CPU利用率	该指标对于统计周期内的测量对象的CPU使用率进行多次采样，表示多次采样的最高值。 单位：%。	0-100%	Redis 3.0 Proxy集群实例Proxy节点	1分钟
memory_usage	内存利用率	该指标用于统计测量对象的内存利用率。 单位：%。	0-100%	Redis 3.0 Proxy集群实例Proxy节点	1分钟
p_connected_clients	活跃的客户数量	该指标用于统计已连接的客户端数量。	>=0	Redis 3.0 Proxy集群实例Proxy节点	1分钟
max_rxpck_per_sec	网卡包接收最大速率	该指标用于统计测量对象网卡在统计周期内每秒接收的最大数据包数。 单位：包/秒	0-1000000包/秒	Redis 3.0 Proxy集群实例Proxy节点	1分钟
max_txpck_per_sec	网卡包发送最大速率	该指标用于统计测量对象网卡在统计周期内每秒发送的最大数据包数。 单位：包/秒	0-1000000包/秒	Redis 3.0 Proxy集群实例Proxy节点	1分钟
max_rxB_per_sec	入网最大带宽	该指标用于统计测量对象网卡每秒接收的最大数据量。 单位：KB/s。	>=0KB/s	Redis 3.0 Proxy集群实例Proxy节点	1分钟
max_txB_per_sec	出网最大带宽	该指标用于统计测量对象网卡每秒发送的最大数据量。 单位：KB/s。	>=0KB/s	Redis 3.0 Proxy集群实例Proxy节点	1分钟

指标ID	指标名称	指标含义	取值范围	测量对象	监控周期（原始指标）
avg_rxpck_per_sec	网卡包接收平均速率	该指标用于统计测量对象网卡在统计周期内每秒接收的平均数据包数。 单位：包/秒	0-10000000 包/秒	Redis 3.0 Proxy集群实例Proxy节点	1分钟
avg_txpck_per_sec	网卡包发送平均速率	该指标用于统计测量对象网卡在统计周期内每秒发送的平均数据包数。 单位：包/秒	0-10000000 包/秒	Redis 3.0 Proxy集群实例Proxy节点	1分钟
avg_rxB_per_sec	入网平均带宽	该指标用于统计测量对象网卡每秒接收的平均数据量。 单位：KB/s。	>= 0KB/s	Redis 3.0 Proxy集群实例Proxy节点	1分钟
avg_txB_per_sec	出网平均带宽	该指标用于统计测量对象网卡每秒发送的平均数据量。 单位：KB/s。	>= 0KB/s	Redis 3.0 Proxy集群实例Proxy节点	1分钟

表 14-6 Redis 4.0/5.0 Proxy 集群和读写分离实例的 Proxy 节点支持的监控指标

指标ID	指标名称	指标含义	取值范围	测量对象	监控周期（原始指标）
node_status	实例节点状态	显示Proxy节点状态是否正常。	<ul style="list-style-type: none"> ● 0: 表示正常 ● 1: 表示异常 	Proxy集群、读写分离实例Proxy节点	1分钟
cpu_usage	CPU利用率	该指标对于统计周期内的测量对象的CPU使用率进行多次采样，表示多次采样的最高值。 单位：%。	0-100%	Proxy集群、读写分离实例Proxy节点	1分钟

指标ID	指标名称	指标含义	取值范围	测量对象	监控周期（原始指标）
cpu_avg_usage	CPU平均使用率	该指标对于统计周期内的测量对象的CPU使用率进行多次采样，表示多次采样的平均值。 单位：%。	0-100%	Proxy集群、读写分离实例Proxy节点	1分钟
memory_usage	内存利用率	该指标用于统计测量对象的内存利用率。 单位：%。	0-100%	Proxy集群、读写分离实例Proxy节点	1分钟
connected_clients	活跃的客户端数量	该指标用于统计已连接的客户端数量，包括系统监控、配置同步和业务相关的连接数，不包括来自从节点的连接。	>=0	Proxy集群、读写分离实例Proxy节点	1分钟
instantaneous_ops	每秒并发操作数	该指标用于统计每秒处理的命令数。	>=0	Proxy集群、读写分离实例Proxy节点	1分钟
instantaneous_input_kbps	网络瞬时输入流量	该指标用于统计瞬时的输入流量。 单位：KB/s。	>=0KB/s	Proxy集群、读写分离实例Proxy节点	1分钟
instantaneous_output_kbps	网络瞬时输出流量	该指标用于统计瞬时的输出流量。 单位：KB/s。	>=0KB/s	Proxy集群、读写分离实例Proxy节点	1分钟
total_net_input_bytes	网络收到字节数	该指标用于统计周期内收到的字节数。 单位：可在控制台进行选择，如KB、MB、byte等。	>=0	Proxy集群、读写分离实例Proxy节点	1分钟
total_net_output_bytes	网络发送字节数	该指标用于统计周期内发送的字节数。 单位：可在控制台进行选择，如KB、MB、byte等。	>=0	Proxy集群、读写分离实例Proxy节点	1分钟

指标ID	指标名称	指标含义	取值范围	测量对象	监控周期（原始指标）
connections_usage	连接数使用率	该指标用于统计当前连接数与最大连接数限制的百分比。单位：%。	0-100%	Proxy集群、读写分离实例Proxy节点	1分钟
command_max_rt	最大时延	节点从接收命令到发出响应的时延最大值。单位：us。	>=0us	Proxy集群、读写分离实例Proxy节点	1分钟
command_avg_rt	平均时延	节点从接收命令到发出响应的时延平均值。单位：us。	>=0us	Proxy集群、读写分离实例Proxy节点	1分钟

Memcached 实例监控指标

📖 说明

监控指标的维度请参考[维度](#)。

表 14-7 Memcached 实例支持的监控指标

指标ID	指标名称	指标含义	取值范围	测量对象	监控周期（原始指标）
cpu_usage	CPU利用率	该指标对于统计周期内的测量对象的CPU使用率进行多次采样，表示多次采样的最高值。单位：%。	0-100%	Memcached实例	1分钟
memory_usage	内存利用率	该指标用于统计测量对象的内存利用率。单位：%。	0-100%	Memcached实例	1分钟
net_in_throughput	网络输入吞吐量	该指标用于统计网口平均每秒的输入流量。单位：byte/s。	>= 0byte/s	Memcached实例	1分钟

指标ID	指标名称	指标含义	取值范围	测量对象	监控周期（原始指标）
net_out_throughput	网络输出吞吐量	该指标用于统计网口平均每秒的输出流量。 单位：byte/s。	>=0byte/s	Memcached实例	1分钟
mc_connected_clients	活跃的客户数量	该指标用于统计已连接的客户端数量，不包括来自从节点的连接。	>=0	Memcached实例	1分钟
mc_used_memory	已用内存	该指标用于统计已使用的内存字节数。 单位：byte。	>=0byte	Memcached实例	1分钟
mc_used_memory_rss	已用内存RSS	该指标用于统计已使用的RSS内存。即实际驻留“在内存中”的内存数。包含堆内存，但不包括换出的内存。 单位：byte。	>=0byte	Memcached实例	1分钟
mc_used_memory_peak	已用内存峰值	该指标用于统计服务器启动以来使用内存的峰值。 单位：byte。	>=0byte	Memcached实例	1分钟
mc_memory_frag_ratio	内存碎片率	该指标用于统计当前的内存碎片率。其数值上等于used_memory_rss / used_memory。	>=0	Memcached实例	1分钟
mc_connections_received	新建连接数	该指标用于统计周期内新建的连接数。	>=0	Memcached实例	1分钟
mc_commands_processed	处理的命令数	该指标用于统计周期内处理的命令数。	>=0	Memcached实例	1分钟
mc_instantaneous_ops	每秒并发操作数	该指标用于统计每秒处理的命令数。	>=0	Memcached实例	1分钟

指标ID	指标名称	指标含义	取值范围	测量对象	监控周期（原始指标）
mc_net_input_bytes	网络收到字节数	该指标用于统计周期内收到的字节数。 单位：byte。	>=0byte	Memcached实例	1分钟
mc_net_output_bytes	网络发送字节数	该指标用于统计周期内发送的字节数。 单位：byte。	>=0byte	Memcached实例	1分钟
mc_instantaneous_input_kbps	网络瞬时输入流量	该指标用于统计瞬时的输入流量。 单位：KB/s。	>=0KB/s	Memcached实例	1分钟
mc_instantaneous_output_kbps	网络瞬时输出流量	该指标用于统计瞬时的输出流量。 单位：KB/s。	>=0KB/s	Memcached实例	1分钟
mc_rejected_connections	已拒绝的连接数	该指标用于统计周期内因为超过maxclients而拒绝的连接数量。	>=0	Memcached实例	1分钟
mc_expired_keys	已过期的键数量	该指标用于统计周期内因过期而被删除的键数量。	>=0	Memcached实例	1分钟
mc_evicted_keys	已驱逐的键数量	该指标用于统计周期内因为内存不足被删除的键数量。	>=0	Memcached实例	1分钟
mc_cmd_get	数据查询请求次数	该指标用于统计服务收到的数据查询请求次数。	>=0	Memcached实例	1分钟
mc_cmd_set	数据存储请求次数	该指标用于统计服务收到的数据存储请求次数。	>=0	Memcached实例	1分钟
mc_cmd_flush	数据清空请求次数	该指标用于统计服务收到的数据清空请求次数。	>=0	Memcached实例	1分钟
mc_cmd_touch	数据有效期修改请求次数	该指标用于统计服务收到的数据有效期修改请求次数。	>=0	Memcached实例	1分钟
mc_get_hits	数据查询命中次数	该指标用于统计数据查询成功次数。	>=0	Memcached实例	1分钟

指标ID	指标名称	指标含义	取值范围	测量对象	监控周期（原始指标）
mc_get_misses	数据查询未命中次数	该指标用于统计数据因键不存在而失败的查询次数。	>=0	Memcached实例	1分钟
mc_delete_hits	数据删除命中次数	该指标用于统计数据删除成功次数。	>=0	Memcached实例	1分钟
mc_delete_misses	数据删除未命中次数	该指标用于统计因键不存在而失败的数据删除次数。	>=0	Memcached实例	1分钟
mc_incr_hits	算数加命中次数	该指标用于统计算数加操作成功次数。	>=0	Memcached实例	1分钟
mc_incr_misses	算数加未命中次数	该指标用于统计因键不存在而失败的算数加操作次数。	>=0	Memcached实例	1分钟
mc_decr_hits	算数减命中次数	该指标用于统计算数减操作成功次数。	>=0	Memcached实例	1分钟
mc_decr_misses	算数减未命中次数	该指标用于统计因键不存在而失败的算数减操作次数。	>=0	Memcached实例	1分钟
mc_cas_hits	CAS命中次数	该指标用于统计CAS操作成功次数。	>=0	Memcached实例	1分钟
mc_cas_misses	CAS未命中次数	该指标用于统计因键不存在而失败的CAS操作次数。	>=0	Memcached实例	1分钟
mc_cas_badval	CAS数值不匹配次数	该指标用于统计因CAS值不匹配而失败的CAS次数。	>=0	Memcached实例	1分钟
mc_touch_hits	数据有效期修改命中次数	该指标用于统计数据有效期修改成功次数。	>=0	Memcached实例	1分钟
mc_touch_misses	数据有效期修改未命中次数	该指标用于统计因键不存在而失败的数据有效期修改次数。	>=0	Memcached实例	1分钟
mc_auth_cmds	认证请求次数	该指标用于统计认证请求次数。	>=0	Memcached实例	1分钟

指标ID	指标名称	指标含义	取值范围	测量对象	监控周期（原始指标）
mc_auth_errors	认证失败次数	该指标用于统计认证失败次数。	>=0	Memcached实例	1分钟
mc_curr_items	存储的数据条目	该指标用于统计存储的数据条目。	>=0	Memcached实例	1分钟
mc_command_max_delay	命令最大时延	统计命令最大时延。 单位：ms。	>=0ms	Memcached实例	1分钟
mc_is_slow_log_exist	是否存在慢日志	统计实例是否存在慢日志。 说明 该监控不统计由migrate、slaveof、config、bgsave、bgrewriteaof命令导致的慢日志。	<ul style="list-style-type: none"> 1：表示存在 0：表示不存在。 	Memcached实例	1分钟
mc_keyspace_hits_percent	访问命中率	统计实例的访问码命中率。 单位：%。	0-100%	Memcached实例	1分钟

维度

Key	Value
dc_instance_id	Redis实例
dc_cluster_redis_node	数据节点
dc_cluster_proxy_node	Redis 3.0 Proxy集群实例Proxy节点
dc_cluster_proxy2_node	Redis 4.0及以上版本Proxy集群和读写分离实例Proxy节点
dc_memcached_instance_id	Memcached实例

14.2 DCS 常用的监控指标

本章节主要列举Redis的常用监控指标。

表 14-8 常用监控指标说明

指标名称	说明
CPU利用率	<p>该指标统计的是每个统计周期（分钟级就是每1分钟，秒级就是每5秒）内的最大值。</p> <ul style="list-style-type: none"> 如果是单机和主备实例，支持查看实例级别的CPU使用情况。 如果是Proxy集群实例，支持查看数据节点和Proxy节点的CPU使用情况。 如果是Cluster集群，仅支持查看数据节点的CPU使用情况。
内存利用率	<p>该指标统计的是每个统计周期（分钟级就是每1分钟，秒级就是每5秒）内的内存使用情况。</p> <ul style="list-style-type: none"> 如果是单机和主备实例，支持查看实例级别的内存使用情况。 如果是Proxy集群实例，支持查看实例级别和节点级别的内存使用情况。 如果是Cluster集群，仅支持查看数据节点的内存使用情况。 <p>须知 内存利用率统计是扣除预留内存的。</p>
活跃的客户数量	<p>该指标统计的是瞬时的已连接客户端数量，也叫连接并发数。</p> <p>如果是主备和集群实例，不包括从节点连接。</p> <p>活跃的客户数量上限，可以查看实例规格下对应实例类型的“连接数上限”。</p>
每秒并发操作数	<p>该指标统计的是瞬时的每秒处理的命令数。</p> <p>每秒并发操作数上限，可以查看实例规格下对应实例类型的“参考性能（QPS）”。</p>
网络瞬时输入流量	<p>该指标用于统计瞬时的输入数据流量。</p> <ul style="list-style-type: none"> 如果是实例级别的网络瞬时输入流量，所有节点输入数据流量汇总后展示。 如果是节点级别，统计的是本节点的输入数据流量。
网络瞬时输出流量	<p>该指标用于统计瞬时的输出数据流量。</p> <ul style="list-style-type: none"> 如果是实例级别的网络瞬时输出流量，所有节点输出数据流量汇总后展示。 如果是节点级别，统计的是本节点的输出数据流量。
带宽使用率	<p>该指标计算当前流量带宽与最大带宽限制的百分比。</p> <p>带宽使用率 = (网络瞬时输入流量 + 网络瞬时输出流量) / (2 * 最大带宽限制) * 100%</p>

指标名称	说明
处理的命令数	该指标统计的是周期内处理的命令数，周期默认为1分钟。和每秒并发操作数主要区别在于监控周期。每秒并发操作数，统计的是周期内的一个瞬时的处理命令数；处理的命令数，统计的是周期内处理的命令数总和。
流控次数	该指标用于统计周期内流量超过该实例规格对应的最大带宽的次数。 实例规格对应的最大带宽，可以查看 实例规格 下对应实例类型的“基准/最大带宽”。
慢查询	该指标用于统计实例是否存在慢查询。 慢查询产生的原因，请查看 慢查询 。

14.3 查看 DCS 性能监控

您可以通过云监控服务集中监控DCS的各种性能监控指标。

查看 DCS 性能监控

- 步骤1 登录[分布式缓存服务管理控制台](#)。
- 步骤2 在管理控制台左上角单击 ，选择实例所在的区域。
- 步骤3 单击左侧菜单栏的“缓存管理”，进入缓存实例信息页面。
- 步骤4 单击需要查看性能监控指标的缓存实例，进入实例基本信息页面。
- 步骤5 单击“性能监控”，页面显示该实例的所有监控指标信息。

说明

您也可以在需要查看的缓存实例的“操作”列，单击“查看监控”，进入云监控服务的页面查看，这和缓存实例信息页面“性能监控”页签内容一致。

----结束

14.4 配置 DCS 监报告警

本章节主要介绍部分监控指标的告警策略，以及配置操作。在实际业务中，请按照以下告警策略，配置监控指标的告警规则。

Redis 实例告警策略

表 14-9 Redis 实例配置告警的指标

指标名称	取值范围	告警策略	是否接近性能上限	告警处理建议
CPU利用率	0~100%	告警阈值： >70% 连续触发次数：2 告警级别： 重要	否	结合业务分析是否由于业务上涨导致的，判断是否需要扩容。 单机/主备实例，无法扩展CPU能力，如需扩展CPU能力，请考虑切换为集群实例。 该指标仅针对单机、主备、Proxy 集群实例设置，Cluster 集群实例级别不支持该指标，仅在数据节点支持，即需要在实例详情的“性能监控”中选择“数据节点”页签查看。
CPU平均使用率	0~100%	告警阈值： >70% 连续触发次数：2 告警级别： 重要	否	结合业务分析是否由于业务上涨导致的，判断是否需要扩容。 单机/主备实例，无法扩展CPU能力，如需扩展CPU能力，请考虑切换为集群实例。 该指标仅针对单机、主备、Proxy 集群实例设置，Cluster 集群实例级别不支持该指标，仅在数据节点支持，即需要在实例详情的“性能监控”中选择“数据节点”页签查看。
内存利用率	0~100%	告警阈值： >70% 连续触发次数：2 告警级别： 紧急	否	建议进行扩容。
活跃的客户端数量	0~10000	告警阈值： >8000 连续触发次数：2 告警级别： 重要	否	建议结合业务代码对连接池等进行优化，避免连接数超过最大限制。 仅单机和主备实例配置该指标。如果是集群实例，在数据节点和 Proxy 节点配置即可。 单机和主备实例，最大连接数限制为10000，可以根据业务情况对阈值进行调整。

指标名称	取值范围	告警策略	是否接近性能上限	告警处理建议
新建连接数 (个/min)	>=0	告警阈值: >10000 连续触发次数: 2 告警级别: 次要	-	排查是否使用短连接, 或者客户端异常连接。建议使用长连接, 避免使用短连接影响性能。 仅单机和主备实例配置该指标。如果是集群实例, 在数据节点和Proxy节点配置即可。
网络瞬时输入流量	>=0	告警阈值: >规格基准带宽的80% 连续触发次数: 2 告警级别: 重要	是	结合业务分析和规格带宽限制, 判断是否需要扩容。 仅Redis 3.0实例的单机/主备实例进行配置, 建议按Redis 3.0规格基准带宽的80%进行配置。其他实例不配置。
网络瞬时输出流量	>=0	告警阈值: >规格基准带宽的80% 连续触发次数: 2 告警级别: 重要	是	结合业务分析和规格带宽限制, 判断是否需要扩容。 仅Redis 3.0实例的单机/主备实例进行配置, 建议按Redis 3.0规格基准带宽的80%进行配置。其他实例不配置。

Memcached 实例告警策略

表 14-10 Memcache 实例建议配置告警的指标

指标名称	取值范围	告警策略	是否接近性能上限	告警处理建议
CPU利用率	0~100%	告警阈值: >70% 连续触发次数: 2 告警级别: 重要	否	结合业务分析是否由于业务上涨导致的。 如果是单机/主备实例, 无法扩展CPU能力, 需要结合业务分析是否可进行业务拆分或在客户端使用多个实例组建集群。
内存利用率	0~100%	告警阈值: >65% 连续触发次数: 2 告警级别: 次要	否	建议扩容。

指标名称	取值范围	告警策略	是否接近性能上限	告警处理建议
活跃的客户 端数量	0~10000	告警阈值： >8000 连续触发次 数：2 告警级别： 重要	否	建议结合业务代码对连接池等进行优化，避免连接数超过最大限制。
新建连接 数	>=0	告警阈值： >10000 连续触发次 数：2 告警级别： 次要	-	排查是否使用短连接，或者客户端异常连接。建议使用长连接，避免使用短连接影响性能。
网络瞬时 输入流量	>=0	告警阈值： >规格基准 带宽的80% 连续触发次 数：2 告警级别： 重要	是	结合业务分析和规格带宽限制，判断是否需要扩容。 不同实例规格的带宽，请查看 实例规格 页面。
网络瞬时 输出流量	>=0	告警阈值： >规格基准 带宽的80% 连续触发次 数：2 告警级别： 重要	是	结合业务分析和规格带宽限制，判断是否需要扩容。 不同实例规格的带宽，请查看 实例规格 页面。
认证失败 次数	>=0	告警阈值： >0 连续触发次 数：1 告警级别： 紧急	-	检查密码配置是否正确。

Redis 实例数据节点告警策略

表 14-11 Redis 实例数据节点建议配置告警的指标

指标名称	取值范围	告警策略	是否接近性能上限	告警处理建议
CPU利用率	0~100%	告警阈值： >70% 连续触发次数：2 告警级别： 重要	否	结合业务分析是否由于业务上涨导致的。 需要分析各个数据节点的CPU使用率分布是否均匀，如果节点普遍CPU高，需要考虑扩容，集群扩容会增加数据节点，分担CPU压力。 如果是单个节点CPU上涨，需要业务侧分析是否存在热key，优化业务侧代码消除热key。
CPU平均使用率	0~100%	告警阈值： >70% 连续触发次数：2 告警级别： 重要	否	结合业务分析是否由于业务上涨导致的，判断是否需要扩容。 如果读写分离/单机/主备实例，无法扩展CPU能力，需要考虑切换为集群实例。
内存利用率	0~100%	告警阈值： >70% 连续触发次数：2 告警级别： 重要	否	结合业务分析是否由于业务上涨导致的。 需要分析各个数据节点的内存利用率分布是否均匀，如果节点普遍内存利用率高，需要考虑扩容。如果是单个节点内存上涨，需要业务侧分析是否存在大key，优化业务侧代码消除热大key。
活跃的客户数量	0~10000	告警阈值： >8000 连续触发次数：2 告警级别： 重要	否	分析业务，是否合理，如果结合业务分析连接数是合理的，建议调整告警阈值。
新建连接数	>=0	告警阈值： >10000 连续触发次数：2 告警级别： 次要	-	新建连接数多，可能是短连接导致，建议使用长连接，避免使用短连接影响性能。

指标名称	取值范围	告警策略	是否接近性能上限	告警处理建议
是否存在慢日志	0~1	告警阈值： >0 连续触发次数：1 告警级别： 重要	-	通过慢查询功能分析具体的慢日志命令。
带宽使用率	0~200%	告警阈值： >90% 连续触发次数：2 告警级别： 重要	是	<p>可结合网络瞬时输入流量和网络瞬时输出流量，分析业务是读业务和还是写业务导致的流量上涨。</p> <p>对于单个节点带宽使用率上涨，需要分析是否有存在大key。</p> <p>其中，带宽使用率超过100%，不一定导致限流，有没有被流控需要看流控次数指标。</p> <p>带宽使用率没有超过100%，也有可能有限流，因为带宽使用率是上报周期实时值，一个上报周期检查一次。流控检查是秒级的。有可能存在上报周期期间，流量有秒级冲高，然后回落，待上报带宽使用率指标时已恢复正常。</p>
流控次数	>=0	告警阈值： >0 连续触发次数：1 告警级别： 紧急	是	<p>结合规格限制、网络瞬时输入流量和网络瞬时输出流量，查看是否扩容解决。</p> <p>说明 Redis 4.0以上版本的实例才支持该指标，Redis 3.0实例不支持。</p>

Redis 实例 Proxy 节点告警策略

表 14-12 Proxy 节点建议配置告警的指标

指标名称	取值范围	告警策略	是否接近性能上限	告警处理建议
CPU利用率	0~100%	告警阈值： >70% 连续触发次数：2 告警级别： 紧急	是	建议考虑扩容，扩容会增加 proxy 节点。

指标名称	取值范围	告警策略	是否接近性能上限	告警处理建议
内存利用率	0~100%	告警阈值： >70% 连续触发次数：2 告警级别： 紧急	是	建议考虑扩容，扩容会增加 proxy 节点。
活跃的客户端数量	0-30000	告警阈值： >20000 连续触发次数：2 告警级别： 重要	否	建议结合业务代码对连接池等进行优化，避免连接数超过最大限制。

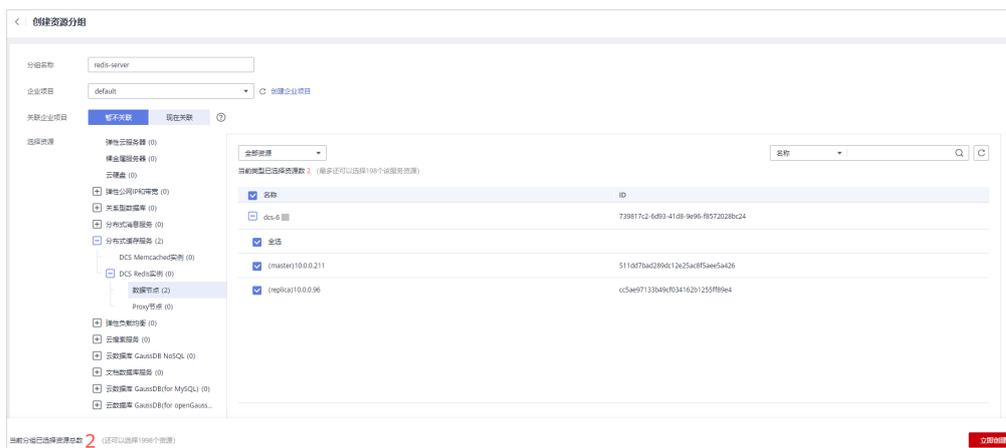
配置告警（按资源分组）

云监控服务支持资源分组功能，在使用DCS缓存服务时，您可以按照实例级别、数据节点、Proxy节点创建资源分组，从分组角度管理实例，管理告警规则，可以极大的降低运维复杂度，提高运维效率。资源分组的更多信息，请参考[资源分组介绍](#)。

步骤1 创建资源分组。

1. 登录云监控服务管理控制台，在左侧选择“资源分组”，然后单击页面右上角的“创建资源分组”按钮。
2. 按照界面提示，设置分组名称，选择需要添加的数据节点。
数据节点，可以同时选择不同实例的数据节点。

图 14-1 创建资源分组

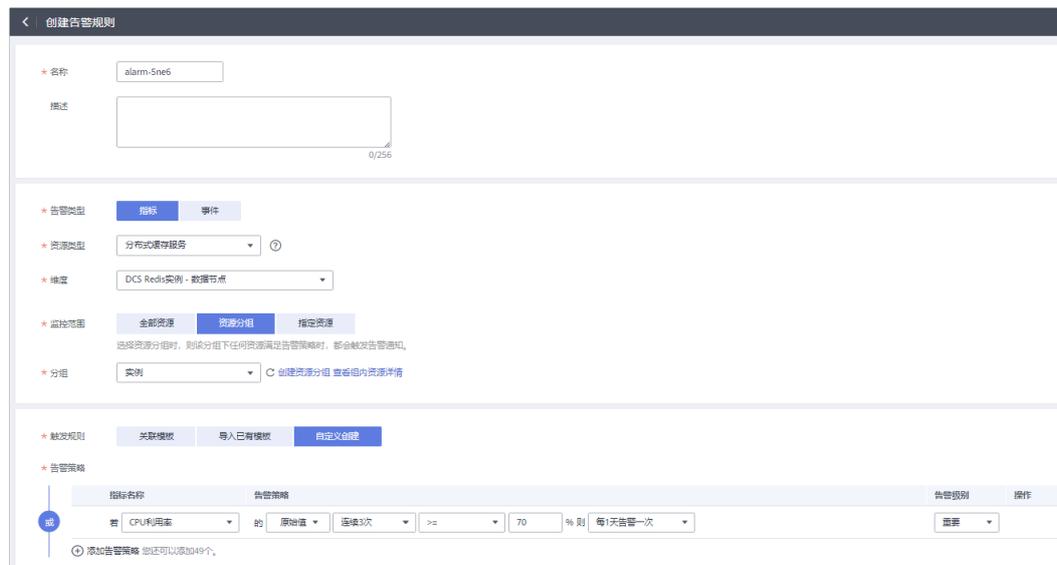


3. 单击“立即创建”，完成资源分组的创建。

步骤2 在左侧选择“告警 > 告警规则”，单击“创建告警规则”，为资源分组设置告警信息。

如下图所示，为资源分组内的所有数据节点创建CPU利用率的告警规则。

图 14-2 为资源分组创建告警规则



步骤3 单击“立即创建”，等待创建告警规则成功。

----结束

配置步骤（按指定资源）

以配置是否存在慢日志（is_slow_log_exist）监控指标的告警规则为例：

步骤1 登录[分布式缓存服务管理控制台](#)。

步骤2 在管理控制台左上角单击 ，选择实例所在的区域。

步骤3 单击左侧菜单栏的“缓存管理”，进入缓存实例信息页面。

步骤4 在需要查看的缓存实例的“操作”列，单击“查看监控”，进入该实例的监控指标页面。

图 14-3 查看实例监控指标

名称	状态	缓存类型	实例类型	CPU	规格 (GB)	已用/可用内存	连接地址	企业项目	标签	计费方式	操作
dc-100 0c259b33-3a0b-4be4...	运行中	基础版 Redis 5.0	Cluster-集群	x86	8	9/8.1...	redis-0c259b33...	default	-	按量计费	查看监控 刷新 更多

步骤5 在实例监控指标页面中，找到指标名称为“是否存在慢日志”的指标项，鼠标移动到

指标区域，然后单击指标右上角的 ，创建告警规则。

跳转到创建告警规则页面。

步骤6 在告警规则页面，设置告警信息。

1. 设置告警名称和告警的描述。
2. 设置告警策略和告警级别。

如图14-4所示，在指标监控时，如果连续2个周期，客户执行了耗时命令，产生了慢查询，则产生告警，如果未及时处理，则每一天发送一次告警通知，直至指标恢复到0，表示不存在慢日志。

图 14-4 设置告警内容

3. 设置“发送通知”开关。当开启时，设置告警生效时间、产生告警时通知的对象以及触发的条件。
4. 单击“立即创建”，等待创建告警规则成功。

📖 说明

- 如果创建告警规则有问题，可查看云监控服务的用户指南“[创建告警规则](#)”。
- 如果需要修改或停用所创建的告警，请参考[告警规则管理](#)。

----结束

15 查看 DCS 审计日志

DCS通过云审计服务（Cloud Trace Service，简称CTS）为您提供云服务资源的操作记录，记录内容包括您从公有云管理控制台或者开放API发起的云服务资源操作请求以及每次请求的结果，供您查询、审计和回溯使用。

云审计服务主要支持的 DCS 操作

表 15-1 云审计服务主要支持的 DCS 操作列表

操作类型	资源类型	事件名称
创建实例	Redis	createDCSInstance
提交创建实例请求	Redis	submitCreateDCSInstanceRequest
批量删除实例	Redis	batchDeleteDCSInstance
删除实例	Redis	deleteDCSInstance
修改实例信息	Redis	modifyDCSInstanceInfo
修改实例配置	Redis	modifyDCSInstanceConfig
修改实例密码	Redis	modifyDCSInstancePassword
停止实例	Redis	stopDCSInstance
提交停止实例请求	Redis	submitStopDCSInstanceRequest
重启实例	Redis	restartDCSInstance
提交重启实例请求	Redis	submitRestartDCSInstanceRequest
启动实例	Redis	startDCSInstance
提交启动实例请求	Redis	submitStartDCSInstanceRequest
清空实例	Redis	flushDCSInstance

操作类型	资源类型	事件名称
批量停止实例	Redis	batchStopDCSInstance
提交批量停止实例请求	Instance	submitBatchStopDCSInstanceRequest
批量重启实例	Redis	batchRestartDCSInstance
提交批量重启实例请求	Redis	submitBatchRestartDCSInstanceRequest
批量启动实例	Redis	batchStartDCSInstance
提交批量启动实例请求	Instance	submitBatchStartDCSInstanceRequest
恢复实例数据	Redis	restoreDCSInstance
提交恢复实例数据请求	Redis	submitRestoreDCSInstanceRequest
备份实例数据	Redis	backupDCSInstance
提交备份实例数据请求	Redis	submitBackupDCSInstanceRequest
删除实例备份文件	Redis	deleteInstanceBackupFile
删除后台任务记录	Redis	deleteDCSInstanceJobRecord
实例规格变更	Redis	modifySpecification
提交实例规格变更请求	Redis	submitModifySpecificationRequest
创建实例订单	Redis	createInstanceOrder
创建实例规格变更订单	Redis	createSpecificationChangeOrder
更新企业项目ID	Redis	updateEnterpriseProjectId
主备切换	Redis	masterStandbySwitchover
关闭公网访问	Redis	disablePublicNetworkAccess
开启公网访问	Redis	enablePublicNetworkAccess
重置实例密码	Redis	resetDCSInstancePassword
提交清空实例请求	Redis	submitFlushDCSInstanceRequest
WebCli登录	Redis	webCliLogin

操作类型	资源类型	事件名称
webCli命令执行	Redis	webCliCommand
webCli登出	Redis	webCliLogout
离线迁移	Redis	offlineMigrate
计费模式变更	Redis	billingModeChange
更新实例标签	Redis	updateInstanceTag
修改白名单配置	Instance	modifyWhiteList
修改实例带宽	Redis	modifyBandwidth

查看审计日志

查看DCS云审计日志，请参考[查看审计事件](#)。