分布式缓存服务

用户指南

发布日期　2025-11-14

目 录

1 产品介绍

1.1 分布式缓存服务是什么?

分布式缓存服务（Distributed Cache Service，简称DCS）是一款兼容Redis的高速内存数据处理引擎，为您提供即开即用、安全可靠、弹性扩容、便捷管理的在线分布式缓存能力，满足用户高并发及数据快速访问的业务诉求。

- 即开即用

  DCS提供单机、主备和集群不同类型的缓存实例，拥有从128MB到1024GB的丰富内存规格。您可以通过控制台直接创建，无需单独准备服务器资源。

  Redis 4.0/5.0/6.0/7.0版本采用容器化部署，秒级完成创建。

- 安全可靠

  借助统一身份认证、虚拟私有云、云监控与云审计等安全管理服务，全方位保护实例数据的存储与访问。

  灵活的容灾策略，主备/集群实例从单AZ（可用区）内部署，到支持跨AZ部署。

- 弹性伸缩

  DCS提供对实例内存规格的在线扩容与缩容服务，帮助您实现基于实际业务量的成本控制，达到按需使用的目标。

- 便捷管理

  可视化Web管理界面，在线完成实例重启、参数修改、数据备份恢复等操作。DCS还提供基于RESTful的管理API，方便您进一步实现实例自动化管理。

- 在线迁移

  提供可视化Web界面迁移功能，支持备份文件导入和在线迁移两种方式，您可以通过控制台直接创建迁移任务，提高迁移效率。

DCS Redis

Redis是一种支持Key-Value等多种数据结构的存储系统。可用于缓存、事件发布或订阅、高速队列等**典型应用场景**。Redis使用ANSI C语言编写，提供字符串（**String**）、哈希（**Hash**）、列表（**List**）、集合结构（**Set**、**Sorted_Set**）、流（**Stream**）等数据类型的直接存取。数据读写基于内存，同时可持久化到磁盘。

DCS Redis拥有灵活的实例配置供您选择：

表 1-1 DCS Redis 灵活的实例配置

实例类型	提供单机、主备、Proxy集群、Cluster集群类型，分别适配不同的业务场景。 单机：适用于应用对可靠性要求不高、仅需要缓存临时数据的业务场景。单机实例支持读写高并发，但不做持久化，实例重启或关闭后原有缓存数据不会加载。 主备：包含一个主节点，一个备节点，主备节点的数据通过实时复制保持一致，当主节点故障后，备节点自动升级为主节点。 Proxy集群：在Cluster集群的基础上，增加挂载Proxy节点和ELB节点，通过ELB节点实现负载均衡，将不同请求分发到Proxy节点，实现客户端高并发请求。每个Cluster集群分片默认是一个双副本的主备实例，当主节点故障后，同一分片中的备节点会升级为主节点来继续提供服务。 Cluster集群：通过分片化分区来增加缓存的容量和并发连接数，每个分片包含一个主节点和0到多个备节点，分片本身对外不可见。分片中主节点故障后，同一分片中备节点会升级为主节点来继续提供服务。用户可通过读写分离技术，在主节点上写，从备节点读，从而提升缓存的整体读写能力。
规格	Redis提供128MB~1024GB的多种规格。
兼容开源Redis版本	DCS提供不同的实例版本，分别兼容开源Redis的4.0、5.0、6.0、7.0。
底层架构	基于大规格虚拟机部署，单节点QPS达10万。
高可用与容灾	主备与集群实例提供Region内的跨可用区部署，实现实例内部节点间的电力、网络层面物理隔离。

有关开源Redis技术细节，您可以访问Redis官方网站**https://redis.io/**了解。

1.2 典型应用场景

Redis 应用场景

很多大型电商网站、视频直播和游戏应用等，存在**大规模数据访问**，对**数据查询效率要求高**，且**数据结构简单**，**不涉及太多关联查询**。这种场景使用Redis，在速度上对传统磁盘数据库有很大优势，能够有效减少数据库磁盘IO，提高数据查询效率，减轻管理维护工作量，降低数据库存储成本。Redis对传统磁盘数据库是一个重要的补充，成为了互联网应用，尤其是支持高并发访问的互联网应用必不可少的基础服务之一。

以下举几个典型样例：

1. **（电商网站）秒杀抢购**

   电商网站的商品类目、推荐系统以及秒杀抢购活动，适宜使用Redis缓存数据库。

   例如秒杀抢购活动，并发高，对于传统关系型数据库来说访问压力大，需要较高的硬件配置（如磁盘IO）支撑。Redis数据库，单节点QPS支撑能达到10万，轻松应对秒杀并发。实现秒杀和数据加锁的命令简单，使用SET、GET、DEL、RPUSH等命令即可。

加锁部分，可参考最佳实践：使用DCS实现热点资源顺序访问

2. **（视频直播）消息弹幕**

   直播间的在线用户列表，礼物排行榜，弹幕消息等信息，都适合使用Redis中的SortedSet结构进行存储。

   例如弹幕消息，可使用ZREVRANGEBYSCORE排序返回，在Redis 5.0中，新增了zpopmax，zpopmin命令，更加方便消息处理。

3. **（游戏应用）游戏排行榜**

   在线游戏一般涉及排行榜实时展现，比如列出当前得分最高的10个用户。使用Redis的有序集合存储用户排行榜非常合适，有序集合使用非常简单，提供多达20个操作集合的命令。

   可参考最佳实践：使用Redis实现排行榜功能

4. **（社交APP）返回最新评论/回复**

   在web类应用中，常有“最新评论”之类的查询，如果使用关系型数据库，经常涉及到按评论时间逆排序，随着评论越来越多，排序效率越来越低，且并发频繁。

   使用Redis的List（链表），例如存储最新1000条评论，当请求的评论数在这个范围，就不需要访问磁盘数据库，直接从缓存中返回，减少数据库压力的同时，提升APP的响应速度。

1.3 实例类型

1.3.1 Redis 单机实例

Redis单机实例为单节点架构，不支持数据持久化，适用于不要求数据可靠性的缓存业务场景。

说明

- Redis 4.0及以上版本不支持升级实例大版本，例如，不支持Redis 4.0单机升级为Redis 5.0单机实例。如果需要使用高版本Redis单机实例，建议重新创建高版本Redis单机实例，然后将原有Redis实例的数据迁移到高版本实例上。
- 单机实例无法保证数据持久性，且不支持数据自动或手动备份，选用前请务必确认风险。

单机实例特点

1. 低成本，适用于开发测试

   单机实例各种规格的成本相对主备减少40%以上。适用于开发、测试环境搭建。

2. 数据不做持久化

   单机实例仅有主节点，无法保证数据持久性，且不支持数据自动或手动备份。

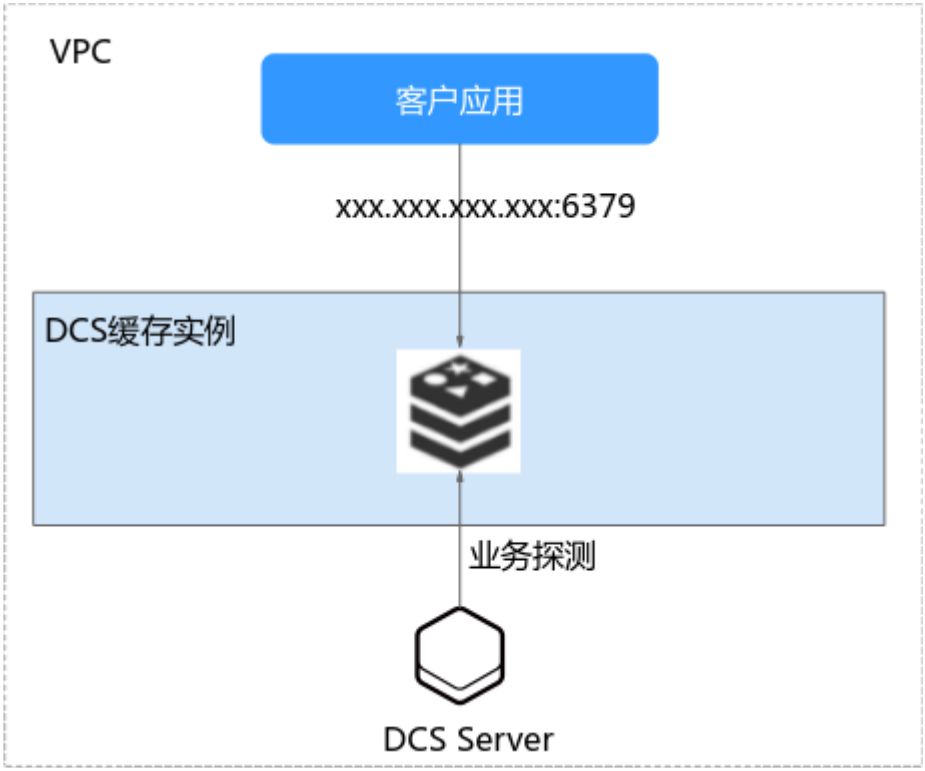
实例架构设计

DCS的Redis单机实例架构，如图1-1所示。

说明

Redis 3.0不支持定义端口，端口固定为6379，Redis 4.0及以上版本支持定义端口，如果不自定义端口，则使用默认端口6379。以下图中以默认端口6379为例，如果已自定义端口，请根据实际情况替换。

图 1-1 Redis 单机实例示意图



示意图说明：

- **VPC**
  虚拟私有云。实例的内部所有服务器节点，都运行在相同VPC中。

  说明

  VPC内访问，客户端需要与实例处于相同VPC。
- **客户应用**
  运行在ECS上的客户应用程序，即实例的客户端。
  Redis实例兼容开源协议，可直接使用开源客户端进行连接，关于客户端连接示例，请参考**连接实例**。
- **DCS缓存实例**
  DCS单机实例只有1个节点，1个Redis进程。

1.3.2 Redis 主备实例

本章节主要介绍Redis缓存类型的主备实例。

说明

Redis 4.0及以上版本不支持升级实例大版本，例如，不支持Redis 4.0主备升级为Redis 5.0主备实例。如果需要使用高版本Redis主备实例，建议重新创建高版本Redis主备实例，然后将原有Redis实例的数据迁移到高版本实例上。

主备实例特点

DCS的主备实例在单机实例基础上，增强服务高可用以及数据高可靠性。

主备实例具有以下特性：

1. **持久化，确保数据高可靠**

   实例默认包含一个主节点和一个备节点，都默认开启数据持久化。

   Redis主备实例的备节点对用户不可见，不支持客户端直接读写数据。

2. **数据同步**

   主备节点通过增量数据同步的方式保持缓存数据一致。

   说明

   当网络发生异常或有节点故障时，主备实例会在故障恢复后进行一次全量同步，保持数据一致性。

3. **故障后自动切换主节点，服务高可用**

   当主节点故障后，连接会有秒级中断、不可用，备节点在15秒到30秒内自动完成主备切换，切换完成后恢复正常访问，无需用户操作，业务平稳运行。

   说明

   - 故障切换期间，会有连接中断和不可用等情况，需要业务侧客户端具备重连/重试机制。
   - 主备切换完成后，原主节点（已切换为从节点）因故障不会立刻恢复，业务继续访问原主节点会失败，可通过配置Redis SDK解决此类情况，具体请参见**多语言连接**。

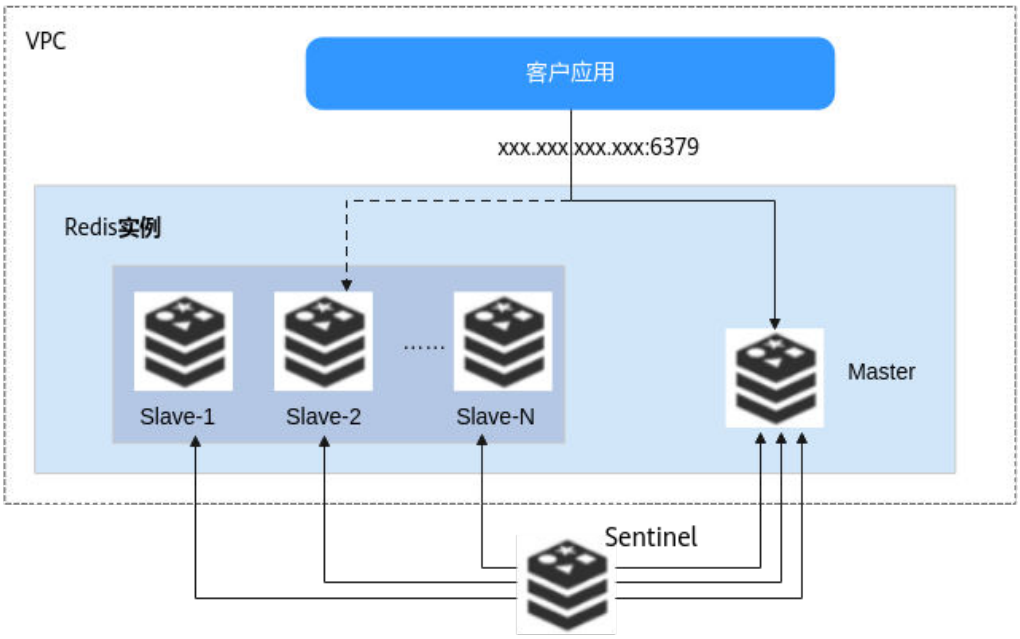
4. **容灾策略**

   跨AZ部署（可用区）：DCS支持将主备、集群实例的主备副本部署在不同的AZ内，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。

Redis 4.0/5.0/6.0/7.0 主备实例架构设计

Redis 4.0/5.0/6.0/7.0主备实例的架构设计，如下图所示。

图 1-2 Redis 4.0/5.0/6.0/7.0 主备实例示意图



图说明如下：

1. Redis 4.0及以上版本主备实例使用哨兵模式（Sentinel）进行管理，Sentinel会一直监控主备节点是否正常运行，当主节点出现故障时，进行主备倒换。

   Sentinel对用户不可见，仅在服务内部中使用。Sentinel的详细介绍可参考**什么是哨兵**。
2. 备节点和主节点规格一致，用户创建主备实例时，默认包含一个主节点和一个备节点。
3. Redis 4.0及以上版本实例支持定义端口，如果不自定义端口，则使用默认端口6379。图中以默认端口6379为例，如果已自定义端口，请根据实际情况替换。

1.3.3 Redis Cluster 集群实例

DCS Redis Cluster集群实例，是原生Cluster的集群版本。Redis Cluster集群实例的特点：

- 兼容Redis原生Cluster集群。
- 继承smart client的设计方案。
- 相比主备，数倍性能提升。

Redis集群实例中， Proxy集群实例不支持读写分离，Cluster集群实例支持从客户端实现读写分离，相关操作请参考**Cluster集群实例读写分离**。

Cluster 集群实例

Cluster版Redis集群兼容**开源Redis的Cluster**，基于smart client和无中心的设计方案，对服务器进行分片。

Cluster版Redis集群每种实例规格对应的分片数，如**表1-2**所示。

每个分片的大小=实例规格/分片数，例如，集群规格为48GB的实例，分片数为6，则每个集群分片的大小为48GB/6=8GB。

表 1-2 Cluster 集群实例规格和分片数的对应关系

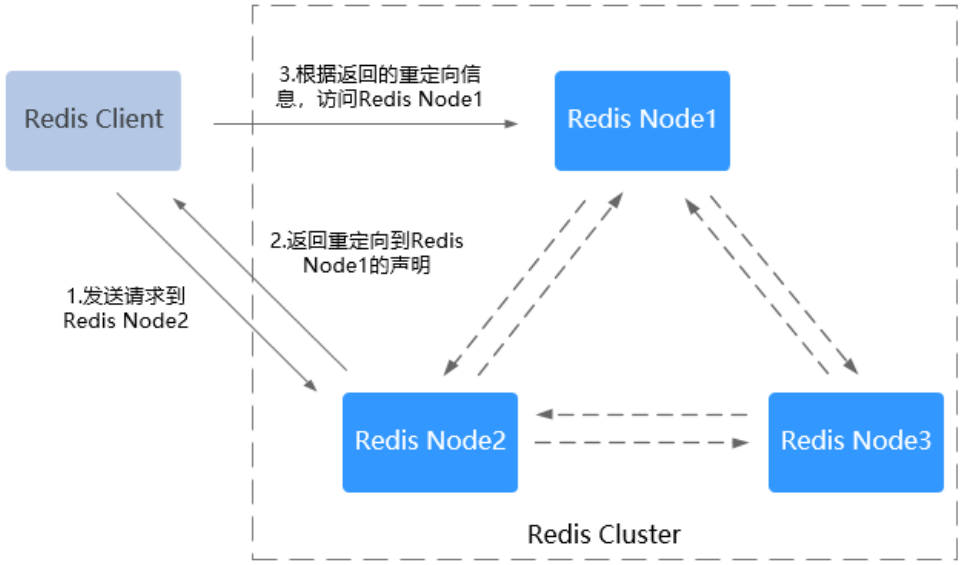
集群版规格	分片数
4GB/8GB/16GB/24GB/32GB	3
48GB	6
64GB	8
96GB	12
128GB	16
192GB	24
256GB	32
384GB	48
512GB	64
768GB	96

集群版规格	分片数
1024GB	128

- 无中心架构

  Redis Cluster的任意节点都可以接收请求，但节点会将请求发送到正确的节点上执行，同时，每一个节点也是主从结构，默认包含一个主节点和一个从节点，由Redis Cluster根据选举算法决定节点主从属性。

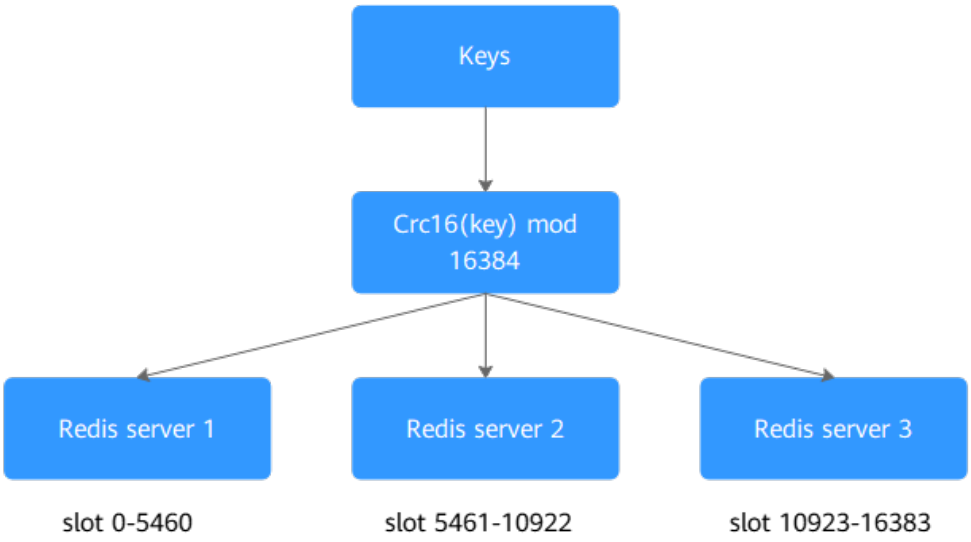
  图 1-3 Redis Cluster 无中心架构



- 数据预分片

  Redis Cluster会预先分配16384个slot，每个Redis的server存储所有slot与redis server的映射关系。key存储在哪个slot中，由Crc16(key) mod 16384的值决定。如下图所示：

  图 1-4 Redis Cluster 预分片示意图



说明

- Redis Cluster集群的每个分片也是一个Redis主备实例。当分片上的主节点故障时，该分片上的连接会有秒级中断、不可用，备节点在15秒到30秒内自动完成主备切换，单分片故障仅影响该分片上的数据访问，不影响其他分片上的数据访问。
- Redis集群单分片主节点故障时，主备切换完成后，该分片原主节点（已切换为从节点）因故障不会立刻恢复，业务继续访问该分片原主节点会失败，可通过配置Redis SDK解决此类情况，具体请参见**多语言连接**。

1.3.4 Redis 实例类型差异

Redis单机、主备、Proxy集群和Cluster集群实例，在特性支持、特性限制以及命令限制有部分差异，具体请查看**表1-3**。

表 1-3 不同实例类型的差异说明

对比项	单机/主备	Cluster集群
兼容Redis版本	单机/主备兼容开源Redis4.0/5.0/6.0/7.0。可在创建实例时选择版本号。	兼容开源Redis 4.0/5.0/6.0/7.0版本。可在创建实例时选择版本号。
特性支持	• 支持event notify。 • 支持pipeline。	• 支持event notify。 • 支持brpop、blpop、brpoplpush。 • 支持发布订阅。
特性限制	仅单机实例不支持数据持久化及备份与恢复的功能。	• lua脚本受限使用，所有的key必须在同一个slot，建议使用**hashtag**技术。 • 需要客户端SDK支持redis cluster协议，需要能够处理"-MOVED"响应。 • 使用pipeline、mset/mget模式时，所有key必须属于同一个slot，否则报错，建议使用**hashtag**技术。 • 使用event notify时，需要建立与每个redis-server的连接，分别处理每个连接上的事件。 • 执行scan、keys等遍历类或者全局类命令时，需要对每个redis-server分别执行该命令。
客户端协议	使用传统Redis客户端即可。	需要客户端支持Redis Cluster协议。
命令限制	不支持的Redis命令，请参考**开源命令兼容性**。	不支持的Redis命令，请参考**开源命令兼容性**。

对比项	单机/主备	Cluster集群
副本数	单机实例为单副本，只有一个节点。 主备实例默认为双副本，默认为一主一从的架构。 在创建Redis主备实例时，支持自定义副本数，形成一主多从的架构。目前Redis 3.0主备不支持自定义副本数。	每个集群分片默认为双副本，支持自定义副本数，每个分片可以是一主多从的架构。在创建实例时，也可以定义为单副本，单副本表示实例只有主节点，无法保障数据高可靠。

1.4 实例规格

1.4.1 Redis 4.0/5.0 实例

本节介绍DCS Redis 4.0/5.0实例的产品规格，包括内存规格、实例可使用内存、最大连接数、最大带宽/基准带宽、参考性能（QPS）等。

实例各项指标如下：

- 实例已使用内存：您可以通过查看监控指标“内存利用率”和“已用内存”查看实例内存使用情况。
- 最大连接数：表示允许客户端同时连接的个数，即连接并发数。最大连接数对应参数maxclients（Proxy集群实例不支持该参数），实例创建后支持在控制台“实例配置 > 参数配置”中修改。具体实例的连接数，可查看监控指标“活跃的客户端数量”。
- QPS：即Query Per Second，表示数据库每秒执行的命令数。
- 带宽：您可以查看监控指标“流控次数”，确认带宽是否超过限额。

说明

- 支持“单机”、“主备”和“Cluster集群”三种类型。
- 仅支持x86的CPU架构，不支持Arm架构。

单机实例

表 1-4 Redis 4.0 和 Redis 5.0 单机实例产品规格

内存规格（GB）	实例可使用内存（GB）	最大连接数（默认/可配）（个）	基准/最大带宽（Mbit/s）	参考性能（QPS）	产品规格编码（对应API的spec_code）
0.125	0.125	10,000/10,000	40/40	80,000	redis.single.xu1.tiny.128

内存规格（GB）	实例可使用内存（GB）	最大连接数（默认/可配）（个）	基准/最大带宽（Mbit/s）	参考性能（QPS）	产品规格编码（对应API的spec_code）
0.25	0.25	10,000/10,000	80/80	80,000	redis.single.xu1.tiny.256
0.5	0.5	10,000/10,000	80/80	80,000	redis.single.xu1.tiny.512
1	1	10,000/50,000	80/80	80,000	redis.single.xu1.large.1
2	2	10,000/50,000	128/128	80,000	redis.single.xu1.large.2
4	4	10,000/50,000	192/192	80,000	redis.single.xu1.large.4
8	8	10,000/50,000	192/192	100,000	redis.single.xu1.large.8
16	16	10,000/50,000	256/256	100,000	redis.single.xu1.large.16
24	24	10,000/50,000	256/256	100,000	redis.single.xu1.large.24
32	32	10,000/50,000	256/256	100,000	redis.single.xu1.large.32
48	48	10,000/50,000	256/256	100,000	redis.single.xu1.large.48
64	64	10,000/50,000	384/384	100,000	redis.single.xu1.large.64

主备实例

主备实例默认2个副本数（包含主副本），主节点个数为1。

主备实例占用的IP个数=主节点个数*副本个数。例如：

主备2副本实例，占用IP个数=1*2=2；

主备3副本实例，占用IP个数=1*3=3。

下表中仅列出了默认副本数为2时，对应的实例规格名称（产品规格编码），如果是其他副本个数，名称中相应修改副本数量。例如，8G规格的x86架构的主备实例，主备2副本的实例规格名称为redis.ha.xu1.large.**r2**.8，3副本为redis.ha.xu1.large.**r3**.8，以此类推。

表 1-5 Redis 4.0 和 Redis 5.0 主备实例产品规格

内存规格（GB）	实例可使用内存（GB）	最大连接数（默认/可配）（个）	基准/最大带宽（Mbit/s）	参考性能（QPS）	产品规格编码（对应API的spec_code）
0.125	0.125	10,000/10,000	40/40	80,000	redis.ha.xu1.tiny.r2.128
0.25	0.25	10,000/10,000	80/80	80,000	redis.ha.xu1.tiny.r2.256
0.5	0.5	10,000/10,000	80/80	80,000	redis.ha.xu1.tiny.r2.512
1	1	10,000/50,000	80/80	80,000	redis.ha.xu1.large.r2.1
2	2	10,000/50,000	128/128	80,000	redis.ha.xu1.large.r2.2
4	4	10,000/50,000	192/192	80,000	redis.ha.xu1.large.r2.4
8	8	10,000/50,000	192/192	100,000	redis.ha.xu1.large.r2.8
16	16	10,000/50,000	256/256	100,000	redis.ha.xu1.large.r2.16
24	24	10,000/50,000	256/256	100,000	redis.ha.xu1.large.r2.24
32	32	10,000/50,000	256/256	100,000	redis.ha.xu1.large.r2.32
48	48	10,000/50,000	256/256	100,000	redis.ha.xu1.large.r2.48
64	64	10,000/50,000	384/384	100,000	redis.ha.xu1.large.r2.64

Cluster 集群实例

Cluster集群实例与单机、主备实例的区别，不仅在于支持高规格内存，在客户端连接数、内网带宽上限、QPS指标都有很大的提升。

- 产品规格名称：下表中仅列出了x86架构，默认副本数为2时，对应的实例规格名称（产品规格编码），如果是其他副本个数，名称中相应修改副本数量。例如，8GB规格的x86 2副本的规格名称为redis.cluster.xu1.large.**r2**.8，3副本为redis.cluster.xu1.large.**r3**.8，以此类推。
- 占用IP个数：占用的IP个数=分片数*副本个数。例如：

  4GB规格的Cluster 3副本实例，分片数为3，则实例占用IP个数=3*3=9。

- 单个节点可使用内存：单个节点可使用内存=实例可使用内存/主节点个数。例如：

  24GB规格实例，实例可使用内存为24GB，主节点个数为3，则单个节点可使用内存=24/3=8GB。

- 单个节点最大连接数：单个节点最大连接数=实例最大连接数/主节点个数。例如：

  4GB规格实例，实例配置了最大连接数为150000，主节点个数为3，则单个节点最大连接数=150000/3=50000个。

说明

- 集群实例的“最大连接数”是实例的最大连接数，单分片的最大连接数=实例最大连接数/分片数。
- 集群实例的“最大带宽/基准带宽”是实例的最大带宽/基准带宽，而不是单个分片的宽带。实例带宽与单分片带宽的关系如下：
  - 实例带宽=单分片带宽*分片数。
  - 当集群实例单分片内存为1 GB时，单分片带宽为384 Mbit/s，当集群实例单分片内存大于1 GB，单分片带宽为768 Mbit/s。

表 1-6 Redis 4.0 和 Redis 5.0 Cluster 集群实例产品规格

规格（GB）	实例可使用内存（GB）	分片数(主节点个数)	实例最大连接数（默认/可配）（个）	基准/最大带宽（Mbit/s）	参考性能（QPS）	产品规格编码（对应API的spec_code）
4	4	3	30,000/150,000	2,304/2,304	240,000	redis.cluster.xu1.large.r2.4
8	8	3	30,000/150,000	2,304/2,304	240,000	redis.cluster.xu1.large.r2.8
16	16	3	30,000/150,000	2,304/2,304	240,000	redis.cluster.xu1.large.r2.16
24	24	3	30,000/150,000	2,304/2,304	300,000	redis.cluster.xu1.large.r2.24
32	32	3	30,000/150,000	2,304/2,304	300,000	redis.cluster.xu1.large.r2.32
48	48	6	60,000/300,000	4,608/4,608	>300,000	redis.cluster.xu1.large.r2.48
64	64	8	80,000/400,000	6,144/6,144	500,000	redis.cluster.xu1.large.r2.64
96	96	12	120,000/600,000	9,216/9,216	>500,000	redis.cluster.xu1.large.r2.96

规格（GB）	实例可使用内存（GB）	分片数(主节点个数)	实例最大连接数（默认/可配）（个）	基准/最大带宽（Mbit/s）	参考性能（QPS）	产品规格编码（对应API的spec_code）
128	128	16	160,000/800,000	12,288/12,288	1,000,000	redis.cluster.xu1.large.r2.128
192	192	24	240,000/1,200,000	18,432/18,432	>1,000,000	redis.cluster.xu1.large.r2.192
256	256	32	320,000/1,600,000	24,576/24,576	>2,000,000	redis.cluster.xu1.large.r2.256
384	384	48	480,000/2,400,000	36,864/36,864	>2,000,000	redis.cluster.xu1.large.r2.384
512	512	64	640,000/3,200,000	49,152/49,152	>2,000,000	redis.cluster.xu1.large.r2.512
768	768	96	960,000/4,800,000	73,728/73,728	>2,000,000	redis.cluster.xu1.large.r2.768
1024	1024	128	1,280,000/6,400,000	98,304/98,304	>2,000,000	redis.cluster.xu1.large.r2.1024

1.4.2 Redis 6.0/7.0 实例

本节介绍DCS Redis 6.0/7.0实例的产品规格，包括内存规格、实例可使用内存、最大连接数、最大带宽/基准带宽、参考性能（QPS）等。

实例各项指标如下：

- 实例已使用内存：您可以通过查看监控指标“内存利用率”和“已用内存”查看实例内存使用情况。
- 最大连接数：表示允许客户端同时连接的个数，即连接并发数。最大连接数对应参数maxclients，实例创建后支持在控制台“实例配置 > 参数配置”中修改。具体实例的连接数，可查看监控指标“活跃的客户端数量”。
- QPS：即Query Per Second，表示数据库每秒执行的命令数。
- 带宽：您可以查看监控指标“流控次数”，确认带宽是否超过限额。

Redis 6.0版本的实例目前支持单机、主备和Cluster集群。CPU类型为x86架构。

单机实例

表 1-7 Redis 6.0/7.0 单机实例产品规格

内存规格（GB）	实例可使用内存（GB）	最大连接数（默认/最大可配）（个）	基准/最大带宽（Mbit/s）	参考性能（QPS）	产品规格编码(对应API的spec_code)
0.125	0.125	10,000/10,000	40/40	80,000	redis.single.xu1.tiny.128
0.25	0.25	10,000/10,000	80/80	80,000	redis.single.xu1.tiny.256
0.5	0.5	10,000/10,000	80/80	80,000	redis.single.xu1.tiny.512
1	1	10,000/50,000	80/80	80,000	redis.single.xu1.large.1
2	2	10,000/50,000	128/128	80,000	redis.single.xu1.large.2
4	4	10,000/50,000	192/192	80,000	redis.single.xu1.large.4
8	8	10,000/50,000	192/192	100,000	redis.single.xu1.large.8
16	16	10,000/50,000	256/256	100,000	redis.single.xu1.large.16
24	24	10,000/50,000	256/256	100,000	redis.single.xu1.large.24
32	32	10,000/50,000	256/256	100,000	redis.single.xu1.large.32
48	48	10,000/50,000	256/256	100,000	redis.single.xu1.large.48
64	64	10,000/50,000	384/384	100,000	redis.single.xu1.large.64

主备实例

表 1-8 Redis 6.0/7.0 主备实例产品规格

内存规格（GB）	实例可使用内存（GB）	最大连接数（默认/最大可配）（个）	基准/最大带宽（Mbit/s）	参考性能（QPS）	产品规格编码(对应API的spec_code)
0.125	0.125	10,000/10,000	40/40	80,000	redis.ha.xu1.tiny.r2.128
0.25	0.25	10,000/10,000	80/80	80,000	redis.ha.xu1.tiny.r2.256
0.5	0.5	10,000/10,000	80/80	80,000	redis.ha.xu1.tiny.r2.512
1	1	10,000/50,000	80/80	80,000	redis.ha.xu1.large.r2.1
2	2	10,000/50,000	128/128	80,000	redis.ha.xu1.large.r2.2
4	4	10,000/50,000	192/192	80,000	redis.ha.xu1.large.r2.4
8	8	10,000/50,000	192/192	100,000	redis.ha.xu1.large.r2.8
16	16	10,000/50,000	256/256	100,000	redis.ha.xu1.large.r2.16
24	24	10,000/50,000	256/256	100,000	redis.ha.xu1.large.r2.24
32	32	10,000/50,000	256/256	100,000	redis.ha.xu1.large.r2.32
48	48	10,000/50,000	256/256	100,000	redis.ha.xu1.large.r2.48
64	64	10,000/50,000	384/384	100,000	redis.ha.xu1.large.r2.64

Cluster 集群实例

- 产品规格编码（实例规格名称）：表1-9中仅列出了默认2副本的实例规格名称，如果是副本个数为1时，名称中相应修改副本数量。例如，4GB规格的2副本实例的规格名称为redis.cluster.xu1.large.r**2**.4，副本数为1时，规格名称为redis.cluster.xu1.large.r**1**.4，其他规格以此类推。
- **占用IP个数**：占用的IP个数=分片数*副本个数。例如：

  8GB规格的Cluster 2副本实例，分片数为3，则实例占用IP个数=3*2=6。

- **单个节点可使用内存**：单个节点可使用内存=实例可使用内存/主节点个数。例如：

  64GB规格实例，实例可使用内存为64G，主节点个数为8，则单个节点可使用内存=64/8=8GB。

- **单个节点最大连接数**：单个节点最大连接数=实例最大连接数/主节点个数。例如：

  24GB规格实例，实例配置了最大连接数为150000，主节点个数为3，则单个节点最大连接数=150000/3=50000个。

说明

下表中的“最大带宽/基准带宽”是实例的最大带宽/基准带宽，而不是单个分片的宽带。实例带宽与单分片带宽的关系如下：

- 实例带宽=单分片带宽*分片数。
- 当集群实例单分片内存为1GB时，单分片带宽为384Mbit/s，当集群实例单分片内存大于1GB，单分片带宽为768Mbit/s。

  例如，24GB规格的Cluster集群实例的分片数为3，单分片内存为24/3=8GB（大于1GB），则该实例的单分片带宽为768Mbit/s。

表 1-9 Redis 6.0/7.0 Cluster 集群实例产品规格

内存规格（GB）	实例可使用内存（GB）	分片数(主节点个数)	最大连接数（默认/最大可配）（个）	基准/最大带宽（Mbit/s）	参考性能（QPS）	产品规格编码(对应API的spec_code)
4	4	3	30,000/150,000	2,304/2,304	240,000	redis.cluster.xu1.large.r2.4
8	8	3	30,000/150,000	2,304/2,304	240,000	redis.cluster.xu1.large.r2.8
16	16	3	30,000/150,000	2,304/2,304	240,000	redis.cluster.xu1.large.r2.16
24	24	3	30,000/150,000	2,304/2,304	300,000	redis.cluster.xu1.large.r2.24
32	32	3	30,000/150,000	2,304/2,304	300,000	redis.cluster.xu1.large.r2.32
48	48	6	60,000/300,000	4,608/4,608	>300,000	redis.cluster.xu1.large.r2.48
64	64	8	80,000/400,000	6,144/6,144	500,000	redis.cluster.xu1.large.r2.64
96	96	12	120,000/600,000	9,216/9,216	>500,000	redis.cluster.xu1.large.r2.96

内存规格（GB）	实例可使用内存（GB）	分片数(主节点个数)	最大连接数（默认/最大可配）（个）	基准/最大带宽（Mbit/s）	参考性能（QPS）	产品规格编码(对应API的spec_code)
128	128	16	160,000/800,000	12,288/12,288	1,000,000	redis.cluster.xu1.large.r2.128
192	192	24	240,000/1,200,000	18,432/18,432	>1,000,000	redis.cluster.xu1.large.r2.192
256	256	32	320,000/1,600,000	24,576/24,576	>2,000,000	redis.cluster.xu1.large.r2.256
384	384	48	480,000/2,400,000	36,864/36,864	>2,000,000	redis.cluster.xu1.large.r2.384
512	512	64	640,000/3,200,000	49,152/49,152	>2,000,000	redis.cluster.xu1.large.r2.512
768	768	96	960,000/4,800,000	73,728/73,728	>2,000,000	redis.cluster.xu1.large.r2.768
1024	1024	128	1,280,000/6,400,000	98,304/98,304	>2,000,000	redis.cluster.xu1.large.r2.1024
2048	2048	128	2,560,000/12,800,000	98,304/98,304	>2,000,000	redis.cluster.xu1.large.r2.2048

1.5 开源命令兼容性

1.5.1 Redis 4.0 支持及禁用的命令

DCS Redis 4.0基于开源4.0.14版本进行开发，兼容开源的协议和命令。

本章节主要介绍DCS Redis 4.0命令的兼容性，包括支持命令列表，禁用命令列表。命令的具体详细语法，请前往**Redis官方网站**查看。

DCS Redis缓存实例支持Redis的绝大部分命令，具体支持的命令，请参考**Redis 4.0支持的命令**，任何兼容Redis协议的客户端都可以访问DCS。

- 因安全原因，部分Redis命令在分布式缓存服务中被禁用，具体请见**Redis 4.0禁用的命令**。
- DCS集群实例支持多个key，但不支持跨slot访问的Redis命令列表，如**实例受限使用命令**所示。
- 部分Redis命令使用时有限制，具体请见**部分命令使用限制**。

Redis 4.0 支持的命令

表1-10和表1-11列举了Redis 4.0单机、主备、cluster集群实例支持的Redis命令。

说明

- Redis高版本的命令，在低版本中不被兼容。判断DCS Redis是否支持某个命令，可通过在Redis-cli执行该命令，如果得到（error）ERR unknown command ‘xxx’的提示，则说明不支持该命令。
- Redis 4.0 Cluster版本集群实例使用pipeline时，要确保管道中的命令都能在同一分片执行。

表 1-10 Redis 4.0 单机、主备、Cluster 集群支持命令清单 1

Keys	String	Hash	List	Set	Sorted Set	Server
DEL	APPEND	HDEL	BLPOP	SADD	ZADD	FLUSHALL
DUMP	BITCOUNT	HEXISTS	BRPOP	SCARD	ZCARD	FLUSHDB
EXISTS	BITOP	HGET	BRPOPLRUSH	SDIFF	ZCOUNT	DBSIZE
EXPIRE	BITPOS	HGETALL	LINDEX	SDIFFSTORE	ZINCRBY	TIME
MOVE	DECR	HINCRBY	LINSERT	SINTER	ZRANGE	INFO
PERSIST	DECRBY	HINCRBYFLOAT	LLEN	SINTERSTORE	ZRANGEBYSCORE	KEYS
PTTL	GET	HKEYS	LPOP	SISMEMBER	ZRANK	CLIENT KILL
RANDOMKEY	GETRANGE	HMGET	LPUSHX	SMEMBERS	ZREMRANGEBYRANK	CLIENT LIST
RENAME	GETSET	HMSET	LRANGE	SMOVE	ZREMRANGEBYCORE	CLIENT GETNAME
RENAMENX	INCR	HSET	LREM	SPOP	ZREVRANGE	CLIENT SETNAME
RESTORE	INCRBY	HSETNX	LSET	SRANDMEMBER	ZREVRANGEBYSCORE	CONFIG GET
SORT	INCRBYFLOAT	HVALS	LTRIM	SREM	ZREVRANK	MONITOR
TTL	MGET	HSCAN	RPOP	SUNION	ZSCORE	SLOWLOG
TYPE	MSET	HSTRLEN	RPOPLPU	SUNIONSTORE	ZUNIONSTORE	ROLE

Keys	String	Hash	List	Set	Sorted Set	Server
SCAN	MSETNX	HLEN	RPOPLPUSH	SSCAN	ZINTERSTORE	SWAPDB
OBJECT	PSETEX	-	RPUSH	-	ZSCAN	MEMORY
PEXPIRE	SET	-	RPUSHX	-	ZRANGEBYLEX	CONFIG
PEXPIREAT	SETBIT	-	LPUSH	-	ZLEXCOUNT	COMMAND
-	SETEX	-	-	-	ZREMRANGEBYSCORE	-
-	SETNX	-	-	-	ZREM	-
-	SETRANGE	-	-	-	-	-
-	STRLEN	-	-	-	-	-
-	BITFIELD	-	-	-	-	-

表 1-11 Redis 4.0 单机、主备、Cluster 集群支持命令清单 2

HyperLogLog	Pub/Sub	Transactions	Connection	Scripting	Geo
PFADD	PSUBSCRIBE	DISCARD	AUTH	EVAL	GEOADD
PFCOUNT	PUBLISH	EXEC	ECHO	EVALSHA	GEOHASH
PFMERGE	PUBSUB	MULTI	PING	SCRIPT EXISTS	GEOPOS
-	PUNSUBSCRIBE	UNWATCH	QUIT	SCRIPT FLUSH	GEODIST
-	SUBSCRIBE	WATCH	SELECT（Cluster集群实例不支持）	SCRIPT KILL	GEORADIUS
-	UNSUBSCRIBE	-	-	SCRIPT LOAD	GEORADIUSBYMEMBER

Redis 4.0 禁用的命令

以下列出了Redis 4.0实例禁用的命令。

表 1-12 Redis 4.0 单机和主备禁用命令

Keys	Server
MIGRATE	SLAVEOF
-	SHUTDOWN
-	LASTSAVE
-	DEBUG相关类
-	SAVE
-	BGSAVE
-	BGREWRITEAOF
-	SYNC
-	PSYNC

表 1-13 Redis 4.0 Cluster 集群禁用命令

Keys	Server	Cluster
MIGRATE	SLAVEOF	CLUSTER MEET
-	SHUTDOWN	CLUSTER FLUSHSLOTS
-	LASTSAVE	CLUSTER ADDSLOTS
-	DEBUG相关类	CLUSTER DELSLOTS
-	SAVE	CLUSTER SETSLOT
-	BGSAVE	CLUSTER BUMPEPOCH
-	BGREWRITEAOF	CLUSTER SAVECONFIG
-	SYNC	CLUSTER FORGET
-	PSYNC	CLUSTER REPLICATE
-	-	CLUSTER COUNT-FAILURE-REPORTS
-	-	CLUSTER FAILOVER
-	-	CLUSTER SET-CONFIG-EPOCH
-	-	CLUSTER RESET

1.5.2 Redis 5.0 支持及禁用的命令

DCS Redis 5.0基于开源5.0.14版本进行开发，兼容开源的协议和命令。

本章节主要介绍DCS Redis 5.0命令的兼容性，包括支持命令列表，禁用命令列表。命令的具体详细语法，请前往**Redis官方网站**查看。

DCS Redis缓存实例支持Redis的绝大部分命令，任何兼容Redis协议的客户端都可以访问DCS。

- 因安全原因，部分Redis命令在分布式缓存服务中被禁用，具体请见**Redis 5.0禁用的命令**。
- DCS集群实例支持多个key，但不支持跨slot访问的Redis命令列表，如**实例受限使用命令**所示。
- 部分Redis命令使用时有限制，具体请见**部分命令使用限制**。

Redis 5.0 支持的命令

- **表1-14**和**表1-15**列举了Redis 5.0单机、主备、Cluster集群实例支持的命令。

说明

- Redis高版本的命令，在低版本中不被兼容。判断DCS Redis是否支持某个命令，可通过在Redis-cli执行该命令，如果得到（error）ERR unknown command ‘xxx’的提示，则说明不支持该命令。
- Redis 5.0 Cluster版本集群实例使用pipeline时，要确保管道中的命令都能在同一分片执行。

表 1-14 Redis 5.0 单机、主备、Cluster 集群支持命令清单 1

Keys	String	Hash	List	Set	Sorted Set	Server
DEL	APPEND	HDEL	BLPOP	SADD	ZADD	FLUSHALL
DUMP	BITCOUNT	HEXISTS	BRPOP	SCARD	ZCARD	FLUSHDB
EXISTS	BITOP	HGET	BRPOPLRUSH	SDIFF	ZCOUNT	DBSIZE
EXPIRE	BITPOS	HGETALL	LINDEX	SDIFFSTORE	ZINCRBY	TIME
MOVE	DECR	HINCRBY	LINSERT	SINTER	ZRANGE	INFO
PERSIST	DECRBY	HINCRBYFLOAT	LLEN	SINTERSTORE	ZRANGEBYSCORE	KEYS
PTTL	GET	HKEYS	LPOP	SISMEMBER	ZRANK	CLIENT KILL
RANDOMKEY	GETRANGE	HMGET	LPUSHX	SMEMBERS	ZREMRANGEBYRANK	CLIENT LIST
RENAME	GETSET	HMSET	LRANGE	SMOVE	ZREMRANGEBYCORE	CLIENT GETNAME
RENAMENX	INCR	HSET	LREM	SPOP	ZREVRANGE	CLIENT SETNAME

Keys	String	Hash	List	Set	Sorted Set	Server
RESTORE	INCRBY	HSETNX	LSET	SRANDMEMBER	ZREVRANGEBYSCORE	CONFIG GET
SORT	INCRBYFLOAT	HVALS	LTRIM	SREM	ZREVRANK	MONITOR
TTL	MGET	HSCAN	RPOP	SUNION	ZSCORE	SLOWLOG
TYPE	MSET	HSTRLEN	RPOPLPU	SUNIONSTORE	ZUNIONSTORE	ROLE
SCAN	MSETNX	HLEN	RPOPLPUSH	SSCAN	ZINTERSTORE	SWAPDB
OBJECT	PSETEX	-	RPUSH	-	ZSCAN	MEMORY
PEXPIREAT	SET	-	RPUSHX	-	ZRANGEBYLEX	CONFIG
PEXPIRE	SETBIT	-	LPUSH	-	ZLEXCOUNT	COMMAND
-	SETEX	-	-	-	ZPOPMIN	-
-	SETNX	-	-	-	ZPOPMAX	-
-	SETRANGE	-	-	-	ZREMRANGEBYSCORE	-
-	STRLEN	-	-	-	ZREM	-
-	BITFIELD	-	-	-	-	-

表 1-15 Redis 5.0 单机、主备、Cluster 集群支持命令清单 2

HyperLoglog	Pub/Sub	Transactions	Connection	Scripting	Geo	Stream
PFADD	PSUBSCRIBE	DISCARD	AUTH	EVAL	GEOADD	XACK
PFCOUNT	PUBLISH	EXEC	ECHO	EVALSHA	GEOHASH	XADD
PFMERGE	PUBSUB	MULTI	PING	SCRIPT EXISTS	GEOPOS	XCLAIM
-	PUNSUBSCRIBE	UNWATCH	QUIT	SCRIPT FLUSH	GEODIST	XDEL

HyperLoglog	Pub/Sub	Transactions	Connection	Scripting	Geo	Stream
-	SUBSCRIBE	WATCH	SELECT（Cluster集群实例不支持）	SCRIPT KILL	GEORADIUS	XGROUP
-	UNSUBSCRIBE	-	-	SCRIPT LOAD	GEORADIUSBYMEMBER	XINFO
-	-	-	-	-	-	XLEN
-	-	-	-	-	-	XPENDING
-	-	-	-	-	-	XRANGE
-	-	-	-	-	-	XREAD
-	-	-	-	-	-	XREADGROUP
-	-	-	-	-	-	XREVRANGE
-	-	-	-	-	-	XTRIM

Redis 5.0 禁用的命令

以下列出了Redis 5.0实例禁用的命令。

表 1-16 Redis 5.0 单机和主备禁用命令

Keys	Server
MIGRATE	SLAVEOF
-	SHUTDOWN
-	LASTSAVE
-	DEBUG相关类
-	SAVE
-	BGSAVE
-	BGREWRITEAOF
-	SYNC
-	PSYNC

表 1-17 Redis 5.0 Cluster 集群禁用命令

Keys	Server	Cluster
MIGRATE	SLAVEOF	CLUSTER MEET
-	SHUTDOWN	CLUSTER FLUSHSLOTS
-	LASTSAVE	CLUSTER ADDSLOTS
-	DEBUG相关类	CLUSTER DELSLOTS
-	SAVE	CLUSTER SETSLOT
-	BGSAVE	CLUSTER BUMPEPOCH
-	BGREWRITEAOF	CLUSTER SAVECONFIG
-	SYNC	CLUSTER FORGET
-	PSYNC	CLUSTER REPLICATE
-	-	CLUSTER COUNT-FAILURE-REPORTS
-	-	CLUSTER FAILOVER
-	-	CLUSTER SET-CONFIG-EPOCH
-	-	CLUSTER RESET

1.5.3 Redis 6.0 支持及禁用的命令

DCS Redis 6.0完全兼容开源Redis 6。

本章节主要介绍DCS Redis 6.0命令的兼容性，包括支持命令列表，禁用命令列表。

DCS Redis缓存实例支持Redis的绝大部分命令，任何兼容Redis协议的客户端都可以访问DCS。

- 因安全原因，部分Redis命令在分布式缓存服务中被禁用，具体请见**Redis 6.0禁用的命令**。
- DCS集群实例支持多个key，但不支持跨slot访问的Redis命令列表，如**实例受限使用命令**所示。
- **部分Redis命令使用时有限制，例如KEYS、FLUSHDB、FLUSHALL等**，具体请见**部分命令使用限制**。

Redis 6.0 支持的命令

各个命令的具体详细语法请前往**Redis官方网站**查看，例如您想了解SCAN命令的使用，可在**Redis官方网站**中搜索框中输入“SCAN”查询详细介绍。

表 1-18 Redis 6.0 实例支持命令清单 1

Generic (Key)	String	Hash	List	Set	Sorted Set	Server
DEL	APPEND	HDEL	BLPOP	SADD	ZADD	FLUSHALL
DUMP	BITCOUNT	HEXISTS	BRPOP	SCARD	ZCARD	FLUSHDB
EXISTS	BITOP	HGET	BRPOPLRUSH	SDIFF	ZCOUNT	DBSIZE
EXPIRE	BITPOS	HGETALL	LINDEX	SDIFFSTORE	ZINCRBY	TIME
MOVE	DECR	HINCRBY	LINSERT	SINTER	ZRANGE	INFO
PERSIST	DECRBY	HINCRBYFLOAT	LLEN	SINTERSTORE	ZRANGEBYSCORE	CONFIG GET
PTTL	GET	HKEYS	LPOP	SISMEMBER	ZRANK	MONITOR
RANDOMKEY	GETRANGE	HMGET	LPUSHX	SMEMBERS	ZREMRANGEBYRANK	SLOWLOG
RENAME	GETSET	HMSET	LRANGE	SMOVE	ZREMRANGEBYCORE	ROLE
RENAMENX	INCR	HSET	LREM	SPOP	ZREVRANGE	SWAPDB
RESTORE	INCRBY	HSETNX	LSET	SRANDMEMBER	ZREVRANGEBYSCORE	MEMORY
SORT	INCRBYFLOAT	HVALS	LTRIM	SREM	ZREVRANK	CONFIG
TTL	MGET	HSCAN	RPOP	SUNION	ZSCORE	ACL
TYPE	MSET	HSTRLEN	RPOPLPU	SUNIONSTORE	ZUNIONSTORE	COMMAND
SCAN	MSETNX	HLEN	RPOPLPUSH	SSCAN	ZINTERSTORE	-
OBJECT	PSETEX	-	RPUSH	SMISMEMBER	ZSCAN	-
PEXPIREAT	SET	-	RPUSHX	-	ZRANGEBYLEX	-
PEXPIRE	SETBIT	-	LPUSH	-	ZLEXCOUNT	-

Generic (Key)	String	Hash	List	Set	Sorted Set	Server
KEYS	SETEX	-	BLMOVE	-	ZPOPMIN	-
COPY	SETNX	-	LMOVE	-	ZPOPMAX	-
-	SETRANGE	-	LPOS	-	ZREMRANGEBYSCORE	-
-	STRLEN	-	-	-	ZREM	-
-	BITFIELD	-	-	-	ZDIFF	-
-	BITFIELD_RO	-	-	-	ZDIFFSTORE	-
-	GETDEL	-	-	-	ZINTER	-
-	GETEX	-	-	-	ZMSCORE	-
-	-	-	-	-	ZRANDMEMBER	-
-	-	-	-	-	ZRANGESTORE	-
-	-	-	-	-	ZUNION	-

表 1-19 Redis 6.0 实例支持命令清单 2

HyperLoglog	Pub/Sub	Transactions	Connection	Scripting	Geo	Stream
PFADD	PSUBSCRIBE	DISCARD	AUTH	EVAL	GEOADD	XACK
PFCOUNT	PUBLISH	EXEC	ECHO	EVALSHA	GEOHASH	XADD
PFMERGE	PUBSUB	MULTI	PING	SCRIPT EXISTS	GEOPOS	XCLAIM
-	PUNSUBSCRIBE	UNWATCH	QUIT	SCRIPT FLUSH	GEODIST	XDEL
-	SUBSCRIBE	WATCH	SELECT（Cluster集群实例不支持）	SCRIPT KILL	GEORADIUS	XGROUP

HyperLoglog	Pub/Sub	Transactions	Connection	Scripting	Geo	Stream
-	UNSUBSCRIBE	-	CLIENT CACHING	SCRIPT LOAD	GEORADIUSBYMEMBER	XINFO
-	-	-	CLIENT GETREDIR	-	-	XLEN
-	-	-	CLIENT INFO	-	-	XPENDING
-	-	-	CLIENT TRACKING	-	-	XRANGE
-	-	-	CLIENT TRACKINGINFO	-	-	XREAD
-	-	-	CLIENT UNPAUSE	-	-	XREADGROUP
-	-	-	CLIENT KILL	-	-	XREVRANGE
-	-	-	CLIENT LIST	-	-	XTRIM
-	-	-	CLIENT GETNAME	-	-	XAUTOCLAIM
-	-	-	CLIENT SETNAME	-	-	XGROUP CREATECONSUMER
-	-	-	HELLO	-	-	-
-	-	-	RESET	-	-	-

Redis 6.0 禁用的命令

表 1-20 Redis 6.0 实例的禁用命令

Generic (Key)	Server	Cluster
MIGRATE	SLAVEOF	CLUSTER MEET
-	SHUTDOWN	CLUSTER FLUSHSLOTS

Generic (Key)	Server	Cluster
-	LASTSAVE	CLUSTER ADDSLOTS
-	DEBUG相关类	CLUSTER DELSLOTS
-	SAVE	CLUSTER SETSLOT
-	BGSAVE	CLUSTER BUMPEPOCH
-	BGREWRITEAOF	CLUSTER SAVECONFIG
-	SYNC	CLUSTER FORGET
-	PSYNC	CLUSTER REPLICATE
-	-	CLUSTER COUNT-FAILURE-REPORTS
-	-	CLUSTER FAILOVER
-	-	CLUSTER SET-CONFIG-EPOCH
-	-	CLUSTER RESET

1.5.4 Redis 7.0 支持及禁用的命令

本章节主要介绍DCS Redis 7.0命令的兼容性，包括支持命令列表，禁用命令列表。

DCS Redis缓存实例支持Redis的绝大部分命令，任何兼容Redis协议的客户端都可以访问DCS。

- 因安全原因，部分Redis命令在分布式缓存服务中被禁用，具体请见**Redis 7.0禁用的命令**。
- DCS集群实例支持多个key，但不支持跨slot访问的Redis命令列表，如**实例受限使用命令**所示。
- **部分Redis命令使用时有限制，例如KEYS、FLUSHDB、FLUSHALL等**，具体请见**部分命令使用限制**。

Redis 7.0 支持的命令

各个命令的具体详细语法请前往**Redis官方网站**查看，例如您想了解SCAN命令的使用，可在**Redis官方网站**中搜索框中输入“SCAN”查询详细介绍。

表 1-21 Redis 7.0 单机/主备/Cluster 集群实例支持命令清单 1

Generic (Key)	String	Hash	List	Set	Sorted Set	Server	Bitmap
COPY	APPEND	HDEL	BLMOVE	SADD	BZMPOP	ACL CAT	BITCOUNT

Generic (Key)	String	Hash	List	Set	Sorted Set	Server	Bitmap
DEL	DECR	HEXISTS	BLMPOP	SCARD	BZPOPMAX	ACL DRYRUN	BITFIELD
DUMP	DECRBY	HGET	BLPOP	SDIFF	BZPOPMIN	ACL GENPASS	BITFIELD_RO
EXISTS	GET	HGETALL	BRPOP	SDIFFSTORE	ZADD	ACL GETUSER	BITOP
EXPIRE	GETDEL	HINCRBY	BRPOPLPUSH	SINTER	ZCARD	ACL LIST	BITPOS
EXPIREAT	GETEX	HINCRBYFLOAT	LINDEX	SINTERCARD	ZCOUNT	ACL LOG	GETBIT
EXPIRETIME	GETRANGE	HKEYS	LINSERT	SINTERSTORE	ZDIFF	ACL USERS	SETBIT
KEYS	GETSET	HLEN	LLEN	SISMEMBER	ZDIFFSTORE	ACL WHOAMI	-
MOVE	INCR	HMGET	LMOVE	SMEMBERS	ZINCRBY	COMMAND COUNT	-
OBJECT ENCODING	INCRBY	HMSET	LMPOP	SMISMEMBER	ZINTER	COMMAND DOCS	-
OBJECT FREQ	INCRBYFLOAT	HRANDFIELD	LPOP	SMOVE	ZINTERCARD	COMMAND GETKEYS	-
OBJECT IDLETIME	LCS	HSCAN	LPOS	SPOP	ZINTERSTORE	COMMAND GETKEYSANDFLAGS	-
OBJECT REFCOUNT	MGET	HSET	LPUSH	SRANDMEMBER	ZLEXCOUNT	COMMAND INFO	-
PERSIST	MSET	HSETNX	LPUSHX	SREM	ZMPOP	COMMAND LIST	-

Generic (Key)	String	Hash	List	Set	Sorted Set	Server	Bitmap
PEXPIRE	MSETNX	HSTRLEN	LRANGE	SSCAN	ZMSCORE	COMMAND	-
PEXPIREAT	PSETEX	HVALS	LREM	SUNION	ZPOPMAX	CONFIG GET	-
PEXPIRETIME	SET	-	LSET	SUNIONSTORE	ZPOPMIN	DBSIZE	-
PTTL	SETEX	-	LTRIM	-	ZRANDMEMBER	FAILOVER	-
RANDOMKEY	SETNX	-	RPOP	-	ZRANGE	FLUSHALL	-
RENAME	SETRANGE	-	RPOPLPUSH	-	ZRANGEBYLEX	FLUSHDB	-
RENAMENX	STRLEN	-	RPUSH	-	ZRANGEBYSCORE	INFO	-
RESTORE	SUBSTR	-	RPUSHX	-	ZRANGESTORE	LASTSAVE	-
SCAN	-	-	-	-	ZRANK	LATENCY DOCTOR	-
SORT	-	-	-	-	ZREM	LATENCY GRAPH	-
SORT_RO	-	-	-	-	ZREMRANGEBYLEX	LATENCY HISTOGRAM	-
TOUCH	-	-	-	-	ZREMRANGEBYRANK	LATENCY HISTORY	-
TTL	-	-	-	-	ZREMRANGEBYSCORE	LATENCY LATEST	-
TYPE	-	-	-	-	ZREVRANGE	LATENCY RESET	-

Generic (Key)	String	Hash	List	Set	Sorted Set	Server	Bitmap
UNLINK	-	-	-	-	ZREVRANGEBYLEX	LOLWUT	-
WAIT	-	-	-	-	ZREVRANGEBYSCORE	MEMORY DOCTOR	-
WAITAOF	-	-	-	-	ZREVRANK	MEMORY MALLOC-STATS	-
-	-	-	-	-	ZSCAN	MEMORY PURGE	-
-	-	-	-	-	ZSCORE	MEMORY STATS	-
-	-	-	-	-	ZUNION	MEMORY USAGE	-
-	-	-	-	-	ZUNIONSTORE	MONITOR	-
-	-	-	-	-	-	REPLCONF	-
-	-	-	-	-	-	RESTORE-ASKING	-
-	-	-	-	-	-	ROLE	-
-	-	-	-	-	-	SLOWLOG GET	-
-	-	-	-	-	-	SLOWLOG LEN	-
-	-	-	-	-	-	SLOWLOG RESET	-
-	-	-	-	-	-	SWAPDB	-
-	-	-	-	-	-	TIME	-

表 1-22 Redis 7.0 单机/主备/Cluster 集群实例支持命令清单 2

HyperLoglog	Pub/Sub	Transactions	Connection	Scripting	Geo	Stream	Cluster（仅Cluster集群支持）
PFADD	PSUBSCRIBE	DISCARD	AUTH	EVAL	GEOADD	XACK	ASKING
PFCOUNT	PUBLISH	EXEC	CLIENT CACHING	EVAL_RO	GEODIST	XADD	CLUSTER COUNTKEYSINSLOT
PFDEBUG	PUBSUB CHANNELS	MULTI	CLIENT GETNAME	EVALSHA	GEOHASH	XAUTOCLAIM	CLUSTER FAILOVER
PFMERGE	PUBSUB NUMPAT	UNWATCH	CLIENT GETREDIR	EVALSHA_RO	GEOPOS	XCLAIM	CLUSTER GETKEYSINSLOT
PFSELFTEST	PUBSUB NUMSUB	WATCH	CLIENT ID	FCALL	GEORADIUS	XDEL	CLUSTER INFO
-	PUBSUB SHARDCHANNELS	-	CLIENT INFO	FCALL_RO	GEORADIUS_RO	XGROUP CREATE	CLUSTER KEYSLOT
-	PUBSUB SHARDNUMSUB	-	CLIENT KILL	FUNCTION DELETE	GEORADIUSBYMEMBER	XGROUP CREATECONSUMER	CLUSTER LINKS
-	PUNSUBSCRIBE	-	CLIENT LIST	FUNCTION DUMP	GEORADIUSBYMEMBER_RO	XGROUP DELCONSUMER	CLUSTER MYID

HyperLoglog	Pub/Sub	Transactions	Connection	Scripting	Geo	Stream	Cluster（仅Cluster集群支持）
-	SPUBLISH	-	CLIENT NO-EVICT	FUNCTION FLUSH	GEOSEARCH	XGROUP DESTROY	CLUSTER MYSHARDID
-	SSUBSCRIBE	-	CLIENT NO-TOUCH	FUNCTION KILL	GEOSEARCHSTORE	XGROUP SETID	CLUSTER NODES
-	SUBSCRIBE	-	CLIENT PAUSE	FUNCTION LIST	-	XINFO CONSUMERS	CLUSTER REPLICAS
-	SUNSUBSCRIBE	-	CLIENT REPLY	FUNCTION LOAD	-	XINFO GROUPS	CLUSTER SHARDS
-	UNSUBSCRIBE	-	CLIENT SETINFO	FUNCTION RESTORE	-	XINFO STREAM	CLUSTER SLAVES
-	-	-	CLIENT SETNAME	FUNCTION STATS	-	XLEN	CLUSTER SLOTS
-	-	-	CLIENT TRACKING	SCRIPT DEBUG	-	XPENDING	READONLY
-	-	-	CLIENT TRACKINGINFO	SCRIPT EXISTS	-	XRANGE	READWRITE
-	-	-	CLIENT UNBLOCK	SCRIPT FLUSH	-	XREAD	-
-	-	-	CLIENT UNPAUSE	SCRIPT KILL	-	XREADGROUP	-
-	-	-	ECHO	SCRIPT LOAD	-	XREVRANGE	-
-	-	-	HELLO	-	-	XSETID	-

HyperLoglog	Pub/Sub	Transactions	Connection	Scripting	Geo	Stream	Cluster（仅Cluster集群支持）
-	-	-	PING	-	-	XTRIM	-
-	-	-	QUIT	-	-	-	-
-	-	-	RESET	-	-	-	-
-	-	-	SELECT（Cluster集群实例不支持）	-	-	-	-

Redis 7.0 禁用的命令

表 1-23 Redis 7.0 单机/主备/Cluster 集群实例的禁用命令

Generic (Key)	Server	Cluster
MIGRATE	ACL DELUSER	CLUSTER ADDSLOTS
-	ACL LOAD	CLUSTER ADDSLOTSRANGE
-	ACL SAVE	CLUSTER BUMPEPOCH
-	ACL SETUSER	CLUSTER COUNT-FAILURE-REPORTS
-	BGREWRITEAOF	CLUSTER DELSLOTS
-	BGSAVE	CLUSTER DELSLOTSRANGE
-	CONFIG RESETSTAT	CLUSTER FLUSHSLOTS
-	CONFIG SET	CLUSTER FORGET
-	CONFIG REWRITE	CLUSTER MEET
-	MODULE LIST	CLUSTER REPLICATE
-	MODULE LOAD	CLUSTER RESET
-	MODULE LOADEX	CLUSTER SAVECONFIG
-	MODULE UNLOAD	CLUSTER SET-CONFIG-EPOCH
-	PSYNC	CLUSTER SETSLOT

Generic (Key)	Server	Cluster
-	REPLICAOF	-
-	SAVE	-
-	SHUTDOWN	-
-	SLAVEOF	-
-	SYNC	-
-	DEBUG	-

1.5.5 Web CLI 命令

本章节主要介绍DCS管理控制台Web CLI工具的命令兼容性，列举支持和禁用的命令列表，命令的具体详细语法，请前往Redis官方网站查看。

当前仅Redis 4.0及以上版本支持Web CLI功能。

说明

- 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。

Web CLI 支持的命令

以下列出了通过Web CLI连接Redis实例时支持的命令。

表 1-24 Web CLI 支持命令清单 1

Keys	String	List	Set	Sorted Set	Server
DEL	APPEND	RPUSH	SADD	ZADD	FLUSHALL
OBJECT	BITCOUNT	RPUSHX	SCARD	ZCARD	FLUSHDB
EXISTS	BITOP	BRPOPLRUSH	SDIFF	ZCOUNT	DBSIZE
EXPIRE	BITPOS	LINDEX	SDIFFSTORE	ZINCRBY	TIME
MOVE	DECR	LINSERT	SINTER	ZRANGE	INFO
PERSIST	DECRBY	LLEN	SINTERSTORE	ZRANGEBYSCORE	CLIENT KILL
PTTL	GET	LPOP	SISMEMBER	ZRANK	CLIENT LIST

Keys	String	List	Set	Sorted Set	Server
RANDOMKEY	GETRANGE	LPUSHX	SMEMBERS	ZREMRANGEBYRANK	CLIENT GETNAME
RENAME	GETSET	LRANGE	SMOVE	ZREMRANGEBYCORE	CLIENT SETNAME
RENAMENX	INCR	LREM	SPOP	ZREVRANGE	CONFIG GET
SCAN	INCRBY	LSET	SRANDMEMBER	ZREVRANGEBYSCORE	SLOWLOG
SORT	INCRBYFLOAT	LTRIM	SREM	ZREVRANK	ROLE
TTL	MGET	RPOP	SUNION	ZSCORE	SWAPDB
TYPE	MSET	RPOPLPU	SUNIONSTORE	ZUNIONSTORE	MEMORY
-	MSETNX	RPOPLPUSH	SSCAN	ZINTERSTORE	-
-	PSETEX	-	-	ZSCAN	-
-	SET	-	-	ZRANGEBYLEX	-
-	SETBIT	-	-	ZLEXCOUNT	-
-	SETEX	-	-	-	-
-	SETNX	-	-	-	-
-	SETRANGE	-	-	-	-
-	STRLEN	-	-	-	-
-	BITFIELD	-	-	-	-

表 1-25 Web CLI 支持命令清单 2

Hash	HyperLoglog	Connection	Scripting	Geo	Pub/Sub
HDEL	PFADD	AUTH	EVAL	GEOADD	UNSUBSCRIBE
HEXISTS	PFCOUNT	ECHO	EVALSHA	GEOHASH	PUBLISH
HGET	PFMERGE	PING	SCRIPT EXISTS	GEOPOS	PUBSUB

Hash	HyperLoglog	Connection	Scripting	Geo	Pub/Sub
HGETALL	-	QUIT	SCRIPT FLUSH	GEODIST	PUNSUBSCRIBE
HINCRBY	-	-	SCRIPT KILL	GEORADIUS	-
HINCRBYFLOAT	-	-	SCRIPT LOAD	GEORADIUSBYMEMBER	-
HKEYS	-	-	-	-	-
HMGET	-	-	-	-	-
HMSET	-	-	-	-	-
HSET	-	-	-	-	-
HSETNX	-	-	-	-	-
HVALS	-	-	-	-	-
HSCAN	-	-	-	-	-
HSTRLEN	-	-	-	-	-

Web CLI 禁用的命令

以下列出了通过Web CLI连接Redis实例时禁用的命令。

表 1-26 通过 Web CLI 禁用的命令 1

Keys	Server	Transactions	Cluster
MIGRATE	SLAVEOF	UNWATCH	CLUSTER MEET
WAIT	SHUTDOWN	REPLICAOF	CLUSTER FLUSHSLOTS
DUMP	DEBUG相关类	DISCARD	CLUSTER ADDSLOTS
RESTORE	CONFIG SET	EXEC	CLUSTER DELSLOTS
WAITAOF	CONFIG REWRITE	MULTI	CLUSTER SETSLOT
-	CONFIG RESETSTAT	WATCH	CLUSTER BUMPEPOCH
-	SAVE	-	CLUSTER SAVECONFIG
-	BGSAVE	-	CLUSTER FORGET
-	BGREWRITEAOF	-	CLUSTER REPLICATE
-	COMMAND	-	CLUSTER COUNT-FAILURE-REPORTS

Keys	Server	Transactions	Cluster
-	KEYS	-	CLUSTER FAILOVER
-	MONITOR	-	CLUSTER SET-CONFIG-EPOCH
-	SYNC	-	CLUSTER RESET
-	PSYNC	-	-
-	ACL	-	-
-	MODULE	-	-
-	COMMAND COUNT	-	-
-	COMMAND DOCS	-	-
-	COMMAND GETKEYS	-	-
-	COMMAND GETKEYSANDFLAGS	-	-
-	COMMAND INFO	-	-
-	COMMAND LIST	-	-
-	RESTORE-ASKING	-	-

表 1-27 通过 Web CLI 禁用的命令 2

List	Connection	Sorted Set	Pub/Sub
BLPOP	SELECT	BZPOPMAX	PSUBSCRIBE
BRPOP	-	BZPOPMIN	SUBSCRIBE
BLMOVE	-	BZMPOP	-
BRPOPLPUSH	-	ZREM	-
BLMPOP	-	ZUNION	-

1.5.6 实例受限使用命令

Cluster集群实例支持多个key，但不支持跨slot访问的Redis命令，如表1-28所示。

Proxy集群实例支持多Key的命令中，部分命令不支持跨slot访问，请参考表1-29。

表 1-28 Cluster 集群实例受限使用的 Redis 命令

命令类型	命令描述
Set（集合）	
SINTER	返回一个集合的全部成员，该集合是所有给定集合的交集
SINTERSTORE	类似SINTER，但结果保存到destination集合
SUNION	返回一个集合的全部成员，该集合是所有给定集合的并集
SUNIONSTORE	和SUNION类似，但它将结果保存到destination集合
SDIFF	返回一个集合的全部成员，该集合是所有给定集合之间的差集
SDIFFSTORE	和SDIFF类似，但它将结果保存到destination集合
SMOVE	将member元素从source集合移动到destination集合
SortedSet（有序集合）	
ZUNIONSTORE	计算给定的一个或多个有序集的并集
ZINTERSTORE	计算给定的一个或多个有序集的交集
HyperLogLog	
PFCOUNT	返回储存在给定键（或多个键）的HyperLogLog的近似基数
PFMERGE	将多个HyperLogLog合并（merge）为一个HyperLogLog
Keys	
RENAME	将key改名
RENAMENX	将key改名，新key必须是之前不存在的
BITOP	对一个或多个保存二进制位的字符串key进行位元操作，并将结果保存到destkey上
RPOPLPUSH	返回并移除存储在source的列表的最后一个元素（列表尾部元素）， 并把该元素放入存储在destination的列表的第一个元素位置（列表头部）
String（字符串）	
MSETNX	同时设置一个或多个key-value对

说明

当用户执行比较耗时的命令（如flushall）时，可能会导致缓存实例在命令执行期间对外不响应用户的其它命令，造成状态监控失效，此时Console上缓存实例的状态会变成异常，命令执行结束后，实例状态会恢复正常。

表 1-29 Proxy 集群多 Key 命令说明

类型	命令
支持跨slot的多Key命令	DEL、MGET、MSET、EXISTS、SUNION、SINTER、SDIFF、SUNIONSTORE、SINTERSTORE、SDIFFSTORE、ZUNIONSTORE、ZINTERSTORE
不支持跨slot的多Key命令	SMOVE、SORT、BITOP、MSETNX、RENAME、RENAMENX、BLPOP、BRPOP、RPOPLPUSH、BRPOPLPUSH、PFMERGE、PFCOUNT、BLMOVE、COPY、GEOSEARCHSTORE、LMOVE、ZRANGESTORE、XREAD、XREADGROUP

1.5.7 部分命令使用限制

本章节主要介绍部分Redis命令使用时的限制。

Key 相关命令使用限制

使用KEYS命令时，若缓存数据量较大，可能会较长时间阻塞其它业务命令操作，甚至可能过高地占用额外内存。因此使用KEYS命令时请尽量描述精确的pattern、不要使用“keys *”进行全通配。建议尽量避免在生产环境使用，否则会影响服务的健康运行。

Server 相关命令使用限制

- 当用户执行比较耗时的命令（如flushall）时，可能会导致缓存实例在命令执行期间对外不响应用户的其它命令，造成状态监控失效，此时Console上缓存实例的状态会变成异常，命令执行结束后，实例状态会恢复正常。
- 使用FLUSHDB、FLUSHALL命令时，若缓存数据量较大，可能会较长时间阻塞其它业务命令操作。
- 不建议在业务高峰使用MONITOR命令，在高并发的场景下执行MONITOR命令可能会影响实例性能，增加时延。

EVAL 和 EVALSHA 相关命令使用限制

- 使用EVAL和EVALSHA命令时，命令参数中必须带有至少1个key。否则客户端会提示“ERR eval/evalsha numkeys must be bigger than zero in redis cluster mode”的错误。
- 使用EVAL和EVALSHA命令时，DCS Redis集群实例使用第一个key来计算slot，用户代码需要保证操作的key是在同一个slot，具体请参考**https://redis.io/commands**
- 使用EVAL命令时：
  - 建议使用前先了解Redis的lua脚本特性，具体可参考**https://redis.io/commands/eval**。
  - lua脚本的执行超时时间为5秒钟，建议不要在lua脚本中使用比较耗时的代码，比如长时间的sleep、大的循环等语句。
  - 调用lua脚本时，建议不要使用随机函数去指定key，否则在主备节点上执行结果不一致，从而导致主备节点数据不一致。

Lua 脚本调试命令

Proxy集群和读写分离实例在执行Lua脚本调试命令时，仅支持异步非阻塞--ldb模式，不支持同步阻塞--ldb-sync-mode。且每个Proxy节点默认限制并发2个。其他Redis实例类型无此限制。

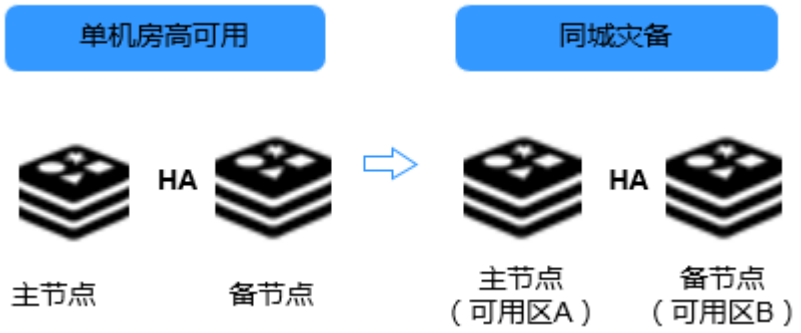
其他限制

单个Redis命令处理时长限制为15秒左右，超过15秒未处理完，会导致客户的其它业务失败，因此内部会触发主从倒换。

1.6 容灾和多活策略

DCS缓存实例都存储着大量关键数据，不论是作为数据库前端缓存，还是作为数据存储引擎，数据的可靠性与服务的连续可用性是DCS服务设计上为客户考虑的核心因素，下图展示了DCS在数据和服务方面的容灾架构设计演进。

图 1-5 DCS 灾备架构演进



根据对数据与服务的不同可靠性要求，您可以选择将缓存实例部署在单可用区内（单机房），或者跨可用区（同城灾备）。

实例单可用区高可用

同一机房即单可用区。单可用区灾备策略主要包括进程/服务高可用，数据持久化到磁盘，以及实例节点间热备三种不同层次。

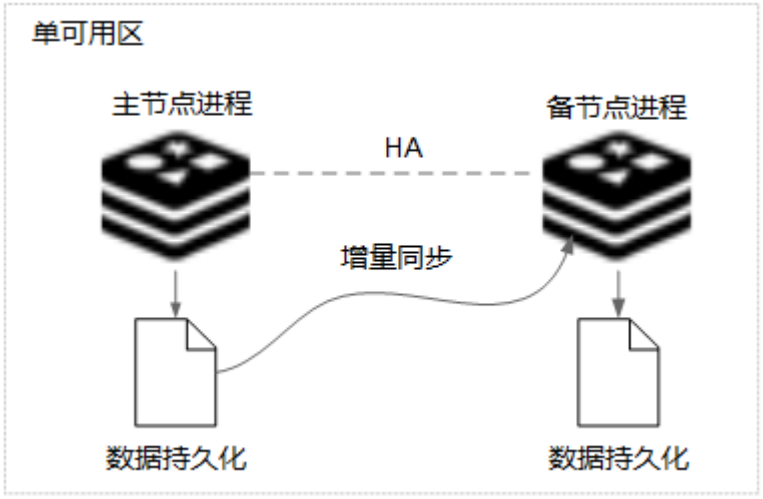
在单可用区内，单机实例通过进程守护的方式确保服务高可用，当DCS监测到缓存实例进程故障，马上拉起一个新的进程继续提供服务。

图 1-6 单可用区内单机实例高可用

除单机实例外，其他实例类型默认都支持数据持久化，数据持久化到主节点磁盘外，还会增量同步到备节点，同时备节点也会持久化一份数据。实现了节点热备和持久化文件多个备份。

主备实例的主备节点进程，以及集群实例每个分片内主备节点进程的数据同步和持久化方式如下图所示。

图 1-7 单可用区内主备实例高可用



实例跨可用区灾备

主备与集群版本的缓存实例支持将主备副本部署在不同的可用区内（即不同的物理机房）。不同可用区的电力、网络相互隔离，当主节点所在的机房因为电力或者网络出现故障，备节点将接管服务，客户端与备节点正常建立连接以及读写数据。

图 1-8 主备实例跨可用区示意图

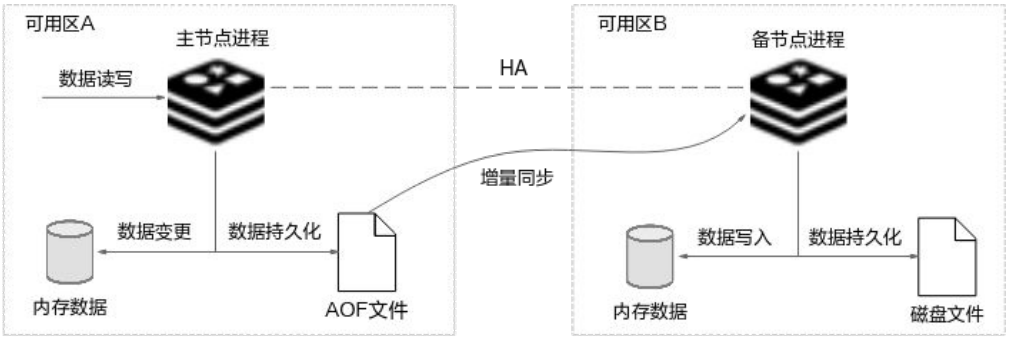


图 1-9 Proxy 集群实例跨可用区示意图

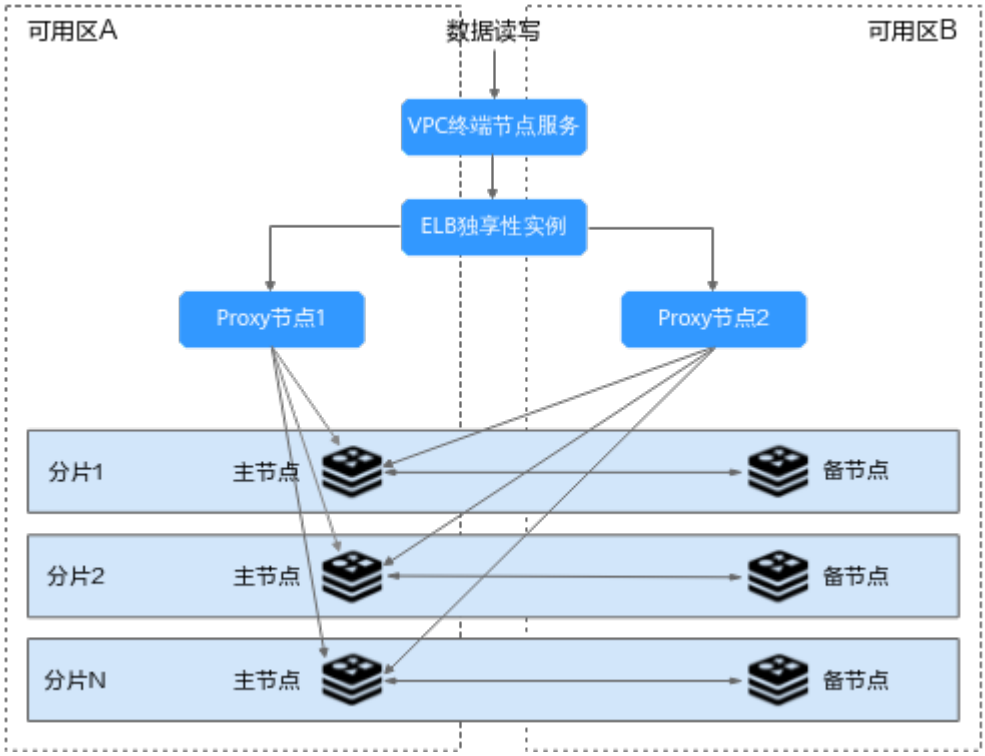
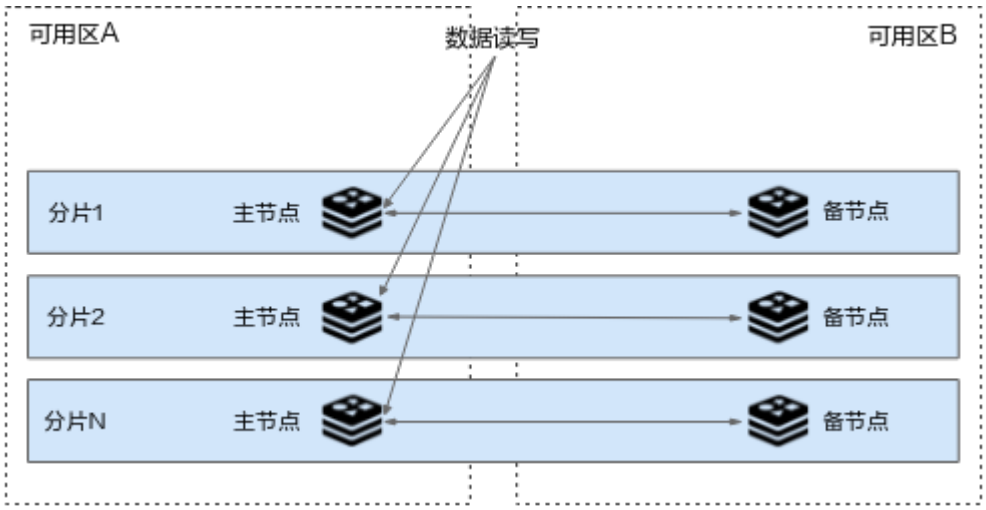


图 1-10 Cluster 集群实例跨可用区示意图



对于同城容灾，只需要在创建主备/集群实例时，选择与主可用区不同的备可用区。

图 1-11 创建实例选择不同可用区部署

说明

- 您的应用也可以部署为跨可用区的HA模式，这样不仅能保证数据高可靠，还能在机房遇到断电或网络故障时，服务继续可用。
- 在可用区故障期间，跨可用区实例的密码修改、命令重命名、变更规格等功能不可用。

1.7 Redis 开源版本特性说明

DCS Redis完全兼容开源版Redis，本文介绍Redis各版本的新特性与兼容性变更。

Redis 开源版 7.2

新特性

关于Redis开源版7.2的新特性，请参见：**valkey 7.2 release note**、**redis 7.2 release note**、**redis 7.0 release note**。

- 引入了多AOF文件（Multi-Part AOF），将AOF文件拆分为多个小文件，提高了持久化的可靠性和恢复效率。

- 引入了Redis Functions，允许用户通过Lua脚本定义和存储函数，并在后续调用中重复使用。
- 引入了新的命令，如ZMPOP、BZMPOP等。
- Redis Cluster中引入了分片发布/订阅（Sharded Pub/Sub），允许在集群模式下更高效地处理发布/订阅消息。

兼容性

- 关于社区演进的Breaking change请参见：**valkey 7.2 release note**、**redis 7.2 release note**、**redis 7.0 release note**。
- 不支持账号管理。

Redis 开源版 6.2

新特性

关于Redis 6.2的新特性请参见：**redis 6.2 release note**、**redis 6.0 release note**。

- 增加了对SSL/TLS加密通信的支持，确保客户端与服务器之间的数据传输安全。
- 引入了新的RESP3协议，提供了更丰富的数据类型和语义，同时保持与RESP2的兼容性。
- 引入了新的命令，如ZRANDMEMBER、COPY等。

兼容性

- 关于社区演进的Breaking change请参见：**redis 6.2 release note**、**redis 6.0 release note**。
- 支持SSL。
- 兼容RESP2、RESP3。

Redis 开源版 5.0

新特性

关于Redis 5.0的新特性请参见：**redis 5.0 release note**。

- 引入了新的Stream数据类型，用于处理消息流和日志数据。
- 改进了RDB文件的格式，减少了持久化文件的大小和加载时间。
- 引入了多个新的命令和选项，如XCLAIM、XINFO、XTRIM等。

兼容性

关于社区演进的Breaking change请参见：**redis 5.0 release note**。

Redis 开源版 4.0

新特性

关于Redis 4.0的新特性请参见：**redis 4.0 release note**。

- 引入了模块系统，允许开发者通过编写模块来扩展Redis的功能，而无需修改Redis的源代码。
- 改进了主从复制协议，支持部分重新同步，即使在主从服务器之间的连接断开后，也能更高效地进行数据同步。

- 新增了LFU缓存淘汰策略，允许Redis根据键的使用频率来进行淘汰，而不仅是基于最近使用时间（LRU）。
- 引入混合持久化模式，允许在AOF文件中嵌入RDB快照，从而在保证数据安全性的同时，提高了恢复速度。
- 引入了多个新的命令和选项，如MEMORY、SWAPDB等。

兼容性

- 关于社区演进的Breaking change请参见：**redis 4.0 release note**。
- 不支持SSL。
- 支持RESP2。

1.8 与开源服务的差异

DCS提供单机、主备、集群等丰富的实例类型，满足用户高读写性能及快速数据访问的业务诉求。支持丰富的实例管理操作，帮助用户省去运维烦恼。用户可以聚焦于业务逻辑本身，而无需过多考虑部署、监控、扩容、安全、故障恢复等方面的问题。

DCS基于开源Redis、Memcached向用户提供一定程度定制化的缓存服务，因此，除了拥有开源服务缓存数据库的优秀特性，DCS提供更多实用功能。

与开源 Redis 差异

表 1-30 DCS 与自建开源 Redis 的差异说明

比较项	开源Redis	DCS Redis
服务搭建	从自行准备服务器资源到Redis搭建，需要0.5~2天。	• Redis 3.0版本5~15分钟完成创建。 • Redis 4.0及以上版本，采用容器化部署，8秒完成创建。
版本	-	深度参与开源社区，及时支持最新Redis的版本。目前支持Redis 4.0、Redis 5.0、Redis 6.0和Redis 7.0版本。
安全	自行保证网络与服务器的安全。	• 使用云上虚拟私有云与安全组，确保网络安全。 • 主备与集群多副本、定时备份，确保数据高可靠。
性能	-	单节点达10万QPS（Query Per Second）。

比较项	开源Redis	DCS Redis
监控	提供简单的信息统计。	提供30余项监控指标，并支持用户自定义监控阈值和告警策略。 • 指标类型丰富 – 常见的外部业务监控和统计：命令数、并发操作数、连接数、客户端数、拒绝连接数等。 – 常见的资源占用监控和统计：cpu占用率、物理内存占用、网络输入/输出流量等。 – 常见的关键内部监控和统计：键个数、键过期个数、容量占用量、pubsub通道个数、pubsub模式个数、keyspace命中、keyspace错过。 • 自定义监控阈值及告警 提供基于各项监控制定阈值告警，支持客户自定义，便于及时发现业务异常。
备份恢复	支持。	• 提供定时与手动备份数据能力，支持备份文件下载到本地。 • 支持控制台一键恢复数据。
可视化维护缓存参数	不具备，需要自行开发。	• web控制台可视化维护。 • 可在线修改配置参数。 • 支持在web控制台连接并操作数据。
可扩展性	需要中断服务。首先为服务器调整运行内存，然后调整Redis内存配置并重启操作系统与服务。	• 提供不中断服务的在线扩容或缩容能力。 • 规格可根据实际需要，在DCS支持的规格范围内进行扩容或者缩容。

1.9 约束与限制

网络限制

云服务采用虚拟私有云（VPC）管理各服务的网络安全。

- 对于DCS缓存实例，客户端需要部署在与DCS缓存实例相同VPC的弹性云服务器上。
- Redis 4.0/5.0/6.0/7.0版本实例，需要将客户端所在弹性云服务器的IP地址加入实例白名单，允许访问。白名单配置请参考**管理实例白名单**。

其他说明

DCS提供了基于RESTful的API接口，在进行接口调用时，需要先获取DCS的地区和终端节点信息，具体请参考**地区和终端节点信息**。

1.10 基本概念

缓存实例

DCS向用户提供服务的最小资源单位。

缓存实例拥有Redis、Memcached两种存储引擎，每种引擎有单机、主备、集群等不同实例类型。不同实例类型含有多种规格。

详情参考：**产品规格介绍**，**实例类型介绍**

项目

项目（Project）用于将OpenStack的资源（计算资源、存储资源和网络资源）进行分组和隔离。项目可以是一个部门或者一个项目组。一个账户中可以创建多个项目。

免密访问

Redis和Memcached两种引擎，可以不设置密码，在VPC内直接连接实例进行数据读写。由于不涉及密码鉴权，数据读写延时会更低。

对于实例数据敏感性一般的业务，您可以对实例开启免密访问。

具体使用请参考：**开启实例免密访问**

跨可用区部署

将主备实例部署在不同的AZ（可用区域）内，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。

在创建Redis或者Memcached主备或集群实例时，可以为节点选择可用区。

分片

也叫条带，指Redis集群的一个管理组，对应一个redis-server进程。一个Redis集群由若干条带组成，每个条带负责若干个slot（槽），数据分布式存储在slot中。Redis集群通过条带化分区，实现超大容量存储以及并发连接数提升。

每个集群实例由多个分片组成，每个分片默认为一个双副本的主备实例。分片数等于实例中主节点的个数。

副本

指缓存实例的节点。单副本表示实例没有备节点，双副本表示实例有备节点（一个主节点，一个备节点），例如主备实例默认为双副本，当主备实例的副本数设置为3时，表示该实例有1个主节点，2个备节点。单机实例，只有一个节点。

1.11 权限管理

如果您需要对云服务平台上创建的DCS资源，给企业中的员工设置不同的访问权限，以达到不同员工之间的权限隔离，您可以使用统一身份认证服务（Identity and Access

Management，简称IAM）进行精细的权限管理。该服务提供用户身份认证、权限分配、访问控制等功能，可以帮助您安全的控制云服务资源的访问。

通过IAM，您可以在云服务账号中给员工创建IAM用户，并使用策略来控制IAM用户对云服务资源的访问范围。例如您的员工中有负责软件开发的人员，您希望他们拥有DCS的使用权限，但是不希望他们拥有删除DCS实例等高危操作的权限，那么您可以使用IAM为开发人员创建用户，通过授予仅能使用DCS，但是不允许删除DCS实例的权限策略，控制他们对DCS资源的使用范围。

如果账号已经能满足您的要求，不需要创建独立的IAM用户进行权限管理，您可以跳过本章节，不影响您使用DCS服务的其它功能。

DCS 权限

默认情况下，账号管理员创建的IAM用户没有任何权限，需要将其加入用户组，并给用户组授予策略或角色，才能使得用户组中的用户获得对应的权限，这一过程称为授权。授权后，用户就可以基于被授予的权限对云服务进行操作。

DCS部署时通过物理区域划分，为项目级服务。授权时，“作用范围”需要选择“区域级项目”，然后在指定区域对应的项目中设置相关权限，并且该权限仅对此项目生效；如果在“所有项目”中设置权限，则该权限在所有区域项目中都生效。访问DCS时，需要先切换至授权区域。

权限根据授权精细程度分为角色和策略。

- 角色：IAM最初提供的一种根据用户的工作职能定义权限的粗粒度授权机制。该机制以服务为粒度，提供有限的服务相关角色用于授权。由于云服务平台各服务之间存在业务依赖关系，因此给用户授予角色时，可能需要一并授予依赖的其他角色，才能正确完成业务。角色并不能满足用户对精细化授权的要求，无法完全达到企业对权限最小化的安全管控要求。
- 策略：IAM最新提供的一种细粒度授权的能力，可以精确到具体服务的操作、资源以及请求条件等。基于策略的授权是一种更加灵活的授权方式，能够满足企业对权限最小化的安全管控要求。例如：针对DCS服务，账号管理员能够控制IAM用户仅能对DCS实例进行指定的管理操作。权限策略以API接口为粒度进行权限拆分，权限的最小粒度为API授权项（action），DCS支持的API授权项请参见《分布式缓存服务 API参考》中的“权限策略和授权项”章节。

如表1-31所示，包括了DCS的所有系统权限。

表 1-31 DCS 系统策略

系统角色/策略名称	描述	类别	依赖关系
DCS FullAccess	分布式缓存服务所有权限，拥有该权限的用户可以操作所有分布式缓存服务的功能。	系统策略	无
DCS UserAccess	分布式缓存服务普通用户权限（无实例创建、修改、删除、扩容和缩容的权限）。	系统策略	无
DCS ReadOnlyAccess	分布式缓存服务的只读权限，拥有该权限的用户仅能查看分布式缓存服务数据。	系统策略	无

系统角色/策略名称	描述	类别	依赖关系
DCS Administrator	分布式缓存服务管理员权限，拥有该权限的用户可以操作所有分布式缓存服务的功能。	系统角色	依赖Server Administrator和Tenant Guest角色，在同项目中勾选依赖的角色。
DCS AgencyAccess	分布式缓存服务申请创建租户委托时需要授权的操作权限。该权限为租户委托权限，用于租户在需要时委托DCS服务对租户资源做以下相关操作，与授权用户操作无关。 • 查询子网 • 查询子网列表 • 查询端口 • 查询端口列表 • 更新端口 • 创建端口	系统策略	无

说明

由于DCS UserAccess策略和DCS FullAccess策略存在差异，如果您同时配置了这两个系统策略，由于DCS UserAccess策略存在Deny，根据Deny优先原则，您无法执行实例创建、修改、删除、扩容和缩容操作。

表1-32列出了DCS常用操作与系统策略的授权关系，您可以参照该表选择合适的系统策略。

表 1-32 常用操作与系统策略的关系

操作	DCS FullAccess	DCS UserAccess	DCS ReadOnlyAccess
修改实例配置参数	√	√	×
删除实例后台任务	√	√	×
Web CLI	√	√	×
修改实例运行状态	√	√	×
缓存实例扩容	√	×	×

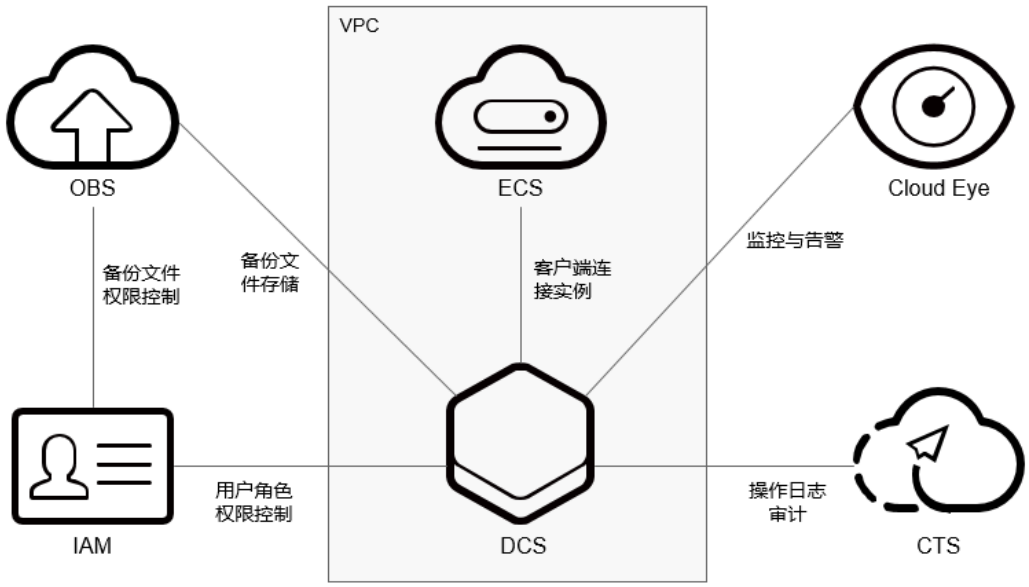
操作	DCS FullAccess	DCS UserAccess	DCS ReadOnlyAccess
修改实例访问密码	√	√	×
修改缓存实例	√	×	×
实例主备倒换	√	√	×
备份实例数据	√	√	×
分析实例的大key或者热key	√	√	×
创建缓存实例	√	×	×
删除实例数据备份文件	√	√	×
升级实例版本	√	√	×
恢复实例数据	√	√	×
重置实例访问密码	√	√	×
迁移实例数据	√	√	×
下载备份实例数据	√	√	×
删除缓存实例	√	×	×
查询实例配置参数	√	√	√
查询实例数据恢复日志	√	√	√
查询实例数据备份日志	√	√	√
查询缓存实例信息	√	√	√
查询实例后台任务	√	√	√
查询实例升级信息	√	√	√
查询实例列表	√	√	√
查看实例性能监控	√	√	√
修改参数模板	√	√	×

操作	DCS FullAccess	DCS UserAccess	DCS ReadOnlyAccess
删除参数模板	√	√	×
创建参数模板	√	√	×
参数模板列表	√	√	√
查询参数模板	√	√	√

1.12 与其他服务的关系

DCS在使用时与其他服务配合使用，本节简单介绍虚拟私有云、弹性云服务器、统一身份认证服务、云监控服务、云审计服务以及对象存储服务。

图 1-12 DCS 缓存服务与其他服务的关系



虚拟私有云

虚拟私有云（Virtual Private Cloud，简称VPC）是用户在云上申请的隔离的、私密的虚拟网络环境。用户可以自由配置VPC内的IP地址段、子网、安全组等子服务。

分布式缓存服务运行于虚拟私有云，由虚拟私有云协助管理IP和带宽。虚拟私有云还具备安全组访问控制功能，通过绑定安全组并设置访问规则，可以增强访问分布式缓存服务的安全性。

弹性云服务器

弹性云服务器（Elastic Cloud Server，简称ECS）是一种可随时自助获取、可弹性伸缩的云服务器，帮助用户打造可靠、安全、灵活、高效的应用环境。

成功申请分布式缓存服务后，您可以通过弹性云服务器创建的弹性云主机，连接和使用分布式缓存实例。

统一身份认证服务

统一身份认证（Identity and Access Management，简称IAM）是系统的身份管理服务，包括用户身份认证、权限分配、访问控制等功能。

通过统一身份认证服务，实现对分布式缓存服务的访问控制。

云监控服务

云监控服务（Cloud Eye）是云上提供的安全、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。

云审计服务

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。

对象存储服务

对象存储服务（Object Storage Service，简称OBS）是一个基于对象的海量存储服务，为客户提供海量、安全、高可靠、低成本的数据存储能力，包括：创建、修改、删除桶，上传、下载、删除对象等。

DCS使用OBS存储实例数据备份文件。

2 DCS 权限管理

2.1 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。

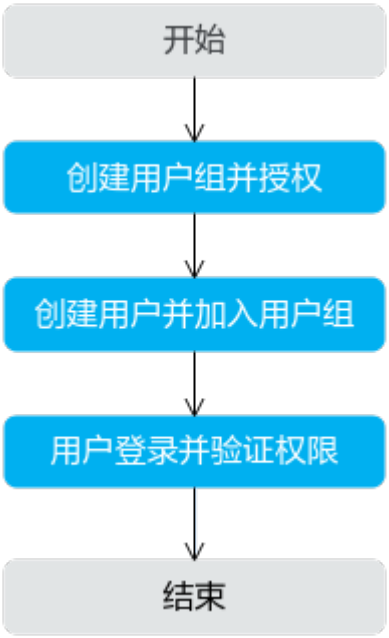
本章节以创建用户并授予“DCS ReadOnlyAccess”权限为例，为您介绍对用户授权的方法，操作流程如**图2-1**所示。

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**权限管理**。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**系统权限**。

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。

2.2 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《分布式缓存服务 API参考》中的“权限策略和授权项”章节。

目前云服务平台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"
                  ]
            }
      ]
}
```

3 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，请参见本手册的操作指导。

  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**查看监控指标**。

  如果您开启了云审计服务，系统会将对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**查看云审计日志**。

- 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。

使用 DCS 缓存实例

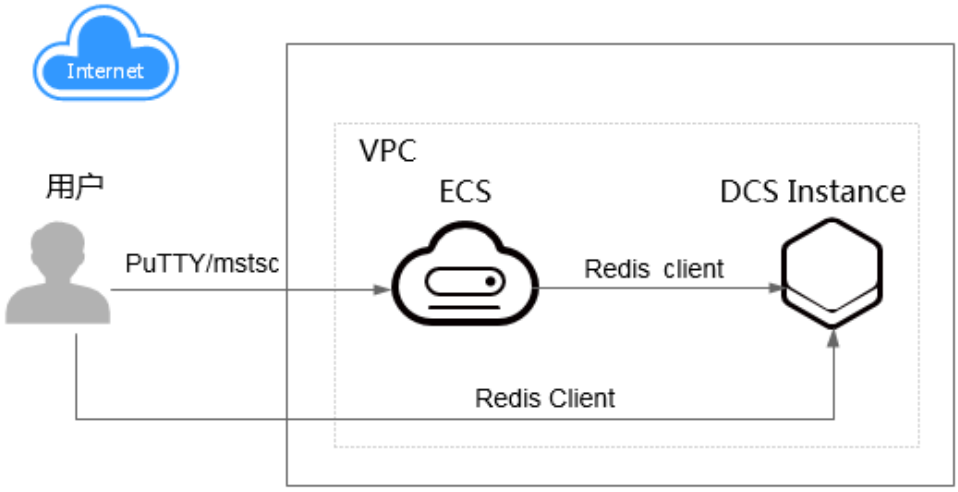
您创建DCS缓存实例后，可参考**连接实例**，了解如何连接缓存实例。兼容开源Redis/Memcached协议的客户端都可以连接DCS的Redis/Memcached缓存实例，客户端连接成功后，您就可以开始享受DCS带来的高效数据读写。

须知

DCS本身不涉及用户敏感信息。使用DCS处理数据的目的、范围、处理方式、时限等请遵从当地适用的法律法规。DCS本身不建议传输和存储敏感数据，如果传输和存储敏感数据，请自行加密后再传输和存储。

访问缓存实例的方法，如下图所示。

图 3-1 访问 DCS 服务简图



说明

- 客户端所在弹性云服务器（Elastic Cloud Server）建议和缓存实例处于同一虚拟私有云（Virtual Private Cloud）。

4 快速入门

4.1 创建实例

4.1.1 创建前准备

在创建实例之前，请先根据您的实际业务需要，明确实例创建需求，完成以下工作：

1. 确定缓存实例版本。

   不同的Redis版本，特性会不同，可参考**Redis开源版本特性说明**。

2. 确定缓存实例类型，即实例架构。

   确定缓存类型后，需要明确实例架构，当前支持的实例架构有单机、主备和Cluster集群。实例规格特点和架构，可参考**选择实例类型**。

3. 确定实例规格。

   确定实例架构后，需要明确实例规格大小。实例支持的连接数和带宽，可参考**产品规格**。

4. 确定选择的区域以及实例是否跨可用区部署。

   选择的区域，建议选择接近您应用程序的区域，减少网络延时。

   一个区域对应多个可用区（AZ），当前DCS支持将主备实例/集群实例部署在不同的AZ内，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。

   说明

   - 当主备、或者集群Redis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。

5. 确定实例是否配置备份策略。

   当前除单机实例外，其他类型实例都支持配置备份恢复策略。关于备份恢复，可参考**备份与恢复概述**。

4.1.2 准备实例依赖资源

使用DCS服务前，若采用VPC内连接的方式，您需要创建虚拟私有云（Virtual Private Cloud，以下简称VPC），并且配置安全组与子网。VPC为DCS服务提供一个隔离的、您可以自主配置和管理的虚拟网络环境，提升资源的安全性，简化网络部署。

如果您已有以下依赖资源 ，可重复使用，不需要多次创建。

创建 VPC 和子网

步骤1 登录管理控制台。

步骤2 在管理控制台左上角单击 ，选择区域和项目。

步骤3 单击页面上方的“服务列表”，选择“网络 > 虚拟私有云”。

步骤4 单击“申请虚拟私有云”。

步骤5 根据界面提示创建虚拟私有云。如无特殊需求，界面参数均可保持默认。

关于创建VPC的详细信息可以参考《虚拟私有云用户指南》中“虚拟私有云和子网 > 虚拟私有云 > 创建虚拟私有云和子网”章节 。

创建虚拟私有云时，会同时创建子网，若需要额外创建子网，请参考**步骤6**和**步骤7**。

说明

- 在创建虚拟私有云时，配置参数“网段”，即为VPC的地址范围，若配置该参数，则VPC内的子网地址必须在VPC的地址范围内。
- 若创建虚拟私有云用于发放DCS实例时，可不用配置虚拟私有云的“网段”。

步骤6 在左侧导航栏选择“虚拟私有云 > 子网”，进入子网页面。

步骤7 单击“创建子网”。根据界面提示创建子网。如无特殊需求，界面参数均可保持默认。

关于创建子网的详细信息可以参考《虚拟私有云用户指南》中“虚拟私有云和子网 > 子网 ”章节。

----结束

4.1.3 创建 Redis 实例

您可以根据业务需要创建相应计算能力和存储空间的Redis实例。

前提条件

已准备好**实例依赖资源**。

创建 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 ，选择区域和项目。

步骤3 单击“购买缓存实例”，进入实例创建页面。

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 根据**创建前准备**，设置以下基本信息。

1. 在“缓存类型”区域，选择缓存实例类型，默认缓存实例类型为“Redis”。
2. “产品类型”，目前支持的产品类型为“基础版”。
3. “CPU架构”，当前支持“x86计算”。
4. 在“版本号”区域，选择Redis版本。

   当前DCS支持的Redis版本有：4.0、5.0、6.0、7.0。

   说明

   - 不同Redis版本差异请参考**Redis开源版本特性说明**。
   - 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。
   - 客户端连接Redis Cluster集群实例与连接到单机、主备、Proxy集群实例的方式不同，连接示例请参考**连接Redis实例**。
5. 在“实例类型”区域，选择单机、主备或Cluster集群实例类型。
6. 在“可用区”区域，您可根据实际情况选择。

   除单机实例外，其他实例类型的实例除了可以选择主节点的“可用区”，还可以为备节点选择“备可用区”。

   说明

   - 当Redis主备/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。
   - 如果提高访问速度，可选择和应用同一个可用区。
   - 每个region有若干个可用区。当可用区资源不足时，可用区会置灰，此时，请选择另一个可用区。
7. 在“副本数”区域，选择实例副本数，默认为2副本（包含主副本）。

   当选择Redis 4.0及以上版本，且实例类型为主备、Cluster集群时，页面才显示“副本数”。
8. 在“实例规格”区域，选择符合您的规格。

   您的默认配额请以控制台显示为准。

   您如需增加配额，单击规格下方的“申请扩大配额”，申请增加配额。

图 4-1 创建 Redis 实例

步骤7 设置实例网络环境信息。

1. 在“虚拟私有云”区域，选择已经创建好的虚拟私有云、子网。
2. 设置实例IP地址。

   Cluster集群实例仅支持自动分配地址，其他实例类型支持自动分配IP地址或手动分配IP地址，用户选择手动分配IP地址时，可以输入一个在当前子网下可用的IP。

   Redis 4.0及以上版本实例支持自定义端口。自定义端口范围为1~65535；如果未自定义，则使用默认端口6379。
3. 在“安全组”下拉列表，可以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。

   Redis 4.0及以上版本是基于VPCEndpoint，暂不支持安全组，请在实例创建后配置白名单，参考**管理实例白名单**。

步骤8 设置实例的“名称”和“企业项目”。

创建实例时，名称长度不能少于4位字符串。批量创建实例时，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 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**章节。
- “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，表示连接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。

步骤10 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**备份与恢复概述**。

3. 当前暂不支持创建实例时配置“公网访问”。

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. 设置实例的“描述”。

步骤11 设置创建实例的数量。

步骤12 实例信息配置完成后，单击“Next”，进入规格确认页面。

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13 确认实例信息无误后，提交请求。

步骤14 缓存实例创建成功后，您可以在“缓存管理”页面，查看并管理自己的缓存实例。

1. Redis 4.0及以上版本采用容器化部署，秒级完成创建。
2. 缓存实例创建成功后，默认“状态”为“运行中”。

----结束

4.2 连接实例

4.2.1 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。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。

- 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：**DCS实例是否支持跨VPC访问？** 。

4.2.2 使用 redis-cli 连接 Redis 实例

介绍使用同一VPC内弹性云服务器ECS上的redis-Cli连接Redis实例的方法。更多的客户端的使用方法，请参考**https://redis.io/clients**。

说明

- 本文操作步骤涉及实例端口时，统一以默认端口6379为例，如果已自定义端口，请根据实际情况替换。
- **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。
  - Cluster集群连接命令：

    ./redis-cli -h {dcs_instance_address} -p 6379 -a {password} **-c**
  - 单机、主备、Proxy集群连接命令：

    ./redis-cli -h {dcs_instance_address} -p 6379 -a {password}

  具体连接操作， 请查看**步骤3**和**步骤4**。
- 当Redis 6.0单机、主备实例开启了SSL时，需要配置SSL证书路径：

  ./redis-cli -h {dcs_instance_address} -p 6379 -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。

前提条件

- 已成功申请Redis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》。
- 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++
```

操作步骤（Linux 版）

步骤1 查看并获取待连接Redis实例的IP地址和端口。

具体步骤请参见**查看和修改DCS实例信息**。

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

   cd src

   make

   make install

步骤3 连接Redis非Cluster集群实例。

如果是单机/主备实例，请执行以下操作。

./redis-cli -h ${dcs_instance_address} -p 6379 -a ${password}

说明

1. 如果实例为免密实例，连接实例使用命令：./redis-cli -h ${dcs_instance_address} -p 6379
2. 如果实例为非免密实例，连接实例使用命令：./redis-cli -h ${dcs_instance_address} -p 6379 -a ${password}
3. 如果忘记实例访问密码或需要重置密码，可以重置密码，参考**重置缓存实例密码**。

步骤4 连接Redis Cluster集群实例。

如果是Cluster集群实例，请执行以下操作。

1. 执行以下命令连接Redis实例。

   ./redis-cli -h {dcs_instance_address} -p 6379 -a {password} -c

   其中，**{dcs_instance_address}**为Redis实例的IP地址，“6379”为Redis实例的端口，**{password}**为Cluster集群实例的密码，**-c**连接集群节点时使用。IP地址和端口获取见**步骤1**。

   如下所示，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>
   ```
2. 查看Cluster集群节点信息。

   cluster nodes

   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
   0988ae8fd3686074c9afdcce73d7878c81a33ddc 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
```

备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Crc16(key) mod 16384的值决定。

如下所示，数据写入时，根据Crc16(key) mod 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>
```

----结束

操作步骤（Windows 版）

Windows版本的Redis客户端安装包，请单击**这里**下载编译包（非Source code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，“6379”为Redis实例的端口。IP地址和端口获取见**查看和修改DCS实例信息**，请按实际情况修改后执行。

4.2.3 多语言连接

4.2.3.1 Java 客户端

4.2.3.1.1 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Lettuce。因此，如需在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。

约束与限制

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”。
- 查看并获取待连接Redis实例的IP地址和端口。具体步骤请参见**查看和修改DCS实例信息**。

Pom 配置

```
<!-- 引入spring-data-redis组件 -->
<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-data-redis</artifactId>
   <!--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、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、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，并且空闲连接数>最小空闲连接数时直接逐出
        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，并且空闲连接数>最小空闲连接数时直接逐出
        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 4-1 RedisStandaloneConfiguration 参数

参数	默认值	说明
hostName	localhost	连接Redis实例的IP地址。
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。如果实例已开启免密访问，无需输入实例的访问密码。忘记密码或需要重置密码请参见**重置缓存实例密码**。

表 4-2 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Redis实例的密码。如果实例已开启免密访问，无需输入实例的访问密码。忘记密码或需要重置密码请参见**重置缓存实例密码**。

表 4-3 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 && 空闲连接数>最小空闲连接数）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 4-4 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。

  连接池的大小没有固定标准，建议根据业务流量合理配置，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

4.2.3.1.2 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。

约束与限制

Springboot版本不得低于2.3.12.RELEASE，Lettuce版本不得低于**6.3.0.RELEASE**，netty版本要求4.1.100.Final及以上。

前提条件

- 已成功创建Redis实例，且状态为“运行中”。
- 查看并获取待连接Redis实例的IP地址和端口。具体步骤请参见**查看和修改DCS实例信息**。

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>6.3.0.RELEASE</version>
</dependency>

<dependency>
   <groupId>io.netty</groupId>
   <artifactId>netty-transport-native-epoll</artifactId>
   <version>4.1.100.Final</version>
   <classifier>linux-x86_64</classifier>
</dependency>
```

基于 application.properties 配置

- 单机、主备、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、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;
    /**
     *  TCP_KEEPALIVE 配置参数:
     *  两次 keepalive 间的时间间隔 = TCP_KEEPALIVE_TIME = 30
     *  连接空闲多久开始 keepalive = TCP_KEEPALIVE_TIME/3 = 10
     *  keepalive 几次之后断开连接 = TCP_KEEPALIVE_COUNT = 3
     */
    private static final int TCP_KEEPALIVE_TIME = 30;

    /**
     * TCP_USER_TIMEOUT 连接空闲限制时间，解决Lettuce长时间超时问题。
     * refer: https://github.com/lettuce-io/lettuce-core/issues/2082
     */
    private static final int TCP_USER_TIMEOUT = 30;

    @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()
            .keepAlive(SocketOptions.KeepAliveOptions.builder()
                // 两次 keepalive 间的时间间隔
                .idle(Duration.ofSeconds(TCP_KEEPALIVE_TIME))
                // 连接空闲多久开始 keepalive
                .interval(Duration.ofSeconds(TCP_KEEPALIVE_TIME/3))
                // keepalive 几次之后断开连接
                .count(3)
                // 是否开启保活连接
                .enable()
                .build())
            .tcpUserTimeout(SocketOptions.TcpUserTimeoutOptions.builder()
                // 解决服务端rst导致的长时间超时问题
                .tcpUserTimeout(Duration.ofSeconds(TCP_USER_TIMEOUT))
                .enable()
                .build())
            // tcp 连接超时设置
            .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))
                // Proxy集群实例无需设置readFrom
                .readFrom(ReadFrom.MASTER)
                .clientOptions(clientOptions)
                .build();

        return clientConfiguration;
    }
}
```

- 单机、主备、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;
    /**
     *  TCP_KEEPALIVE 配置参数:
     *  两次 keepalive 间的时间间隔 = TCP_KEEPALIVE_TIME = 30
     *  连接空闲多久开始 keepalive = TCP_KEEPALIVE_TIME/3 = 10
     *  keepalive 几次之后断开连接 = TCP_KEEPALIVE_COUNT = 3
     */
    private static final int TCP_KEEPALIVE_TIME = 30;

    /**
     * TCP_USER_TIMEOUT 连接空闲限制时间，解决Lettuce长时间超时问题。
     * refer: https://github.com/lettuce-io/lettuce-core/issues/2082
     */
    private static final int TCP_USER_TIMEOUT = 30;

    @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()
            .keepAlive(SocketOptions.KeepAliveOptions.builder()
                // 两次 keepalive 间的时间间隔
                .idle(Duration.ofSeconds(TCP_KEEPALIVE_TIME))
                // 连接空闲多久开始 keepalive
                .interval(Duration.ofSeconds(TCP_KEEPALIVE_TIME/3))
                // keepalive 几次之后断开连接
                .count(3)
                // 是否开启保活连接
                .enable()
                .build())
            .tcpUserTimeout(SocketOptions.TcpUserTimeoutOptions.builder()
                // 解决服务端rst导致的长时间超时问题
                .tcpUserTimeout(Duration.ofSeconds(TCP_USER_TIMEOUT))
                .enable()
                .build())
            // tcp 连接超时设置
            .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)
                // Proxy集群实例无需设置readFrom
                .readFrom(ReadFrom.MASTER)
                .build();
        return clientConfiguration;
    }

    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，并且空闲连接数>最小空闲连接数时直接逐出

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.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;
    /**
     *  TCP_KEEPALIVE 配置参数:
     *  两次 keepalive 间的时间间隔 = TCP_KEEPALIVE_TIME = 30
     *  连接空闲多久开始 keepalive = TCP_KEEPALIVE_TIME/3 = 10
     *  keepalive 几次之后断开连接 = TCP_KEEPALIVE_COUNT = 3
     */
    private static final int TCP_KEEPALIVE_TIME = 30;

    /**
     * TCP_USER_TIMEOUT 连接空闲限制时间，解决Lettuce长时间超时问题。
     * refer: https://github.com/lettuce-io/lettuce-core/issues/2082
     */
    private static final int TCP_USER_TIMEOUT = 30;

    @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);
        return connectionFactory;
    }

    @Bean
    public LettuceClientConfiguration clientConfiguration() {
        SocketOptions socketOptions = SocketOptions.builder()
            .keepAlive(SocketOptions.KeepAliveOptions.builder()
                // 两次 keepalive 间的时间间隔
                .idle(Duration.ofSeconds(TCP_KEEPALIVE_TIME))
                // 连接空闲多久开始 keepalive
                .interval(Duration.ofSeconds(TCP_KEEPALIVE_TIME/3))
                // keepalive 几次之后断开连接
                .count(3)
                // 是否开启保活连接
                .enable()
                .build())
            .tcpUserTimeout(SocketOptions.TcpUserTimeoutOptions.builder()
                // 解决服务端rst导致的长时间超时问题
                .tcpUserTimeout(Duration.ofSeconds(TCP_USER_TIMEOUT))
                .enable()
                .build())
            // tcp 连接超时设置
            .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;
    /**
     *  TCP_KEEPALIVE 配置参数:
     *  两次 keepalive 间的时间间隔 = TCP_KEEPALIVE_TIME = 30
     *  连接空闲多久开始 keepalive = TCP_KEEPALIVE_TIME/3 = 10
     *  keepalive 几次之后断开连接 = TCP_KEEPALIVE_COUNT = 3
     */
    private static final int TCP_KEEPALIVE_TIME = 30;
```

```
    /**
     * TCP_USER_TIMEOUT 连接空闲限制时间，解决Lettuce长时间超时问题。
     * refer: https://github.com/lettuce-io/lettuce-core/issues/2082
     */
    private static final int TCP_USER_TIMEOUT = 30;

    @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()
            .keepAlive(SocketOptions.KeepAliveOptions.builder()
                // 两次 keepalive 间的时间间隔
                .idle(Duration.ofSeconds(TCP_KEEPALIVE_TIME))
                // 连接空闲多久开始 keepalive
                .interval(Duration.ofSeconds(TCP_KEEPALIVE_TIME/3))
                // keepalive 几次之后断开连接
                .count(3)
                // 是否开启保活连接
                .enable()
                .build())
            .tcpUserTimeout(SocketOptions.TcpUserTimeoutOptions.builder()
                // 解决服务端rst导致的长时间超时问题
                .tcpUserTimeout(Duration.ofSeconds(TCP_USER_TIMEOUT))
                .enable()
                .build())
            // tcp 连接超时设置
            .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，并且空闲连接数>最小空闲连接数时直接逐出，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、Proxy集群实例配置

```
@Bean
public LettuceClientConfiguration clientConfiguration() {
    SocketOptions socketOptions = SocketOptions.builder()
            .keepAlive(SocketOptions.KeepAliveOptions.builder()
                // 两次 keepalive 间的时间间隔
                .idle(Duration.ofSeconds(TCP_KEEPALIVE_TIME))
                // 连接空闲多久开始 keepalive
                .interval(Duration.ofSeconds(TCP_KEEPALIVE_TIME/3))
                // keepalive 几次之后断开连接
                .count(3)
                // 是否开启保活连接
                .enable()
                .build())
            .tcpUserTimeout(SocketOptions.TcpUserTimeoutOptions.builder()
                // 解决服务端rst导致的长时间超时问题
                .tcpUserTimeout(Duration.ofSeconds(TCP_USER_TIMEOUT))
                .enable()
                .build())
```

```
            // tcp 连接超时设置
            .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))
        // Proxy集群实例无需设置readFrom
        .readFrom(ReadFrom.MASTER)
        .clientOptions(clientOptions)
        .useSsl()
        .build();

    return clientConfiguration;
}
```

- Cluster集群实例配置

```
@Bean
public LettuceClientConfiguration clientConfiguration() {
    SocketOptions socketOptions = SocketOptions.builder()
            .keepAlive(SocketOptions.KeepAliveOptions.builder()
                // 两次 keepalive 间的时间间隔
                .idle(Duration.ofSeconds(TCP_KEEPALIVE_TIME))
                // 连接空闲多久开始 keepalive
                .interval(Duration.ofSeconds(TCP_KEEPALIVE_TIME/3))
                // keepalive 几次之后断开连接
                .count(3)
                // 是否开启保活连接
                .enable()
                .build())
            .tcpUserTimeout(SocketOptions.TcpUserTimeoutOptions.builder()
                // 解决服务端rst导致的长时间超时问题
                .tcpUserTimeout(Duration.ofSeconds(TCP_USER_TIMEOUT))
                .enable()
                .build())
            // tcp 连接超时设置
            .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 4-5 LettuceConnectionFactory 参数

参数	类型	默认值	说明
configuration	RedisConfiguration	-	redis连接配置，常用两个子类： • RedisStandaloneConfiguration • RedisClusterConfiguration
clientConfiguration	LettuceClientConfiguration	-	客户端配置参数，常用子类： LettucePoolingClientConfiguration（用于池化）
shareNativeConnection	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 4-6 RedisStandaloneConfiguration 参数

参数	默认值	说明
hostName	localhost	连接Redis实例的IP地址
port	6379	连接端口号
database	0	数据库下标
password	-	连接Redis实例的密码。如果实例已开启免密访问，无需输入实例的访问密码。忘记密码或需要重置密码请参见**重置缓存实例密码**。

表 4-7 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Redis实例的密码。如果实例已开启免密访问，无需输入实例的访问密码。忘记密码或需要重置密码请参见**重置缓存实例密码**。

表 4-8 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
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 4-9 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 4-10 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**

参数	类型	默认值	说明
disconnectedBehavior	DisconnectedBehavior	DisconnectedBehavior.DEFAULT	连接断开时的行为，**建议值：ACCEPT_COMMANDS** ● 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 4-11 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 4-12 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**

参数	默认值	说明
softMinEvictableIdleTimeMillis	-1	连接空闲多久后逐出，（空闲时间>该值 && 空闲连接数>最小空闲连接数）时直接逐出，**建议值：1800000，**单位：毫秒
minEvictableIdleTimeMillis	1800000	根据minEvictableIdleTimeMillis判断逐出，**建议值：-1**，关闭该策略，改用softMinEvictableIdleTimeMillis策略
timeBetweenEvictionRunsMillis	-1	空闲连接逐出的检测周期，**建议值：60000**，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**。

4.2.3.1.3 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。

约束与限制

- 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、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”。
- 查看并获取待连接Redis实例的IP地址和端口。具体步骤请参见**查看和修改DCS实例信息**。

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、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 4-13 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
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 4-14 单机、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

参数	默认值	说明
subcriptionPerConnection	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 4-15 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	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。

参数	默认值	说明
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。
subcriptionPerConnection	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 4-16 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	连接Redis实例的密码。如果实例已开启免密访问，无需输入实例的访问密码。忘记密码或需要重置密码请参见**重置缓存实例密码**。
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。

参数	默认值	说明
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。
subcriptionPerConnection	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：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摘除。

- 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。

  连接池的大小没有固定标准，建议根据业务流量合理配置，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。

4.2.3.2 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。

约束与限制

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”。
- 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》。
- 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。

Redis-py 客户端连接 Redis

- 当连接单机、主备、Proxy集群实例时，请参见Redis-py客户端连接非Cluster集群实例。
- 当连接Cluster集群实例时，请参见Redis-py客户端连接Cluster集群实例。

Redis-py 客户端连接非 Cluster 集群实例

步骤1 查看并获取待连接Redis实例的IP地址和端口。

具体步骤请参见**查看和修改DCS实例信息**。

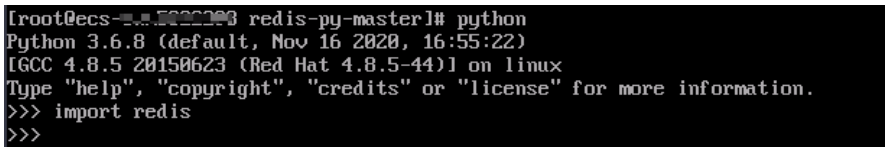
步骤2 登录弹性云服务器。

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。

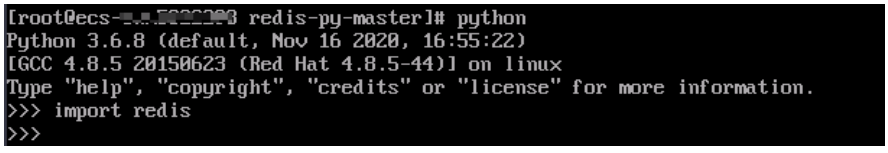
1. 如果弹性云服务器操作系统没有自带Python，可以使用yum方式安装。
   ```
   yum install python
   ```
   要求系统Python版本为3.6+，当默认Python版本小于3.6时，可通过以下操作修改Python默认版本。
   a. 删除Python软链接文件： rm -rf python
   b. 重新创建新指向Python：ln -s pythonX.X.X python，其中X为Python具体版本号。
2. 安装Python和Python Redis客户端redis-py。
   a. 如果系统没有自带Python，可以使用yum方式安装。
   b. 下载并解压redis-py。
      ```
      wget https://github.com/andymccurdy/redis-py/archive/master.zip
      unzip master.zip
      ```
   c. 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 4-2 执行 python

3. 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）：
   a. 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 4-3 进入命令行模式

b. 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，“6379”为Redis实例的端口。IP地址和端口获取见**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。如果实例为免密访问，则省略命令中的, password='******'。

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 4-4 连接 redis 成功

----结束

Redis-py 客户端连接 Cluster 集群实例

步骤1 查看并获取待连接Redis实例的IP地址和端口。

具体步骤请参见**查看和修改DCS实例信息**。

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。

1. 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
   ```

   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。

   a. 删除Python软链接文件： rm -rf python

   b. 重新创建新指向Python：ln -s pythonX.X.X python，其中X为Python具体版本号。

2. 安装redis-py-cluster客户端。

   a. 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
   ```

   b. 解压压缩包。

   ```
   tar -xvf redis-py-cluster-2.1.3.tar.gz
   ```

   c. 进入到解压目录后安装Python Redis客户端redis-py-cluster。

   ```
   python setup.py install
   ```

3. 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）：

   a. 执行**python**命令，进入命令行模式。

   b. 在命令行中执行以下命令，连接Redis实例。

   ```
   >>> 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'
>>>
```

其中，******为创建Redis实例时自定义的密码，请按实际情况修改后执行。如果实例为免密访问，则省略命令中的, password='******'。

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。

----结束

4.2.3.3 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”。
- 查看并获取待连接Redis实例的IP地址和端口。具体步骤请参见**查看和修改DCS实例信息**。
- 已创建弹性云服务器，创建弹性云服务器的方法，请参见《弹性云服务器用户指南》。

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 4-5 终端输入

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{
        Addrs:    []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地址以及端口。IP地址和端口获取见**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。如果实例为免密访问，则省略命令中的密码配置。

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

4.2.3.4 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。

约束与限制

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”。
- 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》。
- 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（可通过**gcc --version**命令查询gcc版本，如果已安装gcc编译环境，会返回gcc版本）。

  如果ECS未安装gcc编译环境，以CentOS系统为例，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++
  ```

Hiredis 客户端连接 Redis

步骤1 查看并获取待连接Redis实例的IP地址和端口。

具体步骤请参见**查看和修改DCS实例信息**。

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，“6379”为Redis实例的端口。IP地址和端口获取见**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。如果实例为免密访问，则省略命令中的密码配置。

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!
```

⚠ 注意

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

4.2.3.5 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。

前提条件

- 已成功创建Redis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》。
- 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（可通过**gcc --version**命令查询gcc版本，如果已安装gcc编译环境，会返回gcc版本）。

  如果ECS未安装gcc编译环境，以CentOS系统为例，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++
  ```

StackExchange.Redis 客户端连接 Redis

步骤1 查看并获取待连接Redis实例的IP地址和端口。

具体步骤请参见**查看和修改DCS实例信息**。

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。

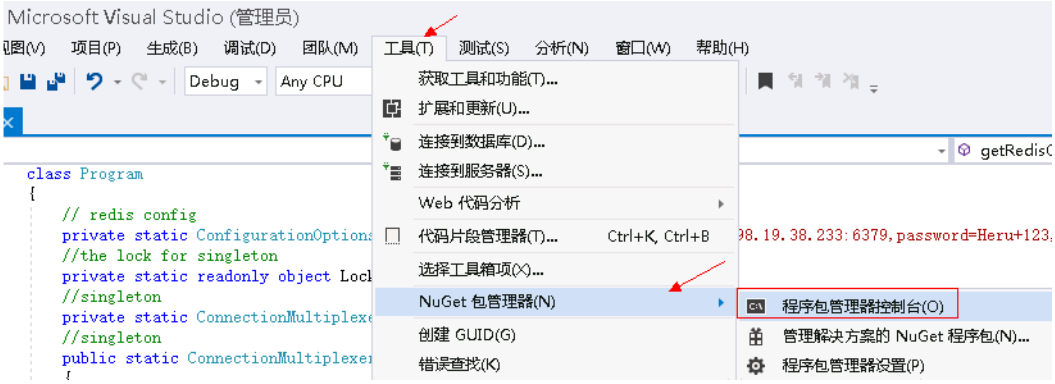
步骤4 启动VS 2017，新建一个工程。

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4-6**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 4-6 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地址以及端口。IP地址和端口获取见**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。如果实例为免密访问，则省略命令中的密码配置。

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

4.2.3.6 PHP 客户端

4.2.3.6.1 Phpredis 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。

约束与限制

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”。
- 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》。
- 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（可通过**gcc --version**命令查询gcc版本，如果已安装gcc编译环境，会返回gcc版本）。

  如果ECS未安装gcc编译环境，以CentOS系统为例，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++
```

Phpredis 客户端连接 Redis

步骤1 查看并获取待连接Redis实例的IP地址和端口。

具体步骤请参见查看和修改DCS实例信息。

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，{port}为Redis实例的端口。IP地址和端口获取见**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。如果实例为免密访问，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

4.2.3.6.2 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”。
- 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》。
- 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。

Predis 客户端连接 Redis

步骤1 查看并获取待连接Redis实例的IP地址和端口。

具体步骤请参见**查看和修改DCS实例信息**。

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 'predis/autoload.php';
      $servers = array(
       'tcp://{redis_instance_address}:{port}'
      );
      $options = array('cluster' => 'redis');
      $client = new Predis\Client($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，{port}为Redis实例真实的端口。IP地址和端口获取见**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

4.2.3.7 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。

约束与限制

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”。
- 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》。
- 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（可通过**gcc --version**命令查询gcc版本，如果已安装gcc编译环境，会返回gcc版本）。

  如果ECS未安装gcc编译环境，以CentOS系统为例，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++
```

Ioredis 客户端连接 Redis

- Ioredis客户端连接Redis，如果客户端服务器为Ubuntu(debian系列)，请参见客户端服务器为Ubuntu(debian系列)。
- Ioredis客户端连接Redis，如果客户端服务器为centos(redhat系列)，请参见客户端服务器为centos(redhat系列)。

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地址和端口。

具体步骤请参见**查看和修改DCS实例信息**。

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/v4.28.5/node-v4.28.5.tar.gz --no-check-certificate
tar -xvf node-v4.28.5.tar.gz
cd node-v4.28.5
./configure
make
make install
```

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，**port**为Redis实例的端口。IP地址和端口获取见**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。如果实例为免密访问，则省略命令中的密码配置。

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地址和端口。

具体步骤请参见**查看和修改DCS实例信息**。

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/v4.28.5/node-v4.28.5.tar.gz --no-check-certificate
tar -xvf node-v4.28.5.tar.gz
cd node-v4.28.5
./configure
make
make install
```

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，**port**为Redis实例的端口。IP地址和端口获取见步骤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。如果实例为免密访问，则省略命令中的密码配置。

步骤7 运行示例脚本，连接Redis实例。

```
node ioredisdemo.js
```

----结束

4.2.4 控制台连接 Redis 实例

DCS支持通过管理控制台的Web CLI功能连接Redis实例。只有Redis 4.0及以上版本的实例支持该功能，Redis 3.0不支持。

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。

前提条件

只有当实例处于“运行中”状态，才能执行此操作。

操作步骤

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 ，选择区域和项目。

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 选中实例，然后单击“操作”栏下的“更多 > 连接Redis”，进入Web CLI登录界面。

图 4-7 连接 Redis

步骤5 输入实例的密码进入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15分钟会连接超时，再次登录需要重新输入访问密码。

----结束

4.3 查看和修改 DCS 实例信息

本节介绍如何在DCS管理控制台查看DCS缓存实例的详细信息。

操作步骤

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 ，选择区域和项目。

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。

  不选择过滤属性，直接在搜索栏输入关键字搜索时，默认按照实例名称进行搜索。

- 支持通过指定属性的关键字查询对应的DCS缓存实例。

  单击搜索栏，选择实例属性后，输入对应属性的关键字进行搜索，可同时选择多个不同的过滤属性。

  例如，选择“状态>运行中”，“实例类型>主备”，或选择“缓存类型>Redis 5.0”等。

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。参数说明如下表所示。

表 4-17 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的缓存类型，同时还会展示版本号，例如，Redis 5.0。
	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。具体操作请参考**升级DCS实例小版本**。
	实例类型	DCS缓存实例类型，支持“单机”、“主备”和“Cluster集群”。
	规格	DCS缓存实例规格。

信息类型	参数	说明
	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，具体操作请参考**调整DCS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： • 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。
	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。
	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	当前支持密码访问和免密访问两种方式。
	IP地址	DCS缓存实例的IP和端口号。单击参数后的可以修改端口号。
网络信息	可用区	缓存节点所属的可用区。
	虚拟私有云	DCS缓存实例所在的私有网络。
	子网	DCS缓存实例所属子网。
	安全组	DCS缓存实例所关联的安全组。 Redis 4.0及以上版本实例是基于VPCEndpoint，暂不支持安全组，单击“配置”可以配置白名单。
付费信息	计费方式	按需计费。
	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。
实例拓扑	-	查看实例拓扑图，将鼠标移动到具体实例图标，可以查看该实例的总体监控信息，或者单击实例图标，可以查看实例历史监控信息。 仅主备和集群实例显示实例的拓扑图。

----结束

5 实例日常操作

5.1 变更规格

DCS管理控制台支持变更Redis缓存实例规格，即扩容/缩容，您可以根据实际需要，选择合适的实例规格。

说明

- **执行实例规格变更操作，建议在业务低峰期进行**。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。
- 当前除Redis 3.0单机和主备实例外，其他实例类型不支持跨实例类型的变更。
- 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。
- 变更规格过程中会有秒级业务中断，需要业务连接Redis的模块支持连接中断后重连。
- 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。

实例类型变更前须知

- **支持实例类型变更明细如下：**
  - Redis 3.0和Memcached单机实例支持变更为主备实例，Redis 4.0/Redis 5.0/Redis 6.0单机实例不支持变更。
  - Redis 3.0主备实例支持变更为Proxy集群实例，Redis 4.0/Redis 5.0/Redis 6.0主备实例暂不支持变更。

    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。
  - 集群实例，不支持实例类型变更。
- **实例类型变更影响：**
  - Redis 3.0单机实例类型变更为Redis 3.0主备实例。

    连接会有秒级中断，大约1分钟左右的只读。
  - Redis 3.0主备实例类型变更为Redis 3.0 Proxy实例。

    连接会中断，5~30分钟只读。

实例规格大小变更前须知

- **支持扩容和缩容明细如下：**

  表 5-1 DCS 实例规格变更说明

缓存类型	单机实例	主备实例	Cluster集群实例	Proxy集群实例
Redis 3.0	支持扩容和缩容	支持扩容和缩容	不涉及	支持扩容
Redis 4.0/5.0	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	不涉及
Redis 6.0/7.0	支持扩容和缩容	支持扩容和缩容	支持扩容、缩容和副本数变更	不涉及
Memcached	支持扩容和缩容	支持扩容和缩容	不涉及	不涉及

  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。

  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。

- **实例规格大小变更影响如下：**
  - 单机和主备实例规格大小变更
    - Redis 4.0及以上版本实例变更期间，连接会有秒级中断，大约1分钟的只读。
    - Redis 3.0实例变更期间，连接会中断，5~30分钟只读。
    - 如果是扩容，只扩大实例的内存，不会提升CPU处理能力。
    - 如果是单机实例规格变更，由于单机实例不支持持久化，没有数据可靠性，变更实例可能会丢失数据。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。
    - 主备实例的备份记录，缩容后不能使用。
  - 集群实例规格大小变更
    - 规格变更分片数未减少时，连接不中断，但会占用CPU，导致性能有20%以内的下降，扩容数据迁移期间，访问时延会增大。
    - 集群实例扩容会新增加数据节点，数据自动负载均衡到新的数据节点。
    - 规格变更分片数减少时，会删除节点，请确保应用中没有直接引用这些删除的节点。删除节点会导致连接闪断。
    - 规格变更后实例每个节点的已用内存必须小于节点最大内存的70%，否则将不允许变更。

- 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，建议在业务低峰期进行。
- 实例规格变更期间，会进行数据迁移，访问正在迁移的key时，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。
- 请在规格变更前先使用缓存分析中的大key分析，确保实例中没有大key（≥512MB）存在，否则可能会导规格变更失败。
- 变更规格前的备份记录不能恢复。

- **Redis实例副本数变更须知：**

  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。

操作步骤

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 ，选择区域和项目。

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 在需要规格变更的实例右侧，单击“操作”栏下的“更多 > 变更规格”，进入到分布式缓存服务变更规格页面。

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。

步骤6 单击“提交订单”，开始变更DCS缓存实例。

在界面上您可以选择跳转到后台任务列表，查看变更任务的状态，具体可参考**查看实例后台任务**。

DCS单机和主备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。实例规格变更成功后，实例状态切换为“运行中”。

说明

- 当单机实例规格变更失败时，实例对用户暂不可用，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，实例对用户仍然可用，实例规格仍然为变更前的规格，部分管理操作（如参数配置、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。

----结束

5.2 重启实例

DCS管理控制台支持重启运行中的DCS缓存实例，且可批量重启DCS缓存实例。

须知

- 重启DCS缓存实例后，单机实例中原有的数据将被删除，其他实例类型若关闭了AOF持久化（参数配置appendonly为no），数据将清空，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，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。

前提条件

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。

操作步骤

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 ，选择区域和项目。

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 勾选“名称”栏下的相应DCS缓存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“重启”。

步骤6 单击“是”，完成重启DCS缓存实例。

重启DCS缓存实例大约需要10秒到30分钟。DCS缓存实例重启成功后，缓存实例状态切换为“运行中”。

说明

- 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”栏下的“重启”。
- 重启DCS缓存实例具体需要时长同实例的缓存大小有关。

----结束

5.3 删除实例

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、已关闭、故障时才能执行删除操作。
- 删除操作无法撤销，且实例中的缓存数据以及备份文件都将被删除，请谨慎操作。如需保留数据备份记录，建议在删除实例前，将实例的备份文件下载到本地永久保存。
- 仅Redis 4.0及以上版本的实例支持选择删除的实例是否放入回收站。

- 如果是集群实例，会将实例的所有节点删除。

操作步骤

删除缓存实例

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 ，选择区域和项目。

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 勾选“名称”栏下的需要删除的DCS缓存实例左侧的方框，可选一个或多个，单击信息栏左上侧的“删除”。

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”。

步骤5 在删除窗口选择“是否放入回收站”，开启后删除的实例将放入回收站，后续可以进行恢复或销毁。

- 放入回收站的实例在7日内支持恢复或销毁，超过7天将自动从回收站内销毁。如何恢复或销毁回收站内的实例，请参考**恢复或销毁回收站内的DCS实例**。
- 如果实例删除前是运行中的状态，开启“是否放入回收站”后，删除实例时会自动备份数据（单机实例除外），用于恢复实例时的数据恢复。
- 如果实例删除前是故障或已关闭状态，开启“是否放入回收站”后，删除实例时不会自动备份数据，会保存该实例删除前最新的备份记录，用于恢复实例时的数据恢复。如果实例删除前没有备份记录，则无法进行数据恢复，建议定期对实例进行备份。

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”栏下的“更多 > 删除”。

----结束

删除创建失败的缓存实例

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 ，选择区域和项目。

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 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5 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6 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

5.4 主备切换

DCS管理控制台支持手动切换DCS缓存实例的主备节点，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。

只有主备实例支持该操作。

Redis 4.0及以上版本的实例，如果需要手动切换集群实例分片的主备节点，请在实例的节点管理功能中操作，具体操作可参考**管理分片与副本**。

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。

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。

操作步骤

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 ，选择区域和项目。

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 在需要进行主备切换的缓存实例右侧，单击“操作”栏下的“更多 > 主备切换”，单击“确认”，完成主备切换。

----结束

5.5 清空实例数据

Redis 4.0及以上版本的实例，支持在控制台执行“清空实例数据”的功能。

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。

前提条件

只有当Redis实例处于“运行中”状态，才能执行此操作。

操作步骤

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 ，选择区域和项目。

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 勾选“名称”栏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“数据清空”。

步骤6 单击“是”，清空实例数据。

----结束

5.6 导出实例列表

DCS管理控制台支持全量导出缓存实例信息功能，导出形式为Excel。

操作步骤

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 ，选择区域和项目。

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“导出”，即可下载缓存实例列表。

图 5-1 缓存实例列表

Name	ID	Status	AZ	Cache Eng	Instance	Specifica	Used/Avai	Connecti	Created(U	Billing M	VPC	VPC ID	Enterpris	Domain	Description
dcs-zfvc	2ae6ed5f-	RUNNING	AZ2	Redis 5.0	Master/St	0.125	2/128	(1.172.17.1	(2023-05-1	Pay-per-u	cae-devop	5026c30b-	default		null
dcs-ikpr	765031cd-	RUNNING	AZ2, AZ1	Redis 5.0	Master/St	0.125	2/128	(1.172.17.1	(2023-05-0	Pay-per-u	cae-devop	5026c30b-	default		null
dcs-ds2q	408a232f-	RUNNING	AZ2	Redis 6.0	Single-no	0.125	1/128	(0.172.17.1	(2023-05-1	Pay-per-u	cae-devop	5026c30b-	default		null

----结束

5.7 命令重命名

Redis 4.0及以上版本的实例创建之后，支持重命名高危命令。当前支持重命名的高危命令有command、keys、flushdb、flushall、hgetall、scan、hscan、sscan、和zscan，其他命令暂时不支持。

操作步骤

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 ，选择区域和项目。

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 在需要进行重命名命令的缓存实例右侧，单击“操作”栏下的“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。

说明

- 因为涉及安全性，页面不会显示这些命令，请记住重命名后的命令。
- 命令重命名提交后，系统会自动重启实例，实例完成重启后重命名生效。
- 需要还原某个重命名命令的话，和原命令填写相同即可还原。
- 重命名的长度范围为4到64个字符，重命名需以字母开头，且只能包含字母、数字、下划线，和中划线。

----结束

5.8 实例关机

Redis 4.0及以上版本的实例，支持实例关机操作，实例关机后停止数据读写，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、和数据迁移等操作。

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。

注意

考虑用户数据安全或防止误关机操作，除单机实例外，关机前需有实例数据备份记录。备份数据的操作，请参见备份恢复实例缓存数据。

操作步骤

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 ，选择区域和项目。

步骤3 单击左侧菜单栏的“缓存管理”。

步骤4 在需要关机的缓存实例右侧，单击“操作”栏下的“关机”。

步骤5 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。

说明

实例关机后如需开机，请以相同方式单击缓存实例右侧“操作”栏下的“启动”，重新开机实例。

----结束

5.9 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操作并消耗网络带宽。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

- 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。
- 实例单分片带宽的调整范围在单分片的基准带宽（默认带宽）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。
- 目标带宽只支持设置为8的整数倍。如果设置的值不为8的整数倍，订单提交后将自动向下取8的倍数。
- 调整带宽的计费方式仅支持按需计费（按小时结算费用），请注意配置费用的变化。

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 ，选择区域和项目。

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”。

步骤6 在弹出的“调整带宽”页面，设置需要调整的带宽值。

集群实例支持仅调整单个分片带宽，如果需要同时调整多个分片带宽时，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。

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，订单提交后会自动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）的宽带变更为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。

步骤7 确认新的带宽值及带宽费用后，在“带宽调整确认”处勾选确认，再单击“提交订单”。

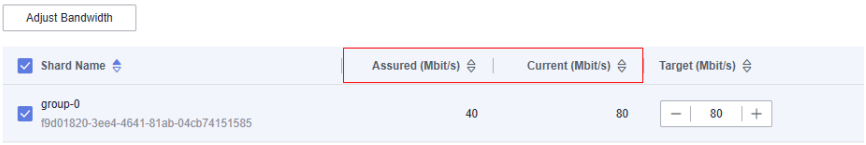
调整带宽任务的状态为“成功”后，新的带宽值立即生效。

----结束

如何查看基准带宽和调整后的带宽

在手动调整带宽的页面，可以查看实例每个分片的“基准带宽”和“当前带宽”。对于已经调整过带宽的实例，“当前带宽”即调整后的带宽值。

图 5-2 查看带宽值



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。
- 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。

  例如一个3分片的集群实例，每个分片调整后的带宽为800 Mbit/s，该集群实例总带宽为2400 Mbit/s。

5.10 升级 DCS 实例小版本

DCS会不定期升级实例小版本，优化产品功能和修复产品缺陷，提升服务稳定性。本章节介绍如何升级已创建实例的小版本到最新版。

升级实例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。

约束与限制

- 仅Redis 4.0及以上版本实例支持升级实例的小版本。
- 实例仅支持升级为最新小版本，不支持指定为其他版本。
- 实例升级小版本时，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。
- 实例升级小版本或代理版本后不支持回退。

升级影响

- 建议在业务低峰期执行实例小版本升级。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。
- 实例升级小版本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。

升级 DCS 实例小版本

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 ，选择区域和项目。

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，进入实例详情页面。

步骤5 在“基本信息”区域，单击“小版本”后的“升级小版本”。

步骤6 单击“确定”，提交实例升级任务。待升级版本任务的状态显示“成功”后，升级版本完成。

----结束

5.11 升级 Redis 3.0 实例大版本

Redis 3.0版本较老，开源社区已不再对其进行更新，DCS提供的Redis高版本兼容Redis 3.0，建议客户尽快将DCS Redis 3.0升级到高版本。本章节介绍如何将Redis 3.0一键升级到高版本。

约束与限制

- 目前仅支持升级Redis 3.0实例到高版本，暂不支持Redis 4.0及以上版本的实例升级大版本。
- 建议在业务低峰期进行实例升级操作。
- 实例升级大版本后不支持回退。

升级影响

- 在版本升级的过程中会产生一分钟以内的只读和秒级的连接闪断，请确保客户端应用具备重连机制和处理异常的能力。
- Redis 3.0实例版本升级后带宽有所降低，请评估变更后的带宽能否满足业务需求。
- 升级前的备份文件在升级后不可用于恢复实例。

升级 Redis 3.0 实例大版本

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 ，选择区域和项目。

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，进入实例详情页面。

步骤5 在“基本信息”区域，单击“缓存类型”后的“升级大版本”。

步骤6 在升级大版本页面选择“目标版本”，并确认目标版本的带宽和价格。

步骤7 单击“提交订单”。当升级任务的状态显示“成功”后，升级版本完成。

说明

Redis 3.0实例升级后，对于原实例和高版本实例都存在的配置参数，在升级后会继承原实例的参数配置值；对于升级后高版本实例新增的配置参数，会保持该配置参数的默认值。

----结束

5.12 恢复或销毁回收站内的 DCS 实例

DCS提供了实例回收站，删除实例时开启了“是否放入回收站”的实例默认会放入回收站。用户可以对回收站内的实例进行恢复或销毁。回收站内的实例不计费，实例恢复成功后重新计费。

恢复回收站内的实例，即按照原实例规格信息、参数和功能配置重新创建一个新实例，并将原实例的备份数据恢复到新实例。

约束与限制

- 仅Redis 4.0及以上版本的实例支持放入回收站。
- 回收站内的实例仅保留7天，7天后自动从回收站内销毁。
- **恢复后的实例，实例ID为新ID，域名连接地址为新地址。**
- **实例恢复后，SSL功能默认关闭，调整过的实例带宽不支持恢复，仅支持恢复为默认带宽。**
- 如果实例删除/退订前是运行中的状态，删除/退订实例时会自动备份数据（单机实例除外），用于恢复实例时的数据恢复。
- 如果实例删除/退订前是故障或已关闭的状态，删除/退订实例时不会自动备份数据，会保存该实例删除/退订前最新的备份记录，用于恢复实例时的数据恢复。如果实例删除/退订前没有备份记录，则无法进行数据恢复，建议定期对实例进行备份。

恢复回收站内的 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 ，选择区域和项目。

步骤3 单击左侧菜单栏的“回收站”。

步骤4 单击需要恢复的实例右侧对应的“恢复”。

步骤5 确定需要恢复该实例后，单击“确定”。

步骤6 在恢复已删除实例页面，确认实例信息。

- 如果实例删除/退订前是运行中的状态，删除/退订实例时会自动备份数据（单机实例除外），用于恢复实例时的数据恢复。
- 如果实例删除/退订前是故障或已关闭的状态，删除/退订实例时不会自动备份数据，会保存该实例删除/退订前最新的备份记录，用于恢复实例时的数据恢复。如果实例删除/退订前没有备份记录，则无法进行数据恢复，建议定期对实例进行备份。
- 如果回收站内的实例有备份记录，备份ID处会显示具体ID，如果实例没有备份记录，备份ID处为空。

步骤7 配置“网络配置”和“访问管理”。配置方式可以参考**购买Redis实例**。

IP地址和端口默认为原实例IP和端口，如果原IP被占用，则需要自动分配和手动分配其他地址。

步骤8 确认后单击“立即恢复”，进入实例信息确认页面。

步骤9 确认实例信息无误后，提交请求。

步骤10 任务提交成功后，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。

实例创建后，如果是有备份记录的实例，自动进行备份迁移，单击“数据迁移”查看该实例备份迁移任务，迁移成功后，即实例数据恢复完成。

----结束

销毁回收站内的 DCS 实例

如果需要销毁回收站内的实例，请参考以下操作销毁回收站内的实例。销毁操作无法撤销，且缓存实例和备份文件都将被删除，请谨慎操作。

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 ，选择区域和项目。

步骤3 单击左侧菜单栏的“回收站”。

步骤4 勾选需要销毁的实例，单击“立即销毁”。

步骤5 如果您确定要销毁实例，请输入DESTROY（单击“一键输入”可以直接输入DESTROY）后单击“确定”。

当页面提示成功销毁缓存实例时，表示实例销毁成功。

----结束

6 实例配置管理

6.1 配置管理说明

- 一般情况下，缓存实例的创建、配置运行参数、重启、修改缓存实例密码、重置缓存实例密码、备份恢复、规格变更等管理操作均不支持同时进行。即当实例正在进行其中一个操作时，如果您执行其他操作，界面会提示缓存实例正在进行相应操作，请稍后进行重试操作。
- 在如下场景下，您需要尽快执行后续的管理操作，以恢复业务，则支持同时进行：

  在备份缓存实例过程中，支持重启缓存实例，此时备份操作会强制中断，备份任务的执行结果可能为成功或者失败。

须知

当实例的部分节点出现故障时：

- 由于DCS服务的高可用性，您可以正常读写实例，缓存实例状态仍然处于“运行中”。
- DCS服务内部会自动修复故障，或者由服务运维人员手工修复故障。
- 故障修复期间，管理域的部分操作暂不支持，包括修改配置参数、修改密码、重置密码、备份恢复、规格变更等，您可以等故障节点恢复之后进行重试操作或联系技术支持。

6.2 修改实例配置参数

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，需要将实例持久化功能关闭，则需要将“appendonly”修改为“no”。

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。

操作步骤

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 ，选择区域和项目。

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 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”页签进入配置界面。

步骤6 单击“修改”。

步骤7 根据需要修改配置参数。

各参数的详细介绍见表6-1，一般情况下，按照系统默认值设置参数即可。

表 6-1 Redis 缓存实例配置参数说明

参数名	参数解释	取值范围	默认值
active-expire-num	过期键定期删除时随机检查key的数量。 Redis 4.0及以上版本的实例支持该参数。	1～1,000	20
timeout	客户端空闲N秒（timeout参数的取值）后将关闭连接。当N=0时，表示禁用该功能。 Proxy集群实例不支持该参数。	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。 Redis支持三种不同的调用fsync的方式： 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。 单机实例不支持该参数。	• no • always • everysec	no

参数名	参数解释	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（即持久化功能），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。 yes：开启。 no：关闭。 单机实例不支持该参数。	• yes • no	yes
client-output-buffer-limit-slave-soft-seconds	slave客户端output-buffer超过client-output-buffer-slave-soft-limit设置的大小，并且持续时间超过此值时，服务端会主动断开连接。 单机实例不支持该参数。	0~60 单位：秒	60
client-output-buffer-slave-hard-limit	对slave客户端output-buffer的硬限制（单位为字节），如果slave客户端output-buffer大于此值，服务端会主动断开连接。 单机实例不支持该参数。	取值范围与实例的类型及规格有关	默认值与实例的类型及规格有关
client-output-buffer-slave-soft-limit	对slave客户端output-buffer的软限制（单位为字节），如果output-buffer大于此值并且持续时间超过client-output-buffer-limit-slave-soft-seconds设置的时长，服务端会主动断开连接。 单机实例不支持该参数。	取值范围与实例的类型及规格有关	默认值与实例的类型及规格有关

参数名	参数解释	取值范围	默认值
maxmemory-policy	在达到内存上限（maxmemory）时DCS将如何选择要删除的内容。有8个取值供选择： 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。	取值范围与实例的版本有关	默认值与实例的版本及类型有关
lua-time-limit	Lua脚本的最长执行时间。	100～5,000 单位：毫秒	5,000
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。	• yes • no	no
maxclients	最大同时连接的客户端个数。 Proxy集群实例不支持该参数。	取值范围与实例的类型及规格有关	默认值与实例的类型及规格有关
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	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。 单机实例不支持该参数。	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。 单机实例不支持该参数。	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表中只有少量记录时，使用有利于节约内存的数据结构来对hashes进行编码。	1~10000	512
hash-max-ziplist-value	当hash表中最大的取值不超过预设阈值时，使用有利于节约内存的数据结构来对hashes进行编码。	1~10000	64
set-max-intset-entries	当一个集合仅包含字符串且字符串为在64位有符号整数范围内的十进制整数时，使用有利于节约内存的数据结构对集合进行编码。	1~10000	512
zset-max-ziplist-entries	当有序集合中只有少量记录时，使用有利于节约内存的数据结构对有序序列进行编码。	1~10000	128
zset-max-ziplist-value	当有序集合中的最大取值不超过预设阈值时，使用有利于节约内存的数据结构对有序集合进行编码。	1~10000	64

参数名	参数解释	取值范围	默认值
latency-monitor-threshold	延时监控的采样时间阈值（最小值）。 阈值设置为0：不做监控，也不采样。 阈值设置为大于0：将记录执行耗时大于阈值的操作。 可以通过LATENCY等命令获取统计数据和配置、执行采样监控。 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	取值范围	默认值
notify-keyspace-events	notify-keyspace-events选项的参数为空字符串时，功能关闭。另一方面，当参数不是空字符串时，功能开启。notify-keyspace-events的参数可以是以下字符的任意组合，它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不能与g$lshzxe同时出现，不能出现相同字母。举个例子，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。 Proxy集群实例不支持该参数。	请参考该参数的描述。	Ex

参数名	参数解释	取值范围	默认值
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用于配置记录到慢查询的命令执行时间阈值。	0～1,000,000 单位：微秒	10,000
slowlog-max-len	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

说明

1. 表6-1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。
2. 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。
3. notify-keyspace-events参数的其他描述：
   - 有效值为[K|E|KE][A|g|l|s|h|z|x|e|$]，即输入的参数中至少要有一个K或者E。
   - A为“g$lshzxe”所有参数的集合别名。A与“g$lshzxe”中任意一个不能同时出现。
   - 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。若将参数设为字符串"AKE"表示发送所有类型的通知。
4. 不同实例类型支持配置的参数和取值可能略有不同，请以控制台显示为准。

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。

----结束

配置参数典型使用场景

以appendonly参数为例，介绍修改该参数的典型使用场景如下：

- 若将Redis当做缓存使用，业务对Redis缓存数据的丢失不敏感的场景下，可以将实例持久化功能关闭，这样有助于提升Redis性能，此时只需要将“appendonly”配置参数修改为“no”即可，具体操作可参考操作步骤。
- 若将Redis当做数据库使用，或对Redis缓存数据丢失较敏感的场景，可以将实例持久化功能开启，此时只需要将“appendonly”配置参数值修改为“yes”，具体操作可参考操作步骤。开启实例持久化后，若对Redis缓存中的数据写入磁盘频率和对Redis性能的影响需要综合考虑，可结合“appendfsync”配置参数一起使用，Redis支持三种不同的调用fsync的方式：
  - 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。

说明

目前只有主备、Redis 4.0及以上版本的集群实例可通过控制台修改appendonly、appendfsync参数配置。

6.3 查看实例后台任务

对实例的一些操作，如规格变更、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。

操作步骤

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，选择区域和项目。

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 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”栏下的“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

6.4 管理标签

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。

说明

如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。标签如果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。

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，您最多可以给实例添加20个标签。另外，您还可以进行修改和删除标签。

标签共由两部分组成：“标签键”和“标签值”，其中，“标签键”和“标签值”的命名规则如表6-2所示。

表 6-2 标签命名规则

参数名称	规则
标签键	● 不能为空。 ● 对于同一个实例，Key值唯一。 ● 首尾字符不能为空格。 ● 长度不超过128个字符。 ● 可以包含任意语种的字母、数字、空格和_ . : = + - @。 ● 不能以_sys_开头。
标签值	● 长度不超过255个字符。 ● 可以包含任意语种的字母、数字、空格和_ . : / = + - @。 ● 首尾字符不能为空格。

操作步骤

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 ，选择区域和项目。

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 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“实例配置>标签”页签，进入标签管理页面。

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”对话框。您可以直接在“标签键”和“标签值”中输入设置标签。

     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。
  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“确定”。

----结束

6.5 管理分片与副本

本节主要介绍如何查询Redis 4.0及以上版本实例分片和副本信息，以及将集群实例的从节点手动升级为主节点的操作。

当前仅Redis 4.0及以上版本的主备、集群实例支持该功能，单机实例和Redis 3.0版本实例不支持该功能。

- 主备实例，分片数为1，默认是一个一主一从的双副本架构，支持通过“分片与副本”查看分片信息，如果需要手动切换主从节点，请执行**主备切换**操作。
- 集群实例由多个分片组成，每个分片默认都是一个双副本架构，您可以通过“分片与副本”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。不同实例规格对应的分片数，具体请参考**Redis Cluster集群实例**。

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 ，选择区域和项目。

步骤3 单击左侧菜单栏的“缓存管理”。进入缓存管理页面。

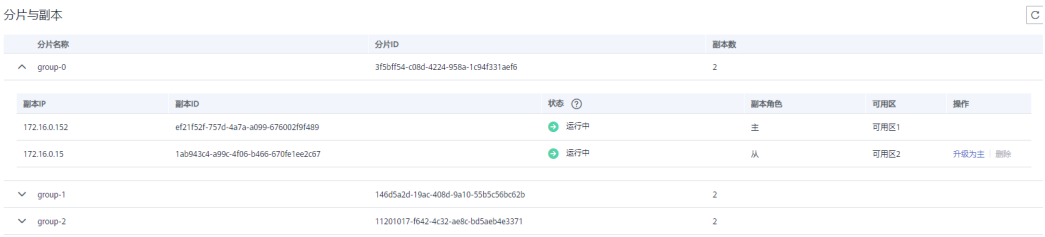
步骤4 单击缓存实例名称，进入该实例的基本信息页面。

步骤5 单击“分片与副本”页签，进入分片与副本页面。

界面显示该实例的所有分片列表，以及每个分片的副本列表。

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。

图 6-1 分片与副本列表



- 对于集群实例，可以将分片中的从副本升级为主副本。
  a. 选择角色为从的副本，单击“升级为主”。
  b. 单击“是”，将选择的副本升级为主。
- 如果是主备实例，可以设置从副本的“主备切换优先级”。
  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和切换。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。

----结束

6.6 分析 Redis 实例大 Key 和热 Key

大Key和热Key问题是Redis使用中的常见问题，本章节主要介绍对Redis实例进行大Key和热Key分析，通过大Key和热Key分析，可以监控到占用空间过大的Key，以及该Redis实例存储数据中被访问最多的Key。

大Key分析使用限制和说明：

- 所有Redis实例都支持。
- 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。
- 在进行大Key分析时，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（不高于10%），所以建议在业务低谷期进行大Key分析。
- 对于大Key分析结果，每个Redis实例默认最多保存100条记录（string类型保存top20，list/set/zset/hash类型保存top80，每种类型最多20条），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及以上版本实例支持，并且实例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，要考虑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分析是对于主节点的分析，在进行分析时，客户访问性能会略有影响（不高于10%）。
- 对于热Key分析结果，每个Redis实例默认最多保存100条记录。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

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。

大 Key 分析操作步骤

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 ，选择区域和项目。

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 单击"分析与诊断>缓存分析"页签。

步骤6 在“缓存分析”页面的“大Key分析”页签，您可以立即对实例进行大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，可以单击分析列表“操作”列的“查看”，查看分析结果。

您可以查询当前实例不同数据类型的大Key分析结果。

说明

分析结果中，string类型显示top20的记录，list/set/zset/hash类型显示top80的记录。具体分析记录，请以实际返回结果为准。

表 6-3 大 Key 分析结果参数说明

参数名称	参数说明
Key名称	大Key的名称。
类型	大Key的类型，包括string和list/set/zset/hash数据类型。
大小	大Key的Value的大小或元素的个数。
分片	集群实例大Key所在的分片。
Database	大Key所在的DB。

----结束

热 Key 分析操作步骤

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 ，选择区域和项目。

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 单击"分析与诊断 > 缓存分析"页签。

步骤6 在“缓存分析”页面的“热Key分析”页签，您可以对实例进行热Key分析或者设置定时任务，每日自动分析。

说明

如果无法执行热Key分析，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。如果是已经配置为allkeys-lfu或者volatile-lfu，即可立即进行热Key分析。

步骤7 当分析任务结束后，可以单击分析列表“操作”列的“查看”，查看分析结果。

您可以查询当前实例的热Key分析结果。

说明

热Key分析结果，每个Redis实例默认显示top100的记录。

表 6-4 热 Key 分析结果参数说明

参数名称	参数说明
Key名称	热Key的名称。
类型	热Key的类型，包括String、Hash、List、Set、Sorted Set等数据类型。
大小	热Key的Value的大小。
频度	表示某个key在一段时间的访问频度，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	集群实例热Key所在的分片。
DataBase	热Key所在的DB。

----结束

6.7 管理实例白名单

本章节主要介绍如何管理Redis 4.0及以上版本实例的白名单，如果需要指定的IP地址才能访问实例，您需要将指定的IP地址加入到实例白名单中。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 单击左侧菜单栏的“缓存管理”，进入实例信息页面。

步骤2 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3 单击"实例配置>白名单配置"页签，然后单击“创建白名单分组”。

步骤4 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 6-5 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。	DCS-test

参数名称	参数说明	示例
IP地址/地址段	每个实例最多可以添加20个IP地址/地址段。如果有多个，可以用逗号分隔。 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

步骤5 设置完之后，单击“确定”。

创建成功之后默认开启白名单功能，只有该分组白名单中的IP地址才允许访问实例。

说明

- 在白名单列表，您可以单击“编辑”修改该分组下的IP地址/地址段。或者单击“删除”，删除该白名单分组。
- 开启白名单功能后，您可以单击白名单列表左上角的“停用白名单”，让所有与实例VPC相通的IP都能访问该实例。

----结束

6.8 查询 Redis 实例慢查询

慢查询是Redis用于记录命令执行时间过长请求的机制。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。

慢日志查询结果由实例两个配置参数决定，如下：

- 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记录的慢查询个数由slowlog-max-len参数的值决定，默认值为128个。当慢查询个数超过128时，会将旧的慢查询删除，记录新的慢查询。

实例配置参数的修改以及参数解释，请参考**修改实例配置参数**。

控制台查看慢查询

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 ，选择区域和项目。

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 单击需要进行慢日志查询的DCS缓存实例名称，进入该实例的基本信息页面。

步骤5 单击"分析与诊断 > 慢查询"。

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。

说明

- 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。
- 目前最长仅支持查询近7天内的慢查询记录。

----结束

6.9 查询 Redis 实例运行日志

您可以在控制台配置Redis实例日志采集任务，根据时间采集redis.log日志内容，采集成功后，您可以将日志下载到本地，查看实例的运行日志。

Redis 4.0及以上版本的实例支持该功能。

操作步骤

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 ，选择区域和项目。

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”。

步骤6 单击“采集日志文件”，配置日志采集信息。选择采集时长后，单击“确定”。

如果是主备和集群实例，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”，可以下载该文件。

说明

- 采集成功的日志文件只保留7天，请及时下载。
- Redis内核产生日志较少，您选择的时段内日志可能为空。

----结束

6.10 实例诊断

使用场景

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。

使用限制

Redis 4.0及以上版本的实例支持实例诊断。

操作步骤

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 ，选择区域和项目。

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 单击“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。默认诊断实例所有节点。
- 时间区间：支持诊断实例7天内的数据，每次诊断最长时间周期为10分钟。

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。单击“删除”，可以删除诊断记录。

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

6.11 配置 Redis SSL 数据加密传输

Redis 6.0单机、主备、Cluster集群实例支持开启SSL链路传输加密，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。

约束与限制

- SSL和客户端IP透传功能无法同时开启，开启SSL后，数据进行加密传输，加密链路下不支持携带客户端IP。
- **开启SSL后，实例的读写性能会有所下降。**
- **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。

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

7 备份恢复实例缓存数据

7.1 备份与恢复概述

介绍如何通过管理控制台对DCS实例进行数据备份，以及恢复备份的数据。

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。

  定时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。

  定时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，并存储到对象存储服务（OBS）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。

  缓存实例在使用过程中，备份数据不会自动清除，您可根据需要手动删除备份数据。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。

备份的其他说明

- 支持备份的实例类型
  - 只有“主备”、“Proxy集群”和“Cluster集群”实例类型的Redis实例支持数据备份与恢复功能，“单机”Redis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。
  - 只有“主备”类型的Memcached实例支持数据备份与恢复功能，“单机”类型Memcached实例暂不支持。
- 备份原理

  Redis 3.0实例采用Redis的AOF方式进行持久化，Redis 4.0及以上版本的实例，如果是手动备份，支持选择RDB格式和AOF格式进行持久化；如果是自动备份，仅支持RDB格式进行持久化。

  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对象存储服务（OBS）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。

  数据备份只针对用户的key-value型数据，不包含实例配置等其他数据。
- 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。

  由于备节点停止将发生的最新数据变化持久化到磁盘文件，备份期间主节点如有新的数据写入，备份文件也不会包含备份期间的数据变化。
- 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。
- 定时备份异常的处理

  定时备份任务触发后，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。

  实例备份失败或者因为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

  定时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，但至少会保留一个数据备份文件。

  手动备份的数据保存期限无限制，由用户根据需要自行删除。

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。

关于数据恢复

- 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 设置自动备份策略

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，请参考本章节操作步骤。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、集群缓存实例，且实例处于运行中状态。

操作步骤

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 ，选择区域和项目。

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 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。

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。

说明

如果修改了备份策略的“保留天数”，新的保留天数策略仅对新的备份文件生效，对于修改备份策略前已经备份的文件无效。

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 手动备份实例

当您需要及时备份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实例的数据。

手动备份的实例数据默认永久保存，如需清理，请自行删除。

前提条件

已成功申请主备DCS缓存实例，且实例处于运行中状态。

操作步骤

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 ，选择区域和项目。

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 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。

仅Redis 4.0及以上版本的实例支持选择备份格式，其他实例不支持。

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。

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。

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 实例恢复

您可以将已备份的缓存数据恢复到DCS缓存实例中。

前提条件

- 已成功创建主备或集群DCS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**。

操作步骤

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 ，选择区域和项目。

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 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。

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}**。
- 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。

前提条件

实例已做备份且没有过期。

操作步骤

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 ，选择区域和项目。

步骤3 单击左侧菜单栏的“缓存管理”。进入缓存管理页面。

“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 选择下载方式。

包括以下两种下载方式：

- URL下载
  a. 设置URL有效期并单击“查询”按钮。
  b. 通过URL列表下载备份文件。

说明

如果复制下载链接，并在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。

- OBS下载

  按照页面的下载步骤描述操作即可。

----结束

8 使用 DCS 迁移数据

8.1 使用 DCS 迁移介绍

迁移概览

DCS Redis支持备份文件导入（离线迁移）和在线迁移两种迁移方式，其中，在线迁移支持增量数据迁移。

- 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。备份文件导入的数据来源分为OBS桶和Redis实例两种方式。
  - OBS桶导入方式：您需要先将源Redis的数据备份并下载，然后将备份数据文件上传到与DCS Redis实例同一租户下相同Region下的对象存储服务（OBS）中，DCS从对象存储服务（OBS）中读取备份数据，并将数据迁移到DCS Redis中。

    支持从其他云厂商Redis服务、自建Redis迁移到DCS Redis。
  - Redis实例导入方式：您需要先将源Redis的数据进行备份，然后可将源实例备份数据迁移到DCS Redis中。
- 在线迁移：在满足源Redis和目标Redis的网络相通、源Redis未禁用SYNC和PSYNC命令这两个前提下，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。

当前使用DCS控制台支持的迁移能力，如下表所示，您可以根据业务实际情况，选择迁移方式。

表 8-1 DCS 支持的迁移能力

迁移类型	源端	目标端：DCS服务		
		单机/主备	Proxy集群	Cluster集群
备份文件导入	AOF/RDB文件	√	√	√
在线迁移	DCS Redis：单机/主备	√	√	√

	DCS Redis：Proxy集群 **说明** Redis 3.0 proxy不支持作为源端迁移，4.0/5.0 proxy支持作为源端迁移。	√	√	√
	DCS Redis：Cluster集群	√	√	√
	自建Redis：单机/主备	√	√	√
	自建Redis：Proxy集群	√	√	√
	自建Redis：Cluster集群	√	√	√
	其他云Redis服务：单机/主备	√	√	√
	其他云Redis服务：Proxy集群	√	√	√
	其他云Redis服务：Cluster集群	√	√	√
	说明 源端**其他云Redis**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，使用在线迁移的方式，可以将源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。**迁移方案概览**			

说明

- **DCS Redis**，指的是分布式缓存服务的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。
- **其他云Redis服务**，指的是其他云厂商的Redis服务。
- √表示支持，×表示不支持。

8.2 备份文件导入方式

8.2.1 备份文件导入方式-OBS 桶

场景描述

当前DCS支持将备份数据通过DCS控制台迁移到DCS Redis。

您需要先将Redis数据备份下载到本地，然后将备份数据文件上传到与DCS Redis实例同一租户下相同Region下的OBS桶中，最后在DCS控制台创建迁移任务，DCS从OBS桶中读取数据，将数据迁移到DCS Redis中。

上传OBS桶的文件支持.aof、.rdb、.zip、.tar.gz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。

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

- OBS桶所在区域必须跟Redis目标实例所在区域相同。
- 上传的数据文件必须为.aof、.rdb、.zip、.tar.gz的格式，zip文件需包含aof或rdb文件。
- 如果是其他云厂商的单机版Redis和主备版Redis，您需要在备份页面创建备份任务，然后下载备份文件。
- 如果是其他云厂商的集群版Redis，在备份页面创建备份后会有多个备份文件，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择。

准备目标 Redis 实例

- 如果您还没有DCS Redis，请先创建，创建操作，请参考**创建Redis实例**。
- 如果您已有DCS Redis，则不需要重复创建，在迁移之前，您可以根据需要清空目标实例的已有数据。
  - 清空操作请参考**清空Redis实例数据**。
  - 如果没有清空目标实例数据，当目标实例存在与源Redis实例相同的key时，迁移后，会覆盖目标Redis实例原来的数据。
- 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。

创建 OBS 桶并上传备份文件

步骤1 创建OBS桶。

1. 登录OBS管理控制台，单击右上角的“创建桶”。
2. 在显示的“创建桶”页面，选择“区域”。
   OBS桶所在区域必须跟Redis目标实例所在区域相同。
3. 设置“桶名称”。
   桶名称的命名规则，请满足界面的要求。
4. 设置“存储类别”，当前支持“标准存储”、“温存储”和“冷存储”。
5. 设置“桶策略”，您可以为桶配置私有、公共读、或公共读写策略。
6. 设置“默认加密”。
7. 设置完成后，单击“立即创建”，等待OBS桶创建完成。

步骤2 通过OBS Browser+客户端，上传备份数据文件到OBS桶。

如果上传的备份文件较小，且不超过5GB，请执行**步骤3**，通过OBS控制台上传即可；

如果上传的备份文件大于5GB，请执行以下操作，需下载OBS Browser+客户端，安装并登录，创建OBS桶，然后上传备份文件。

1. 下载OBS Browser+客户端。

   具体操作，请参考《对象存储服务 工具指南 (OBS Browser+)》的“下载OBS Browser+”章节。

2. 安装OBS Browser+客户端。

   具体操作，请参考《对象存储服务 工具指南 (OBS Browser+)》的“安装OBS Browser+”章节。

3. 登录OBS Browser+客户端。

   具体操作，请参考《对象存储服务 工具指南 (OBS Browser+)》的“登录OBS Browser+”章节。

4. 创建桶。

   具体操作，请参考《对象存储服务 用户指南》的“控制台指南 > 入门 > 创建桶”章节。

5. 上传备份数据。

步骤3 通过OBS控制台，上传备份数据文件到OBS桶。

如果上传的备份文件较小，且不超过5GB，请执如下步骤：

1. 在OBS管理控制台的桶列表中，单击桶名称，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”对话框。
4. 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 8-1 上传对象

5. 可选：勾选“KMS加密”，用于加密上传文件。

6. 单击“上传”。

----结束

创建迁移任务

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，选择区域和项目。

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。

步骤6 在源实例区域，“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。

说明

上传的备份文件格式支持.aof、.rdb、.zip、.tar.gz，您可以上传任意其中一种。

图 8-2 备份文件导入

步骤7 在“备份文件”处单击“添加备份文件”，选择需要迁移的备份文件。

步骤8 在目标实例区域，选择**准备目标Redis实例**中创建的目标Redis。

步骤9 输入目标实例的密码，单击“测试连接”，测试密码是否符合要求。免密访问的实例，请直接单击“测试连接”。

步骤10 单击“立即创建”。

步骤11 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

8.2.2 备份文件导入方式-Redis 实例

场景描述

当前DCS支持将自建Redis的数据通过DCS控制台迁移到DCS Redis。

您需要先将自建Redis的数据进行备份，然后在DCS控制台创建迁移任务，将备份数据文件迁移到DCS 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

已创建主备或集群目标实例，且源实例已写入数据并备份成功。

步骤 1：获取源 Redis 实例名称及密码

获取准备迁移的源Redis实例名称。

步骤 2：准备目标 Redis 实例

- 如果您还没有DCS Redis，请先创建，创建操作，请参考**创建Redis实例**。
- 如果您已有DCS Redis，则不需要重复创建，在迁移之前，您可以根据需要清空实例数据。
  - 清空操作请参考**清空Redis实例数据**。
  - 如果没有清空目标实例数据，当目标实例存在与源Redis实例相同的key时，迁移后，会覆盖目标Redis实例原来的数据。

步骤 3：创建迁移任务

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 ，选择区域和项目。

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。

步骤6 “数据来源”选择“Redis实例”。

步骤7 在“源Redis实例”中选择**步骤1：获取源Redis实例名称及密码**中的Redis实例。

步骤8 在“备份记录”中选择需要迁移的备份文件。

步骤9 在“目标Redis实例”选择**步骤2：准备目标Redis实例**中创建的目标Redis。

步骤10 输入目标实例的密码，单击“测试连接”，测试密码是否正确。免密访问的实例，请直接单击“测试连接”。

步骤11 单击“立即创建”。

步骤12 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

8.3 在线迁移方式

场景描述

在满足源Redis和目标Redis的网络相通、源Redis未禁用SYNC和PSYNC命令这两个前提下，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。

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

- 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，否则迁移失败，选择DCS Redis实例进行在线迁移时，会自动放开SYNC命令。
- 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-slave-hard-limit和client-output-buffer-slave-soft-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

- 在迁移之前，请先阅读**使用DCS迁移介绍**，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/主备实例迁移到集群实例，由于目标集群实例只有一个DB，请先确保源Redis实例DB0以外的DB是否有数据，如果有，建议先将数据使用开源Rump工具迁移到DB0，否则会出现迁移失败，具体迁移操作请参考**使用Rump在线迁移**。

步骤 1：获取源 Redis 的信息

- 当源端为云服务Redis时，需获取准备迁移的源Redis实例的名称。
- 当源端为自建Redis时，需获取准备迁移的源Redis实例的IP和端口，或者域名和端口。

步骤 2：准备目标 Redis 实例

- 如果您还没有目标Redis，请先创建，创建操作，请参考**创建Redis实例**。
- 如果您已有目标Redis，则不需要重复创建，但在迁移之前，您需要清空实例数据。清空操作，请参考**清空Redis实例数据**。

  如果没有清空，如果存在与源Redis实例相同的key，迁移后，会覆盖目标Redis实例原来的数据。

步骤 3：检查网络

说明

- 配置在线迁移任务时，如果选择的源Redis或目标Redis为“云服务Redis”，则界面上要求所选云服务Redis必须与迁移任务处于相同的VPC，否则可能导致迁移任务无法连接所选云服务Redis实例。
- 特殊场景下，如果提前打通了迁移任务与所选云服务Redis实例间跨VPC访问，则可不用满足所选云服务Redis与迁移任务处于相同VPC的约束。

在创建在线迁移任务时，与源Redis、目标Redis间网络要求可参考表8-2。

表 8-2 在线迁移任务与源 Redis、目标 Redis 间网络要求

源Redis类型	目标Redis类型	创建在线迁移任务网络要求
云服务Redis	云服务Redis	创建在线迁移任务时，要求在线迁移任务与源Redis和目标Redis在同一个VPC，如果在线迁移任务与源Redis或目标Redis不在同一个VPC，则需要打通迁移任务与源Redis或目标Redis间的跨网络访问。如需打通跨网络访问，请参考《虚拟私有云用户指南》的“对等连接”章节，查看和创建对等连接。
云服务Redis	自建Redis	创建在线迁移任务时，要求在线迁移任务与源Redis在同一个VPC，然后再单独打通迁移任务与目标端自建Redis间的跨网络访问。 如需打通跨网络访问，请参考《虚拟私有云用户指南》的“对等连接”章节，查看和创建对等连接
自建Redis	云服务Redis	创建在线迁移任务时，要求在线迁移任务与目标Redis在同一个VPC，然后再单独打通迁移任务与源端自建Redis间的跨网络访问。 如需打通跨网络访问，请参考《虚拟私有云用户指南》的“对等连接”章节，查看和创建对等连接
自建Redis	自建Redis	创建在线迁移任务后，需要分别打通迁移任务与源端自建Redis、目标端自建Redis间的跨网络访问。 如需打通跨网络访问，请参考《虚拟私有云用户指南》的“对等连接”章节，查看和创建对等连接

步骤 4：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。

步骤2 在管理控制台左上角单击 ，选择区域和项目。

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”。进入创建在线迁移任务页面。

步骤5 设置迁移任务名称和描述。

步骤6 配置在线迁移任务虚拟机资源的VPC、子网和安全组。

创建在线迁移任务时，需要选择迁移虚拟机资源的VPC和安全组，并确保迁移资源能访问源Redis和目标Redis实例。

须知

- 创建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“迁移机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。

步骤7 单击“立即创建”。

步骤8 单击“提交”，创建在线迁移任务成功。

----结束

配置在线迁移任务

步骤1 创建完在线迁移任务之后，在“在线迁移”的列表，单击“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。

从其他云Redis到DCS Redis的数据迁移，支持全量迁移+增量迁移，全量迁移及增量迁移的功能及限制如表8-3所示。

表 8-3 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 8-3 选择迁移方法

步骤3 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”。

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。

步骤5 分别配置源Redis和目标Redis。

1. Redis类型，支持“云服务Redis”和“自建Redis”，需要根据迁移场景选择数据来源。
   - 云服务Redis：DCS Redis实例，需要选择与迁移任务处于相同VPC的DCS Redis服务。
   - 自建Redis：其他云厂商、本地数据中心自行搭建的Redis，需要输入Redis的连接地址。

   说明

   当源Redis和目标Redis属于DCS不同Region，则打通网络后，目标Redis实例无论是自建Redis或DCS Redis实例，在“目标Redis实例”区域，只能选中自建Redis，输入实例相关信息。
2. 如果是密码访问模式实例，在输入连接实例密码后，您可以单击密码右侧的“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”。

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”。

说明

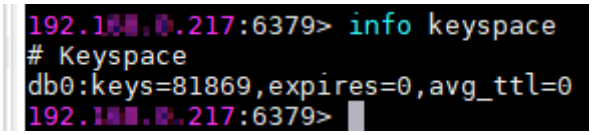
- 如果是增量迁移，迁移任务会在迁移开始后，一直保持迁移中状态，直到您在“操作”列单击“停止”，手动停止迁移。
- 数据迁移后，目标端与源端重复的Key会被覆盖。

----结束

迁移后验证

迁移完成后，请使用redis-cli连接源Redis和目标Redis，确认数据的完整性。

1. 连接源Redis和目标Redis。
2. 输入info keyspace，查看keys参数和expires参数的值。



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。

8.4 交换 DCS 实例 IP

场景描述

实例规格变更目前只支持同类型实例间的扩容和缩容，不支持跨实例类型的变更。因此可以通过“数据迁移+交换IP”方式实现跨实例类型的规格变更。同时，还可通过该方式更改实例可用区。

- 通过在线迁移方式将数据迁移之后，交换两个实例的IP。
- 交换IP后支持回滚功能。

说明

- Redis 4.0及以上版本的实例支持实例交换IP。
- 只有源实例和目标实例都为分布式缓存服务Redis实例才支持实例交换IP。

前提条件

- 获取源实例及目标实例信息，可参考**准备目标Redis实例**准备目标实例。
- 参考**检查网络**确保源实例和目标实例网络互通。
- 创建的目标实例端口需要与源实例保持一致。
- 进行实例交换IP满足的条件为：
  - 源实例和目标实例都为分布式缓存服务Redis实例。
  - 交换IP支持的能力如下**表8-4**。

表 8-4 交换 ip 能力

源端	目标端
单机/主备/Proxy集群	单机/主备/Proxy集群

交换 IP 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。
3. 创建的目标端实例端口需要与源实例端口保持一致。
4. 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。
5. 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。
6. 如果源端是主备实例，交换IP时不会交换备节点IP，请确保应用中没有直接引用备节点IP。
7. 如果应用中有直接引用域名，请选择交换域名，否则通过域名会连接到源实例。
8. 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。
9. 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。

交换 IP 操作步骤

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“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。

步骤6 配置在线迁移任务虚拟机资源的VPC、子网和安全组。

创建在线迁移任务时，需要选择迁移虚拟机资源的VPC和安全组，并确保迁移资源能访问源Redis和目标Redis实例。

步骤7 参考**配置在线迁移任务**配置迁移任务，此处迁移方式只能选择“全量迁移+增量迁移”。

步骤8 在“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9 在交换IP弹框中，在交换域名区域，选择是否交换域名。

说明

- 如果使用域名，则必须要选择交换域名，否则客户端应用需要修改使用的域名。
- 如果没有使用域名，则直接更新两个实例的DNS。

步骤10 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。

----结束

回滚 IP 操作步骤

若您想将实例IP切换成原始的IP，请执行以下操作。

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“数据迁移”。

步骤4 在“在线迁移”页面，迁移任务状态为“IP交换成功”，单击操作列的“更多 > 回滚IP”。

步骤5 在确认框中，单击“确定”，IP回滚任务提交成功。当任务状态显示为“IP回滚成功”表示回滚任务完成。

----结束

9 密码管理

9.1 关于实例连接密码的说明

DCS的缓存实例提供了密码控制访问功能，确保缓存数据足够安全。

实例连接密码可以在创建实例时增加，或者在实例创建后，通过本章节的**重置缓存实例密码**操作设置实例连接密码。

综合安全与便利考虑，您可自行选择是否开启免密访问。

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32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（ `~!@#$^&*()-_=+\|{},<.>/? ）

安全使用实例密码

1. Redis-cli连接时隐藏密码。

   Linux操作系统中，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。

9.2 修改缓存实例密码

DCS管理控制台支持修改DCS缓存实例的密码。

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DCS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（长连接断开重连时才需要使用新密码，断开前还可以继续使用旧密码）。

前提条件

已成功创建DCS缓存实例。

操作步骤

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 ，选择区域和项目。

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 在需要修改密码的DCS缓存实例右侧，单击“操作”栏下的“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，其他操作不受影响。

DCS账号密码必须满足以下复杂度要求：

- 密码不能为空。
- 新密码不能和旧密码相同。
- 密码长度在8到32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

9.3 重置缓存实例密码

当您忘记了DCS缓存实例密码时，可通过DCS重置密码功能，重新设置一个密码，可使用新密码使用DCS缓存实例。

说明

- Redis支持通过重置密码功能将密码模式修改为免密模式，或者将免密模式修改为密码模式，具体请参考**修改Redis实例的访问方式**章节。
- 只有处于“运行中”状态的DCS缓存实例支持重置密码。
- 重置密码后，服务端无需重启，立即生效。客户端需使用重置后的密码才能连接（长连接断开重连时才需要使用新密码，断开前还可以继续使用旧密码）。

前提条件

已成功创建DCS缓存实例。

操作步骤

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 ，选择区域和项目。

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 在需要重置密码的DCS缓存实例右侧，单击“操作”栏下的“更多 > 重置密码”。

步骤5 系统弹出重置密码对话框。输入“新密码”和“确认密码”。

说明

DCS账号密码必须满足以下复杂度要求：

- 密码不能为空。
- 密码长度在8到32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”完成密码重置。

说明

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。

----结束

9.4 修改 Redis 实例的访问方式

使用场景

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

- 当您需要通过免密访问模式连接Redis实例，可通过开启Redis实例的免密访问功能，清空Redis实例的密码。

说明

- 只有处于“运行中”状态的Redis实例支持修改访问方式。
- 免密模式存在安全风险，之后您可以通过重置密码进行密码设置。

操作步骤

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 ，选择区域和项目。

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 在需要修改访问方式的Redis实例右侧，单击“操作”栏下的“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

10 参数模板

10.1 查看参数模板信息

本节介绍如何在分布式缓存服务管理控制台查看参数模板的详细信息。

操作步骤

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 ，选择区域和项目。

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10-1**。

表 10-1 Redis 缓存实例配置参数说明

参数名	参数解释	取值范围	默认值
active-expire-num	过期键定期删除时随机检查key的数量。 Redis 4.0及以上版本的实例支持该参数。	1～1,000	20
timeout	客户端空闲N秒（timeout参数的取值）后将关闭连接。当N=0时，表示禁用该功能。 Proxy集群实例不支持该参数。	0~7200 单位：秒	0

参数名	参数解释	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。 Redis支持三种不同的调用fsync的方式： 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。 单机实例不支持该参数。	• no • always • everysec	no
appendonly	指定是否在每次更新操作后进行日志记录（即持久化功能），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。 yes：开启。 no：关闭。 单机实例不支持该参数。	• yes • no	yes
client-output-buffer-limit-slave-soft-seconds	slave客户端output-buffer超过client-output-buffer-slave-soft-limit设置的大小，并且持续时间超过此值时，服务端会主动断开连接。 单机实例不支持该参数。	0~60 单位：秒	60
client-output-buffer-slave-hard-limit	对slave客户端output-buffer的硬限制（单位为字节），如果slave客户端output-buffer大于此值，服务端会主动断开连接。 单机实例不支持该参数。	取值范围与实例的类型及规格有关	默认值与实例的类型及规格有关

参数名	参数解释	取值范围	默认值
client-output-buffer-slave-soft-limit	对slave客户端output-buffer的软限制（单位为字节），如果output-buffer大于此值并且持续时间超过client-output-buffer-limit-slave-soft-seconds设置的时长，服务端会主动断开连接。 单机实例不支持该参数。	取值范围与实例的类型及规格有关	默认值与实例的类型及规格有关
maxmemory-policy	在达到内存上限（maxmemory）时DCS将如何选择要删除的内容。有8个取值供选择： 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。	取值范围与实例的版本有关	默认值与实例的版本及类型有关
lua-time-limit	Lua脚本的最长执行时间。	100～5,000 单位：毫秒	5,000
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。	• yes • no	no

参数名	参数解释	取值范围	默认值
maxclients	最大同时连接的客户端个数。 Proxy集群实例不支持该参数。	取值范围与实例的类型及规格有关	默认值与实例的类型及规格有关
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
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。 单机实例不支持该参数。	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。 单机实例不支持该参数。	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表中只有少量记录时，使用有利于节约内存的数据结构来对hashes进行编码。	1~10000	512
hash-max-ziplist-value	当hash表中最大的取值不超过预设阈值时，使用有利于节约内存的数据结构来对hashes进行编码。	1~10000	64
set-max-intset-entries	当一个集合仅包含字符串且字符串为在64位有符号整数范围内的十进制整数时，使用有利于节约内存的数据结构对集合进行编码。	1~10000	512
zset-max-ziplist-entries	当有序集合中只有少量记录时，使用有利于节约内存的数据结构对有序序列进行编码。	1~10000	128

参数名	参数解释	取值范围	默认值
zset-max-ziplist-value	当有序集合中的最大取值不超过预设阈值时，使用有利于节约内存的数据结构对有序集合进行编码。	1~10000	64
latency-monitor-threshold	延时监控的采样时间阈值（最小值）。 阈值设置为0：不做监控，也不采样。 阈值设置为大于0：将记录执行耗时大于阈值的操作。 可以通过LATENCY等命令获取统计数据和配置、执行采样监控。 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	取值范围	默认值
notify-keyspace-events	notify-keyspace-events选项的参数为空字符串时，功能关闭。另一方面，当参数不是空字符串时，功能开启。notify-keyspace-events的参数可以是以下字符的任意组合，它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不能与g$lshzxe同时出现，不能出现相同字母。举个例子，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。 Proxy集群实例不支持该参数。	请参考该参数的描述。	Ex

参数名	参数解释	取值范围	默认值
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用于配置记录到慢查询的命令执行时间阈值。	0～1,000,000 单位：微秒	10,000
slowlog-max-len	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

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10-1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

10.2 创建自定义参数模板

您可以根据业务需要创建不同缓存版本和缓存类型的自定义参数模板，可以创建多个自定义参数模板。

操作步骤

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 ，选择区域和项目。

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”栏下的“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”栏下的“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10-2，一般情况下，按照系统默认值设置参数即可。

表 10-2 Redis 缓存实例配置参数说明

参数名	参数解释	取值范围	默认值
active-expire-num	过期键定期删除时随机检查key的数量。 Redis 4.0及以上版本的实例支持该参数。	1～1,000	20
timeout	客户端空闲N秒（timeout参数的取值）后将关闭连接。当N=0时，表示禁用该功能。 Proxy集群实例不支持该参数。	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。 Redis支持三种不同的调用fsync的方式： 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。 单机实例不支持该参数。	• no • always • everysec	no
appendonly	指定是否在每次更新操作后进行日志记录（即持久化功能），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。 yes：开启。 no：关闭。 单机实例不支持该参数。	• yes • no	yes

参数名	参数解释	取值范围	默认值
client-output-buffer-limit-slave-soft-seconds	slave客户端output-buffer超过client-output-buffer-slave-soft-limit设置的大小，并且持续时间超过此值时，服务端会主动断开连接。 单机实例不支持该参数。	0~60 单位：秒	60
client-output-buffer-slave-hard-limit	对slave客户端output-buffer的硬限制（单位为字节），如果slave客户端output-buffer大于此值，服务端会主动断开连接。 单机实例不支持该参数。	取值范围与实例的类型及规格有关	默认值与实例的类型及规格有关
client-output-buffer-slave-soft-limit	对slave客户端output-buffer的软限制（单位为字节），如果output-buffer大于此值并且持续时间超过client-output-buffer-limit-slave-soft-seconds设置的时长，服务端会主动断开连接。 单机实例不支持该参数。	取值范围与实例的类型及规格有关	默认值与实例的类型及规格有关

参数名	参数解释	取值范围	默认值
maxmemory-policy	在达到内存上限（maxmemory）时DCS将如何选择要删除的内容。有8个取值供选择： 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。	取值范围与实例的版本有关	默认值与实例的版本及类型有关
lua-time-limit	Lua脚本的最长执行时间。	100～5,000 单位：毫秒	5,000
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。	• yes • no	no
maxclients	最大同时连接的客户端个数。 Proxy集群实例不支持该参数。	取值范围与实例的类型及规格有关	默认值与实例的类型及规格有关
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	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。 单机实例不支持该参数。	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。 单机实例不支持该参数。	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表中只有少量记录时，使用有利于节约内存的数据结构来对hashes进行编码。	1~10000	512
hash-max-ziplist-value	当hash表中最大的取值不超过预设阈值时，使用有利于节约内存的数据结构来对hashes进行编码。	1~10000	64
set-max-intset-entries	当一个集合仅包含字符串且字符串为在64位有符号整数范围内的十进制整数时，使用有利于节约内存的数据结构对集合进行编码。	1~10000	512
zset-max-ziplist-entries	当有序集合中只有少量记录时，使用有利于节约内存的数据结构对有序序列进行编码。	1~10000	128
zset-max-ziplist-value	当有序集合中的最大取值不超过预设阈值时，使用有利于节约内存的数据结构对有序集合进行编码。	1~10000	64

参数名	参数解释	取值范围	默认值
latency-monitor-threshold	延时监控的采样时间阈值（最小值）。 阈值设置为0：不做监控，也不采样。 阈值设置为大于0：将记录执行耗时大于阈值的操作。 可以通过LATENCY等命令获取统计数据和配置、执行采样监控。 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	取值范围	默认值
notify-keyspace-events	notify-keyspace-events选项的参数为空字符串时，功能关闭。另一方面，当参数不是空字符串时，功能开启。notify-keyspace-events的参数可以是以下字符的任意组合，它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不能与g$lshzxe同时出现，不能出现相同字母。举个例子，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。 Proxy集群实例不支持该参数。	请参考该参数的描述。	Ex

参数名	参数解释	取值范围	默认值
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用于配置记录到慢查询的命令执行时间阈值。	0～1,000,000 单位：微秒	10,000
slowlog-max-len	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

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，因此不支持修改该四个参数。
2. 表10-2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。
3. 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。
4. notify-keyspace-events参数的其他描述：
   - 有效值为[K|E|KE][A|g|l|s|h|z|x|e|$]，即输入的参数中至少要有一个K或者E。
   - A为“g$lshzxe”所有参数的集合别名。A与“g$lshzxe”中任意一个不能同时出现。
   - 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。若将参数设为字符串"AKE"表示发送所有类型的通知。

步骤8 单击“确定”，完成创建自定义参数模板。

----结束

10.3 修改自定义参数模板

您可以根据业务需要修改自定义参数模板的名称、描述和配置参数。

操作步骤

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 ，选择区域和项目。

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 可以通过两种方式修改自定义参数模板。

- 单击需要修改的自定义模板右侧“操作”栏下的“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10-3，一般情况下，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10-3，一般情况下，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。

表 10-3 Redis 缓存实例配置参数说明

参数名	参数解释	取值范围	默认值
active-expire-num	过期键定期删除时随机检查key的数量。 Redis 4.0及以上版本的实例支持该参数。	1～1,000	20
timeout	客户端空闲N秒（timeout参数的取值）后将关闭连接。当N=0时，表示禁用该功能。 Proxy集群实例不支持该参数。	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。 Redis支持三种不同的调用fsync的方式： 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。 单机实例不支持该参数。	• no • always • everysec	no

参数名	参数解释	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（即持久化功能），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。 yes：开启。 no：关闭。 单机实例不支持该参数。	• yes • no	yes
client-output-buffer-limit-slave-soft-seconds	slave客户端output-buffer超过client-output-buffer-slave-soft-limit设置的大小，并且持续时间超过此值时，服务端会主动断开连接。 单机实例不支持该参数。	0~60 单位：秒	60
client-output-buffer-slave-hard-limit	对slave客户端output-buffer的硬限制（单位为字节），如果slave客户端output-buffer大于此值，服务端会主动断开连接。 单机实例不支持该参数。	取值范围与实例的类型及规格有关	默认值与实例的类型及规格有关
client-output-buffer-slave-soft-limit	对slave客户端output-buffer的软限制（单位为字节），如果output-buffer大于此值并且持续时间超过client-output-buffer-limit-slave-soft-seconds设置的时长，服务端会主动断开连接。 单机实例不支持该参数。	取值范围与实例的类型及规格有关	默认值与实例的类型及规格有关

参数名	参数解释	取值范围	默认值
maxmemory-policy	在达到内存上限（maxmemory）时DCS将如何选择要删除的内容。有8个取值供选择： 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。	取值范围与实例的版本有关	默认值与实例的版本及类型有关
lua-time-limit	Lua脚本的最长执行时间。	100～5,000 单位：毫秒	5,000
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。	• yes • no	no
maxclients	最大同时连接的客户端个数。 Proxy集群实例不支持该参数。	取值范围与实例的类型及规格有关	默认值与实例的类型及规格有关
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	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。 单机实例不支持该参数。	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。 单机实例不支持该参数。	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表中只有少量记录时，使用有利于节约内存的数据结构来对hashes进行编码。	1~10000	512
hash-max-ziplist-value	当hash表中最大的取值不超过预设阈值时，使用有利于节约内存的数据结构来对hashes进行编码。	1~10000	64
set-max-intset-entries	当一个集合仅包含字符串且字符串为在64位有符号整数范围内的十进制整数时，使用有利于节约内存的数据结构对集合进行编码。	1~10000	512
zset-max-ziplist-entries	当有序集合中只有少量记录时，使用有利于节约内存的数据结构对有序序列进行编码。	1~10000	128
zset-max-ziplist-value	当有序集合中的最大取值不超过预设阈值时，使用有利于节约内存的数据结构对有序集合进行编码。	1~10000	64

参数名	参数解释	取值范围	默认值
latency-monitor-threshold	延时监控的采样时间阈值（最小值）。 阈值设置为0：不做监控，也不采样。 阈值设置为大于0：将记录执行耗时大于阈值的操作。 可以通过LATENCY等命令获取统计数据和配置、执行采样监控。 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	取值范围	默认值
notify-keyspace-events	notify-keyspace-events选项的参数为空字符串时，功能关闭。另一方面，当参数不是空字符串时，功能开启。notify-keyspace-events的参数可以是以下字符的任意组合，它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不能与g$lshzxe同时出现，不能出现相同字母。举个例子，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。 Proxy集群实例不支持该参数。	请参考该参数的描述。	Ex

参数名	参数解释	取值范围	默认值
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用于配置记录到慢查询的命令执行时间阈值。	0～1,000,000 单位：微秒	10,000
slowlog-max-len	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

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，因此不支持修改该四个参数。
2. 表10-3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。
3. 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。
4. notify-keyspace-events参数的其他描述：
   - 有效值为[K|E|KE][A|g|l|s|h|z|x|e|$]，即输入的参数中至少要有一个K或者E。
   - A为“g$lshzxe”所有参数的集合别名。A与“g$lshzxe”中任意一个不能同时出现。
   - 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。若将参数设为字符串"AKE"表示发送所有类型的通知。

----结束

10.4 删除自定义参数模板

您可以根据需要删除自定义参数模板。

操作步骤

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，选择区域和项目。

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 单击需要删除的自定义模板右侧“操作”栏下的“删除”。

步骤6 单击“是”，完成删除自定义参数模板。

----结束

11 监控

11.1 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。

表 11-1 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
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

监控指标的维度请参考**维度**。

表 11-2 Redis 3.0 实例支持的监控指标

指标ID	指标名称	含义	取值范围	进制	测量对象（维度）	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。	0～100 单位：% 	不涉及	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（内存利用率统计是扣除预留内存的）。	0～100 单位：% 	不涉及	Redis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。	>= 0 单位：byte/s 	1024(IEC)	Redis实例	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。	>= 0 单位：byte/s 	1024(IEC)	Redis实例	1分钟
node_status	实例节点状态	实例节点状态，状态正常时为0，异常时为1。	-	不涉及	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，不包括来自从节点的连接。	>=0	不涉及	Redis实例	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	不涉及	Redis实例	1分钟

指标ID	指标名称	含义	取值范围	进制	测量对象（维度）	监控周期（原始指标）
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。	>=0 单位：byte	1024(IEC)	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。阻塞操作如BLPOP，BRPOP，BRPOPLPUSH。	>=0	不涉及	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。	>=0 单位：byte	1024(IEC)	Redis实例	1分钟
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。包含和堆，但不包括换出的内存。	>=0 单位：byte	1024(IEC)	Redis实例	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。	>=0 单位：byte	1024(IEC)	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。	>=0 单位：byte	1024(IEC)	Redis实例	1分钟
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 单位：%	不涉及	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。	>=0	不涉及	Redis实例	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	不涉及	Redis实例	1分钟

指标ID	指标名称	含义	取值范围	进制	测量对象（维度）	监控周期（原始指标）
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	不涉及	Redis实例	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。	>=0 单位：byte	1024(IEC)	Redis实例	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。	>=0 单位：byte	1024(IEC)	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。	>=0 单位：KiB/s	1024(IEC)	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。	>=0 单位：KiB/s	1024(IEC)	Redis实例	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	不涉及	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。	>=0	不涉及	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。	>=0	不涉及	Redis实例	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	不涉及	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。	>=0	不涉及	Redis实例	1分钟

指标ID	指标名称	含义	取值范围	进制	测量对象（维度）	监控周期（原始指标）
pubsub_channels	Pubsub通道个数	该指标用于统计Pub/Sub通道个数。	>=0	不涉及	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	不涉及	Redis实例	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)	0～100 单位：%	不涉及	Redis实例	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	不涉及	Redis实例	1分钟
auth_errors	认证失败次数	统计实例的认证失败次数。	>=0	不涉及	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、bgrewriteaof命令导致的慢日志。	• 1：表示存在 • 0：表示不存在。	不涉及	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	不涉及	Redis实例（单机/主备）	1分钟

Redis 4.0 及以上版本实例监控指标

说明

- 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11-3**中“指标含义”的说明。
- 监控指标的维度请参考**维度**。

表 11-3 Redis 4.0 及以上版本实例支持的监控指标

指标ID	指标名称	含义	取值范围	进制	测量对象（维度）	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。	0～100 单位：% 	不涉及	Redis实例（单机/主备）	1分钟
command_max_delay	命令最大时延	统计实例的命令最大时延。	>=0 单位：ms	不涉及	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。	>=0	不涉及	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。	• 1：表示存在 • 0：表示不存在。	不涉及	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。	0～100 单位：% 	不涉及	Redis实例	1分钟

指标ID	指标名称	含义	取值范围	进制	测量对象（维度）	监控周期（原始指标）
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	不涉及	Redis实例	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) 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。	0～100 单位：% 	不涉及	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。	>= 0 单位：byte	1024(IEC)	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。	>= 0 单位：byte	1024(IEC)	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。	0～100 单位：%	不涉及	Redis实例	1分钟
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。包含和堆，但不包括换出的内存。	>= 0 单位：byte	1024(IEC)	Redis实例	1分钟

指标ID	指标名称	含义	取值范围	进制	测量对象（维度）	监控周期（原始指标）
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	不涉及	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	不涉及	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	不涉及	Redis实例	1分钟
rx_controlled	流控次数	统计周期内被流控的次数。	>=0	不涉及	Redis实例	1分钟
bandwidth_usage	带宽使用率	计算当前流量带宽（网络瞬时输入流量与网络瞬时输出流量的平均值）与最大带宽限制的百分比。	>=0 单位：% 	不涉及	Redis实例	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。	>=0 单位：% 	不涉及	Redis实例	1分钟
Instance Node Status	实例节点状态	实例节点状态，状态正常时为0，异常时为1。	-	不涉及	Redis实例	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。	>=0 单位：μs 	不涉及	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。	>=0 单位：μs 	不涉及	Redis实例（单机）	1分钟

指标ID	指标名称	含义	取值范围	进制	测量对象（维度）	监控周期（原始指标）
cpu_avg_usage	CPU平均使用率	该指标用于统计当前CPU平均使用率的百分比。	>=0 单位：% 	不涉及	Redis实例（单机/主备）	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	不涉及	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，不包括来自从节点的连接。	>= 0	不涉及	Redis实例	1分钟
del	DEL	该指标用于统计平均每秒del操作数。	0～500000 单位：Count/s	不涉及	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	不涉及	Redis实例	1分钟
expire	EXPIRE	该指标用于统计平均每秒expire操作数。	0～500000 单位：Count/s	不涉及	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	不涉及	Redis实例	1分钟

指标ID	指标名称	含义	取值范围	进制	测量对象（维度）	监控周期（原始指标）
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。	0～500000 单位：Count/s	不涉及	Redis实例	1分钟
hdel	HDEL	该指标用于统计平均每秒hdel操作数。	0～500000 单位：Count/s	不涉及	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。	0～500000 单位：Count/s	不涉及	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。	0～500000 单位：Count/s	不涉及	Redis实例	1分钟
hmset	HMSET	该指标用于统计平均每秒hmset操作数。	0～500000 单位：Count/s	不涉及	Redis实例	1分钟
hset	HSET	该指标用于统计平均每秒hset操作数。	0～500000 单位：Count/s	不涉及	Redis实例	1分钟

指标ID	指标名称	含义	取值范围	进制	测量对象（维度）	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。	>=0 单位：KiB/s	1024(IEC)	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。	>=0 单位：KiB/s	1024(IEC)	Redis实例	1分钟
memory_frag_ratio	内存碎片率	该指标用于统计当前的内存碎片率。	>= 0	不涉及	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。	0～500000 单位：Count/s	不涉及	Redis实例	1分钟
mset	MSET	该指标用于统计平均每秒mset操作数。	0～500000 单位：Count/s	不涉及	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	不涉及	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	不涉及	Redis实例	1分钟
set	SET	该指标用于统计平均每秒set操作数。	0～500000 单位：Count/s	不涉及	Redis实例	1分钟

指标ID	指标名称	含义	取值范围	进制	测量对象（维度）	监控周期（原始指标）
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。	>= 0 单位：byte	1024(IEC)	Redis实例	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。	>= 0 单位：byte	1024(IEC)	Redis实例	1分钟
sadd	SADD	该指标用于统计平均每秒sadd操作数。	0～500000 单位：Count/s	不涉及	Redis实例	1分钟
smembers	SMEMBERS	该指标用于统计平均每秒smembers操作数。	0～500000 单位：Count/s	不涉及	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。	>=0	不涉及	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。	>= 0 单位：byte	1024(IEC)	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。	0～100 单位：%	不涉及	Redis实例	1分钟

Redis 实例数据节点监控指标

说明

- Redis主备、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 11-4 实例中数据节点监控指标

指标ID	指标名称	含义	取值范围	进制	测量对象（维度）	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。	0～100 单位：% 	不涉及	Redis集群实例数据节点 Redis 4.0及以上版本主备实例数据节点	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。	0～100 单位：% 	不涉及	Redis集群实例数据节点 Redis 4.0及以上版本主备实例数据节点	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，不包括来自从节点的连接。	>=0	不涉及	Redis集群实例数据节点 Redis 4.0及以上版本主备实例数据节点	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	不涉及	Redis 4.0及以上版本主备、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。	>= 0 单位：byte	1024(IEC)	Redis 4.0及以上版本主备、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	不涉及	Redis集群实例数据节点 Redis 4.0及以上版本主备实例数据节点	1分钟

指标ID	指标名称	含义	取值范围	进制	测量对象（维度）	监控周期（原始指标）
used_memory	已用内存	该指标用于统计Redis已使用的内存字节数。	>= 0 单位：byte	1024(IEC)	Redis集群实例数据节点 Redis 4.0及以上版本主备实例数据节点	1分钟
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，包含和堆，但不包括换出的内存。	>= 0 单位：byte	1024(IEC)	Redis集群实例数据节点 Redis 4.0及以上版本主备实例数据节点	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。	>= 0 单位：byte	1024(IEC)	Redis集群实例数据节点 Redis 4.0及以上版本主备实例数据节点	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。	>= 0 单位：byte	1024(IEC)	Redis集群实例数据节点 Redis 4.0及以上版本主备实例数据节点	1分钟
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	不涉及	Redis集群实例数据节点 Redis 4.0及以上版本主备实例数据节点	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。	>=0	不涉及	Redis集群实例数据节点 Redis 4.0及以上版本主备实例数据节点	1分钟

指标ID	指标名称	含义	取值范围	进制	测量对象（维度）	监控周期（原始指标）
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	不涉及	Redis集群实例数据节点 Redis 4.0及以上版本主备实例数据节点	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	不涉及	Redis集群实例数据节点 Redis 4.0及以上版本主备实例数据节点	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。	>=0 单位：byte	1024(IEC)	Redis集群实例数据节点 Redis 4.0及以上版本主备实例数据节点	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。	>=0 单位：byte	1024(IEC)	Redis集群实例数据节点 Redis 4.0及以上版本主备实例数据节点	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。	>=0 单位：KiB/s	1024(IEC)	Redis集群实例数据节点 Redis 4.0及以上版本主备实例数据节点	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。	>=0 单位：KiB/s	1024(IEC)	Redis集群实例数据节点 Redis 4.0及以上版本主备实例数据节点	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	不涉及	Redis集群实例数据节点 Redis 4.0及以上版本主备实例数据节点	1分钟

指标ID	指标名称	含义	取值范围	进制	测量对象（维度）	监控周期（原始指标）
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	不涉及	Redis集群实例数据节点 Redis 4.0及以上版本主备实例数据节点	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	不涉及	Redis集群实例数据节点 Redis 4.0及以上版本主备实例数据节点	1分钟
pubsub_channels	Pubsub通道个数	该指标用于统计Pub/Sub通道个数。	>=0	不涉及	Redis集群实例数据节点 Redis 4.0及以上版本主备实例数据节点	1分钟
pubsub_patterns	Pubsub模式个数	该指标用于统计Pub/Sub模式个数。	>=0	不涉及	Redis集群实例数据节点 Redis 4.0及以上版本主备实例数据节点	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)	0～100 单位：% 	不涉及	Redis集群实例数据节点 Redis 4.0及以上版本主备实例数据节点	1分钟
command_max_delay	命令最大时延	统计节点的命令最大时延。	>=0 单位：ms	不涉及	Redis集群实例数据节点 Redis 4.0及以上版本主备实例数据节点	1分钟

指标ID	指标名称	含义	取值范围	进制	测量对象（维度）	监控周期（原始指标）
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命令导致的慢日志。	• 1：表示存在 • 0：表示不存在。	不涉及	Redis集群实例数据节点 Redis 4.0及以上版本主备实例数据节点	1分钟
keys	缓存键总数	该指标用于统计Redis缓存中键总数。	>=0	不涉及	Redis集群实例数据节点 Redis 4.0及以上版本主备实例数据节点	1分钟
sadd	SADD	该指标用于统计平均每秒sadd操作数。	0～500000 单位：Count/s	不涉及	Redis集群实例数据节点 Redis 4.0及以上版本主备实例数据节点	1分钟
smembers	SMEMBERS	该指标用于统计平均每秒smembers操作数。	0～500000 单位：Count/s	不涉及	Redis集群实例数据节点 Redis 4.0及以上版本主备实例数据节点	1分钟
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。	-	不涉及	Redis 4.0/Redis 5.0集群实例数据节点的**备节点**	1分钟

指标ID	指标名称	含义	取值范围	进制	测量对象（维度）	监控周期（原始指标）
del	DEL	该指标用于统计平均每秒del操作数。	0～500000 单位：Count/s	不涉及	Redis集群实例数据节点 Redis 4.0及以上版本主备实例数据节点	1分钟
expire	EXPIRE	该指标用于统计平均每秒expire操作数。	0～500000 单位：Count/s	不涉及	Redis集群实例数据节点 Redis 4.0及以上版本主备实例数据节点	1分钟
get	GET	该指标用于统计平均每秒get操作数。	0～500000 单位：Count/s	不涉及	Redis集群实例数据节点 Redis 4.0及以上版本主备实例数据节点	1分钟
hdel	HDEL	该指标用于统计平均每秒hdel操作数。	0～500000 单位：Count/s	不涉及	Redis集群实例数据节点 Redis 4.0及以上版本主备实例数据节点	1分钟
hget	HGET	该指标用于统计平均每秒hget操作数。	0～500000 单位：Count/s	不涉及	Redis集群实例数据节点 Redis 4.0及以上版本主备实例数据节点	1分钟
hmget	HMGET	该指标用于统计平均每秒hmget操作数。	0～500000 单位：Count/s	不涉及	Redis集群实例数据节点 Redis 4.0及以上版本主备实例数据节点	1分钟

指标ID	指标名称	含义	取值范围	进制	测量对象（维度）	监控周期（原始指标）
hmset	HMSET	该指标用于统计平均每秒hmset操作数。	0~500000 单位：Count/s	不涉及	Redis集群实例数据节点 Redis 4.0及以上版本主备实例数据节点	1分钟
hset	HSET	该指标用于统计平均每秒hset操作数。	0~500000 单位：Count/s	不涉及	Redis集群实例数据节点 Redis 4.0及以上版本主备实例数据节点	1分钟
mget	MGET	该指标用于统计平均每秒mget操作数。	0~500000 单位：Count/s	不涉及	Redis集群实例数据节点 Redis 4.0及以上版本主备实例数据节点	1分钟
mset	MSET	该指标用于统计平均每秒mset操作数。	0~500000 单位：Count/s	不涉及	Redis集群实例数据节点 Redis 4.0及以上版本主备实例数据节点	1分钟
set	SET	该指标用于统计平均每秒set操作数。	0~500000 单位：Count/s	不涉及	Redis集群实例数据节点 Redis 4.0及以上版本主备实例数据节点	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。	>=0 单位：Count	不涉及	Redis集群实例数据节点 Redis 4.0及以上版本主备实例数据节点	1分钟

指标ID	指标名称	含义	取值范围	进制	测量对象（维度）	监控周期（原始指标）
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 单位：% 	不涉及	Redis集群实例数据节点 Redis 4.0及以上版本主备实例数据节点	1分钟

Proxy 节点监控指标

说明

- **Proxy集群实例支持Proxy节点监控指标。**
- 监控指标的维度请参考**维度**。

表 11-5 Redis 3.0 Proxy 集群实例中 Proxy 节点监控指标

指标ID	指标名称	含义	取值范围	进制	测量对象（维度）	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。	0～100 单位：% 	不涉及	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。	0～100 单位：% 	不涉及	Proxy节点	1分钟
p_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。	>=0	不涉及	Proxy节点	1分钟

指标ID	指标名称	含义	取值范围	进制	测量对象（维度）	监控周期（原始指标）
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。	0~10000000 单位：Packet/s	不涉及	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。	0~10000000 单位：Packet/s	不涉及	Proxy节点	1分钟
max_rxkB_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。	>= 0 单位：KiB/s	1024(IEC)	Proxy节点	1分钟
max_txkB_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。	>= 0 单位：KiB/s	1024(IEC)	Proxy节点	1分钟
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。	0~10000000 单位：Packet/s	不涉及	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。	0~10000000 单位：Packet/s	不涉及	Proxy节点	1分钟
avg_rxkB_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。	>= 0 单位：KiB/s	1024(IEC)	Proxy节点	1分钟

指标ID	指标名称	含义	取值范围	进制	测量对象（维度）	监控周期（原始指标）
avg_txkB_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。	>= 0 单位：KiB/s	1024(IEC)	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 11-6 Memcached 实例支持的监控指标

指标ID	指标名称	含义	取值范围	进制	测量对象（维度）	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。	0～100 单位：%	不涉及	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。	0～100 单位：%	不涉及	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。	>= 0 单位：byte/s	1024(IEC)	Memcached实例	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。	>= 0 单位：byte/s	1024(IEC)	Memcached实例	1分钟

指标ID	指标名称	含义	取值范围	进制	测量对象（维度）	监控周期（原始指标）
mc_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，不包括来自从节点的连接。	>=0	不涉及	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。	>=0 单位：byte	1024(IEC)	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。包含和堆，但不包括换出的内存。	>=0 单位：byte	1024(IEC)	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。	>=0 单位：byte	1024(IEC)	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	不涉及	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	不涉及	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	不涉及	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	不涉及	Memcached实例	1分钟
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。	>=0 单位：byte	1024(IEC)	Memcached实例	1分钟

指标ID	指标名称	含义	取值范围	进制	测量对象（维度）	监控周期（原始指标）
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。	>=0 单位：byte	1024(IEC)	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。	>=0 单位：KiB/s	1024(IEC)	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。	>=0 单位：KiB/s	1024(IEC)	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	不涉及	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	不涉及	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	不涉及	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	不涉及	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	不涉及	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	不涉及	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	不涉及	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	不涉及	Memcached实例	1分钟

指标ID	指标名称	含义	取值范围	进制	测量对象（维度）	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	不涉及	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	不涉及	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	不涉及	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	不涉及	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	不涉及	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	不涉及	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	不涉及	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	不涉及	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	不涉及	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	不涉及	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	不涉及	Memcached实例	1分钟

指标ID	指标名称	含义	取值范围	进制	测量对象（维度）	监控周期（原始指标）
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	不涉及	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	不涉及	Memcached实例	1分钟
mc_auth_errors	认证失败次数	该指标用于统计认证失败次数。	>=0	不涉及	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	不涉及	Memcached实例	1分钟
mc_command_max_delay	命令最大时延	统计命令最大时延。	>=0 单位：ms	不涉及	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。	• 1：表示存在 • 0：表示不存在。	不涉及	Memcached实例	1分钟
mc_keyspace_hits_perc	访问命中率	统计实例的访问码命中率。	0～100 单位：%	不涉及	Memcached实例	1分钟

维度

Key	Value
dcs_instance_id	Redis实例
dcs_cluster_redis_node	数据节点
dcs_cluster_proxy_node	Proxy节点（Redis 3.0）
dcs_memcached_instance_id	Memcached实例

11.2 DCS 常用的监控指标

本章节主要列举Redis的常用监控指标。

表 11-7 常用监控指标说明

指标名称	说明
最大CPU使用率	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。 ● 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	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。 ● 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。 **须知** 内存利用率统计是扣除预留内存的。
活跃的客户端数量	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。 活跃的客户端数量上限，可以查看**实例规格**下对应实例类型的“连接数上限”。
每秒并发操作数	该指标统计的是瞬时的每秒处理的命令数。 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）”。
网络瞬时输入流量	该指标用于统计瞬时的输入数据流量。 ● 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	该指标用于统计瞬时的输出数据流量。 ● 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。

指标名称	说明
带宽使用率	该指标计算当前流量带宽与最大带宽限制的百分比。 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%
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**Redis实例慢查询**。

11.3 查看监控指标

您可以通过性能监控页面查看DCS的各种指标。

操作步骤

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 ，选择区域和项目。

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，这和在缓存实例信息页面“性能监控”页签内容一致。

----结束

11.4 必须配置的告警监控

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 11-8 Redis 实例配置告警的指标

指标名称	正常范围	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。 如果单机/主备实例，无法扩展CPU能力，需要考虑切换为集群实例。 该指标仅针对Proxy集群、单机、主备实例设置，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、主备实例设置，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：重要	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。 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。 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。

指标名称	正常范围	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）	0~10000	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。

Redis 实例数据节点告警策略

表 11-9 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
最大CPU使用率	0~100 单位：%	告警阈值：>90% 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。

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平均使用率	0~100 单位：%	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 单位：%	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。
活跃的客户端数量	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。
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。

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
带宽使用率	0~200 单位：%	告警阈值：>90% 连续触发次数：2 告警级别：重要	是	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。 对于单个节点带宽使用率上涨，需要分析是否有存在大key。 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。 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间隔期间，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。
流控次数	>=0	告警阈值：>0 连续触发次数：1 告警级别：紧急	是	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。

配置步骤

以配置**CPU利用率**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 ，选择区域和项目。

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 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。

步骤5 在实例监控指标页面中，找到指标名称为“CPU利用率”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. 设置告警策略和告警级别。
2. 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。

3. 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 用户指南 》的“使用告警功能>创建告警规则和告警通知”章节。
- 如果需要修改或停用所创建的告警，请参考《 云监控服务 用户指南 》的“使用告警功能>告警规则管理”章节。

----结束

12 云审计服务支持的关键操作

12.1 云审计服务支持的 DCS 操作列表

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。

本节主要介绍云审计服务支持的DCS操作列表。

表 12-1 云审计服务支持的 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

操作类型	资源类型	事件名称
提交启动实例请求	Redis	submitStartDCSInstanceRequest
清空实例	Redis	flushDCSInstance
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
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
更新实例标签	Redis	updateInstanceTag
修改白名单配置	Instance	modifyWhiteList

12.2 查看云审计日志

开启了云审计服务后，系统开始记录DCS资源的操作。云审计服务管理控制台保存最近7天的操作记录。本节介绍如何在云审计服务管理控制台查看最近7天的操作记录。

操作步骤

步骤1 登录管理控制台。

步骤2 在管理控制台左上角单击，选择区域和项目。

说明

此处请选择与您的应用服务相同的区域。

步骤3 单击页面上方的“服务列表”，选择“管理与部署 > 云审计服务”，进入云审计服务信息页面。

步骤4 单击左侧导航树的“事件列表”，进入事件列表信息页面。

步骤5 事件列表支持通过筛选来查询对应的操作事件。当前事件列表支持四个维度的组合查询，详细信息如下：

- 事件类型、事件来源、资源类型和筛选类型。

  在下拉框中选择查询条件。其中，事件来源选择“DCS”。

  其中筛选类型选择事件名称时，还需选择某个具体的事件名称。

  选择资源ID时，还需选择或者手动输入某个具体的资源ID。

  选择资源名称时，还需选择或手动输入某个具体的资源名称。
- 操作用户：在下拉框中选择某一具体的操作用户，此操作用户指用户级别，而非租户级别。
- 事件级别：可选项为“所有事件级别”、“normal”、“warning”、“incident”，只可选择其中一项。
- 起始时间、结束时间：可通过选择时间段查询操作事件。

步骤6 在需要查看的记录左侧，单击展开该记录的详细信息，展开记录如图12-1所示。

图 12-1 展开记录

步骤7 在需要查看的记录右侧，单击“查看事件”，弹出一个窗口，如图12-2所示，显示了该操作事件结构的详细信息。

图 12-2 查看事件


```
{
    "time": "04/16/2018 11:14:59 GMT+08:00",
    "user": {
        "name": "[redacted]",
        "id": "5b64419de7f54010ace3ba9d63ece3c4",
        "domain": {
            "name": "[redacted]",
            "id": "cc9c0076188843ea938743dd166c7fef"
        }
    },
    "request": {
        "name": "[redacted]",
        "description": "",
        "engine": "Redis",
        "engine_version": "3.0.7",
        "capacity": 2,
        "password": "******",
        "vpc_id": "47c7ec3a-181f-4c3d-930f-de255c330f8b",
        "security_group_id": "2907f342-5960-4513-b8a4-1ba31eceabc9",
        "subnet_id": "cb6cbaf3-ebf0-4e62-8793-570d2a6de24b",
        "available_zones": [
            "ae04cf9d61544df3806a3feeb401b204"
        ],
        "product_id": "00301-31100-0--0",
        "no_password_access": false,
        "maintain_begin": "02:00:00",
```


----结束

13 数据迁移指南

13.1 概述

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。

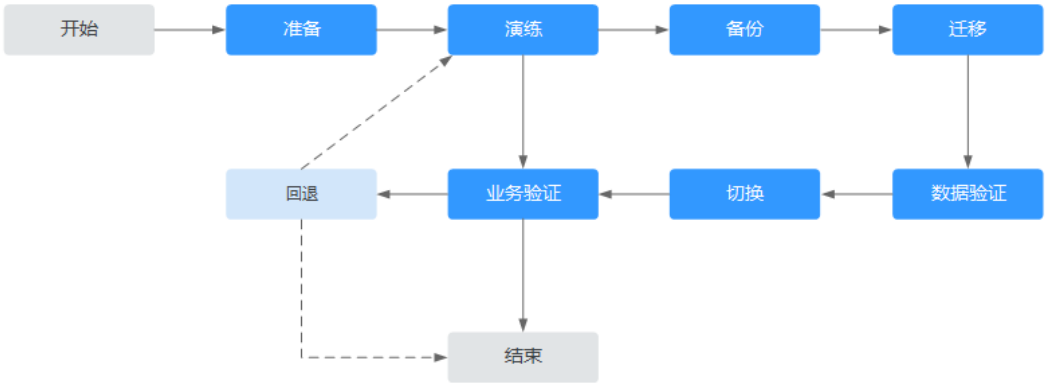
在迁移时需要分析业务系统使用到的缓存相关命令（附：**DCS命令兼容性说明参考**），在演练阶段对命令逐一验证。如有需要，可联系客服。

须知

- 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。

13.2 迁移流程介绍

图 13-1 迁移流程示意图



评估

获取当前待迁移的缓存数据信息（可参考**缓存数据信息**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实例配置（单机/主备/集群）
- 业务与各实例的连接关系

根据获取到的信息规划DCS缓存实例，包括：

- 申请缓存实例数量
- 各缓存实例的规格、类型（单机/主备/集群）
- 缓存实例与业务所属网络规划（VPC/子网/安全组）

说明

redis-cli -h ${redis_address} -p ${port}

- 查看数据分布情况，确认有数据的数据库编号以及各自的key数量。

  info keyspace

  查看各DB存储的key数量，并记录下来，供迁移验证对比。
- 查看数据占用空间，确认用于中转的ECS可用磁盘空间是否足够，实例规格与剩余可用内存是否足够。

  info memory

  参考**used_memory_human**的值。

准备

当完成迁移评估后，需要准备以下内容：

1. 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。
2. 网络资源

   按照业务规划创建虚拟私有云与子网。
3. 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。
4. 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。
5. 相关工具安装

   包括FTP工具、Redis迁移工具等。
6. 信息收集

   信息收集包括参与人员联系方式，服务器地址、登录信息，缓存实例信息与数据库信息等。

7. 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。

演练

演练的目的主要有以下：

1. 验证迁移工具与过程的可行。
2. 发掘迁移过程中遇到的问题，并做出有效的改进。
3. 评估迁移耗时。
4. 优化迁移步骤，验证部分工作并行的可行性，提高迁移效率。

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。

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。

数据验证

缓存数据的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

业务切换

1. 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。
2. 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。
3. 如果业务整体由数据中心或其他云厂商迁移到云服务，业务和缓存数据的迁移可并行。

业务验证

业务切换后建议验证内容包含以下：

1. 业务应用与DCS缓存实例的连通。
2. 通过业务操作对缓存数据的增删改查。
3. 如果条件满足，进行压测，确认性能满足业务峰值压力。

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 13-1 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	info memory命令查询得到，参考used_memory_human的值。 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 keyspace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，部分开源工具需要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。
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。

迁移源	信息项	说明
	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	-
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不同版本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**
...	...	*其他配置信息*。

13.3 迁移方案说明

迁移工具

表 13-2 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。	• 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。

工具/命令/服务	特点	说明
Redis-cli	• 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**。
RedisShake	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。
自行开发迁移脚本	灵活，根据实际情况适配。	-

迁移方案

说明

自建Redis，指的是在本服务、其他云厂商、本地数据中心自行搭建的Redis。

表 13-3 迁移方案

迁移场景	工具	迁移案例	迁移说明
自建Redis迁移至DCS	DCS控制台界面一键式迁移	• 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自建Redis**。 • 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	**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）	-
	RedisShake	**使用RedisShake工具在线迁移自建Redis Cluster集群**	-

迁移场景	工具	迁移案例	迁移说明
DCS实例间迁移	DCS控制台界面一键式迁移	低版本Redis实例迁移到高版本Redis实例，例如Redis 3.0迁移至Redis 6.0： • 如果源Redis实例和目标Redis实例的网络连通，推荐**使用迁移任务在线迁移DCS Redis实例**。 • 如果网络不连通，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，**建议高版本不要迁移到低版本，否则迁移失败**。
		不同Region的Redis实例迁移，推荐**使用备份文件离线迁移DCS Redis实例**。	由于DCS Redis实例默认是禁用了SYNC和PSYNC命令，在相同Region执行在线迁移时，会默认放通SYNC和PSYNC命令，但是在不同Region迁移时，没有放通该命令操作，所以无法使用在线迁移，推荐使用备份文件迁移。
		不同账号的Redis实例迁移，例如从账号A迁移到账号B： • 推荐**使用备份文件离线迁移DCS Redis实例**。 • 如果可以打通网络，也可以**使用迁移任务在线迁移DCS Redis实例**。	-
其他云厂商Redis服务迁移至DCS	DCS控制台界面一键式迁移	• 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。
	Rump	**使用Rump在线迁移其他云厂商Redis**	-
	RedisShake	**使用RedisShake离线迁移其他云厂商Redis**	-

迁移场景	工具	迁移案例	迁移说明
		使用RedisShake在线迁移其他云厂商Redis	-

13.4 自建 Redis 迁移至 DCS

13.4.1 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。

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配置为源端实例最大内存的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**。
- **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。**

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的源端实例迁移到集群实例，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**创建实例**。
- 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。

- **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《虚拟私有云用户指南》的“对等连接”章节，查看和创建对等连接，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《云专线服务用户指南》。

----结束

配置在线迁移任务

步骤1 单击 “继续配置”，配置在线迁移的源Redis、目标Redis等信息。

如果还没准备好配置资源，可以单击“立即创建”创建迁移任务，等配置资源准备好后在数据迁移页面单击需要配置的迁移任务右侧的“配置”，继续配置在线迁移任务。

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13-4所示。

表 13-4 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 13-2 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“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。

步骤6 单击“立即创建”。

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”。

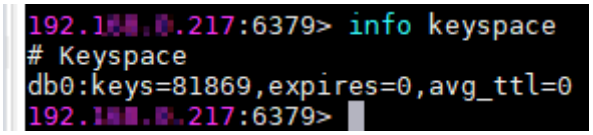
- 如果出现迁移失败，建议单击迁移任务名称，进入迁移任务详情页面，通过“迁移日志”排查迁移失败的原因。
- 如果是增量迁移，全量迁移后会一直处于增量迁移中的状态。
- 如需手动停止迁移中的任务，勾选迁移任务左侧的方框，单击迁移任务上方的“停止”，即可停止迁移。

----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**使用redis-cli连接Redis实例**。
2. 输入info keyspace，查看keys参数和expires参数的值。



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。

13.4.2 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**。
- 如果目标实例规格小于源实例规格，可能会导致离线迁移失败。

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的源端实例迁移到集群实例，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**创建实例**。
- 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

步骤1 创建OBS桶。

1. 登录OBS管理控制台，单击右上角的“创建桶”。
2. 在显示的“创建桶”页面，选择“区域”。

   OBS桶所在区域必须跟Redis目标实例所在区域相同。
3. 设置“桶名称”。

   桶名称的命名规则，请满足界面的要求。
4. 设置“存储类别”，当前支持“标准存储”、“温存储”和“冷存储”。
5. 设置“桶策略”，您可以为桶配置私有、公共读、或公共读写策略。
6. 设置“默认加密”。
7. 设置完成后，单击“立即创建”，等待OBS桶创建完成。

步骤2 通过OBS Browser+客户端，上传备份数据文件到OBS桶。

如果上传的备份文件较小，且不超过5GB，请执行**步骤3**，通过OBS控制台上传即可；

如果上传的备份文件大于5GB，请执行以下操作，需下载OBS Browser+客户端，安装并登录，创建OBS桶，然后上传备份文件。

1. 下载OBS Browser+客户端。

   具体操作，请参考《对象存储服务 工具指南 (OBS Browser+)》的“下载OBS Browser+”章节。
2. 安装OBS Browser+客户端。

   具体操作，请参考《对象存储服务 工具指南 (OBS Browser+)》的“安装OBS Browser+”章节。
3. 登录OBS Browser+客户端。

   具体操作，请参考《对象存储服务 工具指南 (OBS Browser+)》的“登录OBS Browser+”章节。
4. 创建桶。

   具体操作，请参考《对象存储服务 用户指南》的“控制台指南 > 入门 > 创建桶”章节。
5. 上传备份数据。

步骤3 通过OBS控制台，上传备份数据文件到OBS桶。

如果上传的备份文件较小，且不超过5GB，请执如下步骤：

1. 在OBS管理控制台的桶列表中，单击桶名称，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”对话框。
4. 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 13-3 上传对象

5. 可选：勾选“KMS加密”，用于加密上传文件。
6. 单击“上传”。

----结束

创建迁移任务

步骤1 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 “源数据”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 单击“添加备份文件”，选择需要迁移的备份文件。

图 13-4 备份文件导入

步骤7 在“目标数据”区域，选择**前提条件**中准备的“目标Redis实例”。

步骤8 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9 单击“立即创建”。

步骤10 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**使用redis-cli连接Redis实例**。
2. 输入info keyspace，查看keys参数和expires参数的值。

```
192.1[illegible].0.217:6379> info keyspace
# Keyspace
db0:keys=81869,expires=0,avg_ttl=0
192.1[illegible].0.217:6379>
```

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。

13.4.3 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。如果是需要通过OBS桶将源端备份数据迁移到DCS缓存实例，请参见**使用备份文件离线迁移自建Redis**。

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。

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**。
- 如果目标实例规格小于源实例规格，可能会导致离线迁移失败。
- 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。

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**创建实例**。
- 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《弹性云服务器用户指南》。

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
   cd src
   make
   ```
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。
   - 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 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ECS。

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**安全组配置和选择**。

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
```

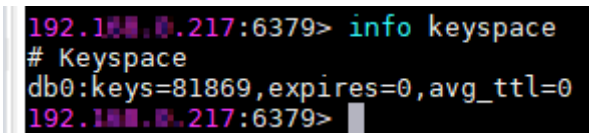
{dcs_instance_address}为目标Redis连接地址，{port}为目标Redis的端口，{password}为目标Redis的连接密码。

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。

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**使用redis-cli连接Redis实例**。
2. 输入info keyspace，查看keys参数和expires参数的值。



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。

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。

13.4.4 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。如果是需要通过OBS桶将源端备份数据迁移到DCS缓存实例，请参见**使用备份文件离线迁移自建Redis**。

约束与限制

- 建议选择业务量较少的时间段进行迁移。
- 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**创建实例**。
- 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《弹性云服务器用户指南》。
- **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
   cd src
   make
   ```

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 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ECS。

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**安全组配置和选择**。

导入数据

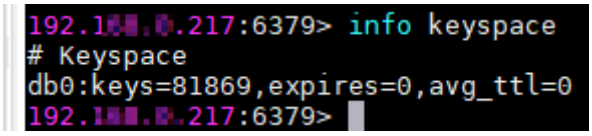
可借助RedisShake工具完成数据导入。

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**使用redis-cli连接Redis实例**。
2. 输入info keyspace，查看keys参数和expires参数的值。



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。

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。

13.4.5 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Redis Cluster集群的数据在线迁移到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**创建实例**。
- 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《弹性云服务器用户指南》。

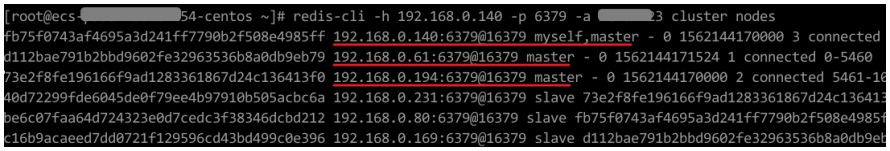
  ECS请选择与DCS Cluster集群实例相同虚拟私有云、子网和安全组，并且需要绑定弹性公网IP。
- 自建的源Redis Cluster集群如果是在本地或者其他云厂商的服务器上自建，需要允许被公网访问。

获取源 Redis 和目标 Redis 节点信息

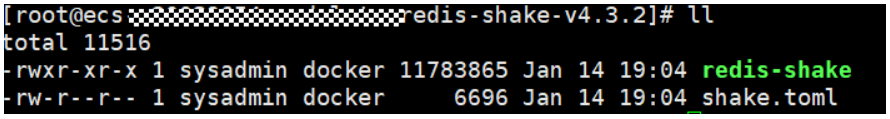
1. 分别连接源端和目标端Redis。连接Redis的方法请参考**使用redis-cli连接Redis实例**。

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。



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4.3.2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/v4.3.2/redis-shake-v4.3.2-linux-amd64.tar.gz
```
3. 执行命令解压RedisShake文件。
```
mkdir redis-shake-v4.3.2
tar -C redis-shake-v4.3.2 -xzvf redis-shake-v4.3.2-linux-amd64.tar.gz
```


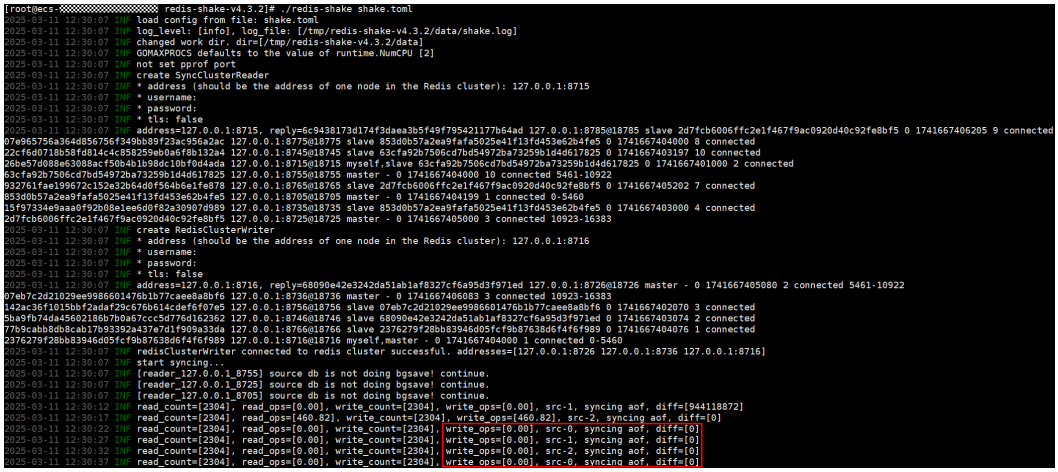
4. 执行命令进入解压后的文件目录。
```
cd redis-shake-v4.3.2
```
5. 编辑RedisShake工具配置文件shake.toml，补充源端与目标端信息。
```
vim shake.toml
```
修改内容如下：
```
[sync_reader]
#源端实例是Redis Cluster集群时，配置为true
cluster = true
#源端Cluster集群任意一个节点的IP地址与端口
address = {redis_ip}:{redis_port}
#如果无密码，本项不填
password = {source_redis_password}
[redis_writer]
#目标端实例是Redis Cluster集群时，配置为true
cluster = true
#目标端Cluster集群任意一个节点的IP地址与端口
address = {redis_ip}:{redis_port}
#如果无密码，本项不填
password = {target_redis_password}
```
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 shake.toml
```
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：
```
syncing aof
```
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）：

```
write_ops=[0.00], src-*, syncing aof, diff=[0]
```

图 13-5 RedisShake 在线迁移示意图



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**使用redis-cli连接Redis实例**。
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，建议及时清理RedisShake配置文件中的配置。

13.4.6 使用 RedisShake 工具离线迁移自建 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。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，或者源端实例部署在其他云厂商Redis服务中，无法实现在线迁移的场景。

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离线迁移到DCS Cluster集群。

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DCS Cluster集群Redis，创建Redis的方法，请参见**创建实例**。

  创建的目标Redis内存规格不能小于源Redis。
- 已创建弹性云服务器ECS，创建弹性云服务器的方法，请参见《弹性云服务器用户指南》。ECS请选择与DCS Cluster集群实例相同虚拟私有云、子网和安全组。

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**使用redis-cli连接Redis实例**。
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。

   ```
   [root@ecs-[redacted]54-centos ~]# redis-cli -h 192.168.0.140 -p 6379 -a [redacted]3 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 73e2f8fe196166f9ad1283361867d24c13641
   be6c07faa64d724323e0d7cedc3f38346dcbd212 192.168.0.80:6379@16379 slave fb75f0743af4695a3d241ff7790b2f508e4985f
   c16b9acaeed7dd0721f129596cd43bd499c0e396 192.168.0.169:6379@16379 slave d112bae791b2bbd9602fe32963536b8a0db9eb
   ```

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 release-v2.1.2-20220329.tar.gz
   ```

   ```
   [root@ecs-437a tem]# tar -xvf release-v2.1.2-20220329.tar.gz
   bin/redis-shake.conf
   bin/redis-shake.darwin
   bin/redis-shake.linux
   bin/redis-shake.windows
   [root@ecs-437a tem]# ll
   total 17960
   drwxr-xr-x 2 root root     4096 Apr 22 19:28 bin
   -rw-r--r-- 1 root root 18383025 Apr 22 19:21 release-v2.1.2-20220329.tar.gz
   ```

   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弹性云服务器。

导出备份文件

1. 执行命令进入解压后的RedisShake文件目录。

   ```
   cd bin
   ```

   ```
   [root@ecs-437a tem]# cd bin
   [root@ecs-437a bin]# ll
   total 34776
   -rw-r--r-- 1 502 games    13693 Mar 29  2022 redis-shake.conf
   -rwxr-xr-x 1 502 games 11740624 Mar 29  2022 redis-shake.darwin
   -rwxr-xr-x 1 502 games 11797281 Mar 29  2022 redis-shake.linux
   -rwxr-xr-x 1 502 games 12048384 Mar 29  2022 redis-shake.windows
   ```

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
   #如果无密码，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
#如果无密码，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**使用redis-cli连接Redis实例**。
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，建议及时清理RedisShake配置文件中的配置。

13.5 DCS 实例间迁移

13.5.1 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。

约束与限制

- 在线迁移不支持公网方式直接迁移。
- 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的源端实例迁移到集群实例，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-slave-hard-limit和client-output-buffer-slave-soft-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**。
- **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。**

前提条件

- 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。
- 如果您还没有目标Redis，请先创建，创建操作，请参考**创建实例**。
- 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

⚠ 注意

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。因此需要在源Redis相同账号和区域下创建在线迁移任务。否则无法进行在线迁移。

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。

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。

- 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《虚拟私有云用户指南》的“对等连接”章节创建对等连接，打通网络。
- 当源Redis和目标Redis属于DCS不同Region，请参考《云专线服务用户指南》，通过云专线打通网络。

----结束

配置在线迁移任务

步骤1 单击 “继续配置”，配置在线迁移的源Redis、目标Redis等信息。

如果还没准备好配置资源，可以单击“立即创建”创建迁移任务，等配置资源准备好后在数据迁移页面单击需要配置的迁移任务右侧的“配置”，继续配置在线迁移任务。

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13-5**所示。

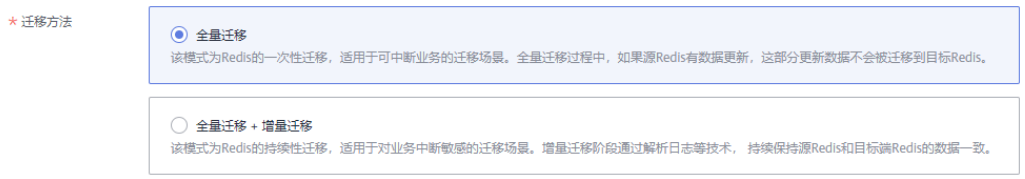
如果实例迁移后需要**交换DCS实例IP**，迁移方法必须选择“全量迁移+增量迁移”。

表 13-5 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**。

迁移类型	描述
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 13-6 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“源数据”和“目标数据”。

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。

   ⚠ 注意

   源Redis实例为DCS实例时，请不要选择“自建Redis”，否则无法启动迁移。

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”。

步骤6 单击“立即创建”。

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。
- 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。

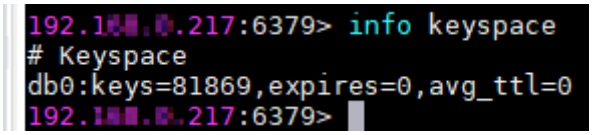
----结束

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**使用redis-cli连接Redis实例**。
2. 分别对源Redis和目标Redis执行**info keyspace**，查看keys参数和expires参数的值。

图 13-7 查看实例数据



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。

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及以上版本的实例支持实例交换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。

⚠ 注意

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。
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，请确保应用中没有直接引用备节点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。

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。

- 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

13.5.2 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。

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**。
- 如果目标实例规格小于源实例规格，可能会导致离线迁移失败。

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**手动备份实例**。
  - 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**创建实例**。

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。

使用备份文件离线迁移 DCS Redis 实例

- 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**。

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 源数据“数据来源”选择“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 在“备份记录”中选择需要迁移的备份文件。

步骤9 “目标Redis实例”请选择**前提条件**中准备的目标Redis。

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 单击“立即创建”。

步骤12 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。

- 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《对象存储服务 用户指南》的“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。如果上传的文件总大小超过5GB，请单击“上传对象”窗口上方的“超过5GB如何上传？”，根据参考文档操作。
6. 单击“上传”。

步骤3 进入分布式缓存服务管理控制台。

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“源数据”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。

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“目标数据”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 单击“立即创建”。

步骤12 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”。

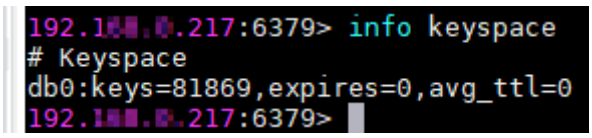
----结束

迁移后验证

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**使用redis-cli连接Redis实例**。
2. 分别对源Redis和目标Redis执行**info keyspace**，查看keys参数和expires参数的值。

   图 13-8 查看实例数据

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。

13.6 其他云厂商 Redis 迁移至 DCS

13.6.1 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。

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配置为源端实例最大内存的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**。
- **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。**

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的源端实例迁移到集群实例，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**创建实例**。
- 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。

- **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《云专线服务用户指南》。

----结束

配置在线迁移任务

步骤1 单击 “继续配置”，配置在线迁移的源Redis、目标Redis等信息。

如果还没准备好配置资源，可以单击“立即创建”创建迁移任务，等配置资源准备好后在数据迁移页面单击需要配置的迁移任务右侧的“配置”，继续配置在线迁移任务。

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13-6所示。

表 13-6 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**。
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 13-9 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。

步骤6 单击“立即创建”。

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”。

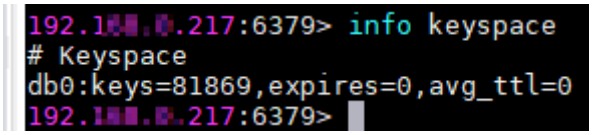
- 如果出现迁移失败，建议单击迁移任务名称，进入迁移任务详情页面，通过“迁移日志”排查迁移失败的原因。
- 如果是增量迁移，全量迁移后会一直处于增量迁移中的状态。
- 如需手动停止迁移中的任务，勾选迁移任务左侧的方框，单击迁移任务上方的“停止”，即可停止迁移。

----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**使用redis-cli连接Redis实例**。
2. 输入info keyspace，查看keys参数和expires参数的值。



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。

13.6.2 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**。
- 如果目标实例规格小于源实例规格，可能会导致离线迁移失败。

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的源端实例迁移到集群实例，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**创建实例**。
- 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

步骤1 创建OBS桶。

1. 登录OBS管理控制台，单击右上角的“创建桶”。
2. 在显示的“创建桶”页面，选择“区域”。

   OBS桶所在区域必须跟Redis目标实例所在区域相同。
3. 设置“桶名称”。

   桶名称的命名规则，请满足界面的要求。
4. 设置“存储类别”，当前支持“标准存储”、“温存储”和“冷存储”。
5. 设置“桶策略”，您可以为桶配置私有、公共读、或公共读写策略。
6. 设置“默认加密”。
7. 设置完成后，单击“立即创建”，等待OBS桶创建完成。

步骤2 通过OBS Browser+客户端，上传备份数据文件到OBS桶。

如果上传的备份文件较小，且不超过5GB，请执行**步骤3**，通过OBS控制台上传即可；

如果上传的备份文件大于5GB，请执行以下操作，需下载OBS Browser+客户端，安装并登录，创建OBS桶，然后上传备份文件。

1. 下载OBS Browser+客户端。

   具体操作，请参考《对象存储服务 工具指南 (OBS Browser+)》的“下载OBS Browser+”章节。
2. 安装OBS Browser+客户端。

   具体操作，请参考《对象存储服务 工具指南 (OBS Browser+)》的“安装OBS Browser+”章节。
3. 登录OBS Browser+客户端。

具体操作，请参考《对象存储服务 工具指南 (OBS Browser+)》的“登录OBS Browser+”章节。

4. 创建桶。

   具体操作，请参考《对象存储服务 用户指南》的“控制台指南 > 入门 > 创建桶”章节。

5. 上传备份数据。

步骤3 通过OBS控制台，上传备份数据文件到OBS桶。

如果上传的备份文件较小，且不超过5GB，请执如下步骤：

1. 在OBS管理控制台的桶列表中，单击桶名称，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”对话框。
4. 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 13-10 上传对象

5. 可选：勾选“KMS加密”，用于加密上传文件。
6. 单击“上传”。

----结束

创建迁移任务

步骤1 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 “源数据”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 单击“添加备份文件”，选择需要迁移的备份文件。

图 13-11 备份文件导入

步骤7 在“目标数据”区域，选择**前提条件**中准备的“目标Redis实例”。

步骤8 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9 单击“立即创建”。

步骤10 确认迁移信息，然后单击“提交”，开始创建迁移任务。

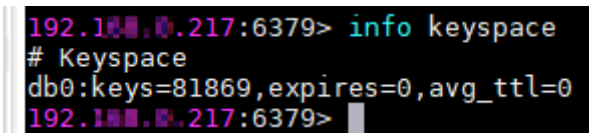
可返回迁移任务列表中，观察对应的迁移任务的状态，迁移成功后，任务状态显示“成功”。

----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**使用redis-cli连接Redis实例**。
2. 输入info keyspace，查看keys参数和expires参数的值。



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。

13.6.3 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/写值独立于关键工作。

以下是Rump的主要特性：

- 通过SCAN非阻塞式地获取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**创建实例**。
- 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《弹性云服务器用户指南》。

  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 ~]#
```

13.6.4 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

1. 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
   ```

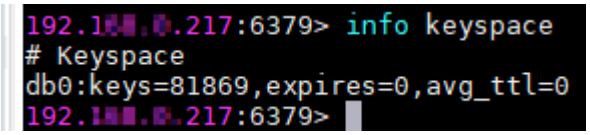
   - {target_redis_address}：目标Redis实例的连接地址。
   - {target_redis_port}：目标Redis实例的连接端口号。
   - {target_redis_password}：目标Redis实例的连接密码。

   在命令返回的结果中，获取所有master节点的IP端口，如下如所示：

   ```
   [root@ecs-[redacted]54-centos ~]# redis-cli -h 192.168.0.140 -p 6379 -a [redacted]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 73e2f8fe196166f9ad1283361867d24c13641
   be6c07faa64d724323e0d7cedc3f38346dcbd212 192.168.0.80:6379@16379 slave fb75f0743af4695a3d241ff7790b2f508e4985f
   c16b9acaeed7dd0721f129596cd43bd499c0e396 192.168.0.169:6379@16379 slave d112bae791b2bbd9602fe32963536b8a0db9eb
   ```

2. 在准备好的ECS上安装迁移工具RedisShake。
   a. 登录ECS。
   b. 在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
      ```
   c. 执行命令解压RedisShake文件。
      ```
      tar -xvf redis-shake-v2.0.3.tar.gz
      ```
      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
      #如果无密码，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：
   a. 连接源Redis和目标Redis。连接Redis的方法请参考**使用redis-cli连接Redis实例**。
   b. 输入info keyspace，查看keys参数和expires参数的值。

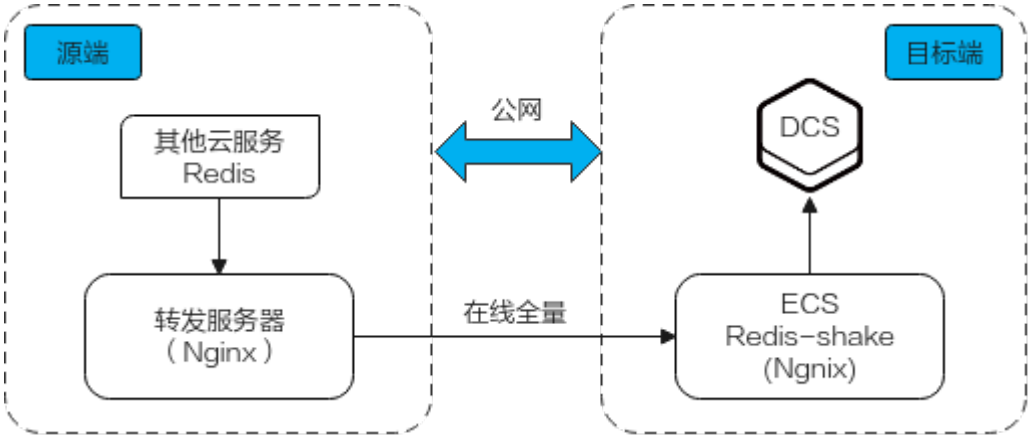
   c. 对比源Redis和目标Redis的keys参数分别减去expires参数的差值。如果差值一致，则表示数据完整，迁移正常。

13.6.5 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DCS中。

图 13-12 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），DCS为集群实例时，该方案不支持（DCS 集群实例目前只支持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.el7.ngx.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。

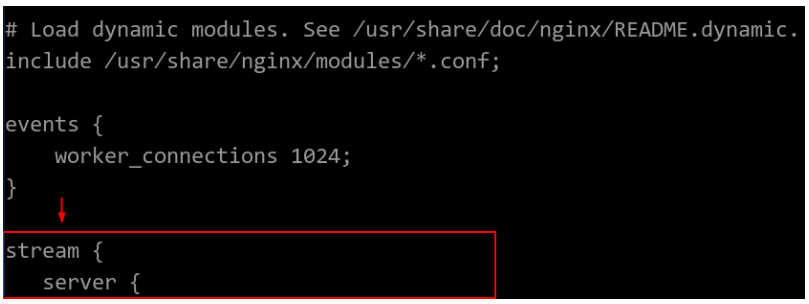
步骤3 在源端转发服务器配置安全组。

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ECS的公网IP地址，并放开来自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 13-13 配置位置要求 1



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 13-14 验证结果 1

步骤5 配置ECS的Nginx转发配置。

1. 登录LinuxECS，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：

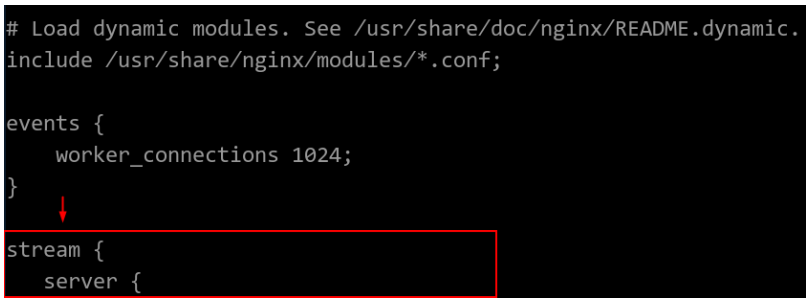
```
stream {
    server {
        listen 6666;
        proxy_pass {source_ecs_address}:6379;
    }
}
```

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 13-15 配置位置要求 2



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。

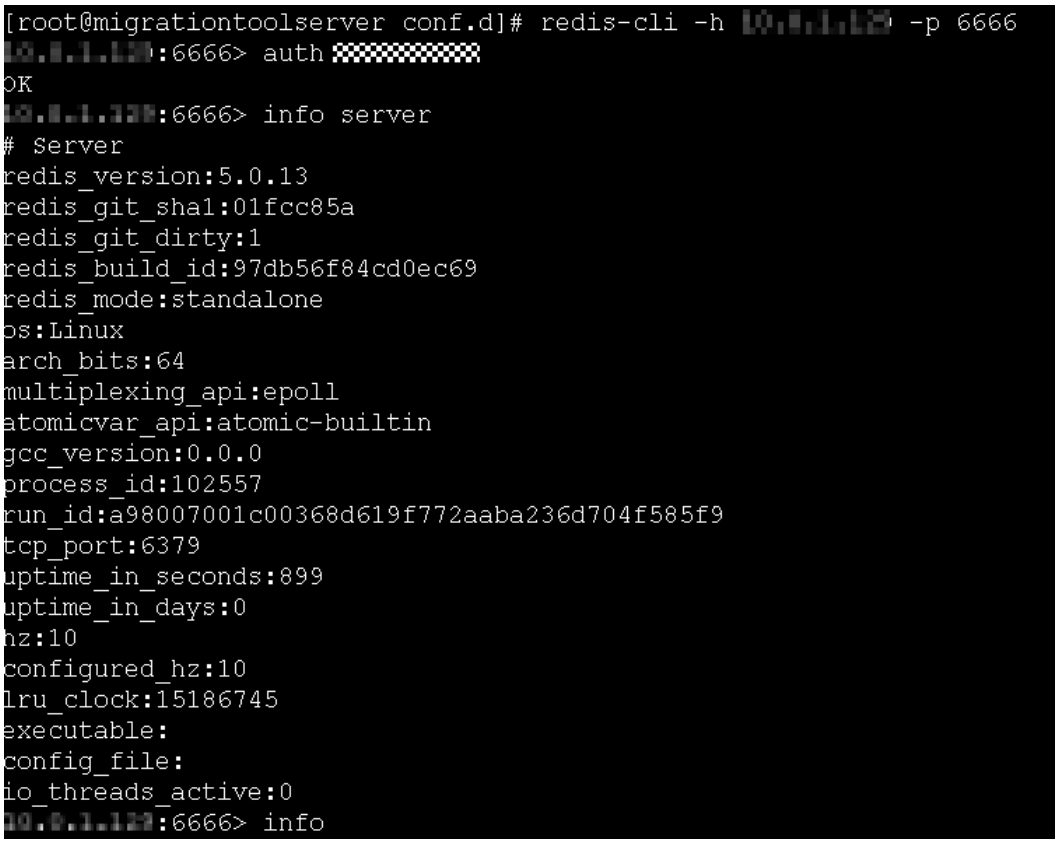
图 13-16 验证结果 2

步骤6 在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ECS公网IP地址，6666为ECS监听端口，{password}为源端Redis密码，如无密码可不填。

图 13-17 连接示例



步骤7 准备迁移工具RedisShake。

1. 登录ECS。
2. 在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

     ECS公网IP地址和映射源端转发服务器的端口(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

  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

  Redis实例的连接地址和端口，用英文冒号隔开。

- target.password_raw

  Redis实例的密码，如未设置密码，无需填写。

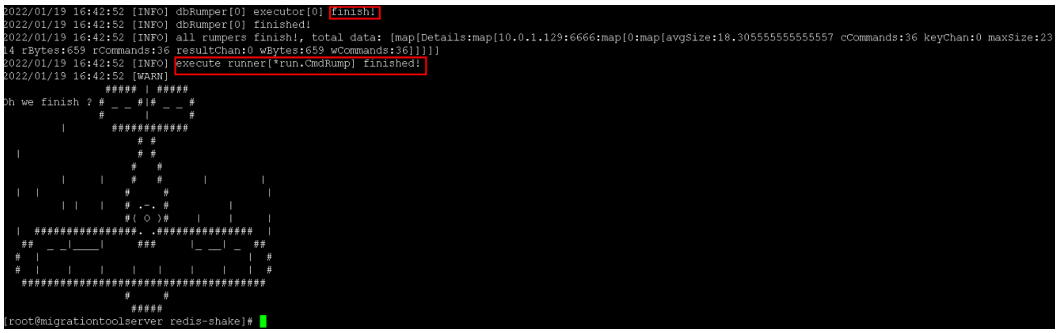
3. 按下Esc键退出编辑模式，输入**:wq!**按回车键保存配置并退出编辑界面。

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 13-18 迁移过程

图 13-19 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**使用redis-cli连接Redis实例**。

2. 输入**info keyspace**，查看keys参数和expires参数的值。
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，表示数据完整，迁移正常。

步骤11 迁移验证完成后，建议及时清理RedisShake配置文件中的配置。

----结束

13.7 DCS 实例迁移下云

场景介绍

您可以通过DCS控制台的在线迁移功能，将DCS实例迁移到自建Redis。另外，您也可以通过导出RDB文件，然后导入本地Redis实例或者自建Redis中。

推荐方案

- DCS控制台在线迁移功能

  在线迁移操作，操作可以参考**使用在线迁移**，选择目标Redis的数据源时，选择“自建Redis”，填写目标Redis地址。
- 使用Redis-cli导出DCS实例RDB文件或者控制台导出实例数据文件，然后使用RedisShake导入。

  关于RedisShake的安装和使用，请参考**使用RedisShake工具迁移自建Redis Cluster集群**及**RedisShake配置说明**。
- Rump

  支持在线迁移，网络条件允许的情况下，可以考虑使用此方式。指导说明请参考**使用Rump在线迁移**。

14 常见问题

14.1 实例类型/版本

14.1.1 DCS 实例的 CPU 规格是怎么样的

使用DCS实例的用户无需关心CPU规格的指标，仅需关心QPS，带宽，内存大小等核心指标。

Redis实例基于开源Redis构造，开源Redis使用单个主线程处理命令，只能利用一个核的CPU，因此，只需认为单个Redis节点仅使用1核CPU即可。提升Redis实例的内存大小，CPU规格不变。

Redis由于社区版单线程处理模型的限制，**如需增加实例CPU处理性能，请使用集群类型的Redis实例，通过增加分片的方式，来增加整个集群的处理性能。**集群实例每个节点默认分配1核CPU进行处理。

14.1.2 如何查询 Redis 实例的原生版本

连接需要查询的实例，执行info命令，即可查看Redis实例的原生版本。

```
INFO
# Server
redis_version:5.0.14
```

14.2 客户端和网络连接

14.2.1 安全组配置和选择

由于Redis 3.0/Memcached和Redis 4.0/5.0/6.0实例的部署模式不一样，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：通过配置安全组访问规则控制，不支持白名单功能。安全组配置操作请参考本章节操作。
- Redis 4.0/5.0/6.0：不支持安全组，只支持通过白名单控制。白名单配置操作，请参考**管理实例白名单**。

本节主要介绍**VPC内访问**DCS Redis 3.0/Memcached缓存实例。

VPC 内访问 Redis 3.0/Memcached 实例

客户端只能部署在与DCS缓存实例处于相同虚拟私有云（VPC）和相同子网的弹性云服务器（ECS）上。

除了ECS、DCS缓存实例必须处于相同VPC和相同子网之外，还需要将安全组分别配置了正确的规则，客户端才能访问DCS缓存实例。

- 如果ECS、DCS缓存实例配置了相同的安全组，安全组创建后，默认包含组内网络访问不受限制的规则。
- 如果ECS、DCS缓存实例配置了不同的安全组，可参考如下配置方式：

  说明

  - 假设ECS、DCS缓存实例分别配置了安全组：sg-ECS、sg-DCS。
  - 假设DCS缓存实例服务端口为6379。
  - 以下规则，远端可使用安全组，也可以使用具体的IP地址。

  a. 配置ECS所在安全组。

  ECS所在安全组需要增加出方向规则，以保证客户端能正常访问DCS缓存实例。如果出方向规则不受限，则不用添加。

  b. 配置DCS缓存实例所在安全组。

  DCS实例所在安全组需要增加入方向规则，以保证能被客户端访问。

须知

缓存实例的入方向规则中，远端地址建议使用与子网同网段的IP地址。

慎用“0.0.0.0/0”，避免绑定相同安全组的弹性云服务器遭受Redis漏洞攻击。

14.2.2 DCS 实例支持公网访问吗?

不支持在DCS实例绑定弹性IP进行公网访问的方式。您必须通过同一虚拟私有云下的弹性云服务器来访问缓存实例，以确保缓存数据的安全。

14.2.3 DCS 实例是否支持跨 VPC 访问?

跨VPC访问，即客户端和实例是否在同一个VPC。

一般情况下，不同VPC间网络不互通，不在同一VPC下的弹性云服务器无法访问DCS缓存实例。

对于单机和主备类型的DCS缓存实例，可以通过创建VPC对等连接，将两个VPC的网络打通，实现跨VPC访问DCS缓存实例。

用户通过VPC对等访问DCS缓存实例时，除了满足VPC对等网跨VPC访问的约束之外，还存在如下约束：

- 当创建实例时使用了172.16.0.0/12~24网段时，客户端不能在192.168.1.0/24、192.168.2.0/24、192.168.3.0/24网段。
- 当创建实例时使用了192.168.0.0/16~24网段时，客户端不能在172.31.1.0/24、172.31.2.0/24、172.31.3.0/24网段。
- 当创建实例时使用了10.0.0.0/8~24网段时，客户端不能在172.31.1.0/24、172.31.2.0/24、172.31.3.0/24网段。

关于创建和使用VPC对等连接，请参考《虚拟私有云 用户指南》的“对等连接”章节。

须知

DCS Redis集群实例不支持跨VPC访问，比如不能通过建立VPC对等连接的方式，从一个VPC去访问另一个VPC的集群实例。

14.2.4 Redis 连接时报错：“(error) NOAUTH Authentication required”。

报错信息是指实例设置了免密访问。连接时不输入密码，即可避免上述错误。

14.2.5 客户 Http 的 Server 端关闭导致 Redis 访问失败

原因分析：客户端使用长连接，或者连接池，用完后关闭与DCS实例的连接，再次使用时，出现报错。

解决方案：使用长连接或连接池，用完后不要关闭连接；如果发现连接中断，请重新建连。

14.2.6 客户端出现概率性超时错误

针对低概率超时错误，是Redis使用的正常现象。Redis使用受到网络传输、客户端设置超时时间等因素影响，可能出现单个请求超时问题。

建议客户业务编码时，具备重试操作，提升业务的可靠性，避免低概率的单次请求失败时业务失败。

当出现了连接超时问题时，可以优先检查Redis是否开启了aof持久化功能，这需要根据业务需求，决定是否开启，防止出现阻塞，连接不上的情况。

如果出现超时错误概率频繁，请联系服务运维。

14.2.7 使用 Jedis 连接池报错如何处理?

在使用Jedis连接池JedisPool模式下，比较常见的报错如下：

```
redis.clients.jedis.exceptions.JedisConnectionException: Could not get a resource from the pool
```

首先确认DCS缓存实例是正常运行中状态，然后按以下步骤进行排查。

步骤1 网络

1. 核对IP地址配置

   检查jedis客户端配置的ip地址是否与DCS缓存实例配置的子网地址一致。

2. 测试网络

   在客户端使用ping和Telnet小工具测试网络。

   - 如果ping不通：

     VPC内访问Redis 3.0/Memcached实例时，要求客户端与DCS缓存实例的VPC相同，安全组相同或者DCS缓存实例的**安全组放开了6379端口访问**。

     VPC内访问Redis实例时，要求客户端与DCS缓存实例的VPC相同。

   - 如果IP地址可以ping通，telnet对应的端口不通，则尝试重启实例，如重启后仍未恢复，请联系技术支持。

步骤2 检查连接数是否超限

查看已建立的网络连接数是否超过JedisPool配置的上限。如果连接数接近配置的上限值，则建议重启服务观察。如果明显没有接近，排除连接数超限可能。

Unix/Linux系统使用**:**

netstat -an | grep 6379 | grep ESTABLISHED | wc -l

Windows系统使用:

netstat -an | find "6379" | find "ESTABLISHED" /C

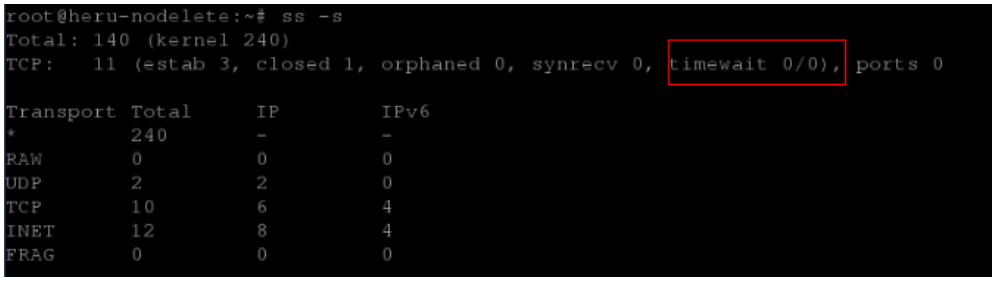
步骤3 检查JedisPool连接池代码

如果连接数接近配置的上限，请分析是业务并发原因，或是没有正确使用JedisPool所致。

对于JedisPool连接池的操作，每次调用**jedisPool.getResource()**方法之后，需要调用**jedisPool.returnResource()**或者**jedis.close()**进行释放，优先使用close()方法。

步骤4 客户端TIME_WAIT是否过多

通过**ss -s**查看**time wait**链接是否过多。



如果**TIME_WAIT**过多，可以调整内核参数（/etc/sysctl.conf）：

```
##当出现SYN等待队列溢出时，启用cookies来处理，可防范少量SYN攻击
net.ipv4.tcp_syncookies = 1
##允许将TIME-WAIT sockets重新用于新的TCP连接
net.ipv4.tcp_tw_reuse = 1
##开启TCP连接中TIME-WAIT sockets的快速回收
net.ipv4.tcp_tw_recycle = 1
##修改系统默认的TIMEOUT时间
net.ipv4.tcp_fin_timeout = 30
```

调整后重启生效：**/sbin/sysctl -p**

步骤5 无法解决问题

如果按照以上原因排查之后还有问题，可以通过抓包并将异常时间点、异常信息以及抓包文件发送给技术支持协助分析。

抓包可使用tcpdump工具，命令如下：

tcpdump -i eth0 tcp and port 6379 -n -nn -s 74 -w dump.pcap

Windows系统下还可以安装Wireshark工具抓包。

说明

网卡名请改成实际的网卡名称。

----结束

14.2.8 客户端访问 Redis 实例出现“ERR unknown command”的原因是什么?

有以下可能原因：

1. 命令拼写不正确

   如下图所示，命令拼写有误，Redis实例返回“ERR unknown command”，删除String的正确命令为**del**。

   ```
   192.168.0.244:6379> delete hellokitty
   (error) ERR unknown command 'delete'
   192.168.0.244:6379> del hellokitty
   (integer) 1
   192.168.0.244:6379>
   ```

2. 在低版本Redis实例运行高版本命令

   例如，在Redis 3.0版本运行Redis 5.0新增的Stream相关命令，Redis实例返回命令出错信息。

   ```
   192.168.0.244:6379> xadd stream01 * field01 teststring
   (error) ERR unknown command 'xadd'
   192.168.0.244:6379> info server
   # Server
   redis_version:3.0.7.9
   redis_git_sha1:10fba618
   ```

3. 部分命令被禁用

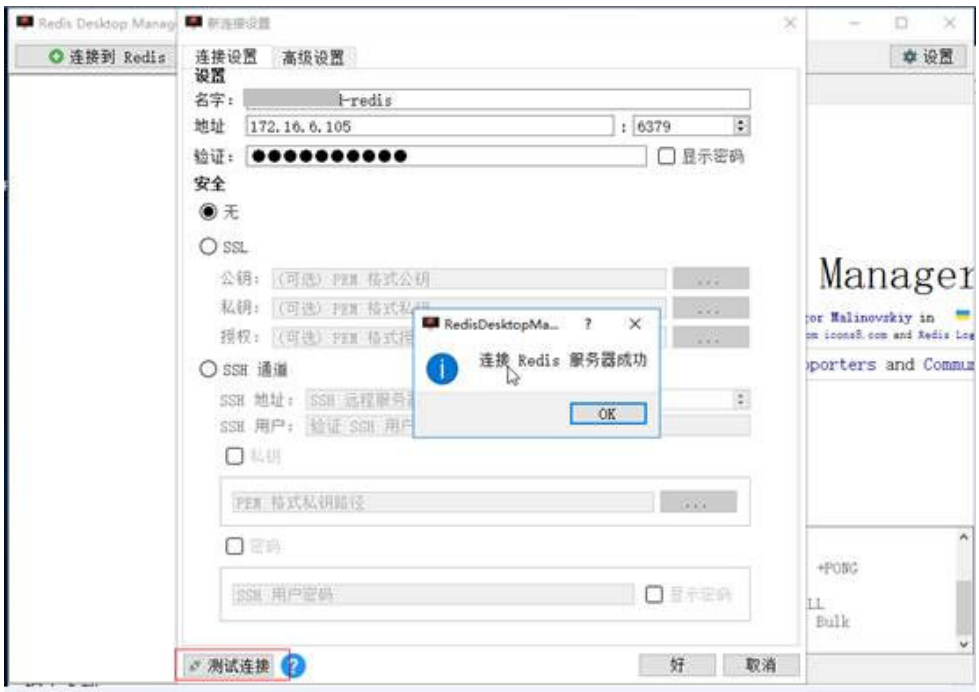
   DCS Redis实例接口与开源Redis在数据访问方面完全兼容。但因易用性和安全性的原因，部分管理操作不能从Redis客户端发起，具体禁用的命令清单，请参考**Redis命令**。

14.2.9 如何使用 Redis-desktop-manager 访问 Redis 实例?

如下介绍通过内网使用Redis-desktop-manager访问Redis实例的操作：

1. 填写DCS实例子网地址，端口6379，以及相应密码。
2. 单击左下角“测试连接”。
   提示成功后，说明连接正常。

图 14-1 通过内网使用 Redis-desktop-manager 访问 Redis 实例



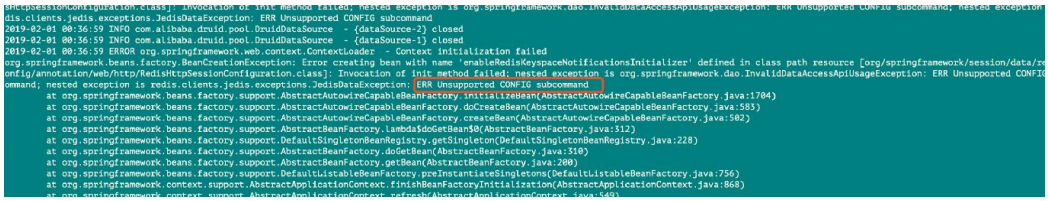
说明

使用Redis-desktop-manager访问DCS集群实例时，执行redis命令是正常的，但是左侧显示异常，这个是因为DCS集群是基于codis架构，info命令的输出和原生的redis不一样。

14.2.10 使用 SpringCloud 时出现 ERR Unsupported CONFIG subcommand 怎么办?

DCS的Redis实例可以配合Spring_Session进行Session共享。DCS的Redis实例对接SpringCloud时，遇到如下错误信息：

图 14-2 Spring Cloud 报错信息



原因为出于安全考虑，DCS暂不支持客户端发起的CONFIG命令，需要按如下步骤进行操作：

1. 通过管理控制台修改Redis实例的配置参数notify-keyspace-event，将值指定为“Egx”。

2. 在Spring框架的XML配置文件中，增加如下：

   <util:constant

   static-
   field="org.springframework.session.data.redis.config.ConfigureRedisAction.NO_OP"/>

3. 修改Spring相关代码，通过启用ConfigureRedisAction.NO_OP这个bean组件，禁止通过客户端调用CONFIG命令，避免报错。

```
@Bean
public static ConfigureRedisAction configureRedisAction() {
    return ConfigureRedisAction.NO_OP;
}
```

更多说明，可参考**Spring官方文档**。

须知

Redis单机和主备实例支持Spring的Session共享，Redis集群版不支持。

14.2.11 连接实例必须使用密码吗？如何获取密码？

- Redis实例支持密码模式和免密模式。Redis本身支持不设置密码，客户端可以直接连接Redis缓存服务并使用，但出于安全考虑，建议尽量选用密码模式，通过密码来鉴权验证，提升安全性。若选用密码模式，您需要在创建实例时自定义密码。
- Memcached实例支持密码模式和免密模式，用户可以根据自身应用特点选用支持文本和二进制协议的任何Memcached客户端。若选用密码模式，您需要在创建实例时自定义密码。
- 如需修改Redis访问方式、修改或重置密码，请参考**密码管理**。

14.2.12 使用 Redis 实例的发布订阅(pubsub)有哪些注意事项？

使用Redis发布订阅功能时有如下事项请注意：

- 客户端需要及时消费和处理消息。

  客户端订阅了channel之后，如果接收消息不及时，可能导致DCS实例消息堆积，当达到消息堆积阈值（默认值为32MB），或者达到某种程度（默认8MB）一段时间（默认为1分钟）后，服务器端会自动断开该客户端连接，避免导致内部内存耗尽。
- 客户端需要支持重连。

  当连接断开之后，客户端需要使用subscribe或者psubscribe重新进行订阅，否则无法继续接收消息。
- 不建议用于消息可靠性要求高的场景中。

  Redis的pubsub不是一种可靠的消息系统。当出现客户端连接退出，或者极端情况下服务端发生主备切换时，未消费的消息会被丢弃。

14.2.13 Redis 实例连接失败的原因排查

初步排查：

- 检查连接地址

连接地址可从管理控制台的实例详情页面获取。

- 检查密码

  密码输入错误时，端口可以连接上，但鉴权认证失败。

- 检查端口

  VPC内访问，Redis实例端口默认为6379。

- 检查带宽是否使用超限

  当实例使用带宽达到实例规格上限，可能会导致部分Redis连接超时现象。

- 如果是Redis 3.0实例，检查安全组的入方向规则

  VPC内访问时，如果Redis客户端和Redis实例绑定了不同的安全组，则需要将Redis实例的入方向安全组放开6379端口。

  具体请参考：**安全组配置和选择**。

- 如果是Redis 4.0及以上版本实例，检查白名单配置

  如果实例开启了白名单，在使用客户端连接时，需要确保客户端IP在白名单内，如果不在白名单，会出现连接失败。

  具体配置操作，可以参考**管理实例白名单**。

  客户端IP如果有变化，需要将变化后的IP加入白名单。

- 检查实例配置参数notify-keyspace-events

  建议将notify-keyspace-events参数配置为Egx。

进阶排查

- Jedis连接池报错
- 出现**Read timed out**或**Could not get a resource from the pool**

  排查是否使用了keys命令，keys命令会消耗大量资源，造成Redis阻塞。建议使用scan命令替代，且避免频繁执行。

14.2.14 使用短连接访问 Redis 出现“Cannot assign requested address”错误

问题描述

应用程序通过短连接访问Redis实例时，报错：Cannot assign requested address。

问题分析

出现这种错误的应用程序使用的架构基本都是php-fpm加上phpredis，这种架构在并发量较大的情况下，处于TIME-WAIT状态下的TCP连接数较多，客户端无法分配出新的端口，则会出现“Cannot assign requested address”问题。

处理方案

- 方案一：使用pconnect替换connect。

  此方案的思路是用长连接替代短连接，减少TCP连接，同时可以避免每次请求都会重新建立连接的问题，减少延时。

  之前连接Redis的代码如下：

```
$redis->connect('${Hostname}',${Port});
$redis->auth('${Inst_Password}');
```

现使用pconnect替换connect，即使用persistent connection的方式连接。

```
$redis->pconnect('${Hostname}', ${Port}, 0, NULL, 0, 0, ['auth' => ['${Inst_Password}']]);
```

说明

- 示例中的连接参数请根据业务实现情况修改，${Hostname}、${Port}和${Inst_Password}为Redis实例的连接地址、端口号和密码。
- PhpRedis应为5.3.0及以上版本，且建议使用这种pconnect初始化方式，避免断连时出现no auth问题。

- 方案二：修改客户端所在ECS实例的tcp_max_tw_buckets内核参数。

  此方案的思路是直接复用处于TIME-WAIT状态的端口，但是如果ECS和后端服务之间有重传，连接可能会失败，所以建议使用pconnect的方案。

  a. 连接客户端所在ECS实例。

  b. 执行以下命令，查看ip_local_port_range和tcp_max_tw_buckets参数。

  sysctl net.ipv4.tcp_max_tw_buckets net.ipv4.ip_local_port_range

  系统显示类似如下：

  ```
  net.ipv4.tcp_max_tw_buckets = 262144
  net.ipv4.ip_local_port_range = 32768  61000
  ```

  c. 执行以下命令，修改tcp_max_tw_buckets参数，确保tcp_max_tw_buckets的值比ip_local_port_range范围的值小。

  sysctl -w net.ipv4.tcp_max_tw_buckets=10000

一般情况推荐使用方案一，对于一些特定场景（业务代码牵涉过多组件不易变更等场景），需要更快地满足高并发，可以使用方案二。

14.2.15 连接池选择及 Jedis 连接池参数配置建议

Jedis 连接池优势

Lettuce客户端及Jedis客户端比较如下：

- Lettuce：
  - Lettuce客户端没有连接保活探测，错误连接存在连接池中会造成请求超时报错。
  - Lettuce客户端未实现testOnBorrow等连接池检测方法，无法在使用连接之前进行连接校验。
- Jedis：
  - Jedis客户端实现了testOnBorrow、testWhileIdle、testOnReturn等连接池校验配置。

    开启testOnBorrow在每次借用连接前都会进行连接校验，可靠性最高，但是会影响性能（每次Redis请求前会进行探测）。
  - testWhileIdle可以在连接空闲时进行连接检测，合理配置阈值可以及时剔除连接池中的异常连接，防止使用异常连接造成业务报错。
  - 在空闲连接检测之前，连接出现问题，可能会造成使用该连接的业务报错，此处可以通过参数控制检测间隔（timeBetweenEvictionRunsMillis）。

因此，Jedis客户端在面对连接异常，网络抖动等场景下的异常处理和检测能力明显强于Lettuce，可靠性更强。

Jedis 连接池参数配置建议

表 14-1 Jedis 连接池参数配置建议

参数	配置介绍	配置建议
maxTotal	最大连接，单位：个	根据Web容器的Http线程数来进行配置，估算单个Http请求中可能会并行进行的Redis调用次数，例如：Tomcat中的Connector内的maxConnections配置为150，每个Http请求可能会并行执行2个Redis请求，在此之上进行部分预留，则建议配置至少为：150 x 2 + 100= 400 **限制条件：**单个Redis实例的最大连接数。maxTotal和客户端节点数（CCE容器或业务VM数量）数值的乘积要小于单个Redis实例的最大连接数。 例如：Redis主备实例配置maxClients为10000，单个客户端maxTotal配置为500，则最大客户端节点数量为20个。
maxIdle	最大空闲连接，单位：个	配置与maxTotal一致。
minIdle	最小空闲连接，单位：个	一般来说建议配置为maxTotal的X分之一，例如此处常规配置建议为：100。 对于性能敏感的场景，为了防止经常连接数量抖动造成影响，可以配置与maxIdle一致，例如：400。
maxWaitMillis	最大获取连接等待时间，单位：毫秒	获取连接时最大的连接池等待时间，根据单次业务最长容忍的失败时间减去执行命令的超时时间得到建议值。例如：Http最长容忍的失败时间为15s，Redis请求的timeout设置为10s，则此处可以配置为5s。
timeout	命令执行超时时间，单位：毫秒	单次执行Redis命令最大可容忍的超时时间，根据业务程序的逻辑进行选择，出于对网络容错等考虑建议配置为不小于210ms。特殊的探测逻辑或者环境异常检测等，可以适当调整达到秒级。
minEvictableIdleTimeMillis	空闲连接逐出时间，大于该值的空闲连接一直未被使用则会被释放，单位：毫秒	如果希望系统不会经常对连接进行断链重建，此处可以配置一个较大值（xx分钟），或者此处配置为-1并且搭配空闲连接检测进行定期检测。

参数	配置介绍	配置建议
timeBetweenEvictionRunsMillis	空闲连接探测时间间隔，单位：毫秒	根据系统的空闲连接数量进行估算，例如系统的空闲连接探测时间配置为30s，则代表每隔30s会对连接进行探测，如果30s内发生异常的连接，经过探测后会进行连接排除。根据连接数的多少进行配置，如果连接数太大，配置时间太短，会造成请求资源浪费。对于几百级别的连接，常规来说建议配置为30s，可以根据系统需要进行动态调整。
testOnBorrow	向资源池借用连接时是否做连接有效性检测（ping），检测到的无效连接将会被移除。	对于业务连接极端敏感的，并且性能可以接受的情况下，可以配置为True，一般来说建议配置为False，不启用连接空闲检测。
testWhileIdle	是否在空闲资源监测时通过ping命令监测连接有效性，无效连接将被销毁。	True
testOnReturn	向资源池归还连接时是否做连接有效性检测（ping），检测到无效连接将会被移除。	False
maxAttempts	在JedisCluster模式下，您可以配置maxAttempts参数来定义失败时的重试次数。	建议配置3-5之间，默认配置为5。 根据业务接口最大超时时间和单次请求的timeout综合配置，最大配置不建议超过10，否则会造成单次请求处理时间过长，接口请求阻塞。

14.3 Redis 使用

14.3.1 如何理解分片数与副本数?

什么是分片

分片也叫**条带**，指Redis集群的一个管理组，对应一个redis-server进程。一个Redis集群由若干条带组成，每个条带负责若干个slot（槽），数据分布式存储在slot中。Redis集群通过条带化分区，实现超大容量存储以及并发连接数提升。

每个集群实例由多个分片组成，每个分片默认为一个双副本的主备实例。分片数等于实例中主节点的个数。

什么是副本

副本指缓存实例的**节点**，包含主节点和备节点。单副本表示实例没有备节点，双副本表示实例有备节点（一个主节点，一个备节点）。例如主备实例的副本数设置为3时，表示该实例有1个主节点，2个备节点。

不同实例类型的副本和分片数

- **单机实例：**单机实例只有1个节点，1个Redis进程，当Redis进程故障后，DCS为实例重新拉起一个新的Redis进程。
- **主备实例：**分片数为1，包含一个主节点，一个或多个备节点。当主节点出现故障时，会进行主备倒换，恢复业务。
- **集群实例：**集群实例由多个分片组成，每个分片默认是一个双副本的主备实例。例如一个3分片，2副本的集群实例，则每个分片都有2个节点（1个主节点，1个备节点）。

实例类型	分片数	副本数	负载均衡	占用IP数
单机	单分片	单副本，不支持多副本	-	1个
主备	单分片	默认双副本，支持配置为2-10副本	不支持	占用IP个数=副本数
Proxy集群	多分片	双副本，不支持其他副本数	支持	1个
Cluster集群	多分片	默认双副本，支持配置为1-5副本	不支持	占用IP个数=副本数*分片数

如何查看副本 IP

通过实例的节点管理功能，可以查看实例的副本IP。查看方式请参见**管理分片与副本**。

14.3.2 Redis 实例 CPU 使用率达到 100%的原因

- 可能原因1：
  客户的业务负载过重，QPS过高，导致CPU被用满。
- 可能原因2：
  使用了keys等消耗资源的命令。这会导致CPU使用率超高，容易触发主备倒换。
- 可能原因3：
  发生Redis的持久化重写操作，CPU使用率增加。

详情请参考**Redis实例CPU使用率高问题排查和解决**。

14.3.3 Redis 实例能否修改 VPC 和子网?

实例的VPC和子网，创建后不允许修改。如果要修改，请重新创建实例，在创建时选择指定的VPC和子网。如果实例已有数据需要迁移，可在创建实例之后，使用**数据迁移**进行迁移。

14.3.4 Redis 4.0 及以上版本实例为什么没有安全组信息?

目前Redis 4.0及以上版本实例是基于VPCEndpoint，暂不支持安全组，支持白名单配置，参考**管理实例白名单**。

如果需要指定的IP地址才能访问Redis实例，您需要将指定的IP地址加入到实例白名单中。

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。

14.3.5 Redis 实例支持的单个 Key 和 Value 数据大小是否有限制?

- Key的大小上限为512M。

  建议key的大小不超过1kb，这样既节约存储空间，也利于Redis进行检索。
- String类型的value值上限为512M。
- 集合、链表、哈希等key类型，单个元素的value上限为512M。

  事实上，集合、链表、哈希都可以看成由String类型的key按照一定的映射关系组合而成。

同时，请注意避免对大Value进行长时间高并发写入，这样会影响网络传输效率，也会增加redis-server的内部处理耗时，从而导致请求时延较大。

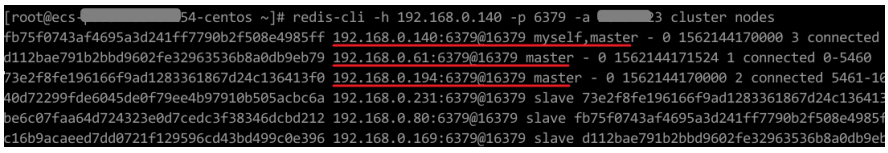
14.3.6 Redis 集群可以读取每个节点的 IP 地址吗?

Redis 3.0版本的集群实例（Proxy版本）的使用方式与单机、主备实例相同，无需知晓后端地址。

Redis 4.0及以上版本的集群实例（Cluster版本）可以使用**cluster nodes**命令获取。

redis-cli -h *{redis_address}* -p *{redis_port}* -a *{redis_password}* cluster nodes

在命令返回的结果中，获取所有master节点的IP端口，如下图所示。



14.3.7 Redis 实例是否支持读写分离？Cluster 集群实例如何配置读写分离

配置说明

- **Redis Cluster集群实例**，使用**cluster nodes**查询所有主备节点，客户端连接备节点，并在节点上做配置，开启备节点只读访问，从而实现读写分离。

  查询集群节点命令如下：

  ```
  redis-cli -h {redis_address} -p {redis_port} -a {redis_password} cluster nodes
  ```

  从节点配置只读模式，请参考**READONLY命令**。

14.3.8 Redis 实例是否支持多 DB 方式?

Redis单机和主备缓存实例支持多DB，默认256个，DB编号为0～255。默认使用的是DB0。

Redis集群实例不支持多DB，只有一个DB，即DB0。

14.3.9 Redis 集群实例是否支持原生集群?

当前DCS Redis 3.0版本仅支持Proxy集群，Redis 4.0及以上版本支持原生集群。

14.3.10 哨兵原理

Sentinel 概览

Redis Sentinel为Redis实现高可用。实际使用中，您可以使用Sentinel帮助Redis在无需人工干预的情况下抵御某些类型的故障，Redis Sentinel还能够完成其他辅助任务，如监控、通知和客户端配置。详细介绍可参考**Redis官网**。

Sentinel 原理

Redis Sentinel是一个分布式系统，Sentinel的设计基础在于多个Sentinel进程协同工作，这样做的好处有：

1. 只有当多个哨兵一致同意某主节点不可用，才执行故障检测，这能够降低误报的可能性。
2. 即使有些Sentinel进程故障，Sentinel系统也能正常工作，从而抵御故障。

从更大范围来看，Sentinel加上Redis主从节点以及连接到Sentinel和Redis的客户端，整体也构成一个更大的分布式系统。

Sentinel 功能

- 监控：Sentinel不间断地检查主从节点是否都在正常工作。
- 通知：如果Redis中某节点故障，Sentinel可以通过API通知系统管理员或其他计算机程序。
- 自动故障切换：如果主节点异常，Sentinel启动故障切换，将一个从节点升主，其他从节点从新的主节点进行复制，并通知使用该Redis的应用程序使用新地址进行连接。
- 客户端配置来源：Sentinel充当客户端服务发现的权威来源。客户端连接到Sentinel，请求当前负责特定业务的Redis主节点地址。如果发生故障切换，Sentinels将下发新地址。

14.3.11 Redis 实例是否支持配置哨兵模式?

Redis 4.0及以上版本的主备实例，以及集群实例的每个分片（每个分片也是一个主备实例），都使用哨兵模式（Sentinel）进行管理，Sentinel会一直监控主备节点是否正常运行，当主节点出现故障时，进行主备倒换。

Redis 3.0不支持哨兵模式。

14.3.12 Redis 默认的数据逐出策略是什么?

逐出指将数据从缓存中删除，以腾出更多的存储空间容纳新的缓存数据，详情请参见官网**逐出策略**。Redis实例支持在配置运行参数中**查看或修改Redis实例使用的逐出策略**。

Redis 实例支持的逐出策略

在达到内存上限（maxmemory）时Redis支持选择以下8种数据逐出策略：

- noeviction：在这种策略下，如果缓存达到了配置的上限，实例将不再处理客户端任何增加缓存数据的请求，比如写命令，实例直接返回错误给客户端。缓存达到上限后，实例只处理删除和少数几个例外请求。
- allkeys-lru：根据LRU（Least recently used，最近最少使用）算法尝试回收最少使用的键，使得新添加的数据有空间存放。
- volatile-lru：根据LRU（Least recently used，最近最少使用）算法尝试回收最少使用的键，但仅限于在过期集合的键，使得新添加的数据有空间存放。
- allkeys-random：回收随机的键使得新添加的数据有空间存放。
- volatile-random：回收随机的键使得新添加的数据有空间存放，但仅限于在过期集合的键。
- volatile-ttl：回收在过期集合的键，并且优先回收存活时间（TTL）较短的键，使得新添加的数据有空间存放。
- allkeys-lfu：从所有键中驱逐最不常用的键。
- volatile-lfu：从具有“expire”字段集的所有键中驱逐最不常用的键。

说明

当没有键满足回收前提条件时，数据逐出策略volatile-lru、volatile-random、volatile-ttl与noeviction策略相同，具体见上文noeviction介绍。

查看或修改 Redis 实例使用的逐出策略

Redis实例支持通过修改maxmemory-policy参数配置，查看及修改实例的数据逐出的策略。

14.3.13 使用 redis-exporter 出错怎么办?

通过在命令行启动redis-exporter，根据界面输出，查看是否存在错误，根据错误描述，进行问题排查。

```
[root@ecs-swk /]./redis_exporter -redis.addr 192.168.0.23:6379
INFO[0000] Redis Metrics Exporter V0.15.0    build date:2018-01-19-04:08:01 sha1:
a0d9ec4704b4d35cd08544d395038f417716a03a
  Go:go1.9.2
INFO[0000] Providing metrics at :9121/metrics
INFO[0000] Connecting to redis hosts: []string{192.168.0.23:6379}
INFO[0000] Using alias:[]string{""}
```

14.3.14 Redis 的安全加固方面有哪些建议?

在众多开源缓存技术中，Redis无疑是目前功能最为强大，应用最多的缓存技术之一，但是原生Redis版本在安全方面非常薄弱，很多地方不满足安全要求，如果暴露在公网上，极易受到恶意攻击，导致数据泄露和丢失。

针对DCS的Redis实例，您在使用过程中，可参考如下建议：

- 网络连接配置
  a. 敏感数据加密后存储在Redis实例。
     对于敏感数据，尽量加密后存储。
  b. 对安全组设置有限的、必须的允许访问规则。
     安全组与VPC均是用于网络安全访问控制的配置，以端口最少放开原则配置安全组规则，降低网络入侵风险。

c. 客户端应用所在ECS设置防火墙。

客户端应用所在的服务器建议配置防火墙过滤规则。

d. 设置实例访问密码。

e. 配置实例白名单。

- Redis-cli使用

a. 隐藏密码

安全问题：通过在redis-cli指定-a参数，密码会被ps出来，属于敏感信息。

解决方案：修改Redis源码，在main方法进入后，立即隐藏掉密码，避免被ps出来。

b. 禁用脚本通过sudo方式执行

安全问题： redis-cli访问参数带密码敏感信息，会被ps出来，也容易被系统记录操作日志。

解决方案：改为通过API方式（Python可以使用redis-py）来安全访问，禁止通过sudo方式切换到dbuser账号使用redis-cli。

14.3.15 实例是否支持自定义或修改端口?

如果是Redis 3.0实例和Memcached实例，访问端口固定为以下端口，不支持指定端口，也不支持修改；如果是Redis 4.0/5.0/6.0实例，支持创建实例时自定义端口，不支持修改端口。

- Redis 3.0

  VPC内使用实例6379端口。

- Memcached

  仅支持VPC内访问，端口为11211端口。

- Redis 4.0及以上版本

  创建实例时，可选择自定义端口和使用默认端口，自定义端口范围为1~65535，如果没有自定义，则使用默认端口6379。

如果实例与客户端的安全组不同，还需要修改安全组配置，放开端口访问。具体修改方法，请参考：**安全组配置和选择**。

14.3.16 实例是否支持修改访问地址?

DCS实例创建后，实例连接地址不支持修改。

如果需要更换实例IP地址，需要重新创建实例，在创建实例时，选择“手动分配IP地址”，指定实例的IP地址，然后使用在线迁移方式，将旧的实例数据迁移到新的实例。

有关DCS实例的客户端访问，请参考**连接缓存实例**。

14.3.17 实例无法删除是什么原因?

可能原因如下：

- 实例状态不是“运行中”。

  只有当实例处于“运行中”状态，才能执行删除操作。

- 确认实例是否为创建失败的实例。

单击缓存实例列表上方“创建失败的实例”后的图标或者数量，如果是创建失败的实例，会显示在弹出的“创建失败的实例”窗口中，请从该界面中进行删除。

14.3.18 DCS 实例是否支持跨可用区部署?

Redis主备和集群实例、Memcached主备实例支持跨可用区（AZ）部署。

- 当主备或者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。
- 实例跨可用区部署时，主备节点之间同步效率与同AZ部署相比基本无差异。

14.3.19 集群实例启动时间过长是什么原因?

可能原因：在集群实例启动过程中，实例节点内部会进行状态、数据的同步。如果在完成同步之前就持续写入较多的数据，会导致实例内部同步耗费较长时间，实例状态一直处于“启动中”。直到同步完成，集群实例状态才会切换到“运行中”。

解决方案：建议等集群实例启动完成后，再恢复业务数据写入。

14.3.20 DCS Redis 有没有后台管理软件?

没有。Redis的配置信息与使用信息可通过Redis-cli查询；对Redis实例的监控数据可通过云监控服务查看，监控数据的设置与查看方法，请参考**监控**章节。

14.3.21 DCS 缓存实例的数据被删除之后，能否找回?

DCS缓存实例自行删除或者通过Redis客户端发送命令手动删除的数据，不能找回。如果实例执行了备份操作，则通过备份文件可以对数据进行恢复，但是恢复会覆盖备份时间到恢复这段时间的写入数据。

Redis实例（单机实例除外）通过控制台的“备份与恢复”功能将已备份的数据恢复到DCS缓存实例中，参考**实例恢复**。

另外，如果DCS缓存实例被删除，实例中原有的数据将被删除，实例的备份数据也会删除，请谨慎操作。在删除实例之前，您可以将实例的备份文件下载，本地永久保存，如需恢复数据，可将本地备份文件迁移到新的实例中。下载备份数据的方式，请参考**下载实例备份文件**。

14.3.22 Redis 实例是否支持 SSL 加密传输?

Redis 6.0/7.0实例SSL默认关闭，如需开启SSL加密传输，请参考**SSL设置**。

Redis 3.0/4.0/5.0不支持SSL加密传输，仅支持明文传输。

14.3.23 为什么实例实际可用内存比申请规格小而且已使用内存不为0?

由于系统开销会占用部分资源，主备实例的持久化也需要一部分资源，所以Redis 3.0和Memcached实例创建后，缓存实例实际可用内存小于申请规格。除了用户存储数据外，Redis-server内部的buffer以及内部数据结构会占用一部分内存。所以缓存实例创建后，实例已使用内存量不为0。其他版本的实例不涉及该问题。

14.3.24 如何查看 Redis 内存占用量

当前DCS Redis提供了以下与内存相关的指标。

表 14-2 Redis 实例支持的监控指标

指标ID	指标名称	含义	取值范围	测量对象&维度	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：%。	0-100 %	测量对象： Redis实例 测量维度： dcs_instance_id	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。	>=0byte	测量对象： Redis实例 测量维度： dcs_instance_id	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。	>= 0byte	测量对象： Redis实例 仅Redis 4.0及以上的版本支持 测量维度： dcs_instance_id	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中数据内存所占当前已用总内存的百分比。 单位：%。	0-100 %	测量对象： Redis实例 仅Redis 4.0及以上的版本支持 测量维度： dcs_instance_id	1分钟

指标ID	指标名称	含义	取值范围	测量对象&维度	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。	>=0byte	测量对象：Redis实例 测量维度：dcs_instance_id	1分钟
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	测量对象：Redis实例 测量维度：dcs_instance_id	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。	>=0byte	测量对象：Redis实例 测量维度：dcs_instance_id	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。	>=0byte	测量对象：Redis实例 测量维度：dcs_instance_id	1分钟

14.3.25 Cluster 集群实例容量和性能未达到瓶颈，但某个分片容量或性能已过载是什么原因?

这是由于Cluster集群采用的是分片设计理念，每个**具体的Key只能分布到某一个具体的分片节点上**，计算Key的分布过程有以下两个步骤：

1. 针对Key值进行CRC16算法计算后对16384取模，得到对应的槽位（Slot）值。
2. 根据S槽位（Slot）和分片的映射关系，找到Key具体应该属于的分片，并且进行存取。

所以，Key并没有均匀分布在实例的各个分片上，是根据计算结果进行存取的。在大Key和热Key存在时，就会出现某个分片容量或性能已过载，但其他分片内存负载还是很低，并没有达到容量和性能的瓶颈。

14.3.26 DCS 是否支持外部扩展模块、插件或者 Module?

DCS Redis不支持加载外部扩展模块、插件和Module。DCS后续也没有Module的规划。

14.3.27 访问 Redis 返回“Error in execution”

访问Redis返回Error in execution; nested exception is io.lettuce.core.RedisCommandExecutionException: OOM command not allowed when used memory > 'maxmemory'。

OOM代表的就是超过了最大内存，报错中OOM command not allowed when used memory > 'maxmemory'的‘maxmemory’这个参数是Redis服务端对最大内存的配置，可以看到这个是内存使用满了。

若Redis实例内存使用率并未达到100%，有可能当前写入数据的那个节点的mem达到最大值。通过redis-cli -h <redis_ip> -p 6379 -a <redis_password> -c --bigkeys 连接到集群的各个节点进行分析。如果连接的从节点，需要在执行bigkeys命令之前，先发送READONLY命令。

14.3.28 Redis key 丢失是什么原因

redis实例是不会主动丢失数据的，key丢失一般有这几种情况：1、key过期；2、key被逐出；3、key被删除。

按照顺序进行排查：

1. 查看key是否过期。
2. 查看监控，分析是否会触发键逐出机制。
3. 去服务端分析info查看是否有删除key的操作。

14.3.29 访问 Redis 报 OOM 错误提示

问题描述

访问Redis返回Error in execution; nested exception is io.lettuce.core.RedisCommandExecutionException: OOM command not allowed when used memory > 'maxmemory'。

问题排查

OOM代表的就是超过了最大内存，报错中OOM command not allowed when used memory > 'maxmemory'的‘maxmemory’这个参数是Redis服务端对最大内存的配置，可以看到这是内存使用满了。

若Redis实例内存使用率并未达到100%，有可能当前写入数据的那个节点的mem达到最大值。通过**redis-cli -h <redis_ip> -p 6379 -a <redis_password> -c --bigkeys**连接到集群的各个节点进行分析。如果连接的从节点，需要在执行**bigkeys**命令之前，先发送**READONLY**命令。

14.3.30 不同编程语言如何使用 Cluster 集群客户端

当前DCS Cluster集群对比Proxy集群的优势和特性：

表 14-3 Cluster 集群与 Proxy 集群差异

对比项	Cluster集群	Proxy集群
原生兼容性	高	中
客户端兼容性	中（需要客户端开启集群模式）	高
性价比	高	中
时延	低时延	中等时延
读写分离	原生支持（客户端SDK配置）	Proxy实现
性能	高	中

Cluster集群由于没有代理层，在时延和性能方面具备一定的优势；但是对于客户端使用方面，由于Cluster集群使用开源的Redis Cluster协议，在客户端的兼容性方面略差于Proxy集群。

推荐的Cluster集群客户端：

表 14-4 Cluster 集群客户端

客户端语言	客户端类型	Cluster集群参考文档
Java	Jedis	**https://github.com/xetorthio/jedis#jedis-cluster**
Java	Lettuce	**https://github.com/lettuce-io/lettuce-core/wiki/Redis-Cluster**
PHP	php redis	**https://github.com/phpredis/phpredis#readme**
Go	Go Redis	Cluster集群：**https://pkg.go.dev/github.com/go-redis/redis/v8#NewClusterClient** Proxy集群或单机主备：**https://pkg.go.dev/github.com/go-redis/redis/v8#NewClient**

客户端语言	客户端类型	Cluster集群参考文档
Python	redis-py-cluster	https://github.com/Grokzen/redis-py-cluster#usage-example
C	hiredis-vip	https://github.com/vipshop/hiredis-vip?_ga=2.64990636.268662337.1603553558-977760105.1588733325
C++	redis-plus-plus	https://github.com/sewenew/redis-plus-plus?_ga=2.64990636.268662337.1603553558-977760105.1588733325#redis-cluster
Node.js	node-redis io-redis	https://github.com/NodeRedis/node-redis https://github.com/luin/ioredis

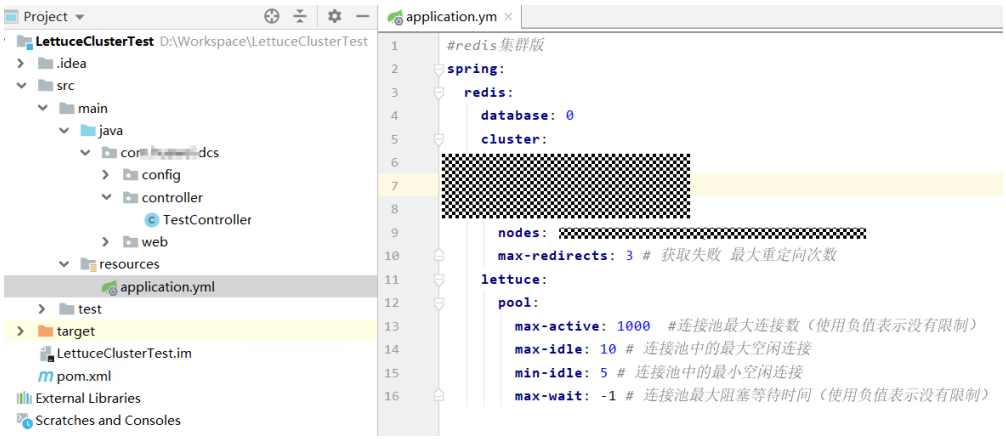
官方推荐的开源客户端列表：https://redis.io/clients。

14.3.31 使用 Cluster 的 Redis 集群时建议配置合理的超时时间

客户端配置问题导致无法连接。

当集群实例备节点故障情况下，客户端使用SpringBoot + Lettuce的方式连接Redis，使用的Lettuce客户端在连接集群时，需要与所有节点先建立连接（包括故障节点）。

- 在未配置timeout超时的情况下，模拟备节点故障时，可能出现分钟级的超时阻塞（Lettuce客户端的老版本默认超时为120s，新版本默认为60s），配置如下图：



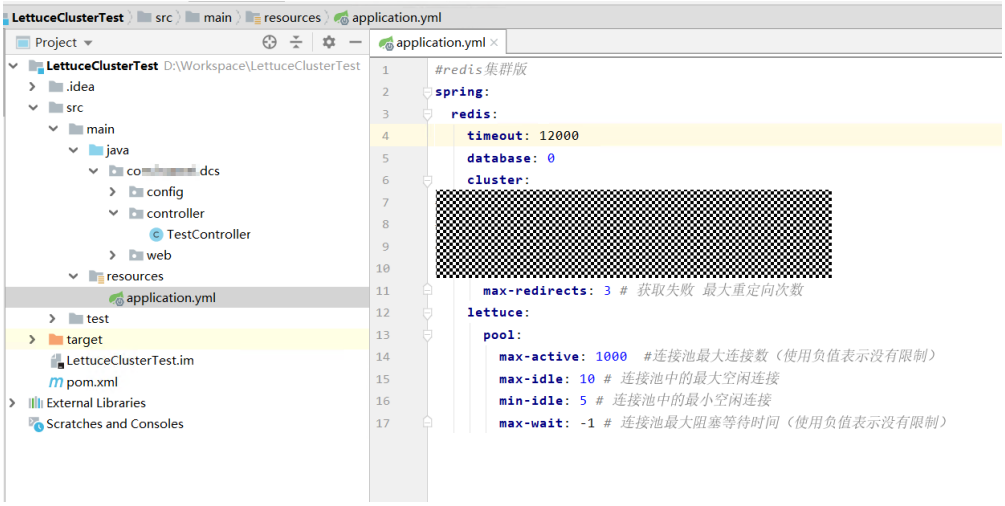
可能会导致端到端业务访问时间过长（最长达到默认超时时间），如下图所示：

```
[root@ecs-a776 ~]# time curl -X get http://1[redacted].0.0.1:8080/test/evalsha



false
real    2m0.632s
user    0m0.003s
sys     0m0.004s
```

- 在客户端侧添加timeout参数后，备节点超时时间大幅度缩短，并且可以根据客户自己的业务诉求进行调整，配置如下：



配置后查看端到端业务访问时间如下图所示：

```
[root@ecs-a776 ~]# time curl -X get http://1[redacted].0.0.1:8080/test/evalsha
false
real    0m12.627s
user    0m0.000s
sys     0m0.004s
```

因此在未配置timeout参数情况下，客户端在建立连接时，故障节点由于未配置timeout超时，在建立连接时会出现连接阻塞的情况。

建议：用户需根据业务能容忍的超时时间进行设置，例如在一次HTTP端到端请求中，需要请求两次Redis，而HTTP请求的最大超时时间为10s，则建议将超时时间配置为5s，防止由于超时时间过长或者未配置超时时间造成故障场景下的业务受损。

14.3.32 实例是否支持变更可用区

不支持直接变更实例的可用区。

如需改变可用区，可通过“数据迁移+交换IP”方式的方式，在新的可用区创建实例后，进行数据迁移，实现可用区的变更。具体操作请参见**交换DCS实例IP**。

如果是主备实例，可以通过新增和删除副本的方式（新增副本时支持选择副本可用区），变更备节点的可用区。新增和删除副本的操作请参见**变更规格**。

14.3.33 hashtag 的原理、规则及用法示例

hashtag 原理

单实例上的mset、lua脚本等处理多key时，是一个原子性(atomic)操作，所有给定key都会在同一时间内被执行。集群每次通过对key进行hash计算到不同的分片，所以集群

上同时执行多个key，不再是原子性操作，会存在某些给定 key 被更新而另外一些给定key没有改变的情况，其原因是需要设置的多个key可能分配到不同的机器上。因此集群引入了hashtag来对多key同时操作，在设置了hashtag的情况下，集群会根据hashtag决定key分配到的slot， 当两个key拥有相同的hashtag时, 它们会被分配到同一个slot。

hashtag 使用规则

第一次出现“{”和接下来第一次出现的“}”之间有内容。

例如：

- 这两个键{user1000}.following和{user1000}.followers由于只有一对{}，将user1000来计算hash。
- 对于键foo{}{bar}，整个键foo{}{bar}将像往常一样计算hash，因为第一次出现的“{”后面跟“}”中间没有字符。
- 对于键foo{{bar}}zap，子字符串{bar将被计算hash，因为它是第一次出现“{”和第一次出现“}”之间的子字符串。
- 对于键foo{bar}{zap}的子字符串bar将被计算hash，因为只使用第一个“{”和“}”。

hashtag 用法示例

当如下操作时：

```
EVAL "redis.call('set',KEYS[1],ARGV[1]) redis.call('set',KEYS[2],ARGV[2])" 2 key1 key2 value1 value2
```

出现以下报错：

ERR 'key1' and 'key2' not in the same slot

可通过hashtag进行解决：

```
EVAL "redis.call('set',KEYS[1],ARGV[1]) redis.call('set',KEYS[2],ARGV[2])" 2 {user}key1 {user}key2 value1 value2
```

14.3.34 重启实例后缓存数据会保留吗?

单机缓存实例重启后，原有的数据将被删除。

主备和集群实例（单副本集群除外）默认支持AOF持久化，实例重启后原有的数据会保留。如果关闭了AOF持久化（appendonly参数修改为no即AOF持久化功能关闭），实例重启后原有的数据将被删除。

14.4 Redis 命令

14.4.1 如何清空 Redis 数据?

注意数据清空功能为高危操作，请谨慎执行。

- Redis 3.0实例

  Redis 3.0实例不支持在DCS控制台上执行“数据清空”功能。需要使用Redis-cli客户端连接实例，执行flushdb或者flushall命令进行清空。

  flushall：清空整个实例的数据。

flushdb：清空当前DB中的数据。

- Redis 4.0及以上版本实例

  Redis 4.0及以上版本实例数据清空，可以使用Redis-cli客户端连接实例，执行flushdb或者flushall命令清空，也可以使用DCS控制台上的“数据清空”功能，一次全量清空Redis数据，还可以通过管理控制台的Web CLI功能连接Redis实例，使用flushdb命令进行清空。

  如果是Cluster集群实例，集群实例不支持多DB，由分片组成，如果使用命令清空，需要对集群每个分片都执行flushdb或者flushall命令，否则容易出现数据清空不彻底的问题。

  说明

  - 目前只有Redis 4.0及以上版本的实例支持在DCS控制台上执行“数据清空”功能及通过管理控制台的Web CLI功能连接Redis实例执行flushdb命令清空数据。
  - 在web cli界面使用flushdb命令，一次只会清理一个分片，如果有多个分片，需要用命令行连接到每个分片的主节点上，挨个执行flushdb。
  - Web CLI方式不支持清空Cluster集群的数据。

14.4.2 如何在 Redis 中查找匹配的 Key 和遍历所有 Key?

查找匹配 Key

在大Key和热Key分析中，不支持按照指定格式分析，如果需要查找指定前缀或者后缀格式的Key，您可以使用scan命令，根据指定格式进行匹配查找。

例如，需要查找Redis实例中包含a关键字的Key，可以使用Redis-cli工具，执行以下命令：

./redis-cli -h *{redis_address}* -p *{port}* [-a *password*] --scan --pattern '*a*'

遍历所有 Key

由于keys命令复杂度高，容易导致Redis无响应，所以不建议使用keys命令遍历实例所有的Key。如果需要在Redis实例中遍历所有的Key，可以使用Redis-cli工具，执行以下命令可以遍历Redis实例的所有key。

./redis-cli -h *{redis_address}* -p *{port}* [-a *password*] --scan --pattern '*'

scan命令的使用方法，可以参考**Redis官方网站**。

14.4.3 在 Web CLI 执行 keys 命令报错“permission denied”

Web CLI已禁用**keys**命令，请使用**Redis-cli**执行。

14.4.4 高危命令如何禁用?

Redis 4.0及以上版本的实例创建之后，支持重命名高危命令。当前支持重命名的高危命令有command、keys、flushdb、flushall、hgetall、scan、hscan、sscan、和zscan，其他命令暂时不支持。

您可以在创建实例时进行重命名以上高危命令，或在创建完成后，在缓存管理页面，选中实例，单击操作列的“更多 > 命令重命名”进行重命名以上高危命令。

说明

- 目前Redis不支持直接禁用命令，涉及到以上高危命令，可以使用命令重命名。关于DCS实例支持和禁用的命令请参考**开源命令兼容性**章节。
- 命令重命名提交后，系统会自动重启实例，实例完成重启后重命名生效。
- 因为涉及安全性，页面不会显示这些命令，请记住重命名后的命令。

14.4.5 是否支持 pipeline 命令?

支持。

注意：Redis Cluster版本集群实例使用pipeline时，要确保管道中的命令都能在同一分片执行。

14.4.6 Redis 是否支持 INCR/EXPIRE 等命令?

支持。命令兼容性相关说明请参考“**命令兼容性说明**”章节。

14.4.7 Redis 命令执行失败的可能原因

Redis命令执行失败，一般有以下可能原因：

- 命令拼写错误

  如下图所示，命令拼写有误，Redis实例返回“ERR unknown command”，删除key的正确命令为**del**。

  ```
  192.168.0.244:6379> delete hellokitty
  (error) ERR unknown command 'delete'
  192.168.0.244:6379> del hellokitty
  (integer) 1
  192.168.0.244:6379>
  ```

- 在低版本Redis实例运行高版本命令

  如下图所示，在Redis 3.0版本运行Redis 5.0新增的Stream相关命令，Redis实例返回命令出错信息。

  ```
  192.168.0.244:6379> xadd stream01 * field01 teststring
  (error) ERR unknown command 'xadd'
  192.168.0.244:6379> info server
  # Server
  redis_version:3.0.7.9
  redis_git_sha1:10fba618
  ```

- DCS Redis不支持的部分命令

  出于安全原因，DCS禁用了部分命令，具体参考**Redis命令的兼容性**，查看禁用命令与受限使用命令。

- 执行lua脚本失败

  例如报错：ERR unknown command 'EVAL' ，说明您的Redis实例属早期创建的低版本Redis实例，不支持lua脚本，这种情况请联系技术支持，升级您的Redis实例。

- 执行setname和getname失败

  说明您的Redis实例属早期创建的低版本Redis实例，不支持这两个命令，这种情况请联系技术支持，升级您的Redis实例。

14.4.8 Redis 命令执行不生效

如果客户端代码业务异常，怀疑是Redis命令不生效，则可以通过Redis-cli命令进行命令执行和数据查看，判断Redis命令执行是否异常。

以下列举两个场景：

- 场景一：通过设置key值和查看key值，即可判断该命令是否生效。

  Redis通过set命令写String类型数据，但是数据未变化，则可以使用Redis-cli命令访问Redis实例，执行如下命令：

  ```
  192.168.2.2:6379> set key_name key_value
  OK
  192.168.2.2:6379> get key_name
  "key_value"
  192.168.2.2:6379>
  ```

- 场景二：通过expire命令设置过期事件，但是怀疑过期时间不对，则可以执行如下操作:

  设置10秒过期时间，然后执行ttl命令查看过期时间，如下图表示，执行ttl命令时，过期时间剩下7秒。

  ```
  192.168.2.2:6379> expire key_name 10
  (integer) 1
  192.168.2.2:6379> ttl key_name
  (integer) 7
  192.168.2.2:6379>
  ```

说明

Redis客户端和服务端通过二进制协议进行通信，使用Redis-cli、Jedis、Python客户端并没有差异。

因此如果怀疑Redis有问题，但是使用Redis-cli排查没问题，那就很可能是业务代码存在问题，如果日志没有明显错误信息，则建议在代码添加日志支撑进一步分析。

14.4.9 Redis 命令执行是否有超时时间？超时了会出现什么结果?

Redis超时分为客户端超时和服务端超时。

- 客户端命令超时时间一般由客户端代码自行控制，业务侧需要根据自己的业务特点选择合适的超时时间（例如Java的Lettuce客户端，该参数名为timeout）。

  客户端如果发生命令执行超时，根据不同客户端的逻辑控制，可能会发生超时报错、命令堵塞、客户端连接重试等情况。

- Redis服务端Timeout默认配置为0，不会主动断开连接，如果需要修改配置，可以参考**修改实例配置参数**。

  如果实例配置了该Timeout参数值（不为0），当客户端与服务端空闲连接超过该参数值时，连接会断开。

14.4.10 Redis 的 Key 是否能设置为大小写不敏感?

DCS Redis和开源Redis保持一致，key对大小写敏感，且不支持设置大小写不敏感功能。

14.4.11 Redis 是否支持查看使用次数最多的命令?

Redis不支持对历史命令的记录，也不支持查看使用次数最多的命令。

14.4.12 Web CLI 的常见报错

1. ERR Wrong number of arguments for 'xxx' command

   该报错代表执行的Redis命令存在参数错误（语法错误），可以参考开源Redis命令协议介绍进行命令构造。

2. ERR unknown command 'xxx'

   该报错代表此命令为未知命令或者非redis协议定义的合法命令，可以参考开源Redis命令协议介绍进行命令构造。

3. ERR Unsupported command: 'xxx'

   该报错代表命令在DCS的Redis实例场景下禁用，可以参考**支持和禁用的Web CLI命令**。

14.5 扩容缩容与实例升级

14.5.1 Redis 实例是否支持大版本升级? 如 Redis 4.0 升级到 Redis 5.0?

目前仅Redis 3.0实例支持升级，操作详细请参见**升级Redis 3.0实例大版本**。

Redis 4.0及以上版本不支持升级实例大版本。Redis不同版本的底层架构不一样，在创建Redis实例时，确定Redis版本后，将不能修改，如Redis 4.0的实例不能升级到Redis 5.0。如您的业务需要使用Redis高版本的功能特性，可重新创建高版本Redis实例，然后将原有Redis实例的数据迁移到高版本实例上。具体数据迁移操作，可参考**使用DCS迁移数据**。

14.5.2 在维护时间窗内对实例维护是否有业务中断?

在实例维护时间窗内，服务运维要对实例进行维护操作时，会提前和用户沟通确认；具体升级操作以及影响，服务运维人员会提前和用户确认，用户不用担心维护窗内，实例运行异常的问题。

14.5.3 DCS 实例规格变更是否需要关闭或重启实例?

实例处于运行中的状态即可进行规格变更，不会涉及实例资源的重启操作。

14.5.4 DCS 实例规格变更的业务影响

执行实例规格变更操作，建议在业务低峰期进行，在实例规格变更时，会有如下影响。

实例类型变更前须知

- **支持实例类型变更明细如下：**
  - Redis 3.0和Memcached单机实例支持变更为主备实例，Redis 4.0/Redis 5.0/Redis 6.0单机实例不支持变更。
  - Redis 3.0主备实例支持变更为Proxy集群实例，Redis 4.0/Redis 5.0/Redis 6.0主备实例暂不支持变更。

    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。
  - 集群实例，不支持实例类型变更。
- **实例类型变更影响：**
  - Redis 3.0单机实例类型变更为Redis 3.0主备实例。

    连接会有秒级中断，大约1分钟左右的只读。
  - Redis 3.0主备实例类型变更为Redis 3.0 Proxy实例。

    连接会中断，5~30分钟只读。

实例规格大小变更前须知

- **支持扩容和缩容明细如下：**

  表 14-5 DCS 实例规格变更说明

缓存类型	单机实例	主备实例	Cluster集群实例	Proxy集群实例
Redis 3.0	支持扩容和缩容	支持扩容和缩容	不涉及	支持扩容
Redis 4.0/5.0	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	不涉及
Redis 6.0/7.0	支持扩容和缩容	支持扩容和缩容	支持扩容、缩容和副本数变更	不涉及
Memcached	支持扩容和缩容	支持扩容和缩容	不涉及	不涉及

  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。

  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。
- **实例规格大小变更影响如下：**
  - 单机和主备实例规格大小变更
    - Redis 4.0及以上版本实例变更期间，连接会有秒级中断，大约1分钟的只读。

  - Redis 3.0实例变更期间，连接会中断，5~30分钟只读。
  - 如果是扩容，只扩大实例的内存，不会提升CPU处理能力。
  - 如果是单机实例规格变更，由于单机实例不支持持久化，没有数据可靠性，变更实例可能会丢失数据。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。
  - 主备实例的备份记录，缩容后不能使用。
- 集群实例规格大小变更
  - 规格变更分片数未减少时，连接不中断，但会占用CPU，导致性能有20%以内的下降，扩容数据迁移期间，访问时延会增大。
  - 集群实例扩容会新增加数据节点，数据自动负载均衡到新的数据节点。
  - 规格变更分片数减少时，会删除节点，请确保应用中没有直接引用这些删除的节点。删除节点会导致连接闪断。
  - 规格变更后实例每个节点的已用内存必须小于节点最大内存的70%，否则将不允许变更。
  - 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，建议在业务低峰期进行。
  - 实例规格变更期间，会进行数据迁移，访问正在迁移的key时，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。
  - 请在规格变更前先使用缓存分析中的大key分析，确保实例中没有大key（≥512MB）存在，否则可能会导规格变更失败。
  - 变更规格前的备份记录不能恢复。

- **Redis实例副本数变更须知：**

  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。

14.5.5 Redis/Memcached 实例变更失败的原因

- 检查是否有其他任务在执行。

  实例变更过程中，同时有其他任务在执行。例如实例正在重启的同时，执行删除或扩容操作，或者实例正在扩容的时候，执行删除操作。

  遇到实例变更操作失败，可以稍后尝试，如果仍然存在问题，请技术支持。
- 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。

14.5.6 DCS 实例如何缩容?

DCS实例支持扩容和缩容明细如下：

表 14-6 DCS 实例规格变更说明

缓存类型	单机实例	主备实例	Cluster集群实例	Proxy集群实例
Redis 3.0	支持扩容和缩容	支持扩容和缩容	不涉及	支持扩容
Redis 4.0/5.0	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	不涉及
Redis 6.0/7.0	支持扩容和缩容	支持扩容和缩容	支持扩容、缩容和副本数变更	不涉及
Memcached	支持扩容和缩容	支持扩容和缩容	不涉及	不涉及

实例扩容、缩容操作请参考**变更规格**。

如果Redis 3.0 Proxy集群需要缩容，可以先进行数据备份，然后另外创建对应规格的Proxy集群实例，使用备份文件导入方式，将备份数据文件导入到新的Proxy集群实例。待数据迁移完成后，再释放原来规格的Proxy集群实例。在线迁移操作，可以参考**备份文件导入方式**。

14.5.7 使用 Lettuce 连接 Cluster 集群实例时，规格变更的异常处理

问题现象

使用lettuce连接Cluster集群实例，实例执行规格变更后，分片数有变化时，部分槽位（Slot）会迁移到新分片上，当客户端连接到新分片时会出现以下异常问题：

图 14-3 异常现象

```
org.springframework.data.redis.RedisSystemException: Redis exception; nested exception is io.lettuce.core.RedisException: io.lettuce.core.R
edisException: java.lang.IllegalArgumentException: Connection to 192.168.78.125:6379 not allowed. This connection point is not known in the
 cluster view
```

详情可参考Lettuce社区：**Connection to X not allowed. This connection point is not known in the cluster view**.

问题分析

Cluster集群规格变更原理：

客户端根据RESP2协议的内容，启动后从Cluster集群获取节点拓扑信息（Cluster Nodes），并将其拓扑关系维护在客户端的内存数据结构中。

对于数据访问，客户端会根据Key值按照CRC16算法进行Hash计算Slot信息，根据内存中保存的节点拓扑关系和Slot的对应信息进行请求自动路由。

在扩容/缩容过程中，当实例分片数发生变化时，存在节点拓扑关系和Slot对应信息的变化，需要客户端进行拓扑关系的自动更新，否则可能造成请求路由失败或者路由位置错误等，造成客户端访问报错。

例如，3分片Cluster集群实例扩容为6分片Cluster集群实例时，节点拓扑关系和Slot对应信息变化如下图所示：

图 14-4 Cluster 集群实例扩容前

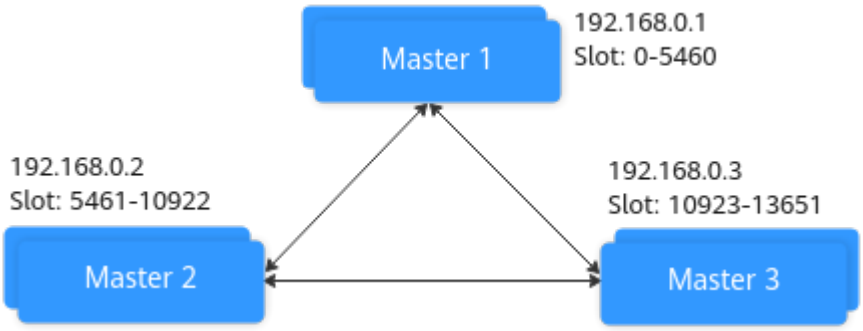
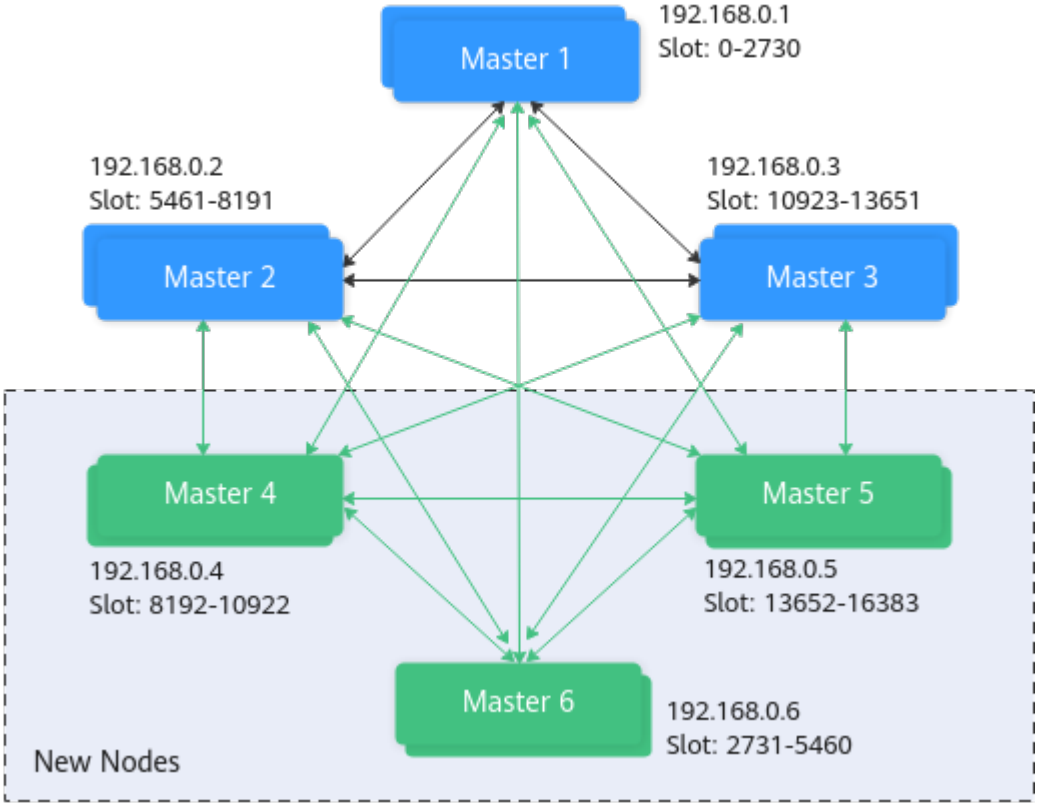


图 14-5 Cluster 集群实例扩容后



解决方案

方案一（推荐方案）：

开启Cluster集群自动刷新拓扑配置。

```
ClusterTopologyRefreshOptions topologyRefreshOptions = ClusterTopologyRefreshOptions.builder()
            // 每隔time毫秒周期性刷新
           .enablePeriodicRefresh(Duration.ofMillis(time))
       // MOVED重定向, ASK重定向, 重连, 未知节点(since 5.1), 槽位不在当前所有分片中(since 5.2),当出现这五种情况时会触发自适应刷新
```

```
        .enableAllAdaptiveRefreshTriggers()
        .build();
```

具体实现请参考**Lettuce客户端连接Cluster集群实例**。

说明

Lettuce客户端连接Cluster集群实例，如果未开启拓扑刷新，规格变更后，需要重启客户端。

方案二：

关闭“验证集群节点成员资格开关”，关闭方式如下：

```
ClusterClientOptions clusterClientOptions = ClusterClientOptions.builder()
        .validateClusterNodeMembership(false)
        .build();
```

原理：若validateClusterNodeMembership为true时，连接前检查当前连接地址是否在集群拓扑关系中（通过CLUSTER NODES获得），若不在则会出现上述异常问题。

说明

关闭“验证集群节点成员资格开关”的影响：

- 缺少防止安全漏洞的检验；
- 若未开启集群自动刷新拓扑，当Cluster集群执行变更规格后，若分片数增加时，可能会产生MOVED重定向请求，这个重定向过程会增加集群的网络负担和单次请求耗时；若分片数因删除减少时，会出现无法连接已删除分片的异常情况。

14.6 监控告警

14.6.1 如何查看 Redis 实例的实时并发连接数和最大连接数

查看 Redis 实例实时并发连接数

当您需要查看DCS实例收到的实时连接数时，可以通过控制台查看，查看方法，请参考**查看监控指标**。

进入监控页面后，找到“**活跃的客户端数量**”监控项。您可以单击该监控项的右上角的查看按钮  ，使用大图模式查看。

在弹出的“活跃的客户端数量”页面，根据需要选择查看的时间段，例如，若要查看10分钟内的连接数，您可以将时间自定义为10分钟。由于监控数据采集的是周期内增加的连接数，您可以通过监控图表，查看这个时间段的连接数的走势，并统计10分钟内的连接总数。

另外，您还可以通过以上操作方法查看其它常用监控数据，查看您的实例运行情况。

- CPU利用率
- 内存利用率
- 已用内存
- 每秒并发操作数

查看或修改实例最大连接数

您可以在控制台创建实例页面查看实例默认及最大可配的最大连接数。

创建实例后，您也可以通过DCS控制台“实例配置 > 参数配置”页面，查看或修改**maxclients**参数值，即最大连接数。（ Proxy集群实例不支持修改该参数 ）

14.6.2 Redis 命令是否支持审计?

Redis是高性能读写，不支持命令审计，如果命令支持审计，性能会受到很大的影响，所以命令打印不出来。

14.6.3 Redis 监控数据异常处理方法

当对Redis监控数据存在疑问或异议时，可以使用Redis-cli访问Redis实例，执行**info all**命令，查看进程记录的指标。info all输出详解可参考：**https://redis.io/docs/latest/commands/info/**。

14.6.4 为什么实例实际可用内存比申请规格小而且已使用内存不为0?

由于系统开销会占用部分资源，主备实例的持久化也需要一部分资源，所以Redis 3.0和Memcached实例创建后，缓存实例实际可用内存小于申请规格。除了用户存储数据外，Redis-server内部的buffer以及内部数据结构会占用一部分内存。所以缓存实例创建后，实例已使用内存量不为0。

14.6.5 监控数据出现实例已使用内存略大于实例可使用内存是什么原因?

DCS单机和主备实例已使用内存为redis-server进程统计的已使用内存。集群是基于分片机制实现的，集群的已使用内存为各个分片redis-server的已使用内存的总和。

由于开源redis-server内部机制的原因，有时会出现DCS缓存实例已使用内存略大于可使用内存的情况，此为正常现象。

Redis的used_memory超过max_memory的原因

Redis通过zmalloc分配内存，不会在每一次分配内存时都检查是否会超过max_memory，而是在周期任务以及命令处理的开头处等地方，判断一次当前的used_memory是否超过max_memory，如果超过就触发逐出操作。所以，对于max_memory策略的限制实施并不是实时、刚性的，会出现某个时间used_memory大于max_memory的情形。

14.6.6 为什么带宽使用率指标会超过 100%

带宽使用率基本信息如下:

指标ID	指标名称	含义	取值范围	测量对象&维度	监控周期（原始指标）
bandwidth_usage	带宽使用率	当前流量带宽与最大带宽限制的百分比	0-200%	测量对象： Redis 4.0及以上版本 主备、集群实例数据节点 测量维度： dcs_cluster_node	1分钟

其中，带宽使用率的计算公式为：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%。

该公式中同时计算了网络瞬时输入流量和网络瞬时输出流量，这两个指标值是有统计主从同步的流量的。所以统计的总流量使用量会比正常的业务流量大一些，会发生带宽使用率指标超过100%的情况。

判断当前是否被限流，请使用**流控次数**这个指标，这个指标值大于0时，表示当前已使用的带宽超过最大带宽限制，产生流控。

限流时，流控次数指标是不统计主从同步流量的，所以有时候会出现带宽使用率指标超过100%，但流控次数为0的情况。

14.6.7 监控指标中存在已拒绝的连接数是什么原因?

当监控指标中出现“已拒绝的连接数”时，请确认客户端连接数是否已经超过实例的最大连接数限制。

说明

Redis 4.0及以上版本的实例，仅在主备、集群和读写分离实例的数据节点中支持查看“已拒绝的连接数”。

- 查看最大连接数：单击实例名称，进入实例详情页面，选择“实例配置 > 参数配置”页签，查看maxclients参数的值（读写分离实例暂不支持该参数，可通过**实例规格**查询实例最大连接数）。
- 查看实际连接数：单击实例名称，进入实例详情页面，选择“性能监控”页签，找到“活跃的客户端数量”监控项查看。

如果客户端连接数已到达连接上限，可以根据需要调整maxclients参数，如果maxclients参数已经是最大可配连接数，仍不满足需求，则需要考虑增加实例分片。

14.6.8 触发限流（流控）的原因和处理建议

Redis产生流控，说明redis在周期内的使用流量超过该实例规格的最大带宽。

说明

实例规格对应的最大带宽，可以查看**实例规格**，或在控制台的创建实例页面查看对应实例类型的“基准/最大带宽”。

带宽使用率不高时，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。而流控检查是秒级的，有可能存在上报周期间隔期间，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。

对于主备实例：

- 如果实例一直有流控但是带宽使用率不高，这说明可能存在业务微突发问题，或者大Key热Key问题，建议对实例进行自动诊断分析，优先排除大Key热Key问题。
- 如果带宽使用率居高不下，说明带宽可能存在超限风险，需要扩容处理（容量越大，带宽越大）。

对于集群实例：

- 仅有单个或少量几个分片出现流控，则多数为该分片存在大Key热Key问题。
- 所有或大多数分片同时出现流控或者带宽使用率高的问题，这说明实例的带宽达到了瓶颈，建议扩容实例。

说明

- DCS控制台提供了大Key和热Key的分析功能，您可参考**缓存分析**减少大key和热key。
- 如果用户执行了keys等消耗资源的命令，也可能会导致CPU和带宽使用率增加，从而出现流控。

14.7 数据备份/导出/迁移

14.7.1 如何导出 Redis 实例数据?

- 主备或集群实例：

  主备和集群实例支持备份功能，可以执行以下操作将数据导出：

  a. 进入缓存管理页面，切换到“备份与恢复”页签，查看实例的备份记录。
  b. 如没有记录，则手动执行备份动作，执行完后，单击“下载”，根据提示完成数据的下载操作。

  说明

  如果您的实例创建时间非常早，由于实例版本没有升级而无法兼容备份恢复功能，请联系技术支持将缓存实例升级到最新版本，升级后就可以支持备份恢复功能。

- 单机实例：

  单机实例不支持备份功能，用户可以通过Redis-cli客户端导出rdb文件，但是使用Redis-cli导出rdb文件依赖SYNC命名。

  - 放通了SYNC命令的单机实例（例如Redis 3.0单机实例，未禁用SYNC命令），可以通过执行以下命令，将单机实例上的数据导出：

    redis-cli -h {source_redis_address} -p 6379 [-a password] --rdb {output.rdb}

  - 禁用了SYNC命令的单机实例（例如Redis 4.0和Redis 5.0单机实例，禁用了SYNC命令），建议将单机实例的数据迁移到主备实例，然后使用主备实例的备份功能。

14.7.2 是否支持控制台导出 RDB 格式的 Redis 备份文件?

- Redis 3.0实例

  Redis 3.0是通过AOF文件持久化的，控制台仅支持备份和下载AOF文件，RDB格式文件可以通过Redis-cli导出：

  redis-cli -h *{redis_address}* -p 6379 [-a *password*] --rdb {output.rdb}

- Redis 4.0及以上版本实例

  Redis 4.0及以上版本实例支持选择AOF和RDB格式进行持久化，支持在控制台备份和下载AOF和RDB文件。

14.7.3 迁移过程中为什么进程总是被 kill?

可能原因：内存不够。

解决方案：对执行迁移命令的服务器扩充内存。

14.7.4 Redis 在线数据迁移是迁移整个实例数据么?

如果是单机和主备实例之间进行迁移，是迁移实例所有的数据，不管存在哪个DB都会进行迁移，且数据所在的DB序号不会变；

如果是集群实例，由于集群实例只有一个DB0节点，会迁移DB0上所有槽内的数据。

14.7.5 Redis 实例支持数据持久化吗？开启持久化有什么影响?

是否支持持久化

单机：不支持持久化。

主备和集群（单副本集群除外）：支持持久化。

Redis 实例支持的持久化方式

- Redis实例默认仅支持AOF的方式进行持久化，同时支持客户自行开关数据持久化配置。创建的实例（单机或单副本集群除外）默认开启AOF持久化。
- Redis实例默认不支持RDB持久化，因此也无法支持客户自行配置save参数。如果需要进行RDB持久化，可以使用主备或者集群实例的备份恢复功能，备份恢复时，Redis 4.0及以上版本实例，可以支持选择生成RDB持久化文件并且自动转储到OBS中。

持久化的磁盘是什么类型

Redis 4.0及以上版本的实例，持久化的磁盘是SSD类型。

开启/关闭 AOF 持久化的影响

开启AOF持久化后，由于Redis-Server进程需要在AOF文件中记录对应的操作信息，用来进行数据持久化。开启持久化可能存在的影响：

- 当出现底层计算节点磁盘硬件故障或者IO故障时，可能会造成时延冲高或者主备倒换等情况发生。
- Redis-Server进程会定期进行AOF重写操作，重写期间可能会造成短暂的时延冲高，AOF重写规则请参考**AOF文件在什么情况下会被重写**。

如果在缓存场景下使用DCS实例进行应用加速，建议可以关闭持久化参数以获得更高的性能和稳定性。

关闭持久化需根据实际业务慎重操作，关闭持久化后在极端故障场景（例如主备节点同时故障等）下可能出现缓存数据丢失的问题。

如何开启/关闭 Redis 持久化

在实例的配置参数中将appendonly参数设置为no即可关闭AOF持久化，设置为yes即开启AOF持久化。（单机实例不支持持久化）

配置参数的操作请参考**修改实例配置参数**。

14.7.6 AOF 文件在什么情况下会被重写

AOF文件重写涉及到以下概念。

- 重写时间窗：目前该时间窗为凌晨1:00 - 4:59。
- 磁盘阈值：即磁盘的使用率超过50%，即认为达到阈值。
- 数据集使用内存：实例的一个监控指标，用于统计Redis中数据集占用的内存。

AOF文件在以下三种情况下会被重写。

- 如果磁盘达到阈值，无论是否处于时间窗内：当AOF文件大小 > 数据集使用内存时，实例AOF文件会被重写。
- 如果磁盘未达到阈值，处于重写时间窗内：当AOF文件大小 > 数据集使用内存的1.5倍时，实例AOF文件会被重写。
- 如果磁盘未达到阈值，未处于重写时间窗内：当AOF文件大小 > 实例最大内存的4.5倍时，实例AOF文件会被重写。

14.7.7 一个数据迁移能迁移到多个目标实例么？

不能，一个迁移任务只能迁移到一个目标实例。要迁移到多个目标实例需要创建多个迁移任务。

14.7.8 怎么放通 SYNC 和 PSYNC 命令？

- DCS云服务内部的Redis之间进行迁移：
  - 如果迁移任务和源端实例在相同账号下的相同Region，在配置在线迁移任务时，源端实例通过选择DCS实例（云服务Redis）的方式进行配置，会自动放通源端实例的SYNC和PSYNC命令。
  - 如果迁移任务和源端实例在不同账号或不同Region，在配置在线迁移任务时，源端实例不能通过选择DCS实例（云服务Redis）的方式进行配置，不会自动放通源端实例的SYNC和PSYNC命令，因此无法使用控制台的在线迁移。推荐使用备份文件导入方式迁移。
  - 自建Redis迁移至DCS，默认没有禁用SYNC和PSYNC命令。
- 其他云厂商迁移到DCS云服务：
  - 一般云厂商都是禁用了SYNC和PSYNC命令，如果使用DCS控制台的在线迁移功能，需要联系源端的云厂商运维人员放通此命令。离线迁移，推荐使用备份文件导入方式。
  - 如果不需要增量迁移，可以参考**使用Redis-shake工具在线全量迁移其他云厂商Redis**进行全量迁移，该方式不依赖于SYNC和PSYNC。

14.7.9 迁移或导入备份数据时，相同的 Key 会被覆盖吗?

在迁移或导入备份数据时，源端与目标端重复的数据会被覆盖；源端没有，目标端有的数据会保留。

因此，如果在迁移后目标端与源端数据不一致，可能是目标端在迁移前有未清除的数据。

14.7.10 使用 Rump 在线迁移

背景说明

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互相迁移，以及不同实例的数据库之间迁移。

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/写值独立于关键工作。

以下是Rump的主要特性：

- 通过SCAN非阻塞式获取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。

须知

1. Rump工具不支持迁移到DCS集群实例。请改用其他工具，如redis-port或Redis-cli。
2. 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。
3. 建议停业务迁移。迁移过程中如果不断写入新的数据，可能会丢失少量Key。

步骤 1：安装 Rump

1. 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;
2. 解压缩后，添加可执行权限。

   mv rump-0.0.3-linux-amd64 rump;

   chmod +x rump;

步骤 2：迁移数据

rump -from {*source_redis_address*} **-to** {*target_redis_address*}

参数/选项说明：

- *{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 ~]#
```

14.7.11 不同类型的操作系统间进行数据传递和操作，需要注意什么？

建议将数据文件格式转换后再执行导入。

windows系统转换成类unix系统的文件格式：

dos2unix *{filename}*

类unix系统转换成windows系统的文件格式：

unix2dos *{filename}*

14.7.12 源 Redis 使用了多 DB，能否迁移数据到集群实例？

DCS单机和主备实例支持256个库，编号0～255。

- 如果目的实例为集群实例。集群实例只有1个库。

  两个解决思路：

  a. 源Redis的不同DB合到同一个数据库。

  b. 申请多个DCS缓存实例。

  迁移后实例连接地址和数据库编号有变化，业务注意改造和适配。

14.7.13 只想迁移部分数据时应该怎么处理？

控制台在线迁移功能，不支持迁移指定数据库。如果需要单独迁移Redis中的**指定数据库**，可以使用RedisShake导出或者导入指定的数据库。

RedisShake的安装和使用可以参考**使用Redis-Shake工具迁移自建Redis Cluster集群**和**Redis-shake配置说明**。

如果单独迁移**指定数据**，建议自行开发脚本，获取指定的key及数据，然后导入DCS缓存实例中。

14.7.14 源 Redis 迁移到集群实例中有哪些限制和注意事项?

- Proxy版集群实例

  使用方式与单机、主备实例类似，但是默认只有1个DB，不支持select命令。数据文件批量导入时，遇到select命令会返回错误提示并忽略，同时继续将剩余数据导入。

  举例：

  源Redis在数据库编号0和2中有数据，生成的AOF或RDB文件包含了这两个库。

  在导入到Proxy集群实例时会忽略“select 2”的命令，然后继续导入源数据库2中的数据到DB0中。

  用户需要注意以下：

  - 源Redis中不同数据库包含了相同的key，则导入时，编号靠前的数据库的key的value会被靠后的数据库中的key覆盖。
  - 源Redis使用了多个数据库，数据迁移到DCS集群实例后，都存储在同一数据库中，不支持select命令。业务需要做适配。

- Cluster版集群实例

  Cluster版集群除了只有1个DB外，导入方式与其他类型的Redis实例也有差异。Cluster集群的数据，必须由客户端分别连接各分片节点，将数据分别导入。各分片节点的IP地址查询命令：

  redis-cli -h *{Redis Cluster IP}* -p 6379 -a *{password}* cluster nodes

  返回的节点地址清单中，标记为master的节点IP地址即为Cluster集群的分片节点地址。

14.7.15 在线迁移需要注意哪些?

- 网络

  在线迁移首先需要打通网络，迁移任务必须和源Redis、DCS缓存实例二者网络互通。

- 工具

  在线迁移工具，推荐使用DCS控制台的在线迁移功能。

- 数据完整性

  如果选择中断业务，则迁移完成后检查数据量和关键key。

  如果选择不中断业务，则用户需要考虑增量数据的迁移。

- 迁移过程源端扩容影响迁移结果

  在线迁移期间源端扩容操作会影响迁移，有可能导致迁移失败，也有可能会影响客户的数据，客户如果在迁移期间源端实例的内存不够用需要扩容，建议先中断迁移任务 ，然后再扩容。

- 迁移时间

  迁移操作建议在业务低峰期进行。

- 版本限制

  低版本可以到高版本，高版本也可以到低版本，不同版本，在迁移时需要分析业务系统使用到的缓存命令在目的端实例是否兼容。

14.7.16 在线迁移能否做到完全不中断业务?

可以使用应用双写的方式，即在迁移过程中业务数据继续从源Redis中正常读取，同时将数据的增删改操作在DCS的Redis实例中执行一遍。

保持以上状态运行一段时间后（等待较多的旧数据过期删除），把系统的缓存数据库正式切到DCS。如涉及业务系统迁移云服务，需要在缓存数据库切换前完成业务系统的部署。

不推荐使用这种方式。原因如下：

1. 网络无法保证稳定快速，如果源Redis实例不在DCS，则需要使用公网访问DCS，效率不高。
2. 同时写2份数据，需要用户自行修改代码实现。
3. 源Redis实例的数据逐出策略各有差异，迁移耗时可能较长，数据完整性保障难度大。

14.7.17 在线迁移实例源端报“Disconnecting timedout slave”和“overcoming of output buffer limits”

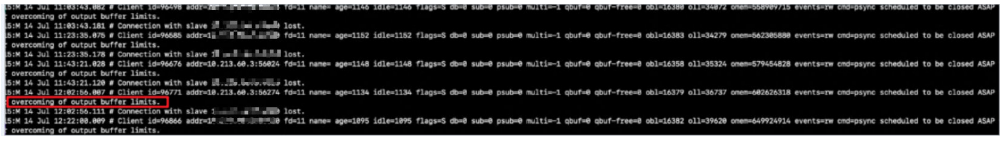
当进行在线迁移时可能会出现如下报错：

- 源端报“**Disconnecting timedout slave**”，如下图：

```
19361:M 30 Aug 18:01:16.567 # Disconnecting timedout slave: 1
19361:M 30 Aug 18:01:16.567 # Connection with slave  lost.
19361:M 30 Aug 18:01:39.354 * Slave :<unknown-slave-port> asks for synchronization
19361:M 30 Aug 18:01:39.354 * Full resync requested by slave :<unknown-slave-port>
19361:M 30 Aug 18:01:39.354 * Starting BGSAVE for SYNC with target: disk
19361:M 30 Aug 18:01:39.686 * Background saving started by pid 56274
56274:C 30 Aug 18:02:44.339 * DB saved on disk
56274:C 30 Aug 18:02:44.611 * RDB: 1477 MB of memory used by copy-on-write
19361:M 30 Aug 18:02:45.203 * Background saving terminated with success
19361:M 30 Aug 18:02:58.117 * Synchronization with slave <unknown-slave-port> succeeded
19361:M 30 Aug 18:04:59.281 # Disconnecting timedout slave: :<unknown-slave-port>
19361:M 30 Aug 18:04:59.281 # Connection with slave :<unknown-slave-port> lost.
19361:M 30 Aug 18:05:25.059 * Slave  asks for synchronization
19361:M 30 Aug 18:05:25.059 * Full resync requested by slave 1
19361:M 30 Aug 18:05:25.059 * Starting BGSAVE for SYNC with target: disk
19361:M 30 Aug 18:05:25.395 * Background saving started by pid 3256
3256:C 30 Aug 18:06:33.029 * DB saved on disk
```

  解决方法：建议将源端Redis实例的repl-timeout参数值配置为300秒。

- 源端报“**overcoming of output buffer limits**”，如下图：



解决方法：建议将源端Redis实例的client-output-buffer-limit参数值配置为实例最大内存的20%。如果源端Redis是DCS实例，请配置client-output-buffer-slave-hard-limit和client-output-buffer-slave-soft-limit参数值为实例最大内存的20%。

14.7.18 使用 Rump 工具迁移数据，命令执行后无报错，但 Redis 容量无变化

Rump工具的具体使用，请参考**数据迁移指南**。

可能原因：

- Rump工具不支持迁移到集群实例。
- Rump命令参数有误。

14.7.19 DCS 实例是否兼容低版本 Redis 迁移到高版本

支持，目前Redis高版本是支持兼容低版本的。

源端是DCS Redis，自建Redis，或者其他云厂商Redis的低版本或相同版本实例，都可以迁移到DCS的目标端实例。

14.8 大 Key/热 Key 分析

14.8.1 什么是大 Key/热 Key?

名词	定义
大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。

14.8.2 存在大 Key/热 Key，有什么影响?

类别	影响
大Key	**造成规格变更失败。** Redis集群变更规格过程中会进行数据rebalance（节点间迁移数据），单个Key过大的时候会触发Redis内核对于单Key的迁移限制，造成数据迁移超时失败，Key越大失败的概率越高，大于512MB的Key可能会触发该问题。
	造成数据迁移失败。 数据迁移过程中，如果一个大Key的元素过多，则会阻塞后续Key的迁移，后续Key的数据会放到迁移机的内存Buffer中，如果阻塞时间太久，则会导致迁移失败。

类别	影响
	容易造成集群分片不均的情况。 ● 各分片内存使用不均。例如某个分片占用内存较高甚至首先使用满，导致该分片Key被逐出，同时也会造成其他分片的资源浪费。 ● 各分片的带宽使用不均。例如某个分片被频繁流控，其他分片则没有这种情况。
	客户端执行命令的时延变大。 对大Key进行的慢操作会导致后续的命令被阻塞，从而导致一系列慢查询。
	导致实例流控。 对大Key高频率的读会使得实例出方向带宽被打满，导致流控，产生大量命令超时或者慢查询，业务受损。
	导致主备倒换。 对大Key执行危险的DEL操作可能会导致主节点长时间阻塞，从而导致主备倒换。
热Key	**容易造成集群分片不均的情况。** 造成热Key所在的分片有大量业务访问而同时其他的分片压力较低。这样不仅会容易产生单分片性能瓶颈，还会浪费其他分片的计算资源。
	使得CPU冲高。 对热Key的大量操作可能会使得CPU冲高，如果表现在集群单分片中就可以明显地看到热Key所在的分片CPU使用率较高。这样会导致其他请求受到影响，产生慢查询，同时影响整体性能。业务量突增场景下甚至会导致主备切换。
	易造成缓存击穿。 热Key的请求压力过大，超出Redis的承受能力易造成缓存击穿，即大量请求将被直接指向后端的数据库，导致数据库访问量激增甚至宕机，从而影响其他业务。

14.8.3 为了减少大 Key 和热 Key 过大，有什么使用建议?

- **string类型控制在10KB以内**，hash、list、set、zset**元素尽量不超过5000个**。
- Key的命名前缀为业务缩写，禁止包含特殊字符（比如空格、换行、单双引号以及其他转义字符）。
- Redis事务功能较弱，不建议过多使用。
- 短连接性能差，推荐使用带有连接池的客户端。
- 如果只是用于数据缓存，容忍数据丢失，建议关闭持久化。
- 大Key/热Key的优化方法，请参考下表。

类别	方法
大Key	**进行大Key拆分。** 分为以下几种场景： ● **该对象为String类型的大Key：**可以尝试将对象分拆成几个Key-Value， 使用MGET或者多个GET组成的pipeline获取值，分拆单次操作的压力。如果是集群实例，由于集群实例包含多个分片，拆分后的Key会自动平摊到集群实例的多个分片上，从而降低对单个分片的影响。 ● **该对象为集合类型的大Key，并且需要整存整取：**在设计上严格禁止这种场景的出现，因为无法拆分。有效的方法是将该大Key从Redis去除，单独放到其余存储介质上。 ● **该对象为集合类型的大Key，每次只需操作部分元素：**将集合类型中的元素分拆。以Hash类型为例，可以在客户端定义一个分拆Key的数量N，每次对HGET和HSET操作的field计算哈希值并取模N，确定该field落在哪个Key上，实现上类似于Redis Cluster的计算slot的算法。
	将大Key单独转移到其余存储介质。 无法拆分的大Key建议使用此方法，将不适用Redis能力的数据存至其它存储介质，并在Redis中删除该大Key。 **注意** 禁止使用DEL直接删除大Key，可能会造成Redis阻塞，甚至主备倒换。
热Key	**使用客户端缓存/本地缓存。** 该方案需要提前了解业务的热点Key有哪些，设计客户端/本地和远端Redis的两级缓存架构，热点数据优先从本地缓存获取，写入时同时更新，这样能够分担热点数据的大部分读压力。缺点是需要修改客户端架构和代码，改造成本较高。
	设计熔断/降级机制。 热Key极易造成缓存击穿，高峰期请求都直接透传到后端数据库上，从而导致业务雪崩。因此热Key的优化一定需要设计系统的熔断/降级机制，在发生击穿的场景下进行限流和服务降级，保护系统的可用性。

14.8.4 如何分析 Redis 3.0 实例的热 Key?

由于Redis 3.0本身不提供热Key能力，您可以参考以下方法进行分析。

- 方法1：进行业务结构和业务实现分析，找到可能的热Key。
  例如，某商品在秒杀，或者用户登录，对业务代码分析，很容易找到热Key。
  优点：简单易行。
  缺点：需要对业务代码比较了解，另外对于一些复杂的业务场景，不太容易分析。
- 方法2：在客户端代码中，调用Redis的函数中，进行访问Key的记录，进而统计出热Key。
  缺点：需要代码进行侵入式修改。

- 方法3：抓包分析
  优点：简单易行

14.8.5 如何提前发现大 Key 和热 Key?

方法	说明
使用DCS自带的大Key和热Key分析工具进行分析	请参考**缓存分析**。
通过redis-cli的bigkeys和hotkeys参数查找大Key和热Key	• Redis-cli提供了bigkeys参数，能够使redis-cli以遍历的方式分析Redis实例中的所有Key，并返回Key的整体统计信息与每个数据类型中Top1的大Key，bigkeys仅能分析并输入六种数据类型（STRING、LIST、HASH、SET、ZSET、STREAM），命令示例为：**redis-cli -h <实例的连接地址> -p <端口> -a <密码> --bigkeys**。 • 自Redis 4.0版本起，redis-cli提供了hotkeys参数，可以快速帮您找出业务中的热Key，该命令需要在业务实际运行期间执行，以统计运行期间的热Key。命令示例为：**redis-cli -h <实例的连接地址> -p <端口> -a <密码> --hotkeys**。热Key的详情可以在结果中的summary部分获取到。

对于Redis 3.0实例，由于Redis 3.0本身不支持热Key分析，推荐可以使用**配置告警**的方法，帮助您发现热Key。

- 配置节点级别的**内存利用率**监控指标的告警
  如果某个节点存在大key，这个节点比其他节点内存使用率高很多，会触发告警，便于用户发现潜在的大key。
- 配置节点级别的**入网最大带宽**、**出网最大带宽**、**CPU利用率**监控指标的告警
  如果某个节点存在热key，这个节点的带宽占用、CPU利用率都比其他节点高，该节点会容易触发告警，便于用户发现潜在热key。

14.9 主备倒换

14.9.1 发生主备倒换的原因有哪些?

主备倒换有以下几种可能的场景：

- 用户自行从DCS控制台界面发起“主备倒换”操作，切换主实例。
- DCS检测到主备实例的主节点存在故障后，触发实例“主备倒换”操作。
  例如，使用了keys等消耗资源的命令，导致CPU超高，触发主备导致。
- 用户在DCS界面上执行重启操作，可能触发备节点升为主节点，即主备倒换。
- 单机、主备实例在扩容过程中，会发生主备倒换。
  扩容过程中，实例会创建新规格的节点作为备节点，主节点数据全量+增量同步到备节点后进行主备切换并删除原节点，完成扩容。

发生主备倒换后，系统会上报主备倒换事件，收到该事件通知后，请查看客户端业务否存在异常，如果业务不正常，则需要确认客户端tcp连接是否正常，是否支持在主备倒换后重新建立tcp连接恢复业务。

14.9.2 主备倒换的业务影响

DCS主备或者集群实例发生异常时，会触发内部主备倒换，并自动恢复，在异常检测和恢复期间，可能会影响业务，时间在半分钟内。

14.9.3 主备实例发生主备倒换后是否需要客户端切换 IP?

不需要。主节点故障后，IP地址绑定到备节点，绑定后，原备节点升级为主节点。

14.9.4 Redis 主备同步机制怎样?

Redis主备实例即主从实例。一般情况下，Redis主节点的更新会自动复制到关联的备节点。但由于Redis异步复制的技术，备节点更新可能会落后于主节点。例如，主节点的I/O写入速度超过了备节点的同步速度，或者因异常原因导致的主节点和备节点的网络延迟，使得备节点与主节点存在滞后或者部分数据不一致，若此时进行主备切换，未及时完成同步的少量数据可能会丢失。

15 故障排除

15.1 Redis 连接失败问题排查和解决

概述

本章节主要描述Redis连接过程出现的问题，以及解决方法。

问题分类

当您发现与Redis实例连接出现异常时，可以根据本文的内容，从以下几个方面进行排查。

- **Redis和ECS之间的连接问题**
- **密码问题**
- **实例配置问题**
- **客户端连接问题**
- **带宽超限导致连接问题**
- **性能问题导致连接超时**

Redis 和 ECS 之间的连接问题

客户端所在的ECS必须和Redis实例在同一个VPC内，并且需要确保ECS和Redis之间可以正常连接。

- 如果是Redis 3.0实例，Redis和ECS的安全组没有配置正确，连接失败。

  解决方法：配置ECS和Redis实例所在安全组规则，允许Redis实例被访问。具体配置，可以参考**安全组配置和选择**。
- 如果是Redis 4.0及以上版本实例，开启了白名单功能，连接失败。

  如果实例开启了白名单，在使用客户端连接时，需要确保客户端IP是否在白名单内，如果不在白名单，会出现连接失败。具体配置操作，可以参考**配置实例白名单**。客户端IP如果有变化，需要将变化后的IP加入白名单。
- Redis实例和ECS不在同一个Region。

解决方法：不支持跨Region访问，可以在ECS所在的Region创建Redis实例，创建时注意选择与ECS相同VPC，创建之后，使用**数据迁移**进行迁移，将原有Redis实例数据迁移到新实例中。

- Redis实例和ECS不在同一个VPC。

  不同的VPC，网络是不相通的，不在同一VPC下的ECS无法访问Redis实例。可以通过创建VPC对等连接，将两个VPC的网络打通，实现跨VPC访问Redis实例。

  关于创建和使用VPC对等连接，请参考《虚拟私有云用户指南》的“对等连接”文档说明。

密码问题

密码输入错误时，端口可以连接上，但鉴权认证会失败。如果忘记了密码，可以**重置缓存实例密码**。

实例配置问题

连接Redis时存在拒绝连接，可登录分布式缓存服务控制台，进入实例详情页面，调整实例参数maxclients的配置，具体操作可参考**修改实例配置参数**。

客户端连接问题

- 在使用Redis-cli连接Cluster集群时，连接失败。

  解决方法：请检查连接命令是否加上**-c**，在连接Cluster集群节点时务必使用正确连接命令。

  - Cluster集群连接命令：

    ./redis-cli -h *{dcs_instance_address}* -p 6379 -a {password} -c
  - 单机、主备、Proxy集群连接命令：

    ./redis-cli -h *{dcs_instance_address}* -p 6379 -a {password}

  具体连接操作，请参考**Redis-cli连接**。
- 出现Read timed out或Could not get a resource from the pool。

  解决方法：

  - 排查是否使用了keys命令，keys命令会消耗大量资源，造成Redis阻塞。建议使用scan命令替代，且避免频繁执行。
  - 排查实例是否是Redis 3.0，Redis 3.0底层用的是sata盘，当Redis数据持久化即AOF时，会触发偶现的磁盘性能问题，导致连接异常，可更换Redis实例为4.0 和5.0版本，其底层是ssd盘，磁盘性能更高，或若不需要持久化可关闭AOF。
- 出现unexpected end of stream错误，导致业务异常。

  解决方法：

  - Jedis连接池调优，建议参考**Jedis参数配置建议**进行配置连接池参数。
  - 排查是否大key较多，建议根据**优化大key**排查优化。
- 连接断开。

  解决方法：

  - 调整应用超时时间。
  - 优化业务，避免出现慢查询。
  - 建议使用scan命令替代keys命令。

- Jedis连接池问题，请参考**使用Jedis连接池报错如何处理？**。

带宽超限导致连接问题

当实例已使用带宽达到实例规格最大带宽，可能会导致部分Redis连接超时现象。

您可以查看监控指标“流控次数”，统计周期内被流控的次数，确认带宽是否已经达到上限。

然后，检查实例是否有大Key和热Key，如果存在大Key或者单个Key负载过大，容易造成对于单个Key的操作占用带宽资源过高。大Key和热Key操作，请参考**缓存分析**。

性能问题导致连接超时

使用了keys等消耗资源的命令，导致CPU使用率超高；或者实例没有设置过期时间、没有清除已过期的Key，导致存储的数据过多，一直在内存中，内存使用率过高等，这些都容易出现访问缓慢、连接不上等情况。

- 建议客户改成scan命令或者禁用keys命令。
- 查看监控指标，并配置对应的告警。监控项和配置告警步骤，可查看**必须配置的告警监控**。

  例如，可以通过监控指标“内存利用率”和“已用内存”查看实例内存使用情况、“活跃的客户端数量”查看实例连接数是否达到上限等。
- 检查实例是否存在大Key和热Key。

  DCS控制台提供了大Key和热Key的分析功能，具体使用，请参考**缓存分析**。

15.2 Redis 实例 CPU 使用率高问题排查和解决

问题现象

Redis实例CPU使用率短时间内冲高。

可能原因

1. 客户的业务负载过重，QPS过高，导致CPU被用满，排查方法请参考**排查QPS是否过高**。
2. 使用了keys等消耗资源的命令，排查及处理措施请参考**查找并禁用高消耗命令**。
3. 发生Redis的持久化重写操作，排查及处理措施请参考**是否存在Redis的持久化重写操作**。

排查 QPS 是否过高

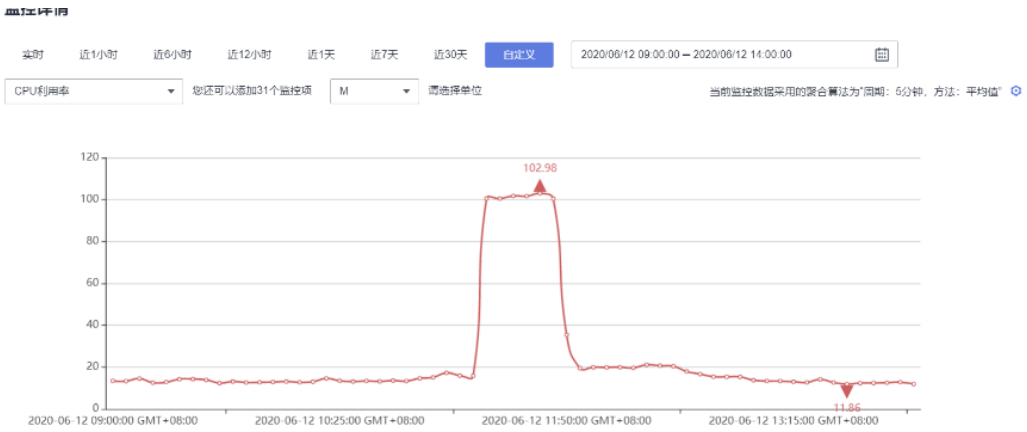
在分布式缓存服务控制台的缓存管理页面，单击实例进入实例详情界面，单击左侧的性能监控，进入性能监控页面，可以查询实例级别的每秒并发操作数（ QPS ）。如果QPS过高，建议优化客户业务或者**升级实例规格**，不同实例规格的QPS请参考**实例规格**章节。

查找并禁用高消耗命令

使用了keys等消耗资源的命令，高消耗资源的命令即时间复杂度为O(N)或更高的命令，通常情况下，命令时间复杂度越高，在执行时消耗的资源越高，这会导致CPU使

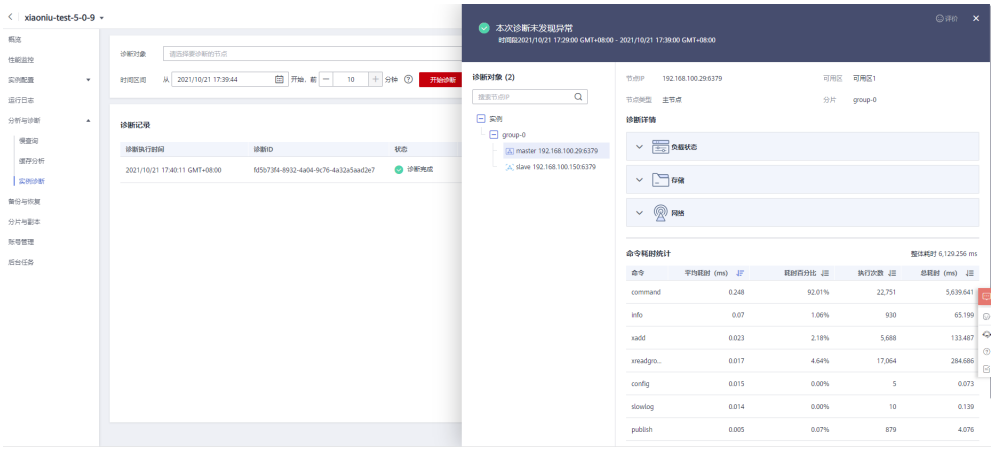
用率超高，容易触发主备倒换。关于各命令对应的时间复杂度信息请参见**Redis官网**。例如，使用了keys等消耗资源的命令，导致CPU超高，建议客户改成scan命令或者禁用keys命令。

步骤1 通过性能监控功能，确认CPU使用率高的具体时间段。



步骤2 通过下述方法，找出高消耗的命令。

- 慢查询功能会记录执行超过指定时间阈值的命令，通过分析慢查询的语句和执行时长可帮助您找出高消耗命令，具体操参见**查询Redis实例慢查询**。
- 通过实例诊断功能，选择CPU冲高的时间点进行诊断后，可以看到报告中的对应时间段命令的执行情况以及CPU耗时百分比，具体操作参见**实例诊断**。



步骤3 处理措施。

- 评估并禁用高风险命令和高消耗命令，例如**FLUSHALL**、**KEYS**、**HGETALL**等。
- 优化业务，例如避免频繁执行数据排序操作。
- **可选：**根据业务情况，选择下述方法对实例进行调整：

  扩容实例增强实例处理能力。

----结束

是否存在 Redis 的持久化重写操作

对于主备和集群实例，DCS Redis实例默认开启AOF数据落盘，实例开启了AOF持久化功能后，Redis会定期进行AofRewrite的磁盘整理，AOF磁盘持久化整理一般在以下2种场景执行：

- 数据量写入不大，AOF文件不大时，固定在每天的凌晨1-4点进行AOF持久化重写。所以容易出现这个时间点实例CPU使用率超高的现象。
- 数据量写入过大，AOF文件大小超过阈值（缓存实例容量的3-5倍）时，不论当前的所处的时间，会自动触发后台AOF持久化重写。

Redis的持久化重写操作（Bgsave或Bgrewriteaof）比较消耗CPU资源（请参考**为什么使用Fork执行Bgsave和Bgrewriteaof**），Bgsave和Bgrewriteaof会调用系统的Fork机制，造成CPU短暂时间冲高。

如果客户没有需要用到持久化功能，建议将该功能关闭（请根据实际业务慎重操作，关闭持久化功能会导致极端故障场景下恢复时，由于没有落盘造成的数据丢失）。关闭操作：在实例详情页面，选择“实例配置 > 参数配置”页签，将“appendonly”修改为“no”。

15.3 Redis 实例内存使用率高问题排查和解决

问题现象

Redis可提供高效的数据库服务，当内存不足时，可能导致Key频繁被逐出、响应时间上升、QPS（每秒访问次数）不稳定等问题，进而影响业务运行。由于Redis自身运行机制（主从同步、延迟释放等），内存占用率可能出现略微超过100%的情况，此为正常情况，此时内存已经写满，用户需要考虑扩容，或者清理一些无用的数据。通常情况下，当内存使用率超过95%时需要及时关注。

排查原因

1. 查询指定时段的内存使用率信息，具体操作请参见**查看监控指标**。“内存利用率”指标持续接近100%。
2. 查询内存使用率超过95%的时间段内，“已逐出的键数量”和“命令最大时延”，均呈现显著上升趋势，表明存在内存不足的问题。

   建议客户登录控制台，参考**缓存分析**和**查询Redis实例慢查询**，执行大Key扫描和慢查询。如果实例没有设置过期时间，会导致存储数据太多，内存被占满。
3. Redis实例如果内存满了但是key不多，可能原因是客户端缓冲区（output buffer）占用过多的内存空间。

   可以在Redis-cli客户端连接实例后，执行大key扫描命令：redis-cli --bigkeys，然后执行info，查看output buffer占用情况。

处理措施

- 查找大key、热key，方法如下：
  - DCS控制台提供了大Key和热Key的分析功能，您可参考**缓存分析**减少大key和热key。
  - 通过redis-cli的bigkeys和hotkeys参数查找大Key和热Key：
    - Redis-cli提供了bigkeys参数，能够使redis-cli以遍历的方式分析Redis实例中的所有Key，并返回Key的整体统计信息与每个数据类型中Top1的大Key，bigkeys仅能分析并输入六种数据类型（STRING、LIST、HASH、SET、ZSET、STREAM），命令示例为：**redis-cli -h <实例的连接地址> -p <端口> -a <密码> --bigkeys**。
    - 自Redis 4.0版本起，redis-cli提供了hotkeys参数，可以快速帮您找出业务中的热Key，该命令需要在业务实际运行期间执行，以统计运行期间的

热Key。命令示例为：**redis-cli -h <实例的连接地址> -p <端口> -a <密码> --hotkeys**。热Key的详情可以在结果中的summary部分获取到。

- 提前发现大key、热key。
  - 参考**配置告警**配置节点级别的**内存利用率**监控指标的告警。

    如果某个节点存在大key，这个节点比其他节点内存使用率高很多，会触发告警，便于您发现潜在的大key。
  - 参考**配置告警**配置节点级别的**入网最大带宽**、**出网最大带宽**、**CPU利用率**监控指标的告警。

    如果某个节点存在热key，这个节点的带宽占用、CPU利用率都比其他节点高，该节点会容易触发告警，便于您发现潜在热key。

    说明

    由于Redis 3.0实例不支持热key分析，您可以使用该方式帮助您发现热key。
- 其他优化建议：
  - **string类型控制在10KB以内**，hash、list、set、zset**元素尽量不超过5000**。
  - key的命名前缀为业务缩写，禁止包含特殊字符(比如空格、换行、单双引号以及其他转义字符)。
  - Redis事务功能较弱，不建议过多使用。
  - 短连接性能差，推荐使用带有连接池的客户端。
  - 如果只是用于数据缓存，容忍数据丢失，建议关闭持久化。
- 如果实例内存使用率通过以上方式仍然很高，请考虑在业务低峰期扩大实例规格。具体操作请参见**变更实例规格**。

15.4 排查 Redis 实例带宽使用率高的问题

概述

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。不同的实例规格对应的最大带宽有所不同，当超过该规格的最大带宽时，将对应用服务的数据访问性能造成影响。本节讲述如何排查Redis实例带宽使用率高的问题。

操作步骤

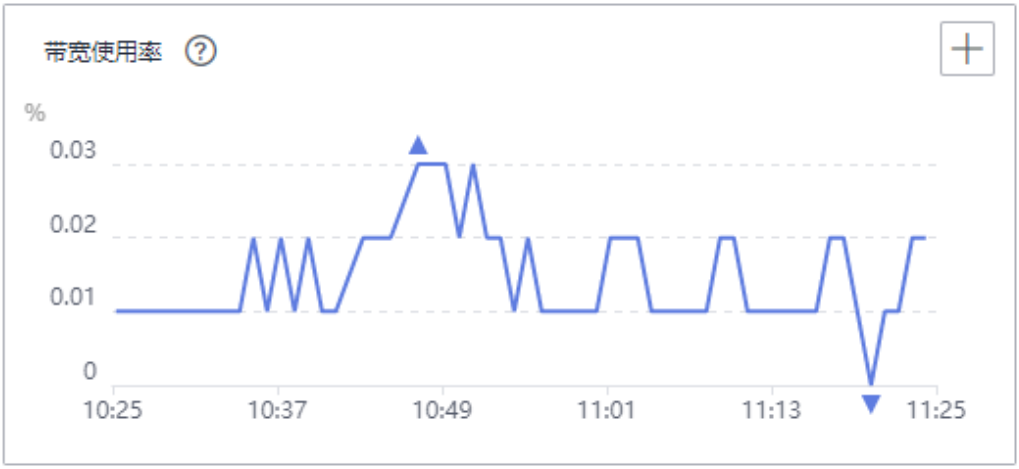
步骤1 查询带宽使用率。

查询实例在指定时段的带宽使用率。具体操作请参见**查看监控指标**。

通常来说，“网络瞬时输入流量”和“网络瞬时输出流量”快速上升，并持续大于实例最大带宽的80%时，需引起注意，可能流量不足。

需关注的监控指标为带宽使用率如下图。带宽使用率的计算公式：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%。

图 15-1 带宽使用率示例



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。

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间隔期间，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。

步骤2 优化带宽使用率。

1. 当业务的访问量与预期带宽消耗不匹配，例如带宽使用率的增长趋势和QPS的增长趋势明显不一致（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）。对于单个节点带宽使用率上涨，您可以通过缓存分析功能，发现实例中存在的大Key，具体操作请参见**缓存分析**。对大Key（通常大于10 KB）进行优化，例如将大Key拆分、减少对大Key的访问、删除不必要的大Key等。
2. 经过上述步骤优化后流量使用率依旧较高，可评估升级至更大内存的规格，以承载更大的网络流量。具体操作请参见**变更实例规格**。

   说明

   在正式升级实例的规格前，您可以先创建一个实例，测试要升级到的目标规格是否能够满足业务的负载需求，测试完成后可将其释放。释放实例请参考**删除实例**。

----结束

15.5 数据迁移失败问题排查

在使用控制台进行数据迁移时，如果出现迁移方案选择错误、在线迁移源Redis没有放通SYNC和PSYNC命令、源Redis和目标Redis网络不连通等问题，都会导致迁移失败。

本章节主要介绍使用DCS控制台进行数据迁移时迁移失败的问题排查和解决。

排查步骤

步骤1 查看迁移日志。

- 出现如下错误，表示迁移任务底层资源不足，需要联系技术支持处理。
  ```
  create migration ecs failed, flavor
  ```

- 出现如下错误，表示在线迁移时，源Redis没有放通SYNC和PSYNC命令，需要联系技术支持放通命令。
  source redis unsupported command: psync

步骤2 检查迁移方案。Redis实例间迁移，高版本不支持迁移到低版本。

步骤3 检查源Redis是否放通SYNC和PSYNC命令，迁移任务底层资源与源Redis、目标Redis网络是否连通。

如果是在线迁移，才涉及该操作。

在线迁移，必须满足源Redis和目标Redis的网络相通、源Redis已放通SYNC和PSYNC命令这两个前提，否则，会迁移失败。

- 网络

  检查DCS的源Redis、目标Redis、迁移任务所需虚拟机是否在同一个VPC，如果不在同一个VPC，则需要建立VPC对等连接，打通网络。关于创建和使用VPC对等连接，请参考《虚拟私有云用户指南》的“对等连接”文档说明。

  如果是同一个VPC，则检查安全组（Redis 3.0实例）或白名单（Redis 4.0/5.0/6.0实例）是否放通端口和IP，确保网络是连通的；

  源Redis和目标Redis必须允许迁移任务底层虚拟机访问。实例安全组或白名单配置，请参考**安全组配置和选择**、**管理实例白名单**。

  源Redis和目标Redis属于不同的云厂商时，需使用云专线服务打通网络。

- 命令

  默认情况下，一般云厂商都是禁用了SYNC和PSYNC命令，如果要放通，需要联系云厂商运维人员放通命令。

  - DCS内部进行迁移：
    - 自建Redis迁移至DCS，默认没有禁用SYNC和PSYNC命令；
    - DCS服务之间进行迁移，如果是同一账号相同Region进行在线迁移，在执行迁移时，会自动放通SYNC和PSYNC命令；
    - 如果是不同Region或相同Region不同账号进行在线迁移，不会自动放通SYNC和PSYNC命令，无法使用控制台上的在线迁移功能。推荐使用备份文件导入方式迁移。
  - 其他云厂商迁移到DCS服务：
    - 一般云厂商都是禁用了SYNC和PSYNC命令，如果使用在线迁移功能，需要联系源端的云厂商运维人员放通此命令。如果使用离线迁移，推荐使用备份文件导入的方式。
    - 如果不需要增量迁移，可以参考**使用Redis-shake工具在线全量迁移其他云厂商Redis**进行全量迁移，该方式不依赖于SYNC和PSYNC。

步骤4 检查源Redis是否存在大Key。

如果源Redis存在大key，建议将大key打散成多个小key后再迁移。

步骤5 检查目标Redis的规格是否大于迁移数据大小、是否有其他任务在执行。

如果目标Redis的实例规格小于迁移数据大小，迁移过程中，内存被占满，会导致迁移失败。

如果目标Redis存在正在执行的主备切换，建议联系技术支持关闭主备切换后，重新执行数据迁移。待迁移完成后，重新开启主备切换。

步骤6 检查迁移操作是否正确。

检查填写的IP地址、实例密码是否正确。

步骤7 排查白名单。

步骤8 如果无法解决，请联系技术支持。

----结束

A 文档修订记录

表 A-1 文档修订记录

发布日期	修订记录
2025-09-01	新增**调整DCS实例带宽**。
2025-07-28	更新**创建Redis实例**和**删除实例**，新增**恢复或销毁回收站内的DCS实例**。
2025-06-03	新增Redis 7.0实例，下线Redis 3.0和Memcached文档。 新增**实例关机**。
2025-04-28	• 新增**升级DCS实例小版本**、**升级Redis 3.0实例大版本**。 • 更新**查看和修改DCS实例信息**。
2024-11-26	**Redis 6.0/7.0实例**中增加Cluster集群实例。
2024-06-25	在**创建Redis实例**和**管理标签**章节增加标签策略说明。
2023-09-26	在**修改实例配置参数**中补充参数active-expire-num。
2023-04-03	第九次正式发布。 • 增加**约束与限制**、**DCS业务使用流程**、**交换DCS实例IP**、**DCS常用的监控指标**、**数据迁移指南**。 • 新增部分常见问题及其他文档优化。
2023-01-03	第八次正式发布。 • 更新**变更规格**，增加副本数变更。 • 新增**参数模板**。 • 新增Redis 6.0及相关内容。 • 新增**配置Redis SSL数据加密传输**。 • 更新**创建Redis实例**。

发布日期	修订记录
2022-10-19	第七次正式发布。 更新常见问题，增加以下章节： **连接池选择及Jedis连接池参数配置建议** **Redis key丢失是什么原因** **重启实例后缓存数据会保留吗?** **如何查看Redis实例的实时并发连接数和最大连接数** **监控指标中存在已拒绝的连接数是什么原因?** **触发限流（流控）的原因和处理建议** **大Key/热Key分析** 增加**故障排除**。
2022-04-12	第一次正式发布。