分布式缓存服务

用户指南

发布日期　2022-04-12

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）是一款内存数据库服务，兼容了Redis和Memcached两种内存数据库引擎，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、主备和集群三种类型的缓存实例，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。

  其中Redis4.0和Redis5.0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，每个分片是一个主节点和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3.0、4.0、5.0。
底层架构	基于虚拟机的**标准版**，单节点QPS达10万/秒。
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。

DCS Memcached

Memcached是一种内存Key-Value缓存系统，它支持简单字符串数据的存取，通常作为后端数据库内容缓存，以提升web的应用性能，降低对后端数据库的性能依赖，具体了解请参考**Memcached典型应用场景**。

DCS全面兼容Memcached协议并增强实现了双机热备和数据持久化。

表 1-2 DCS Memcached 灵活的实例配置

实例类型	提供单机、主备两种类型，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，实例重启后原有缓存数据不会加载。 主备：包含一个主节点和一个备节点，主备节点的数据通过实时复制保持一致，备节点对用户不可见且不能直接读写数据，当主节点故障后，备节点自动升级为主节点。
内存规格	单机和主备实例均提供2GB、4GB、8GB、16GB、32GB、64GB共6种内存规格。

高可用与容灾	主备实例提供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Memcached技术细节，您可以访问Memcached官方网站**https://memcached.org/**

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。

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Redis5.0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。

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，往往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。

Memcached 典型应用场景

Memcached主要存储字符串类的简单key-value数据。

1. 静态页面缓存。

Web页面的内容片段，包括HTML，CSS和图片等静态数据，内容修改操作少，读取频繁，可以缓存到DCS Memcached实例，提高网站的访问性能。

2. 数据库前端缓存。

   在动态系统中存在对大量数据读多写少的场景，如社交、博客网站大量查询用户信息、好友信息、文章信息等。为了减少磁盘数据库负载，提升性能，这些经常需要从数据库读取的数据，可以缓存在Memcached中。

   适宜缓存的数据主要有：

   - 经常被读取，实时性要求不高，且可以自动过期的数据。

     例如网站首页最新文章列表、某排行等数据，虽然新数据不断产生，但对用户体验影响比较小。这类数据可使用典型的缓存策略，设置合理的过期时间，当数据过期以后再从数据库中读取。为了让编辑或者其它人员能马上看到效果，可以再定一个缓存清除/刷新策略。

   - 经常被读取并且实时性要求强的数据。

     比如用户的好友列表，用户文章列表，用户阅读记录等。这类数据首先被载入到Memcached中，当发生更改（添加、修改、删除）时就刷新缓存数据。

3. 秒杀功能。

   商品下单操作，牵涉数据库读取，写入订单，更改库存，及事务一致性要求， 对于使用传统型数据库的平台来说，秒杀活动阶段要避免订单创建后库存缺货，同时提供流畅的用户体验，压力巨大。

   可以利用Memcached的incr/decr功能， 在内存中存储商品的库存量， 秒杀的抢单过程主要在内存中完成，速度非常快，抢单成功即得一个订单号，这时再去支付页面完成订单的后续操作。

说明

不适用Memcached的应用场景：

- 单个缓存对象大于1M

  Memcached单个缓存对象的value值不能超过1M。超过1M的场景，建议使用Redis。

- Key的长度大于250字符

  如确需使用Memcached，可将key先进行md5，得到散列值，然后存储key对应的散列值。

- 业务需要保证数据高可靠

  开源Memcached不支持数据持久化，无法存副本、备份以及数据迁移。

  注意：DCS的Memcached主备版本实现了数据持久化，具体可联系技术支持。

- 对数据结构和处理有高级要求

  Memcached只支持key-value简单结构，不支持链表、集合等高级数据结构以及排序等一系列复杂操作。

1.3 实例类型

1.3.1 Redis 单机实例

DCS Redis单机实例有三个版本选择，Redis3.0、Redis4.0和Redis5.0。

单机实例特点

1. 系统资源消耗低，支持高QPS

单机实例不涉及数据同步、数据持久化所需消耗的系统开销，因此能够支撑更高的并发。Redis单机实例QPS达到10万以上。

2. 进程监控，故障后自动恢复

   DCS部署了业务高可用探测，单机实例故障后，30秒内会重启一个新的进程，恢复业务。

3. 即开即用，数据不做持久化

   单机实例开启后不涉及数据加载，即开即用。如果服务QPS较高，可以考虑进行数据预热，避免给后端数据库产生较大的并发冲击。

4. 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。

总体说来，单机实例支持读写高并发，但不做持久化，实例重启时不保存原有数据。单机实例主要服务于数据不需要由缓存实例做持久化的业务场景，如数据库前端缓存，用以提升数据读取效率，减轻后端并发压力。当缓存中查询不到数据，可穿透至磁盘数据库中获取，同时，重启服务/缓存实例时，可从磁盘数据库中获取数据进行预热，降低后端服务在启动初期的压力。

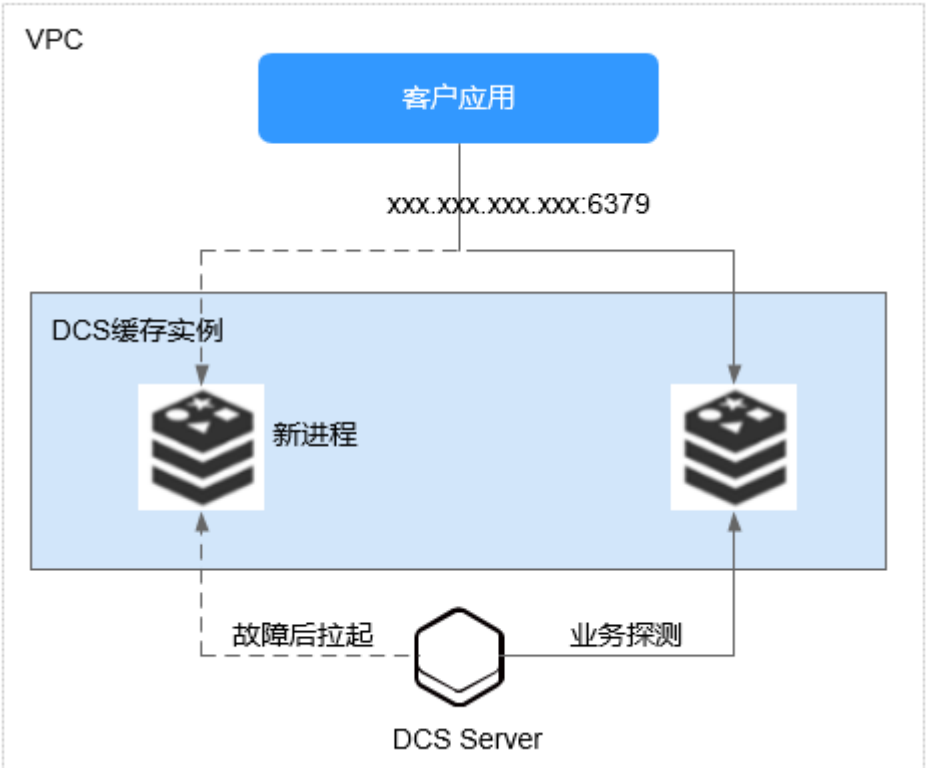
实例架构设计

DCS的Redis单机实例架构，如图1-1所示。

说明

Redis3.0不支持定义端口，端口固定为6379，Redis4.0和Redis5.0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，并且配置安全组访问规则。

  相关参考：**安全组配置和选择**。

- **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。

  DCS实时探测实例可用性，当Redis进程故障后，DCS为实例重新拉起一个新的Redis进程，恢复业务。

1.3.2 Redis 主备实例

DCS Redis、Memcached两种缓存类型都支持主备实例，本章节主要介绍Redis缓存类型的主备实例，有三个版本选择，Redis3.0、Redis4.0和Redis5.0。

说明

不支持Redis版本的升级，例如，不支持Redis 3.0主备升级为Redis 4.0/5.0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，备节点在30秒内自动完成主备切换，无需用户操作，业务平稳运行。

4. **容灾策略**

   跨AZ部署（可用区）：DCS支持将主备实例部署在不同的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。

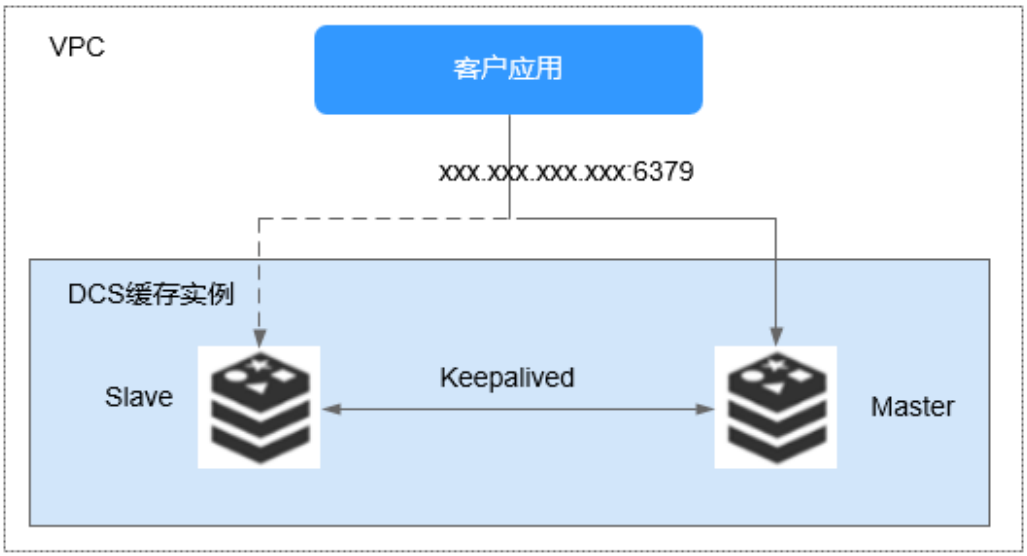
实例架构设计

DCS的Redis主备实例架构，如图1-2所示。

说明

Redis3.0不支持定义端口，端口固定为6379，Redis4.0和Redis5.0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 1-2 主备实例示意图



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，客户端需要与主备实例处于相同VPC，并且配置安全组访问规则。

  相关参考：**安全组配置和选择**。

- **客户应用**

  运行在ECS上的客户应用程序，即Redis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主备实例包含了Master和Slave两个节点。默认开启数据持久化功能，同时保持节点间数据同步。

  DCS实时探测实例可用性，当主节点故障后，备节点升级为主节点，恢复业务。

  Redis的访问端口默认为6379。

1.3.3 Redis Proxy 集群实例

DCS Redis的集群实例有两种版本可供选择，一种是基于LVS+Proxy的高可用集群版本（以下简称为Proxy版Redis集群），另一种是原生Cluster的集群版本。Proxy版集群兼容开源Redis 3.0，Cluster版本兼容开源Redis的4.0和5.0。

本章节主要介绍Redis3.0Proxy集群实例。

说明

- 在连接Proxy集群实例时，客户端不需要做特殊配置，使用方式与单机、主备实例相同，使用实例IP地址或域名连接即可，不需要知晓和使用Proxy节点或分片地址。

Redis3.0 Proxy 集群实例

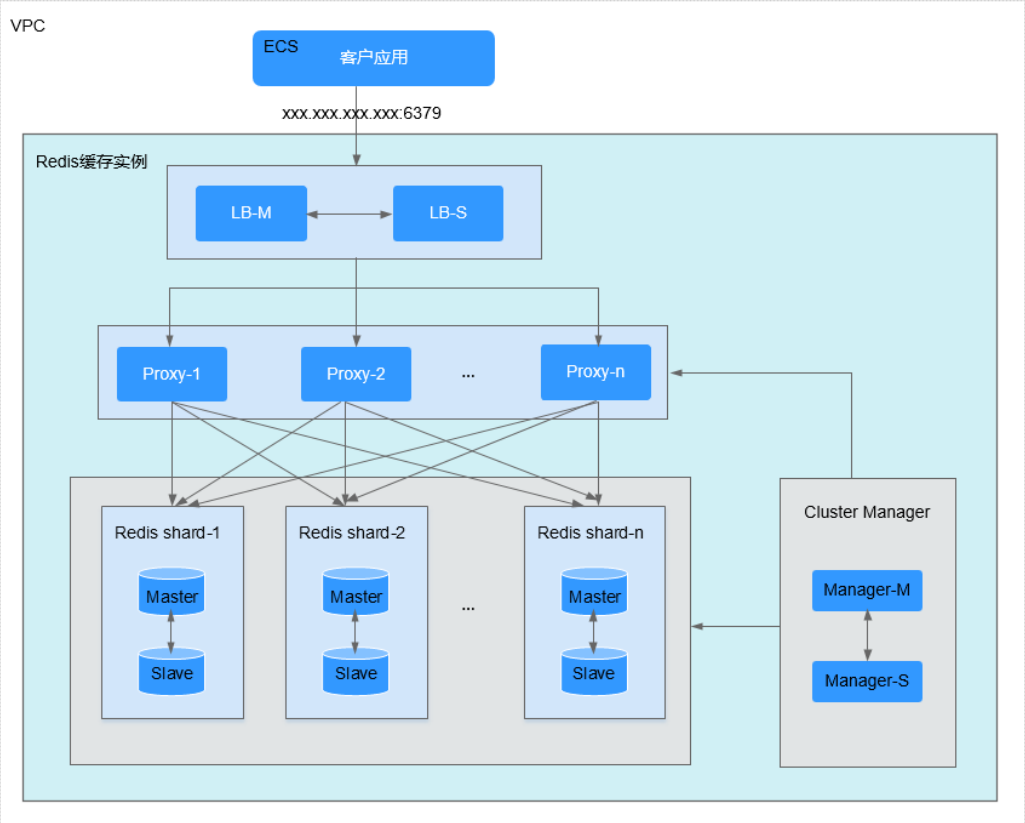
DCS Redis3.0 Proxy集群实例基于开源Redis 3.0版本构建，兼容**开源codis**，提供64G~1024G多种大容量规格版本，用于满足**百万级以上并发**与**大容量数据缓存**的需要。Redis集群的数据分布式存储和读取，由DCS内部实现，用户无需投入开发与运维成本。

Redis集群实例由“负载均衡器”、“Proxy服务器”、“集群配置管理器”、“**集群分片**”共4个部分组成。

表 1-3 Redis3.0 集群实例规格和 Proxy 数、分片数的对应关系

集群版规格	Proxy节点数	分片数（Shard）
64GB	3	8
128GB	6	16
256GB	8	32

图 1-3 Redis Proxy 集群实例示意图



示意图说明：

- **VPC**

  虚拟私有云。集群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Proxy集群实例处于相同VPC，并且配置安全组访问规则。

  相关参考：**安全组配置和选择**。

- 客户应用程序

  客户应用程序，即Redis集群客户端。

  Redis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**。

- **LB-M/LB-S**

  负载均衡服务器，采用主备高可用方式。Redis集群实例提供访问的IP地址，即为负载均衡服务器地址。

- **Proxy**

  Redis集群代理服务器。用于实现Redis集群内部的高可用，以及承接客户端的高并发请求。

  支持使用Proxy节点的IP连接集群实例。

- **Redis shard**

  Redis集群的分片。

  每个分片也是一个Redis主备实例，分片上的主实例故障时，系统会自动进行主备切换，集群正常提供服务。

某个分片的主备实例都故障，集群可正常提供服务，但该分片上的数据不能读取。

- **Cluster manager**

  集群配置管理器，用于存储集群的配置信息与分区策略。用户不能修改配置管理器的信息。

1.3.4 Redis Cluster 集群实例

DCS Redis的集群实例有两种版本可供选择，一种是基于LVS+Proxy的高可用集群版本（以下简称为Proxy版Redis集群），另一种是原生Cluster的集群版本。Proxy版集群兼容开源Redis 3.0，Cluster版本兼容开源Redis的4.0和5.0。

本章节主要介绍Redis4.0和Redis5.0的Cluster集群实例。

Redis4.0 和 Redis5.0 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1-4**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。

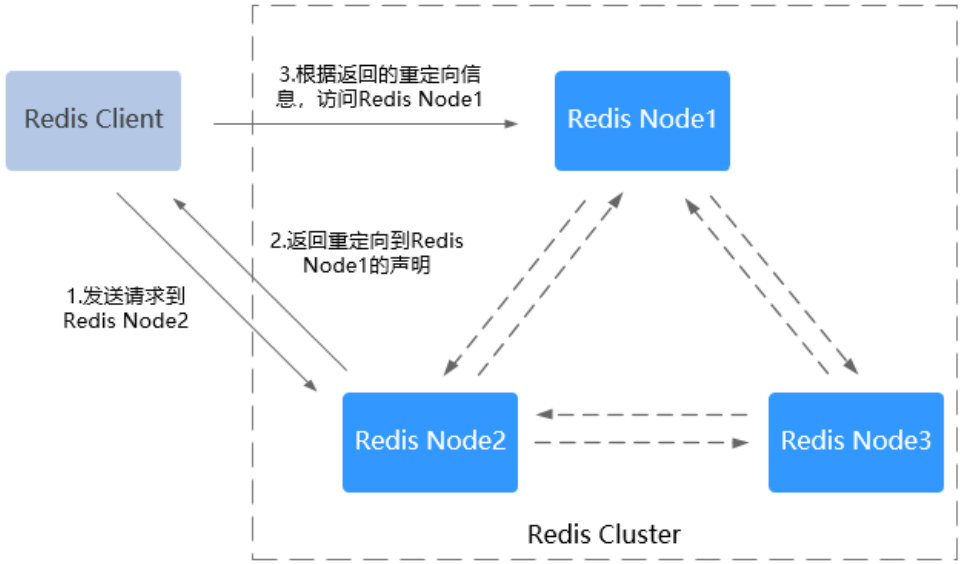
表 1-4 Cluster 集群实例规格和分片数的对应关系

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
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。

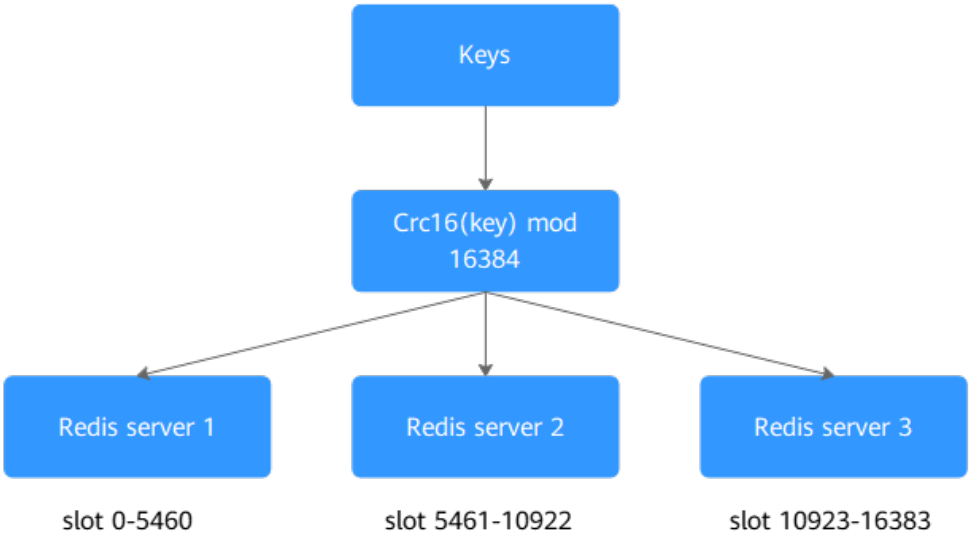
图 1-4 Redis Cluster 无中心架构



- 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 1-5 Redis Cluster 预分片示意图



1.3.5 Memcached 单机实例

本章节主要介绍Memcached单机实例的特点和架构。

单机实例特点

1. 系统资源消耗低，支持高QPS

   单机实例不涉及数据同步、数据持久化所需消耗的系统开销，因此能够支撑更高的并发。Memcached的单机实例QPS达到10万以上。

2. 进程监控，故障后自动恢复

   DCS部署了业务高可用探测，单机实例故障后，30秒内会重启一个新的进程，恢复业务。

3. 即开即用，数据不做持久化

   单机实例开启后不涉及数据加载，即开即用。如果服务QPS较高，可以考虑进行数据预热，避免给后端数据库产生较大的并发冲击。

4. 低成本，适用于开发测试

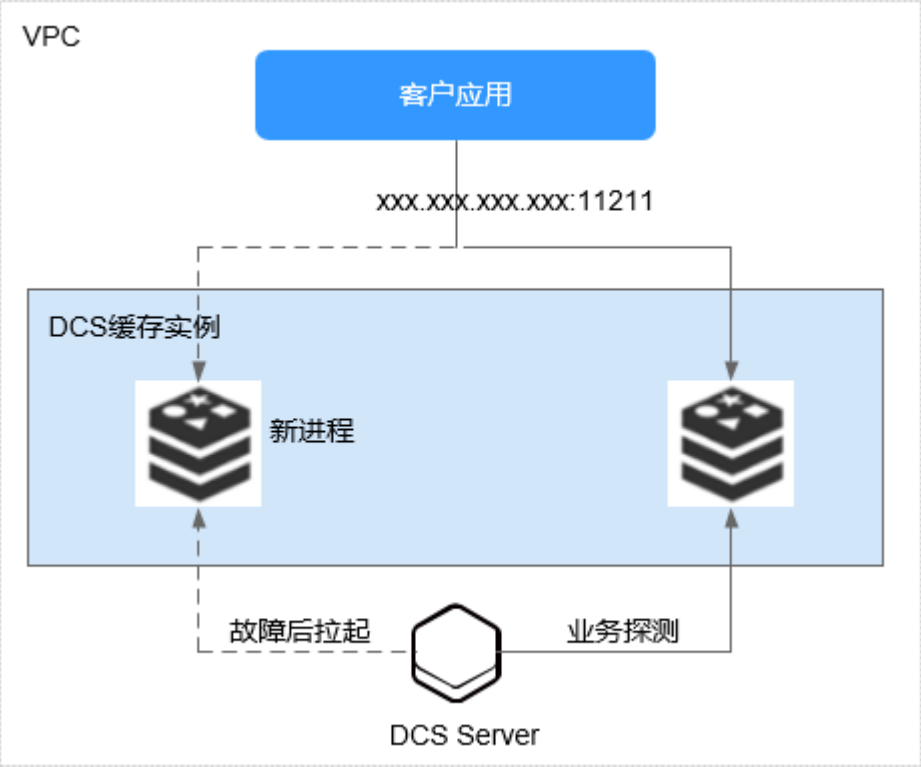
   单机实例各种规格的成本相对主备减少40%以上。适用于开发、测试环境搭建。

总体说来，单机实例支持读写高并发，但不做持久化，实例重启时不保存原有数据。单机实例主要服务于数据不需要由缓存实例做持久化的业务场景，如数据库前端缓存，用以提升数据读取效率，减轻后端并发压力。当缓存中查询不到数据，可穿透至磁盘数据库中获取，同时，重启服务/缓存实例时，可从磁盘数据库中获取数据进行预热，降低后端服务在启动初期的压力。

实例架构设计

DCS的Memcached单机实例架构，如图1-6所示。

图 1-6 Memcached 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，并且配置安全组访问规则。

相关参考：**安全组配置和选择**。

- **客户应用**

  运行在ECS上的客户应用程序，即实例的客户端。

  Memcached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Memcached实例**。

- **DCS缓存实例**

  DCS单机实例只有1个节点，1个Memcached进程。

  DCS实时探测实例可用性，当Memcached进程故障后，DCS为实例重新拉起一个新的Memcached进程，恢复业务。

  Memcached实例访问端口为11211。

1.3.6 Memcached 主备实例

本章节主要描述Memcached主备实例。

主备实例特点

Memcached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。

Memcached主备实例具有以下特性：

1. **持久化，确保数据高可靠**

   实例包含一个主节点和一个备节点，都默认开启数据持久化。同时支持数据持久化，确保数据高可靠。

   Memcached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，备节点在30秒内自动完成主备切换，无需用户操作，业务平稳运行。

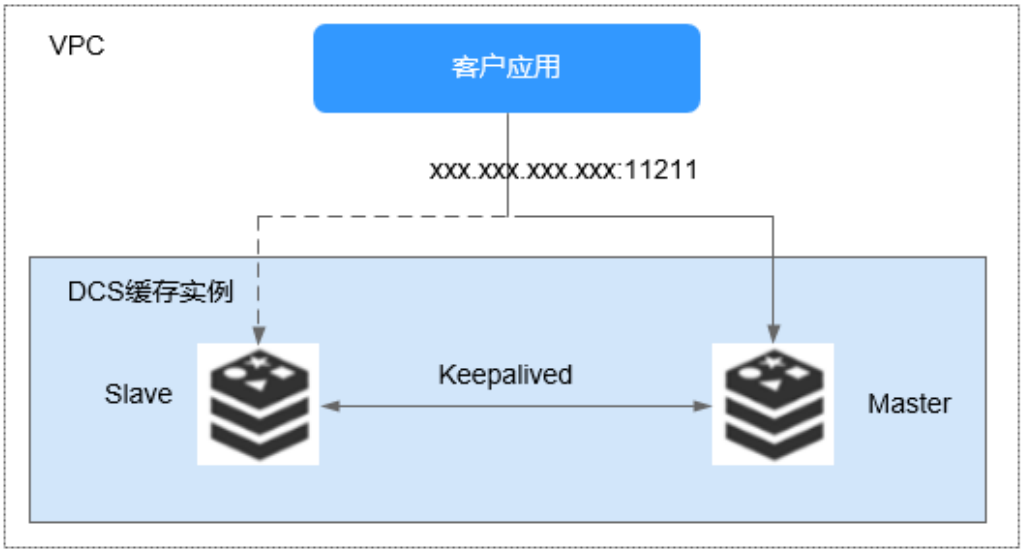
4. **多种容灾策略**

   跨AZ部署（可用区）：DCS支持将主备实例部署在不同的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。

Memcached 主备实例架构设计

DCS的Memcached主备实例架构，如**图1-7**所示。

图 1-7 Memcached 主备实例示意图



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，并且配置安全组访问规则。

  相关参考：**安全组配置和选择**。

- **客户应用**

  运行在ECS上的客户应用程序，即Memcached的客户端。

  Memcached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Memcached实例**。

- **DCS缓存实例**

  DCS实例。主备实例包含了Master和Slave两个节点。默认开启数据持久化功能，同时保持节点间数据同步。

  DCS实时探测实例可用性，当主节点故障后，备节点升级为主节点，恢复业务。

  Memcached实例访问端口为11211。

1.4 实例规格

1.4.1 Redis3.0 实例

本节介绍DCS Redis3.0实例的产品规格，包括内存规格、实例可使用内存、连接数上限、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。

- 连接数上限：表示允许客户端同时连接的个数，即连接并发数。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。

说明

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

单机实例

因系统开销占用一部分资源，Redis单机实例可用内存比实例规格略小，如下表所示。

表 1-5 Redis 3.0 单机实例产品规格

内存规格（GB）	实例可使用内存（GB）	连接数上限（默认/可配）（个）	基准/最大带宽（Mbit/s）	参考性能（QPS）
2	1.5	5,000/50,000	42/512	50,000
4	3.2	5,000/50,000	64/1,536	100,000
8	6.8	5,000/50,000	64/1,536	100,000
16	13.6	5,000/50,000	85/3,072	100,000
32	27.2	5,000/50,000	85/3,072	100,000
64	58.2	5,000/60,000	128/5,120	100,000

主备实例

对于Redis主备实例，需要预留持久化的内存，部分规格的实际可使用与单机实例相比略少，如下表所示。主备实例可以调整实例可用内存，以更好地支持数据持久化、主从同步等后台任务。

表 1-6 Redis 3.0 主备实例产品规格

内存规格（GB）	实例可使用内存（GB）	连接数上限（默认/可配）（个）	基准/最大带宽（Mbit/s）	参考性能（QPS）
2	1.5	5,000/50,000	42/512	50,000
4	3.2	5,000/50,000	64/1,536	100,000
8	6.4	5,000/50,000	64/1,536	100,000
16	12.8	5,000/50,000	85/3,072	100,000
32	25.6	5,000/50,000	85/3,072	100,000

内存规格（GB）	实例可使用内存（GB）	连接数上限（默认/可配）（个）	基准/最大带宽（Mbit/s）	参考性能（QPS）
64	51.2	5,000/60,000	128/5,120	100,000

Proxy 集群实例

Redis Proxy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，客户端连接数、内网带宽上限、QPS指标都有很大的提升。

表 1-7 Redis 3.0 Proxy 集群实例产品规格

规格（GB）	实例可使用内存（GB）	连接数上限（默认/可配）（个）	基准/最大带宽（Mbit/s）	参考性能（QPS）
64	64	90,000/90,000	600/5,120	500,000
128	128	180,000/180,000	600/5,120	500,000
256	256	240,000/240,000	600/5,120	500,000

1.4.2 Redis4.0/5.0 实例

本节介绍DCS Redis4.0和Redis5.0实例的产品规格，包括内存规格、实例可使用内存、连接数上限、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。
- 连接数上限：表示允许客户端同时连接的个数，即连接并发数。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 1-8 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）	连接数上限（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）
1	1	10,000/50,000	80/80	80,000	x86： redis.single.xu1.large.1 Arm： redis.single.au1.large.1
2	2	10,000/50,000	128/128	80,000	x86： redis.single.xu1.large.2 Arm： redis.single.au1.large.2
4	4	10,000/50,000	192/192	80,000	x86: redis.single.xu1.large.4 Arm： redis.single.au1.large.4
8	8	10,000/50,000	192/192	100,000	x86: redis.single.xu1.large.8 Arm： redis.single.au1.large.8
16	16	10,000/50,000	256/256	100,000	x86: redis.single.xu1.large.16 Arm: redis.single.au1.large.16

内存规格（GB）	实例可使用内存（GB）	连接数上限（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）
24	24	10,000/50,000	256/256	100,000	x86: redis.single.xu1.large.24 Arm: redis.single.au1.large.24
32	32	10,000/50,000	256/256	100,000	x86: redis.single.xu1.large.32 Arm: redis.single.au1.large.32
48	48	10,000/50,000	256/256	100,000	x86: redis.single.xu1.large.48 Arm: redis.single.au1.large.48
64	64	10,000/50,000	384/384	100,000	x86: redis.single.xu1.large.64 Arm: redis.single.au1.large.64

主备实例

下表中仅列出了x86和Arm的默认副本数为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主备实例，X86主备2副本的名称为redis.ha.xu1.large.**r2**.8，3副本为redis.ha.xu1.large.**r3**.8，以此类推。

表 1-9 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）	连接数上限（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）
1	1	10,000/50,000	80/80	80,000	x86: redis.ha.xu1.large.r2.1 Arm: redis.ha.au1.large.r2.1
2	2	10,000/50,000	128/128	80,000	x86: redis.ha.xu1.large.r2.2 Arm: redis.ha.au1.large.r2.2
4	4	10,000/50,000	192/192	80,000	x86: redis.ha.xu1.large.r2.4 Arm: redis.ha.au1.large.r2.4
8	8	10,000/50,000	192/192	100,000	x86: redis.ha.xu1.large.r2.8 Arm: redis.ha.au1.large.r2.8
16	16	10,000/50,000	256/256	100,000	x86: redis.ha.xu1.large.r2.16 Arm: redis.ha.au1.large.r2.16
24	24	10,000/50,000	256/256	100,000	x86: redis.ha.xu1.large.r2.24 Arm: redis.ha.au1.large.r2.24

内存规格（GB）	实例可使用内存（GB）	连接数上限（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）
32	32	10,000/50,000	256/256	100,000	x86: redis.ha.xu1.large.r2.32 Arm: redis.ha.au1.large.r2.32
48	48	10,000/50,000	256/256	100,000	x86: redis.ha.xu1.large.r2.48 Arm: redis.ha.au1.large.r2.48
64	64	10,000/50,000	384/384	100,000	x86: redis.ha.xu1.large.r2.64 Arm: redis.ha.au1.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。

下表中仅列出了x86和Arm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，8G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。

表 1-10 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)	实例连接数上限（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	x86: redis.cluster.xu1.large.r2.4 Arm: redis.cluster.au1.large.r2.4
8	8	3	30,000 /150,000	2,304/2,304	240,000	x86: redis.cluster.xu1.large.r2.8 Arm: redis.cluster.au1.large.r2.8
16	16	3	30,000 /150,000	2,304/2,304	240,000	x86: redis.cluster.xu1.large.r2.16 Arm: redis.cluster.au1.large.r2.16
32	32	3	30,000 /150,000	2,304/2,304	300,000	x86: redis.cluster.xu1.large.r2.32 Arm: redis.cluster.au1.large.r2.32

规格（GB）	实例可使用内存（GB）	分片数(主节点个数)	实例连接数上限（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）
64	64	8	80,000 /400,000	6,144/6,144	500,000	x86: redis.cluster.xu1.large.r2.64 Arm: redis.cluster.au1.large.r2.64
128	128	16	160,000 /800,000	12,288/12,288	1,000,000	x86: redis.cluster.xu1.large.r2.128 Arm: redis.cluster.au1.large.r2.128
256	256	32	320,000 /1,600,000	24,576/24,576	>2,000,000	x86: redis.cluster.xu1.large.r2.256 Arm: redis.cluster.au1.large.r2.256
512	512	64	640,000 /3,200,000	49,152/49,152	>2,000,000	x86: redis.cluster.xu1.large.r2.512 Arm: redis.cluster.au1.large.r2.512

规格（GB）	实例可使用内存（GB）	分片数(主节点个数)	实例连接数上限（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）
1024	1024	128	1,280,000/6,400,000	98,304/98,304	>2,000,000	x86: redis.cluster.xu1.large.r2.1024 Arm: redis.cluster.au1.large.r2.1024

1.4.3 Memcached 实例

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

连接数上限：表示允许客户端同时连接的个数，即连接并发数。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。

说明

Memcached实例支持“单机”和“主备”两种类型。

单机实例

因系统开销占用一部分资源，Memcached单机实例可用内存比实例规格略小，如表1-11所示。

表 1-11 Memcached 单机实例产品规格

内存规格（GB）	实例可使用内存（GB）	连接数上限（默认/可配）（个）	基准/最大带宽（Mbit/s）	参考性能（QPS）
2	1.5	5,000/50,000	42/128	50,000
4	3.2	5,000/50,000	64/192	100,000
8	6.8	5,000/50,000	64/192	100,000
16	13.6	5,000/50,000	85/256	100,000
32	27.2	5,000/50,000	85/256	100,000
64	58.2	5,000/50,000	128/384	100,000

主备实例

对于Memcached主备实例，需要预留持久化的内存，其可用内存如表1-12所示。主备实例可以调整实例可用内存，以更好地支持数据持久化、主从同步等后台任务。

表 1-12 Memcached 主备实例产品规格

内存规格（GB）	实例可使用内存（GB）	连接数上限（默认/可配）（个）	基准/最大带宽（Mbit/s）	参考性能（QPS）
2	1.5	5,000/50,000	42/128	50,000
4	3.2	5,000/50,000	64/192	100,000
8	6.8	5,000/50,000	64/192	100,000
16	13.6	5,000/50,000	85/256	100,000
32	27.2	5,000/50,000	85/256	100,000
64	58.2	5,000/50,000	128/384	100,000

1.5 开源命令兼容性

1.5.1 Redis3.0 命令

DCS Redis3.0基于开源3.0.7版本进行开发，兼容开源的协议和命令。

本章节主要介绍DCS Redis3.0命令的兼容性，包括支持命令列表，禁用命令列表，以及不支持的高版本Redis脚本和命令列表，以及命令使用限制说明。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**Redis3.0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**Redis3.0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**集群实例受限使用命令**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**。

Redis3.0 支持的命令

以下列出了Redis3.0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。
- 如果是Proxy集群实例，不支持表格中以下命令：
  - “List”类型中的BLPOP、BRPOP、BRPOPLRUSH命令。
  - “Server”类型的CLIENT相关命令，包括CLIENT KILL、CLIENT GETNAME、CLIENT LIST、CLIENT SETNAME、CLIENT PAUSE、CLIENT REPLY。
  - “Server”类型的MONITOR命令。
  - 如果是比较旧的Proxy集群实例，不支持“Key”类型中的RANDOMKE命令。

表 1-13 Redis3.0 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

Keys	String	Hash	List	Set	Sorted Set	Server
TYPE	MSET	-	RPOPLPU	SUNIONSTORE	ZUNIONSTORE	ROLE
SCAN	MSETNX	-	RPOPLPUSH	SSCAN	ZINTERSTORE	-
OBJECT	PSETEX	-	RPUSH	-	ZSCAN	-
-	SET	-	RPUSHX	-	ZRANGEBYLEX	-
-	SETBIT	-	-	-	-	-
-	SETEX	-	-	-	-	-
-	SETNX	-	-	-	-	-
-	SETRANGE	-	-	-	-	-
-	STRLEN	-	-	-	-	-

表 1-14 Redis3.0 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	SCRIPT KILL	GEORADIUS
-	UNSUBSCRIBE	-	-	SCRIPT LOAD	GEORADIUSBYMEMBER

Redis3.0 禁用的命令

以下列出了Redis3.0实例禁用的命令。

表 1-15 Redis3.0 单机和主备实例禁用命令

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

表 1-16 Redis3.0 Proxy 集群实例禁用命令

Keys	Server	List	Transactions	Connection	Cluster	codis相关
MIGRATE	SLAVEOF	BLPOP	DISCARD	SELECT	CLUSTER	TIME
MOVE	SHUTDOWN	BRPOP	EXEC	-	-	SLOTSINFO
-	LASTSAVE	BRPOPLPUSH	MULTI	-	-	SLOTSDEL
-	DEBUG相关类	-	UNWATCH	-	-	SLOTSMGRTSLOT
-	COMMAND	-	WATCH	-	-	SLOTSMGRTONE
-	SAVE	-	-	-	-	SLOTSCHECK
-	BGSAVE	-	-	-	-	SLOTSMGRTTAGSLOT
-	BGREWRITEAOF	-	-	-	-	SLOTSMGRTTAGONE
-	SYNC	-	-	-	-	-
-	PSYNC	-	-	-	-	-
-	MONITOR	-	-	-	-	-
-	CLIENT相关类	-	-	-	-	-

Keys	Server	List	Transactions	Connection	Cluster	codis相关
-	OBJECT	-	-	-	-	-
-	ROLE	-	-	-	-	-

1.5.2 Redis4.0 命令

DCS Redis4.0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 Redis4.0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**Redis4.0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**Redis4.0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**集群实例受限使用命令**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**。

Redis4.0 支持的命令

表1-17和**表1-18**列举了Redis4.0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 1-17 Redis4.0 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

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

表 1-18 Redis4.0 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	SCRIPT KILL	GEORADIUS
-	UNSUBSCRIBE	-	-	SCRIPT LOAD	GEORADIUSBYMEMBER

Redis4.0 禁用的命令

以下列出了Redis4.0实例禁用的命令。

表 1-19 Redis4.0 单机和主备禁用命令

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

表 1-20 Redis4.0 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
-	COMMAND	CLUSTER SETSLOT
-	SAVE	CLUSTER BUMPEPOCH
-	BGSAVE	CLUSTER SAVECONFIG
-	BGREWRITEAOF	CLUSTER FORGET
-	SYNC	CLUSTER REPLICATE
-	PSYNC	CLUSTER COUNT-FAILURE-REPORTS
-	-	CLUSTER FAILOVER
-	-	CLUSTER SET-CONFIG-EPOCH
-	-	CLUSTER RESET

1.5.3 Redis5.0 命令

DCS Redis5.0基于开源5.0.9版本进行开发，兼容开源的协议和命令。

本章节主要介绍DCS Redis5.0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**Redis5.0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**集群实例受限使用命令**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**。

Redis5.0 支持的命令

- **表1-21**和**表1-22**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 1-21 Redis5.0 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
SCAN	MSETNX	HLEN	RPOPLPUSH	SSCAN	ZINTERSTORE	SWAPDB
OBJECT	PSETEX	-	RPUSH	SPOP	ZSCAN	MEMORY

Keys	String	Hash	List	Set	Sorted Set	Server
PEXPIREAT	SET	-	RPUSHX	-	ZRANGEBYLEX	CONFIG
PEXPIRE	SETBIT	-	LPUSH	-	ZLEXCOUNT	-
-	SETEX	-	-	-	ZPOPMIN	-
-	SETNX	-	-	-	ZPOPMAX	-
-	SETRANGE	-	-	-	ZREMRANGEBYSCORE	-
-	STRLEN	-	-	-	ZREM	-
-	BITFIELD	-	-	-	-	-

表 1-22 Redis5.0 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
-	SUBSCRIBE	WATCH	SELECT	SCRIPT KILL	GEORADIUS	XGROUP
-	UNSUBSCRIBE	-	-	SCRIPT LOAD	GEORADIUSBYMEMBER	XINFO
-	-	-	-	-	-	XLEN
-	-	-	-	-	-	XPENDING
-	-	-	-	-	-	XRANGE
-	-	-	-	-	-	XREAD
-	-	-	-	-	-	XREADGROUP
-	-	-	-	-	-	XREVRANGE

HyperLoglog	Pub/Sub	Transactions	Connection	Scripting	Geo	Stream
-	-	-	-	-	-	XTRIM

Redis5.0 禁用的命令

以下列出了Redis5.0实例禁用的命令。

表 1-23 Redis 5.0 单机和主备禁用命令

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

表 1-24 Redis5.0 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
-	COMMAND	CLUSTER SETSLOT
-	SAVE	CLUSTER BUMPEPOCH
-	BGSAVE	CLUSTER SAVECONFIG
-	BGREWRITEAOF	CLUSTER FORGET
-	SYNC	CLUSTER REPLICATE

Keys	Server	Cluster
-	PSYNC	CLUSTER COUNT-FAILURE-REPORTS
-	-	CLUSTER FAILOVER
-	-	CLUSTER SET-CONFIG-EPOCH
-	-	CLUSTER RESET

1.5.4 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**（网站为：**http://www.redis.cn/commands.html**）查看。

当前仅Redis 4.0、Redis 5.0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 1-25 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
RANDOMKEY	GETRANGE	LPUSHX	SMEMBERS	ZREMRANGEBYRANK	CLIENT GETNAME

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

表 1-26 Web CLI 支持命令清单 2

Hash	HyperLoglog	Connection	Scripting	Geo
HDEL	PFADD	AUTH	EVAL	GEOADD
HEXISTS	PFCOUNT	ECHO	EVALSHA	GEOHASH
HGET	PFMERGE	PING	SCRIPT EXISTS	GEOPOS
HGETALL	-	QUIT	SCRIPT FLUSH	GEODIST
HINCRBY	-	-	SCRIPT KILL	GEORADIUS
HINCRBYFLOAT	-	-	SCRIPT LOAD	GEORADIUSBYMEMBER

Hash	HyperLoglog	Connection	Scripting	Geo
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 1-27 通过 Web CLI 连接单机和主备实例时禁用的命令清单 1

Keys	Server	Transactions	Pub/Sub
MIGRATE	SLAVEOF	UNWATCH	PSUBSCRIBE
WAIT	SHUTDOWN	REPLICAOF	PUBLISH
DUMP	DEBUG相关类	DISCARD	PUBSUB
RESTORE	CONFIG SET	EXEC	PUNSUBSCRIBE
-	CONFIG REWRITE	MULTI	SUBSCRIBE
-	CONFIG RESETSTAT	WATCH	UNSUBSCRIBE
-	SAVE	-	-
-	BGSAVE	-	-
-	BGREWRITEAOF	-	-
-	COMMAND	-	-
-	KEYS	-	-
-	MONITOR	-	-
-	SYNC	-	-
-	PSYNC	-	-
-	ACL	-	-

表 1-28 通过 Web CLI 连接单机和主备实例时禁用的命令清单 2

List	Connection	Sorted Set
BLPOP	SELECT	BZPOPMAX
BRPOP	-	BZPOPMIN
BLMOVE	-	BZMPOP
BRPOPLPUSH	-	-
BLMPOP	-	-

表 1-29 通过 Web CLI 连接 Cluster 集群时禁用的命令 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
-	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	-	CLUSTER FAILOVER
-	MONITOR	-	CLUSTER SET-CONFIG-EPOCH
-	SYNC	-	CLUSTER RESET
-	PSYNC	-	-
-	ACL	-	-

表 1-30 通过 Web CLI 连接 Cluster 集群时禁用的命令 2

Pub/Sub	List	Connection	Sorted Set
PSUBSCRIBE	BLPOP	SELECT	BZPOPMAX

Pub/Sub	List	Connection	Sorted Set
PUBLISH	BRPOP	-	BZPOPMIN
PUBSUB	BLMOVE	-	BZMPOP
PUNSUBSCRIBE	BRPOPLPUSH	-	-
SUBSCRIBE	BLMPOP	-	-
UNSUBSCRIBE	-	-	-

1.5.5 Memcached 命令

Memcached引擎支持基于TCP（Memcached Text Protocol）的文本协议和二进制（Memcached Binary Protocol）协议，任何兼容Memcached协议的客户端都可以访问DCS。

文本协议

Memcached文本协议通过ASCII文本传递命令，便于用户编写客户端和调测问题，甚至可以直接使用Telnet连接Memcached实例。

Memcached文本协议与二进制协议相比，兼容更多的开源类型客户端，但文本协议不支持认证操作。

说明

Memcached实例需要开启免密访问模式后，才能通过文本协议连接。此时Memcached实例的访问将不再受用户名、密码的认证保护。同一VPC内符合安全组规则的任何Memcached客户端均可连接访问Memcached实例，存在安全风险，请谨慎使用。

DCS Memcached实例对文本协议命令的支持情况如表1-31所示。

表 1-31 Memcached 实例文本协议支持的命令说明

命令	功能	是否支持
add	新增数据	是
set	设置数据，主要包括新增或者修改数据	是
replace	更新数据	是
append	向指定key的value末尾追加数据	是
prepend	向指定key的value头部追加数据	是
cas	检查并修改数据	是
get	查询数据	是
gets	查询数据详细信息	是
delete	删除数据	是

命令	功能	是否支持
incr	算数增	是
decr	算数减	是
touch	修改数据过期时间	是
quit	断开连接	是
flush_all	清空数据 **说明** 若提供可选参数delay，则delay取值必须为0。	是
version	查询服务版本信息	是
stats	运行统计信息管理 **说明** 当前只支持查询基础统计信息，不支持可选参数操作命令。	是
cache_memlimit	设置内存使用限制	否
slabs	查询内部存储结构使用情况	否
lru	数据过期删除策略管理	否
lru_crawler	数据过期删除线程管理	否
verbosity	日志级别管理	否
watch	运行事件监控	否

二进制协议

Memcached二进制协议将命令及所操作的内容按照特定结构进行编码后发送，通过预定的字节串表示命令。

Memcached二进制协议与文本协议相比，功能更多，支持SASL认证更加安全，但可用的客户端数量较少。

DCS Memcached实例对二进制协议命令的支持情况如表1-32所示。

表 1-32 Memcached 实例二进制协议支持的命令说明

命令编码	命令	功能	是否支持
0x00	GET	查询数据	是
0x01	SET	设置数据，主要包括新增或者修改数据。	是
0x02	ADD	新增数据	是
0x03	REPLACE	更新数据	是

命令编码	命令	功能	是否支持
0x04	DELETE	删除数据	是
0x05	INCREMENT	算数增	是
0x06	DECREMENT	算数减	是
0x07	QUIT	断开连接	是
0x08	FLUSH	清空数据 **说明** 若提供可选参数delay参数，则delay取值必须为0。	是
0x09	GETQ	查询数据，在出现错误时不返回任何信息。	是
0x0a	NOOP	空操作，相当于ping。	是
0x0b	VERSION	查询服务版本信息	是
0x0c	GETK	查询数据并返回key	是
0x0d	GETKQ	查询数据并返回key，在出现错误时不返回任何信息。	是
0x0e	APPEND	向指定key的value末尾追加数据	是
0x0f	PREPEND	向指定key的value头部增加数据	是
0x10	STAT	查询缓存实例的统计信息 **说明** 当前只支持查询基础统计信息，不支持可选参数操作命令。	是
0x11	SETQ	设置数据，主要包括新增或者修改数据。 在成功时不返回任何信息。	是
0x12	ADDQ	新增数据，在成功时不返回任何信息。	是
0x13	REPLACEQ	更新数据，在成功时不返回任何信息。	是
0x14	DELETEQ	删除数据，在成功时不返回任何信息。	是
0x15	INCREMENTQ	算数增，在成功时不返回任何信息	是
0x16	DECREMENTQ	算数减，在成功时不返回任何信息	是
0x17	QUITQ	断开连接	是
0x18	FLUSHQ	清空数据并不返回任何信息 **说明** 若提供delay参数，则delay取值必须为0。	是

命令编码	命令	功能	是否支持
0x19	APPENDQ	向指定key的value末尾追加数据，在成功时不返回任何信息	是
0x1a	PREPENDQ	向指定key的value头部增加数据，在成功时不返回任何信息	是
0x1c	TOUCH	修改数据过期时间	是
0x1d	GAT	查询数据同时修改过期时间	是
0x1e	GATQ	查询数据同时修改过期时间，在操作失败时不返回任何信息。	是
0x23	GATK	查询数据并返回key，同时修改过期时间。	是
0x24	GATKQ	查询数据并返回key，同时修改过期时间，在操作失败时不返回任何信息。	是
0x20	SASL_LIST_MECHS	获取服务端支持的SASL认证机制	是
0x21	SASL_AUTH	SASL认证	是
0x22	SASL_STEP	SASL复杂认证后续步骤	是

1.5.6 集群实例受限使用命令

集群实例支持多个key，但不支持跨slot访问的Redis命令，如表1-33所示。

表 1-33 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

命令类型	命令描述
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。

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。

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。

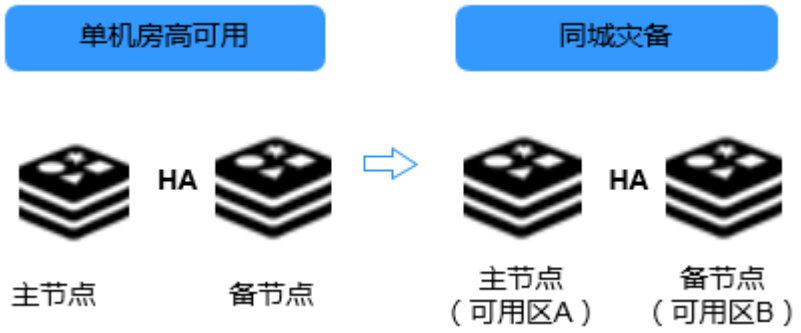
其他限制

- 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 缓存高可用和容灾策略

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 1-8 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

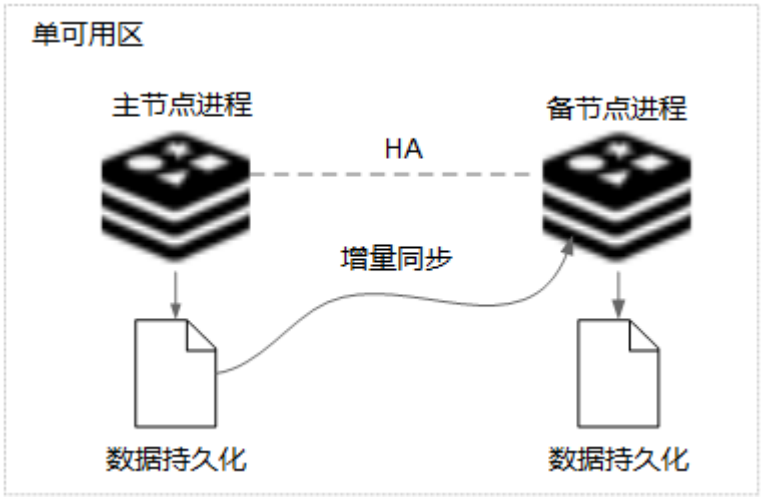
同一机房即单可用区。单可用区灾备策略主要包括进程/服务高可用，数据持久化到磁盘，以及实例节点间热备三种不同层次。

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 1-9 单可用区内单机实例高可用

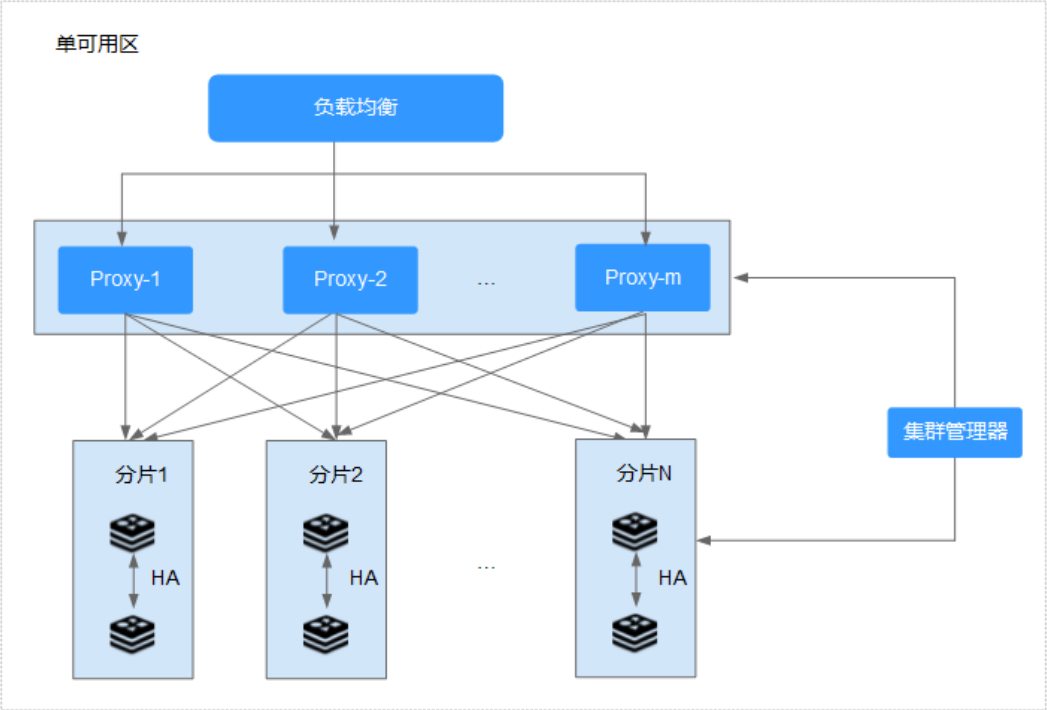
主备实例配置了数据持久化，数据持久化到主节点磁盘外，还会增量同步到备节点，同时备节点也会持久化一份数据。因此，主备实例实现了节点热备和持久化文件多个备份。

图 1-10 单可用区内主备实例高可用



集群版实例类似主备实例，每个条带（实例进程）有持久化文件，也都有对应的副本（备进程及其持久化文件）。

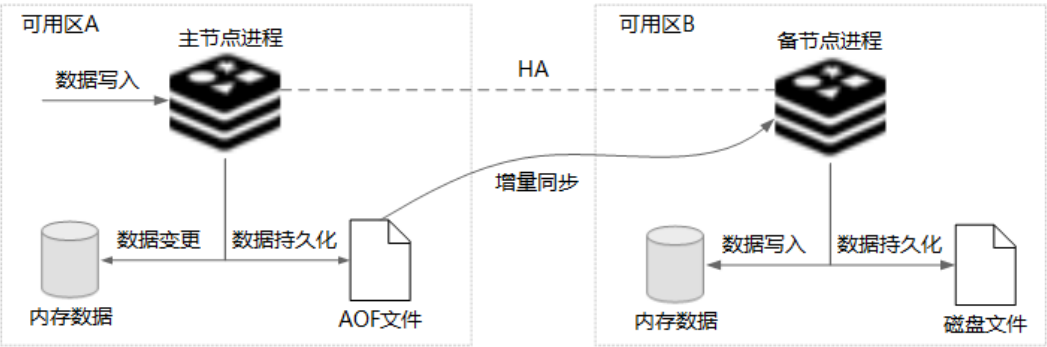
图 1-11 单可用区内集群版实例高可用



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 1-12 实例跨可用区示意图



说明

上图为主备实例跨可用区部署示意图，集群版实例与主备实例类似，每一个条带（进程）都跨可用区部署。

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。

在故障期间，实例备份功能、配置修改、密码修改等功能不可用。

1.7 Redis 版本差异

DCS在创建实例时，Redis可选择“版本号”、“实例类型”。

- **版本号**

  版本号共有3.0，4.0，5.0三个大版本可以选择，它们的区别如下。

  表 1-34 不同版本支持的特性、性能差异说明

比较项	Redis3.0	Redis4.0 & Redis5.0
实例部署模式	采用虚机部署	在物理机上容器化部署
创建实例耗时	3~15分钟，集群约10~30分钟	约8秒
QPS	单节点约10万QPS	单节点约10万QPS
实例类型	支持单机、主备、Proxy集群	支持单机、主备、Cluster集群
实例规格	提供2GB、4GB、8GB直至1024GB多种规格	提供2GB、4GB、8GB直至1024GB多种规格。如果是单机主备实例，还支持128MB、256MB、512MB、1GB四种小规格。
扩容/缩容	支持在线扩容和缩容	支持在线扩容和缩容
备份恢复	主备和集群实例支持	主备和集群实例支持

  说明

  由于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Redis3.0暂不支持升级到Redis4.0或者Redis5.0。如果需要由低版本升级到高版本，建议重新创建高版本实例，然后进行数据迁移。

- **实例类型**

  实例类型分为单机、主备、集群，它们的架构与应用场景，请参考**实例类型**章节。

1.8 Redis 与 Memcached 差异

Redis和Memcached都是非常受欢迎的开源内存数据库，相对关系型数据库，Redis和Memcached使用都简单，且具备高性能。

同为Key-Value数据库，该如何选择?

Memcached适用于数据结构模型简单的场景。Redis适用于数据结构复杂、需要持久化存储数据、存储大key的场景。

具体细节比较，请参考下表。

表 1-35 Redis 与 Memcached 的对比概览

对比项	Redis	Memcached
延时	内存数据库，亚毫秒级延时。	内存数据库，亚毫秒级延时。
易用性	语法简单，易用性强。	语法简单，易用性强。
分布式存储	支持集群方式水平扩展。	支持。
多语言客户端	支持Java、C、Python等三十几种语言的客户端连接。	支持Java、C、Python等十几语言的客户端连接。
线程/进程	单核单线程。 单线程通信，避免不必要的上下文切换与竞争。 采用非阻塞IO（IO多路复用），减少多客户端连接时的资源消耗。	支持多线程，可扩展。 可通过增加CPU数量，提升Memcached性能。 在key的value较大的场景中，性能优势较明显。
持久化存储	支持。 可将每一次写入操作（数据的增加、删除、修改）记录到磁盘文件（AOF文件）中。	支持。 **说明** 开源Memcached不支持持久化存储，DCS Memcached支持持久化存储。
数据结构	支持哈希、列表、集合、有序集合等复杂的数据结构。有更多的应用场景	支持简单的字符串。
Lua脚本支持	支持。	不支持。
快照备份	支持。 快照定期产生，因此不能保证数据100%不丢失。 Redis会fork一个子进程用于生成快照，当数据较多时，可能产生Redis服务短暂中断。	不支持。
Key的Value限制	Key的值最大可以有1GB。	1MB
多数据库	Redis支持多个数据库，默认256个DB。	不支持

由以上对比可知，Redis与Memcached都具有简单易用，性能优越的特点。但在数据结构存储、持久化、备份与迁移、脚本支持等方面有所差异，建议您结合实际应用场景，选择最合适的缓存引擎。

说明

Memcached比较合适小型静态数据的缓存场景，只需要直接读取，不做进一步运算和处理，如html代码片段。

Redis有丰富的数据结构，应用场景更为广泛。

1.9 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 1-36 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。	● Redis3.0版本5~15分钟完成创建。 ● Redis4.0、Redis5.0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Redis3.0、Redis4.0、Redis5.0三个大版本。
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）。
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。

比较项	开源Redis	DCS Redis
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。

与开源 Memcached 差异

表 1-37 DCS 与自建开源 Memcached 的差异说明

比较项	开源社区版Memcached	DCS Memcached
服务搭建	从自行准备服务器资源到Memcached搭建，需要0.5~2天。	5~15分钟完成创建。
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）。
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。

比较项	开源社区版Memcached	DCS Memcached
备份恢复	不支持。	• 提供定时与手动备份数据能力。 • 支持控制台一键恢复数据。
可视化维护	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。
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Memcached内存配置并重启操作系统与服务。	• 提供在线不断服务的扩容能力。 • 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。
持久化	不支持。	主备实例支持持久化。

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。一个帐户中可以创建多个项目。

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，例如主备实例为双副本，Redis集群实例的每个集群节点都为双副本。

维护时间窗

指允许DCS产品服务团队为实例进行升级维护的时间段。

DCS对实例升级维护频率较低，一般每季度一次。虽然频率低，且升级过程不会影响业务，但建议您选择业务量较少的时间段作为维护时间窗。

在创建实例时，都会要求设置一个维护时间窗，您也可以在实例创建后，对维护时间窗进行修改。

具体使用请参考：**配置实例维护时间窗**。

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。

条带

也叫分片，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。

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，您可以在云服务帐号中给员工创建IAM用户，并使用策略来控制他们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。

如果帐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，帐号管理员创建的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，帐号管理员能够控制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授权项请参见细粒度策略支持的授权项。

如**表1**所示，包括了DCS的所有系统权限。

表 1-38 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

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2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 1-39 常用操作与系统策略的关系

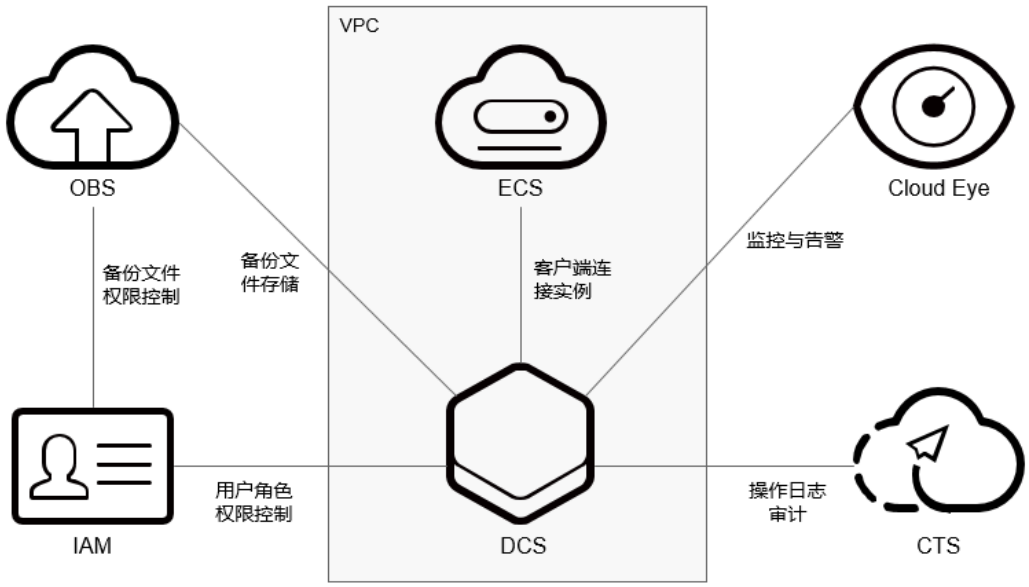
操作	DCS FullAccess	DCS UserAccess	DCS ReadOnlyAccess
修改实例配置参数	√	√	×
删除实例后台任务	√	√	×
Web CLI	√	√	×
修改实例运行状态	√	√	×
缓存实例扩容	√	×	×
修改实例访问密码	√	√	×
修改缓存实例	√	×	×
实例主备倒换	√	√	×
备份实例数据	√	√	×
分析实例的大key或者热key	√	√	×

操作	DCS FullAccess	DCS UserAccess	DCS ReadOnlyAccess
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	√	√	√

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 1-13 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，在您的帐号中，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、高效的其他帐号或者云服务，这些帐号或者云服务可以根据权限进行代运维。

如果帐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。

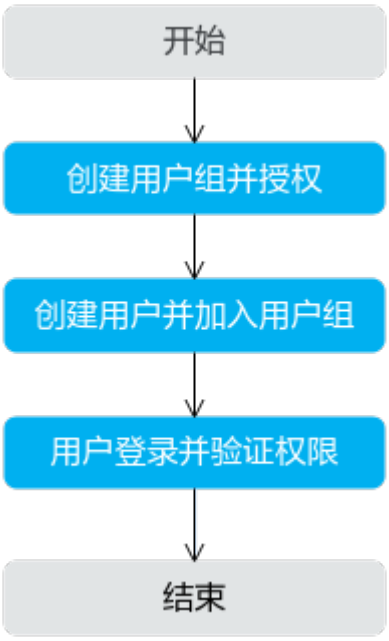
本章节以创建用户并授予“DCS ReadOnlyAccess”权限为例，为您介绍对用户授权的方法，操作流程如**图2-1**所示。

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权限集。

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）请参考细粒度策略支持的授权项。

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 快速入门

3.1 创建实例

3.1.1 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. 确定缓存类型。

   在创建前，需要您根据业务情况选择合适的缓存类型，选择了缓存类型后，不支持更改缓存类型。

   - 若要了解Redis和Memcached，请参考**DCS Redis和DCS Memcached的介绍**。
   - 若要了解Redis和Memcached的区别，请参考**Redis与Memcached之间如何选择**。

2. 确定缓存实例版本。

   不同的Redis版本，特性会不同，可参考**不同Redis版本支持的特性差异说明**。

3. 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、Proxy集群和Cluster集群。实例规格特点和架构，可参考**选择实例类型**。

4. 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**。

5. 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。

6. 确定实例是否配置备份策略。

   当前只有主备和集群实例支持配置备份恢复策略。关于备份恢复，可参考**备份与恢复说明**。

3.1.2 网络环境准备

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VPC，可重复使用，不需要多次创建。

创建 VPC

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“创建虚拟私有云”。

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7**；如果不需要额外创建子网，请执行**步骤8**。

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 > 我的VPC”，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。

步骤8 在左侧导航选择“访问控制 > 安全组”，单击“创建安全组”。根据界面提示创建安全组。如无特殊需求，界面参数均可保持默认。

关于创建安全组的详细信息可以参考《虚拟私有云用户指南》中“安全性 > 安全组 > 创建安全组 ”章节。

----结束

3.1.3 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。

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“创建缓存实例”，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 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。
   本章节选择“Redis”。
2. 在“版本号”区域，选择Redis版本。
   当前DCS支持的Redis版本有：3.0、4.0和5.0。

   说明

   - 如果是Proxy集群实例类型，Redis版本支持选择3.0。
   - 如果是Cluster集群实例类型，Redis版本支持选择4.0和5.0。
3. 在“实例类型”区域，选择单机、主备、Proxy集群或Cluster集群实例类型。
4. 选择“CPU架构”，当前支持“x86计算”。
5. 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Redis4.0和Redis5.0，且实例类型为主备、Cluster集群时，页面才显示“副本数”。
6. 在“可用区”区域，您可根据实际情况选择。
   如果“实例类型”选择了主备、Proxy集群、Cluster集群，页面增加显示“备可用区”，您需要在“备可用区”为备节点设置备可用区。

   说明

   - 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“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。

步骤6 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、IP。
   支持自动分配IP地址和手动分配IP地址，用户可以输入一个在当前子网下可用的IP。
   另外，Redis4.0和Redis5.0支持自定义端口，自定义端口范围为1~65535的任意数字；如果未自定义，则使用默认端口6379。Redis3.0不支持自定义端口，端口都为6379。

2. 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。

   Redis4.0和Redis5.0是基于VPCEndpoint，暂不支持安全组，当选择的Reids版本为4.0或5.0，页面不支持设置该参数，只有Redis版本为3.0时才支持设置实例“安全组”。

步骤7 设置实例密码。

该参数表示连接Redis实例的密码。

说明

DCS服务出于安全考虑，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。

DCS帐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8 单击“更多配置”，设置实例其他信息，包括开启实例自动备份等配置。

1. 设置实例的“名称”和“描述”。

   创建实例时，名称长度为4到64位的字符串。

2. 选择是否开启“自动备份”。

   只有当实例类型为主备或者集群时显示该参数。关于实例备份的说明及备份策略的设置请参考**备份与恢复说明**。

3. 重命名实例高危命令。

   当创建的是Redis4.0和Redis5.0实例时，支持重命名高危命令。当前支持的高危命令有command、keys、flushdb、flushall和hgetall，其他命令暂时不支持重命名。

4. 设置实例维护时间窗。

   设置DCS服务运维对实例进行维护的时间，在维护前，服务运维会提前和您沟通确认。

步骤9 实例信息配置完成后，单击“立即创建”，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10 确认实例信息无误后，提交请求。

步骤11 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 3.0单机和主备实例大约需要5到15分钟，如果集群实例，则需要大约30分钟。

   说明

   Redis4.0和Redis5.0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

3.1.4 创建 Memcached 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Memcached实例。

创建 Memcached 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 单击“创建缓存实例”，进入实例创建页面。

步骤5 在“区域”下拉列表中，选择靠近你应用程序的区域，可降低网络延时、提高访问速度。

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，选择缓存类型。

   本章节单击选择“Memcached”。

2. 在“实例类型”区域，选择实例类型。

   Memcached只支持“单机”和“主备”实例类型。

3. 在“可用区”区域，您可根据实际情况选择。

   说明

   如果提高访问速度，可选择和应用同一个可用区；如果提高可靠性，可选择和应用不在同一个可用区。

   如果“实例类型”选择了主备，页面增加显示“备可用区”参数，您需要在“备可用区”为备实例设置备可用区。

4. 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。

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、IP。

   支持自动分配IP地址和手动分配IP地址，用户可以输入一个在当前子网下可用的IP。

2. 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。

步骤8 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Memcached实例时是否要进行密码验证。

说明

- 选择“免密访问”，存在安全风险，请谨慎使用。
- 实例申请成功后，可以通过“重置密码”进行密码重新设置。
- Memcached密码访问模式下，实例只能使用二进制访问且需要进行SASL鉴权。

- “用户名”：连接Memcached实例的用户名。

  说明

  只有“访问方式”为“密码访问”时，才会显示该参数。

  - 名称不能为空。
  - 只能以英文字母开头。
  - 长度为1到64位的字符串。
  - 仅包含英文字母、数字、下划线（_）和中划线（-）。
- “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Memcached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Memcached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9 单击“更多配置”，设置实例其他信息，包括备份恢复策略、实例维护时间窗等配置。

1. 设置实例的“名称”和“描述”。

   名称长度为4到64位的字符串。
2. 设置实例备份恢复策略。

   只有当实例类型为“主备”时显示该参数。关于实例备份的说明及备份策略的设置请参考**备份与恢复说明**。
3. 设置实例维护时间窗。

   您可以设置DCS服务运维对实例进行维护的时间，可选择22:00-02:00、02:00-06:00、06:00-10:00、10:00-14:00、14:00-18:00和18:00-22:00，在选择的时间段内，则可对实例节点进行维护操作。

步骤10 实例信息配置完成后，单击“立即创建”，进入确认页面。

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11 确认实例信息无误后，提交请求。

步骤12 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. 创建缓存实例大约需要5到15分钟。
2. 缓存实例创建成功后，默认“状态”为“运行中”。

----结束

3.2 连接实例

3.2.1 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

- Redis3.0不支持定义端口，端口固定为6379，Redis4.0和Redis5.0支持定义端口，如果不自定义端口，则使用默认端口6379。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**。

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。

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

   wget http://download.redis.io/releases/redis-5.0.8.tar.gz
3. 执行如下命令，解压Redis客户端源码包。

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

   cd redis-5.0.8

   make

   cd src

步骤3 连接Redis非Cluster集群实例。

如果是Redis3.0、Redis4.0单机/主备、Redis5.0单机/主备实例，请执行以下操作。

./redis-cli -h ${实例IP} -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 ${实例IP} -p 6379
2. 如果实例为非免密实例，连接实例使用命令：./redis-cli -h ${实例IP} -p 6379 -a ${password}

步骤4 连接Redis Cluster集群实例。

如果是Redis4.0 Cluster集群、Redis5.0 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-5.0.8/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**查看实例信息**，请按实际情况修改后执行。

3.2.2 多语言连接

3.2.2.1 Java 客户端

3.2.2.1.1 Jedis

介绍使用同一VPC内弹性云服务器ECS上的Jedis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

- 如果创建Redis实例时设置了密码，使用Jedis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。
- 如果使用JedisCluster连接Redis4.0/5.0 Cluster集群，集群内部拓扑结构会自动刷新，客户端需要自己处理重连问题。

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java编译环境

操作步骤

步骤1 查看并获取待连接Redis实例的IP地址/域名和端口。

具体步骤请参见**查看实例信息**。

步骤2 登录弹性云服务器。

步骤3 首先使用maven在pom.xml添加如下依赖。

```
<dependency>
    <groupId>redis.clients</groupId>
    <artifactId>jedis</artifactId>
    <version>4.1.1</version>
</dependency>
```

步骤4 使用Redis Java (Jedis)客户端连接Redis实例。

获取Redis Java (Jedis)客户端源码，具体方法可参考**获取Jedis源码**。Jedis客户端访问DCS Redis服务，有以下两种方法：

- Jedis单连接

- Jedis连接池连接

具体示例如下：

1. jedis单连方式连接Redis单机/主备/Proxy集群示例。

```
// 密码模式创建连接
String host = "192.168.0.150";
int port = 6379;
String pwd = "passwd";

Jedis client = new Jedis(host, port);
client.auth(pwd);
client.connect();
// 执行set指令
String result = client.set("key-string", "Hello, Redis!");
System.out.println( String.format("set指令执行结果:%s", result) );
// 执行get指令
String value = client.get("key-string");
System.out.println( String.format("get指令执行结果:%s", value) );

// 免密模式创建连接
String host = "192.168.0.150";
int port = 6379;

Jedis client = new Jedis(host, port);
client.connect();
// 执行set指令
String result = client.set("key-string", "Hello, Redis!");
System.out.println( String.format("set指令执行结果:%s", result) );
// 执行get指令
String value = client.get("key-string");
System.out.println( String.format("get指令执行结果:%s", value) );
```

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

2. jedis连接池连接Redis单机/主备/Proxy集群示例。

```
// 密码模式生成连接池配置信息
String ip = "192.168.0.150";
int port = 6379;
String pwd = "passwd";
GenericObjectPoolConfig config = new GenericObjectPoolConfig();
config.setTestOnBorrow(false);
config.setTestOnReturn(false);
config.testWhileIdle(true);
config.setMaxTotal(100);
config.setMaxIdle(100);
config.setMaxWaitMillis(2000);
JedisPool pool = new JedisPool(config, ip, port, 100000, pwd);// 在应用初始化的时候生成连接池
// 在业务操作时，从连接池获取连接
Jedis client = pool.getResource();
try {
    // 执行指令
    String result = client.set("key-string", "Hello, Redis!");
    System.out.println( String.format("set指令执行结果:%s", result) );
    String value = client.get("key-string");
    System.out.println( String.format("get指令执行结果:%s", value) );
} catch (Exception e) {
    // TODO: handle exception
} finally {
    // 业务操作完成，将连接返回给连接池
    if (null != client) {
        pool.returnResource(client);
    }
} // end of try block
// 应用关闭时，释放连接池资源
pool.destroy();
```

```
// 免密模式生成连接池配置信息
String ip = "192.168.0.150";
int port = 6379;
GenericObjectPoolConfig config = new GenericObjectPoolConfig();
config.setTestOnBorrow(false);
config.setTestOnReturn(false);
config.testWhileIdle(true);
config.setMaxTotal(100);
config.setMaxIdle(100);
config.setMaxWaitMillis(2000);
JedisPool pool = new JedisPool(config, ip, port, 100000);// 在应用初始化的时候生成连接池
// 在业务操作时，从连接池获取连接
Jedis client = pool.getResource();
try {
    // 执行指令
    String result = client.set("key-string", "Hello, Redis!");
    System.out.println( String.format("set指令执行结果:%s", result) );
    String value = client.get("key-string");
    System.out.println( String.format("get指令执行结果:%s", value) );
} catch (Exception e) {
    // TODO: handle exception
} finally {
    // 业务操作完成，将连接返回给连接池
    if (null != client) {
        pool.returnResource(client);
    }
} // end of try block
// 应用关闭时，释放连接池资源
pool.destroy();
```

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

连接池testOnBorrow参数开启后支持自动重连，业务从连接池获取redis连接时连接池会检测连接，检测到正常的连接后再返回给业务，会有性能损耗，在性能要求高的使用场景建议不开启该参数，由上层应用自行处理异常和重试。

3. 单连接方式连接Redis Cluster集群代码示例。

   - 加密访问的实例：

     ```
     // 以下是加密访问
     int port = 6379;
     String host = "192.168.144.37";
     // 创建jediscluster
     Set<HostAndPort> nodes = new HashSet<HostAndPort>();
     nodes.add(new HostAndPort(host, port));
     JedisCluster cluster = new JedisCluster(nodes, 5000, 3000, 10, "password", new
     JedisPoolConfig());
     cluster.set("key", "value");
     System.out.println("Connected to RedisCluster:" + cluster.get("key"));
     cluster.close();
     ```

   - 免密访问的实例：

     ```
     int port = 6379;
     String host = "192.168.144.37";
     // 创建jediscluster
     Set<HostAndPort> nodes = new HashSet<HostAndPort>();
     nodes.add(new HostAndPort(host, port));
     JedisCluster cluster = new JedisCluster(nodes);
     cluster.set("key", "value");
     System.out.println("Connected to RedisCluster:" + cluster.get("key"));
     cluster.close();
     ```

   其中，**host**为Redis实例的IP地址/域名，**port**为Redis实例的端口。IP地址/域名和端口获取见**步骤1**，请按实际情况修改后执行。**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。

步骤5 参考Redis Java (Jedis) 客户端源码中的readme文件编译代码并运行Redis Java (Jedis)客户端连接Redis实例。

----结束

3.2.2.1.2 Lettuce

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

说明

如果创建Redis实例时设置了密码，使用Lettuce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、主备、Proxy集群实例可使用Lettuce的RedisClient对象，Cluster集群实例需要使用RedisClusterClient对象。

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java编译环境。

操作步骤

步骤1 查看并获取待连接Redis实例的IP地址/域名和端口。

具体步骤请参见**查看实例信息**。

步骤2 登录弹性云服务器。

步骤3 首先使用maven在pom.xml添加如下依赖。

```
<dependency>
  <groupId>io.lettuce</groupId>
  <artifactId>lettuce-core</artifactId>
  <version>6.1.6.RELEASE</version>
</dependency>
```

步骤4 使用Redis Java (Lettuce)客户端连接实例。

- Lettuce单连方式连接Redis单机、主备、proxy集群示例。

```
// password是连接的密码，不需验证删除password@即可,密码中有特殊字符时需要进行转码
RedisClient redisClient = RedisClient.create("redis://password@host:port");
StatefulRedisConnection<String, String> connection = redisClient.connect();
RedisCommands<String, String> syncCommands = connection.sync();
syncCommands.set("key", "value");
System.out.println("Connected to Redis:" + syncCommands.get("key"));
// 连接关闭
connection.close();
// 客户端关闭
redisClient.shutdown();
```

- Lettuce连接池方式连接Redis单机、主备、proxy集群示例。

```
// password是连接的密码，不需验证删除password@即可,密码中有特殊字符时需要进行转码
RedisClient clusterClient = RedisClient.create("redis://password@host:port");
GenericObjectPoolConfig<StatefulRedisConnection<String, String>> genericObjectPoolConfig = new
GenericObjectPoolConfig();
// 连接池相关参数配置
genericObjectPoolConfig.setMaxIdle(3);
genericObjectPoolConfig.setMinIdle(2);
genericObjectPoolConfig.setMaxTotal(3);
```

```
genericObjectPoolConfig.setMaxWaitMillis(-1);
GenericObjectPool<StatefulRedisConnection<String, String>> pool = ConnectionPoolSupport
        .createGenericObjectPool(() -> clusterClient.connect(), genericObjectPoolConfig);
// 获取连接进行操作
try (StatefulRedisConnection<String, String> con = pool.borrowObject()) {
    RedisCommands<String, String> sync = con.sync();
    sync.set("key", "value");
    System.out.println("Connected by pool:" + sync.get("key"));
} catch (Exception e) {
    e.printStackTrace();
}finally {
    // 资源关闭
    pool.close();
    clusterClient.shutdown();
}
```

- Lettuce连接cluster集群示例。

```
// password是连接的密码，不需验证删除password@即可,密码中有特殊字符时需要进行转码
RedisClusterClient redisClient = RedisClusterClient.create("redis://password@host:port");
StatefulRedisClusterConnection<String, String> connection = redisClient.connect();
RedisAdvancedClusterCommands<String, String> syncCommands = connection.sync();
syncCommands.set("key", "value");
System.out.println("Connected to RedisCluster:"+syncCommands.get("key"));
// 连接关闭
connection.close();
// 客户端关闭
redisClient.shutdown();
```

----结束

3.2.2.1.3 Redisson

介绍使用同一VPC内弹性云服务器ECS上的Redisson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**。

说明

- 如果创建Redis实例时设置了密码，使用Redisson客户端连接Redis时，需要配置密码进行连接，建议不要将明文密码硬编码在代码中。
- 连接单机、主备、Proxy集群实例需要使用Redisson的SingleServerConfig配置对象中的useSingleServer方法，Cluster集群实例需要使用ClusterServersConfig对象中的useClusterServers方法。

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java编译环境。

操作步骤

步骤1 查看并获取待连接Redis实例的IP地址/域名和端口。

具体步骤请参见**查看实例信息**。

步骤2 登录弹性云服务器。

步骤3 首先使用maven在pom.xml添加如下依赖。

```
<dependency>
  <groupId>org.redisson</groupId>
  <artifactId>redisson</artifactId>
```

```
  <version>3.16.8</version>
</dependency>
```

步骤4 连接池推荐配置

连接保活推荐配置：

```
# ping连接间隔（配置PING探测会增加Redis负载，需要根据连接数调整，建议值至少配置为1000以上，连接数越多值配置越大，Redis活跃连接数超过5000时不建议配置）
pingConnectionInterval: 3000
```

单机模式配置示例：（超时时间、连接池大小等需要根据业务实际情况进行调整，本章节仅作为示例值）

```
redisson:
  config:
    singleServerConfig:
      # 连接超时，单位：毫秒
      connectTimeout: 10000
      # 命令等待超时，单位：毫秒
      timeout: 3000
      # 命令失败重试次数
      retryAttempts: 3
      # 命令重试发送时间间隔，单位：毫秒
      retryInterval: 1500
      # 最小空闲连接数
      connectionMinimumIdleSize: 30
      # 连接池大小
      connectionPoolSize: 50
      # redis数据库编号
      database: 0
      # DNS监测时间间隔，单位：毫秒
      dnsMonitoringInterval: 5000
    # ping连接间隔
    pingConnectionInterval: 3000
```

集群模式配置示例：（超时时间、连接池大小等需要根据业务实际情况进行调整）

```
redisson:
  config:
    clusterServersConfig:
      # 连接空闲超时，单位：毫秒
      idleConnectionTimeout: 100000
      # 连接超时，单位：毫秒
      connectTimeout: 10000
      # 命令等待超时，单位：毫秒
      timeout: 3000
      # 命令失败重试次数
      retryAttempts: 3
      # 命令重试发送时间间隔，单位：毫秒
      retryInterval: 1500
      # 失败从节点重连间隔时间
      failedSlaveReconnectionInterval: 3000
      # 失败从节点校验间隔时间
      failedSlaveCheckInterval: 60000
      # 单个连接最大订阅数量
      subscriptionsPerConnection: 5
      # 客户端名称
      clientName: null
      # 发布和订阅连接的最小空闲连接数
      subscriptionConnectionMinimumIdleSize: 1
      # 发布和订阅连接池大小
      subscriptionConnectionPoolSize: 50
      # 从节点最小空闲连接数
      slaveConnectionMinimumIdleSize: 24
      # 从节点连接池大小
      slaveConnectionPoolSize: 64
      # 主节点最小空闲连接数
```

```
masterConnectionMinimumIdleSize: 24
# 主节点连接池大小
masterConnectionPoolSize: 64
# 对主节点变化节点状态扫描的时间间隔 - 单位是毫秒
scanInterval: 1000
# ping连接间隔
pingConnectionInterval: 3000
# 是否保持连接
keepAlive: false
# 默认启用 tcpNoDelay 设置
tcpNoDelay: false
```

步骤5 使用Redis Java (Redisson)客户端连接实例。

- Redisson单连方式连接Redis单机、主备、proxy集群示例。

```
Config config = new Config();
SingleServerConfig singleServerConfig = config.useSingleServer();
singleServerConfig.setAddress("redis://host:port");
// singleServerConfig.setPassword("********");
RedissonClient redisson = Redisson.create(config);
// 测试concurrentMap,put方法的时候就会同步到redis中
ConcurrentMap<String, Object> map = redisson.getMap("FirstMap");
map.put("wanger", "男");
map.put("zhangsan", "nan");
map.put("lisi", "女");
ConcurrentMap resultMap = redisson.getMap("FirstMap");
System.out.println("resultMap==" + resultMap.keySet());
// 测试Set集合
Set mySet = redisson.getSet("MySet");
mySet.add("wanger");
mySet.add("lisi");
Set resultSet = redisson.getSet("MySet");
System.out.println("resultSet===" + resultSet.size());
// 测试Queue队列
Queue myQueue = redisson.getQueue("FirstQueue");
myQueue.add("wanger");
myQueue.add("lili");
myQueue.add("zhangsan");
myQueue.peek();
myQueue.poll();
Queue resultQueue = redisson.getQueue("FirstQueue");
System.out.println("resultQueue===" + resultQueue);
// 关闭连接
redisson.shutdown();
```

- Redisson连接池方式连接Redis单机、主备、proxy集群示例。

```
// 1.初始化
Config config = new Config();
SingleServerConfig singleServerConfig = config.useSingleServer();
singleServerConfig.setAddress("redis://host:6379");
//设置对于master节点的连接池中连接数最大为500
singleServerConfig.setConnectionPoolSize(500);
// 那么这些连接将会自动被关闭，并从连接池里去掉。时间单位是毫秒。
singleServerConfig.setIdleConnectionTimeout(10000);
RedissonClient redisson = Redisson.create(config);
// 测试concurrentMap,put方法的时候就会同步到redis中
ConcurrentMap<String, Object> map = redisson.getMap("FirstMap");
map.put("wanger", "男");
map.put("zhangsan", "nan");
map.put("lisi", "女");
ConcurrentMap resultMap = redisson.getMap("FirstMap");
System.out.println("resultMap==" + resultMap.keySet());
// 测试Set集合
Set mySet = redisson.getSet("MySet");
mySet.add("wanger");
mySet.add("lisi");
Set resultSet = redisson.getSet("MySet");
System.out.println("resultSet===" + resultSet.size());
// 测试Queue队列
Queue myQueue = redisson.getQueue("FirstQueue");
```

```
myQueue.add("wanger");
myQueue.add("lili");
myQueue.add("zhangsan");
myQueue.peek();
myQueue.poll();
Queue resultQueue = redisson.getQueue("FirstQueue");
System.out.println("resultQueue===" + resultQueue);
// 关闭连接
redisson.shutdown();
```

- Redisson连接cluster集群示例。

```
Config config = new Config();
ClusterServersConfig clusterServersConfig = config.useClusterServers();
clusterServersConfig.addNodeAddress("redis://host:port");
// 设置密码
// clusterServersConfig.setPassword("********");
RedissonClient redisson = Redisson.create(config);
ConcurrentMap<String, Object> map = redisson.getMap("FirstMap");
map.put("wanger", "男");
map.put("zhangsan", "nan");
map.put("lisi", "女");
ConcurrentMap resultMap = redisson.getMap("FirstMap");
System.out.println("resultMap==" + resultMap.keySet());
// 2.测试Set集合
Set mySet = redisson.getSet("MySet");
mySet.add("wanger");
mySet.add("lisi");
Set resultSet = redisson.getSet("MySet");
System.out.println("resultSet===" + resultSet.size());
//3.测试Queue队列
Queue myQueue = redisson.getQueue("FirstQueue");
myQueue.add("wanger");
myQueue.add("lili");
myQueue.add("zhangsan");
myQueue.peek();
myQueue.poll();
Queue resultQueue = redisson.getQueue("FirstQueue");
System.out.println("resultQueue===" + resultQueue);
// 关闭连接
redisson.shutdown();
```

----结束

3.2.2.2 SpringBoot 集成 Lettuce

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java编译环境。

操作步骤

步骤1 查看并获取待连接Redis实例的IP地址/域名和端口。

具体步骤请参见**查看实例信息**。

步骤2 登录弹性云服务器。

步骤3 首先使用maven在pom.xml添加如下依赖。

说明

- SpringBoot从2.0起默认使用lettuce客户端进行连接。
- 此次使用的版本：springboot：2.6.6，lettuce：6.1.8。

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-data-redis</artifactId>
</dependency>
```

步骤4 使用SpringBoot集成Lettuce连接实例。

- Springboot+Lettuce单连方式连接Redis单机/主备/Proxy集群示例。
  a. 在application.properties配置文件中加上redis相关配置。

```
spring.redis.host=host
spring.redis.database=0
spring.redis.password=pwd
spring.redis.port=port
```

  b. Redis配置类RedisConfiguration。

```
@Bean
public RedisTemplate<String, Object> redisTemplate(LettuceConnectionFactory
lettuceConnectionFactory) {
    RedisTemplate<String, Object> template = new RedisTemplate<>();
    template.setConnectionFactory(lettuceConnectionFactory);
    //使用Jackson2JsonRedisSerializer替换默认的JdkSerializationRedisSerializer来序列化和反序列化
redis的value值
    Jackson2JsonRedisSerializer<Object> jackson2JsonRedisSerializer = new
Jackson2JsonRedisSerializer<>(Object.class);
    ObjectMapper mapper = new ObjectMapper();
    mapper.setVisibility(PropertyAccessor.ALL, JsonAutoDetect.Visibility.ANY);
    mapper.activateDefaultTyping(LaissezFaireSubTypeValidator.instance,
        ObjectMapper.DefaultTyping.NON_FINAL, JsonTypeInfo.As.PROPERTY);
    jackson2JsonRedisSerializer.setObjectMapper(mapper);
    StringRedisSerializer stringRedisSerializer = new StringRedisSerializer();
    //key采用String的序列化方式
    template.setKeySerializer(stringRedisSerializer);
    // hash的key也采用String的序列化方式
    template.setHashKeySerializer(stringRedisSerializer);
    // value序列化方式采用jackson
    template.setValueSerializer(jackson2JsonRedisSerializer);
    // hash的value序列化方式采用jackson
    template.setHashValueSerializer(jackson2JsonRedisSerializer);
    template.afterPropertiesSet();
    return template;
}
```

  c. Redis操作类RedisUtil。

```
/**
 * 普通缓存获取
 * @param key 键
 * @return 值
 */
public Object get(String key){
    return key==null?null:redisTemplate.opsForValue().get(key);
}

/**
 * 普通缓存放入
 * @param key 键
 * @param value 值
 * @return true成功 false失败
 */
public boolean set(String key,Object value) {
    try {
```

```
            redisTemplate.opsForValue().set(key, value);
            return true;
        } catch (Exception e) {
            e.printStackTrace();
            return false;
        }
    }
```

d. 编写controller类进行测试。

```
 @RestController
public class HelloRedis {
    @Autowired
    RedisUtil redisUtil;


    @RequestMapping("/setParams")
    @ResponseBody
    public String setParams(String name) {
            redisUtil.set("name", name);
            return "success";
        }

    @RequestMapping("/getParams")
    @ResponseBody
    public String getParams(String name) {
    System.out.println("--------------" + name + "--------------");
    String retName = redisUtil.get(name) + "";
    return retName;
    }

    }
```

- SpringBoot+Lettuce连接池方式连接Redis单机/主备/Proxy集群示例。

  a. 在以上maven依赖的基础上添加以下依赖。

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

  b. 在application.properties配置文件中加上redis相关配置。

  ```
  spring.redis.host=host
  spring.redis.database=0
  spring.redis.password=pwd
  spring.redis.port=port
  # 连接超时时间
  spring.redis.timeout=1000
  # 连接池最大连接数（使用负值表示没有限制）
  spring.redis.lettuce.pool.max-active=50
  # 连接池中的最小空闲连接
  spring.redis.lettuce.pool.min-idle=5
  # 连接池中的最大空闲连接
  spring.redis.lettuce.pool.max-idle=50
  # 连接池最大阻塞等待时间（使用负值表示没有限制）
  spring.redis.lettuce.pool.max-wait=5000
  #eviction线程调度时间间隔
  spring.redis.pool.time-between-eviction-runs-millis=2000
  ```

  c. Redis连接配置类RedisConfiguration。

  ```
  @Bean
  public RedisTemplate<String, Object> redisTemplate(LettuceConnectionFactory
  lettuceConnectionFactory) {
      lettuceConnectionFactory.setShareNativeConnection(false);
      RedisTemplate<String, Object> template = new RedisTemplate<>();
      template.setConnectionFactory(lettuceConnectionFactory);
      //使用Jackson2JsonRedisSerializer替换默认的JdkSerializationRedisSerializer来序列化和反序列化
  redis的value值
      Jackson2JsonRedisSerializer<Object> jackson2JsonRedisSerializer = new
  Jackson2JsonRedisSerializer<>(Object.class);
      ObjectMapper mapper = new ObjectMapper();
      mapper.setVisibility(PropertyAccessor.ALL, JsonAutoDetect.Visibility.ANY);
  ```

```
    mapper.activateDefaultTyping(LaissezFaireSubTypeValidator.instance,
        ObjectMapper.DefaultTyping.NON_FINAL, JsonTypeInfo.As.PROPERTY);
    jackson2JsonRedisSerializer.setObjectMapper(mapper);
    StringRedisSerializer stringRedisSerializer = new StringRedisSerializer();
    //key采用String的序列化方式
    template.setKeySerializer(stringRedisSerializer);
    // hash的key也采用String的序列化方式
    template.setHashKeySerializer(stringRedisSerializer);
    // value序列化方式采用jackson
    template.setValueSerializer(jackson2JsonRedisSerializer);
    // hash的value序列化方式采用jackson
    template.setHashValueSerializer(jackson2JsonRedisSerializer);
    template.afterPropertiesSet();
    return template;
}
```

- SpringBoot+Lettuce单连接方式连接Redis Cluster集群代码示例。
  a. 在application.properties配置文件中加上redis相关配置。

```
spring.redis.cluster.nodes=host:port
spring.redis.cluster.max-redirects=3
spring.redis.password= pwd
# 自动刷新时间
spring.redis.lettuce.cluster.refresh.period=60
# 开启自适应刷新
spring.redis.lettuce.cluster.refresh.adaptive=true
spring.redis.timeout=60
```

  b. Redis配置类RedisConfiguration，请务必开启集群自动刷新拓扑配置。

```
@Bean
public LettuceConnectionFactory lettuceConnectionFactory() {
    String[] nodes = clusterNodes.split(",");
    List<RedisNode> listNodes = new ArrayList();
    for (String node : nodes) {
        String[] ipAndPort = node.split(":");
        RedisNode redisNode = new RedisNode(ipAndPort[0], Integer.parseInt(ipAndPort[1]));
        listNodes.add(redisNode);
    }
    RedisClusterConfiguration redisClusterConfiguration = new RedisClusterConfiguration();
    redisClusterConfiguration.setClusterNodes(listNodes);
    redisClusterConfiguration.setPassword(password);
    redisClusterConfiguration.setMaxRedirects(maxRedirects);
     // 配置集群自动刷新拓扑
    ClusterTopologyRefreshOptions topologyRefreshOptions =
ClusterTopologyRefreshOptions.builder()
        .enablePeriodicRefresh(Duration.ofSeconds(period)) //按照周期刷新拓扑
        .enableAllAdaptiveRefreshTriggers() //根据事件刷新拓扑
        .build();

    ClusterClientOptions clusterClientOptions = ClusterClientOptions.builder()
        //redis命令超时时间,超时后才会使用新的拓扑信息重新建立连接
        .timeoutOptions(TimeoutOptions.enabled(Duration.ofSeconds(period)))
        .topologyRefreshOptions(topologyRefreshOptions)
        .build();
    LettuceClientConfiguration clientConfig = LettucePoolingClientConfiguration.builder()
            .commandTimeout(Duration.ofSeconds(timeout))
            .readFrom(ReadFrom.REPLICA_PREFERRED) // 优先从副本读取
            .clientOptions(clusterClientOptions)
            .build();
    LettuceConnectionFactory factory = new
LettuceConnectionFactory(redisClusterConfiguration, clientConfig);
    return factory;
}

@Bean
public RedisTemplate<String, Object> redisTemplate(LettuceConnectionFactory
lettuceConnectionFactory) {
    RedisTemplate<String, Object> template = new RedisTemplate<>();
    template.setConnectionFactory(lettuceConnectionFactory);
    //使用Jackson2JsonRedisSerializer替换默认的JdkSerializationRedisSerializer来序列化和反序列化
redis的value值
```

```
    Jackson2JsonRedisSerializer<Object> jackson2JsonRedisSerializer = new
Jackson2JsonRedisSerializer<>(Object.class);
    ObjectMapper mapper = new ObjectMapper();
    mapper.setVisibility(PropertyAccessor.ALL, JsonAutoDetect.Visibility.ANY);
    mapper.activateDefaultTyping(LaissezFaireSubTypeValidator.instance,
        ObjectMapper.DefaultTyping.NON_FINAL, JsonTypeInfo.As.PROPERTY);
    jackson2JsonRedisSerializer.setObjectMapper(mapper);
    StringRedisSerializer stringRedisSerializer = new StringRedisSerializer();
    //key采用String的序列化方式
    template.setKeySerializer(stringRedisSerializer);
    // hash的key也采用String的序列化方式
    template.setHashKeySerializer(stringRedisSerializer);
    // value序列化方式采用jackson
    template.setValueSerializer(jackson2JsonRedisSerializer);
    // hash的value序列化方式采用jackson
    template.setHashValueSerializer(jackson2JsonRedisSerializer);
    template.afterPropertiesSet();
    return template;
}
```

- springboot+lettuce连接池方式连接Redis Cluster集群代码示例。
  a. 在application.properties配置文件中加上Redis相关配置。

```
spring.redis.cluster.nodes=host:port
spring.redis.cluster.max-redirects=3
spring.redis.password=pwd
spring.redis.lettuce.cluster.refresh.period=60
spring.redis.lettuce.cluster.refresh.adaptive=true
# 连接超时时间
spring.redis.timeout=60s
# 连接池最大连接数（使用负值表示没有限制）
spring.redis.lettuce.pool.max-active=50
# 连接池中的最小空闲连接
spring.redis.lettuce.pool.min-idle=5
# 连接池中的最大空闲连接
spring.redis.lettuce.pool.max-idle=50
# 连接池最大阻塞等待时间（使用负值表示没有限制）
spring.redis.lettuce.pool.max-wait=5000
#eviction线程调度时间间隔
spring.redis.lettuce.pool.time-between-eviction-runs=2000
```

  b. redis配置类RedisConfiguration，请务必开启集群自动刷新拓扑配置。

```
@Bean
 public LettuceConnectionFactory lettuceConnectionFactory() {
    GenericObjectPoolConfig genericObjectPoolConfig = new GenericObjectPoolConfig();
    genericObjectPoolConfig.setMaxIdle(maxIdle);
    genericObjectPoolConfig.setMinIdle(minIdle);
    genericObjectPoolConfig.setMaxTotal(maxActive);
    genericObjectPoolConfig.setMaxWait(Duration.ofMillis(maxWait));

genericObjectPoolConfig.setTimeBetweenEvictionRuns(Duration.ofMillis(timeBetweenEvictionRu
nsMillis));
    String[] nodes = clusterNodes.split(",");
    List<RedisNode> listNodes = new ArrayList();
    for (String node : nodes) {
        String[] ipAndPort = node.split(":");
        RedisNode redisNode = new RedisNode(ipAndPort[0], Integer.parseInt(ipAndPort[1]));
        listNodes.add(redisNode);
    }
    RedisClusterConfiguration redisClusterConfiguration = new RedisClusterConfiguration();
    redisClusterConfiguration.setClusterNodes(listNodes);
    redisClusterConfiguration.setPassword(password);
    redisClusterConfiguration.setMaxRedirects(maxRedirects);
     // 配置集群自动刷新拓扑
    ClusterTopologyRefreshOptions topologyRefreshOptions =
ClusterTopologyRefreshOptions.builder()
        .enablePeriodicRefresh(Duration.ofSeconds(period)) //按照周期刷新拓扑
        .enableAllAdaptiveRefreshTriggers() //根据事件刷新拓扑
        .build();

    ClusterClientOptions clusterClientOptions = ClusterClientOptions.builder()
```

```
        //redis命令超时时间,超时后才会使用新的拓扑信息重新建立连接
        .timeoutOptions(TimeoutOptions.enabled(Duration.ofSeconds(period)))
        .topologyRefreshOptions(topologyRefreshOptions)
        .build();
    LettuceClientConfiguration clientConfig = LettucePoolingClientConfiguration.builder()
            .commandTimeout(Duration.ofSeconds(timeout))
            .poolConfig(genericObjectPoolConfig)
            .readFrom(ReadFrom.REPLICA_PREFERRED) // 优先从副本读取
            .clientOptions(clusterClientOptions)
            .build();
    LettuceConnectionFactory factory = new
LettuceConnectionFactory(redisClusterConfiguration, clientConfig);
    return factory;
}

@Bean
public RedisTemplate<String, Object> redisTemplate(LettuceConnectionFactory
lettuceConnectionFactory) {
    lettuceConnectionFactory.setShareNativeConnection(false);
    RedisTemplate<String, Object> template = new RedisTemplate<>();
    template.setConnectionFactory(lettuceConnectionFactory);
    //使用Jackson2JsonRedisSerializer替换默认的JdkSerializationRedisSerializer来序列化和反序列化
redis的value值
    Jackson2JsonRedisSerializer<Object> jackson2JsonRedisSerializer = new
Jackson2JsonRedisSerializer<>(Object.class);
    ObjectMapper mapper = new ObjectMapper();
    mapper.setVisibility(PropertyAccessor.ALL, JsonAutoDetect.Visibility.ANY);
    mapper.activateDefaultTyping(LaissezFaireSubTypeValidator.instance,
        ObjectMapper.DefaultTyping.NON_FINAL, JsonTypeInfo.As.PROPERTY);
    jackson2JsonRedisSerializer.setObjectMapper(mapper);
    StringRedisSerializer stringRedisSerializer = new StringRedisSerializer();
    //key采用String的序列化方式
    template.setKeySerializer(stringRedisSerializer);
    // hash的key也采用String的序列化方式
    template.setHashKeySerializer(stringRedisSerializer);
    // value序列化方式采用jackson
    template.setValueSerializer(jackson2JsonRedisSerializer);
    // hash的value序列化方式采用jackson
    template.setHashValueSerializer(jackson2JsonRedisSerializer);
    template.afterPropertiesSet();
    return template;
}
```

说明：**host**为Redis实例的IP地址/域名，**port**为Redis实例的端口，请按实际情况修改后执行，**pwd**为创建Redis实例时自定义的密码，请按实际情况修改后执行。推荐使用连接池方式。超时时间（TimeOut），最大连接数（MaxTotal），最小空闲连接（MinIdle），最大空闲连接（MaxIdle），最大等待时间（MaxWait）等相关参数，请根据业务实际来调优。

----结束

3.2.2.3 Python Redis 客户端

介绍使用同一VPC内弹性云服务器ECS上的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**。

说明

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

前提条件

- 已成功申请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python编译环境。

操作步骤

步骤1 查看并获取待连接Redis实例的IP地址/域名和端口。

具体步骤请参见**查看实例信息**。

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。

如果系统没有自带Python，可以使用yum方式安装。

yum install python

说明

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

1. 删除python软链接文件： rm -rf python
2. 重新创建新指向python：ln -s pythonX.X.X python，其中X为python具体版本号。

- 若是单机、主备、proxy集群实例。
  a. 安装Python和Python Redis客户端redis-py。
     i. 如果系统没有自带Python，可以使用yum方式安装。
     ii. 下载并解压redis-py。
     wget https://github.com/andymccurdy/redis-py/archive/master.zip
     unzip master.zip
     iii. 进入到解压目录后安装Python Redis客户端redis-py。
     python setup.py install
     安装后执行**python**命令，返回如下信息说明成功安装redis-py：

     图 3-1 执行 python

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

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

     图 3-2 进入命令行模式

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

ii. 在命令行中执行以下命令，连接Redis实例。

```
r = redis.StrictRedis(host='XXX.XXX.XXX.XXX', port=6379, password='******');
```

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

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

图 3-3 连接 redis 成功

- 若是Cluster集群实例。
  a. 安装redis-py-cluster客户端。
     i. 执行以下命令下载released版本。

     wget https://github.com/Grokzen/redis-py-cluster/releases/download/2.1.3/redis-py-cluster-2.1.3.tar.gz

     ii. 解压压缩包。

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

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

     python setup.py install

  b. 使用redis-py-cluster客户端连接Redis实例。

     以下步骤以命令行模式进行示例（也可以将命令写入python脚本中再执行）：

     i. 执行**python**命令，进入命令行模式。

     ii. 在命令行中执行以下命令，连接Redis实例

```
>>> from rediscluster import RedisCluster

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

>>> rc = RedisCluster(startup_nodes=startup_nodes, decode_responses=True)

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

----结束

3.2.2.4 Go Redis 客户端

介绍使用同一VPC内弹性云服务器ECS上的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**。

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》。

操作步骤

步骤1 查看并获取待连接Redis实例的IP地址/域名和端口。

具体步骤请参见**查看实例信息**。

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

图 3-4 终端输入

步骤6 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/域名以及端口。IP地址/域名和端口获取见**步骤1**，请按实际情况修改后执行。********为创建Redis实例时自定义的密码，请按实际情况修改后执行。

步骤7 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8 执行**./test**连接实例。

----结束

3.2.2.5 C++Redis 客户端（ hiredis ）

介绍使用同一VPC内弹性云服务器ECS上的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**。

说明

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

前提条件

- 已成功申请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。

操作步骤

步骤1 查看并获取待连接Redis实例的IP地址/域名和端口。

具体步骤请参见**查看实例信息**。

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_ip_address}* 6379 *{password}*

   其中，*{redis_instance_address}*为Redis实例的IP地址/域名，“6379”为Redis实例的端口。IP地址/域名和端口获取见**步骤1**，请按实际情况修改后执行。
   *{password}*为创建Redis实例时自定义的密码，请按实际情况修改后执行。

   返回以下回显信息，表示成功连接Redis实例。

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

须知

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

mkdir /usr/lib/hiredis

cp /usr/local/lib/libhiredis.so.0.13 /usr/lib/hiredis/

mkdir /usr/include/hiredis

cp /usr/local/include/hiredis/hiredis.h /usr/include/hiredis/

echo '/usr/local/lib' >>;>>;/etc/ld.so.conf

ldconfig

以上so文件与.h文件的位置，需要替换成实际文件位置。

----结束

3.2.2.6 C# Redis 客户端

介绍使用同一VPC内弹性云服务器ECS上的C# 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**。

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。

操作步骤

步骤1 查看并获取待连接Redis实例的IP地址/域名和端口。

具体步骤请参见**查看实例信息**。

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。

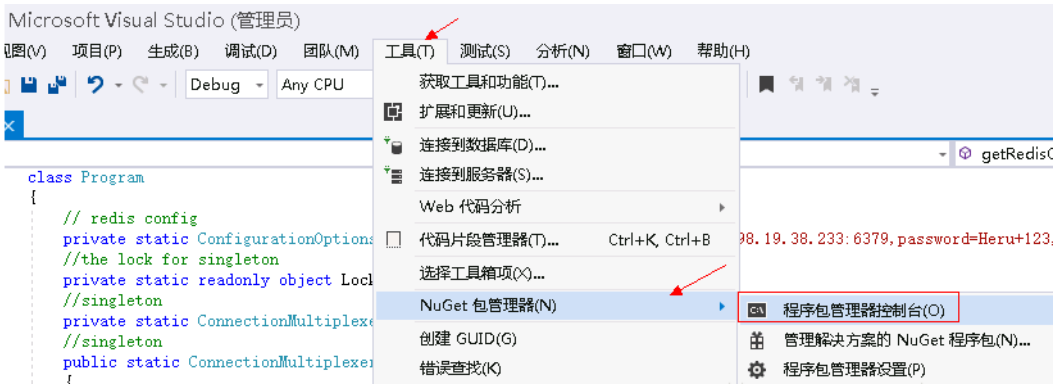
步骤4 启动VS 2017，新建一个工程。

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3-5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 3-5 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("10.10.38.233:6379,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，*10.10.38.233:6379*分别为Redis实例的IP地址/域名以及端口。IP地址/域名和端口获取见**步骤1**，请按实际情况修改后执行。*********为创建Redis实例时自定义的密码，请按实际情况修改后执行。

步骤7 运行代码，控制台界面输出如下，表示连接成功。

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

关于客户端的其他命令，可以参考**StackExchange.Redis**。

----结束

3.2.2.7 PHP 客户端

3.2.2.7.1 phpredis

介绍使用同一VPC内弹性云服务器ECS上的phpredis连接Redis的方法。更多的客户端的使用方法请参考**Redis客户端**。

说明

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

前提条件

- 已成功申请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。

操作步骤

步骤1 查看并获取待连接Redis实例的IP地址/域名和端口。

具体步骤请参见**查看实例信息**。

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 = 6379;
      $user_pwd = "{password}";
      $redis = new Redis();
      if ($redis->connect($redis_host, $redis_port) == false) {
         die($redis->getLastError());
   ```

```
    }
    if ($redis->auth($user_pwd) == false) {
        die($redis->getLastError());
    }
    if ($redis->set("welcome", "Hello, DCS for Redis!") == false) {
        die($redis->getLastError());
    }
    $value = $redis->get("welcome");
    echo $value;
    $redis->close();
?>
```

其中，*{redis_instance_address}*为Redis实例的IP地址/域名，*6379*为Redis实例的端口。IP地址/域名和端口获取见**步骤1**，请按实际情况修改后执行。*{password}*为创建Redis实例时自定义的密码，请按实际情况修改后执行。如果免密访问，请将密码认证的if语句屏蔽。

2. 执行**php redis.php**，连接Redis实例。

----结束

3.2.2.7.2 Predis

介绍使用同一VPC内弹性云服务器ECS上的Predis连接Redis的方法。更多的客户端的使用方法请参考**Redis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php编译环境。

操作步骤

步骤1 查看并获取待连接Redis实例的IP地址/域名和端口。

具体步骤请参见**查看实例信息**。

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/v1.1.10.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-1.1.10.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-1.1.10 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/域名，*{port}*为Redis实例真实的端口。IP地址/域名和端口获取见**步骤1**，请按实际情况修改后执行。*{password}*为创建Redis实例时自定义的密码，请按实际情况修改后执行。如果免密访问，请将password行去掉。

步骤7 执行**php redis.php**连接Redis实例。

----结束

3.2.2.8 Node.js Redis 客户端

介绍使用同一VPC内弹性云服务器ECS上的Node.js 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**。

说明

本章节操作，仅适用于连接单机、主备、Proxy集群实例，如果是使用Node.js Redis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。

操作步骤

- **客户端服务器为Ubuntu(debian系列)**

步骤1 查看并获取待连接Redis实例的IP地址/域名和端口。

具体步骤请参见**查看实例信息**。

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/v0.12.4/node-v0.12.4.tar.gz --no-check-certificate

tar -xvf node-v4.28.5.tar.gz

cd node-v4.28.5

./configure

make

make install

说明

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

步骤4 安装js包管理工具npm。

apt install npm

步骤5 安装NodeJs redis客户端ioredis。

npm install ioredis

步骤6 编辑连接Redis实例的示例脚本。

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

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

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

步骤7 运行示例脚本，连接Redis实例。

node ioredisdemo.js

----结束

- **客户端服务器为centos(redhat系列)**

步骤1 查看并获取待连接Redis实例的IP地址/域名和端口。

具体步骤请参见。

步骤2 登录弹性云服务器。

步骤3 安装Node.js。

yum install nodejs

如果以上命令安装不了，备选方式如下：

wget https://nodejs.org/dist/v0.12.4/node-v0.12.4.tar.gz --no-check-certificate

tar -xvf node-v0.12.4.tar.gz

cd node-v0.12.4

./configure

make

make install

说明

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

步骤4 安装js包管理工具npm。

yum install npm

步骤5 安装Node.js redis客户端ioredis。

npm install ioredis

步骤6 编辑连接Redis实例的示例脚本。

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

```
var Redis = require('ioredis');
var redis = new Redis({
  port: 6379,          // Redis port
  host: '192.168.0.196',   // Redis host
  family: 4,           // 4 (IPv4) or 6 (IPv6)
  password: '******',
  db: 0
});
redis.set('foo', 'bar');
redis.get('foo', function (err, result) {
  console.log(result);
```

```
});
// Or using a promise if the last argument isn't a function
redis.get('foo').then(function (result) {
  console.log(result);
});
// Arguments to commands are flattened, so the following are the same:
redis.sadd('set', 1, 3, 5, 7);
redis.sadd('set', [1, 3, 5, 7]);
// All arguments are passed directly to the redis server:
redis.set('key', 100, 'EX', 10);
```

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

步骤7 运行示例脚本，连接Redis实例。

node ioredisdemo.js

----结束

3.2.3 控制台连接 Redis4.0/5.0 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和Redis 5.0实例支持该功能，Redis3.0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Redis 4.0和Redis 5.0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。

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。

----结束

3.2.4 连接 Memcached 实例

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

前提条件

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

  说明

  1. 您创建的弹性云服务器必须与Memcached实例属于同一个VPC，并配置相同的安全组，以确保弹性云服务器与缓存实例的网络是连通的。
  2. 如果弹性云服务器与Memcached实例不在相同VPC中，可以通过建立VPC对等连接方式连通网络，具体请参考常见问题：**缓存实例是否支持跨VPC访问?**
  3. 如果弹性云服务器与Memcached实例配置了不同的安全组，可以通过设置安全组规则连通网络，具体请参考常见问题：**如何选择和配置安全组?**

- 建议在使用本手册时删除示例代码中的所有注释信息。
- 请确保所有命令行、代码块输入格式都是UTF-8，否则会出现编译出错或者运行失败的情况。

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

步骤5 连接Memcached实例。

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

   which telnet

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

   说明

   - Linux系统中，如果没有安装Telnet客户端，可执行**yum -y install telnet**安装。
   - 在Windows系统中，可通过“控制面板 > 程序 > 打开或关闭Windows功能”，找到并打开“Telnet客户端”功能。

3. 执行如下命令，连接并使用Memcached实例。

   telnet {ip} {port}

   其中{ip} 为Memcached实例的IP地址，{port}为Memcached实例的端口。IP地址和端口获取方法请参考**步骤4**，请按实际情况修改后执行。

   界面提示如下表示连接缓存实例成功。

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

说明

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

输入命令，示例如下（其中黑体部分内容为输入的命令，其他为命令返回内容）：

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

----结束

3.3 查看实例信息

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 单击实例名称，进入实例详情页面。

步骤5 查询DCS缓存实例。

- 支持通过关键字搜索对应的DCS缓存实例。
  直接在搜索栏输入关键字即可。
- 支持通过指定属性的关键字查询对应的DCS缓存实例。
  当前支持的属性有“名称”、“ID”、“IP地址”、“可用区”、“状态”、“实例类型”、“缓存类型”、“CPU”。

更多的搜索设置帮助，请单击搜索栏右侧的搜索帮助。

步骤6 在需要查看的DCS缓存实例左侧，单击该实例的名称，进入实例的基本信息页面。参数说明如表3-1所示。

表 3-1 参数说明

信息类型	参数	说明
基本信息	名称	DCS缓存实例的名称。单击“名称”后的 可以修改实例名称。
	状态	DCS缓存实例状态。
	ID	DCS缓存实例的ID。

信息类型	参数	说明
	缓存类型	DCS的缓存类型，包括Redis和Memcached，其中，如果是Redis，还会展示版本号，例如，Redis 3.0。
	实例类型	DCS缓存实例类型，支持“单机”、“主备”、“Proxy集群”和“Cluster集群”。
	规格（GB）	DCS缓存实例规格。
	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。只有Redis实例才会显示该字段。
	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。
	维护时间窗	运维操作时间。单击参数后的可以修改时间窗。
	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。
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。单击“安全组”后的可以修改安全组。 当前仅Redis3.0支持安全组访问控制，Redis4.0和Redis5.0版本实例是基于VPCEndpoint，暂不支持安全组，故没有该参数。
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

4 操作指南

4.1 实例日常操作

4.1.1 变更规格

DCS管理控制台支持变更Redis和Memcached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**。
- 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，请联系技术支持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。

实例类型变更前须知

- **支持实例类型变更明细如下：**
  - Redis3.0和Memcached单机实例支持变更为主备实例，Redis4.0和Redis5.0单机实例不支持变更。
  - Redis3.0主备实例支持变更为Proxy集群实例，Redis4.0和Redis5.0主备实例暂不支持变更。

    如果Redis3.0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：**
  - Redis3.0单机实例类型变更为Redis3.0主备实例。

    连接会有秒级中断，大约1分钟左右的只读。
  - Redis3.0主备实例类型变更为Redis3.0 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 4-1 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
Redis3.0	支持扩容和缩容	支持扩容和缩容	不涉及	仅支持扩容
Redis4.0	支持扩容和缩容	支持扩容和缩容	支持扩容和缩容	不涉及
Redis5.0	支持扩容和缩容	支持扩容和缩容	支持扩容和缩容	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

  Redis3.0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。

- **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 4.0/5.0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。如果有重要数据，建议先把数据用迁移工具迁移到其他实例备份。
    - 主备实例的备份记录，缩容后不能恢复。
  - 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。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。

- 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，否则可能会导致缩容失败。
- 变更规格前的备份记录不能恢复。

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，单击“下一步”。

步骤6 单击“提交”，开始变更DCS缓存实例。

界面将会自动跳转到后台任务管理界面，您可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

4.1.2 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。
- 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。

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1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

- 默认情况只会重启实例进程。如果勾选“强制重启”，Redis 3.0、Memcached实例将会重启虚拟机，Redis 4.0及以上版本实例不支持强制重启，将会重启实例进程。
- 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

4.1.3 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。

须知

- DCS缓存实例删除后，实例中原有的数据将被删除，且没有备份，请谨慎操作。同时，实例的备份数据也会删除，在删除之前，您可以将实例的备份文件下载，本地永久保存。
- 如果是集群实例，会将实例的所有节点删除。

前提条件

- DCS缓存实例已存在。
- DCS缓存实例状态为运行中、故障时才能执行删除操作。

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。

DCS缓存实例状态为创建中、重启中、升级中、规格变更中、清空数据中、备份中、恢复中时不允许执行删除操作。

步骤5 单击信息栏左上侧的“删除”。

步骤6 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

4.1.4 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。

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”，完成主备切换。

----结束

4.1.5 清空实例数据

只有Redis版本为4.0和5.0的实例支持清空DCS实例数据功能。

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。

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

4.1.6 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 单击搜索栏弹出筛选条件，设置筛选条件，查询对应的DCS缓存实例。

步骤5 单击信息栏右上侧的 按钮，导出缓存实例列表。

页面左下角显示导出的结果，单击可查看导出的实例信息。

----结束

4.1.7 命令重命名

Redis4.0和Redis5.0实例创建之后，支持重命名高危命令。当前支持的高危命令有command、keys、flushdb、flushall和hgetall，其他命令暂时不支持重命名。

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”。

说明

- 您可以一次为多个高危命令进行重命名。
- 命令重命名后，需要重启实例才能生效。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。
- 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。
- 重命名的长度不能小于四位。

----结束

4.2 实例配置管理

4.2.1 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。

4.2.2 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”。

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4-2和表4-3，一般情况下，按照系统默认值设置参数即可。

表 4-2 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
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。	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。	• no • always • everysec	everysec
appendonly	指定是否在每次更新操作后进行日志记录，Redis在默认情况下是异步的把数据写入磁盘，如果不开启，可能会在断电时导致一段时间内的数据丢失。有2个取值供选择： yes：开启。 no：关闭。	• 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（单位为秒），服务端会主动断开连接。	0~60	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。	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。	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，单位为毫秒。	100～5,000	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。	• yes • no	no
maxclients	最大同时连接的客户端个数。	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，单位为字节。	1,048,576～536,870,912	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（单位为字节）。这是一个用来在从节点断开连接时，存放从节点数据的缓冲区，当从节点重新连接时，如果丢失的数据少于缓冲区的大小，可以用缓冲区中的数据开始增量同步。	16,384～1,073,741,824	1,048,576
repl-backlog-ttl	从节点断开后，主节点释放复制积压缓冲区内存的秒数。值为0时表示永不释放复制积压缓冲区内存。	0～604,800	3,600
repl-timeout	主从同步超时时间，单位为秒。	30～3,600	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阀值（最小值），单位为毫秒。 阀值设置为0：不做监控，也不采样； 阀值设置为大于0：，将记录执行耗时大于阀值的操作。 可以通过LATENCY等命令获取统计数据和配置、执行采样监控。	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。	请参考该参数的描述。	Ex
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，单位为微秒。	0～1,000,000	10,000

参数名	参数解释	取值范围	默认值
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4-2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。
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。

表 4-3 Memcached 缓存实例配置参数说明

参数名	参数解释	取值范围	默认值
timeout	客户端与服务端连接空闲超时断开时间，参数设为0表示连接永不断开。	0~7200，单位：秒。	0
maxclients	最大同时连接的客户端个数。	1,000～10,000	10,000
maxmemory-policy	内存使用达到上限时对缓存数据管理策略。 参数说明请参考https://redis.io/topics/lru-cache。	volatile-lru allkeys-lru volatile-random allkeys-random volatile-ttl noeviction	noeviction
reserved-memory-percent	预留给后台进程用于持久化、主从同步等内部处理的内存百分比（相对于最大可用内存的百分比）。	0-80	30

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

目前只有主备和Cluster集群实例可通过控制台修改appendonly、appendfsync参数配置。

4.2.3 修改实例维护时间窗

DCS缓存实例创建后，若需要修改实例维护时间窗，可进入管理控制台的实例基本信息页面进行修改。

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 单击需要修改的DCS缓存实例名称。

步骤5 在打开的DCS缓存实例基本信息页面，单击“维护时间窗”后的 。

步骤6 下拉选择新的维护时间窗。单击 保存修改，单击 取消修改。

修改操作立即生效，可在实例对应的“概览”页面查看修改结果。

----结束

4.2.4 修改实例安全组

DCS缓存实例创建后，若需要修改实例安全组，可进入管理控制台的实例基本信息页面进行修改。

当前仅Redis3.0缓存实例可以修改安全组，Redis4.0和Redis5.0实例不支持安全组，默认放通所有安全组。

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 单击需要修改的DCS缓存实例名称。

步骤5 在打开的DCS缓存实例基本信息页面，单击“安全组”后的。

步骤6 下拉选择新的安全组。单击保存修改，单击取消修改。

说明

此处只能下拉选择已创建的安全组，若需要重新配置安全组，可参考**安全组配置和选择**。

修改操作立即生效，可在实例对应的“概览”页面查看修改结果。

----结束

4.2.5 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”页签，进入后台任务管理页面。

界面显示任务列表。

步骤6 单击，选择“开始日期”和“结束日期”，单击“确认”，界面显示相应时间段内启动的任务。

- 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

4.2.6 查看 Redis 3.0 Proxy 集群实例的数据存储统计信息

您需要查看Redis 3.0 Proxy集群内各个存储节点的数据存储统计信息，当集群中各存储节点的数据存储分布不均匀时，您可对实例进行扩容或者清理数据。

当前仅Redis 3.0 Proxy集群实例支持查看数据存储统计信息，其他实例类型如主备只有一个存储节点，可在实例的基本信息中的已用内存查看，所以不支持。

说明

Cluster集群实例不止有一个存储节点，查看其数据存储统计信息可通过监控中的数据节点监控。

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 选择Redis集群实例，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“节点管理”页签。

界面显示集群实例中各个节点的数据量信息。

当集群的节点数据存储容量满时，需要对实例进行扩容，具体扩容操作，请参考**变更规格**。

----结束

4.2.7 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。

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10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4-4**所示。

表 4-4 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。 ● 长度不超过36个字符。 ● 不能包含“=”，“*”，“<”，“>”，“\”，“,”，“|”，“/”。 ● 首尾字符不能为空格。
标签值	● 长度不超过43个字符。 ● 不能包含“=”，“*”，“<”，“>”，“\”，“,”，“|”，“/”。 ● 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“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“添加标签”，弹出“添加标签”对话框。

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

     您也可以直接在“标签键”和“标签值”中输入设置标签。

  b. 单击“确定”。为实例添加标签成功。
- 修改标签

  单击标签所在行“操作”列下的“编辑”，在弹出的“编辑标签”窗口，输入修改后标签的值，并单击“确定”。
- 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

4.2.8 管理分片与副本

本节主要介绍如何查询Redis 4.0/5.0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 4.0/5.0的主备、集群实例支持该功能，Redis 4.0/5.0单机实例和Redis 3.0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，每个集群是由多个分片组成，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**Redis4.0/5.0 Cluster集群介绍** 。

升级副本

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。

步骤7 选择角色为从的副本，单击“升级为主”。

步骤8 单击“是”，将选择的副本升级为主。

----结束

4.2.9 缓存分析

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），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Redis4.0/Redis5.0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“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。

----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“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

如果是新创建的Redis4.0/Redis5.0实例，maxmemory-policy默认为noeviction，您需要先将参数配置为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 4-5 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。
DataBase	热Key所在的DB。

----结束

4.2.10 管理实例白名单

由于Redis3.0/Memcached和Redis4.0/Redis5.0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：

- Redis3.0/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，具体请参考**安全组配置和选择**。
- Redis4.0/Redis5.0：不支持安全组，只支持通过白名单控制。

本章节主要介绍如何管理Redis4.0/Redis5.0实例白名单，如果需要指定的IP地址才能访问Redis4.0、Redis5.0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 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。

说明

此处请选择与您的应用服务相同的区域。

步骤3 单击左侧菜单栏的“缓存管理”，进入实例信息页面。

步骤4 单击需要创建白名单的DCS缓存实例名称，进入该实例的基本信息页面。

步骤5 单击“白名单”页签，然后单击“创建白名单分组”。

步骤6 在弹出的“创建白名单分组”页面，设置“分组名”和“IP地址/地址段”。

表 4-6 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
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7 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

4.2.11 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**。

说明

如果是Redis3.0 Proxy集群实例，必须是2019年10月14号后创建的才支持慢查询。如果是之前的实例，可提工单申请升级。升级对业务无任何影响，只是控制台新增支持慢查询功能。

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“慢查询”。

步骤6 设置查询时间，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。

----结束

4.2.12 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。

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”页签。

步骤6 单击“新增日志文件”，配置日志采集信息。

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。

----结束

4.2.13 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

- Redis3.0、Memcached实例不支持实例诊断。

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“实例诊断”页签。

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。

步骤7 诊断完成后，在诊断记录列表中可以查看诊断结果中，如果出现异常，单击“查看报告”，查看具体异常的诊断项。

在异常的诊断项中，您可以查看产生异常的原因、异常的影响，以及处理异常的建议。

----结束

4.3 实例备份恢复管理

4.3.1 备份与恢复说明

介绍如何通过管理控制台对DCS缓存实例进行数据备份，以及备份数据恢复。

备份缓存数据的必要性

业务系统日常运行中可能出现一些小概率的异常事件，比如异常导致缓存实例出现大量脏数据，或者在实例出现故障后持久化文件不能重新加载。部分可靠性要求非常高的业务系统，除了要求缓存实例高可用，还要求缓存数据安全、可恢复，甚至永久保存。

DCS支持将当前时间点的实例缓存数据备份并存储到对象存储服务（OBS）中，以便在缓存实例发生异常后能够使用备份数据进行恢复，保障业务正常运行。

备份方式

DCS缓存实例支持自动和手动两种备份方式。

- 自动备份

  您可以通过管理控制台设置一个定时自动备份策略，在指定时间点将实例的缓存数据自动备份存储。

  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，并永久性存储到对象存储服务（OBS）中（您可根据需要手动删除备份数据）。

  您在执行业务系统维护、升级等高危操作前，可以先行备份实例缓存数据。

备份的其他说明

- 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”Memcached实例支持数据备份与恢复功能，“单机”Memcached实例暂不支持。
- 备份原理

  实例采用Redis的AOF方式进行持久化。

  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。

- 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。

- 备份文件的存储与收费

  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。

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。

4.3.2 设置备份策略

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。

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

已成功申请主备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 4-7 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。

----结束

4.3.3 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缓存实例的数据。

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 单击“确定”，开始执行手工备份任务。

备注说明最长不能超过128个字节。

说明

实例备份需耗时10~15分钟，备份期间发生的数据新增或修改记录，将不会保存到备份数据中。

----结束

4.3.4 实例恢复

您可以将已备份数据恢复到DCS缓存实例中。

前提条件

- 已成功申请主备或集群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5~30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

4.3.5 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。

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。

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

4.4 使用 DCS 迁移数据

4.4.1 使用 DCS 迁移介绍

迁移概览

DCS Redis支持备份文件导入和在线迁移两种迁移方式：

- 备份文件导入方式，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。

说明

数据迁移功能，适用于用户直接在DCS控制台创建的DCS实例及自建Redis，如果用户执行数据迁移后，需要将客户端连接DCS实例的访问地址修改为目标实例的访问地址并重启客户端。

如果是通过服务依赖创建的DCS实例（例如，Roma Connect服务创建实例时，通过接口创建的DCS实例），不支持数据迁移。

表 4-8 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	对象存储服务（OBS）：AOF文件 **说明** Redis 4.0/5.0实例和其他开启RDB压缩实例导出的AOF文件都不支持导入。	√	√	×
	对象存储服务（OBS）：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

- **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。

4.4.2 备份文件导入方式

4.4.2.1 备份文件导入方式-OBS 桶

场景描述

当前DCS支持将其他云厂商Redis、自建Redis的数据通过DCS控制台迁移到DCS Redis。

您需要先将其他云厂商Redis、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 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。

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。
- 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。
- 暂不支持导入自建Redis5.0生成的rdb备份文件，如果是自建Redis3.0和Redis4.0，可以使用Redis-cli工具导出.rdb备份文件。其他云厂商Redis只能通过各云的备份页面创建备份任务导出获取，不能通过Redis-cli工具使用命令导出。

- Cluster集群仅支持导入.rdb备份文件，不支持.aof备份文件。

步骤 1：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，但在迁移之前，您需要清空实例数据。
  - 目标实例为Redis4.0及5.0时，清空操作请参考**清空Redis实例数据**。
  - 目标实例为Redis3.0时，执行**flushall**命令进行清空数据。

当前支持迁移到Redis3.0、Redis4.0和Redis5.0，您可以根据实际情况选择。

步骤 2：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。
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. 单次最多支持100个文件同时上传，总大小不超过5GB。

   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。
5. 单击“上传”。

----结束

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 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。

步骤7 在“备份文件”中选择需要迁移的备份文件。

步骤8 在目标实例区域，选择**步骤1：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。

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

4.4.2.2 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。

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，但在迁移之前，您需要清空实例数据。
  - 若目标实例为Redis4.0及5.0，清空操作请参考**清空Redis实例数据**。
  - 若目标实例为Redis3.0，请执行**flushall**命令进行数据清空。

当前支持迁移到Redis3.0、Redis4.0和Redis5.0，您可以根据实际情况选择。

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 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”，测试密码是否符合要求。

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

4.4.3 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。

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Redis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**。

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。

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**。

在线迁移任务与源 Redis、目标 Redis 间网络要求

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4-9**。

表 4-9 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

源Redis类型	目标Redis类型	创建在线迁移任务网络要求
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

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 配置虚拟私有云及安全组。

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4-10所示。

表 4-10 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。

步骤3 分别选择源Redis和目标Redis。

1. “源Redis”，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。
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。

步骤4 在“目标Redis实例”中，选择**准备目标Redis实例**中创建的目标实例。

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”，检查实例密码是否符合要求。

说明

当源Redis和目标Redis属于DCS不同Region，则打通网路后，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。

步骤5 单击“立即创建”。

步骤6 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。

----结束

4.5 密码管理

4.5.1 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。

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，该实例帐户将被锁定5分钟，锁定期间不允许修改密码。

DCS帐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。

4.5.2 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，该实例帐户将被锁定5分钟，锁定期间不允许修改密码。

DCS帐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

4.5.3 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、Memcached实例均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**、**修改Memcached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。

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帐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

4.5.4 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：

- 对于免密访问模式的Redis实例，当需要开启公网访问时，必须要先将实例访问方式由免密访问修改为密码访问模式，然后再开启公网访问。
- 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。
- 为保护Redis实例的网络安全性，已开启公网访问的Redis实例不支持同时开启免密访问。

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

4.5.5 修改 Memcached 实例的访问方式

使用场景

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

- 对于密码方式模式的Memcached实例，当您需要通过免用户名和密码访问模式连接Memcached实例，可通过开启Memcached实例的免密访问功能，清空Memcached实例的用户名和密码。

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

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 在需要修改访问方式的Memcached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

5 监控

5.1 支持的监控指标

功能说明

本节定义了DCS服务上报云监控服务的监控指标的命名空间，监控指标列表和维度定义，用户可以通过云监控服务提供管理控制台或API接口来检索DCS服务产生的监控指标和告警信息。

实例监控指标差异如下：

- 单机实例监控指标

  如果是单机实例，只有实例级别的监控指标，实例监控即为数据节点监控。

- 主备实例监控指标

  支持实例监控和数据节点监控。实例监控是指对主节点的监控，数据节点监控分别是对主节点和备节点的监控。

- 集群实例监控指标

  如果是Proxy集群，支持实例监控、数据节点监控、Proxy节点监控。实例监控是对主节点数据汇总后的监控，数据节点监控是对集群每个分片的监控，Proxy节点监控是对集群每个Proxy节点的监控。

  如果是Cluster集群，支持实例监控和数据节点监控。实例监控是对集群所有主节点数据汇总后的监控，数据节点监控是对集群每个分片的监控。

命名空间

SYS.DCS

Redis3.0 实例监控指标

说明

测量对象&维度列，包含支持该指标的实例和实例类型，以及维度ID。

表 5-1 Redis3.0 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）
cpu_usage	CPU利用率	该指标对于统计周期内的测量对象的CPU使用率进行多次采样，表示多次采样的最高值。 单位：%。	0-100 %	测量对象： Redis实例（单机/主备/集群） 测量维度： dcs_instance_id	1分钟
memory_usage	内存利用率	该指标用于统计测量对象的内存利用率。 单位：%。	0-100 %	测量对象： Redis实例（单机/主备/集群） 测量维度： dcs_instance_id	1分钟
net_in_throughput	网络输入吞吐量	该指标用于统计网口平均每秒的输入流量。 单位：byte/s。	>= 0字节/秒	测量对象： Redis实例（单机/主备/集群） 测量维度： dcs_instance_id	1分钟
net_out_throughput	网络输出吞吐量	该指标用于统计网口平均每秒的输出流量。 单位：byte/s。	>= 0字节/秒	测量对象： Redis实例（单机/主备/集群） 测量维度： dcs_instance_id	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	测量对象： Redis实例（单机/主备/集群） 测量维度： dcs_instance_id	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	测量对象： Redis实例（单机/主备/集群） 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）
client_biggest_in_buf	客户端最大输入缓冲	该指标用于统计客户端所有现存连接的最大输入数据长度。 单位：byte。	>=0byte	测量对象： Redis实例（单机/主备/集群） 测量维度： dcs_instance_id	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	测量对象： Redis实例（单机/主备/集群） 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：byte。	>=0byte	测量对象： Redis实例（单机/主备/集群） 测量维度： dcs_instance_id	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。 单位：byte。	>=0byte	测量对象： Redis实例（单机/主备/集群） 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：byte。	>=0byte	测量对象： Redis实例（单机/主备/集群） 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：byte。	>=0byte	测量对象： Redis实例（单机/主备/集群） 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）
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（单机/主备/集群） 测量维度： dcs_instance_id	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	测量对象： Redis实例（单机/主备/集群） 测量维度： dcs_instance_id	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	测量对象： Redis实例（单机/主备/集群） 测量维度： dcs_instance_id	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	测量对象： Redis实例（单机/主备/集群） 测量维度： dcs_instance_id	1分钟
total_net_input_bytes	网络收到字节数	该指标用于统计周期内收到的字节数。 单位：byte。	>=0byte	测量对象： Redis实例（单机/主备/集群） 测量维度： dcs_instance_id	1分钟
total_net_output_bytes	网络发送字节数	该指标用于统计周期内发送的字节数。 单位：byte。	>=0byte	测量对象： Redis实例（单机/主备/集群） 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）
instantaneous_input_kbps	网络瞬时输入流量	该指标用于统计瞬时的输入流量。 单位：kbit/s。	>=0kbit/s	测量对象： Redis实例（单机/主备/集群） 测量维度： dcs_instance_id	1分钟
instantaneous_output_kbps	网络瞬时输出流量	该指标用于统计瞬时的输出流量。 单位：kbit/s。	>=0kbit/s	测量对象： Redis实例（单机/主备/集群） 测量维度： dcs_instance_id	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	测量对象： Redis实例（单机/主备/集群） 测量维度： dcs_instance_id	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	测量对象： Redis实例（单机/主备/集群） 测量维度： dcs_instance_id	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	测量对象： Redis实例（单机/主备/集群） 测量维度： dcs_instance_id	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	测量对象： Redis实例（单机/主备/集群） 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）
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	测量对象： Redis实例（单机/主备/集群） 测量维度： dcs_instance_id	1分钟
pubsub_channels	Pubsub通道个数	该指标用于统计Pub/Sub通道个数。	>=0	测量对象： Redis实例（单机/主备/集群） 测量维度： dcs_instance_id	1分钟
pubsub_patterns	Pubsub模式个数	该指标用于统计Pub/Sub模式个数。	>=0	测量对象： Redis实例（单机/主备/集群） 测量维度： dcs_instance_id	1分钟
keyspace_hits_perc	缓存命中率	该指标用于统计Redis的缓存命中率，其命中率算法为：keyspace_hits/(keyspace_hits+keyspace_misses) 单位：%。	0-100%	测量对象： Redis实例（单机/主备/集群） 测量维度： dcs_instance_id	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	测量对象： Redis实例（单机/主备/集群） 测量维度： dcs_instance_id	1分钟
auth_errors	认证失败次数	统计实例的认证失败次数。	>=0	测量对象： Redis实例（单机/主备） 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）
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。	测量对象： Redis实例（单机/主备） 测量维度： dcs_instance_id	1分钟
keys	缓存键总数	该指标用于统计Redis缓存中键总数。	>=0	测量对象： Redis实例（单机/主备） 测量维度： dcs_instance_id	1分钟

Redis4.0 和 Redis5.0 实例监控指标

说明

测量对象&维度列，表示支持该指标的实例和实例类型，以及维度ID。

表 5-2 Redis4.0 和 Redis5.0 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）
cpu_usage	CPU利用率	该指标对于统计周期内的测量对象的CPU使用率进行多次采样，表示多次采样的最高值。 单位：%。	0-100%	测量对象： Redis实例（单机/主备） 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）
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	测量对象： Redis实例（单机/主备/集群） 测量维度： dcs_instance_id	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	测量对象： Redis实例（单机/主备/集群） 测量维度： dcs_instance_id	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。	• 1：表示存在 • 0：表示不存在。	测量对象： Redis实例（单机/主备/集群） 测量维度： dcs_instance_id	1分钟
memory_usage	内存利用率	该指标用于统计测量对象的内存利用率。 单位：%。	0-100%	测量对象： Redis实例（单机/主备/集群） 测量维度： dcs_instance_id	1分钟
expires	有过期时间的键总数	该指标用于统计Redis缓存中将会过期失效的键数目。	>=0	测量对象： Redis实例（单机/主备/集群） 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）
keyspace_hits_perc	缓存命中率	该指标用于统计Redis的缓存命中率，其命中率算法为：keyspace_hits/(keyspace_hits +keyspace_misses) 单位：%。	0-100%	测量对象： Redis实例（单机/主备/集群） 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：byte。	>= 0byte	测量对象： Redis实例（单机/主备/集群） 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：byte。	>= 0byte	测量对象： Redis实例（单机/主备/集群） 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中数据集使用的内存所占总内存百分比 单位：%。	0-100%	测量对象： Redis实例（单机/主备/集群） 测量维度： dcs_instance_id	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。 单位：byte。	>= 0byte	测量对象： Redis实例（单机/主备/集群） 测量维度： dcs_instance_id	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	测量对象： Redis实例（单机/主备/集群） 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）
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	测量对象： Redis实例（单机/主备/集群） 测量维度： dcs_instance_id	1分钟
keys	缓存键总数	该指标用于统计Redis缓存中键总数。	>=0	测量对象： Redis实例（单机/主备/集群） 测量维度： dcs_instance_id	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	测量对象： Redis实例（单机/主备/集群） 测量维度： dcs_instance_id	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	测量对象： Redis实例（单机/主备/集群） 测量维度： dcs_instance_id	1分钟
del	DEL	该指标用于统计平均每秒del操作数。 单位：Count/s	0-500000 Count/s	测量对象： Redis实例（单机/主备/集群） 测量维度： dcs_instance_id	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	测量对象： Redis实例（单机/主备/集群） 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）
expire	EXPIRE	该指标用于统计平均每秒expire操作数。 单位：Count/s	0-500000 Count/s	测量对象： Redis实例（单机/主备/集群） 测量维度： dcs_instance_id	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	测量对象： Redis实例（单机/主备/集群） 测量维度： dcs_instance_id	1分钟
get	GET	该指标用于统计平均每秒get操作数。 单位：Count/s	0-500000 Count/s	测量对象： Redis实例（单机/主备/集群） 测量维度： dcs_instance_id	1分钟
hdel	HDEL	该指标用于统计平均每秒hdel操作数。 单位：Count/s	0-500000 Count/s	测量对象： Redis实例（单机/主备/集群） 测量维度： dcs_instance_id	1分钟
hget	HGET	该指标用于统计平均每秒hget操作数。 单位：Count/s	0-500000 Count/s	测量对象： Redis实例（单机/主备/集群） 测量维度： dcs_instance_id	1分钟
hmget	HMGET	该指标用于统计平均每秒hmget操作数。 单位：Count/s	0-500000 Count/s	测量对象： Redis实例（单机/主备/集群） 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）
hmset	HMSET	该指标用于统计平均每秒hmset操作数。 单位：Count/s	0-500000 Count/s	测量对象： Redis实例（单机/主备/集群） 测量维度： dcs_instance_id	1分钟
hset	HSET	该指标用于统计平均每秒hset操作数。 单位：Count/s	0-500000 Count/s	测量对象： Redis实例（单机/主备/集群） 测量维度： dcs_instance_id	1分钟
instantaneous_input_kbps	网络瞬时输入流量	该指标用于统计瞬时的输入流量。 单位：KB/s。	>=0KB/s	测量对象： Redis实例（单机/主备/集群） 测量维度： dcs_instance_id	1分钟
instantaneous_output_kbps	网络瞬时输出流量	该指标用于统计瞬时的输出流量。 单位：KB/s。	>=0KB/s	测量对象： Redis实例（单机/主备/集群） 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	>= 0	测量对象： Redis实例（单机/主备/集群） 测量维度： dcs_instance_id	1分钟
mget	MGET	该指标用于统计平均每秒mget操作数。 单位：Count/s	0-500000 Count/s	测量对象： Redis实例（单机/主备/集群） 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）
mset	MSET	该指标用于统计平均每秒mset操作数。 单位：Count/s	0-500000 Count/s	测量对象： Redis实例（单机/主备/集群） 测量维度： dcs_instance_id	1分钟
pubsub_channels	Pubsub通道个数	该指标用于统计Pub/Sub通道个数	>= 0	测量对象： Redis实例（单机/主备/集群） 测量维度： dcs_instance_id	1分钟
pubsub_patterns	Pubsub模式个数	该指标用于统计Pub/Sub模式个数	>= 0	测量对象： Redis实例（单机/主备/集群） 测量维度： dcs_instance_id	1分钟
set	SET	该指标用于统计平均每秒set操作数。 单位：Count/s	0-500000 Count/s	测量对象： Redis实例（单机/主备/集群） 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：byte	>= 0byte	测量对象： Redis实例（单机/主备/集群） 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：byte	>= 0byte	测量对象： Redis实例（单机/主备/集群） 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）
sadd	Sadd	该指标用于统计平均每秒sadd操作数。 单位：Count/s	0-500000 Count/s	测量对象： Redis实例（单机/主备/集群） 测量维度： dcs_instance_id	1分钟
smembers	Smembers	该指标用于统计平均每秒smembers操作数。 单位：Count/s	0-500000 Count/s	测量对象： Redis实例（单机/主备/集群） 测量维度： dcs_instance_id	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。 单位：Count。	>=0	测量对象： Redis实例（Cluster集群） 测量维度： dcs_instance_id	1分钟
bandwidth_usage	带宽使用率	计算当前流量带宽与最大带宽限制的百分比。	0-200%	测量对象： Redis实例（Cluster集群） 测量维度： dcs_instance_id	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	测量对象： Redis实例（单机/主备/集群） 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。	>=0	测量对象： Redis实例（单机/主备/集群） 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_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%	测量对象： Redis实例（单机/主备/集群） 测量维度： dcs_instance_id	1分钟

Redis 实例节点监控指标

说明

- 本小节主要介绍集群实例中数据节点和Proxy节点的监控指标，其中，Redis3.0 Proxy集群包括数据节点和Proxy节点的监控指标，Redis4.0和Redis5.0 Cluster集群只包括数据节点监控指标。具体监控指标，请查看表5-3和表5-4。
- **测量对象&维度**列，表示支持该指标的实例和实例类型，以及维度ID。

表 5-3 实例中数据节点监控指标

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）
cpu_usage	CPU利用率	该指标对于统计周期内的测量对象的CPU使用率进行多次采样，表示多次采样的最高值。 单位：%。	0-100%	测量对象： Redis3.0/ Redis4.0/ Redis5.0集群实例数据节点 测量维度： dcs_instance_id dcs_cluster_redis_node	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）
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： Redis3.0/ Redis4.0/ Redis5.0集群实例数据节点 测量维度： dcs_cluster_redis_node	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	测量对象： Redis3.0/ Redis4.0/ Redis5.0集群实例数据节点 测量维度： dcs_instance_id dcs_cluster_redis_node	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	测量对象： Redis3.0/ Redis4.0/ Redis5.0集群实例数据节点 测量维度： dcs_instance_id dcs_cluster_redis_node	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）
client_biggest_in_buf	客户端最大输入缓冲	该指标用于统计客户端所有现存连接的最大输入数据长度。 单位：byte。	>=0byte	测量对象： Redis3.0/Redis4.0/Redis5.0集群实例数据节点 测量维度： dcs_instance_id dcs_cluster_redis_node	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	测量对象： Redis3.0/Redis4.0/Redis5.0集群实例数据节点 测量维度： dcs_instance_id dcs_cluster_redis_node	1分钟
used_memory	已用内存	该指标用于统计Redis已使用的内存字节数。 单位：byte。	>=0byte	测量对象： Redis3.0/Redis4.0/Redis5.0集群实例数据节点 测量维度： dcs_instance_id dcs_cluster_redis_node	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，包含和堆，但不包括换出的内存。 单位：byte。	>=0byte	测量对象： Redis3.0/Redis4.0/Redis5.0集群实例数据节点 测量维度： dcs_instance_id dcs_cluster_redis_node	1分钟
used_memory_peak	已用内存峰值	该指标用于统计Redis服务器启动以来使用内存的峰值。 单位：byte。	>=0byte	测量对象： Redis3.0/Redis4.0/Redis5.0集群实例数据节点 测量维度： dcs_instance_id dcs_cluster_redis_node	1分钟
used_memory_lua	Lua已用内存	该指标用于统计Lua引擎已使用的内存字节。 单位：byte。	>=0byte	测量对象： Redis3.0/Redis4.0/Redis5.0集群实例数据节点 测量维度： dcs_instance_id dcs_cluster_redis_node	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）
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： Redis3.0/ Redis4.0/ Redis5.0集群实例数据节点 测量维度： dcs_instance_id dcs_cluster_redis_node	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	测量对象： Redis3.0/ Redis4.0/ Redis5.0集群实例数据节点 测量维度： dcs_instance_id dcs_cluster_redis_node	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	测量对象： Redis3.0/ Redis4.0/ Redis5.0集群实例数据节点 测量维度： dcs_instance_id dcs_cluster_redis_node	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）
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	测量对象： Redis3.0/Redis4.0/Redis5.0集群实例数据节点 测量维度： dcs_instance_id dcs_cluster_redis_node	1分钟
total_net_input_bytes	网络收到字节数	该指标用于统计周期内收到的字节数。 单位：byte。	>=0byte	测量对象： Redis3.0/Redis4.0/Redis5.0集群实例数据节点 测量维度： dcs_instance_id dcs_cluster_redis_node	1分钟
total_net_output_bytes	网络发送字节数	该指标用于统计周期内发送的字节数。 单位：byte。	>=0byte	测量对象： Redis3.0/Redis4.0/Redis5.0集群实例数据节点 测量维度： dcs_instance_id dcs_cluster_redis_node	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）
instantaneous_input_kbps	网络瞬时输入流量	该指标用于统计瞬时的输入流量。 单位：KB/s。	>=0KB/s	测量对象： Redis3.0/Redis4.0/Redis5.0集群实例数据节点 测量维度： dcs_instance_id dcs_cluster_redis_node	1分钟
instantaneous_output_kbps	网络瞬时输出流量	该指标用于统计瞬时的输出流量。 单位：KB/s。	>=0KB/s	测量对象： Redis3.0/Redis4.0/Redis5.0集群实例数据节点 测量维度： dcs_instance_id dcs_cluster_redis_node	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	测量对象： Redis3.0/Redis4.0/Redis5.0集群实例数据节点 测量维度： dcs_instance_id dcs_cluster_redis_node	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）
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	测量对象： Redis3.0/Redis4.0/Redis5.0集群实例数据节点 测量维度： dcs_instance_id dcs_cluster_redis_node	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	测量对象： Redis3.0/Redis4.0/Redis5.0集群实例数据节点 测量维度： dcs_instance_id dcs_cluster_redis_node	1分钟
pubsub_channels	Pubsub通道个数	该指标用于统计Pub/Sub通道个数。	>=0	测量对象： Redis3.0/Redis4.0/Redis5.0集群实例数据节点 测量维度： dcs_instance_id dcs_cluster_redis_node	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）
pubsub_patterns	Pubsub模式个数	该指标用于统计Pub/Sub模式个数。	>=0	测量对象： Redis3.0/Redis4.0/Redis5.0集群实例数据节点 测量维度： dcs_instance_id dcs_cluster_redis_node	1分钟
keyspace_hits_perc	缓存命中率	该指标用于统计Redis的缓存命中率，其命中率算法为：keyspace_hits/(keyspace_hits+keyspace_misses) 单位：%。	0-100%	测量对象： Redis3.0/Redis4.0/Redis5.0集群实例数据节点 测量维度： dcs_instance_id dcs_cluster_redis_node	1分钟
command_max_delay	命令最大时延	统计节点的命令最大时延。 单位：ms。	>=0ms	测量对象： Redis3.0/Redis4.0/Redis5.0集群实例数据节点 测量维度： dcs_instance_id dcs_cluster_redis_node	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）
is_slow_log_exist	是否存在慢日志	统计节点是否存在慢日志。	• 1：表示存在 • 0：表示不存在。	测量对象： Redis3.0/Redis4.0/Redis5.0集群实例数据节点 测量维度： dcs_instance_id dcs_cluster_redis_node	1分钟
keys	缓存键总数	该指标用于统计Redis缓存中键总数。	>=0	测量对象： Redis3.0/Redis4.0/Redis5.0集群实例数据节点 测量维度： dcs_instance_id dcs_cluster_redis_node	1分钟
sadd	Sadd	该指标用于统计平均每秒sadd操作数。 单位：Count/s	0-500000 Count/s	测量对象： Redis3.0/Redis4.0/Redis5.0集群实例数据节点 测量维度： dcs_instance_id dcs_cluster_redis_node	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）
smembers	Smembers	该指标用于统计平均每秒smembers操作数。 单位：Count/s	0-500000 Count/s	测量对象： Redis4.0/Redis5.0集群实例数据节点 测量维度： dcs_instance_id dcs_cluster_redis_node	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。	-	测量对象： Redis4.0/Redis5.0集群实例数据节点的**备节点** 测量维度： dcs_instance_id dcs_cluster_redis_node	1分钟
del	DEL	该指标用于统计平均每秒del操作数。 单位：Count/s	0-500000 Count/s	测量对象： Redis4.0/Redis5.0集群实例数据节点 测量维度： dcs_instance_id dcs_cluster_redis_node	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）
expire	EXPIRE	该指标用于统计平均每秒expire操作数。 单位：Count/s	0-500000 Count/s	测量对象： Redis4.0/Redis5.0集群实例数据节点 测量维度： dcs_instance_id dcs_cluster_redis_node	1分钟
get	GET	该指标用于统计平均每秒get操作数。 单位：Count/s	0-500000 Count/s	测量对象： Redis4.0/Redis5.0集群实例数据节点 测量维度： dcs_instance_id dcs_cluster_redis_node	1分钟
hdel	HDEL	该指标用于统计平均每秒hdel操作数。 单位：Count/s	0-500000 Count/s	测量对象： Redis4.0/Redis5.0集群实例数据节点 测量维度： dcs_instance_id dcs_cluster_redis_node	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）
hget	HGET	该指标用于统计平均每秒hget操作数。 单位：Count/s	0-500000 Count/s	测量对象： Redis4.0/Redis5.0集群实例数据节点 测量维度： dcs_instance_id dcs_cluster_redis_node	1分钟
hmget	HMGET	该指标用于统计平均每秒hmget操作数。 单位：Count/s	0-500000 Count/s	测量对象： Redis4.0/Redis5.0集群实例数据节点 测量维度： dcs_instance_id dcs_cluster_redis_node	1分钟
hmset	HMSET	该指标用于统计平均每秒hmset操作数。 单位：Count/s	0-500000 Count/s	测量对象： Redis4.0/Redis5.0集群实例数据节点 测量维度： dcs_instance_id dcs_cluster_redis_node	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）
hset	HSET	该指标用于统计平均每秒hset操作数。 单位：Count/s	0-500000 Count/s	测量对象： Redis4.0/Redis5.0集群实例数据节点 测量维度： dcs_instance_id dcs_cluster_redis_node	1分钟
mget	MGET	该指标用于统计平均每秒mget操作数。 单位：Count/s	0-500000 Count/s	测量对象： Redis4.0/Redis5.0集群实例数据节点 测量维度： dcs_instance_id dcs_cluster_redis_node	1分钟
mset	MSET	该指标用于统计平均每秒mset操作数。 单位：Count/s	0-500000 Count/s	测量对象： Redis4.0/Redis5.0集群实例数据节点 测量维度： dcs_instance_id dcs_cluster_redis_node	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）
set	SET	该指标用于统计平均每秒set操作数。 单位：Count/s	0-500000 Count/s	测量对象： Redis4.0/Redis5.0集群实例数据节点 测量维度： dcs_instance_id dcs_cluster_redis_node	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。 单位：Count。	>=0	测量对象： Redis4.0/Redis5.0集群实例数据节点 测量维度： dcs_instance_id dcs_cluster_redis_node	1分钟
bandwidth_usage	带宽使用率	计算当前流量带宽与最大带宽限制的百分比。	0-200%	测量对象： Redis4.0/Redis5.0集群实例数据节点 测量维度： dcs_instance_id dcs_cluster_redis_node	1分钟

表 5-4 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	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。 单位：%。	0-100%	测量对象： Redis3.0Proxy集群实例Proxy节点 测量维度： dcs_instance_id dcs_cluster_proxy_node	1分钟
memory_usage	内存利用率	该指标用于统计测量对象的内存利用率。 单位：%。	0-100%	测量对象： Redis3.0Proxy集群实例Proxy节点 测量维度： dcs_instance_id dcs_cluster_proxy_node	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	测量对象： Redis3.0Proxy集群实例Proxy节点 测量维度： dcs_instance_id dcs_cluster_proxy_node	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）
max_rxpck_per_sec	网卡包接收最大速率	该指标用于统计测量对象网卡在统计周期内每秒接收的最大数据包数。 单位：包/秒	0-100000000包/秒	测量对象： Redis3.0Proxy集群实例Proxy节点 测量维度： dcs_instance_id dcs_cluster_proxy_node	1分钟
max_txpck_per_sec	网卡包发送最大速率	该指标用于统计测量对象网卡在统计周期内每秒发送的最大数据包数。 单位：包/秒	0-100000000包/秒	测量对象： Redis3.0Proxy集群实例Proxy节点 测量维度： dcs_instance_id dcs_cluster_proxy_node	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。 单位：KB/s。	>= 0KB/s	测量对象： Redis3.0Proxy集群实例Proxy节点 测量维度： dcs_instance_id dcs_cluster_proxy_node	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）
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。 单位：KB/s。	>= 0KB/s	测量对象： Redis3.0Proxy集群实例Proxy节点 测量维度： dcs_instance_id dcs_cluster_proxy_node	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。 单位：包/秒	0-10000000 包/秒	测量对象： Redis3.0Proxy集群实例Proxy节点 测量维度： dcs_instance_id dcs_cluster_proxy_node	1分钟
avg_txpck_per_sec	网卡包发送平均速率	该指标用于统计测量对象网卡在统计周期内每秒发送的平均数据包数。 单位：包/秒	0-10000000 包/秒	测量对象： Redis3.0Proxy集群实例Proxy节点 测量维度： dcs_instance_id dcs_cluster_proxy_node	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）
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。 单位：KB/s。	>= 0KB/s	测量对象： Redis3.0Proxy集群实例Proxy节点 测量维度： dcs_instance_id dcs_cluster_proxy_node	1分钟
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。 单位：KB/s。	>= 0KB/s	测量对象： Redis3.0Proxy集群实例Proxy节点 测量维度： dcs_instance_id dcs_cluster_proxy_node	1分钟

Memcached 实例监控指标

表 5-5 Memcached 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）
cpu_usage	CPU利用率	该指标对于统计周期内的测量对象的CPU使用率进行多次采样，表示多次采样的最高值。 单位：%。	0-100%	测量对象： Memcached实例 测量维度： dcs_memcached_instance_id	1分钟
memory_usage	内存利用率	该指标用于统计测量对象的内存利用率。 单位：%。	0-100%	测量对象： Memcached实例 测量维度： dcs_memcached_instance_id	1分钟
net_in_throughput	网络输入吞吐量	该指标用于统计网口平均每秒的输入流量。 单位：byte/s。	>= 0byte/s	测量对象： Memcached实例 测量维度： dcs_memcached_instance_id	1分钟
net_out_throughput	网络输出吞吐量	该指标用于统计网口平均每秒的输出流量。 单位：byte/s。	>= 0byte/s	测量对象： Memcached实例 测量维度： dcs_memcached_instance_id	1分钟
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	测量对象： Memcached实例 测量维度： dcs_memcached_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）
mc_used_memory	已用内存	该指标用于统计已使用的内存字节数。 单位：byte。	>=0byte	测量对象： Memcached实例 测量维度： dcs_memcached_instance_id	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。 单位：byte。	>=0byte	测量对象： Memcached实例 测量维度： dcs_memcached_instance_id	1分钟
mc_used_memory_peak	已用内存峰值	该指标用于统计服务器启动以来使用内存的峰值。 单位：byte。	>=0byte	测量对象： Memcached实例 测量维度： dcs_memcached_instance_id	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	测量对象： Memcached实例 测量维度： dcs_memcached_instance_id	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	测量对象： Memcached实例 测量维度： dcs_memcached_instance_id	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	测量对象： Memcached实例 测量维度： dcs_memcached_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）
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	测量对象： Memcached实例 测量维度： dcs_memcached_instance_id	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。 单位：byte。	>=0byte	测量对象： Memcached实例 测量维度： dcs_memcached_instance_id	1分钟
mc_net_output_bytes	网络发送字节数	该指标用于统计周期内发送的字节数。 单位：byte。	>=0byte	测量对象： Memcached实例 测量维度： dcs_memcached_instance_id	1分钟
mc_instantaneous_input_kbps	网络瞬时输入流量	该指标用于统计瞬时的输入流量。 单位：KB/s。	>=0KB/s	测量对象： Memcached实例 测量维度： dcs_memcached_instance_id	1分钟
mc_instantaneous_output_kbps	网络瞬时输出流量	该指标用于统计瞬时的输出流量。 单位：KB/s。	>=0KB/s	测量对象： Memcached实例 测量维度： dcs_memcached_instance_id	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	测量对象： Memcached实例 测量维度： dcs_memcached_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）
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	测量对象： Memcached实例 测量维度： dcs_memcached_instance_id	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	测量对象： Memcached实例 测量维度： dcs_memcached_instance_id	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	测量对象： Memcached实例 测量维度： dcs_memcached_instance_id	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	测量对象： Memcached实例 测量维度： dcs_memcached_instance_id	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	测量对象： Memcached实例 测量维度： dcs_memcached_instance_id	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	测量对象： Memcached实例 测量维度： dcs_memcached_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）
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	测量对象： Memcached实例 测量维度： dcs_memcached_instance_id	1分钟
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	测量对象： Memcached实例 测量维度： dcs_memcached_instance_id	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	测量对象： Memcached实例 测量维度： dcs_memcached_instance_id	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	测量对象： Memcached实例 测量维度： dcs_memcached_instance_id	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	测量对象： Memcached实例 测量维度： dcs_memcached_instance_id	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	测量对象： Memcached实例 测量维度： dcs_memcached_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）
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	测量对象： Memcached实例 测量维度： dcs_memcached_instance_id	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	测量对象： Memcached实例 测量维度： dcs_memcached_instance_id	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	测量对象： Memcached实例 测量维度： dcs_memcached_instance_id	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	测量对象： Memcached实例 测量维度： dcs_memcached_instance_id	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	测量对象： Memcached实例 测量维度： dcs_memcached_instance_id	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	测量对象： Memcached实例 测量维度： dcs_memcached_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）
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	测量对象： Memcached实例 测量维度： dcs_memcached_instance_id	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	测量对象： Memcached实例 测量维度： dcs_memcached_instance_id	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	测量对象： Memcached实例 测量维度： dcs_memcached_instance_id	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	测量对象： Memcached实例 测量维度： dcs_memcached_instance_id	1分钟
mc_command_max_delay	命令最大时延	统计命令最大时延。 单位：ms。	>=0ms	测量对象： Memcached实例 测量维度： dcs_memcached_instance_id	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。	测量对象： Memcached实例 测量维度： dcs_memcached_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）
mc_keyspace_hits_perc	访问命中率	统计实例的访问码命中率。 单位：%。	0-100%	测量对象： Memcached实例 测量维度： dcs_memcached_instance_id	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
dcs_memcached_instance_id	Memcached实例

5.2 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

5.3 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 5-6 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。
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。
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。
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Redis3.0实例的单机/主备实例进行配置，建议按Redis3.0规格基准带宽的80%进行配置。其他实例不配置。

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
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Redis3.0实例的单机/主备实例进行配置，建议按Redis3.0规格基准带宽的80%进行配置。其他实例不配置。

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“下一步”。
4. 在“规则信息”，设置告警名称和告警的描述。
5. 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。

----结束

6 审计

6.1 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 6-1 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	分布式缓存服务	createDCSInstance
提交创建实例请求	分布式缓存服务	submitCreateDCSInstanceRequest
批量删除实例	分布式缓存服务	batchDeleteDCSInstance
删除实例	分布式缓存服务	deleteDCSInstance
修改实例信息	分布式缓存服务	modifyDCSInstanceInfo
修改实例配置	分布式缓存服务	modifyDCSInstanceConfig
修改实例密码	分布式缓存服务	modifyDCSInstancePassword
重启实例	分布式缓存服务	restartDCSInstance
提交重启实例请求	分布式缓存服务	submitRestartDCSInstanceRequest

操作类型	资源类型	事件名称
启动实例	分布式缓存服务	startDCSInstance
提交启动实例请求	分布式缓存服务	submitStartDCSInstanceRequest
清空实例	分布式缓存服务	flushDCSInstance
批量重启实例	分布式缓存服务	batchRestartDCSInstance
提交批量重启实例请求	分布式缓存服务	submitBatchRestartDCSInstanceRequest
批量启动实例	分布式缓存服务	batchStartDCSInstance
提交批量启动实例请求	分布式缓存服务	submitBatchStartDCSInstanceRequest
恢复实例数据	分布式缓存服务	restoreDCSInstance
提交恢复实例数据请求	分布式缓存服务	submitRestoreDCSInstanceRequest
备份实例数据	分布式缓存服务	backupDCSInstance
提交备份实例数据请求	分布式缓存服务	submitBackupDCSInstanceRequest
删除实例备份文件	分布式缓存服务	deleteInstanceBackupFile
删除后台任务记录	分布式缓存服务	deleteDCSInstanceJobRecord
实例规格变更	分布式缓存服务	modifySpecification
提交实例规格变更请求	分布式缓存服务	submitModifySpecificationRequest
创建实例订单	分布式缓存服务	createInstanceOrder
主备切换	分布式缓存服务	masterStandbySwitchover
重置实例密码	分布式缓存服务	resetDCSInstancePassword
提交清空实例请求	分布式缓存服务	submitFlushDCSInstanceRequest

6.2 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6-1所示。

图 6-1 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6-2所示，显示了该操作事件结构的详细信息。

图 6-2 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

7 常见问题

7.1 实例类型/版本

7.1.1 版本差异

DCS在创建实例时，Redis可选择“版本号”、“实例类型”。

- **版本号**

  版本号共有3.0，4.0，5.0，它们的区别如表7-1。更多Redis 4.0和Redis 5.0的特性，请参考“DCS Redis 4.0支持的新特性说明”和“DCS Redis 5.0支持的新特性说明”章节。

  表 7-1 不同版本支持的特性、性能差异说明

比较项	Redis 3.0	Redis 4.0 & Redis 5.0
兼容开源版本	Redis 3.0兼容开源3.0.7版本	Redis 4.0兼容开源4.0.14版本，Redis 5.0兼容开源5.0.9版本
实例部署模式	采用虚机部署	在物理机上容器化部署
CPU架构	支持x86和Arm	支持x86和Arm
创建实例耗时	3~15分钟，集群约10~30分钟	约8秒
QPS	单节点约10万QPS	单节点约10万QPS
公网访问	支持	暂不支持
域名连接	支持VPC内使用域名连接	支持VPC内使用域名连接
可视化数据管理	不支持	提供Web CLI访问Redis，管理数据

比较项	Redis 3.0	Redis 4.0 & Redis 5.0
实例类型	支持单机、主备、Proxy集群	支持单机、主备、Cluster集群
实例规格	提供2G、4G、8G直至1024G多种规格	提供2G、4G、8G直至1024G多种规格，同时单机主备还支持128MB、256MB、512MB、1GB四种小规格实例
扩容/缩容	支持在线扩容和缩容	支持在线扩容和缩容
备份恢复	主备和集群实例支持	主备、集群实例支持

说明

由于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 3.0暂不支持升级到Redis 4.0或者Redis 5.0。如果需要由低版本升级到高版本，建议重新创建高版本实例，然后进行数据迁移。

- **实例类型**

  Redis实例类型分为单机、主备、Proxy集群、Cluster集群，它们的架构与应用场景，请参考“实例类型”章节。

7.1.2 DCS Redis 4.0 支持的新特性说明

与Redis 3.0版本相比，Redis 4.0以上版本，除了开源Redis增加的特性之外，创建耗时也相应缩短。

实例由虚机方式改成了物理机容器化部署，创建实例只需要8~10秒时间完成。

Redis 4.0版本更新的特性，主要涉及三个方面：

1. 新命令的增加，如MEMORY、SWAPDB。
2. Lazyfree机制，延迟删除大key，降低删除操作对系统资源的占用影响。
3. 内存性能优化，即主动碎片整理。

MEMORY 命令

在Redis 3.0及之前，只能通过info memory命令了解有限的几个内存统计信息。Redis 4.0引入新的命令memory，让您能够更深入了解Redis的内存使用情况。

```
127.0.0.1:6379[8]> memory help
1) MEMORY <subcommand> arg arg ... arg. Subcommands are:
2) DOCTOR - Return memory problems reports.
3) MALLOC-STATS -- Return internal statistics report from the memory allocator.
4) PURGE -- Attempt to purge dirty pages for reclamation by the allocator.
5) STATS -- Return information about the memory usage of the server.
6) USAGE <key> [SAMPLES <count>] -- Return memory in bytes used by <key> and its value. Nested values are sampled up to <count
> times (default: 5).
127.0.0.1:6379[8]>
```

usage

输入**memory usage *[key]***，如果当前key存在，则返回key的value实际使用内存估算值；如果key不存在，则返回nil。

```
127.0.0.1:6379[8]> set dcs "DCS is an online, distributed, in-memory cache service compatible with Redis, and Memcached."
OK
127.0.0.1:6379[8]> memory usage dcs
(integer) 141
127.0.0.1:6379[8]>
```

说明

1. usage统计value内存占用，以及key自身的内存占用，不包含key的Expire内存占用。

   ```
   //以下内容基于Redis 5.0.2版本验证，不同Redis版本，统计结果可能有差异。
   192.168.0.66:6379> set a "Hello, world!"
   OK
   192.168.0.66:6379> memory usage a
   (integer) 58
   192.168.0.66:6379> set abc "Hello, world!"
   OK
   192.168.0.66:6379> memory usage abc
   (integer) 60   //key名称长度变化后，内存占用也有变化，说明usage统计包含了key自身的占用
   192.168.0.66:6379> expire abc 1000000
   (integer) 1
   192.168.0.66:6379> memory usage abc
   (integer) 60    //加了过期时间后，内存占用没有改变，说明usage统计不包含expire内存占用
   192.168.0.66:6379>
   ```

2. 对hash、list、set、sorted set等数据类型，usage命令会抽样统计，提供内存占用的估算值。

   使用方式：**memory usage** *keyset* **samples** *1000*

   其中keyset表示一个集合数据类型的key，1000表示抽样个数。

stats

返回当前实例内存使用细节。

使用方法：**memory stats**

```
127.0.0.1:6379[8]> memory stats
 1) "peak.allocated"
 2) (integer) 2412408
 3) "total.allocated"
 4) (integer) 2084720
 5) "startup.allocated"
 6) (integer) 824928
 7) "replication.backlog"
 ... ...
```

以下给出部分数据返回项的具体含义

表 7-2 memory stats

数据返回项	说明
peak.allocated	Redis实例运行过程中，allocator分配的内存峰值。同info memory的used_memory_peak
total.allocated	allocator当前分配的内存字节数。同info memory的used_memory
startup.allocated	Redis启动占用的内存字节数
replication.backlog	Redis复制积压缓冲区（replication backlog）内存使用字节数，通过repl-backlog-size参数设置，默认1M
clients.slaves	在master侧，所有slave clients消耗的内存字节数

数据返回项	说明
clients.normal	Redis所有常规客户端消耗内存节字数
overhead.total	Redis额外的总开销内存字节数； 即分配器分配的总内存total.allocated，减去数据实际存储使用内存。
keys.count	Redis实例中key的数量
keys.bytes-per-key	每个key平均占用字节数。注意，overhead也会均摊到每个key上，因此不能以此值来表示业务实际的key平均长度。
dataset.bytes	表示Redis数据占用的内存容量。即分配的内存总量，减去总的额外开销内存量。
dataset.percentage	表示Redis数据占用内存占总内存分配的百分比
peak.percentage	当前内存使用量与峰值时的占比
fragmentation	表示Redis的内存碎片率

doctor

使用方法： **memory doctor**

used_memory（total.allocated）小于5M，doctor认为内存使用量过小，不做进一步诊断。当满足以下某一点，Redis会给出诊断结果和建议：

1. peak分配内存大于当前total_allocated的1.5倍，即peak.allocated/total.allocated > 1.5，说明内存碎片率高，RSS远大于used_memory
2. High fragmentation/fragmentation大于1.4，说明内存碎片率高
3. 每个Normal Client平均使用内存大于200KB，说明pipeline可能使用不当，或Pub/Sub客户端处理消息不及时
4. 每个Slave Client平均使用内存大于10MB，说明master的写入流量过高

purge

使用方法： **memory purge**

用途：通过调用jemalloc内部命令，进行内存释放。释放对象包括Redis进程占用但未有效使用的内存，即常说的内存碎片。

说明

memory purge只适用于使用jemalloc作为allocator的Redis实例。

Lazy free 机制

解决的痛点/问题

Redis是单线程程序，当运行一个耗时较大的请求时，会导致所有请求排队等待，在请求处理完成前，Redis不能响应其他请求，因此容易引发性能问题。而Redis删除大的集合键时，就属于一种比较耗时的请求。

原理

Redis 4.0提供的一种惰性删除或者说延迟释放机制，主要用于解决删除大key对Redis进程的阻塞，从而避免带来性能与可用性问题。

删除key时，Redis异步延时释放key的内存，把key释放操作放在bio(Background I/O)单独的子线程处理中。

使用方法

1. 主动删除
   - **unlink**

     unlink与del命令目的一样，删除某个key。unlink在删除集合类键时，如果集合键的元素个数大于64个，会把内存释放操作，给单独的bio(Background I/O)线程来执行。因此unlink删除操作能在非常短的时间内完成包含上百万个元素的大key删除。
   - **flushall/flushdb**

     通过对flushall/flushdb添加ASYNC异步清理选项，Redis在清理整个实例或单个DB时，操作都是异步的。
2. 过期key删除、大key驱逐删除

   被动删除有四种场景，每种场景对应一个配置参数，默认都是关闭：

```
lazyfree-lazy-eviction no //针对redis内存使用达到maxmeory，并设置有淘汰策略时，是否采用lazy free机制
lazyfree-lazy-expire no //针对设置有TTL的键，过期后，被redis清理删除时是否采用lazy free机制
lazyfree-lazy-server-del no //针对有些指令在处理已存在的键时，会带有一个隐式的DEL键的操作
slave-lazy-flush no //针对slave进行全量数据同步，slave在加载master的RDB文件前，会运行flushall来清理自己的数据场景
```

   说明

   以上配置如需使用，请咨询技术服务人员。

其他新增命令

1. **swapdb**

   用途：交换同一Redis实例内2个db的数据。

   用法：**swapdb** *dbindex1 dbindex2*
2. **zlexcount**

   用途：在有序集合中，返回符合条件的元素个数。

   用法：**zlexcount** *key min max*

内存使用和性能改进

1. 使用更少的内存来存储相同数量的数据
2. 可以对使用的内存进行碎片整理，并逐渐回收

7.1.3 DCS Redis 5.0 支持的新特性说明

DCS的Redis 5.0版本继承了Redis 4.0版本的所有功能增强以及新的命令，同时还兼容开源Redis 5.0版本的新增特性。

Stream 数据结构

Stream是Redis 5.0引入的一种新数据类型，它是一个全新的支持多播的可持久化消息队列。

Redis Stream的结构示意图如图7-1所示，它是一个可持久化的数据结构，用一个消息链表，将所有加入进来的消息都串起来。

Stream数据结构具有以下特性：

1. Stream中可以有多个消费者组。
2. 每个消费组都含有一个Last_delivered_id，指向消费组当前已消费的最后一个元素（消息）。
3. 每个消费组可以含有多个消费者对象，消费者共享消费组中的Last_delivered_id，相同消费组内的消费者存在竞争关系，即一个元素只能被其中一个消费者进行消费。
4. 消费者对象内还维持了一个Pending_ids，Pending_ids记录已发送给客户端，但是还没完成ACK（消费确认）的元素id。
5. Stream与Redis其他数据结构的比较，见表7-3。

图 7-1 Stream 数据结构示意图

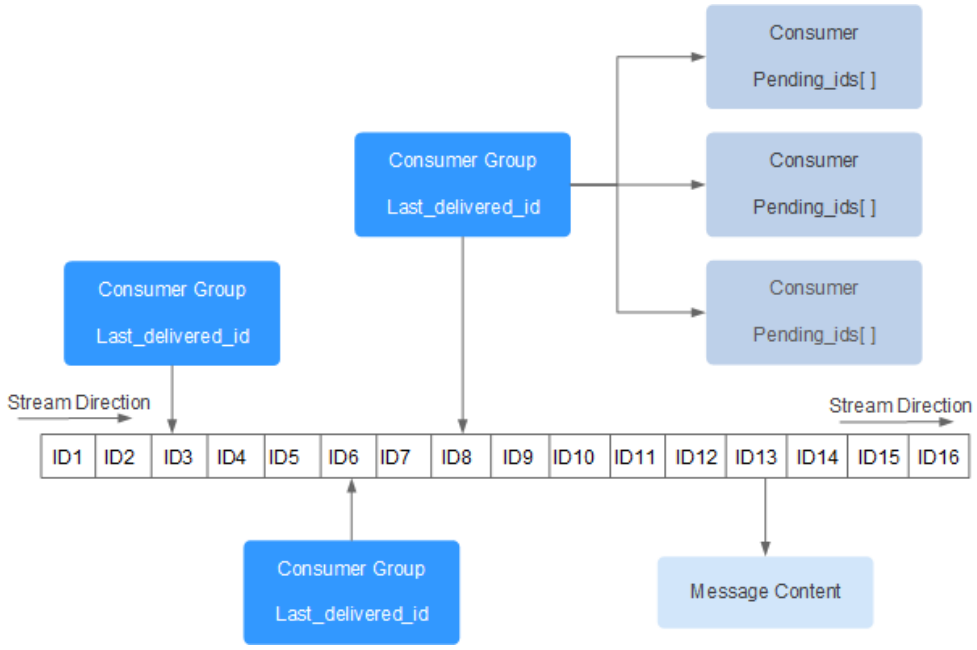


表 7-3 Stream 与 Redis 现有数据结构比较

比较项	Stream	List、Pub/Sub、Zset
复杂度	获取元素高效，复杂度为O(logN)	List获取元素的复杂度为O(N)
offset	支持offset，每个消息元素有唯一id。不会因为新元素加入或者其他元素淘汰而改变id。	List没有offset概念，如果有元素被逐出，无法确定最新的元素
持久化	支持消息元素持久化，可以保存到AOF和RDB中。	Pub/Sub不支持持久化消息。
消费分组	支持消费分组	Pub/Sub不支持消费分组

比较项	Stream	List、Pub/Sub、Zset
消息确认	支持ACK（消费确认）	Pub/Sub不支持
性能	Stream性能与消费者数量无明显关系	Pub/Sub性能与客户端数量正相关
逐出	允许按时间线逐出历史数据，支持block，给予radix tree和listpack，内存开销少。	Zset不能重复添加相同元素，不支持逐出和block，内存开销大。
删除元素	不能从中间删除消息元素。	Zset支持删除任意元素

Stream相关命令介绍

接下来按照使用流程中出现的顺序介绍**Stream相关命令**。详细命令见**表7-4**

1. 首先使用**XADD**添加流元素，即创建Stream，添加流元素时可指定消息数量最大保存范围。
2. 然后通过**XGROUP**创建消费者组。
3. 消费者使用**XREADGROUP**指令进行消费。
4. 客户端消费完毕后使用**XACK**命令确认消息已消费成功。

图 7-2 Stream 相关命令介绍

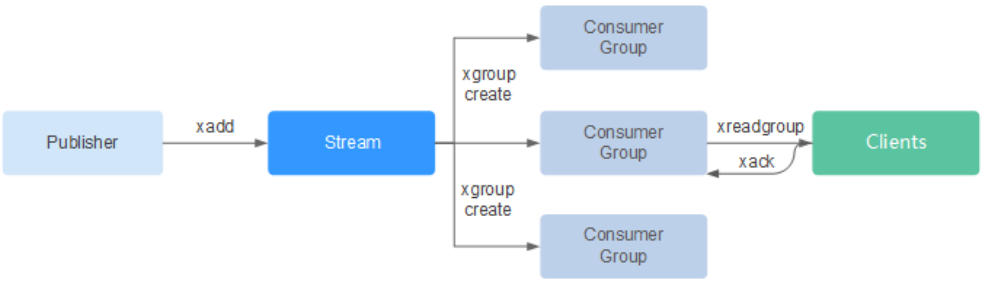


表 7-4 Stream 的详细命令

命令	说明	语法
XACK	从流的消费者组的*待处理条目列表*（简称PEL）中删除一条或多条消息。	XACK key group ID [ID ...]
XADD	将指定的流条目追加到指定key的流中。 如果key不存在，作为运行这个命令的副作用，将使用流的条目自动创建key。	XADD key ID field string [field string ...]

命令	说明	语法
XCLAIM	在流的消费者组上下文中，此命令改变待处理消息的所有权， 因此新的所有者是在命令参数中指定的消费者。	XCLAIM key group consumer min-idle-time ID [ID ...] [IDLE ms] [TIME ms-unix-time] [RETRYCOUNT count] [FORCE] [JUSTID]
XDEL	从指定流中移除指定的条目，并返回成功删除的条目的数量，在传递的ID不存在的情况下， 返回的数量可能与传递的ID数量不同。	XDEL key ID [ID ...]
XGROUP	该命令用于管理流数据结构关联的消费者组。使用XGROUP你可以： ● 创建与流关联的新消费者组。 ● 销毁一个消费者组。 ● 从消费者组中移除指定的消费者。 ● 将消费者组的*最后交付ID*设置为其他内容。	XGROUP [CREATE key groupname id-or-$] [SETID key id-or-$] [DESTROY key groupname] [DELCONSUMER key groupname consumername]
XINFO	检索关于流和关联的消费者组的不同的信息。	XINFO [CONSUMERS key groupname] key key [HELP]
XLEN	返回流中的条目数。如果指定的key不存在，则此命令返回0，就好像该流为空。	XLEN key
XPENDING	通过消费者组从流中获取数据。检查待处理消息列表的接口，用于观察和了解消费者组中哪些客户端是活跃的，哪些消息在等待消费，或者查看是否有空闲的消息。	XPENDING key group [start end count] [consumer]
XRANGE	返回流中满足给定ID范围的条目。	XRANGE key start end [COUNT count]
XREAD	从一个或者多个流中读取数据，仅返回ID大于调用者报告的最后接收ID的条目。	XREAD [COUNT count] [BLOCK milliseconds] STREAMS key [key ...] ID [ID ...]
XREADGROUP	XREAD命令的特殊版本，指定消费者组进行读取。	XREADGROUP GROUP group consumer [COUNT count] [BLOCK milliseconds] STREAMS key [key ...] ID [ID ...]
XREVRANGE	与XRANGE相同，但显著的区别是以相反的顺序返回条目，并以相反的顺序获取开始-结束参数	XREVRANGE key end start [COUNT count]

命令	说明	语法
XTRIM	XTRIM将流裁剪为指定数量的项目，如有需要，将驱逐旧的项目（ID较小的项目）。	XTRIM key MAXLEN [~] count

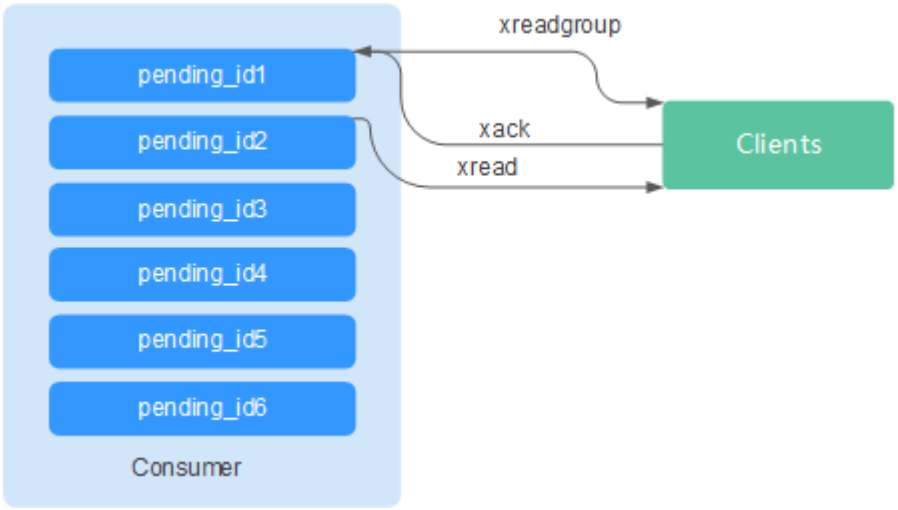
消息（流元素）消费确认

Stream与相比Pub/Sub，不仅增加消费分组模式，还支持消息消费确认。

当一条消息被某个消费者调用XREADGROUP命令读取或调用XCLAIM命令接管的时候， 服务器尚不确定它是否至少被处理了一次。 因此，**一旦消费者成功处理完一条消息，它应该调用XACK知会Stream，这样这个消息就不会被再次处理， 同时关于此消息的PEL（pending_ids）条目也会被清除，从Redis服务器释放内存**。

某些情况下，因为网络问题等，客户端消费完毕后没有调用XACK，这时候PEL内会保留对应的元素ID。待客户端重新连上后，XREADGROUP的起始消息ID建议设置为0-0，表示读取所有的PEL消息及自last_id之后的消息。同时，消费者消费消息时需要能够支持消息重复传递。

图 7-3 ACK 机制解读



内存使用优化

Redis 5.0在上一版本基础上，在内存使用上做了进一步优化。

- 主动碎片整理

  当key被频繁修改，value长度不断变化时，Redis会为key分配新的内存空间。由于Redis追求高性能，实现了自己的内存分配器来管理内存，因此并不会将原有内存释放给OS，从而导致出现内存碎片。当used_memory_rss/used_memory高于1.5，一般认为内存碎片占比过高，内存利用率低。

  因此，合理规划和使用缓存数据，规范数据写入，有助于减少内存碎片的产生。

  Redis 3.0及以下：可以通过定期重启服务解决内存碎片问题。建议实际缓存数据不超过配置可用内存的50%。

  Redis 4.0：支持主动整理内存碎片，服务在运行期间进行自动内存碎片清理。同时Redis 4.0支持通过memory purge命令手动清理内存碎片。

Redis 5.0：增强版主动碎片整理，配合Jemalloc版本更新，更快更智能，延时更低。

- HyperLogLog算法优化

  HyperLogLog是一种基数计数方法，使用少量的内存空间完成海量数据的计数统计，在Redis 5.0中，HyperLogLog算法得到改进，优化了计数统计时的内存使用效率。

  举个例子：B树计数效率非常高，但是内存消耗也比较多。而HyperLogLog可节省大量存储空间。当B树需要1M内存统计，HyperLogLog只需要1kb。

- 内存信息统计报告能力增强

  INFO命令返回信息更加详实。

命令新增和优化

1. **客户端管理增强**
   - Redis-cli支持集群管理

     在Redis 4.0以及之前版本，需要安装redis-trib模块，管理集群。

     Redis 5.0对Redis-cli做了优化，集成了集群的所有管理功能。具体使用可以通过命令**redis-cli --cluster help**查看帮助信息。
   - 优化客户端在频繁连接与中断场景下的性能

     当您的应用需要使用短连接时，这个优化价值凸显。
2. **有序集合使用更简单**

   有序集合新增两个命令：ZPOPMIN和ZPOPMAX。
   - ZPOPMIN key [count]

     删除并返回有序集合key中的最多count个具有最低得分的成员。如果返回多个成员，也会按照得分高低（value值比较），从低到高排列。
   - ZPOPMAX key [count]

     删除并返回有序集合key中的最多count个具有最高得分的成员。如果返回多个成员，也会按照得分高低（value值比较），从高到低排列。
3. **help增加更多子命令说明**

   支持help直接查看快速使用攻略，你不再需要每次登录redis.io去查找。例如，命令行输入stream使用攻略：xinfo help

   ```
   127.0.0.1:6379> xinfo help
   1) XINFO <subcommand> arg arg ... arg. Subcommands are:
   2) CONSUMERS <key> <groupname>  -- Show consumer groups of group <groupname>.
   3) GROUPS <key>                 -- Show the stream consumer groups.
   4) STREAM <key>                 -- Show information about the stream.
   5) HELP                         -- Print this help.
   127.0.0.1:6379>
   ```
4. **Redis-cli命令输入提示**

   Redis-cli在输入完整的命令后，会展示参数提醒，帮助用户记忆命令语法格式。

   如下图所示，输入**zadd**命令，Redis-cli使用浅颜色字体显示zadd的语法。

   ```
   # Cluster
   cluster_enabled:0

   # Keyspace
   db0:keys=1,expires=0,avg_ttl=0
   198.19.59.199:6379> zadd key [NX|XX] [CH] [INCR] score member [score member ...]
   ```

RDB 支持存储 LFU、LRU

Redis 5.0开始，RDB快照文件中增加存储key逐出策略**LRU**和**LFU**：

- FIFO：先进先出。最早存储的数据，优先被淘汰。
- LRU：最近最少使用。长期未使用的数据，优先被淘汰。
- LFU：最不经常使用。在一段时间内，使用次数最少的数据，优先被淘汰。

说明

Redis 5.0的RDB文件格式有变化，向下兼容。因此如果使用快照的方式迁移，可以从Redis低版本迁移到Redis 5.0，但不能从Redis 5.0迁移到低版本。

7.2 客户端和网络连接

7.2.1 安全组配置和选择

本节主要介绍**VPC内访问**DCS缓存实例时如何配置安全组。

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，还需要他们的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，源地址建议使用指定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。

7.2.2 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。

如果您在应用开发调试阶段，可以通过ssh代理方式，实现本地环境访问缓存实例。

7.2.3 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。

7.2.4 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。

7.2.5 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。

如果出现超时错误概率频繁，请联系服务运维。

7.2.6 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**。

   - 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。

```
root@heru-nodelete:~# ss -s
Total: 140 (kernel 240)
TCP:   11 (estab 3, closed 1, orphaned 0, synrecv 0, timewait 0/0), ports 0

Transport Total     IP        IPv6
*         240       -         -
RAW       0         0         0
UDP       2         2         0
TCP       10        6         4
INET      12        8         4
FRAG      0         0         0
```

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
##修改系統默认的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

7.2.7 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

   如下图所示，在Redis3.0版本运行Redis5.0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

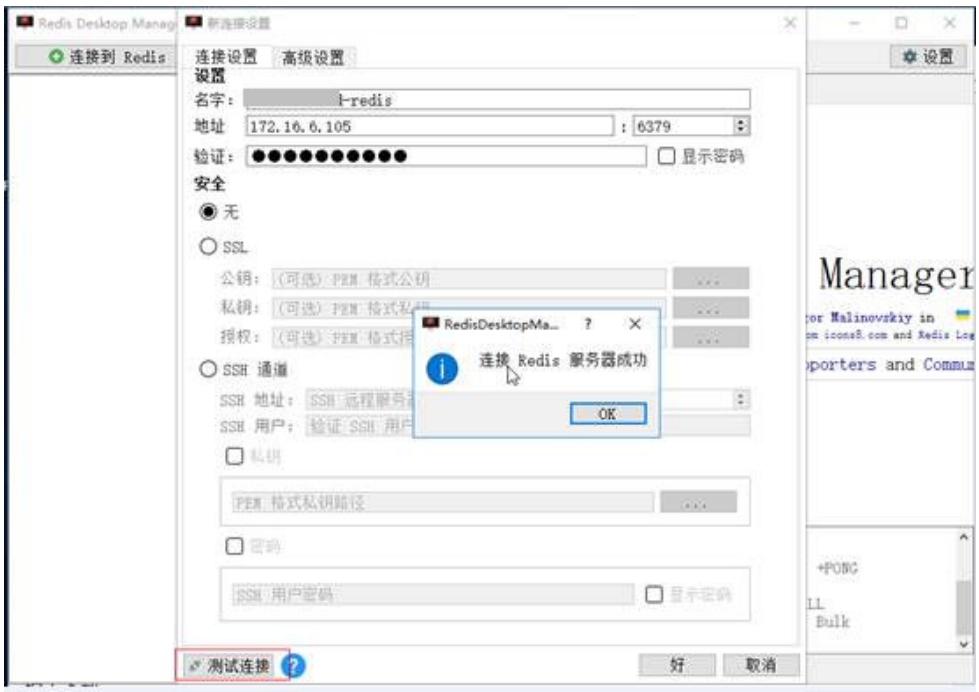
   DCS Redis实例接口与开源Redis在数据访问方面完全兼容。但因易用性和安全性的原因，部分管理操作不能从Redis客户端发起，具体禁用的命令清单，请参考**Redis命令**。

7.2.8 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 7-4 通过内网使用 Redis-desktop-manager 访问 Redis 实例



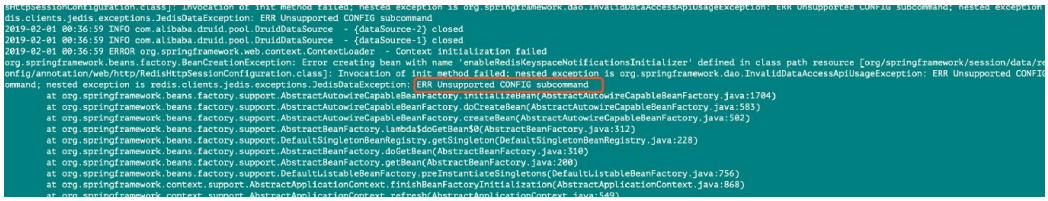
说明

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。

7.2.9 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 7-5 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

仅Redis**单机**和**主备**实例支持Spring的Session共享，Redis集群版不支持。

7.2.10 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。

- 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**。

- 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。

7.2.11 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。

7.3 Redis 使用

7.3.1 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。

7.3.2 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。

7.3.3 Redis4.0 和 Redis5.0 实例为什么没有安全组信息?

目前Redis4.0和Redis5.0版本实例是基于VPCEndpoint，暂不支持安全组。

7.3.4 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。

7.3.5 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&5.0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
```

7.3.6 创建 Redis3.0 版本实例，为什么可使用内存比实例规格少一些?

Redis3.0版本采用虚拟机部署，系统会占用小部分内存。

7.3.7 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。

Redis集群实例不支持多DB。

7.3.8 Redis 集群实例是否支持原生集群?

当前DCS Redis3.0版本支持Proxy集群，Redis4.0和5.0版本支持原生集群。

7.3.9 Redis 实例是否支持配置哨兵模式?

Redis4.0和Redis5.0支持配置哨兵模式（Sentinel），且默认开启。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。

Redis3.0不支持哨兵模式，使用的是keeplived进行监控，当主节点故障时进行主备切换，备节点自动接管服务。

7.3.10 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。当前版本支持在配置运行参数中修改逐出策略。

单机和主备的Redis实例默认的数据逐出策略为不逐出（noeviction）。单机和主备的Redis实例支持在配置运行参数中修改数据逐出策略。

集群Redis实例默认的数据逐出策略为volatile-lru。如果需要集群Redis实例的数据逐出策略，可以联系技术支持。

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。

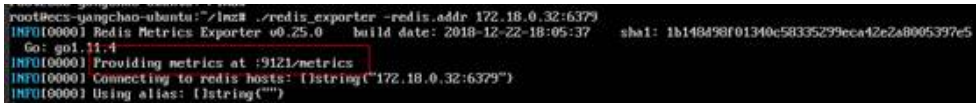
- allkeys-lru：根据LRU（Least recently used，最近最少使用）算法尝试回收最少使用的键，使得新添加的数据有空间存放。
- volatile-lru：根据LRU（Least recently used，最近最少使用）算法尝试回收最少使用的键，但仅限于在过期集合的键，使得新添加的数据有空间存放。
- allkeys-random：回收随机的键使得新添加的数据有空间存放。
- volatile-random：回收随机的键使得新添加的数据有空间存放，但仅限于在过期集合的键。
- volatile-ttl：回收在过期集合的键，并且优先回收存活时间（TTL）较短的键，使得新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。

7.3.11 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。



7.3.12 DCS 的 Redis 实例内存占用率略超过 100%是什么情况?

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。

7.3.13 Redis3.0 Proxy 集群不支持 redisson 分布式锁的原因

redisson分布式锁的加锁和解锁流程如下：

1. redisson分布式锁的加锁和解锁都是执行一段lua脚本功能实现的。
2. 在加锁阶段，需要在lua脚本中执行exists、hset、pexpire、hexists、hincrby、pexpire、pttl命令。
3. 在解锁阶段，需要在lua脚本中执行exists、publish、hexists、pexpire、del命令。

由于Proxy集群支持publish/subscribe(redis的发布订阅)时，是需要在Proxy节点上识别publish/subscribe命令，做一些特殊处理（转发给所有redis-server的节点），因此不支持直接在lua脚本中执行publish命令。

因此，Redis3.0 Proxy集群无法支持redisson的分布式锁机制，如果需要使用redisson分布式锁功能，建议使用Redis4.0或Redis5.0集群。

7.3.14 实例是否支持自定义或修改端口?

如果是Redis3.0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Redis4.0和Redis5.0实例，支持自定义端口，也支持修改端口。

- Redis3.0

  VPC内使用实例6379端口。

- Memcached

  仅支持VPC内访问，端口为11211端口。

- Redis4.0和Redis5.0

  创建Redis4.0和Redis5.0实例时，可选择自定义端口和使用默认端口，定义端口范围为1~65535的任意数字，默认端口为6379，如果没有自定义，则使用默认端口。

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**。

7.3.15 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。

有关DCS实例的客户端访问，请参考**连接缓存实例**。

7.3.16 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。

7.3.17 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。

7.3.18 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。

7.3.19 Redis 实例经常内存满了但是 key 不多的原因

可能原因：客户端缓冲区（output buffer）占用过多的内存空间。

解决方案：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。

7.3.20 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。

DCS的默认数据逐出策略为不逐出，但用户可以通过修改实例配置参数，自行调整策略，由实例根据逐出策略回收键值。

7.3.21 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。

7.4 Redis 命令

7.4.1 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。

- Redis3.0实例

  Redis3.0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。

- Redis4.0和Redis5.0实例

  Redis4.0/5.0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和Redis 5.0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主节点上，挨个执行flushdb。
  - Web CLI方式不支持清空Cluster集群的数据。

7.4.2 高危命令如何重命名?

当前支持重命名Redis4.0和Redis5.0实例高危命令command、keys、flushdb、flushall和hgetall，其他命令，暂不支持重命名。

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。

7.4.3 是否支持 pipeline 命令?

支持。注意：Redis 4.0&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。

7.4.4 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。

7.4.5 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
- 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。

7.4.6 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。

7.4.7 Redis 命令执行是否有超时时间？超时了会出现什么结果？

单个Redis命令处理时长限制为1分钟，目前Redis命令超时时间暂不支持配置。命令处理超时，服务端将会自动断开与客户端的连接。

7.5 扩容缩容与实例升级

7.5.1 Redis 实例是否支持版本升级？如 Redis3.0 升级到 Redis4.0/Redis5.0?

不支持。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Redis3.0的实例不能升级到Redis4.0或Redis5.0。但DCS服务在发现Redis缺陷或者问题时，会主动通知客户修复问题。

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**。

7.5.2 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。

7.5.3 DCS 实例规格变更是否需要停服？

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。

7.5.4 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。

- **实例类型变更影响：**
  - Redis3.0单机实例类型变更为主备实例。
    连接会有秒级中断，大约1分钟左右的只读。

  - Redis3.0主备实例类型变更为Proxy实例。

    连接会中断，5~30分钟只读。

- **实例规格大小变更影响：**
  - 单机和主备实例规格大小变更。

    Redis3.0/4.0/5.0实例变更期间，连接会有秒级中断，大约1分钟的只读。

    如果是扩容，只扩大实例的内存，不会提升CPU处理能力。

    如果是单机实例规格变更，由于单机实例不支持持久化，实例规格变更后，可能不保留数据，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。
  - Proxy集群实例规格大小变更。

    连接不中断，会占用CPU，扩容数据迁移期间，访问时延会增大。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。
  - Cluster集群实例容量大小变更。

    连接不中断，节点CPU会升高，扩容数据迁移期间，访问时延会增大。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。
  - 变更规格前的备份记录不能恢复。

7.5.5 Redis/Memcached 实例变更失败的原因

主要原因为实例变更过程中，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，请联系技术支持。

7.6 监控告警

7.6.1 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。

7.6.2 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：**http://www.redis.cn/commands/info.html**。

7.6.3 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。

7.6.4 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。

7.7 数据备份/导出/迁移

7.7.1 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Redis3.0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Redis5.0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。

7.7.2 是否支持控制台导出 RDB 格式的 Redis 备份文件?

- Redis3.0实例

  Redis3.0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}

- Redis4.0和Redis5.0实例

  Redis4.0/5.0实例备份的文件格式是RDB，从控制台下载备份文件就是RDB格式的备份文件，而不支持通过Redis-cli导出RDB文件。

7.7.3 DCS 支持数据持久化吗?

对于Redis类型的缓存实例：

- 单机：不支持。
- 主备和集群：支持。

对于Memcached类型的缓存实例：

- 单机：不支持。
- 主备：支持。

7.7.4 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非阻塞的获取key，避免KEYS命令造成Redis服务阻塞。
- 支持所有数据类型的迁移。
- 把SCAN和DUMP/RESTORE操作放在同一个管道中，利用pipeline提升数据迁移过程中的网络效率。
- 不使用任何临时文件，不占用磁盘空间。
- 使用带缓冲区的channels，提升源服务器的性能。

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 ~]#
```

7.8 主备倒换

7.8.1 发生主备倒换的原因有哪些?

主备倒换有3种可能场景：

- 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。

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。

7.8.2 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。

7.8.3 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。

7.8.4 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。

7.9 Memcached 使用

7.9.1 Memcached 实例的数据能否 dump 出来分析?

不支持将数据导出分析。

7.9.2 DCS 的 Memcached 兼容的版本号是多少?

DCS的Memcached是基于Redis3.0.7版本引擎实现的，兼容memcache1.5.1版本。

7.9.3 DCS 的 Memcached 支持哪些数据结构?

DCS的Memcached目前仅支持Key-Value的数据结构，暂不支持List等数据结构。

7.9.4 Memcached 实例支持公网访问么?

Memcached实例暂不支持公网访问，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。如果您在应用开发调试阶段，可以通过ssh代理方式，实现本地环境访问实例。

7.9.5 Memcached 实例是否支持修改配置参数?

处于“运行中”状态的Memcached实例支持修改配置参数。

具体修改操作，请参考**配置运行参数**。

7.9.6 DCS 的 Memcached 与自建 Memcached 的区别是什么?

DCS的Memcached与本地自建Memcached的区别如下**表7-5**所示。

表 7-5 DCS Memcached 与自建 Memcached 的区别

比较项	DCS Memcached	自建Memcached
部署	DCS的Memcached简单易用，创建后缓存服务即时可用，实现快速部署，无需关心硬件及软件。	自己搭建Memcached的操作和设置较复杂。
可用性	DCS的Memcached主备实例支持双机热备，能够持续提供稳定的服务，在主节点故障时，备节点秒级自动升级为主节点，避免单点故障。	自建的Memcached服务需要自行实现高可用。

比较项	DCS Memcached	自建Memcached
安全	DCS的Memcached采用VPC和安全组进行网络访问安全控制。	自建的Memcached服务需要用户自行设计并实现安全机制。
扩容	DCS的Memcached支持在线扩容，在控制台简单操作，即可完成扩容。	自建的Memcached服务的扩容稍复杂，需要自己添加硬件配置，重启服务。

7.9.7 DCS 的 Memcached 过期数据清除策略是什么?

DCS的Memcached作为缓存产品是允许用户根据业务需求设置在其中存放数据的过期时间的。例如在执行**add**操作的时候可以设置**expire**过期时间。

DCS的Memcached默认策略为不逐出（noeviction）。当前版本支持在配置运行参数中修改逐出策略。

具体的6种数据逐出策略与Redis逐出策略一致，具体请参考**Redis默认的数据逐出策略是什么?**。

7.9.8 创建 Memcached 实例时如何选择可用区?

简单来说，只要是在同一区域（Region）内，选择任意一个可用区内DCS的Memcached都没有功能上的本质区别。

一般来讲，同一可用区比跨可用区的网络时延更有优势，但是跨可用区从容灾的角度比同可用区更有优势。当应用内部需要更低的网络时延，可以将应用实例组部署在同一个可用区中。

DCS的Memcached目前已支持跨可用区部署，在管理控制台创建DCS的Memcached时进行配置。只要与您的ECS在同一个区域（Region）内，无论选择创建哪个可用区的Memcached都可以实现与ECS正常的连通使用。若您希望获得更低的网络时延，请根据您ECS的可用区选择创建对应可用区的Memcached。

注意：由于资源库存因素，可能出现您在创建DCS的Memcached时仅有一个可用区资源供选择的情况，这不会影响您的正常使用。

A 文档修订记录

表 A-1 文档修订记录

发布日期	修订记录
2022-04-12	第一次正式发布。